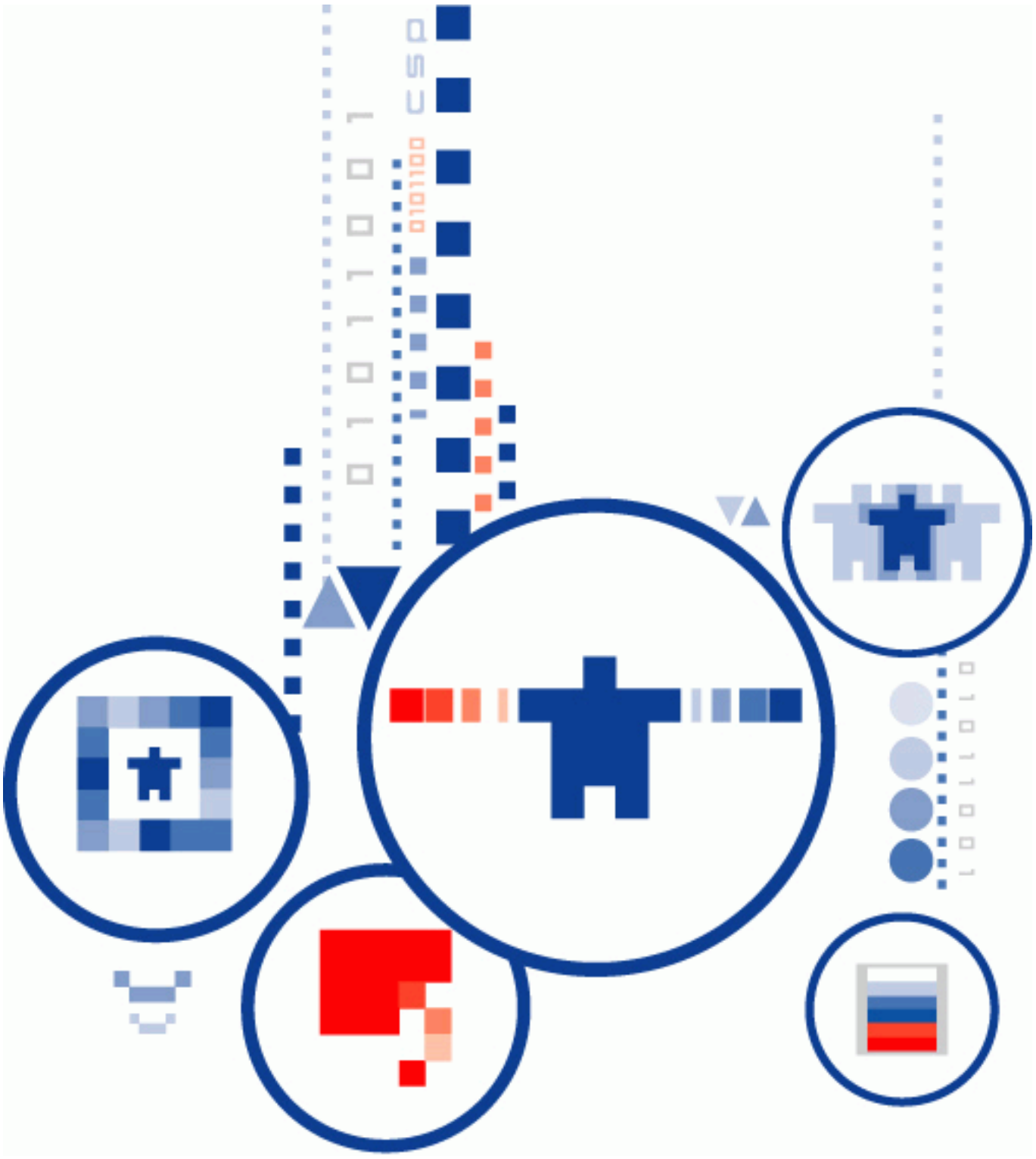


Сервер Электронной Подписи «КриптоПро DSS»

Компонент ПАКМ «КриптоПро HSM»

Руководство разработчика

ИСПОЛЬЗУЕМЫЕ СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ

CAdES	—	Расширенная версия стандарта электронной подписи CMS (CMS Advanced Electronic Signatures)
CMIS	—	Сервисы взаимодействия при управлении контентом (Content Management Interoperability Services)
CRL	—	Список отзыва сертификатов (Certificate Revocation List)
CSP	—	Криптопровайдер (Cryptographic Service Provider)
HSM	—	Аппаратный модуль системы безопасности (Hardware security module)
HOTP	—	алгоритм защищенной аутентификации с использованием одноразового пароля. (HMAC-Based One-Time Password Algorithm)
IIS	—	Набор серверов от компании Microsoft (Internet Information Services)
OATH	—	Набор алгоритмов аутентификации с использованием одноразовых паролей
OAuth	—	Открытый протокол авторизации, который позволяет предоставить третьей стороне ограниченный доступ к защищённым ресурсам пользователя без необходимости передавать ей (третьей стороне) логин и пароль (Open Authorization)
OCSP	—	Протокол получения статуса сертификата в реальном времени (Online Certificate Status Protocol)
OTP	—	Пароль, действительный только для одного сеанса аутентификации (One-Time Password)
REST	—	Архитектурный стиль построения распределенного приложения (Representational State Transfer)
RFC	—	Рекомендация Internet Engineering Task Force (Request for Comments)
SAML	—	Язык разметки декларации безопасности, язык разметки, основанный на языке XML (Security Assertion Markup Language)
SOAP	—	Простой протокол доступа к объектам (Simple Object Access Protocol)
SSL	—	Протокол защиты сокетов (Secure Sockets Layer)
TLS	—	Протокол защиты транспортного уровня (Transport Layer Security)
TOTP	—	OATH-алгоритм создания одноразовых паролей для защищенной аутентификации, генерирующий пароль на основе времени. (Time-based One Time Password Algorithm)
URL	—	Единый указатель ресурсов (Uniform Resource Locator)
МЭ	—	Межсетевой экран
АРМ	—	Автоматизированное рабочее место
БД	—	База данных
ИС	—	Информационная система
СУБД	—	Система управления базой данных
ОС	—	Операционная система
ПО	—	Программное обеспечение
СКЗИ	—	Средство криптографической защиты информации
СЭП	—	Сервер электронной подписи
ЭП	—	Электронная подпись

ПАК	—	Программно-аппаратный комплекс
ПАКМ	—	Программно-аппаратный криптографический модуль
УЦ	—	Удостоверяющий Центр

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Электронная подпись	—	информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию.
Сертификат открытого ключа	—	электронный или бумажный документ, содержащий открытый ключ, информацию о владельце ключа, области применения ключа, подписанный выдавшим его Удостоверяющим центром и подтверждающий принадлежность открытого ключа владельцу.
Квалифицированный сертификат открытого ключа (квалифицированный сертификат)	—	сертификат открытого ключа, выданный аккредитованным удостоверяющим центром или доверенным лицом аккредитованного удостоверяющего центра либо федеральным органом исполнительной власти, уполномоченным в сфере использования электронной подписи.
Владелец сертификата открытого ключа	—	лицо, которому в установленном Федеральным законом (№63-ФЗ от 06.04.2011 г. «Об электронной подписи») порядке выдан сертификат открытого ключа.
Закрытый ключ	—	уникальная последовательность символов, предназначенная для шифрования.
Ключ электронной подписи	—	уникальная последовательность символов, предназначенная для создания электронной подписи
Ключ проверки электронной подписи	—	уникальная последовательность символов, однозначно связанная с ключом электронной подписи и предназначенная для проверки подлинности электронной подписи.
Удостоверяющий центр	—	юридическое лицо или индивидуальный предприниматель, осуществляющие функции по созданию и выдаче сертификатов открытых ключей, а также иные функции, предусмотренные Федеральным законом №63-ФЗ от 06.04.2011 г. «Об электронной подписи».
Средства электронной подписи	—	шифровальные (криптографические) средства, используемые для реализации хотя бы одной из следующих функций - создание электронной подписи, проверка электронной подписи, создание закрытого и открытого ключей.

Аннотация

Настоящий документ содержит описание программного интерфейса сервера электронной подписи «КриптоПро DSS» (СЭП «КриптоПро DSS»), предназначенного для централизованного, защищенного хранения закрытых ключей пользователей, а также для удаленного выполнения операций по созданию электронной подписи в интересах пользователей при взаимодействии с КриптоПро HSM.

Документ предназначен для разработчиков как руководство по использованию программного интерфейса СЭП «КриптоПро DSS».

Информация о разработчике СЭП «КриптоПро DSS»:

ООО «Крипто-Про»
127018, Москва, ул. Сущёвский вал, 18
Телефон: (495) 995 4820
Факс: (495) 995 4820
Uhttp://www.CryptoPro.ru
E-mail: info@CryptoPro.ru

Содержание

1. Описание интерфейса HTTP API для работы с СЭП «КриптоПро DSS»

Процесс создания электронной подписи файла через клиентское приложение состоит из пяти этапов:

1. Отправка документа на сервер электронной подписи;
2. Аутентификация пользователя на сервере электронной подписи;
3. Выбор сертификата, выбор типа подписи, ввод pin-кода (если нужно);
4. Отправка подписанного документа обратно клиентскому приложению;
5. Получение клиентским приложением подписанного документа и дальнейшая его обработка.

Этапы 2, 3 и 4 выполняются на стороне КриптоПро DSS. Соответственно, клиентское приложение должно реализовать отправку документа и получение подписанного документа (этапы 1 и 5).

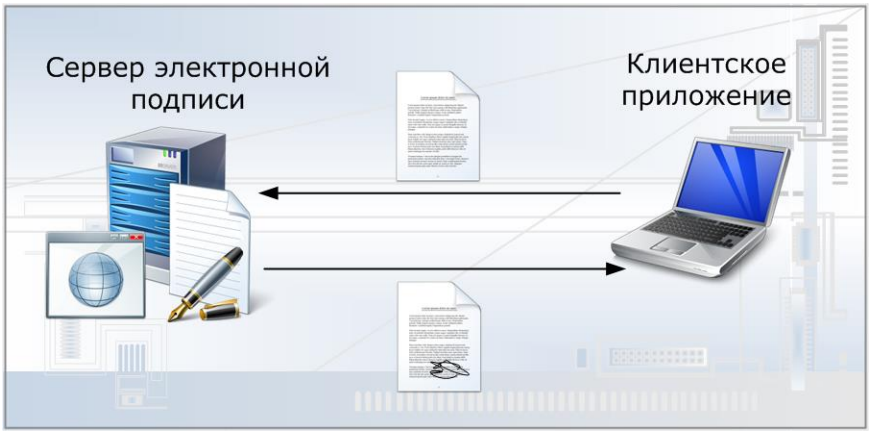


Рисунок 1. Создание электронной подписи файла через клиентское приложение

1.1. Отправка документа

Отправка документа на сервер электронной подписи должна осуществляться по протоколу **http(s)** и использоваться должен метод **POST**. Если сервер электронной подписи располагается по адресу http://localhost/frontend, то отправлять документ нужно по адресу http://localhost/frontend/uploadfile или http://localhost/frontend/SignatureWizard/uploadfile.

Обязательно должны быть указаны следующие параметры:

- filecontent – содержимое файла в формате BASE64,
- filename – имя файла,
- backpost – URL-адрес, на который следует отправить подписанный документ.

Дополнительно можно указать параметр:

- **signaturetype** – тип подписи. Формат параметра:
- type:$param_1$=$value_1$;…$param_N$=$value_N$, где type – формат подписи, $param_i$ – дополнительный параметр подписи (см. Таблица 1)
- **documentInfo** - дополнительная информация о документе (см. п. 4.5).
- С помощью данного параметра можно передать форматированный текст, который будет отображён в Веб-интерфейсе подписи (на странице подтверждения) и добавлен в отправленное пользователю сообщение с одноразовым паролем.
- Перед отправкой значение данного параметра должно быть закодировано в формате **BASE64**.

Так как клиентское приложение является web-клиентом, то для отправки файла следует использовать HTML-форму следующего вида:

```
<form action="http://localhost/frontend/uploadfile"  method="post" >
    <input type="hidden" id="filecontent" name="filecontent"
        value="A45B23DF5N21544RN…" />
    <input type="hidden" id="filename" name="filename"
        value="filetosign.txt" />
    <input type="hidden" name="backpost"
        value="http://localhost/PhpClient/result.php" />
    <input type="hidden" id="signaturetype" name="signaturetype"
        value="GOST3410:hash=true" />
    <input type="hidden" id="documentInfo" name="documentInfo" value="A45B2…"
    <input type="submit" value="Подписать файл" />
</form>
```

1.2. Получение подписанного документа

После того, как электронная подпись будет добавлена к документу, сервер электронной подписи отправит подписанный документ в формате **BASE64** на адрес, указанный в параметре **backpost**. При этом в POST запросе будет всего 2 параметра:

- **filename** – название документа,
- **filecontent** – содержимое подписанного документа в формате **BASE64**.

1.3. Отправка пакета документов

Отправка пакета документов на сервер электронной подписи должна осуществляться по протоколу **http(s)** и использоваться должен метод **POST**. Если сервер электронной подписи располагается по адресу http://localhost/frontend, то отправлять документ нужно по адресу http://localhost/frontend/SignatureWizard/UploadFiles.

Параметры запроса совпадают с параметрами запроса на подпись одного документа (см. пункт 1.1. Однако число параметров **documents** и **documentNames** должно быть равно числу документов в пакете.

Так как клиентское приложение является web-клиентом, то для отправки файла следует использовать HTML-форму следующего вида:

```
<form action="http://localhost/Frontend/SignatureWizard/UploadFiles"
method="post" >
    <input type="hidden" id="filecontent1" name="documents"
        value="A45B23DF5N21544RN…" />
    <input type="hidden" id="filename1" name="documentNames"
        value="filetosign1.txt" />
    <input type="hidden" id="filecontent2" name="documents"
        value="C56BE3DF5NRSS3245…" />
    <input type="hidden" id="filename2" name="documentNames"
        value="filetosign2.txt" />
    <input type="hidden" name="backpost"
        value="http://localhost/PhpClient/result.php" />
    <input type="hidden" id="signaturetype" name="signaturetype"
        value="GOST3410:hash=true" />
    <input type="hidden" id="documentInfo" name="documentInfo" value="A45B2…"
    <input type="submit" value="Подписать файл" />
</form>
```

1.4. Получение подписанного пакета документов

После того как все документы, входящие в пакет, будут подписаны, сервер электронной подписи отправит результат операции на адрес, указанный в параметре **backpost**. При этом в POST запросе будет содержаться следующие параметры:

- **result0, result1, …, resultn**– результат подписи n-го документа в формате **BASE64**,
- **error0, error1, …, errorn** – ошибки при подписи n-го документа.

Например, если пакет документов состоит из двух документов, то ответ сервера будет аналогичен POST-запросу, формируемому формой следующего вида:

```
<form action="http://<backpost>"  method="post" >
    <input type="hidden" id="result0" name="result0"
        value="A45B23DF5N21544RN…" />
    <input type="hidden" id="error0" name="error0"
        value="" />
    <input type="hidden" id="result1" name="result1"
        value="” />
    <input type="hidden" id="error1" name="error1"
        value="Ошибка при подписи - неверный формат" />
</form>
```

1.5. Отправка данных для формирования запроса на сертификат

Отправка документа на сервер электронной подписи должна осуществляться по протоколу **http(s)** и использоваться должен метод **POST**. Если сервер электронной подписи располагается по адресу http://localhost/frontend, то отправлять документ нужно по адресу http://localhost/frontend/uploadcertrequest.

Для отправки запроса на сертификат клиентское приложение должно сформировать строку, содержащую имя субъекта сертификата (Subject Name). Данная строка представляет собой список компонентов различительного имени субъекта разделённых запятой.

Пример:

CN=Иванов Иван Иванович, O=КРИПТО-ПРО Банк, OU=Финансовый отдел

Также клиентское приложение должно подготовить строку, содержащую объектные идентификаторы, которые должны попасть в расширение **ExtendedKeyUsage** сертификата. Данная строка представляет собой перечисление значений объектных идентификаторов через запятую. Данную строку перед отправкой необходимо закодировать в формате **BASE64**.

Полученную строку необходимо перевести в формат **BASE64**

Обязательно должны быть указаны следующие параметры:

- **reqcontent** – содержимое различительного имени субъекта в формате **BASE64**,
- **reqeku** – строка, содержащая объектные сертификаты расширения ExtendedKeyUsage в формате **BASE64**
- **backpost** – URL-адрес, на который следует отправить сформированный запрос на сертификат.
- Так как клиентское приложение является web-клиентом, то для отправки данных следует использовать HTML-форму следующего вида:

```
<form action="http://localhost/frontend/uploadcertrequest"  method="post" >
    <input type="hidden" id="reqcontent" name="reqcontent"
        value="A45B23DF5N21544RN…" />
<input type="hidden" id="reqeku" name="reqeku"
        value="A45B23DF5N21544RN…" />
```

```
    <input type="hidden" name="backpost"
        value="http://localhost/HttpApiDemo/ReceiveCertRequest.aspx" />
    <input type="submit" value="Сформировать запрос" />
</form>
```

1.6. Получение сформированного запроса на сертификат

После того как запрос на сертификат будет сформирован, сервер электронной подписи отправит его в формате **BASE64** на адрес, указанный в параметре **backpost**. При этом в POST запросе будет всего 1 параметр:

- **reqcontent** – содержимое запроса на сертификат в формате **BASE64**.

1.7. Установка сертификата

Отправка устанавливаемого сертификата на сервер электронной подписи должна осуществляться по протоколу **http(s)** и использоваться должен метод **POST**. Если сервер электронной подписи располагается по адресу http://localhost/frontend, то отправлять документ нужно по адресу http://localhost/frontend/uploadcertificate.

Обязательно должны быть указаны следующие параметры:

- **certcontent** – содержимое сертификата в формате **BASE64**,
- **backpost** – URL-адрес, на который следует вернуться после установки сертификата.
- Так как клиентское приложение является web-клиентом, то для отправки данных следует использовать HTML-форму следующего вида:

```
<form action="http://localhost/frontend/uploadcertificate"  method="post" >
    <input type="hidden" id="certcontent" name="certcontent"
        value="A45B23DF5N21544RN…" />
    <input type="hidden" name="backpost"
        value="http://localhost/HttpApiDemo/ReceiveCertRequest.aspx" />
    <input type="submit" value="Установить сертификат" />
</form>
```

После установки сертификата пользователь будет перенаправлен на адрес, указанный в параметре **backpost**. При успешной установке сертификата в QueryString будет содержаться параметр **result** со значением **ok**.

1.8. Зашифрование документа

Отправка документа для зашифрования на сервер электронной подписи должна осуществляться по протоколу **http(s)** и использоваться должен метод **POST**. Если сервер электронной подписи располагается по адресу http://localhost/frontend, то отправлять документ нужно по адресу http://localhost/Frontend/EncryptionWizard/UploadFile.

Обязательно должны быть указаны следующие параметры:

- **filecontent** – содержимое документа в формате **BASE64**,
- **filename** – имя файла,
- **backpost** – URL-адрес, на который следует отправить подписанный документ.

Дополнительно можно указать параметр:

- **encryptionType** – тип шифрования (см. Таблица 3).

Так как клиентское приложение является web-клиентом, то для отправки данных следует использовать HTML-форму следующего вида:

```
<form action="http://localhost/frontend/uploadencryptfile " method="post" >
  <input type="hidden" id="filecontent" name="filecontent" value="Q04…" />
  <input type="hidden" id="filename" name="filename" value="test.txt" />
  <input type="hidden" id="encryptionType" name="encryptionType" value="cms" />
```

```
  <input type="hidden" name="backpost" value="http://localhost/" />
</form>
```

После того, как документ будет зашифрован, сервер электронной подписи отправит подписанный документ в формате **BASE64** на адрес, указанный в параметре **backpost**. При этом в POST запросе будет всего 2 параметра:

- **filename** – название документа,
- **filecontent** – содержимое подписанного документа в формате **BASE64**.

1.9. Расшифрование документа

Отправка документа для расшифрования на сервер электронной подписи должна осуществляться по протоколу **http(s)** и использоваться должен метод **POST**. Если сервер электронной подписи располагается по адресу http://localhost/frontend, то отправлять документ нужно по адресу http://localhost/Frontend/DecryptionWizrd/UploadFile.

Обязательно должны быть указаны следующие параметры:

- **filecontent** – содержимое зашифрованного документа в формате **BASE64**,
- **filename** – имя файла,
- **backpost** – URL-адрес, на который следует отправить подписанный документ.

Дополнительно можно указать параметр:

- **decryptionType** – тип шифрования (см. Таблица 3).

Так как клиентское приложение является web-клиентом, то для отправки данных следует использовать HTML-форму следующего вида:

```
<form action="http://localhost/frontend/uploadencryptfile " method="post" >
<input type="hidden" id="filecontent" name="filecontent" value="MII…" />
<input type="hidden" id="filename" name="filename" value="test.txt" />
<input type="hidden" id=" decryptionType" name=" decryptionType" value="cms" />
<input type="hidden" name="backpost" value="http://localhost/" />
</form>
```

После того, как документ будет зашифрован, сервер электронной подписи отправит подписанный документ в формате **BASE64** на адрес, указанный в параметре **backpost**. При этом в POST запросе будет всего 2 параметра:

- **filename** – название документа,
- **filecontent** – содержимое подписанного документа в формате **BASE64**.

1.10. Сообщения об ошибках

В случае возникновения ошибок Веб-интерфейс пользователя отправит **HTTP POST** запрос с параметром **error**, содержащим сообщение об ошибке, на адрес, указанный в параметре **backpost**.

1.11. Пример клиентского приложения, реализованного на PHP

Данный пример следует использовать только в ознакомительных целях. Для простоты восприятия был опущен ряд проверок безопасности. Кроме того, данный пример рассчитан на то, что файлы хранятся в незащищённой директории.

1.11.1. Загрузка файла

В данном примере первоначальная загрузка файла на сервер реализована в файле **upload.php**.

```
<?php
header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
header("Expires: Sat, 26 Jul 1997 05:00:00 GMT"); // Date in the past
?>
<html>
<head>
    <title>Внешний PHP-клиент для сервера электронной подписи - загрузка файла
для подписи</title>
</head>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    Выберите файл для подписи
    <input type="file" name="file2sign" />
    <input type="submit" name="submit" value="Отправить на подпись" />
</form>

<?php
//предварительно загружаем файл для подписи на сервер в папку uploadedfiles
if ($_SERVER['REQUEST_METHOD'] == 'POST')
{
    $filesize = $_FILES['file2sign']['size'];
    if ($filesize == 0)
    {
        echo 'Файл не загружен. Выберите файл для загрузки';
    }
    else
    {
        $upload_path = './uploadedfiles/';
        $filename = $_FILES['file2sign']['name'];
        move_uploaded_file($_FILES['file2sign']['tmp_name'], $upload_path .
$filename);
        session_start();
        //в сессию запоминаем полное имя файла
        $_SESSION['tmp_filepath'] = $upload_path . $filename;
        header("Location: send2sign.php");
    }
}
?>
</body>
</html>
```

1.11.2. Отправка файла на сервер электронной подписи

В данном примере отправка файла на сервер электронной подписи реализована в файле **send2sign.php**.

```
<?php
header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
header("Expires: Sat, 26 Jul 1997 05:00:00 GMT"); // Date in the past
?>
<html>
<head>
    <title>Внешний PHP-клиент для сервера подписи - загрузка файла для
подписи</title>
</head>
<body>
```

```
<?php

session_start();
if (isset($_SESSION['tmp_filepath']))
{
    echo '<span style="color:green">Файл успешно загружен</span>';

    $filepath = $_SESSION['tmp_filepath'];
    $filename = basename($filepath);

    //считываем файл по пути, который мы получили из сессии
    $binary = fread(fopen($filepath, "r"), filesize($filepath));

    //кодируем его в base64 для сохранения в скрытом поле filecontent
    $base64content = base64_encode($binary);
}
?>

<form action="http://localhost/DSS.Web.Frontend/uploadfile"  method="post" >
    <input type="hidden" id="filecontent" name="filecontent" value="<?php echo
$base64content ?>" />
    <input type="hidden" id="filename" name="filename" value="<?php echo
$filename ?>" />
    <input type="hidden" name="backpost"
value="http://localhost/PhpClient/result.php" />
       <input type="hidden" id="signaturetype" name="signaturetype"
value="gost3410:hash=true" />
    <input type="submit" value="Подписать файл!" />
</form>
```

1.11.3. Получение подписанного документа

В данном примере получение подписанного документа реализовано в файле **result.php**.

```
<?php
header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
header("Expires: Sat, 26 Jul 1997 05:00:00 GMT"); // Date in the past
?>
<html>
<head>
    <title>Внешний PHP-клиент для сервера подписи - результат</title>
</head>
<body>

<?php

if ($_SERVER['REQUEST_METHOD'] == 'POST')
{
    //получаем данные из POST запроса
    $filename = $_POST['filename'];
    $filecontent = $_POST['filecontent'];

    //декодируем содержимое файла
    $decoded = base64_decode($filecontent);

    //сохраняем его на диске
    file_put_contents('./signedfiles/' . $filename, $decoded);

    $realpath = realpath('./signedfiles/');

    echo sprintf("Подписанный файл '%s' успешно сохранён в директории '%s'",
$filename, $realpath);
```

```
    echo '<br><br>';
    echo '<a href="upload.php">Подписать ещё файл</a>';

}else {
    echo 'неверный http метод';
}
?>
```

2. Описание интерфейса SOAP для работы с сервером электронной подписи «КриптоПро DSS»

Описание веб-сервиса доступно по адресу:

- http://<hostname>/SignServer/SignService.svc?wsdl

Также описание сервиса публикуется через конечную точку:

- http://<hostname>/SignServer/SignService.svc/mex

по протоколу WS-MetadataExchange.

Вызов методов Сервера Подписи возможен через несколько конечных точек:

6. http://<hostname>/SignServer/SignService.svc/token,
7. http://<hostname>/SignServer/SignService.svc/token/nosc,
8. http://<hostname>/SignServer/SignService.svc/token/transport,
9. http://<hostname>/SignServer/SignService.svc/token/transport/nosc,
10. http://<hostname>/SignServer/SignService.svc/federation,
11. http://<hostname>/SignServer/SignService.svc/federation/nosc,
12. http://<hostname>/SignServer/SignService.svc/federation/transport,
13. http://<hostname>/SignServer/SignService.svc/federation/transport/nosc,

Конечные точки (1-4) могут быть использованы, если вызывающее приложение уже получило маркер безопасности для аутентификации пользователя на Сервере Подписи. В случае вызова методов Сервера Подписи через конечные точки (5-8) получение маркера безопасности происходит прозрачно для пользователя по протоколу WS-Federation.

Если на Сервере Подписи настроено требование Подтверждения транзакций, то обращения к Серверу подписи возможны только через конечные точки (1-4).

Адреса конечных точек, заканчивающиеся на "nosc", не поддерживают работу в сессии. При каждом обращении к Серверу Подписи требуется аутентификация.

Адреса конечных точек, содержащие "transport", осуществляют защиту сообщения на транспортном уровне с помощью установления защищённого HTTPS соединения с сервером.

2.1. Функции для работы с подписью

SOAP-интерфейс для работы с сервером электронной подписи «КриптоПро DSS» предоставляет следующие функции для работы с подписью:

- SignDocument
- SignDocuments
- EnhanceSignature

2.1.1. Функция SignDocument

Функция **SignDocument** подписывает документ с помощью сертификата пользователя.

```
byte[] SignDocument(byte[] document, SignatureType signatureType, int
certificateID, string pinCode, IDictionary<SignatureParams, string>
signatureParams = null);
```

Параметры:

- **document** – документ для подписи в виде массива байт,
- **signatureType** – тип подписи SignatureType,
- **certificateID** – идентификатор сертификата, которым будет подписан документ,

- **pinCode** – пин-код для контейнера (если пин-код не требуется, то данный параметр должен быть пустым),
- **signatureParams** – словарь дополнительных параметров подписи (опциональный параметр). Типы дополнительных параметров указаны в перечислении SignatureParams.

Возвращаемое значение: подписанный документ в виде массива байт.

Типы подписи и дополнительные параметры описаны в Таблица 1.

Таблица 1 - Дополнительные параметры подписи

Тип подписи	Дополнительные параметры	Значения	Описание
XMLDSig Подпись документа в формате XMLDSig	XMLDSigType	XMLEnveloped	Вложенная XMLDSig подпись
		XMLEnveloping	Присоединённая XMLDSig подпись
		XMLTemplate	XMLDSig подпись по шаблону
GOST3410 Электронная подпись по ГОСТ Р 34.10 - 2001	Hash	False	Подпись данных
		True	Подпись значения хеш-функции ГОСТ Р 34.11 - 94
CAdES Подпись формата CAdES	CAdESType	BES	Подпись в формате CAdES-BES
		T	Подпись в формате CAdES-T
		XLT1	Подпись в формате CAdES-X Long Type 1
	IsDetached	True	Отделённая подпись
		False	Присоединённая подпись
	TSPAddress		Адрес TSP службы (используется только для формата XLT1)
	Hash	True	Подпись данных
		False	Подпись значения хеш-функции ГОСТ Р 34.11 - 94
PDF Подпись PDF документов	PDFFormat	CMS	Подпись PDF документов с использованием формата PKCS7
		CAdEST	Подпись PDF документов с использованием формата CAdES-T
		CAdES	Подпись PDF документов с использованием формата CAdES-X Long Type 1

Тип подписи	Дополнительные параметры	Значения	Описание
	TSPAddress		Адрес TSP службы (используется только для формата CAdES-T, CAdES-X Long Type 1)
	PDFReason		Цель подписания документа
	PDFLocation		Место подписания документа
	PdfSignatureAppearance		Строковое представление шаблона видимой (отображаемой) PDF-подписи. Подробнее см. п. 2.7.14
	PdfSignatureTemplateId		Идентификатор шаблона видимой (отображаемой) PDF-подписи. Подробнее см. п. 2.7.14
MSOffice Подпись документов MS Word и Excel	Отсутствуют		
CMS Подпись формата CAdES-BES	IsDetached	True	Отделённая подпись
		False	Присоединённая подпись
	Hash	True	Подпись данных
		False	Подпись значения хеш-функции ГОСТ Р 34.11 - 94

При подписи документа возможно использование сертификата по умолчанию (см. разделы 2.3.4, 2.6.3). Для использования сертификата по умолчанию в качестве идентификатора сертификата при вызове метода SignDocument необходимо передать 0. В таком случае сервис попытается найти среди сертификатов пользователя сертификат по умолчанию (который должен быть установлен до этого с помощью функции SetCertificateProperty) и в случае успеха использует его.

2.1.2. Функция SignDocuments

Функция **SignDocuments** подписывает пакет документов с помощью сертификата пользователя.

```
DSSSignDocumentResponse SignDocuments(List<byte[]> documents, SignatureType
signatureType, int certificateID, string pinCode, IDictionary<SignatureParams,
string> signatureParams = null);
```

Параметры:

- **documents** – массив документов для подписи,
- **signatureType** – тип подписи SignatureType,

- **certificateID** – идентификатор сертификата, которым будет подписан документ,
- **pinCode** – пин-код для контейнера (если пин-код не требуется, то данный параметр должен быть пустым),
- **signatureParams** – словарь дополнительных параметров подписи (опциональный параметр). Типы дополнительных параметров указаны в перечислении SignatureParams.
-

Возвращаемое значение: объект класса DSSSignDocumentResponse.

Все подписываемые документы должны быть одного формата (соответствующего параметру signatureType).

При подписи документа возможно использование сертификата по умолчанию (см. разделы 2.3.4, 2.6.3). Для использования сертификата по умолчанию в качестве идентификатора сертификата при вызове метода SignDocuments необходимо передать 0. В таком случае сервис попытается найти среди сертификатов пользователя сертификат по умолчанию (который должен быть установлен до этого с помощью функции SetCertificateProperty) и в случае успеха использует его.

2.1.3. Функция EnhanceSignature

Функция **EnhanceSignature** усовершенствует подписи формата CMS (CAdES-BES) до CAdES-T и CAdES-X Long Type 1.

```
byte[] EnhanceSignature(
            byte[] document,
            SignatureType signatureType,
            Dictionary<SignatureParams, string> signatureParams = null);
```

Параметры:

- **documents** – массив документов для подписи,
- **signatureType** – тип подписи SignatureType,
- **signatureParams** – словарь дополнительных параметров подписи (опциональный параметр). Типы дополнительных параметров указаны в перечислении SignatureParams.

Возвращаемое значение: подписанный документ в виде массива байт.

В параметре signatureType может быть переданы значения CAdES или CMS (значения эквивалентны).

В словаре дополнительных параметров signatureParams могут быть переданы следующие значения:

Таблица 2 - Дополнительные параметры доусовершенствования подписи

Параметр	Значение	Описание
TSPAddress	Адрес службы штампов времени.	Обязательный
CADESType	Тип подписи CAdES, до которой требуется доусовершенствовать подпись. Возможные значения: T, XLT1	Обязательный
IsDetached	Если подпись создавалась как откреплённая, то при усовершенствовании необходимо выставить данный флаг в значение true. При усовершенствовании откреплённой подпись передавать оригинальный документ не нужно.	Опциональный.

2.2. Функции шифрования

SOAP-интерфейс для работы с сервером электронной подписи «КриптоПро DSS» предоставляет следующие функции шифрования:

- EncryptDocument.
- DecryptDocument.
-

2.2.1. Функция EncryptDocument

Функция EncryptDocument зашифровывает данные на сертификатах получателей.

```
byte[] EncryptDocument(byte[] document, EncryptionType encryptionType,
IList<byte[]> recipients, IDictionary<string, string> additionalParams);
```

Параметры:

- **document** – шифруемые данные в виде массива байт,
- **encryptionType** – тип шифрования,
- **recipients** – сертификаты получателей,
- **additionalParams** – словарь дополнительных параметров шифрования (опциональный параметр).

Возвращаемое значение: зашифрованный документ в виде массива байт. Типы шифрования и дополнительные параметры описаны в Таблица 3.

Таблица 3 - Дополнительные параметры функции шифрования

Тип шифрования	Дополнительные параметры	Значения	Описание
CMS Шифрование документа в формате CMS	SubjectIdentifierType	SubjectKeyIdentifier	Тип идентификатора субъекта Получатель идентифицируется по хэш-значению от открытого ключа сертификата
		IssuerAndSerialNumber	Тип идентификатора субъекта Получатель идентифицируется по серийному номеру сертификата и различительному имени издателя сертификата (по умолчанию)
	IncludeCertificates	True	Включать сертификаты получателей в сообщение.
		False	Не включать сертификаты получателей в сообщение (по умолчанию)

Если определённый параметр отсутствует (не передавался) в словаре дополнительных параметров, то будет использовано значение по умолчанию.

2.2.2. Функция DecryptDocument

Функция DecryptDocument расшифровывает данные, зашифрованные с использованием сертификата пользователя.

```
DSSDecryptDocumentResponse DecryptDocument(byte[] encryptedDocument,
EncryptionType encryptionType, int certificateID, string pinCode);
```

Параметры:

- **encryptedDocument** – зашифрованные данные в виде массива байт,
- **encryptionType** – тип шифрования,
- **certificateID** – идентификатор сертификата пользователя, используемого для расшифрования документа,
- **pinCode** – пин-код для доступа к закрытого ключа (если пин-код не требуется, то данный параметр должен быть пустым).

Возвращаемое значение: объект класса DSSDecryptDocumentResponse.

Типы шифрования приведены в Таблица 3.

Если заранее не известен сертификат, используемый для расшифровки, то на вход функции DecryptDocument в качестве значения параметра certificateID необходимо передать 0. В этом случае произойдёт только разбор переданного документа и извлечение из него информации о сертификатах, которые могут быть использованы для завершения операции. Идентификаторы данных сертификатов будут возвращены в поле CertificateIDs класса DSSDecryptDocumentResponse.

Если идентификатор известен (заранее или после выполнения разбора зашифрованного документа), его необходимо передать в качестве значения параметра certificateID, указав также пин-код для доступа к соответствующему закрытому ключу.

После расшифровки данные будут возвращены в поле Result класса DSSDecryptDocumentResponse.

2.3. Функции для работы с сертификатами

SOAP-интерфейс для работы с сервером электронной подписи «КриптоПро DSS» предоставляет следующие функции для работы с сертификатами:

- GetCertificates,
- GetCertificate
- GetCertificateContent,
- SetCertificateProperty,
- InstallCertificateBase64,
- InstallCertificateDer,
- DeleteCertificate.

2.3.1. Функция GetCertificates

Функция **GetCertificates** получает список всех сертификатов пользователя.

```
IList<DSSCertificate> GetCertificates()
```

Функция **GetCertificates** не имеет параметров.

Возвращаемое значение: список сертификатов DSSCertificate.

2.3.2. Функция GetCertificate

Функция **GetCertificate** получает сертификат пользователя по идентификатору сертификата.

```
DSSCertificate GetCertificate(int id)
```

Параметры:

- **id** – идентификатор сертификата.

Возвращаемое значение: список сертификатов DSSCertificate.

2.3.3. Функция GetCertificateContent

Функция **GetCertificateContent** получает сертификат пользователя по идентификатору сертификата в определённом формате.

```
string GetCertificateContent(int id, DSSCertificateFormat format)
```

Параметры:

- **id** – идентификатор сертификата;
- **format** – значение формата сертификата из перечисления DSSCertificateFormat.

Возвращаемое значение: сертификат в формате, заданном параметром **format**.

2.3.4. Функция SetCertificateProperty

Функция **SetCertificateProperty** позволяет изменить свойства сертификата:

- Отозвать сертификат
- Приостановить сертификат
- Восстановить сертификат
- Сменить PIN-код
- Установить сертификат по умолчанию

```
void SetCertificateProperty(int id, CertificateProperty property)
```

Параметры:

- **id** - идентификатор сертификата;
- **property** – устанавливаемое свойство сертификата, имеет тип CertificateProperty.

Возвращаемое значение: функция не имеет возвращаемого значения.

2.3.5. Функция InstallCertificateBase64

Функция **InstallCertificateBase64** устанавливает сертификат в БД СЭП.

```
DSSCertificate InstallCertificateBase64(string base64Certificate, string
pinCode)
```

Параметры:

- **base64Certificate** – сертификат в формате BASE64,
- **pinCode** – ПИН-код для контейнера (если ПИН-код не требуется, то данный параметр должен быть пустым).

Возвращаемое значение: установленный сертификат DSSCertificate.

2.3.6. Функция InstallCertificateDer

Функция **InstallCertificateDer** устанавливает сертификат в БД СЭП.

```
DSSCertificate InstallCertificateDer(byte[] binaryCertificate, string pinCode)
```

Параметры:

- **binaryCertificate** – сертификат в бинарном виде,
- **pinCode** – ПИН-код для контейнера (если ПИН-код не требуется, то данный параметр должен быть пустым).

Возвращаемое значение: установленный сертификат DSSCertificate.

2.3.7. Функция DeleteCertificate

Функция **DeleteCertificate** удаляет сертификат из БД СЭП.

```
void DeleteCertificate(int id)
```

Параметры:

- **id** – идентификатор удаляемого сертификата.

Возвращаемое значение: функция не имеет возвращаемого значения.

Если в БД СЭП нет запроса, связанного с данным сертификатом, то закрытый ключ будет также удалён.

2.4. Функции для работы с запросами на сертификат

SOAP-интерфейс для работы с сервером электронной подписи «КриптоПро DSS» предоставляет следующие функции для работы с запросами на сертификат:

- GetRequests
- GetRequest
- DeleteRequest
- CreateRequest
- CreateRequestEx
- SetRequestStatus
- GetRevokeRequests

2.4.1. Функция GetRequests

Функция **GetRequests** возвращает список всех запросов на сертификат для данного пользователя.

```
IList<DSSCertRequest> GetRequests()
```

Функция **GetRequests** не имеет параметров.

Возвращаемое значение: список запросов на сертификат DSSCertRequest.

2.4.2. Функция GetRequest

Функция **GetRequest** возвращает запрос на сертификат по идентификатору для данного пользователя.

```
DSSCertRequest GetRequest(int id)
```

Параметры:

- **id** – идентификатор запроса на сертификат.

Возвращаемое значение: запрос на сертификат DSSCertRequest.

2.4.3. Функция DeleteRequest

Функция **DeleteRequest** удаляет запрос на сертификат из базы данных сервера подписи.

```
void DeleteRequest(int id)
```

Параметры:

- **id** – идентификатор запроса на сертификат. Если в качестве значения передан 0, то будут удалены все запросы на сертификат данного пользователя.

Возвращаемое значение: функция DeleteRequest не имеет возвращаемого значения.

Если в БД СЭП нет сертификата, выпущенного по данному запросу, то закрытый ключ, соответствующий удаляемому запросу, будет также удалён.

2.4.4. Функция CreateRequest

Функция **CreateRequest** создаёт запрос на сертификат.

```
int CreateRequest(string dName, string eKU, string pinCode, int caId)
```

Параметры:

- **dName** – различительное имя субъекта,
- **eKU** – список OID улучшенного использования ключа, разделённых запятой.
- **pinCode** – ПИН-код на контейнер (если данный параметр не задан, то ПИН-код на контейнер установлен не будет).
- **caId** – идентификатор удостоверяющего центра, к которому будет направлен запрос на сертификат.

Возвращаемое значение: идентификатор запроса на сертификат.

Для получения идентификатора удостоверяющего центра необходимо вызвать метод GetDSSPolicy. В полученном объекте DSSPolicy через свойство CAPolicy пользователь может получить список зарегистрированных на Сервере Подписи Удостоверяющих Центров.

2.4.5. Функция CreateRequestEx

Функция **CreateRequest** создаёт запрос на сертификат.

```
int CreateRequestEx(string dName, string pinCode, int caId,
Dictionary<RequestParams, string> requestParams)
```

Параметры:

- **dName** – различительное имя субъекта,
- **pinCode** – ПИН-код на контейнер (если данный параметр не задан, то ПИН-код на контейнер установлен не будет).
- **caId** – идентификатор УЦ, к которому будет направлен запрос на сертификат.
- **requestParams** – словарь дополнительных параметров, используемых при создании запроса на сертификат. Типы поддерживаемых параметров перечислены в RequestParams.

Возвращаемое значение: идентификатор запроса на сертификат в БД СЭП.

Для получения идентификатора УЦ необходимо вызвать метод GetDSSPolicy. В полученном объекте DSSPolicy через свойство CAPolicy пользователь может получить список зарегистрированных на Сервере Подписи Удостоверяющих Центров.

2.4.6. Функция GetRequestContent

Функция **GetRequestContent** получает запрос на сертификат пользователя по идентификатору запроса в определённом формате.

```
string GetRequestContent(string id, CARequestTypeEnum requestType, int caId,
DSSCertificateFormat format)
```

Параметры:

- **id** – идентификатор запроса,
- **requestType** – тип запроса из перечисления CARequestTypeEnum,
- **caId** – идентификатор удостоверяющего центра,
- **format** – значение формата запроса из перечисления DSSCertificateFormat.

Возвращаемое значение: сертификат в формате, заданном параметром **format**.

Для получения идентификатора УЦ необходимо вызвать метод GetDSSPolicy. В полученном объекте DSSPolicy через свойство CAPolicy пользователь может получить список зарегистрированных на Сервере Подписи Удостоверяющих Центров.

2.4.7. Функция GetRevokeRequests

Функция **GetRevokeRequests** возвращает список запросов на отзыв, приостановление, восстановление для указанного сертификата.

```
IList<DSSRevRequest> GetRevokeRequests(int certId)
```

Параметры:

- certId – идентификатор сертификата для которого требуется получить список запросов на отзыв.

Возвращаемое значение: список запросов на отзыв, приостановление, восстановление DSSRevRequest.

2.4.8. Функция SetRequestStatus

Функция **SetRequestStatus** устанавливает статус запроса на сертификат. Данная функция используется только администратором Сервера Подписи для принятия или отклонения запроса на сертификат, запросов на приостановление, восстановление, отзыв сертификата.

```
void SetRequestStatus(RequestStatus status)
```

Параметры:

- **status** – статус запроса RequestStatus.

Возвращаемое значение: функция не имеет возвращаемого значения.

2.5. Другие функции

SOAP-интерфейс для работы с сервером электронной подписи «КриптоПро DSS» предоставляет следующие вспомогательные функции:

- GetTransactionID,
- GetDSSPolicy,
- CreateTransactionToken.

2.5.1. Функция GetTransactionID

Функция **GetTransactionID** является устаревшей. Вместо неё следует использовать функцию **CreateTransactionToken** (см. п. 2.5.3).

Функция **GetTransactionID** возвращает уникальный идентификатор транзакции (используется для механизма подтверждения транзакций)

```
int GetTransactionID()
```

Функция **GetTransactionID** не имеет параметров.

Возвращаемое значение: идентификатор транзакции.

2.5.2. Функция GetDSSPolicy

Функция **GetDSSPolicy** возвращает политику Сервера Подписи, которая содержит настройки данного экземпляра.

```
DSSPolicy GetDSSPolicy()
```

Функция **GetDSSPolicy** не имеет параметров.

Возвращаемое значение: политика Сервера Подписи DSSPolicy.

2.5.3. Функция CreateTransactionToken

Функция **CreateTransactionToken** создаёт транзакцию на сервисе подписи (используется для механизма подтверждения операций см. п. 6.2).

```
string CreateTransactionToken(DSSActions operationCode, Dictionary<string, string> additionalParams, byte[] document)
```

Параметры:

- **operationCode** – идентификатор операции из перечисления DSSActions (кроме, Issue).
- **additionalParams** – дополнительные параметры операции. Набор параметров, соответствующий конкретной операции (см. п. 6.2).
- **document** – содержимое документа (только для операций подписи, улучшения подписи, расшифрования).

Возвращаемое значение: идентификатор транзакции.

2.6. Типы данных

2.6.1. DSSCertRequest

Класс описывает запрос на сертификат. Описание полей класса приведено в Таблица 4.

Таблица 4 - Описание полей класса DSSCertRequest

Поле	Тип	Описание
ID	int	Идентификатор запроса в БД СЭП
Base64Request	string	Запрос на сертификат в формате BASE64
CertificateAuthorityID	int	Идентификатор удостоверяющего центра в БД СЭП, к которому направлен запрос на сертификат
CARequestID	string	Идентификатор запроса на сертификат в удостоверяющем центре
CADisplayName	string	Отображаемое имя УЦ
DistName	string	Различительное имя субъекта
Subject	string	Общее имя субъекта
Status	DSSRequestStatusEnum	Статус запроса в УЦ
CertificateID	int	Идентификатор сертификата, с которым связан запрос.
RequestType	CARequestTypeEnum	Тип запроса к удостоверяющему центру.

2.6.2. DSSCertificate

Класс описывает сертификат пользователя. Описание полей класса приведено в Таблица 5.

Таблица 5 - Описание полей класса DSSCertificate

Поле	Тип	Описание
ID	int	Идентификатор сертификата в БД СЭП
DName	string	Различительное имя субъекта
CertificateBase64	string	Сертификат пользователя в формате BASE64
Status	CertificateStatus	Статус сертификата в БД СЭП.
IsDefault	bool	Свойство определяет, является ли данный сертификат сертификатом по умолчанию (значение True)
CertificateAuthorityID	int	Идентификатор обработчика УЦ.

2.6.3. CertificateProperty

Класс описывает свойство сертификата. Описание полей класса приведено в Таблица 6.

Таблица 6 - Описание полей класса CertificateProperty

Поле	Тип	Описание
PropertyType	CertificatePropertyType	Тип свойства сертификата.
Value	CertificateStatus bool PinInfo	Значение свойства сертификата. Может быть объектом одного из трёх типов в зависимости от значения поля PropertyType.

2.6.4. CertificateStatus

Класс описывает статус сертификата. Описание полей класса приведено в Таблица 7.

Таблица 7 - Описание полей класса CertificateStatus

Поле	Тип	Описание
Value	DSSCertificateStatusEnum	Статус сертификата

Поле	Тип	Описание
RevocationInfo	RevocationInfo	Дополнительная информация об отзыве сертификата (используется, если значение поля Value не равно ACTIVE)
PinCode	string	Пин-код для доступа к закрытому ключ сертификата, которым подписывается запрос на возобновление действия сертификата.
ActiveCertId	int	Идентификатор сертификата, которым подписывается запрос на возобновление сертификата.

2.6.5. RevocationInfo

Поля класса содержать дополнительную информацию об отзыве сертификата. Описание полей приведено в Таблица 8.

Таблица 8 - Описание полей класса RevocationInfo

Поле	Тип	Описание
RevocationReason	CertRevokeReasonEnum	Причину отзыва сертификата.
RevocationDate	DateTime	Дата отзыва.
RevocationComments	String	Комментарии к запросу на отзыв.
UnholdDate	DataTime	Дата, до которой следует приостановить действие сертификата.
RequestDate	DataTime	Дата подачи запроса.
ApproveDate	DateTime	Дата рассмотрения запроса.
UnholdAction	string	Действие, которое требуется выполнить после наступления даты приостановления. Для КриптоПро УЦ 1.5 поле должно иметь значение NULL. Для УЦ 2.0 должно содержать код причины отзыва (см. 2.7.5), кроме кода 6.
SerialNumber	string	Серийный номер отзываемого сертификата.

2.6.6. PinInfo

Класс описывает данные необходимые для изменения пин-кода для доступа к закрытому ключу сертификата. Описание полей класса приведено в Таблица 9.

Таблица 9 - Описание полей класса PinInfo

Поле	Тип	Описание
Pin	string	Текущий пин-код для доступа к закрытому ключу
NewPin	string	Новый пин-код для доступа к закрытому ключу

2.6.7. DSSPolicy

Класс описывает политику Сервера Подписи, которая содержит основные настройки экземпляра. Описание полей класса приведено в Таблица 10.

Таблица 10 - Описание полей класса DSSPolicy

Поле	Тип	Описание
CAPolicy	List<DSSCAPolicy>	Список удостоверяющих центров, настроенных на сервере подписи.
PinCodeMode	PinCodeMode	Режим установки пин-кода на контейнер закрытого ключа.
TspServices	List<TspService>	Список TSP служб, настроенных на сервере подписи.
TransactionConfirmation	bool	Использовать или нет подтверждение транзакций при подписи.
AllowedSignatureType	List<SignatureType>	Список типов подписи, зарегистрированных на сервере подписи. Значение SignatureType смотреть в Таблица 1.
ActionPolicy	List<DSSAction>	Политика подтверждения действий сервера подписи.

2.6.8. DSSCAPolicy

Класс описывает параметры удостоверяющего центра. Описание полей класса приведено в Таблица 11.

Таблица 11 - Описание полей класса DSSCAPolicy

Поле	Тип	Описание
ID	int	Идентификатор УЦ в БД СЭП
Name	string	Отображаемое имя УЦ

Поле	Тип	Описание
Active	bool	Доступен ли УЦ для создания запросов (фактически всегда равно true, так как сервер подписи отдаёт список только активных УЦ)
SNChangesEnable	bool	Разрешено ли редактировать шаблон запроса на сертификат (если false, то данные берутся из первого запроса на сертификат)
NamePolicy	List<SubjectName Component>	Шаблон имени. Содержит список компонентов различительного имени, их порядок и обязательность
EkuTemplates	Dictionary<string, List<string>>	Словарь шаблонов сертификатов в формате {"Наименование шаблона", {Список OID}}
AllowUserMode	bool	Режим работы обработчика удостоверяющего центра.
CAType	DSSCAType	Тип обработчика удостоверяющего центра.

2.6.9. DSSAction

Класс описывает свойства метода сервера подписи. Описание полей класса приведено в Таблица 12.

Таблица 12 - Описание полей класса DSSAction

Поле	Тип	Описание
Action	DSSActions	Тип действия
DisplayName	string	Отображаемое имя действия
Uri	string	Идентификатор действия (является идентификатором соответствующего утверждения)
MfaRequired	Bool	Требование двухфакторной аутентификации. True – требуется, False – не требуется.

2.6.10. DSSDecryptDocumentResponse

Класс описывает возвращаемое значение функции DecryptDocument. Описание полей класса приведено в Таблица 13.

Таблица 13 - Описание полей класса DSSDecryptDocumentResponse

Поле	Тип	Описание
Result	byte[]	Расшифрованный документ в виде массива байт

Поле	Тип	Описание
CertificatesIDs	List<int>	Список идентификаторов сертификатов, которые могут использоваться для расшифрования данного документа

2.6.11. TspService

Класс описывает TSP службу. Описание полей класса приведено в Таблица 14

Таблица 14 - Описание полей класса TspService

Поле	Тип	Описание
Name	string	Идентификатор службы
Title	string	Отображаемое имя службы
Url	string	Адрес службы

2.6.12. DSSRevRequest

Класс описывает запрос на отзыв. Описание полей класса приведено в Таблица 15.

Таблица 15 - Описание полей класса DSSRevRequest

Поле	Тип	Описание
Base64Request	string	Запрос на отзыв в формате Base64
ID	string	Идентификатор запроса
caID	int	Идентификатор обработчика удостоверяющего центра.
Status	DSSRequestStatusEnum	Статус запроса
revInfo	RevocationInfo	Информация о запросе

2.6.13. RequestStatus

Класс описывает статус запроса на отзыв, приостановление восстановление сертификата. Описание полей класса приведено в Таблица 16.

Таблица 16 - Описание полей класса RequestStatus

Поле	Тип	Описание
Status	DSSRequestStatusEnum	Статус запроса

Поле	Тип	Описание
Comment	string	Комментарий
ID	string	Идентификатор запроса
CertificateAuthorityID	int	Идентификатор обработчика удостоверяющего центра.
Type	CARequestTypeEnum	Тип запроса

2.6.14. SubjectNameComponent

Класс описывает компоненты имени субъекта. Описание полей приведено в Таблица 17.

Таблица 17 - Описание полей класса SubjectNameComponent

Поле	Тип	Описание
IsRequired	bool	Требовать обязательного заполнения компоненты имени.
Order	int	Порядковый номер в списке компонент.
OID	string	Объектный идентификатор компоненты имени.
Name	string	Понятное имя компоненты.
Value	string	Значение.
StringIdentifier	string	Строковый идентификатор компоненты.

2.6.15. DSSSignDocumentResponse

Класс описывает результат пакетной подписи документов. Описание полей класса приведено в Таблица 18.

Таблица 18 - Описание полей класса DSSSignDocumentResponse

Поле	Тип	Описание
Results	List<byte[]>	Список подписанных документов.
Errors	List<DSSFault>	Список ошибок, произошедших при подписи пакета документов.

2.7. Перечислимые типы данных

2.7.1. DSSRequestStatusEnum

Перечисление содержит возможные значения статуса запроса на сертификат. Описание элементов приведено в Таблица 19.

Таблица 19 - Перечисление DSSRequestStatusEnum

Код	Наименование	Описание
2	PENDING	Запрос на сертификат принят.
4	ACCEPTED	Запрос на сертификат одобрен. Выпущен сертификат по данному запросу.
8	REJECTED	Запрос на сертификат отклонён.
16	REGISTRATION	Запрос на регистрацию принят.

2.7.2. DSSCertificateStatusEnum

Перечисление содержит возможные значения статуса сертификата. Описание элементов приведено в Таблица 20.

Таблица 20 - Перечисление DSSCertificateStatusEnum

Код	Наименование	Описание
1	ACTIVE	Сертификат действителен.
2	REVOKED	Сертификат отозван.
3	HOLD	Действие сертификата приостановлено.
4	NOT_VALID	Сертификат не действителен.
6	OUT_OF_ORDER	Сертификат доступен только для просмотра. Действия «отзыв», «приостановка» невозможны.

2.7.3. DSSCertificateFormat

Перечисление содержит значения, определяющие формат сертификата, возвращаемого функциями GetCertificateContent и GetRequestContent. Описание элементов приведено в Таблица 21.

Таблица 21 - Перечисление DSSCertificateFormat

Код	Наименование	Описание
1	RAW	Содержимое в формате BASE64.

Код	Наименование	Описание
2	XML	Содержимое в формате XML.
4	FORMATTED	Содержимое, преобразованное с помощью XSL преобразования (задаётся в настройках обработчика удостоверяющего центра, через который сертификат был выпущен). Описание настройки и редактирования XSL преобразования смотреть в Руководстве администратора.

2.7.4. CertificatePropertyType

Перечисление содержит названия свойств сертификата, значение которых можно изменить с помощью функции SetCertificateProperty. Описание элементов приведено в Таблица 22.

Таблица 22 - Перечисление CertificatePropertyType

Наименование	Описание
Status	Статус сертификата. Поле Value объекта CertificateProperty должно быть типа CertificateStatus
Pin	Пин-код для доступа к закрытому ключу сертификата. Поле Value объекта CertificateProperty должно быть типа PinInfo.
IsDefault	Свойство показывает, является ли сертификат сертификатом по умолчанию для данного пользователя. Поле объекта CertificateProperty должно быть типа bool.

2.7.5. CertRevokeReasonEnum

Перечисление содержит причины отзыва сертификата. Описание элементов приведено в Таблица 23.

Таблица 23 - Перечисление CertRevokeReasonEnum

Код	Наименование	Описание
0	CRL_REASON_UNSPECIFIED	Причина неизвестна.
1	CRL_REASON_KEY_COMPROMISE	Компрометация ключей.
2	CRL_REASON_CA_COMPROMISE	Компрометация Центра Сертификации.
3	CRL_REASON_AFFILATION_CHANGED	Имя пользователя или другая информация в сертификате изменена, но нет причины полагать, что секретный ключ скомпрометирован.
4	CRL_REASON_SUPERSEDED	Сертификат заменен другим, но нет причины полагать, что секретный ключ скомпрометирован.
5	CRL_REASON_CESSATION_OF_OPERATION	Сертификат более не нужен для целей, которых он выдавался, но

Код	Наименование	Описание
		нет причины полагать, что секретный ключ скомпрометирован.
6	CRL_REASON_CERTIFICATE_HOLD	Действие сертификата приостановлено.
8	CRL_REASON_REMOVE_FROM_CRL	Код для возобновления действия сертификата.

2.7.6. CARequestTypeEnum

Перечисление содержит типы запросов к удостоверяющему центру. Описание элементов приведено в Таблица 24.

Таблица 24 - Перечисление CARequestTypeEnum

Наименование	Описание
Certificate	Запрос на выпуск, обновление сертификата.
RevokeRequest	Запрос на отзыв, восстановление, приостановление сертификата.

2.7.7. PinCodeMode

Перечисление содержит политику работы с пин-кодом. Описание элементов приведено в Таблица 25.

Таблица 25 - Перечисление PinCodeMode

Наименование	Описание
Required	Установка пин-кода обязательна.
Forbid	Установка пин-кода запрещена.
Allow	Установка пин-кода опциональна.

2.7.8. DSSCAType

Перечисление содержит типы обработчиков удостоверяющих центров. Описание элементов приведено в Таблица 26.

Таблица 26 - Перечисление DSSCAType

Наименование	Описание
CryptoProCA15Enroll	Обработчик КриптоПро УЦ 1.5
CryptoProCA20Enroll	Обработчик КриптоПро УЦ 2.0

Наименование	Описание
DSSOutOfBandEnroll	Обработчик стороннего удостоверяющего центра.

2.7.9. RequestParams

Перечисление содержит типы параметров, задаваемых при создании запроса на сертификат. Описание элементов приведено в Таблица 27.

Таблица 27 - Перечисление RequestParams

Наименование	Описание
RequestType	Тип запроса: First, Renew.
ActiveCertPin	Пин-код действующего сертификата, на котором следует подписать запроса.
ActiveCertId	Идентификатор действующего сертификата, на котором следует подписать запрос.
EkuString	Строка с EKU.
TemplateOid	Объектный идентификатор шаблона сертификата.

2.7.10. SignatureType

Перечисление содержит поддерживаемые Сервером Подписи форматы подписи. Описание элементов приведено в Таблица 28, Таблица 1.

Таблица 28 - Перечисление SignatureType

Наименование	Описание
XMLDSig	Подпись документа в формате XMLDSig
GOST3410	Электронная подпись по ГОСТ Р 34.10 - 2001
CAdES	Подпись формата CAdES-BES, CAdES-T, CAdES-X Long Type 1
PDF	Подпись PDF документов
MSOffice	Подпись документов MS Word и Excel
CMS	Подпись формата CAdES-BES

2.7.11. SignatureParams

Перечисление содержит типы дополнительных параметров, используемых при создании подписи. Описание элементов приведено в Таблица 29, Таблица 1.

Таблица 29 - Перечисление SignatureParams

Наименование	Описание
TSPAddress	Адрес TSP службы
CADESType	Тип CAdES подписи. Может принимать значения: BES, XLT1, T
PDFReason	Цель подписания PDF документа. Дополнительный атрибут для PDF.
PDFLocation	Местоположение. Дополнительный атрибут для PDF.
PDFFormat	Формат подписи PDF. Может принимать значения: CMS, CAdEST, CAdES
XMLDsigType	Тип XML подписи. Возможные значения: XMLEnveloped, XMLEnveloping, XMLTemplate
Hash	Использовать заданное значение хэша. Дополнительный параметр подписи ГОСТ Р 34.10-2001
IsDetached	Создать отделённую подпись. Дополнительный параметр подписи CMS, CAdES. Возможные значения: true, false.
PdfSignatureAppearance	Строковое представление шаблона видимой (отображаемой) PDF-подписи. Подробнее см. п. 2.7.14
PdfSignatureTemplateId	Идентификатор шаблона видимой (отображаемой) PDF-подписи. Подробнее см. п. 2.7.14
PdfCertificationLevel	Уровень сертификации подписи. Возможные значения: NOT_CERTIFIED, CERTIFIED_NO_CHANGES_ALLOWED, CERTIFIED_FORM_FILLING, CERTIFIED_FORM_FILLING_AND_ANNOTATIONS.
XPath	XPath к подписываемому узлу для XML подписи.

Наименование	Описание
HashAlgorithm	Алгоритм хэширования документа.
PdfSignatureSize	Максимально допустимый размер подписи в PDF.
CmsSignatureType	Тип подписи CMS: первая подпись, параллельная или заверяющая. Может принимать значения: Sign, Cosign, Countersign
SignatureIndex	Индекс подписи, для которой создается заверяющая подпись. Индекс отсчитывается от 0.
DataToBeSigned	Отображаемые данные для добавления в подписанный атрибут CMS сообщения. OID атрибута 1.2.643.2.2.44.3.
IncludeDtbs	Показывает, следует ли включать в подписываемые атрибуты атрибут с отображаемыми данными. Данные для атрибута могут быть взяты и маркера безопасности.

2.7.12. MfaPolicy

Перечисление содержит возможные значения политики многофакторной аутентификации. Описание элементов приведено в Таблица 30. Описание операций сервера подписи приведено в разделе

Таблица 30 - Перечисление MfaPolicy

Наименование	Описание
On	Дополнительное подтверждение требуется для каждой операции сервера подписи
Off	Дополнительное подтверждение не требуется.
NotSet	Не задано. Требование дополнительной аутентификации определяется на основе маркера безопасности пользователя.

В режимах On и Off сервер подписи игнорирует любые утверждения, содержащиеся в маркере пользователя.

2.7.13. DSSActions

Перечисление содержит действия сервера подписи, требующие доступа к закрытому ключу. Описание элементов приведено в Таблица 31.

Таблица 31 - Перечисление DSSActions

Наименование	Описание
Issue	Выпуск маркера безопасности («вход» пользователя)
SignDocument	Подпись документа
SignDocuments	Подпись пакета документов
DecryptDocument	Расшифрование документа
CreateRequest	Создание запроса на сертификат
DeleteCertificate	Удаление сертификата
ChangePin	Смена пин-кода для доступа к закрытому ключу сертификата
RenewCertificate	Создание запроса на обновление сертификата
RevokeCertificate	Создание запроса на отзыв сертификата
HoldCertificate	Создание запроса на приостановление действия сертификата
UnholdCertificate	Возобновление действия сертификата

2.7.14. EncryptionType

Перечисление содержит поддерживаемые Сервером Подписи форматы шифрования. Таблица 32 содержит описание элементов.

Таблица 32 - Перечисление EncryptionType.

Наименование	Описание
CMS	Шифрование в формате CMS

2.8. Видимая (отображаемая) подпись PDF-документов.

СЭП КриптоПро DSS позволяет добавлять в PDF-документы видимую (отображаемую) подпись. Видимая подпись может служить как подтверждение факта подписания электронного документа (см. Рисунок 2. Пример PDF-документа с видимой подписью.) при его печати.

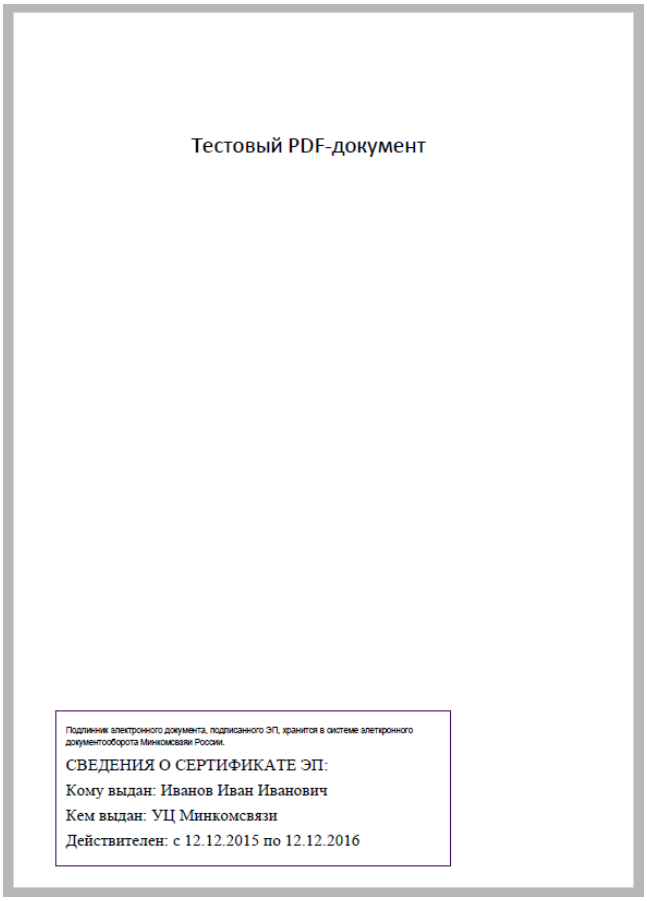


Рисунок 2. Пример PDF-документа с видимой подписью.

2.8.1. Шаблон подписи

Для того чтобы поставить видимую подпись в электронный PDF-документ необходимо передать описание представления подписи в метод SignDocument. Представление подписи (в терминологии PDF «signature appearance») можно описать шаблоном подписи, содержащим параметры данного представления.

Различные представления подписи описываются различными шаблонами.

2.8.2. Параметры шаблонов подписи

1. Размеры.

Размеры всех элементов в представлении подписи и в шаблоне указываются в единицах измерения – типографских пунктах Adobe (points). 1 пункт = 1/72 дюйма = 0,3528 мм.

2. Цвет.

Цвет элементов указывается в системе RGB (red, green, blue; красный, зелёный, синий).

3. Система координат.

Положение представления подписи на странице задаётся в системе координат PDF документа. Точка с координатами (0, 0) соответствует левому нижнему углу страницы.

2.8.3. Составные элементы шаблонов подписи

1. ColorDescription

Описание цвета элемента. Описание полей приведено в Таблица 33.

Таблица 33. Описание полей класса ColorDescription.

Поле	Тип	Описание
Red	int	Значение красной компоненты цвета
Green	int	Значение зелёной компоненты цвета
Blue	int	Значение синей компоненты цвета

2. SignatureRect

Описание прямоугольника подписи. Описание полей приведено в Таблица 34.

Таблица 34. Описание полей класса SignatureRect.

Поле	Тип	Описание
LowerLeftX	int	X координата левого нижнего угла прямоугольника
LowerLeftY	int	Y координата левого нижнего угла прямоугольника
UpperRightX	int	X координата правого верхнего угла прямоугольника
UpperRightY	int	Y координата правого верхнего угла прямоугольника
BorderRadius	int	Радиус скругления углов прямоугольника
BorderWeight	int	Толщина линии границы прямоугольника. 0 – отсутствие границы (значение по умолчанию)
BorderColor	ColorDescription	Цвет границы прямоугольника. По умолчанию (0, 0, 0)
BackgroundColor	ColorDescription	Цвет фона прямоугольника. По умолчанию (255, 255, 255)
ContentMargin	int	Отступ от границы прямоугольника до содержимого представления подписи

3. FontDescription

Описание шрифта. Описание полей класса приведено в таблице Таблица 35.

Таблица 35. Описание полей класса FontDescription.

Поле	Тип	Описание
FontSize	int	Размер шрифта

Поле	Тип	Описание
FontFamily	int	Название шрифта. Допустимые значения: times, arial По умолчанию times
FontColor	ColorDescription	Цвет шрифта

4. TextBlock

Описание блока текста. Блок текста – это набор символов, заканчивающийся переводом строки (абзац). Описание полей класса приведено в таблице Таблица 36.

Таблица 36. Описание полей класса TextBlock.

Поле	Тип	Описание
Text	string	Содержимое блока
Font	FontDescription	Описание шрифта
Margin	int	Отступ от границы прямоугольника до текста.

5. ImageBlock

Описание изображения – логотипа или фона прямоугольника подписи. Для задания изображения в качестве фона необходимо указать только поле Image.

Таблица 37. Описание полей класса ImageBlock.

Поле	Тип	Описание
Image	string	Байты изображения, закодированное в Base64. Поддерживаемые форматы изображений: JPEG, JPEG2000, GIF, PNG, BMP, WMF, TIFF, CCITT, JBIG2.
LowerLeftX	int	X координата левого нижнего угла прямоугольника. Только для описания логотипа.
LowerLeftY	int	Y координата левого нижнего угла прямоугольника. Только для описания логотипа.
Scale	Int	Масштаб изображения. Только для описания логотипа.

2.8.4. Типы шаблонов подписи

В данном разделе приводится описание зарегистрированных в СЭП КриптоПро DSS шаблонов видимой PDF-подписи.

1. Простой текстовый шаблон.

Идентификатор: 1

Описание полей шаблона приведено в Таблица 38.

Таблица 38. Описание полей простого текстового шаблона.

Поле	Тип	Описание
Content	TextBlock[]	Содержимое представления в виде массива текстовых блоков
Page	int	Номер страницы (первая страница имеет номер 1), на которой следует расположить представление
Rect	SignatureRect	Описание прямоугольника подписи.
TemplateId	int	Идентификатор шаблона. Должен иметь значение 1.

2. Шаблон с логотипом и текстом.

Идентификатор: 2

Описание полей шаблона приведено в Таблица 38.

Таблица 39. Описание полей шаблона с логотипом и текстом.

Поле	Тип	Описание
Content	TextBlock[]	Содержимое представления в виде массива текстовых блоков
Page	int	Номер страницы (первая страница имеет номер 1), на которой следует расположить представление
Icon	ImageBlock	Описание параметров логотипа.
Rect	SignatureRect	Описание прямоугольника подписи.
TemplateId	int	Идентификатор шаблона. Должен иметь значение 2.

3. Шаблон в виде изображения.

Идентификатор: 3

Описание полей шаблона приведено в Таблица 39.

Таблица 40. Описание полей шаблона в виде изображения.

Поле	Тип	Описание
Content	TextBlock[]	Содержимое представления в виде массива текстовых блоков
Page	int	Номер страницы (первая страница имеет номер 1), на которой следует расположить представление
Background	ImageBlock	Описание параметров изображения.
Rect	SignatureRect	Описание прямоугольника подписи.
TemplateId	int	Идентификатор шаблона. Должен иметь значение 3.

2.8.5. Передача параметров шаблона в метод подписи документов

Если в документ необходимо поставить видимую подпись, то в дополнительные параметры следует положить два значения: первое – идентификатор шаблона в качестве значения параметра **PdfSignatureTemplateId**, второе – параметры шаблона в качестве значения параметра **PdfSignatureAppearance**.

Содержимое шаблона должно быть представлено как Json объект, сериализованный в UTF-8 строку и закодированный в BASE64.

Пример 1:

```
{
	"Content": [{
		"Text": "Подлинник электронного документа, подписанного ЭП, хранится в системе элеткронного документооборота Минкомсвзяи России.",
		"Font": {
			"FontSize": 4,
			"FontFamily": "arial",
			"FontStyle": 0,
			"FontColor": {
				"Red": 0,
				"Green": 0,
				"Blue": 0
			}
		}
	},
	{
		"Text": "СВЕДЕНИЯ О СЕРТИФИКАТЕ ЭП:",
		"Font": {
			"FontSize": 8,
			"FontFamily": "times",
			"FontStyle": 0,
			"FontColor": {
				"Red": 0,
				"Green": 0,
				"Blue": 0
			}
		}
	},
	{
		"Text": "Кому выдан: Иванов Иван Иванович",
		"Font": {
			"FontSize": 8,
			"FontFamily": "times",
			"FontStyle": 0,
			"FontColor": {
				"Red": 0,
				"Green": 0,
				"Blue": 0
			}
		}
	},
	{
		"Text": "Кем выдан: УЦ Минкомсвязи",
		"Font": {
			"FontSize": 8,
			"FontFamily": "times",
			"FontStyle": 0,
			"FontColor": {
				"Red": 0,
				"Green": 0,
				"Blue": 0
			}
```

```
            }
      },
      {
            "Text": "Действителен: с 12.12.2015 по 12.12.2016",
            "Font": {
                  "FontSize": 8,
                  "FontFamily": "times",
                  "FontStyle": 0,
                  "FontColor": {
                        "Red": 0,
                        "Green": 0,
                        "Blue": 0
                  }
            }
      }],
      "TemplateId": 1,
      "Rect": {
            "LowerLeftX": 215,
            "LowerLeftY": 10,
            "UpperRightX": 405,
            "UpperRightY": 85,
            "BorderRadius": 0,
            "BorderWeight": 1,
            "BorderColor": {
                  "Red": 75,
                  "Green": 13,
                  "Blue": 100
            },
            "BackgroundColor": null,
            "ContentMargin": 5
      },
      "Page": 1
}
```


В BASE64


```
eyJDb250ZW50IjpbeyJUZXh0Ijoi0J/QvtC00LvQuNC90L3QuNC6INGN0LvQtdC
60YLRgNC+0L3QvdC+0LPQviDQtNC+0LrRg9C80LXQvdGC0LAsINC/0L7QtNC/0L
jRgdCw0L3QvdC+0LPQviDQrdCfLCDRhdGA0LDQvdC40YLRgdGPINCyINGB0LjRg
dGC0LXQvNC1INGN0LvQtdGC0LrRgNC+0L3QvdC+0LPQviDQtNC+0LrRg9C80LXQ
vdGC0L7QvtCx0L7RgNC+0YLQsCDQnNC40L3QutC+0LzRgdCy0LfRj9C4INCg0L7
RgdGB0LjQuC4iLCJGb250Ijp7IkZvbnRTaXplIjo0LCJGb250RmFtaWx5IjoiYX
JpYWwiLCJGb250U3R5bGUiOjAsIkZvbnRDb2xvciI6eyJSZWQiOjAsIkdyZWVuI
jowLCJCbHVlIjowfX19LHsiVGV4dCI6ItCh0JLQldCU0JXQndCY0K8g0J4g0KHQ
ldCg0KLQmNCk0JjQmtCQ0KLQlSDQrdCfOiIsIkZvbnQiOnsiRm9udFNpemUiOjg
sIkZvbnRGYW1pbHkiOiJ0aW1lcyIsIkZvbnRTdHlsZSI6MCwiRm9udENvbG9yIj
p7IlJlZCI6MCwiR3JlZW4iOjAsIkJsdWUiOjB9fX0seyJUZXh0Ijoi0JrQvtC80
YMg0LLRi9C00LDQvTog0JrQvtC30YvRgNC10LIg0JDQu9C10LrRgdC10Lkg0J7Q
u9C10LPQvtCy0LjRhyIsIkZvbnQiOnsiRm9udFNpemUiOjgsIkZvbnRGYW1pbHk
iOiJ0aW1lcyIsIkZvbnRTdHlsZSI6MCwiRm9udENvbG9yIjp7IlJlZCI6MCwiR3
JlZW4iOjAsIkJsdWUiOjB9fX0seyJUZXh0Ijoi0JrQtdC8INCy0YvQtNCw0L06I
NCj0KYg0JzQuNC90LrQvtC80YHQstGP0LfQuCIsIkZvbnQiOnsiRm9udFNpemUi
OjgsIkZvbnRGYW1pbHkiOiJ0aW1lcyIsIkZvbnRTdHlsZSI6MCwiRm9udENvbG9
yIjp7IlJlZCI6MCwiR3JlZW4iOjAsIkJsdWUiOjB9fX0seyJUZXh0Ijoi0JTQtd
C50YHRgtCy0LjRgtC10LvQtdC9OiDRgSAxMi4xMi4yMDE1INC/0L4gMTIuMTIuM
jAxNiIsIkZvbnQiOnsiRm9udFNpemUiOjgsIkZvbnRGYW1pbHkiOiJ0aW1lcyIs
IkZvbnRTdHlsZSI6MCwiRm9udENvbG9yIjp7IlJlZCI6MCwiR3JlZW4iOjAsIkJ
sdWUiOjB9fX1dLCJUZW1wbGF0ZUlkIjoxLCJSZWN0Ijp7Ikxvd2VyTGVmdFgiOj
IxNSwiTG93ZXJMZWZ0WSI6MTAsIlVwcGVyUmlnaHRYIjo0MDUsIlVwcGVyUmlna
HRZIjo4NSwiQm9yZGVyUmFkaXVzIjowLCJCb3JkZXJXZWlnaHQiOjEsIkJvcmRl
ckNvbG9yIjp7IlJlZCI6NzUsIkdyZWVuIjoxMywiQmx1ZSI6MTAwfSwiQmFja2d
yb3VuZENvbG9yIjpudWxsLCJDb250ZW50TWFyZ2luIjo1fSwiUGFnZSI6MX0=
```

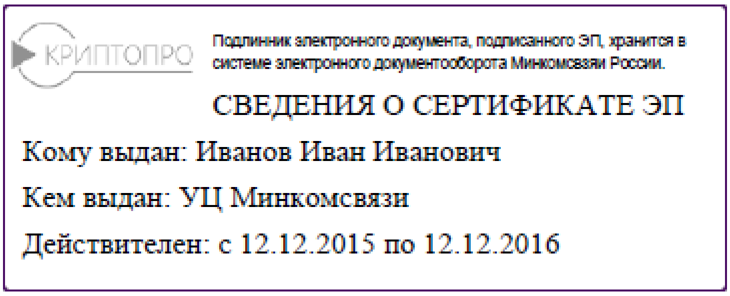
Пример 2:

```
{
      "Content": [{
            "Text": "Подлинник электронного документа, подписанного ЭП, хранится
в системе элеткронного документооборота Минкомсвзяи России.",
      "Margin": 50,
            "Font": {
                  "FontSize": 4,
                  "FontFamily": "arial",
                  "FontStyle": 0,
                  "FontColor": {
                        "Red": 0,
                        "Green": 0,
                        "Blue": 0
                  }
            }
      },
      {
            "Text": "СВЕДЕНИЯ О СЕРТИФИКАТЕ ЭП",
      "Margin": 50,
            "Font": {
                  "FontSize": 8,
                  "FontFamily": "times",
                  "FontStyle": 0,
                  "FontColor": {
                        "Red": 0,
                        "Green": 0,
                        "Blue": 0
                  }
            }
      },
      {
            "Text": "Кому выдан: Иванов Иван Иванович",
            "Font": {
                  "FontSize": 8,
                  "FontFamily": "times",
                  "FontStyle": 0,
                  "FontColor": {
                        "Red": 0,
                        "Green": 0,
                        "Blue": 0
                  }
            }
      },
      {
            "Text": "Кем выдан: УЦ Минкомсвязи",
            "Font": {
                  "FontSize": 8,
                  "FontFamily": "times",
                  "FontStyle": 0,
                  "FontColor": {
                        "Red": 0,
                        "Green": 0,
                        "Blue": 0
                  }
            }
      },
      {
            "Text": "Действителен: с 12.12.2015 по 12.12.2016",
            "Font": {
                  "FontSize": 8,
                  "FontFamily": "times",
                  "FontStyle": 0,
```

```
                  "FontColor": {
                        "Red": 0,
                        "Green": 0,
                        "Blue": 0
                  }
            }
      }],
      "TemplateId": 2,
      "Icon": {
            "Image":
"iVBORw0KGgoAAAANSUhEUgAAAIIAAAA6CAYAAAB8moHlAAAABGdBTUEAAYagMeiWXwAAAAlwSFlz
AAAOxAAADsQBlSsOGwAAAAJiS0dEAP+Hj8y/AAAAJXRFWHRkYXRlOmNyZWF0ZQAyMDE2LTExLTI1
VDA5OjIyOjA0KzAxOjAwg5fT4gAAACV0RVh0ZGF0ZTptb2RpZnkAMjAxNi0xMS0yNVQwOToyMjow
NCswMTowMPLKa14AAArsSURBVHhe7ZwHrBRVF8cvnYD0Ji30JgFCbwZCRzqCYOgQkCIBqcFCVXon
AkovQZSHD0JJCCGQEGoINSAdpSq9F0WYb37n3eFb9u3s2903b0z87/nnb0z87/nnHvO3U1m
2SjDf57k+q/hP44RgkEwQjAIRggGwQjBIBghGISoC4HZ6suXL/WW4W0hyfIIPOznz5+rAwcOqP37
96tTp06pW7du6b1xpEYZUhUuXFiVK1dOvf/++yyVKlmyjZPoIQ7TwXAi3b99WO3fuVHv2
7FHnzzp1TqVKlUsWLF1f58uVT2bJlU+nSpZMHjUgePHigrl+/rn777Td14P7Vuuq2Me2ooIz8cj
RX1GQzTwVAirVq1S69atUy9evFBVqlRR7du3VdtAP/Pb75JBYuXpuuVptk/5u
XGrUqFEqT548t+QtHgiBMz/vHnzZIQ3a9ZMtWjRQkZ/pOzbt09t3tiYOsq99VJmIjbNnZ/kiMd9nx9v3S9yxzbrmjbgLT09JpHuXpsIwRVvW7WFWJRKFXAtPUfgpqmLaZrA9s44VyCW2eDDdnu+SVnOqwJuBkLQgD1Jx9TEE1cbXF3L10MEpqIONmYuPp7UJlKlOCo9IMMTYTSLjFDkpCLjs17uwyfMBfSCTVf23GpMvnRj8wDkUzRtcXRWR6Jzpx7aUyNWtyzvYHZQ==
```

```
        "LowerLeftX": null,
              "LowerLeftY": 50,
        "Scale": 40
    },
        "Rect": {
              "LowerLeftX": 215,
              "LowerLeftY": 10,
              "UpperRightX": 405,
              "UpperRightY": 85,
              "BorderRadius": 0,
              "BorderWeight": 1,
              "BorderColor": {
                    "Red": 75,
                    "Green": 13,
                    "Blue": 100
              },
              "BackgroundColor": null,
              "ContentMargin": 5
        },
        "Page": 1
}
```

Рисунок 3. Отображение подписи с логотипом и текстом.



Пример 3:


```
{ "Background": {
          "Image":
"iVBORw0KGgoAAAANSUhEUgAAALgAAABMCAYAAADeMH3IAAAABGdBTUEAALGPC/xhBQAAAAFzUkdC
AK7OHOkAAAAgY0hSTQAAeiYAAICEAAD6AAAAgOgAAHUwAADqYAAAOpgAABdwnLpRPAAAAAZiS0dE
AP8A/wD/oL2nkwAAAAlwSFlzAAAuIwAALiMBeKU/dgAAD21JREFUeNrtnXl0VNUdxz8zAWRJAhGQ
LRiJS6WpRUWagqLiftxwOYYKrUXqQeWota21LVBX6ylKa3ailuoWy1oi6FKq7hSFrGKHJWqCGLY
wxYkQBaSmf7xvc9chiTzJpktyf2eMyeTmTf33fte9/7ub70PHBwcHBwcHBwcHBwcHBwc
HNouAqnuAEB+UTFAB6AH0A/oD/QFcoBM088QUAHsALYAG4Ey4Csg9MWcCakehkMaIiUEN4QG6AY
BkYAw4ECYACQjQjfGPYD5cB64ENgOWCsSj5nKSwcR2Ld5wdGpzvmd2mgjOOLvYAYUNmQQ1Nqz4Q4S
1A4DGYj4vYFc4AhgINDdNF8GLAH+Abxm/dIcs+ChN+ANqKcjh8iEQiAwtJoQ19ja2R5uSKaX45cP
UEvyd0TEHggcD5wOnIJIH0ZS/Wngb8DmyHYc2g8SSnCLkLL0Qsa9HxN4FLABQBt632a3ZQ2M5Wq
PwP4HjAS6Awm0Rvb0gYWS2pfQowFTgN2AfMAx4FLBBmI/ViK83Qo632M5BU
vM7n0DqQEIIbsnUFfgRMBvogSX0P0o+rE0kyc/4s4HLgFmS8fgbchsheStCcwBZiI
pPTDwAySaPhZfTkaTbKxaAW5G3gQqHQkb/uIG8EtvUDfgdcibwhk4EXgQ2P2pIJTpVzfgWuDX5v0f
gLuAvY7kbRvBOd3GPAAkpYgR8Cz5EicsPXq8VeROrrkevxZ2iF6WZNTIc2iIx4NGJIko0k9zjk
prsG+aVTbtSVrywhp2B0GE261cjgPRsFjJbkFIwOla8sSWkfHRKtktwWwK4F1xV4F4fP+Pbk
WP2YB9yEIqG3ogmJk+RtEy3SwRWMDpwVe8rgKIBgi48PwaIYn7DOG0PPDATQJf49ULjEo1J+WfXXZoPuKhg38D
uAPoAvwWBekL1FMv8LALOSPHwjciYJFfAZ+rg0M2BkjL10QVUCLRkqeGuvWSdZWeBVn2PfNatY
rHE5tBE0i+Ac4+ofEOsk/arIevBIqE+hZqf4LYp7COm2d4T2+FLQTJVQ5bmi4DBpjM7pzC+ZAC
bkSh1YDYSEz2R0g1+Fn0n9+F43hO1p
1kJuh/YDPypKBpJwz6MgTsHq1+8NbO7SYxVST36KcrcdHNIOfiR4fySS5Pf10LfcI/AXpr
VmgOwQHqgPdR+HseJtc6lURvURvthjzEzW0ZA
xLlZ/GouwT1UAv9GAZN3gNp4ktzcrBNRXvefgT2RN8iPpF2N8iKpI5B9XeBXyCKog+
xyqysM51pmmjoiEyWAQqAH6AqoZWWEZrPspF7+JzmDtQduMO099zMWS0UGfula1SBtDNn4mqojzk
onLAU1HiWwBlcS4G/ok8Xzw/DzmAcAk5Dzwgwpz7nV5G5Z5bYww96Eb9Q6MCtO/hShVokGOmN93
```

```
QfUDFyAbrysSnh+bvi8DamLhWEsDPV2AS1HFzP3A0aW5hQFrJscDBaga6CDCWFHJ76PswDJzQQFO
RvngNShauR2l8V5rLtavgOwIyT8YpRtkRulTFgoO/QI4NuK7ASgPfqBpJ6uRV6Y53wRUk4rpX3bE
cTkoee30iPYyUdme19egGe88RNRyoAR4EdW6XoccBVcCHRpY8fqh9OZjfPT7SHNs/4g2TjL3KoAm
bJk5fhxySNwD9C7NLfxa2lvvj0TR8EeAw4H/AHOA/yLB8xwSpP3s30dDB19HRUdv4MfA+UjSPlua
W7gdEqe2WDfoEhRJnY7qMMPm8yCSGDdjXJzmwvcCilAabxZwW35Rca35Phzxt7FzjgdOQMZ15CoY
QCvIZOB/UYZxrrleHmaZl41uqEhkgRlnY7gS+fRnI9XRNvwDaMLdYr7LAh7LLyqODMpVmzbeitLv
Ecjp0NDY16OJv936rDvixl2IK5Mwaq3BIDPuDubaLsWoJdY1uMj8PhdNrq34QLwI7g3kGJRReBkq
cHilNLewMt4kt4h2BpIKTyDDN2SpCgfAC9HnFxVvA/6EbuZUJOWWxnD6E4EbkJF9MVFWwcZ03ib6
6PsY6/shZixPIhLURLQVzi8qXoeSy6qA24GPgKX5RcWw5JGDztPYPYtldTZthEtzC3cBf0WBwEeA
c4C5pq1DUKVVF7RSfR55/tLcwr3UF5A/jTx3k0tzC0PRuBXvfHDQpDnVdORxYFhpbmEwXmqLdVOH
IWlUgqS3r5xz65i5SAqMiuG8WYgcq9AF96tnJxJBpH6UoQLrmoaug/msCgmg9WiSdkxGBy0SLkAB
wXOsr09GK9k0GiB3xP9vI3tlLHCc34uTKGSajpSYzh8Ri+4UBYPNQBcDvwGqmuEd+Ard6L7RDoxQ
TYYgKVlOeuwMdjgyjJ+lXi1oCluBp9DOA0clua+VKB+/H/W1CJcgYr8Bja8c1uezTTsXQvQVJZEE
99AP+CUy7CYCPVpI9DxE7tVIz93TTNdXB6TbVTV1kEXuk5A+/yjKdU/GtfOD49AWGYugaTeg9d3b
5u8JSe5rAK2C+5BnKBv4jul7hc82NiBvygigU7SDk3WTAsC3EDHnoMhopxhI7vl0c5FaUoHKzspb
4Nc9Crn0Vvg4tjtSTdYgSz/lsCbeILQabYnh55uBbchz0bBFnRj0RyvgB+b/HGR0fgrRHRLm+xBy
G+YS3dsVVyPTDzoh//R3kRV+Z2lu4boogwshD8CzyANSjYzYshZsFNQL5ddsRB6KaMdPQDtojSX9
ErJ6IOOtMobfVCIV61CAUCBARjgxNLeEWBbwcySsXjSfZSI7YGeMze5EAq9ztANTtcxWoaBLrY9j
A0hiL0CqQW+0PAV85G53RjO9d35R8QC0nI9HVf6DkTsrmrvpJJQ1ORNTY5pmCCF9NtZ7GUBqQrwN
iU4oFtC7NLewP6oVKEJ+7PORYPnc6jvEXhvsdTnqrEy2BN+HytlmID2qzocLMYBm7Ewkqe5AQaU6
YHZ+UXFjemcdMBTp/nWIAB1NH97AigA25lqkXjVZhdxb4SaOTRV2UB/w2e3zN12R5N8JEIif9A4h
lWk28mMHEcf2I1fs7dSrJyDBVY3stFjQE61CVdEOTBbB61BEagbwMs3LXQmYAU1Dy9P9KEo5txGS
B5FuNxVJiP3Iy/Al/vdCvBpJ/bHEvowmFNZEW4OMtUGogMMP+iA1bY13YeOEoOnDFOrzX3aia74F
K0xvVJdy8/kQ77OmOGHlqQxGHrCoBTbJIPgXSFo+hdnZqoWBn32ItB2QwVkNzG+A5F64+BVi3you
hDwMN6Bdud6BtE3SWomMzLOBxU2saPbKM9yMcUUC+lMBvA5s93Gf9yK17zw06cp8tN8PrczPcGC0
s0EkUgfficLQF6LgQlnehmUtDt2bm1eBlrtXgIdQoCBeqkMYScTJaAuMx63zpiM2oVXR25I6Gnqg
3JDFSPikBBYPSpBddTk07te2Pr8IqWPzI9ppEIkgeDVK6rkC+Y0/yduwLBzPcL0h224Uen4Thd5P
hxaT3MvQKwK+jVaK8gRco3hehzByXQaQatAjv6j4oOtgZV1OQikVD+NDAiYB7wJ/R3kyp8DBJLf+
H4bC9HOQqzAq4knwEDIgJiKd9Q1gf6KSrczN3YVIvgypQSPjMAYvq24mshvSWXp7+BQF00ahFed4
TMageQWRq3Ua8gjdSxrsOma4UYsSvFai/J5xQLYXDDTkzkRZq7PQqjodiJqHAvHTwdebC/sEJost
icUPO9BK8RAi5QRE+OYgjEL3LwLFkP7ktozNEqTTTjH9X46ivbXIVTrUvL8V5dGk0/Yem1Aa82Q0
+SYh+2AX8p8PQU/veBlNUj+6OtBygu9Gy8sDKInG16yKEe8hHf4gi9m6udsQya9A0TGQfv4R/nzt
HlagCOl8ZLg1hT2mX8sju4U2QPJzEz5D2ZBNBY+q0fUtbewAcx3CKFawHOWmnIq2jg4iT8V08/16
7zcRWIfyelb76Pdac+yXEZ8vRCqd7+1DLK/KJqR+PIOSsY5FaRme0fqqGVtMBQ/NrejZj/IZZpiT
V7u6zPSBVQgSpD6gk04Su1FY+nbQvELmlZSStTBK4n8QmFMXCJT3rd5D162+9H0Hh6TDr4oSQMGR
YuDRwpE3ru0YqsvoGK4bAfTMl9ekVUgIh/YFPwSvRYlRLyGXTl3XuhqQfvcUUlc+RnqZg0NawY+K
4ulCX4dZrYdMPY0y++5EPuO09zrYsDYQPcGMx6H1YB9ySzcZpW5WGkJETeQLKPHlUtJs03ufY7gU
uTj9bpfgkHp4eUZnAZub4luz3ISWe24hkuI3IR/meOS7bC3IM/3ORgW7q0ifSh2HxuHlGUWtAorH
Ex4GIV/4EJQfci9p/hxK0+8uKFlrIgqMjAe+Sud+O8SOeEirtShHexeq2LgM0ncjedOvIHri2tVI
ak8lemDHoRWiRU9ZM08vA0W/wijtcTiKaq7NKRhNOj29zAqAjEMRxBrgJ8jP3ypsB4fY0OLHCFqP
6PsApWJ6NZcfAqXpQnKL3EUoAtsNrTxP4nz4bRZxeU6mIfl+VBjQC5F8JKoYWZ1qkhtyd0R69nRU
4nU30sFT9pBah8QjLgQHPFWlCu1xkYMSfs5Clu7KnILRtckmeX5Rsdev7qjAeCoqip2GEqVirfRx
aGWI+85M1hZntyL9NojUgHswGXHJIJWlkhyHUkhHI9fSHSivuMaRu+0jIVuPGXJ1QjueTkGuxBVI
apZg0ikTQTDLe9MTGZM3m/O/j3zer+Gv4NihDSBhe+tZRDsR7R56AfK0vIoKExYSR6Jb5zsUeXOu
Rx6dShSMuo8GNn93aNtI+OaRlspyBSqXGoJItwjV1r2Jku1rITbyWaQ+BG1Ddp45z1Dz+SLkMXkV
p2+3SyRld1SLiAMQAa9CunEQVZgsQRJ9BSJ7ORHPlbHayEA7Vh2K9hYcCpwGFKKtB6pQydos9LSD
cq8Nh/aHpG7/a5G0D/KyXIIqqfsgsu9F9Xnrzd8dSI2pRqH1LOSGHICKaPsistchA/Yt9DiVRZjI
pCN2+0bK9re2DNFBaAvd4Uiq5yEDsTMNpxLUIRVnG6p//ABtC/YeKnhO6zwYh+QiHTZwt1163dAm
MH2Aw1BkNBNNhCokzctRQW8Z2oqtEpykdnBwcHBwcHBwcHBwcHBwcHBof/g/oBUGLguZ
XBkAAAAldEVYdGRhdGU6Y3JlYXRlADIwMTYtMTAtMTRUMDk6NDA6MzQrMDI6MDDZ/SvcAAAAJXRF
WHRkYXRlOm1vZGlmeQAyMDE2LTEwLTE0VDA5OjQwOjM0KzAyOjAwqKCTYAAAABl0RVh0U29mdHdh
cmUAQWRvYmUgSW1hZ2VSZWFkeXHJZTwAAAAASUVORK5CYII=",
  },
  "TemplateId": 3,
      "Rect": {
            "LowerLeftX": 10,
            "LowerLeftY": 10,
            "UpperRightX": 194,
            "UpperRightY": 86,
      },
      "Page": 1
}
```

3. Описание интерфейса REST для работы с сервером электронной подписи «КриптоПро DSS»

REST интерфейс предоставляет конечные точки для доступа к ресурсам СЭП:

- Сертификаты пользователя.
- Запросы пользователя.
- Политика СЭП.
- Документы.

В рамках REST API взаимодействие с СЭП осуществляется посредством HTTP-запросов.

Общий префикс для всех конечных точек сервиса:

https://<host>/<AppName>/rest/api, где

host – адрес сервера, на котором расположен экземпляр приложения СЭП,

AppName – название веб-приложения СЭП.

3.1. Формат входных и возвращаемых значений

Входные параметры могут передаваться внутри URL-адреса запроса и в теле самого запроса (только для HTTP методов POST и PUT). Внутри URL-адреса могут передаваться только примитивные типы (строки (string), числа (int)).

Все объекты передаются и возвращаются в формате JSON, списки возвращаются в виде массивов JSON объектов. Массивы байтов передаются в виде строк в кодировке BASE64.

3.2. Конечная точка Certificates

Данная конечная точка предоставляет доступ к сертификатам пользователя СЭП. Перечень методов, реализуемых конечной точкой **Certificates** представлен в Таблица 41.

Таблица 41. Описание методов конечной точки Certificates.

HTTP метод	Путь	Описание
GET	/certificates	Получение списка сертификатов пользователя
Параметры	Тип	Описание
Запрос не имеет параметров		
Возвращаемое значение	Тип	Описание
	List<DSSCertificate>	Список всех сертификатов пользователя
GET	/certificates/{key}	Получение сертификата с заданным идентификатором
Параметры	Тип	Описание
key	int	Идентификатор сертификата

HTTP метод	Путь	Описание
Возвращаемое значение	Тип	Описание
	DSSCertificate	Сертификат пользователя с заданным id
POST	/certificates	Создание (установка) сертификата.
Параметры	Тип	Описание
certificate	string	Содержимое сертификата в формате BASE64.
Возвращаемое значение	Тип	Описание
	DSSCertificate	Сертификат пользователя.
DELETE	/certificates/{key}	Удаление сертификата с заданным идентификатором key.
Параметры	Тип	Описание
key	int	Идентификатор сертификата
Возвращаемое значение	Отсутствует	
PATCH	/certificates/{key}/status	Изменение статуса сертификата с идентификатором key.
Параметры	Тип	Описание
key	int	Идентификатор сертификата, статус которого требуется изменить.
status	CertificateStatus	Новый статус сертификата.
Возвращаемое значение	Отсутствует	
PATCH	/certificates/{key}/pin	Изменение пин-кода для доступа к закрытому ключу сертификата с идентификатором key.
Параметры	Тип	Описание
key	int	Идентификатор сертификата, статус которого требуется изменить.
pinRequest		Запрос на изменение пин-кода для доступа к закрытому ключу сертификата.
Возвращаемое значение	Отсутствует	
PATCH	/certificates/{key}/default	Назначение сертификата сертификатом по умолчанию.

HTTP метод	Путь	Описание
Параметры	Тип	Описание
Default	bool	Значение свойства, показывающего является сертификат, сертификатом по умолчанию.
Возвращаемое значение	Отсутствует	
PATCH	/certificates/{key}/friendlyName	Изменение дружественного имени сертификата
Параметры	Тип	Описание
FriendlyName	string	Дружественное имя сертификата. Длина дружественного имени не может превышать 255 символов. Для удаления дружественного имени требуется передать пустую строку.

3.3. Конечная точка Requests

Данная конечная точка предоставляет набор методов для управления запросами на сертификат пользователя. Перечень методов, реализуемых конечной точкой **Requests** представлен в Таблица 42.

Таблица 42. Описание методов конечной точки Requests.

HTTP метод	Путь	Описание
GET	/requests	Получение списка запросов пользователя на сертификат.
Параметры	Тип	Описание
Запрос не имеет параметров		
Возвращаемое значение	Тип	Описание
	List<DSSCertRequest>	Список всех запросов пользователя на сертификат.
GET	/requests/{key}	Получение запроса пользователя на сертификат с заданным идентификатором.
Параметры	Тип	Описание
Key	int	Идентификатор сертификата.
Возвращаемое значение	Тип	Описание
	DSSCertRequest	Запрос пользователя на сертификат с заданным идентификатором.

HTTP метод	Путь	Описание
POST	/requests	Создание нового запроса на сертификат.
Параметры	Тип	Описание
Request	CertificateRequest	Запрос на создание сертификата.
Возвращаемое значение	Тип	Описание
	DSSCertRequest	Запрос пользователя на сертификат.
DELETE	/requests/{key}	Удаление запроса на сертификат с заданным идентификатором key.
Параметры	Тип	Описание
Key	int	Идентификатор запроса на сертификат.
Возвращаемое значение	Отсутствует	
PATCH	/requests/{key}/status	Изменение статуса запроса на сертификат с идентификатором key.
Параметры	Тип	Описание
Key	int	Идентификатор запроса на сертификат, статус которого требуется изменить.
Status	RequestStatus	Новый статус запроса на сертификат.
Возвращаемое значение	Отсутствует	

3.4. Конечная точка Documents

Данная конечная точка предоставляет набор методов для управления документами пользователя. Перечень методов, реализуемых конечной точкой **Documents** представлен в Таблица 43.

Таблица 43. Описание методов конечной точки Documents.

HTTP метод	Путь	Описание
POST	/documents	Отправка документа в СЭП
Параметры	Тип	Описание
	Document	Информация о документ.
	Тип	Описание

HTTP метод	Путь	Описание
Возвращаемое значение	byte[]	Подпись документа (или подписанный документ) в виде массива байтов
POST	/documents/enhancesignature	Усовершенствование подписи.
Параметры	Тип	Описание
	Document	Информация о документе и типе подписи.
Возвращаемое значение	Тип	Описание
	byte[]	Усовершенствованная подпись в виде массива байтов.
POST	/documents/encrypt	Шифрование документа.
Параметры	Тип	Описание
	Document	Информация о документе и типе шифрования.
Возвращаемое значение	Тип	Описание
	byte[]	Результат шифрования в виде массива байтов.
POST	/documents/decrypt	Расшифрование документа
Параметры	Тип	Описание
	Document	Информация о документе и типе шифрования.
Возвращаемое значение	Тип	Описание
	byte[]	Расшифрованные данные в виде массива байтов.
POST	/documents/decrypt/parse	Разбор метаданных зашифрованного сообщения. Данный метод позволяет получить информацию о доступных сертификатах, с помощью которых возможно расшифровать данный документ.
Параметры	Тип	Описание
	Document	Информация о документе и типе шифрования.
Возвращаемое значение	Тип	Описание
	List<int>	Список идентификаторов сертификатов пользователя, с помощью которых документ можно расшифровать.

HTTP метод	Путь	Описание
POST	/documents/packagesignature	Запрос на пакетную подпись документа.
Параметры	Тип	Описание
	DocumentPackage	Информация о пакете документов и типе подписи.
Возвращаемое значение	Тип	Описание
	DSSSignDocumentResponse	Результат подписи пакета документов.

3.5. Конечная точка Policy

Данная конечная точка предоставляет доступ к настройкам СЭП. Перечень методов, реализуемых конечной точкой **Policy** представлен в Таблица 44.

Таблица 44. Описание методов конечной точки Policy.

HTTP метод	Путь	Описание
GET	/policy	Получение настроек СЭП
Параметры	Тип	Описание
Метод не имеет параметров		
Возвращаемое значение	Тип	Описание
	DSSPolicy	Политика СЭП

3.6. Конечная точка Transactions

Конечная точка функционал по созданию транзакций на сервисе подписи. Перечень методов, реализуемых конечной точкой **Transactions** представлен в Таблица 45.

Таблица 45. Описание методов конечной точки Transactions.

HTTP метод	Путь	Описание
POST	/transactions	Создание новой транзакции.
Параметры	Тип	Описание
	Transaction	Информация о транзакции.
Возвращаемое значение	Тип	Описание
	string	Идентификатор транзакции.

3.7. Типы данных

В данном разделе представлены типы данных, специфичные для REST интерфейса СЭП.

3.7.1. Document

Объекты данного типа содержат информацию об отправляемых в СЭП документах. Описание полей класса Document приведено в Таблица 46.

Таблица 46. Описание полей класса Document.

Поле	Тип	Описание
Content	byte[]	Содержимое документа
Name	string	Название документа
Signature	DocumentSignature	Информация о подписи документа.
Encryption	DocumentEncryption	Информация о параметрах шифрования документа.
Decryption	DocumentDecryption	Информация о параметрах расшифрования документа.

3.7.2. DocumentSignature

Объекты данного типа содержат информацию о требуемой подписи документа. Описание полей класса приведено в Таблица 47.

Таблица 47. Описание полей класса DocumentSignature.

Поле	Тип	Описание
SignatureType	SignatureType	Тип подписи
Parameters	Dictionary<SignatureParams, string>	Дополнительные параметры подписи
CertificateId	int	Идентификатор сертификата
PinCode	string	Пин-код для доступа к закрытому ключу сертификата

3.7.3. AgreeKeyInfo

Объекты данного типа содержат информацию о параметрах формирования ключа согласования. Описание полей класса приведено в Таблица 48.

Таблица 48. Описание полей класса AgreeKeyInfo.

Поле	Тип	Описание
PublicKeyInfo	byte[]	Байты внешнего открытого ключа.
Ukm	byte[]	Ключевой материал пользователя (user key material)
Mask	byte[]	Аддитивная маска для экспорта ключа
CertificateId	int	Идентификатор сертификата пользователя
PinCode	string	Пин-код для доступа к ЗК сертификата пользователя

3.7.4. Transaction

Объекты данного типа содержат информацию о транзакции на сервисе подписи. Описание полей класса приведено в Таблица 49.

Таблица 49. Описание полей класса Transaction.

Поле	Тип	Описание
OperationCode	DSSAction	Идентификатор операции, для которой формируется транзакция.
Document	byte[]	Данные транзакции.
Parameters	TransactionParameter[]	Массив параметров транзакции.
Documents	List<DocumentContent>	Содержимое документов, входящих в пакет при пакетной подписи.

3.7.5. TransactionParameter

Объекты данного типа содержат информацию о параметре транзакции. Описание полей класса приведено в Таблица 50.

Таблица 50. Описание полей класса TransactionParameter.

Поле	Тип	Описание
Name	string	Название параметра
Value	string	Значение параметра

3.7.6. CertificateRequest

Объекты данного типа содержат информацию о создаваемом запросе на сертификат. Описание полей класса приведено в Таблица 51.

Таблица 51. Описание полей класса CertificateRequest.

Поле	Тип	Описание
AuthorityId	int	Идентификатор удостоверяющего центра, к которому направлен запрос на сертификат.
PinCode	string	Пин-код для доступа к закрытому ключу сертификата.
Template	string	Идентификатор шаблона сертификата, по которому создаётся запрос.
DistinguishedName	IDictionary<string, string>	Набор компонентов различительного имени субъекта в виде пар {oid, значение.
Parameters	IDictionary<RequestParams, string>	Словарь дополнительных параметров запроса.

3.7.7. DocumentEncryption

Объекты данного тип содержат информацию о параметрах шифрования документа. Таблица 52 содержит описание полей класса.

Таблица 52. Описание полей класса DocumentEncryption.

Поле	Тип	Описание
EncryptionType	EncryptionType	Тип шифрования.
Parameters	Dictionary<string, string>	Дополнительные параметры шифрования.
Certificates	List<byte[]>	Список сертификатов получателей.

3.7.8. DocumentDecryption

Объекты данного типа содержат информацию о параметрах расшифрования документа. Таблица 53 содержит описание полей класса.

Таблица 53. Описание полей класса DocumentDecryption.

Поле	Тип	Описание
EncryptionType	EncryptionType	Тип шифрования.
CertificateId	int	Идентификатор сертификата получателя.

Поле	Тип	Описание
PinCode	string	ПИН-код для доступа к закрытому ключу сертификата.

3.7.9. DocumentPackage

Объекты данного типа содержат информацию о пакете документов. Таблица 54 содержит описание полей класса.

Таблица 54. Описание полей класса DocumentPackage

Поле	Тип	Описание
Documents	List<DocumentContent>	Содержимое документов, входящих в пакет.
Name	string	Название документа
Signature	DocumentSignature	Информация о подписи документа.
Encryption	DocumentEncryption	Информация о параметрах шифрования документа.
Decryption	DocumentDecryption	Информация о параметрах расшифрования документа.

3.7.10. DocumentContent

Объекты данного типа содержат информацию об одном документе из пакета документов. Таблица 55 содержит описание полей класса.

Таблица 55. Описание полей класса DocumentContent

Поле	Тип	Описание
Name	string	Название документа.
Content	byte[]	Содержимое документа.

3.8. Обработка ошибок

В случае успешного выполнения операции статус HTTP-ответа сервера имеет значение 200. В случае возникновения ошибки сервер вернёт статус из диапазона 400-500. Тело ответа будет содержать JSON объект с полем **Message**, хранящим информацию об ошибке.

Пример ответа с ошибкой авторизации.

```
HTTP/1.1 401 Unauthorized
Content-Length: 61
Content-Type: application/json; charset=utf-8
WWW-Authenticate: Bearer
Date: Tue, 10 May 2016 10:30:36 GMT
```

```
{"Message":"Authorization has been denied for this request."}
```

3.9. Аутентификация и авторизация

Аутентификация и авторизация при использовании REST интерфейса осуществляется с использованием JWT маркера доступа, получаемому по протоколу OAuth 2.0.

Для получения маркера доступа Центр Идентификации КриптоПро DSS предоставляет соответствующие конечные точки, реализующие протоколы OAuth 2.0 и OpenId Connect 1.0.

3.10. Примеры

В данном разделе приведены примеры запросов и ответов сервера для разных конечных точек. В примерах указаны только значимые HTTP-заголовки. Подразумевается, что клиентское приложение уже получило маркер доступа.

Маркер доступа, а также BASE64 содержимое сертификатов, в примерах указывается в урезанном виде для обеспечения большей наглядности.

3.10.1. Получение списка сертификатов пользователя

Запрос.

```
GET http://cloudcsp/SignServer/rest/api/certificates HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG…
```

Ответ.

```
HTTP/1.1 200 OK
Content-Length: 2826
Content-Type: application/json; charset=utf-8
Date: Tue, 10 May 2016 10:18:47 GMT

[{"CertificateBase64":"MII…","DName":"CN=bob","ID":2,"IsDefault":false,"Certific
ateAuthorityID":1,"CspID":null,"HashAlgorithms":null,"ProviderName":null,"Provid
erType":0,"CertificateType":1,"Status":{"Value":1,"RevocationInfo":null,"PinCode
":null,"ActiveCertId":0}}]
```

3.10.2. Получение сертификата с заданным id

Запрос.

```
GET http://cloudcsp/SignServer/rest/api/certificates/2 HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG…
```

Ответ.

```
HTTP/1.1 200 OK
Content-Length: 2826
Content-Type: application/json; charset=utf-8
Server: Microsoft-IIS/8.5
X-Powered-By: ASP.NET
Date: Tue, 10 May 2016 10:24:47 GMT

{"CertificateBase64":"MII…","DName":"CN=bob","ID":2,"IsDefault":false,"Certifica
teAuthorityID":1,"CspID":null,"HashAlgorithms":null,"ProviderName":null,"Provide
rType":0,"CertificateType":1,"Status":{"Value":1,"RevocationInfo":null,"PinCode"
:null,"ActiveCertId":0}}
```

3.10.3. Отправка документа на подпись

Запрос.

```
POST http://cloudcsp/SignServer/rest/api/documents HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG…
Content-Type: application/json; charset=utf-8
Content-Length: 146

{"Content":"AQID","Name":"Тестовый
документ","Signature":{"Type":1,"Parameters":{"Hash":"False"},"CertificateId":2,
"PinCode":"1"}}
```

Ответ.

```
HTTP/1.1 200 OK
Content-Length: 90
Content-Type: application/json; charset=utf-8
Date: Tue, 10 May 2016 10:18:47 GMT

"0a//ZlPvQr9+Y2RpTI7pBnD6KdbplVfpH7Ymthiz59oa3+uCwP3FpJGWtCTUa1ncwWeXhucdwzrRN+6
yC9ykKA=="
```

3.10.4. Получение политики СЭП

Запрос.

```
GET http://cloudcsp/SignServer/rest/api/policy HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG…
```

Ответ.

```
HTTP/1.1 200 OK
Content-Length: 4352
Content-Type: application/json; charset=utf-8
Date: Tue, 10 May 2016 10:31:17 GMT

{"CAPolicy":[{"ID":1,"Name":"Тестовый УЦ КРИПТО-
ПРО","Active":true,"AllowUserMode":false,"SNChangesEnable":true,"NamePolicy":[{"
IsRequired":false,"Order":24,"OID":"1.2.643.100.5","Name":"ОГРНИП","Value":null,
"StringIdentifier":"ОГРНИП"},{"IsRequired":false,"Order":23,"OID":"1.2.643.100.3
","Name":"СНИЛС","Value":null,"StringIdentifier":"СНИЛС"},{"IsRequired":false,"O
rder":21,"OID":"1.2.643.100.1","Name":"ОГРН","Value":null,"StringIdentifier":"ОГ
РН"},{"IsRequired":false,"Order":18,"OID":"1.2.643.3.131.1.1","Name":"ИНН","Valu
e":null,"StringIdentifier":"INN"},{"IsRequired":false,"Order":17,"OID":"1.2.840.
113549.1.9.1","Name":"Электронная
почта","Value":null,"StringIdentifier":"E"},{"IsRequired":false,"Order":16,"OID"
:"2.5.4.6","Name":"Страна/регион","Value":null,"StringIdentifier":"C"},{"IsRequi
red":false,"Order":15,"OID":"2.5.4.8","Name":"Область","Value":null,"StringIdent
ifier":"S"},{"IsRequired":false,"Order":14,"OID":"2.5.4.7","Name":"Город","Value
":null,"StringIdentifier":"L"},{"IsRequired":false,"Order":13,"OID":"2.5.4.10","
Name":"Организация","Value":null,"StringIdentifier":"O"},{"IsRequired":false,"Or
der":12,"OID":"2.5.4.11","Name":"Подразделение","Value":null,"StringIdentifier":
"OU"},{"IsRequired":true,"Order":11,"OID":"2.5.4.3","Name":"Общее
имя","Value":null,"StringIdentifier":"CN"},{"IsRequired":false,"Order":6,"OID":"
2.5.4.9","Name":"Адрес","Value":null,"StringIdentifier":"2.5.4.9"},{"IsRequired"
:false,"Order":5,"OID":"1.2.840.113549.1.9.2","Name":"Неструктурированное
имя","Value":null,"StringIdentifier":"1.2.840.113549.1.9.2"},{"IsRequired":false
,"Order":4,"OID":"2.5.4.12","Name":"Должность/звание","Value":null,"StringIdenti
fier":"T"},{"IsRequired":false,"Order":2,"OID":"2.5.4.42","Name":"Имя","Value":n
ull,"StringIdentifier":"G"},{"IsRequired":false,"Order":1,"OID":"2.5.4.4","Name"
:"Фамилия","Value":null,"StringIdentifier":"SN"}],"EKUTemplates":{"Сертификат
пользователя
УЦ":["1.3.6.1.5.5.7.3.4","1.2.643.2.2.34.6","1.3.6.1.5.5.7.3.2"],"Временный
сертификат Оператора
```

```
УЦ":["1.2.643.2.2.34.2","1.2.643.2.2.34.5","1.3.6.1.5.5.7.3.2"],"Временный
сертификат пользователя
УЦ":["1.2.643.2.2.34.2","1.2.643.2.2.34.6","1.3.6.1.5.5.7.3.2"],"Сертификат
ipsec":["1.3.6.1.5.5.8.2.2","1.3.6.1.5.5.7.3.2","1.3.6.1.5.5.7.3.1"],"Сертификат
аудитора журнала УЦ":["1.2.643.2.2.34.16","1.3.6.1.5.5.7.3.2"],"Сертификат входа
в домен по
УЭК":["1.3.6.1.4.1.311.20.2.2","1.2.643.2.2.34.6","1.3.6.1.5.5.7.3.2"],"Сертифик
ат входа со смарт-
картой":["1.2.643.2.2.34.6","1.3.6.1.4.1.311.20.2.2","1.3.6.1.5.5.7.3.2"],"Серти
фикат контроллера домена
(winlogon)":["1.2.643.2.2.34.24","1.3.6.1.5.5.7.3.1","1.3.6.1.5.5.7.3.2"],"Серти
фикат оператора службы OCSP":["1.3.6.1.5.5.7.3.9"],"Сертификат оператора службы
штампов времени":["1.3.6.1.5.5.7.3.8"],"Сертификат Оператора
УЦ":["1.2.643.2.2.34.5","1.3.6.1.5.5.7.3.2"],"Сертификат подписи (тест
УЭК)":["1.3.6.1.5.5.7.3.4","1.2.643.2.2.34.6","1.3.6.1.5.5.7.3.2"],"Сертификат
подписи с лицензией (тест
УЭК)":["1.3.6.1.5.5.7.3.4","1.2.643.2.2.34.31","1.2.643.2.2.34.30","1.2.643.2.2.
34.6","1.3.6.1.5.5.7.3.2","1.3.6.1.4.1.311.10.6.2"],"Сертификат пользователя
DSS":["1.2.643.2.2.34.6","1.3.6.1.5.5.7.3.2","1.3.6.1.4.1.311.10.6.2"],"Сертифик
ат сервера":["1.3.6.1.5.5.7.3.1"],"Совсем временный
сертификат":["1.3.6.1.5.5.7.3.4","1.2.643.2.2.34.6","1.3.6.1.5.5.7.3.2","1.3.6.1
.4.1.311.10.5.1"]},"CAType":0}],"CSPsPolicy":[{"ID":1,"ProviderName":"Crypto-Pro
GOST R 34.10-2001 Cryptographic Service
Provider","ProviderType":75,"KeyLength":512,"HashAlgorithms":null,"Alias":"GOST
R 34.11-94 + GOST R 34.10-2001
512"}],"ActionPolicy":null,"PinCodeMode":2,"TspServices":[{"Name":"TestTSP","Tit
le":"TestTSP","Url":"http://www.cryptopro.ru/tsp/tsp.srf"}],"TransactionConfirma
tion":0,"AllowedSignatureTypes":[1,5,2,0,4,3]}
```

4. Описание интерфейса SOAP для работы с сервисом управления пользователями ЦИ «КриптоПро DSS»

Сервис управления пользователями ЦИ DSS входит в состав компонента «Центр Идентификации» СЭП DSS и предназначен для управления учетными записями пользователей ЦИ DSS через SOAP-интерфейс.

Описание веб-сервиса доступно по адресам:

- http://<hostname>/<ApplicationName>/UserManagement.svc?wsdl,
- https://<hostname>/<ApplicationName>/UserManagement.svc?wsdl,

где <ApplicationName> – имя веб-приложения Центра Идентификации, по умолчанию имеет значение STS.

Описание интерфейса сервиса находится в библиотеке CryptoPro.DSS.Common.dll, имя интерфейса CryptoPro.DSS.Common.UserManagementService.IUserManagementService.

Интерфейс IUserManagementService предоставляет следующие методы:

- Функция RegisterUser;
- Функция AddExternalLogin;
- Функция GetUser;
- Функция GetUserById;
- Функция SetUserPhoneNumber;
- Функция ResetPassword;
- Функция SetUserSimAuthToken;
- Функция SetUserOtpToken;
- Функция AssignAuthenticationMethod;
- Функция RemoveAuthenticationMethod;
- Функция GetUserAuthenticationScheme;
- Функция DeleteUser;
- Функция SetUserGroup;
- Функция SetUserOperationPolicy;
- Функция GetUserOperationPolicy;
- Функция SetUserDistinguishName;
- Функция SetUserProperty;
- Функция GetUsers;
- Функция GetPolicy;
- Функция SetUserMobileAuthToken;
- Функция UpdateUserMobileAuthToken;
- Функция SetMobileAuthDeviceThumbprint;
- Функция DeleteUserMobileAuthToken;
- Функция RemoveUserAuthenticationData;
- Функция GetUserPhoneInfo;
- Функция GetUserEmailInfo;
- Функция GetUserOtpTokenInfo;
- Функция GetUserSimAuthTokenInfo;
- Функция GetUserMobileAuthTokenInfo.

Вызов методов Сервиса управления пользователями возможен через несколько конечных точек:

1. https://<hostname>/<ApplicationName>/UserManagement.svc/cert
2. http://<hostname>/<ApplicationName>/UserManagement.svc/token

При работе с конечной точкой UserManagement.svc/cert сообщение защищено на транспортном уровне с помощью установления защищённого HTTPS соединения с сервером. Конечные точки различаются способом аутентификации. Первая конечная точка использует клиентский сертификат при аутентификации, вторая – SAML-токен, полученный от одного из доверенных Центров Идентификации.

4.1. Функции сервиса

4.1.1. Функция RegisterUser

Функция **RegisterUser** используется для создания новой учетной записи пользователя в ЦИ DSS.

```
string RegisterUser(IDictionary<IdentifierType, string> identifiers, string
groupName = null);
```

Параметры функции:

- **identifiers –** словарь идентификаторов, которые будут назначены пользователю после создания его учетной записи. Типы идентификаторов указаны в перечислении IdentifierType.
- **groupName –** имя группы, в которую будет добавлен созданный пользователь.

Возвращаемое значение: строка с глобальным идентификатором (Guid) пользователя в БД ЦИ DSS. Данное значение используется в качестве идентификатора пользователя в остальных методах сервиса.

4.1.2. Функция AddExternalLogin

Функция **AddExtenalLogin** используется для назначения внешнего логина пользователю, который используется для аутентификации пользователей, обладающих учетной записью в стороннем доверенном Центре Идентификации.

```
void AddExternalLogin(string userId, string login, string idpName);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **login** – имя учетной записи пользователя на стороннем Центре Идентификации,
- **idpName –** имя доверенного издателя стороннего Центра Идентификации.

4.1.3. Функция GetUser

Функция **GetUser** используется для получения информации о пользователе. Поиск осуществляется по одному из идентификаторов пользователя.

```
DssUserInfo GetUser(IdentifierType type, string identifierValue);
```

Параметры функции:

- **type** – тип используемого для поиска идентификатор пользователя,
- **identifierValue** – значение используемого для поиска идентификатора.

Возвращаемое значение: структура типа **DssUserInfo**, содержащая информацию об учетной записи пользователя.

4.1.4. Функция GetUserById

Функция **GetUserById** используется для получения информации об учетной записи пользователя по его глобальному идентификатору.

```
DssUserInfo GetUserById(string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

Возвращаемое значение: структура типа **DssUserInfo**, содержащая информацию об учетной записи пользователя.

4.1.5. Функция SetUserPhoneNumber

Функция **SetUserPhoneNumber** используется для добавления к учетной записи пользователя информации о номере мобильного телефона.

```
void SetUserPhoneNumber(string userId, string phoneNumber);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **phoneNumber** – номер мобильного телефона пользователя.

4.1.6. Функция ResetPassword

Функция **ResetPassword** используется для сброса пароля пользователя. Позднее новый пароль может быть установлен пользователем самостоятельно.

```
string ResetPassword (string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

Возвращаемое значение: строка, содержащая новый пароль пользователя. Новый пароль генерируется случайно в соответствии с политикой паролей, выставленной на Центре Идентификации DSS.

4.1.7. Функция SetUserSimAuthToken

Функция **SetUserSimAuthToken** используется для добавления к учетной записи пользователя информации о методе аутентификации типа SimAuth.

```
void SetUserSimAuthToken(string userId, string iccId, string providerId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **iccId** – уникальный серийный номер SIM-карты (ICCID),
- **providerId –** идентификатор профиля мастер ключа шифрования. Может быть получен из политики Сервиса управления пользователями (см. метод Функция GetPolicy).

4.1.8. Функция SetUserOtpToken

Функция **SetUserOtpToken** используется для добавления к учетной записи пользователя информации о методе аутентификации типа OATH.

```
void SetUserOtpToken(string userId, string serial, string firstOtp, string
secondOtp);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **serial** – серийный номер OTP-токена,
- **firstOtp** – первый одноразовый пароль, сгенерированный OTP-токеном,
- **secondOtp** – второй одноразовый пароль, следующий непосредственно после первого одноразового пароля в цепочке одноразовых паролей, сгенерированных OTP-токеном. В случае, если в качестве токена выступает TOTP-токен, то в качестве второго пароля повторно передается первый пароль.

4.1.9. Функция AssignAuthenticationMethod

Функция **AssignAuthenticationMethod** используется для добавления метода аутентификации в схему аутентификации пользователя.

```
void AssignAuthenticationMethod (string userId, string methodUri, int level);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **methodUri** – идентификатор метода аутентификации. Список доступных методов аутентификации описан в таблице.
- **level** – шаг аутентификации, на который будет назначен запрошенный метод аутентификации.

Таблица 56. Список методов аутентификации, доступных в ЦИ DSS

Идентификатор метода	Описание метода	Шаг
http://schemas.microsoft.com/ws/2012/09/identity/authenticationmethod/none	От пользователя требуется только логин.	0
http://dss.cryptopro.ru/identity/authenticationmethod/password	Аутентификация по паролю, хранимому в БД ЦИ	0
http://dss.cryptopro.ru/identity/authenticationmethod/otpviasms	Аутентификация с использованием одноразовых паролей, отправляемых в SMS сообщении	1+
http://dss.cryptopro.ru/identity/authenticationmethod/otpviaemail	Аутентификация с использованием одноразовых паролей, отправляемых в сообщении электронной почты	1+
http://dss.cryptopro.ru/identity/authenticationmethod/oath	Аутентификация с помощью одноразовых паролей, полученных по протоколу OATH (TOTP и HOTP)	1+
http://dss.cryptopro.ru/identity/authenticationmethod/simauth	Аутентификация по ключу SIM-карты	1+
http://dss.cryptopro.ru/identity/authenticationmethod/mobile	Аутентификация в системе PayControl	1+

4.1.10. Функция RemoveAuthenticationMethod

Функция **RemoveAuthenticationMethod** используется для удаления метода аутентификации из схемы аутентификации пользователя.

```
void RemoveAuthenticationMethod (string userId, string methodUri);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,

- **methodUri** – идентификатор метода аутентификации. Список доступных методов аутентификации описан в пункте 3.

4.1.11. Функция GetUserAuthenticationScheme

Функция **GetUserAuthenticationScheme** используется для получения информации о схеме аутентификации пользователя.

```
IList<AuthenticationInfo> GetUserAuthenticationScheme (string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

Возвращаемое значение: список объектов AuthenticationInfo.

4.1.12. Функция DeleteUser

Функция **DeleteUser** используется для удаления учетной записи пользователя из БД ЦИ DSS.

```
void DeleteUser (string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

4.1.13. Функция SetUserGroup

Функция **SetUserGroup** для назначения пользователю группы пользователей, зарегистрированной в ЦИ DSS.

```
void SetUserGroup (string userId, string groupName);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **groupName** – имя группы, которую следует назначить пользователю. Список доступных групп может быть получен из политики сервиса.

4.1.14. Функция SetUserOperationPolicy

Функция **SetUserOperationPolicy** используется для назначения пользователю политики подтверждения операций, требующих доступ к закрытому ключу.

```
void SetUserOperationPolicy(string userId, IList<DSSActions> operations);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **operations** – список операций, для которых требуется подтверждение операции. Для списка всех доступных операций см. 2.7.13.

4.1.15. Функция GetUserOperationPolicy

Функция **GetUserOperationPolicy** используется для получения информации о политике подтверждения операций, требующих доступ к закрытому ключу пользователя.

```
IList<OperationPolicy> GetUserOperationPolicy(string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

Возвращаемое значение: список объектов OperationPolicy.

4.1.16. Функция SetUserDistinguishName

Функция **SetUserDistinguishName** используется для назначения пользователю различительного имени субъекта.

```
void SetUserDistinguishName(string userId, IDictionary<int, string> components);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **components** – словарь компонент различительного имени субъекта. В качестве ключа словаря используется идентификатор компоненты в БД ЦИ, который может быть получен из политики Сервиса управления пользователями (см. метод Функция GetPolicy).

4.1.17. Функция SetUserProperty

Функция **SetUserProperty** используется для изменения свойств учетной записи пользователя.

```
void SetUserProperty(string userId, UserProperty property);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **property** – объект UserProperty, описывающий тип и новое значение свойства учетной записи.

4.1.18. Функция GetUsers

Функция **GetUsers** используется для получения информации о пользователях ЦИ DSS. Данная функция позволяет осуществить поиск пользователей в БД с применением фильтров по различным полям учетной записи пользователя.

```
UserRecordsResponse GetUsers(UserRecordsRequest request);
```

Параметры функции:

- **request** – запрос на получение информации о пользователях, включает в себя список фильтров и требуемые параметры объема выборки (см. UserRecordsRequest).

Возвращаемое значение: объект типа UserRecordsResponse, содержащий в себе информацию об учетных записях пользователей и размер полученной выборки.

4.1.19. Функция GetPolicy

Функция **GetPolicy** используется для получения политики Сервиса управления пользователями ЦИ DSS.

```
UmsPolicy GetPolicy();
```

Возвращаемое значение: объект типа UmsPolicy.

4.1.20. Функция SetUserMobileAuthToken

Функция **SetUserMobileAuthToken** используется для назначения пользователю данных для аутентификации в системе MobileAuth.

```
MobileAuthCreateInfo SetUserMobileAuthToken(string userId, string
userContactInfo, string userContactInfoType);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,

- **userContactInfo** – контактная информация пользователя, используемая для передачи QR-кода,
- **userContactInfoType** – тип контактной информации (“PhoneNumber” или “EmailAddress”).

Возвращаемое значение: объект типа MobileAuthCreateInfo

4.1.21. Функция UpdateUserMobileAuthToken

Функция **UpdateUserMobileAuthToken** используется для обновления назначенных пользователю данных для аутентификации в системе MobileAuth.

```
MobileAuthUpdateInfo UpdateUserMobileAuthToken(string userId, string
userContactInfo, string userContactInfoType);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **userContactInfo** – контактная информация пользователя, используемая для передачи QR-кода,
- **userContactInfoType** – тип контактной информации (“PhoneNumber” или “EmailAddress”).

Возвращаемое значение: объект типа MobileAuthUpdateInfo

4.1.22. Функция SetMobileAuthDeviceThumbprint

Функция **SetMobileAuthDeviceThumbprint** используется для назначения отпечатка устройства пользователя в системе MobileAuth.

```
void SetMobileAuthDeviceThumbprint(string userId, string deviceThumbprint);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **deviceThumbprint** – отпечаток устройства пользователя.

4.1.23. Функция DeleteUserMobileAuthToken

Функция **DeleteUserMobileAuthToken** используется для удаления пользователя из внешней системы аутентификации MobileAuth.

```
void DeleteUserMobileAuthToken(string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

4.1.24. Функция RemoveUserAuthenticationData

Функция **RemoveUserAuthenticationData** используется для удаления данных аутентификации, соответствующих указанному методу аутентификации.

```
void RemoveUserAuthenticationData(string userId, string methodUri);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя,
- **methodUri** – идентификатор метода аутентификации (см. Таблица 56).

4.1.25. Функция GetUserPhoneInfo

Функция **GetUserPhoneInfo** используется для получения информации о номере мобильного телефона пользователя.

```
UserPhoneInfo GetUserPhoneInfo(string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

Возвращаемое значение: объект типа UserPhoneInfo

4.1.26. Функция GetUserEmailInfo

Функция **GetUserEmailInfo** используется для получения информации об адресе электронной почты пользователя.

```
UserEmailInfo GetUserEmailInfo(string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

Возвращаемое значение: объект типа UserEmailInfo

4.1.27. Функция GetUserOtpTokenInfo

Функция **GetUserOtpTokenInfo** используется для получения информации о назначенном пользователю OTP-токене.

```
UserOtpTokenInfo GetUserOtpTokenInfo(string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

Возвращаемое значение: объект типа UserOtpTokenInfo

4.1.28. Функция GetUserSimAuthInfo

Функция **GetUserSimAuthInfo** используется для получения информации о назначенном пользователю Sim-карте для аутентификации по протоколу SimAuth.

```
UserSimAuthInfo GetUserSimAuthInfo(string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

Возвращаемое значение: объект типа UserSimAuthInfo

4.1.29. Функция GetUserMobileAuthInfo

Функция **GetUserOtpTokenInfo** используется для получения данных аутентификации пользователя в системе MobileAuth

```
UserMobileAuthInfo GetUserMobileAuthInfo(string userId);
```

Параметры функции:

- **userId** – глобальный идентификатор пользователя.

Возвращаемое значение: объект типа UserMobileAuthInfo

4.1.30. Функция GetAuthnTokens

Функция **GetAuthnTokens** используется для получения информации о средствах аутентификации пользователей ЦИ DSS. Данная функция позволяет осуществить поиск средств аутентификации с применением фильтров по различным параметрам средства аутентификации.

```
TokenRecordsResponse GetAuthnTokens(TokenRecordsRequest request);
```

Параметры функции:

- **request** – запрос на получение информации о средствах аутентификации, включает в себя список фильтров и требуемые параметры объема выборки (см. TokenRecordsRequest).

Возвращаемое значение: объект типа TokenRecordsResponse, содержащий в себе информацию об учетных записях пользователей и размер полученной выборки.

4.2. Типы данных, используемых сервисом.

4.2.1. UmsPolicy

Таблица 57. Описание полей класса UmsPolicy

Поле	Тип	Описание
AvaliableIdentifierTypes	IList<IdentifierType>	Содержит список типов идентификаторов, доступных для использования при регистрации учетной записи пользователя
AuthenticationPolicy	AuthenticationPolicy	Содержит политику аутентификации ЦИ DSS
AllowUserRegistration	bool	Показывает, разрешена ли самостоятельная регистрация пользователей в ЦИ DSS
Group	IList<IdentityGroupInfo>	Содержит список имен групп, зарегистрированных на данном ЦИ
Rdns	IList<RdnInfo>	Содержит информацию о компонентах различительного имени субъекта, зарегистрированных в БД ЦИ DSS
IdentityProviders	IList<IdentityProviderInfo>	Содержит информацию о сторонних Центрах Идентификации, зарегистрированных в ЦИ DSS
CryptoProviders	IList<CryptoProviderInfo>	Содержит информацию о криптографических провайдерах, зарегистрированных в ЦИ DSS

4.2.2. AuthenticationPolicy

Таблица 58. Описание полей класса AuthenticationPolicy

Поле	Тип	Описание
Mode	MfaPolicy	Глобальная политика подтверждения операций, требующих доступа к закрытому ключу пользователя.
EditableByUser	bool	Значение, показывающее, позволено ли пользователю редактировать свою политику подтверждения операций
AuthMethods	IList<AuthnMethodDescription>	Список доступных методов аутентификации
OperationPolicy	IList<OperationPolicy>	Политика подтверждения операций по умолчанию.

4.2.3. IdentityGroupInfo

Таблица 59. Описание полей класса IdentityGroupInfo

Поле	Тип	Описание
IdentityProviderName	string	Имя Центра Идентификации
GroupList	IList<string>	Список имен групп, зарегистрированных на данном Центре Идентификации

4.2.4. RdnInfo

Таблица 60. Описание полей класса RdnInfo

Поле	Тип	Описание
Oid	string	Объектный идентификатор компонента имени
DisplayName	string	Отображаемое имя компонента имени
StringIdentifier	string	Строковый идентификатор компонента имени
Order	int	Порядок компонента имени
MinLength	int	Минимальная длина значения компонента имени
MaxLength	int	Максимальная длина значения компонента имени
Required	bool	Значение, показывающее, является ли данный компонент обязательным
DefaultValue	string	Значение по умолчанию для компонента

4.2.5. IdentityProviderInfo

Таблица 61. Описание полей класса IdentityProviderInfo

Поле	Тип	Описание
Description	string	Описание Центра Идентификации
IssuerName	string	Имя доверенного издателя Центра Идентификации
DisplayName	string	Отображаемое имя Центра Идентификации

4.2.6. CryptoProviderInfo

Таблица 62. Описание полей класса CryptoProviderInfo

Поле	Тип	Описание
Id	Guid	Идентификатор профиля провайдера
Name	String	Имя профиля провайдера
Description	String	Описание профиля провайдера
Type	CryptoProviderProfileType	Тип профиля провайдера

4.2.7. AuthnMethodDescription

Таблица 63. Описание полей класса AuthnMethodDescription

Поле	Тип	Описание
Identifier	String	Идентификатор метода аутентификации
Type	AuthnLevel	Тип метода аутентификации

4.2.8. OperationPolicy

Таблица 64. Описание полей класса OperationPolicy

Поле	Тип	Описание
Action	DSSActions	Тип операции
ConfirmationRequired	bool	Значение, показывающее, требуется ли подтверждение операции

4.2.9. UserProperty

Таблица 65. Описание полей класса UserProperty

Поле	Тип	Описание
PropertyType	UserPropertyType	Тип свойства учетной записи
Value	bool string	Новое значение свойства учетной записи. Может быть объектом одного из двух типов в зависимости от значения поля PropertyType.

4.2.10. AuthenticationInfo

Таблица 66. Описание полей класса AuthenticationInfo

Поле	Тип	Описание
MethodUri	string	Идентификатор метода аутентификации
Level	int	Уровень, на котором применяется данный метод аутентификации.

4.2.11. DssUserInfo

Таблица 67. Описание полей класса DssUserInfo

Поле	Тип	Описание
UserId	string	Глобальный идентификатор пользователя
Login	string	Логин пользователя
PhoneNumber	string	Номер мобильного телефона пользователя
Email	string	Адрес электронной почты пользователя
PhoneConfirmed	bool	Значение, показывающее, подтвержден ли номер телефона пользователя
EmailConfirmed	bool	Значение, показывающее, подтвержден ли адрес электронной почты пользователя
DisplayName	string	Отображаемое имя пользователя
DistinguishName	string	Различительное имя пользователя
AccountLocked	bool	Значение, показывающее, является ли запись заблокированной
Group	string	Имя группы, в которой состоит пользователь

Поле	Тип	Описание
CreationDate	DateTime	Дата создание учетной записи пользователя
LockoutDate	DateTime	Дата снятия блокировки с учетной записи пользователя
LastLoginDate	DateTime	Дата последнего удачного входа пользователя

4.2.12. UserRecordsRequest

Таблица 68. Описание полей класса UserRecordsRequest

Поле	Тип	Описание
StartPosition	int	Стартовая позиция выборки
EndPosition	int	Конечная позиция выборки
Filters	IList<UserFilter>	Список фильтров, по которым осуществляется выбор пользователя

4.2.13. UserFilter

Таблица 69. Описание полей класса UserFilter

Поле	Тип	Описание
Column	ColumnTypeEnum	Тип колонки, к которой применяется фильтр
Operation	FilterOperationEnum	Операция, используемая в фильтре
Value	object	Значение фильтра

4.2.14. UserRecordsResponse

Таблица 70. Описание полей класса UserRecordsResponse

Поле	Тип	Описание
UserInfos	IList<DssUserInfo>	Список учетных записей, соответствующих заданным фильтрам
TotalCount	int	Общее количество записей в БД
AffectedCount	int	Количество отфильтрованных записей

4.2.15. MobileAuthCreateInfo

Таблица 71. Описание полей класса MobileAuthCreateInfo

Поле	Тип	Описание
ExternalUserId	string	Идентификатор пользователя во внешней системе
QrCode	string	Base64-строка, содержащая QR-код для подтверждения
KeyExpirationTime	DateTime	Время истечения ключа пользователя

4.2.16. MobileAuthUpdateInfo

Таблица 72. Описание полей класса MobileAuthUpdateInfo

Поле	Тип	Описание
QrCode	string	Base64-строка, содержащая QR-код для подтверждения
KeyExpirationTime	DateTime	Время истечения ключа пользователя

4.2.17. MobileAuthCreateInfoEx

Таблица 73. Описание полей класса MobileAuthCreateInfoEx

Поле	Тип	Описание
ExternalUserId	string	Идентификатор пользователя во внешней системе
QrCode	string	Base64-строка, содержащая QR-код для подтверждения
KeyExpirationTime	DateTime	Время истечения ключа пользователя
XmlKeyInfo	string	Вектор аутентификации в формате XML

4.2.18. MobileAuthUpdateInfoEx

Таблица 74. Описание полей класса MobileAuthUpdateInfoEx

Поле	Тип	Описание
QrCode	string	Base64-строка, содержащая QR-код для подтверждения
KeyExpirationTime	DateTime	Время истечения вектора аутентификации пользователя
XmlKeyInfo	string	Вектор аутентификации в формате XML

4.2.19. UserPhoneInfo

Таблица 75. Описание полей класса UserPhoneInfo

Поле	Тип	Описание
PhoneNumber	string	Номер мобильного телефона пользователя
Confirmed	bool	Значение, показывающее, является ли номер телефона подтвержденным

4.2.20. UserEmailInfo

Таблица 76. Описание полей класса UserEmailInfo

Поле	Тип	Описание
Email	string	Адрес электронной почты пользователя
Confirmed	bool	Значение, показывающее, является ли адрес электронной почты подтвержденным

4.2.21. UserOtpTokenInfo

Таблица 77. Описание полей класса UserOtpTokenInfo

Поле	Тип	Описание
Serial	string	Серийный номер назначенного OTP-токена
Type	string	Тип назначенного OTP-токена

4.2.22. UserSimAuthInfo

Таблица 78. Описание полей класса UserSimAuthInfo

Поле	Тип	Описание
IccId	string	Идентификатор SIM-карты пользователя
ProfieId	string	Идентификатор криптопровайдера
PhoneNumber	string	Номер мобильного телефона

4.2.23. UserMobileAuthInfo

Таблица 79. Описание полей класса UserSimAuthInfo

Поле	Тип	Описание
UserId	string	Идентификатор пользователя в системе PayControl
KeyExpirationTime	DateTime	Время истечения пользовательского ключа

4.2.24. DssTokenInfo

Таблица 80. Описание полей класса DssTokenInfo

Поле	Тип	Описание
Id	long	Идентификатор средства аутентификации
Serial	string	Серийный номер средства аутентификации
UserName	string	Имя пользователя, для которого назначено данное средство аутентификации
TokenType	string	Тип средства аутентификации. Принимает одно из следующих значений: HOTP, TOTP, SimAuth, MobileAuth, AirKeyAuth, RutokenAuth, JctokenAuth.
Parameters	Dictionary<string, string>	Основные параметры средства аутентификации

4.2.25. TokenRecordsRequest

Таблица 81. Описание полей класса TokenRecordsRequest

Поле	Тип	Описание
StartPosition	int	Стартовая позиция выборки
EndPosition	int	Конечная позиция выборки
Filters	IList<TokenFilter>	Список фильтров, по которым осуществляется выбор средств аутентификации

4.2.26. TokenFilter

Таблица 82. Описание полей класса TokenFilter

Поле	Тип	Описание
Column	TokenColumnTypeEnum	Тип колонки, к которой применяется фильтр

Поле	Тип	Описание
Operation	FilterOperationEnum	Операция, используемая в фильтре
Value	object	Значение фильтра

4.2.27. TokenRecordsResponse

Таблица 83. Описание полей класса TokenRecordsResponse

Поле	Тип	Описание
TokenInfos	IList<DssTokenInfo>	Список учетных записей, соответствующих заданным фильтрам
TotalCount	int	Общее количество средств аутентификации
AffectedCount	int	Количество отфильтрованных записей

4.3. Перечислимые типы данных, используемые сервисом.

4.3.1. IdentifierType

Перечисление **IdentifierType** содержит типы идентификаторов, которые могут быть назначены пользователю при регистрации учетной записи.

Таблица 84. Перечисление IdentifierType

Наименование	Описание
Login	В качестве идентификатора используется логин пользователя
PhoneNumber	В качестве идентификатора используется номер мобильного телефона пользователя
Email	В качестве идентификатора используется адрес электронной почты пользователя

4.3.2. UserPropertyType

Перечисление **UserPropertyType** содержит типы свойств, которые могут быть изменены для учетной записи пользователя.

Таблица 85. Перечисление UserPropertyType

Наименование	Описание
DisplayName	Отображаемое имя пользователя
AccountState	Состояние учетной записи пользователя (заблокировано/разблокировано)

4.3.3. CryptoProviderProfileType

Перечисление **CryptoProviderProfileType** содержит типы профилей криптографических провайдеров, зарегистрированных в БД ЦИ DSS.

Таблица 86. CryptoProviderProfileType

Наименование	Описание
GOST	Провайдером используются алгоритмы семейства ГОСТ
RSA_AES	Провайдером используются связка алгоритмов RSA+AES.

4.3.4. AuthnLevel

Перечисление **AuthnLevel** содержит типы уровней аутентификации, на которые может быть назначен тот или иной метод аутентификации.

Таблица 87. Перечисление AuthnLevel

Наименование	Описание
Primary	Метод может быть назначен в качестве метода первичной аутентификации
Secondary	Метод может быть назначен в качестве метода вторичной аутентификации
PrimaryAndSecondary	Метод может быть назначен как в качестве метода первичной аутентификации, так и в качестве метода вторичной аутентификации

4.3.5. ColumnTypeEnum

Перечисление **ColumnTypeEnum** содержит типы колонок профиля пользователя, к которым может быть применен фильтр.

Таблица 88. Перечисление ColumnTypeEnum

Наименование	Описание
Login	Логин пользователя
PhoneNumber	Номер мобильного телефона пользователя
Email	Адрес электронной почты
CreateDate	Дата создания учетной записи
GroupId	Идентификатор группы пользователя

4.3.6. FilterOperationEnum

Перечисление **FilterOperationEnum** содержит типы операций для фильтрации.

Таблица 89. Перечисление FilterOperationEnum

Наименование	Описание
Equal	Операция равенства
NotEqual	Операция неравенства
Like	Операция подобия
Greater	Операция «больше чем»
Less	Операция «меньше чем»

4.3.7. TokenColumnTypeEnum

Перечисление **TokenColumnTypeEnum** содержит типы колонок параметров средства аутентификации, к которым может быть применен фильтр.

Таблица 90. Перечисление TokenColumnTypeEnum

Наименование	Описание
Id	Идентификатор средства аутентификации
Serial	Серийный номер средства аутентификации
UserId	Идентификатор пользователя, которому назначено данное средство аутентификации
TokenType	Тип средства аутентификации

4.4. Примеры использования сервиса управления пользователями

4.4.1. Добавление пользователя с аутентификацией по паролю

```
IUserManagementService umsChannel; // WCF-канал к сервису

// получение политики сервиса
var policy = umsChannel.GetPolicy();

// проверка того, что Логин является доступным идентификатором
if (!policy.AvailableIdentifiers.Contrains(IdentifierType.Login))
{
      return;
}

// создание списка идентификаторов пользователя
var ids = new Dictionary<IdentifierType, string>();
ids.Add(IdentifierType.Login, “SampleLogin”);

// регистрация пользователя
var userId = umsChannel.RegisterUser(ids):

// получение идентификатора метода аутентификации по паролю из политики сервиса
var methodUri = policy. AuthenticationPolicy.AuthMethods.First(x =>
x.Identifier.Contains("password")).Identifier;

var step = 1; // аутентификация по паролю является первичной

// сброс пароля пользователю (оператор не может назначить пароль напрямую)
var password = umsChannel.ResetPassword(userId);

// назначение метода аутентификации пользователю
umsChannel.AssignAuthenticationMethod(userId, methodUri, step);
```

4.4.2. Добавление пользователя с аутентификацией при помощи одноразовых паролей по SMS.

```
IUserManagementService umsChannel; // WCF-канал к сервису

// получение политики сервиса
var policy = umsChannel.GetPolicy();

// проверка того, что Логин является доступным идентификатором
if (!policy.AvailableIdentifiers.Contrains(IdentifierType.Login))
{
      return;
}

// создание списка идентификаторов пользователя
var ids = new Dictionary<IdentifierType, string>();
ids.Add(IdentifierType.Login, “SampleLogin”);

// регистрация пользователя
var userId = umsChannel.RegisterUser(ids):

// получение идентификатора метода аутентификации без пароля из политики сервиса
var methodUriNone = policy. AuthenticationPolicy.AuthMethods.First(x =>
x.Identifier.Contains("none")).Identifier;
```

```
// получение идентификатора метода аутентификации с при помощи одноразовых
паролей по SMS
var methodUriSms = policy. AuthenticationPolicy.AuthMethods.First(x =>
x.Identifier.Contains("otpviasms")).Identifier;

var stepNone = 1; // безпарольная аутентификация является первичной
var stepSms = 2; // аутентификация через SMS является вторичной

// назначение номера телефона пользователю
var phoneNumber = "88000000000";
umsChannel.SetUserPhoneNumber(userId, phoneNumber);

// назначение методов аутентификации пользователю
umsChannel.AssignAuthenticationMethod(userId, methodUriNone, stepNone);
umsChannel.AssignAuthenticationMethod(userId, methodUriSms, stepSms);
```

4.4.3. Добавление пользователя с аутентификацией по протоколу OATH

```
IUserManagementService umsChannel; // WCF-канал к сервису

// получение политики сервиса
var policy = umsChannel.GetPolicy();

// проверка того, что Логин является доступным идентификатором
if (!policy.AvailableIdentifiers.Contrains(IdentifierType.Login))
{
      return;
}

// создание списка идентификаторов пользователя
var ids = new Dictionary<IdentifierType, string>();
ids.Add(IdentifierType.Login, "SampleLogin");

// регистрация пользователя
var userId = umsChannel.RegisterUser(ids):

// получение идентификатора метода аутентификации без пароля из политики сервиса
var methodUriNone = policy. AuthenticationPolicy.AuthMethods.First(x =>
x.Identifier.Contains("none")).Identifier;

// получение идентификатора метода аутентификации по протоколу OATH
var methodUriOath = policy. AuthenticationPolicy.AuthMethods.First(x =>
x.Identifier.Contains("oath")).Identifier;

var stepNone = 1; // безпарольная аутентификация является первичной
var stepOath = 2; // аутентификация по протоколу OATH является вторичной

// назначение OTP-токена пользователю
var serialNumber = "A000001"; // серийный номер токена
var firstOtp = ""; // первый пароль, полученный на токене
var secondOtp = ""; // второй пароль, полученный на токене
umsChannel.SetUserOtpToken(userId, firstOtp, secondOtp);

// назначение методов аутентификации пользователю
umsChannel.AssignAuthenticationMethod(userId, methodUriNone, stepNone);
umsChannel.AssignAuthenticationMethod(userId, methodUriOath, stepOath);
```

4.4.4. Добавление пользователя с аутентификацией по протоколу SimAuth

```
IUserManagementService umsChannel; // WCF-канал к сервису

// получение политики сервиса
var policy = umsChannel.GetPolicy();

// проверка того, что Логин является доступным идентификатором
if (!policy.AvailableIdentifiers.Contrains(IdentifierType.Login))
{
      return;
}

// создание списка идентификаторов пользователя
var ids = new Dictionary<IdentifierType, string>();
ids.Add(IdentifierType.Login, “SampleLogin”);

// регистрация пользователя
var userId = umsChannel.RegisterUser(ids):

// получение идентификатора метода аутентификации без пароля из политики сервиса
var methodUriNone = policy. AuthenticationPolicy.AuthMethods.First(x =>
x.Identifier.Contains("none")).Identifier;

// получение идентификатора метода аутентификации по протоколу SimAuth
var methodUriSimAuth = policy. AuthenticationPolicy.AuthMethods.First(x =>
x.Identifier.Contains("simauth")).Identifier;

var stepNone = 1; // безпарольная аутентификация является первичной
var stepSimAuth = 2; // аутентификация по протоколу SimAuth является вторичной

// получение идентификатора профиля криптопровайдера
var profileId = policy.CryptoProviders.First(x => x.Type ==
CryptoProviderProfileType.GOST).Id;
var iccId = “12345678901234567 89”

// назначение параметров sim-карты пользователю
umsChannel.SetUsetSimAuthToken(userId, iccId, profileId);

// назначение методов аутентификации пользователю
umsChannel.AssignAuthenticationMethod(userId, methodUriNone, stepNone);
umsChannel.AssignAuthenticationMethod(userId, methodUriSimAuth, stepSimAuth);
```

4.4.5. Назначение пользователю политики подтверждения операций, требующих доступ к закрытому ключу

```
IUserManagementService umsChannel; // WCF-канал к сервису

// получение политики сервиса
var policy = umsChannel.GetPolicy();

// получение идентификатора пользователя
var user = umsChannel.GetUser(IdentifierType.Login, “SampleLogin”);

// создание списка операций, требующих подтверждения
var actionList = new List<DSSActions>();
actionList.Add(DSSActions.SignDocument);
actionList.Add(DSSActions.CreateRequest);

// назначение политики пользователю
umsChannel.SetUserOperationPolicy(user.Id, actionList);
```

4.4.6. Добавление пользователя в группу

```
IUserManagementService umsChannel; // WCF-канал к сервису

// получение политики сервиса
var policy = umsChannel.GetPolicy();

// получение идентификатора пользователя
var user = umsChannel.GetUser(IdentifierType.Login, “SampleLogin”);

var sampleIdp = ‘SampleSts”;

// получение списка групп
var groups = policy.Groups.First(x => x.IdentityProviderName == sampleIdp);

// получение имени группы
var groupName = groups.First();

// назначение группы пользователю
umsChannel.SetUserGroup(user.Id, groupName);
```

4.4.7. Назначение различительного имени пользователя

```
IUserManagementService umsChannel; // WCF-канал к сервису

// получение политики сервиса
var policy = umsChannel.GetPolicy();

// получение идентификатора пользователя
var user = umsChannel.GetUser(IdentifierType.Login, “SampleLogin”);

// получение идентификаторов компонент имени из политики сервиса
var commonNameId = policy.Rnds.First(x => x.StringIdentifier == “CN”).Id;
var orgId = policy.Rdns.First(x=>x.StringIdentifier == “O”).Id;

// создание словаря компонент различительно имени пользователя
var userRdns = Dictionary<int, string>();
userRdns.Add(commonNameId, “SampleCN”);
userRdns.Add(orgId, “SampleOrg”);
// назначение различительног имени пользователю
umsChannel.SetUserDistinguishName(user.Id, userRdns);
```

4.4.8. Управление статусом учетной записи пользователя (блокировка, разблокировка)

```
IUserManagementService umsChannel; // WCF-канал к сервису

// получение политики сервиса
var policy = umsChannel.GetPolicy();

// получение идентификатора пользователя
var user = umsChannel.GetUser(IdentifierType.Login, “SampleLogin”);

// создание UserProperty для блокировки учетной записи
var property = new UserProperty { PropertyType =
UserPropertyType.AccountLockedState, Value = true };

// применение свойства к пользователю
umsChannel.SetUserProperty(property);
```

4.4.9. Пример осуществления выборки пользователей с применением ряда фильтров

```
IUserManagementService umsChannel; // WCF-канал к сервису

// получение политики сервиса
var policy = umsChannel.GetPolicy();

// создание списка фильтров
var filterList = new List<UserFilter>();

// фильтр для выбора пользователей с логином, включающим в себя в качестве
подстроки указанное значение
var loginFilter = new UserFilter { Column = ColumnTypeEnum.Login, Operation =
FilterOperationEnum.Like, Value = "%Sample%"};

// фильтр для выбора пользователей, у которых не подтвержден номер мобильного
телефона
var phoneConfirmedFilter = new UserFilter { Column =
ColumnTypeEnum.PhoneNumberComfirmed, Operation = FilterOperationEnum.Equal,
Value = false };

// добавление фильтров в список
filterList.Add(loginFilter);
filterList.Add(phonConfirmedFilter);

// создание запроса на получение выборки пользователей
var request = new UserRecordsRequest();
request.StartPosition = 1; // начальная позиция в выборке
request.EndPosition = 100; // конечная позиция в выборке
request.Filters = filterList;

// получение ответа от сервиса
var response = umsChannel.GetUsers(request);

var userInfos = response.UserInfos; // список с учетными записями
var totalCount = response.TotalCount; // общее количество записей
var affected = response.AffectedCount; // количество записей в выборке
```

4.4.10. Пример осуществления выборки средств аутентификации с применением ряда фильтров

```
IUserManagementService umsChannel; // WCF-канал к сервису

// получение политики сервиса
var policy = umsChannel.GetPolicy();

// создание списка фильтров
var filterList = new List<TokenFilter>();

// фильтр для выбора средств аутентификации с серийным номером, включающим в
себя в качестве подстроки указанное значение
var serialFilter = new TokenFilter { Column = TokenColumnTypeEnum.Serial,
Operation = FilterOperationEnum.Like, Value = "%001%" };

// фильтр для выбора средств аутентификации определенного типа
var tokenTypeFilter = new TokenFilter { Column = TokenColumnTypeEnum.TokenType,
Operation = FilterOperationEnum.Equal, Value = "HOTP" };

// добавление фильтров в список
filterList.Add(serialFilter);
filterList.Add(tokenTypeFilter);
```

```
// создание запроса на получение выборки средств аутентификации
var request = new TokenRecordsRequest();
request.StartPosition = 1; // начальная позиция в выборке
request.EndPosition = 100; // конечная позиция в выборке
request.Filters = filterList;

// получение ответа от сервиса
var response = umsChannel.GetAuthnTokens(request);

var tokenInfos = response.TokenInfos; // список средств аутентификации
var totalCount = response.TotalCount; // общее количество записей
var affected = response.AffectedCount; // количество записей в выборке
```

5. Описание интерфейса REST для работы с Сервисом управления пользователями ЦИ «КриптоПро DSS»

REST интерфейс предоставляет конечные точки для доступа к ресурсам Сервиса управления пользователями:

- Создание/редактирование/удаление пользователя.
- Назначение аутентификационной информации пользователя.
- Назначение методов аутентификации пользователю.
- Назначение политики доступа к операциям СЭП для пользователя.

В рамках REST API взаимодействие с Сервисом управления пользователями осуществляется посредством HTTP-запросов.

Общий префикс для всех конечных точек сервиса:

https://<host>/<AppName>/ums, где

- host – адрес сервера, на котором расположен экземпляр приложения СЭП,
- AppName – название веб-приложения СЭП.

5.1. Формат входных и возвращаемых значений

Входные параметры могут передаваться внутри URL-адреса запроса и в теле самого запроса (только для HTTP методов POST и PUT). Внутри URL-адреса могут передаваться только примитивные типы (строки (string), числа (int)).

Все объекты передаются и возвращаются в формате JSON, списки возвращаются в виде массивов JSON объектов. Массивы байтов передаются в виде строк в кодировке BASE64.

5.2. Конечная точка Policy

Данная конечная точка позволяет получить доступ к политике сервиса управления пользователями ЦИ «КриптоПро DSS».

HTTP метод	Путь	Описание
GET	/policy	Получение политики сервиса
Параметры	Тип	Описание
Параметры не требуются		
Возвращаемое значение	Тип	Описание
	UmsPolicy	Политика сервиса управления пользователями

Пример выполнения метода:

Запрос:

```
GET https://grand-pc/STS/ums/policy/ HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json; charset=utf-8
Date: Thu, 29 Dec 2016 14:29:24 GMT
Content-Length: 4375

{"AvaliableIdentifierTypes":[0,2,1],"AuthenticationPolicy":{"Mode":0,"EditableBy
User":false,"AuthMethods":[{"Identifier":"http://schemas.microsoft.com/ws/2012/0
9/identity/authenticationmethod/none","Type":1},{"Identifier":"http://dss.crypto
pro.ru/identity/authenticationmethod/certificate","Type":1},{"Identifier":"http:
//dss.cryptopro.ru/identity/authenticationmethod/password","Type":1},{"Identifie
r":"http://dss.cryptopro.ru/identity/authenticationmethod/saml","Type":1},{"Iden
tifier":"http://dss.cryptopro.ru/identity/authenticationmethod/otpviasms","Type"
:2},{"Identifier":"http://dss.cryptopro.ru/identity/authenticationmethod/oath","
Type":2},{"Identifier":"http://dss.cryptopro.ru/identity/authenticationmethod/si
mauth","Type":2},{"Identifier":"http://dss.cryptopro.ru/identity/authenticationm
ethod/otpviaemail","Type":2},{"Identifier":"http://dss.cryptopro.ru/identity/aut
henticationmethod/mobile","Type":2},{"Identifier":"http://dss.cryptopro.ru/ident
ity/authenticationmethod/mtmo","Type":2}],"OperationPolicy":[{"Action":1,"Confir
mationRequired":false},{"Action":2,"ConfirmationRequired":false},{"Action":4,"Co
nfirmationRequired":false},{"Action":8,"ConfirmationRequired":false},{"Action":1
6,"ConfirmationRequired":false},{"Action":32,"ConfirmationRequired":false},{"Act
ion":64,"ConfirmationRequired":false},{"Action":128,"ConfirmationRequired":false
},{"Action":256,"ConfirmationRequired":false},{"Action":512,"ConfirmationRequire
d":false},{"Action":1024,"ConfirmationRequired":false}]},"AllowUserRegistration"
:true,"Groups":[{"IdentityProviderName":"realsts","GroupList":["Default"]}],"Rdn
s":[{"Id":1,"Oid":"1.2.643.100.1","DisplayName":"ОГРН","StringIdentifier":"OGRN"
,"Order":16,"MinLength":13,"MaxLength":13,"Required":false,"DefaultValue":null},
{"Id":2,"Oid":"1.2.643.100.5","DisplayName":"ОГРНИП","StringIdentifier":"OGRNIP"
,"Order":15,"MinLength":15,"MaxLength":15,"Required":false,"DefaultValue":null},
{"Id":3,"Oid":"1.2.643.100.3","DisplayName":"СНИЛС","StringIdentifier":"SNILS","
Order":14,"MinLength":11,"MaxLength":11,"Required":false,"DefaultValue":null},{"
Id":4,"Oid":"1.2.643.3.131.1.1","DisplayName":"ИНН","StringIdentifier":"INN","Or
der":13,"MinLength":12,"MaxLength":12,"Required":false,"DefaultValue":null},{"Id
":5,"Oid":"1.2.840.113549.1.9.1","DisplayName":"Электронная
почта","StringIdentifier":"E","Order":12,"MinLength":0,"MaxLength":128,"Required
":false,"DefaultValue":null},{"Id":6,"Oid":"2.5.4.6","DisplayName":"Страна","Str
ingIdentifier":"C","Order":11,"MinLength":0,"MaxLength":2,"Required":false,"Defa
ultValue":null},{"Id":7,"Oid":"2.5.4.8","DisplayName":"Область","StringIdentifie
r":"S","Order":10,"MinLength":0,"MaxLength":128,"Required":false,"DefaultValue":
null},{"Id":8,"Oid":"2.5.4.7","DisplayName":"Город","StringIdentifier":"L","Orde
r":9,"MinLength":0,"MaxLength":128,"Required":false,"DefaultValue":null},{"Id":9
,"Oid":"2.5.4.10","DisplayName":"Организация","StringIdentifier":"O","Order":8,"
MinLength":0,"MaxLength":64,"Required":false,"DefaultValue":null},{"Id":10,"Oid"
:"2.5.4.11","DisplayName":"Подразделение","StringIdentifier":"OU","Order":7,"Min
Length":0,"MaxLength":64,"Required":false,"DefaultValue":null},{"Id":11,"Oid":"2
.5.4.3","DisplayName":"Общее
имя","StringIdentifier":"CN","Order":6,"MinLength":0,"MaxLength":128,"Required":
true,"DefaultValue":null},{"Id":12,"Oid":"2.5.4.9","DisplayName":"Адрес","String
Identifier":"Street","Order":5,"MinLength":0,"MaxLength":30,"Required":false,"De
faultValue":null},{"Id":13,"Oid":"2.5.4.12","DisplayName":"Должность","StringIde
ntifier":"T","Order":4,"MinLength":0,"MaxLength":64,"Required":false,"DefaultVal
ue":null},{"Id":14,"Oid":"2.5.4.43","DisplayName":"Инициалы","StringIdentifier":
"I","Order":3,"MinLength":0,"MaxLength":5,"Required":false,"DefaultValue":null},
{"Id":15,"Oid":"2.5.4.42","DisplayName":"Имя","StringIdentifier":"G","Order":2,"
MinLength":0,"MaxLength":16,"Required":false,"DefaultValue":null},{"Id":16,"Oid"
:"2.5.4.4","DisplayName":"Фамилия","StringIdentifier":"SN","Order":1,"MinLength"
:0,"MaxLength":40,"Required":false,"DefaultValue":null}],"IdentityProviders":[{"
Description":null,"IssuerName":"realsts","DisplayName":null}],"CryptoProviders":
[],"MobileAuthSettings":{"DeviceFingerprintRequired":false,"KeyInfoDivideRequire
d":false}}
Expires: -1
Date: Tue, 20 Dec 2016 13:36:37 GMT
```

5.3. Конечная точка User

Данная конечная точка предоставляет доступ к управлению пользователями ЦИ «КриптоПро DSS». Перечень методов, реализуемых конечной точкой **User** представлен ниже.

5.3.1. Добавление пользователя

HTTP метод	Путь	Описание
POST	/user	Добавление пользователя в БД
Параметры	Тип	Описание
identifiers	Dictionary<IdentifierType, string>	Список идентификаторов пользователя
Возвращаемое значение	Тип	Описание
	string	Внутренний идентификатор пользователя

Данная функция регистрирует пользователя в ту же группу безопасности, в которой состоит регистрирующий пользователя оператор. Если оператор состоит в нескольких группах, и необходимо конкретно указать в какую группу должен быть добавлен пользователь, для регистрации пользователя следует воспользоваться функцией «Добавление пользователя с указанием имени группы».

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 74
{"Login":"RestTestUser","Email":"test@cp.ru","PhoneNumber":"+79150510528"}
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 20 Dec 2016 13:36:37 GMT
Content-Length: 38
"19a9e99c-8490-4c7d-b984-ab78cbdae066"
```

5.3.2. Добавление пользователя с указанием имени группы

HTTP метод	Путь	Описание
POST	/user/{groupName}	Добавление пользователя в БД
Параметры	Тип	Описание
groupName	string	Имя группы, в которую будет добавлен пользователь
identifiers	Dictionary<IdentifierType, string>	Список идентификаторов пользователя

HTTP метод	Путь	Описание
Возвращаемое значение	Тип	Описание
	string	Внутренний идентификатор пользователя

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/Default HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 74
{"Login":"RestTestUser","Email":"test@cp.ru","PhoneNumber":"+79150510528"}
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 20 Dec 2016 13:36:37 GMT
Content-Length: 38
"19a9e99c-8490-4c7d-b984-ab78cbdae066"
```

5.3.3. Удаление пользователя

HTTP метод	Путь	Описание
DELETE	/user/{id}	Удаление пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/f6b537fd-affc-443a-8121-6686c8853088 HTTP/1.1
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
```

5.3.4. Получение информации о пользователе по его внутреннему идентификатору

HTTP метод	Путь	Описание
GET	/user/{id}	Получение информации о пользователе по внутреннему идентификатору
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
	DssUserInfo	Информация о пользователе

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user/f6b537fd-affc-443a-8121-6686c8853088 HTTP/1.1
Content-Length: 0
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 20 Dec 2016 14:41:56 GMT
Content-Length: 334

{"UserId":"9d885609-eefb-4daf-b6bb-
864f199c44cf","Login":"RestTestUser","PhoneNumber":null,"Email":null,"PhoneConfi
rmed":false,"EmailConfirmed":false,"DisplayName":null,"DistinguishName":null,"Ac
countLocked":false,"Group":"Default","CreationDate":"2016-12-
20T17:41:55.823","LockoutDate":null,"LastLoginDate":"2016-12-20T17:41:55.823"}
```

5.3.5. Получение информации о пользователе по идентификатору

HTTP метод	Путь	Описание
GET	/user?type={type}&value={value}	Получение информации о пользователе по идентификатору
Параметры	Тип	Описание
type	IdentifierType	Тип идентификатора пользователя
value	string	Значение идентификатора пользователя
Возвращаемое значение	Тип	Описание
	DssUserInfo	Информация о пользователе

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user?type=Login&value=RestTestUser HTTP/1.1
Content-Length: 0
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 20 Dec 2016 14:41:56 GMT
Content-Length: 334

{"UserId":"9d885609-eefb-4daf-b6bb-
864f199c44cf","Login":"RestTestUser","PhoneNumber":null,"Email":null,"PhoneConfi
rmed":false,"EmailConfirmed":false,"DisplayName":null,"DistinguishName":null,"Ac
countLocked":false,"Group":"Default","CreationDate":"2016-12-
20T17:41:55.823","LockoutDate":null,"LastLoginDate":"2016-12-20T17:41:55.823"}
```

5.3.6. Получение связанных логинов пользователя

HTTP метод	Путь	Описание
GET	/user/{id}/login	Получение списка связанных логинов пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
	List<ExternalLoginInfo>	Список связанных логинов пользователя

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user/323c0fcc-5b10-4966-8a42-a505ba643b98/login
HTTP/1.1
Content-Length: 0
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 20 Dec 2016 14:41:56 GMT
Content-Length: 94
[{"Login":"ADFSRestUser","IssuerName":"ADFS"},{"Login":"RestTestUser","IssuerNam
e":"realsts"}]
```

5.3.7. Добавление внешнего логина

HTTP метод	Путь	Описание
POST	/user/{id}/login	Добавление внешнего логина
Параметры	Тип	Описание

HTTP метод	Путь	Описание
id	string	Внутренний идентификатор пользователя
loginInfo	ExternalLoginInfo	Информация о добавляемом логине
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/2c73fbf7-ab97-4d12-b893-bb491e43ff52/login
HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 48

{"Login":"SampleLogin","IssuerName":"SampleSts"}
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 20 Dec 2016 14:41:56 GMT
```

5.3.8. Получение номера мобильного телефона пользователя

HTTP метод	Путь	Описание
GET	/user/{id}/phonenumber	Получение номера мобильного телефона пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
	UserPhoneInfo	Информация о номере мобильного телефона пользователя

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user/d0e89f4b-fee1-4a30-a3ff-9e2647908e0b/phonenumber
HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 20 Dec 2016 15:51:54 GMT
Content-Length: 47

{"PhoneNumber":"79150000000","Confirmed":false}
```

5.3.9. Добавление номера мобильного телефона пользователю

HTTP метод	Путь	Описание
POST	/user/{id}/phonenumber	Добавление номера мобильного телефона
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
phoneNumber	string	Номер мобильного телефона
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/e33f155a-d452-4225-a7e5-5868602b6d96/phonenumber
HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 14
Connection: Keep-Alive

"+79150000000"
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 16:10:37 GMT
```

5.3.10. Удаление номера мобильного телефона пользователя

Примечание: удаление возможно лишь тогда, когда номер мобильного телефона не выбран в качестве типа идентификатора, доступного для регистрации пользователя (см. 4.2.1)

HTTP метод	Путь	Описание
DELETE	/user/{id}/phonenumber	Удаление номера мобильного телефона пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/phonenumber
HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
Content-Length: 0
```

5.3.11. Получение адреса электронной почты пользователя

HTTP метод	Путь	Описание
GET	/user/{id}/email	Получение адреса электронной почты пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
	UserEmailInfo	Информация о номере мобильного телефона пользователя

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user/d0e89f4b-fee1-4a30-a3ff-9e2647908e0b/email HTTP/1.1
Content-Length: 0
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 20 Dec 2016 15:51:54 GMT
Content-Length: 47

{"Email":"sample@cp.ru","Confirmed":false}
```

5.3.12. Добавление адреса электронной почты

HTTP метод	Путь	Описание
POST	/user/{id}/email	Добавление адреса электронной почты
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
email	string	Адрес электронной почты
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/e33f155a-d452-4225-a7e5-5868602b6d96/email
HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 14
Connection: Keep-Alive

"sample@cp.ru":
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 16:10:37 GMT
Content-Length: 0
```

5.3.13. Удаление адреса электронной почты пользователя

Примечание: удаление возможно лишь тогда, когда адрес электронной не выбран в качестве типа идентификатора, доступного для регистрации пользователя (см. 4.2.1)

HTTP метод	Путь	Описание
DELETE	/user/{id}/email	Удаление адреса электронной почты
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/email
HTTP/1.1
Content-Length: 0
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
Content-Length: 0
```

5.3.14. Получение информации об OTP-токене

HTTP метод	Путь	Описание
GET	/user/{id}/oath	Получение информации об OTP-токене
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя

HTTP метод	Путь	Описание
Возвращаемое значение	Тип	Описание
	UserOtpTokenInfo	Информация об OTP-токене пользователя

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user/efb67416-9469-4771-aa77-6f1e7db7a5fe/oath HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Mon, 26 Dec 2016 11:52:19 GMT

{"Serial":"AA000002","Type":"HOTP"}
```

5.3.15. Назначение OTP-токена пользователю

HTTP метод	Путь	Описание
POST	/user/{id}/oath	Назначение OTP-токена пользователю
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
tokenInfo	OAthTokenInfo	Информация о назначаемом токене
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/efb67416-9469-4771-aa77-6f1e7db7a5fe/oath HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 62

{"Serial":"AA000002","FirstOtp":"253311","SecondOtp":"568278"}
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 16:10:37 GMT
Content-Length: 0
```

5.3.16. Удаление назначенного OTP-токена

HTTP метод	Путь	Описание
DELETE	/user/{id}/oath	Удаление информации об OTP-токене из учетной записи пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/oath
HTTP/1.1
Content-Length: 0
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
Content-Length: 0
```

5.3.17. Получение информации о Sim-карте пользователя для аутентификации через SimAuth

HTTP метод	Путь	Описание
GET	/user/{id}/simauth	Получение информации о Sim-карте пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
	UserSimAuthInfo	Информация о Sim-карте пользователя

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user/efb67416-9469-4771-aa77-6f1e7db7a5fe/simauth
HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Mon, 26 Dec 2016 12:49:44 GMT
```

```
Content-Length: 109

{"IccId":"896010265977857677","ProfileId":"3E906AFF-8639-463E-AF59-
07A44A96DD21","PhoneNumber":"79150000000"}
```

5.3.18. Назначение Sim-карты пользователю для аутентификации через SimAuth

HTTP метод	Путь	Описание
POST	/user/{id}/simauth	Назначение Sim-карты пользователю
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
tokenInfo	SimAuthTokenInfo	Информация о назначаемой Sim-карте
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/f11eb4f5-ceed-48fc-8082-ae6e768fd457/simauth
HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 81

{"IccId":"896010265977857677","ProfileId":"3e906aff-8639-463e-af59-
07a44a96dd21"}
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 16:10:37 GMT
Content-Length: 0
```

5.3.19. Удаление назначенной Sim-карты пользователя

HTTP метод	Путь	Описание
DELETE	/user/{id}/simauth	Удаление информации о Sim-карте из учетной записи пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание

HTTP метод	Путь	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/simauth
HTTP/1.1
Content-Length: 0
Connection: Keep-Alive
```

Ответ

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
Content-Length: 0
```

5.3.20. Получение информации о данных аутентификации пользователя в системе MobileAuth

HTTP метод	Путь	Описание
GET	/user/{id}/mobileauth	Получение информации об аутентификационных данных пользователя в системе MobileAuth
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
	UserMobileAuthInfo	Данные аутентификации пользователя

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user/efb67416-9469-4771-aa77-6f1e7db7a5fe/mobileauth
HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Wed, 28 Dec 2016 16:36:46 GMT
Content-Length: 91

{"UserId":"5ba41e08-09b2-46f5-a70d-b0ca79c45fca","KeyExpirationTime":"2017-12-
28T00:00:00"}
```

5.3.21. Назначение данных аутентификации пользователя в системе MobileAuth

HTTP метод	Путь	Описание
POST	/user/{id}/mobileauth	Назначение данных аутентификации пользователя в системе MobileAuth
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
tokenInfo	MobileAuthTokenInfo	Информация о назначаемых данных аутентификации MobileAuth
Возвращаемое значение	Тип	Описание
	MobileAuthCreateInfoEx	Результат назначения данных аутентификации пользователю

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/da625bb0-e847-4d86-83a0-df3fa029c4b5/mobileauth
HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 69
Connection: Keep-Alive

{"UserContactInfo":"79150000000", "UserContactInfoType":"PhoneNumber",
"NeedXmlKeyInfo":true, "KeyInfoPinCode":"123456"}
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Wed, 28 Dec 2016 16:36:46 GMT
Content-Length: 7719

{"ExternalUserId":"5ba41e08-09b2-46f5-a70d-
b0ca79c45fca","QrCode":"R0lGO...","KeyExpirationTime":"2017-12-28T00:00:00",
"XmlKeyInfo":"<keyData xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><systemId>d7e4d5a5-3eb6-
46b3-8f31-1d31c70f2058</systemId><userId>da625bb0-e847-4d86-83a0-
df3fa029c4b5</userId><expDate>2018-07-
27</expDate><withFingerPrint>false</withFingerPrint><storeType>DSS</storeType><i
nteractionExternalURL>http://stenddss.cryptopro.ru/MyDssServerExternal/Interacti
onService.svc</interactionExternalURL><onlineConfirmURL>http://</onlineConfirmUR
L><encryptedKey>EQIAADSAAAAOZgAABIAJgAAAAAAAAAAAAAAAAAAAAAD77YvgJMjVqjmfiOi5YUaL
0t493RqUICtqAxxuyv53nxNN3B5gkE7O6eVUdHNES7kRAgAANIAAAA5mAAAEgAmAAAAAAAAAAAAAAAA
AAAAAJFUkUPLuL3WKPrxdMRy7i/BSf7eIGyKUCIIEuG+tAbZaIna/ZVt2WCNu7GJbGTAJQ==</encryp
tedKey></keyData>"}
```

5.3.22. **Обновление данных аутентификации пользователя в системе MobileAuth**

HTTP метод	Путь	Описание
PATCH	/user/{id}/mobileauth	Обновление данных аутентификации пользователя в системе MobileAuth
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
tokenInfo	MobileAuthTokenInfo	Информация об обновляемых данных аутентификации MobileAuth
Возвращаемое значение	Тип	Описание
	MobileAuthUpdateInfoEx	Результат обновления данных аутентификации пользователя

Пример выполнения метода:

Запрос:

```
PATCH https://dss/STS/ums/user/da625bb0-e847-4d86-83a0-df3fa029c4b5/mobileauth
HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 69
Connection: Keep-Alive

{"UserContactInfo":"79150000000", "UserContactInfoType":"PhoneNumber",
"NeedXmlKeyInfo":true, "KeyInfoPinCode":"123456"}
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Wed, 28 Dec 2016 16:36:46 GMT
Content-Length: 7719

{"ExternalUserId":"5ba41e08-09b2-46f5-a70d-
b0ca79c45fca","QrCode":"R0lGO...","KeyExpirationTime":"2017-12-28T00:00:00",
"XmlKeyInfo":"<keyData xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><systemId>d7e4d5a5-3eb6-
46b3-8f31-1d31c70f2058</systemId><userId>da625bb0-e847-4d86-83a0-
df3fa029c4b5</userId><expDate>2018-07-
27</expDate><withFingerPrint>false</withFingerPrint><storeType>DSS</storeType><i
nteractionExternalURL>http://stenddss.cryptopro.ru/MyDssServerExternal/Interacti
onService.svc</interactionExternalURL><onlineConfirmURL>http://</onlineConfirmUR
L><encryptedKey>EQIAADSAAAAOZgAABIAJgAAAAAAAAAAAAAAAAAAAAAD77YvgJMjVqjmfiOi5YUaL
0t493RqUICtqAxxuyv53nxNN3B5gkE7O6eVUdHNES7kRAgAANIAAAA5mAAAEgAmAAAAAAAAAAAAAAAA
AAAAAJFUkUPLuL3WKPrxdMRy7i/BSf7eIGyKUCIIEuG+tAbZaIna/ZVt2WCNu7GJbGTAJQ==</encryp
tedKey></keyData>"}
```

5.3.23. Удаление данных аутентификации пользователя в системе MobileAuth

HTTP метод	Путь	Описание
DELETE	/user/{id}/mobileauth	Удаление данных аутентификации пользователя в системе MobileAuth
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/mobileauth
HTTP/1.1
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
Content-Length: 0
```

5.3.24. Получение информации о схеме аутентификации пользователя

HTTP метод	Путь	Описание
GET	/user/{id}/mobileauth	Получение информации о схеме аутентификации пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
	List<AuthenticationInfo>	Список шагов аутентификации пользователя

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user/9ca74fb4-815d-48ed-a36c-8925f0d39c26/authmethod
HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Wed, 28 Dec 2016 16:56:32 GMT
```

```
Content-Length: 192

[{"MethodUri":"http://schemas.microsoft.com/ws/2012/09/identity/authenticationmethod/none","Level":1},{"MethodUri":"http://dss.cryptopro.ru/identity/authenticationmethod/otpviasms","Level":2}]
```

5.3.25. Назначение метода аутентификации «Только идентификация»

HTTP метод	Путь	Описание
POST	/user/{id}/authmethod/idonly	Назначение метода аутентификации «Только идентификация»
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Level	int	Номер шага, на который назначается метод
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/ca7a8ce2-e6ec-43bf-a3e3-086d12efd6a5/authmethod/idonly?level=1 HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Date: Thu, 29 Dec 2016 13:13:29 GMT
Content-Length: 0
```

5.3.26. Удаление метода аутентификации «Только идентификация»

HTTP метод	Путь	Описание
DELETE	/user/{id}/authmethod/idonly	Удаление метода аутентификации «Только идентификация» из схемы аутентификации пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/
authmethod/idonly HTTP/1.1
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
```

5.3.27. Назначение метода аутентификации по паролю

HTTP метод	Путь	Описание
POST	/user/{id}/authmethod/password	Назначение метода аутентификации по паролю
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Level	int	Номер шага, на который назначается метод
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/ca7a8ce2-e6ec-43bf-a3e3-
086d12efd6a5/authmethod/password?level=1 HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Date: Thu, 29 Dec 2016 13:13:29 GMT
Content-Length: 0
```

5.3.28. Удаление метода аутентификации по паролю

HTTP метод	Путь	Описание
DELETE	/user/{id}/authmethod/password	Удаление метода аутентификации по паролю
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/
authmethod/password HTTP/1.1
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
```

5.3.29. Назначение метода аутентификации по одноразовым паролям, отправленным по СМС

HTTP метод	Путь	Описание
POST	/user/{id}/authmethod/otpviasms	Назначение метода аутентификации по одноразовым паролям, отправленным по СМС
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Level	int	Номер шага, на который назначается метод
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/ca7a8ce2-e6ec-43bf-a3e3-
086d12efd6a5/authmethod/otpviasms?level=1 HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Date: Thu, 29 Dec 2016 13:13:29 GMT
Content-Length: 0
```

5.3.30. Удаление метода аутентификации по одноразовым паролям, отправленным по СМС

HTTP метод	Путь	Описание
DELETE	/user/{id}/authmethod/otpviasms	Удаление метода аутентификации по одноразовым паролям, отправленным по СМС
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя

HTTP метод	Путь	Описание
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/
authmethod/otpviasms HTTP/1.1
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
```

5.3.31. Назначение метода аутентификации по одноразовым паролям, отправленным по электронной почте

HTTP метод	Путь	Описание
POST	/user/{id}/authmethod/otpviaemail	Назначение метода аутентификации по одноразовым паролям, отправленным по СМС
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Level	int	Номер шага, на который назначается метод
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/ca7a8ce2-e6ec-43bf-a3e3-
086d12efd6a5/authmethod/otpviaemail?level=1 HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Date: Thu, 29 Dec 2016 13:13:29 GMT
Content-Length: 0
```

5.3.32. Удаление метода аутентификации по одноразовым паролям, отправленным по электронной почте

HTTP метод	Путь	Описание
DELETE	/user/{id}/authmethod/otpviaemail	Удаление метода аутентификации по одноразовым паролям, отправленным по СМС
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/
authmethod/otpviaemail HTTP/1.1
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
```

5.3.33. Назначение метода аутентификации по стандарту OAth

HTTP метод	Путь	Описание
POST	/user/{id}/authmethod/oath	Назначение метода аутентификации по стандарту OAth
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Level	int	Номер шага, на который назначается метод
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/ca7a8ce2-e6ec-43bf-a3e3-
086d12efd6a5/authmethod/oath?level=1 HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Date: Thu, 29 Dec 2016 13:13:29 GMT
Content-Length: 0
```

5.3.34. Удаление метода аутентификации по стандарту OAth

HTTP метод	Путь	Описание
DELETE	/user/{id}/authmethod/oath	Удаление метода аутентификации по стандарту OAth
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/
authmethod/oath HTTP/1.1
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
```

5.3.35. Назначение метода аутентификации SimAuth

HTTP метод	Путь	Описание
POST	/user/{id}/authmethod/SimAuth	Назначение метода аутентификации SimAuth
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Level	int	Номер шага, на который назначается метод
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/ca7a8ce2-e6ec-43bf-a3e3-
086d12efd6a5/authmethod/simauth?level=1 HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Date: Thu, 29 Dec 2016 13:13:29 GMT
Content-Length: 0
```

5.3.36. Удаление метода аутентификации SimAuth

HTTP метод	Путь	Описание
DELETE	/user/{id}/authmethod/simauth	Удаление метода аутентификации SimAuth
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/
authmethod/simauth HTTP/1.1
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
```

5.3.37. Назначение метода аутентификации MobileAuth

HTTP метод	Путь	Описание
POST	/user/{id}/authmethod/mobileauth	Назначение метода аутентификации MobileAuth
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Level	int	Номер шага, на который назначается метод
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/ca7a8ce2-e6ec-43bf-a3e3-
086d12efd6a5/authmethod/MobileAuth?level=1 HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Date: Thu, 29 Dec 2016 13:13:29 GMT
Content-Length: 0
```

5.3.38. Удаление метода аутентификации MobileAuth

HTTP метод	Путь	Описание
DELETE	/user/{id}/authmethod/simauth	Удаление метода аутентификации MobileAuth
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
DELETE https://dss/STS/ums/user/fb4764f9-8327-4e3c-8f41-495022973b77/
authmethod/MobileAuth HTTP/1.1
Connection: Keep-Alive
```

Ответ:

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 13:36:37 GMT
```

5.3.39. Получение политики подтверждения операций пользователем

HTTP метод	Путь	Описание
GET	/user/{id}/operationpolicy	Получение политики подтверждения операций пользователем
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
	List<OperationPolicy>	Политика подтверждения операций пользователем

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user/efb67416-9469-4771-aa77-
6f1e7db7a5fe/operationpolicy HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Thu, 29 Dec 2016 13:28:45 GMT
Content-Length: 473

[{"Action":1,"ConfirmationRequired":false},{"Action":2,"ConfirmationRequired":tr
ue},{"Action":4,"ConfirmationRequired":false},{"Action":8,"ConfirmationRequired"
:false},{"Action":16,"ConfirmationRequired":false},{"Action":32,"ConfirmationReq
uired":false},{"Action":64,"ConfirmationRequired":false},{"Action":128,"Confirma
tionRequired":false},{"Action":256,"ConfirmationRequired":false},{"Action":512,"
ConfirmationRequired":false},{"Action":1024,"ConfirmationRequired":true}]
```

5.3.40. Назначение политики подтверждения операций пользователем

HTTP метод	Путь	Описание
POST	/user/{id}/mobileauth	Назначение политики подтверждения операций пользователем
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
actions	List<DSSActions>	Список операций, для которых требуется подтверждение
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/da625bb0-e847-4d86-83a0-df3fa029c4b5/mobileauth
HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 69
Connection: Keep-Alive

{"UserContactInfo":"79150000000","UserContactInfoType":"PhoneNumber"}
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```

5.3.41. Блокировка/разблокировка пользователя

HTTP метод	Путь	Описание
POST	/user/{id}/lockout?lock={value}	Блокировка/разблокировка пользователя
Параметры	Тип	Описание

HTTP метод	Путь	Описание
id	string	Внутренний идентификатор пользователя
lock	bool	Требуемое действие. Истина – заблокировать пользователя, ложь - разблокировать
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/e585058f-760c-4ca6-9998-
a9871f3b8f0a/lockout?lock=true HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```

5.3.42. Назначение различительного имени субъекта

HTTP метод	Путь	Описание
POST	/user/{id}/dn	Назначение различительного имени субъекта
Параметры	Тип	Описание
id	String	Внутренний идентификатор пользователя
rdns	Dictionary<int, string>	Словарь компонент имени, назначаемых пользователю. В качестве ключа используется идентификатор компонента имени в БД DSS. (см. 4.2.1)
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/8b6a9c5c-9e1d-420b-87f0-b9606fda1429/dn HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 33

{"11":"SampleCN","10":"SampleOU"}
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```

5.3.43. Получение имени группы, в которой состоит пользователь

HTTP метод	Путь	Описание
GET	/user/{id}/group	Получение имени группы, в которой состоит пользователь
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
	string	Имя группы, в которой состоит пользователь

Пример выполнения метода:

Запрос:

```
GET https://dss/STS/ums/user/efb67416-9469-4771-aa77-6f1e7db7a5fe/group HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Thu, 29 Dec 2016 13:28:45 GMT
Content-Length: 9

"Default"
```

5.3.44. Назначение группы пользователю

Примечание. Пользователю может быть назначена группа лишь в том случае, если оператор, осуществляющий управление пользователем состоит в запрошенной группе.

HTTP метод	Путь	Описание
POST	/user/{id}/group	Назначение группы пользователю
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
groupName	string	Имя назначаемой группы
Возвращаемое значение	Тип	Описание
Метод не возвращает дополнительных данных		

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/da625bb0-e847-4d86-83a0-df3fa029c4b5/group
HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 9
```

```
"NewGroup"
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```

5.3.45. Сброс пароля пользователя

HTTP метод	Путь	Описание
POST	/user/{id}/password	Сброс пароля пользователя
Параметры	Тип	Описание
id	string	Внутренний идентификатор пользователя
Возвращаемое значение	Тип	Описание
	string	Новый пароль пользователя

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/user/da625bb0-e847-4d86-83a0-df3fa029c4b5/password
HTTP/1.1
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```

5.4. Конечная точка Users

Конечная точка Users используется для осуществления фильтрованной выборки пользователей DSS.

HTTP метод	Путь	Описание
POST	/users	Запрос фильтрованной выборки пользователей DSS
Параметры	Тип	Описание
request	UserRecordsRequest	Запрос на получение выборки
Возвращаемое значение	Тип	Описание
	UserRecordsResponse	Выборка пользователей, соответствующая заданным фильтрам

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/users HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 100

{"StartPosition":0,"EndPosition":1,"Filters":[{"Column":0,"Operation":0,"Value":
"AnotherTestUser"}]}
```

Ответ:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Date: Thu, 29 Dec 2016 15:47:44 GMT
Content-Length: 384

{"UserInfos":[{"UserId":"bd8bce50-7f9e-4e37-aa27-
a84c74ed9050","Login":"AnotherTestUser","PhoneNumber":null,"Email":null,"PhoneCo
nfirmed":false,"EmailConfirmed":false,"DisplayName":null,"DistinguishName":null,
"AccountLocked":false,"Group":"Default","CreationDate":"2016-12-
29T18:47:39.97","LockoutDate":null,"LastLoginDate":"2016-12-
29T18:47:39.97"}],"TotalCount":2,"AffectedCount":1}
```

5.5. Конечная точка AuthnTokens

Конечная точка AuthnTokens используется для осуществления фильтрованной выборки средств аутентификации пользователей DSS.

HTTP метод	Путь	Описание
POST	/authntokens	Запрос фильтрованной выборки средств аутентификации
Параметры	Тип	Описание
request	TokenRecordsRequest	Запрос на получение выборки
Возвращаемое значение	Тип	Описание
	TokenRecordsResponse	Выборка средств аутентификации, соответствующая заданным фильтрам

Пример выполнения метода:

Запрос:

```
POST https://dss/STS/ums/authntokens HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 90

{"StartPosition":0,"EndPosition":9,"Filters":[{"Column":1,"Operation":2,"Value":
"%001%"}]}
```

Ответ:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Date: Fri, 08 Sep 2017 15:57:54 GMT
Content-Length: 202

{"TokenInfos":[{"Id":1,"Serial":"AA000001","UserName":null,"TokenType":"HOTP","P
arameters":{"Digits":"6","Crypto":"SHA1","LookAheadWindow":"10","IterationNumber
":"1"}}],"TotalCount":6,"AffectedCount":1}
```

5.6. Типы данных

В данном разделе представлены типы данных, специфичные для REST интерфейса Сервиса Управления пользователями ЦИ «КриптоПро DSS».

5.6.1. ExternalLoginInfo

Таблица 91. Описание полей класса ExternalLoginInfo

Поле	Тип	Описание
Login	string	Внешний логин пользователя
IssuerName	string	Имя доверенного издателя

5.6.2. OAthTokenInfo

Таблица 92. Описание полей класса OAthTokenInfo

Поле	Тип	Описание
Serial	string	Серийный номер токена
FirstOtp	string	Первый пароль для синхронизации
SecondOtp	string	Второй пароль для синхронизации (для токенов типа TOTP должен быть выставлен в null)

5.6.3. SimAuthTokenInfo

Таблица 93. Описание полей класса SimAuthTokenInfo

Поле	Тип	Описание
IccId	string	Идентификатор Sim-карты
ProfileId	string	Идентификатор криптопровайдера

5.6.4. MobileAuthTokenInfo

Таблица 94. Описание полей класса MobileTokenInfo

Поле	Тип	Описание
UserContactInfo	string	Контактная информация пользователя
UserContactInfoType	string	Тип контактной информации пользователя
NeedXmlKeyInfo	bool	Флаг, показывающий, нужно ли возвращать вектор аутентификации в формате XML

Поле	Тип	Описание
KeyInfoPinCode	string	ПИН-код для расшифрования ключа, переданного в формате XML. ПИН-код должен быть не короче, чем указано в конфигурации компонента myDSS в параметре SecondKeyPartLength

5.6.5. Обработка ошибок

В случае успешного выполнения операции статус HTTP-ответа сервера имеет значение 200. В случае возникновения ошибки сервер вернёт статус из диапазона 400-500. Тело ответа будет содержать JSON-объект с полем **Message**, хранящим информацию об ошибке.

Пример ответа с внутренней ошибкой сервера.

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json; charset=utf-8
Date: Thu, 29 Dec 2016 15:53:21 GMT
Content-Length: 4245

{"Message":"An error has occurred.","ExceptionMessage":"Значение не может быть
неопределенным.\r\nИмя параметра:
IssuerName","ExceptionType":"System.ArgumentNullException","StackTrace":"…"}
```

6. Аутентификация пользователей СЭП

Аутентификацию пользователей СЭП осуществляет компонент Центр Идентификации. Результатом успешной процедуры аутентификации является маркер доступа, дающий право выполнять определённые действия над ресурсами пользователя на Сервиса Подписи.

6.1. Общие сведения

Все способы аутентификации пользователей, поддерживаемые СЭП, делятся на две группы:

1. Методы первичной аутентификации.

Главной задачей методов этой группы является идентификация пользователей. СЭП поддерживает следующие методы первичной аутентификации:

- **По логину и паролю**. Для успешной аутентификации пользователю требуется предоставить логин и пароль от своей учётной записи DSS.
- **По сертификату**. Для успешной аутентификации пользователю требуется предоставить действительный сертификат.
- **По SAML-маркеру**. Для успешной аутентификации пользователю требуется предоставить маркер доступа от доверенного ЦИ.
- **«Только идентификация»**. Для успешной аутентификации пользователю требуется предоставить только логин от учётной записи DSS. Применение данного способа имеет смысл только совместно с каким-либо способом вторичной аутентификации.

2. Методы вторичной аутентификации.

Методы вторичной аутентификации используются для дополнительного подтверждения учётных данных пользователя. СЭП поддерживает следующие методы вторичной аутентификации:

- Одноразовый пароль, отправляемый по SMS.
- Одноразовый пароль, отправляемый по Email.
- Одноразовый пароль, создаваемый по протоколу OATH (HOTP и TOTP).
- Аутентификация с помощью мобильного приложения myDSS.
- Аутентификация с помощью апплета на SIM-карте.

6.2. Подтверждение операций

6.2.1. Общие сведения

Вторичную аутентификацию можно использовать для дополнительного подтверждения следующих действий:

- доступ пользователя к Веб-интерфейсу Сервера подписи и Личному кабинету на ЦИ (далее «Подтверждение входа»),
- обращение к закрытому ключу сертификата (далее «Подтверждение операций»).

Обращение к закрытому ключу сертификата происходит при следующих операциях:

- подпись документа,
- смена PIN-кода,
- удаление сертификата,

- расшифрование документа,
- создание запроса на сертификат.

Требуется или нет дополнительное подтверждение для выполнения действия, клиент может определить на основе политики сервера подписи, вызвав метод GetDSSPolicy SOAP интерфейса или отправив запрос к конечной точке Policy REST интерфейса. В поле **ActionPolicy** возвращаемого в этом случае объекта будет находится список свойств операций сервера подписи. В этом списке необходимо найти нужно действие и проверить значение поля **MfaRequired**. Для действий, требующих подтверждения данное значение будет равно **true**.

Следует также отметить, что политику можно игнорировать в случае, если значение глобальной политики подтверждения операций (поле TransactionConfirmation объекта класса DSSPolicy) имеет значение **On** или **Off**.

Для подтверждения транзакции Сервис Подписи требует, чтобы клиент предоставил маркер безопасности, содержащий следующий набор утверждений:

1. http://dss.cryptopro.ru/identity/claims/dsstransactionid

Идентификатор подтверждаемой транзакции. Данный идентификатор формируется Сервисом Подписи с помощью вызова метода GetTransactionID() (см. пункт 2.5.1).

2. http://dss.cryptopro.ru/identity/claims/dsstransactionexpiredate

Срок истечения транзакции, содержит дату и время истечения срока транзакции в формате универсального времени (UTC) **yyyy-MM-ddTHH:mm:ss.fffZ** (например, 2012-05-18T13:34:44.435Z). После истечения указанного срока транзакция считается недействительной. Значение данного утверждения настраивается на Центре Идентификации.

Последовательность действий для подтверждения операции состоит из следующих шагов:

1. Получение маркера доступа для операций, не требующих подтверждения (далее маркер доступа общего назначения).

Формат запросов зависит от конкретного протокола взаимодействия (WS-Trust или OAuth). Подробнее эта процедура рассмотрена в разделе 6.3.

2. Создание транзакции на сервисе подписи.
3. Получение маркера доступа для операции, требующей подтверждения.

Формат запросов зависит от конкретного протокола взаимодействия (WS-Trust или OAuth). Подробнее эта процедура рассмотрена в подразделах раздела 6.3.

4. Вызов операции, требующей подтверждения.

6.2.2. Создание транзакции на сервисе подписи.

Для создания транзакции на сервисе подписи используются методы:

- CreateTransactionToken (см. п. 2.5.3) SOAP-интерфейса сервиса подписи.
- Конечная точка Transactions (см. п. 3.6) REST-интерфейса сервиса подписи.

Далее рассматривается общий набор данных, который необходимо предоставить сервису подписи для создания записи о транзакции.

Для создания транзакции клиентское приложение передаёт следующий набор данных

- Тип операции. Элемент из перечисления DSSActions (за исключением Issue).
- Параметры операции (необходимый перечень параметров для каждой операции приведён в таблице Таблица 95). Все параметры передаются в виде пар {ключ, значение}.

- Данные операции. Содержимое документа, над которым выполняется операции.

Данный параметр представляет собой содержимое документа в виде массива байтов. Имеет значение только для операций подписи, усовершенствования подписи, расшифрования (операция шифрования не требует подтверждения, так как для её осуществления не требуется получать доступ к закрытому ключу пользователя).

Таблица 95. Параметры подтверждаемых операций на Сервисе Подписи

Операция	Параметры	Описание
SignDocument	SignatureType	Тип подписи. В качестве значения выступают элементы перечисления SignatureType
	DocumentInfo	Название документа.
	DocumentType	Тип документа. Данный параметр используется для формирования отображаемых данных (см. п. 6.3.3).
	CertificateID	Идентификатор сертификата.
SignDocuments	CertificateID	Идентификатор сертификата.
	SignatureType	Тип подписи. В качестве значения выступают элементы перечисления SignatureType.
DecryptDocument	CertificateID	Идентификатор сертификата.
	DocumentInfo	Название документа.
CreateRequest	CAId	Идентификатор обработчика УЦ.
	CertDistinguishedName	Различительное имя субъекта сертификата.
	CertTemplateOid	Объектный идентификатор (oid) шаблона сертификата.
DeleteCertificate DeleteCertificate ChangePin RenewCertificate RevokeCertificate HoldCertificate UnholdCertificate	CertificateID	Идентификатор сертификата.

6.3. Получение маркера доступа

Процедура аутентификации выполняется различными способами, зависящими от протокола получения маркера доступа. ЦИ КриптоПро DSS предоставляет следующие протоколы:

1. WS-Trust 1.3;

2. OAuth 2.0 + OpenId Connect 1.0.

6.3.1. Протокол WS-Trust 1.3

Спецификация WS-Trust 1.3 определяет интерфейс Службы Маркеров Безопасности (Security Token Service, STS). Взаимодействие клиента и данной службы происходит путём обмена сообщениями специального вида. Существует два типа сообщений:

- **RequestSecurityToken (RST)** – запрос на выпуск маркера безопасности. В данном сообщении передаются параметры маркера безопасности, идентификатор проверяющей стороны, где он будет использоваться, и другая служебная информация.
- **RequestSecurityTokenResponse (RSTR)** – ответные сообщения, которыми обмениваются клиент и служба в ходе взаимодействия. В последнем RSTR сообщении служба возвращает запрашиваемый маркер безопасности.

STS представляет собой SOAP сервис, поэтому первичная аутентификация пользователя осуществляется согласно спецификации WS-Security, то есть аутентификационные данные передаются в заголовках SOAP сообщений. Для поддержки различных способов первичной аутентификации существуют отдельный конечные точки с соответствующими привязками.

В Таблица 96 представлена информация о том, какой способ первичной аутентификации поддерживают конечные точки STS ЦИ.

Таблица 96 - Конечные точки STS ЦИ КриптоПро DSS

Относительный адрес	Способ аутентификации	Транспортная безопасности
/	Логин и пароль	Нет
/username/transport	Логин и пароль	Да
/issuedtoken	Маркер безопасности	Нет
/issuedtoken/transport	Маркер безопасности	Да
/cert	Сертификат	Да

Использование транспортной безопасности подразумевает установление TLS соединения между клиентом и сервисом.

Вторичная аутентификация и подтверждение операций реализованы согласно процедуре User Interactive Challenge (UIC), описанной в документе http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.html#_Toc212615471.

Согласно процедуре UIC, RSTR сообщение Центра Идентификации расширяется элементом **InteractiveChallenge**. Наличие этого элемента в RSTR свидетельствует о том, что ЦИ было недостаточно данных первичной аутентификации, и для получения маркера требуется дополнительное подтверждение.

Далее приводится описание XML-схемы элемента InteractiveChallenge:

```
<xs:element name='InteractiveChallenge' type='tns:InteractiveChallengeType' />
  <xs:complexType name='InteractiveChallengeType' >
    <xs:sequence>
      <xs:element ref='tns:Title' minOccurs='0' maxOccurs='1' />
      <xs:element ref='tns:TextChallenge' minOccurs='1' maxOccurs='unbounded' />
    </xs:sequence>
  </xs:complexType>
  <xs:element name='Title' type='tns:TitleType'/>
  <xs:complexType name="TitleType">
```

```
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="TextChallenge" type="tns:TextChallengeType"/>
  <xs:complexType name="TextChallengeType">
    <xs:attribute name="RefID" type="xs:anyURI" use="required"/>
    <xs:attribute name="Label" type="xs:string" use="optional"/>
  </xs:complexType>
```

Описание полей InteractiveChallenge приведено в Таблица 97.

Таблица 97 - Описание элементов InteractiveChallenge

Элемент	Тип данных	Значение
Title	string	Текст, описывающий общее назначение запрашиваемого у пользователя одноразового пароля.
RefID	string	Уникальный идентификатор подтверждаемой транзакции, формируемый ЦИ DSS.
Label	string	Текст с дополнительным описанием назначения одноразового пароля.

Значения полей Title и Label вызывающее приложение может использовать для формирования пользовательского интерфейса для запроса одноразового пароля.

В ответ пользователь должен отправить сообщение RSTR расширенное элементом **InteractiveChallengeResponse**. Данный элемент содержит ответ на аутентификационное испытание. Описание полей InteractiveChallengeResponse приведено в Таблица 98.

Далее приводится описание XML-схемы **InteractiveChallengeResponse**:

```
<xs:element name='InteractiveChallengeResponse'
type='tns:InteractiveChallengeResponseType' />
  <xs:complexType name='InteractiveChallengeResponseType' >
    <xs:sequence>
      <xs:element ref='tns:TextChallengeResponse' minOccurs='1'
      maxOccurs='unbounded'/>
  </xs:complexType>
  <xs:element name='TextChallengeResponse'
type='tns:TextChallengeResponseType'/>
  <xs:complexType name='TextChallengeResponseType'>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="RefId" type="xs:anyURI" use="required"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
```

Таблица 98 - Описание элементов InteractiveChallengeResponse

Элемент	Тип данных	Значение
RefID	string	Уникальный идентификатор подтверждаемой транзакции, формируемый ЦИ DSS.
TextChallengeResponse	string	Ответ на аутентификационное испытание.

Значение атрибута RefID должно быть взято из запроса (RSTR) одноразового пароля ЦИ. Данный идентификатор является уникальный и позволяет объединить всю цепочку

сообщений, которыми обмениваются клиент и сервис. В зависимости от способа аутентификации значения TextChallengeResponse могут быть различными.

Рассмотрим последовательность запросов, необходимых для получения маркера безопасности. В приведённых примерах заголовки SOAP сообщений сокращены и не содержат узла, отвечающего за безопасность. Само SOAP сообщение также не защищено. Конкретный вид заголовков и метод защиты сообщения зависит от конечной точки, к которой направлен запрос.

1. Запрос клиента на выпуск маркера безопасности.

Для получения маркера безопасности клиентское приложение формирует RST запрос следующего вида:

```
<s:Envelope xmlns:a="http://www.w3.org/2005/08/addressing"
xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Issue</a:Action>
  </s:Header>
  <s:Body>
    <trust:RequestSecurityToken xmlns:trust="http://docs.oasis-open.org/ws-
sx/ws-trust/200512">
      <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
<wsa:Address>https://dss.cryptopro.ru/SignServer/SignService.svc/token/transport
/nosc</wsa:Address>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <trust:Claims xmlns:auth="http://docs.oasis-
open.org/wsfed/authorization/200706" Dialect="http://docs.oasis-
open.org/wsfed/authorization/200706/authclaims">
        <auth:ClaimType
Uri="http://dss.cryptopro.ru/identity/claims/dsstransactiontokenid"
Optional="false">
          <auth:Value>5904CE31-B789-45BA-89CB-750F194CFE54</auth:Value>
        </auth:ClaimType>
        <auth:ClaimType
Uri="http://dss.cryptopro.ru/identity/claims/dsstransactionexpiredate"
Optional="false"/>
      </trust:Claims>
      <trust:KeyType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/SymmetricKey</trust:KeyType>
      <trust:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</trust:RequestType>
    </trust:RequestSecurityToken>
  </s:Body>
</s:Envelope>
```

Для RST в заголовке SOAP сообщения указывается действие с идентификатором http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue. В узле **AppliesTo** передаётся идентификатор проверяющей стороны, он должен быть равен адресу конечной точки сервиса подписи, где будет использован данный маркер. В данном примере это https://dss.cryptopro.ru/SignServer/SignService.svc/token/transport/nosc.

Если требуется подтвердить операцию доступа к закрытому ключу, то в RST включается узел **Claims**, в котором указывается требование поместить в маркер безопасности утверждение: http://dss.cryptopro.ru/identity/claims/dsstransactiontokenid со значением идентификатора транзакции (см. п. 6.2.2).

Тип ключа **KeyType**, помещаемого в маркер безопасности, должен быть равен http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey, а тип RST запроса содержать идентификатор http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue.

2. Ответ Центра Идентификации, содержащий испытание.

Если ЦИ начинает процедуру вторичной аутентификации, то его ответное сообщение RSTR будет иметь следующий вид:

```
<s:Envelope xmlns:a="http://www.w3.org/2005/08/addressing"
xmlns:s="http://www.w3.org/2003/05/soap-envelope">
      <s:Header/>
      <s:Body>
            <trust:RequestSecurityTokenResponse xmlns:trust="http://docs.oasis-
open.org/ws-sx/ws-trust/200512">
                  <InteractiveChallenge
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://docs.oasis-
open.org/ws-sx/ws-trust/200802">
                        <Title>На ваш номер отправлено SMS сообщение с
одноразовым паролем</Title>
                        <TextChallenge RefID="aa04a501-0058-49bc-b30a-
75a82a05d677" Label="Введите одноразовый пароль для подтверждения подписи"/>
                        <ContextData RefID="aa04a501-0058-49bc-b30a-
75a82a05d677"/>
                  </InteractiveChallenge>
                  <wsp:AppliesTo
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
                        <wsa:EndpointReference
xmlns:wsa="http://www.w3.org/2005/08/addressing">

      <wsa:Address>http://localhost/SignService/</wsa:Address>
                        </wsa:EndpointReference>
                  </wsp:AppliesTo>
            </trust:RequestSecurityTokenResponse>
      </s:Body>
</s:Envelope>
```

Испытание содержится в узле **InteractiveChallenge.** Клиентское приложение может использовать его для отображение пользователю соответствующих текстовых сообщений и полей ввода. Важным является параметр **RefID**, содержащий идентификатор текущей сессии вторичной аутентификации. Его нужно передавать во все последующие RSTR сообщения, отправляемые ЦИ.

3. Ответ клиентского приложения на испытание.

Для ответа на аутентификационные испытание клиент формирует RSTR сообщение.

```
<s:Envelope xmlns:a="http://www.w3.org/2005/08/addressing"
xmlns:s="http://www.w3.org/2003/05/soap-envelope">
      <s:Header>
            <a:Action s:mustUnderstand="1">http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/Issue</a:Action>
      </s:Header>
      <s:Body>
            <trust:RequestSecurityTokenResponseCollection
xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
                  <trust:RequestSecurityTokenResponse>
                        <InteractiveChallengeResponse
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://docs.oasis-
open.org/ws-sx/ws-trust/200802">
                              <TextChallengeResponse RefId="aa04a501-0058-49bc-
b30a-75a82a05d677">RLRRAY</TextChallengeResponse>
                        </InteractiveChallengeResponse>
                        <wsp:AppliesTo
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
                              <wsa:EndpointReference
xmlns:wsa="http://www.w3.org/2005/08/addressing">
```

```
      <wsa:Address>http://localhost/SignService/</wsa:Address>
                              </wsa:EndpointReference>
                        </wsp:AppliesTo>
                        <trust:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</trust:RequestType>
                  </trust:RequestSecurityTokenResponse>
            </trust:RequestSecurityTokenResponseCollection>
      </s:Body>
</s:Envelope>
```

Для RSTR в заголовке SOAP сообщения указывается действие с идентификатором http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/Issue. Ответ на аутентификационное испытание помещается в узел **InteractiveChallengeResponse**. Атрибут **RefID** должен иметь значение, совпадающее со значением этого же атрибута из RSTR сообщения, полученного от ЦИ на предыдущем шаге.

4. Ответ ЦИ, содержащий маркер безопасности.

Если процедура вторичной аутентификации завершена успешна, ЦИ вернёт маркер безопасности в RSTR сообщении.

```
<s:Envelope xmlns:a="http://www.w3.org/2005/08/addressing"
xmlns:s="http://www.w3.org/2003/05/soap-envelope">
      <s:Header>
            <a:Action s:mustUnderstand="1">http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTRC/IssueFinal</a:Action>
      </s:Header>
      <s:Body>
            <trust:RequestSecurityTokenResponseCollection
xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
                  <trust:RequestSecurityTokenResponse>
                        <trust:KeySize>256</trust:KeySize>
                        <trust:Lifetime>
                              <wsu:Created xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2013-11-
07T14:25:53.327Z</wsu:Created>
                              <wsu:Expires xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2013-11-
07T14:35:53.327Z</wsu:Expires>
                        </trust:Lifetime>
                        <wsp:AppliesTo
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
                              <wsa:EndpointReference
xmlns:wsa="http://www.w3.org/2005/08/addressing">

      <wsa:Address>http://localhost/SignService/</wsa:Address>
                              </wsa:EndpointReference>
                        </wsp:AppliesTo>
                        <trust:RequestedSecurityToken>
                            ....
                        </trust:RequestedSecurityToken>
                        <trust:RequestedProofToken>
                              <trust:BinarySecret/>
                        </trust:RequestedProofToken>
                        <trust:RequestedAttachedReference>
                              <o:SecurityTokenReference
k:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1" xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-
secext-1.1.xsd" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">
                                    <o:KeyIdentifier
ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.0#SAMLAssertionID">_1ee3984d-a9f6-4d68-ab1b-08ce3889b48e</o:KeyIdentifier>
```

```
                              </o:SecurityTokenReference>
                        </trust:RequestedAttachedReference>
                        <trust:RequestedUnattachedReference>
                              <o:SecurityTokenReference
k:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1" xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-
secext-1.1.xsd" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">
                                    <o:KeyIdentifier
ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.0#SAMLAssertionID">_1ee3984d-a9f6-4d68-ab1b-08ce3889b48e</o:KeyIdentifier>
                              </o:SecurityTokenReference>
                        </trust:RequestedUnattachedReference>

      <trust:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</trust:TokenType>
                        <trust:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</trust:RequestType>
                        <trust:KeyType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/SymmetricKey</trust:KeyType>
                  </trust:RequestSecurityTokenResponse>
            </trust:RequestSecurityTokenResponseCollection>
      </s:Body>
</s:Envelope>
```

Маркер безопасности содержится в элементе **RequestedSecurityToken** (в примере содержимое маркера удалено), содержимое данного узла непрозрачно для клиентского приложения. Маркер доступа должен быть включён в заголовок SOAP сообщения, как **новый** XML-узел, в том же виде, в котором был получен в RSTR сообщении от ЦИ.

Дополнительно в RSTR возвращается срок действия маркера (элемент **Lifetime**), идентификатор маркера (**RequestedAttachedReference**, значение данного узла можно использовать для ссылки на криптографический ключ из маркера доступа).

Криптографический ключ для клиентского приложения возвращается в узле **RequestedProofToken**. Его значение необходимо использовать для защиты сообщений к сервису подписи.

6.3.2. Протокол OAuth 2.0

Для аутентификации пользователя и авторизации запроса к REST-интерфейсу СЭП клиентское приложение (далее клиент) обязано использовать протоколы OpenId Connect 1.0 и OAuth 2.0.

ЦИ КриптоПро DSS поддерживает следующие сценарии авторизации.

1. Авторизация с использованием кода авторизации.

Подробное описание данного сценария можно найти в стандарте по ссылке https://tools.ietf.org/html/rfc6749#section-4.1.

Данный сценарий используется для получения маркеров доступа и маркеров обновления. В этом сценарии клиентское приложение может проходить аутентификацию.

Для получения кода авторизации клиент должен перенаправить браузер пользователя по URL следующего вида:

```
https://dss.cryptopro.ru/STS/oauth/authorize?client_id=dss.sample&response_type=
code&scope=dss&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto
```

Параметрами данного URL являются:

- **redirect_uri** – зарегистрированный на Центре Идентификации адрес возврата (по этому адресу будет возвращён запрошенный код авторизации). Допустимые значения данного параметра сохраняются в ЦИ на этапе регистрации клиента.
- **client_id** – идентификатор клиента. Допустимое значения данного параметра сохраняется в ЦИ на этапе регистрации клиента.

- **response_type** – в данном сценарии всегда должен иметь значение **code**.
- **scope** – области использования маркера. Должен содержать значение **dss**.
- **resource –** идентификатор ресурса, для доступа к которому выпускается маркер безопасности.

В ответ на данный запрос ЦИ отобразит пользователю страницу аутентификации. Процесс аутентификации может занять несколько шагов и состоять из множества перенаправлений на другие ресурсы. После успешной аутентификации ЦИ сформирует следующий ответ, который отправит на **redirect_uri**:

Пример ответа сервера авторизации:

```
HTTP/1.1 302 Found
Location: urn:ietf:wg:oauth:2.0:oob:auto#code=c86322fd3a4cf85fb51249ab3fb4fcd1
Content-Length: 0
```

В этом запросе клиенту требуется из URI, содержащегося в заголовке **Location**, извлечь фрагмент (часть URI после символа #) и получить значения параметра **code**.

Для получения маркера доступа используется конечная точка **/token**. Клиент формирует следующий HTTP-запрос:

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Accept: application/json
Authorization: Basic d3BmLmh5YnJpZDpzZWNyZXQ=
Content-Type: application/x-www-form-urlencoded
Host: cloudcsp.cp.ru:4430
Content-Length: 266

grant_type=authorization_code&code=c86322fd3a4cf85fb51249ab3fb4fcd1&redirect_uri
=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto&client_id=dss.sample
```

В заголовке Authorization HTTP-запроса клиент может передать свои аутентификационные данные.

В результате обработки данного запроса ЦИ вернёт следующий ответ (значение маркера доступа было урезано для большей наглядности примера):

```
HTTP/1.1 302 Found HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 7069
Content-Type: application/json; charset=utf-8
Expires: -1
Date: Fri, 23 Sep 2016 10:06:12 GMT

{"access_token":"eyJ0…","expires_in":300}
```

Клиент должно использовать Access Token в том виде, в котором он был возвращён сервером авторизации, данный маркер для приложения непрозрачен. Клиент не должен пытаться его декодировать.

Клиенту следует обращать внимание на поле expires_in в ответе сервера авторизации. Данное поле содержит срок действия маркера доступа в секундах. При истечении срока действия, клиент должен запросить новый маркер.

2. Авторизация с использованием неявного разрешения

Подробное описание данного сценария можно найти в стандарте по ссылке https://tools.ietf.org/html/rfc6749#section-4.2.

В этом сценарии клиент не проходит аутентификацию. Для получения маркера доступа клиентское приложение должно перенаправить браузер пользователя по URL следующего вида:

```
https://dss.cryptopro.ru/STS/oauth/authorize?client_id=dss.sample&response_type=
token&scope=dss&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto&
resource=http%3A%2F%2Fsga10.cp.ru%2FDssSignServer%2Frest
```

Параметрами данного URL являются

- **redirect_uri** – зарегистрированный на Центре Идентификации адрес возврата (по этому адресу будет возвращён запрошенный код авторизации). Допустимые значения данного параметра сохраняются в ЦИ на этапе регистрации клиента.
- **client_id** – идентификатор клиента. Допустимое значения данного параметра сохраняется в ЦИ на этапе регистрации клиента.
- **response_type** – в данном сценарии всегда должен иметь значение **token**.
- **scope** – области использования маркера. Должен содержать значение **dss**.
- **resource –** идентификатор ресурса, для доступа к которому выпускается маркер безопасности.

В ответ на данный запрос ЦИ отобразит пользователю страницу аутентификации. Процесс аутентификации может занять несколько шагов и состоять из множества перенаправлений на другие ресурсы. После успешной аутентификации Центр Идентификации сформирует следующий ответ, который отправит на **redirect_uri**:

```
HTTP/1.1 200 OK
Content-Length: 4927
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html ng-app="app" ng-csp ng-controller="LayoutCtrl">
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>IdentityServer3</title>
    <link href='/IdentityServer3/core/assets/styles.min.css' rel='stylesheet'>

</head>
<body lang="en">
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="navbar-header">
            <a href="/IdentityServer3/core/">
                <span class="navbar-brand">IdentityServer3</span>
            </a>
        </div>
        <ul class="nav navbar-nav" ng-show="model.currentUser" ng-cloak>
            <li class="dropdown">
                <a href="#" class="dropdown-toggle" data-
toggle="dropdown">{{model.currentUser}} <b class="caret"></b></a>
                <ul class="dropdown-menu">
                    <li><a href="{{model.logoutUrl}}">Logout</a></li>
                    <li class="divider" ng-
show="model.loginWithDifferentAccountUrl"></li>
                    <li><a href="{{model.loginWithDifferentAccountUrl}}" ng-
show="model.loginWithDifferentAccountUrl">Login With Different Account</a></li>
                </ul>
            </li>
        </ul>
    </div>

    <div class='container page-authorizeresponse' ng-cloak>
        <div class="page-header">
    <h1>Please wait...</h1>
</div>

<div class="row">
    <div class="col-md-6 col-sm-6">
        <form method="post" action="http://localhost/wpf.hybrid">
            <input type="hidden" name="access_token" value="eyJ0…" />
            <input type="hidden" name="token_type" value="Bearer" />
            <input type="hidden" name="expires_in" value="3600" />
```

```
            <input type="hidden" name="scope" value="offline_access dss" />
        </form>
    </div>
</div>
    </div>

    <script id='modelJson'
type='application/json'>{&quot;siteUrl&quot;:&quot;https://cloudcsp/IdentityServer3/core/&quot;,&quot;siteName&quot;:&quot;IdentityServer3&quot;,&quot;currentUser&quot;:null,&quot;logoutUrl&quot;:null,&quot;custom&quot;:null}</script>
    <script src="/IdentityServer3/core/assets/scripts.2.5.0.js"></script>
    <script src='/IdentityServer3/core/assets/app.FormPostResponse.js'></script>
</body>
</html>
```

Клиент должен использовать Access Token в том виде, в котором он был возвращён сервером авторизации, данный маркер для приложения прозрачен. Приложение не должно пытаться его декодировать.

Клиенту следует обращать внимание на поле expires_in в ответе сервера авторизации. Данное поле содержит срок действия маркера доступа в секундах. При истечении срока действия, необходимо запросить новый маркер.

3. Авторизация с использованием учётных данных пользователя.

Подробное описание данного сценария можно найти в стандарте по ссылке https://tools.ietf.org/html/rfc6749#section-4.3.

В этом сценарии клиент может проходить аутентификацию.

Для получения маркера доступа используется конечная точка **/token**. Клиент формирует следующий HTTP-запрос:

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Accept: application/json
Authorization: Basic d3BmLmh5YnJpZDpzZWNyZXQ=
Content-Type: application/x-www-form-urlencoded
Host: cloudcsp.cp.ru:4430
Content-Length: 266

grant_type=password&client_id=dss.sample&scope=dss&username=login&password=1&resource=http%3A%2F%2Fcloudcsp%2FDssSignServer%2Frest
```

Параметры данного запроса:

- **grant_type** - тип разрешения, в данном сценарии должен быть равен **password**.
- **client_id** – идентификатор клиента. Допустимое значения данного параметра сохраняется в ЦИ на этапе регистрации клиента.
- **scope** – области использования маркера. Должен содержать значение **dss**.
- **username –** логин пользователя.
- **password** – пароль пользователя.
- **resource –** идентификатор ресурса, для доступа к которому выпускается маркер безопасности.

В заголовке Authorization HTTP-запроса клиент может передать свои аутентификационные данные.

После успешной обработки запроса ЦИ вернёт следующий HTTP-ответ:

```
HTTP/1.1 302 Found HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 7069
Content-Type: application/json; charset=utf-8
Expires: -1
Date: Fri, 23 Sep 2016 10:06:12 GMT
```

```
{"access_token":"eyJ0…","expires_in":300}
```

Клиент должен использовать Access Token в том виде, в котором он был возвращён сервером авторизации, данный маркер для приложения прозрачен. Приложение не должно пытаться его декодировать.

Клиенту следует обращать внимание на поле expires_in в ответе сервера авторизации. Данное поле содержит срок действия маркера доступа в секундах. При истечении срока действия, необходимо запросить новый маркер.

Данный сценарий подходит для клиентских приложений с высоким уровнем доверия, так они имеют полный доступ к учётным данным пользователя. После получения маркера, клиент должен удалить учётные данные из своей временной памяти, клиент не должен сохранять пользовательские учётные данные в долговременной памяти.

4. Подтверждение операций.

ЦИ КриптоПро DSS поддерживает расширение протокола OAuth 2.0, позволяющее передавать идентификатор транзакции в запросах на авторизацию в сценариях с кодом авторизации и неявным разрешением.

Для этого в формируемый URL добавляется параметр **dss_transaction_token_id** со значением идентификатора транзакции (см. п. 6.2.2):

```
https://dss.cryptopro.ru/STS/oauth/authorize?client_id=dss.sample&response_type=
code&scope=dss&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto&
resource=http%3A%2F%2Fsga10.cp.ru%2FDssSignServer%2Frest&dss_transaction_token_i
d=E2E15B25-CDEE-43B3-9F98-3B45245CD635
```

Пример.

1. Отправка запроса на создание транзакции (для аутентификации используется предварительно полученный маркер доступа).

Формат запроса описан в пункте 3.7.4.

Параметры транзакции:

- SignatureType – тип подписи.
- **CertificateID –** идентификатор сертификата.
- **DocumentInfo –** название документа.

```
POST http://sga10.cp.ru/DssSignServer/rest/api/transactions/ HTTP/1.1
Content-Type: application/json; charset=utf-8
Authorization: Bearer eAy…
Content-Length: 1477
Host: sga10.cp.ru

{"OperationCode":”SignDocument”,"Parameters":[{"Name":"SignatureType","Value":"C
MS"},{"Name":"CertificateID","Value":"6444"},{"Name":"DocumentInfo","Value":"Sig
ma"}],"Document":"MS…”
```

2. Получение идентификатора транзакции.

В случае успешного запроса сервис вернёт идентификатор транзакции.

```
HTTP/1.1 200 OK
Content-Length: 38
Content-Type: application/json; charset=utf-8
Server: Microsoft-IIS/10.0
Date: Tue, 07 Feb 2017 09:12:55 GMT

"b08d8043-a029-461e-b063-ca9d0d9ca396"
```

3. Отправка запроса на подтверждение транзакции.

Клиент формирует запрос на подтверждение, добавляя к обычному OAuth-запросу на авторизацию информацию об идентификаторе транзакции.

```
https://dss.cryptopro.ru/STS/oauth/authorize?client_id=dss.sample&response_type=
code&scope=dss&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto&
resource=http%3A%2F%2Fsga10.cp.ru%2FDssSignServer%2Frest&dss_transaction_token_i
d=b08d8043-a029-461e-b063-ca9d0d9ca396
```

4. Получение маркера доступа.

В случае успешного подтверждения операции, сервис сформирует новый маркер доступа, который следует использовать при выполнении операции.

Далее приведён пример для сценария с кодом авторизации.

```
HTTP/1.1 302 Found
Cache-Control: no-cache
Pragma: no-cache
Expires: -1
Location:
urn:ietf:wg:oauth:2.0:oob:auto?code=bea6a6f9e06768ef881c085cea0a5239&state=3bfad
8e4ce77193ecaf5fd7c4aeef7bafa221bc3acf25249c5ac1ad73eceb9ab62dcd1f4ccb94da6db83d
5e857a65cd2d59106edde16763280d8e530b4c6a7ed
Date: Tue, 07 Feb 2017 09:13:02 GMT
Content-Length: 0
```

```
POST /STS/oauth/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Accept: application/json
Content-Length: 266
Host: dss.cryptopro.ru

grant_type=authorization_code&code=bea6a6f9e06768ef881c085cea0a5239&redirect_uri
=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto&client_id=dss.sample
```

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 10494
Content-Type: application/json; charset=utf-8
Expires: -1
Date: Tue, 07 Feb 2017 09:13:03 GMT

{"access_token":"ey… ","expires_in":300}
```

5. Вызов подтверждаемой операции.

После получения маркера доступа клиент отправляет запрос сервису подписи. Особенностью выполнения операции с подтверждением является то, что передавать содержимое документа в параметрах запроса не нужно, так как оно уже сохранено на сервере в параметрах транзакции.

```
POST http://dss.cryptopro.ru/SignServer/rest/api/documents/ HTTP/1.1
Content-Type: application/json; charset=utf-8
Authorization: Bearer eyJ…
Connection: Keep-Alive
Content-Length: 124
Host: dss.cryptopro.ru

{"Content":null,"Name":"Sigma","Signature":{"Type":5,"Parameters":{"IsDetached":
"false"},"CertificateId":6444,"PinCode":""}}}
```

6.3.3. Сервис подтверждения операций

ЦИ КриптоПро DSS предоставляет отдельный REST-сервис для подтверждения операций. Им могут воспользоваться клиенты, не способные провести авторизацию по протоколу OAuth (например, из-за ограничения на использования браузера или web-view).

Данный сервис реализует одну конечную точку **Confirmation**, её описание приведено Таблица 99.

Таблица 99. Описание методов конечной точки Confirmation.

HTTP метод	Путь	Описание
POST	/confirmation	Запрос на подтверждение операции.
Параметры	Тип	Описание
	RequestConfirmation	
Возвращаемое значение	Тип	Описание
	ConfirmationResponse	

Данная конечная точка реализует процесс подтверждения операций, похожий на процедуру User Interactive Challenge из стандарта WS-Trust 1.4 (см. п. 6.3.1). Для подтверждения операции клиент и сервис обмениваются сообщениями RequestConfirmation и ConfirmationResponse, первое содержит ответ на аутентификационное испытание, второе запрос на аутентификационное испытание. В результате успешного подтверждения сервер в последнем ConfirmationResponse вернёт маркер доступа. Его клиент должен будет использовать для аутентификации при выполнении подтверждаемой операции.

В Таблица 100 представлено описание полей класса RequestConfirmation.

Таблица 100. Описание полей класса RequestConfirmation.

Поле	Тип	Описание
TransactionTokenId	string	Идентификатор транзакции.
Resource	string	Идентификатор ресурса, для которого будет выпущен маркер доступа (аналог параметра AppliesTo в RST-сообщении протокола WS-Trust).
ChallengeResponse	InteractiveChallengeResponse	Ответ на аутентификационное испытание (см. Таблица 98).

В таблице Таблица 101 представлено описание полей класса ConfirmationResponse.

Таблица 101. Описание полей класса ConfirmationResponse

Поле	Тип	Описание
TransactionTokenId	string	Идентификатор транзакции.
Challenge	InteractiveChallenge	Запрос на выполнение аутентификационное испытание (см. Таблица 97).

Поле	Тип	Описание
AccessToken	string	Маркер доступа.
IsFinal	bool	Является ли данный ответ последним в процессе подтверждения.
IsError	bool	Содержит ли данный ответ ошибку обработки запроса.
Error	string	Ошибка обработки запроса.
ErrorDescription	string	Подробное описание ошибки обработки запроса.

6.3.4. Получение результатов подтверждения операции

Результат подтверждения операции может быть возвращён синхронно в ответ на отправку ответа на аутентификационное испытание, либо асинхронно.

Асинхронно результат подтверждения возвращается при использовании аутентификации через мобильное приложение myDSS.

Для получения результатов асинхронного подтверждения прикладная система может использовать два варианта взаимодействия с ЦИ КриптоПро DSS.

1. Периодический опрос ЦИ о состоянии транзакции.
2. Использование оповещения об изменении состояния транзакции.

В первом случае для получения текущего статуса транзакции прикладная система отправляет запросы на конечную точку **/confirmation** следующего вида:

Запрос

```
POST https://dss.cryptopro.ru/STS/confirmation HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGc…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 171

{"Resource":"https://dss.cryptopro.ru/SignServer/rest/api","ChallengeResponse":{
"TextChallengeResponse":[{"RefId":"fdd05368-acf1-4372-bea2-b51a77ea1923"}]}}
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 2657
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:15:17 GMT

{"IsFinal":false,"IsError":false}
```

В случае если транзакция ещё находится в обработке, значение поля IsFinal будет выставлено в false. В случае перехода транзакции в конечное состояние (ошибка или успех) поле IsFinal примет значение true. В случае ошибки подтверждении транзакции поле IsError будет заполнено значение true, а поле Error будет содержать сообщение об ошибке.

Для того чтобы постоянно не опрашивать ЦИ о статусе транзакции можно использовать функционал оповещения через callback. Для этого на стороне прикладной системы должны быть реализован веб-сервис, способный обработать HTTP POST запрос со следующим содержимым в теле:

```
{"Result":"success","TransactionId":"9ef977c4-d867-440c-8918-
97fce5f16e1d","Error":""}
```

Поле Result будет содержать success в случае успешного подтверждения и failed в случае ошибки или отклонения, поле Error используется для передачи более подробной информации об ошибке подтверждения.

Для передачи URL адреса сервиса обработки callback'ов прикладная система должна использовать параметр CallbackUri в запросе на создание транзакции подтверждения к конечной точке /confirmation

Пример.

```
{"Resource":"https://simdss.cryptopro.ru/SignServer/rest/api","CallbackUri":"https://localhost/","ConfirmationScope":"test-confirmation-scope","ConfirmationParams":{"CpTime":"26.01.2018 16:15:42"}}
```

6.3.5. Обработка ошибок

Сервис подтверждения операций возвращает ошибки в HTTP ответе в виде JSON объекта.

Таблица 102. Описание ошибок сервиса подтверждения операций

Идентификатор ошибки	Описание
invalid_transaction	Идентификатор транзакции не задан или транзакция с указанным идентификатором не найдена.
transaction_expired	Срок действия транзакции истёк.
max_attempts_exceeded	Исчерпаны попытки подтвердить транзакцию одноразовым паролем.
transaction_pending	У пользователя уже есть неподтверждённая транзакция.
internal_error	Внутренняя ошибка сервиса при подтверждении транзакции.
authentication_failed	Ошибка аутентификации, переданы неправильные аутентификационные данные или пользователь не может быть аутентифицирован.
authentication_declined	Пользователь отказался от аутентификации.

6.4. Отображение подписываемых данных

Сервис Подписи «КриптоПро DSS» позволяет формировать отображаемое представление обрабатываемого документа, которое может быть показано пользователю в процессе подтверждения операции.

Для этого на Сервисе Подписи предусмотрен механизм плагинов, преобразующих документы в XML особого вида.

Схема XML-документа приведена ниже:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://www.cryptopro.ru/schemas/2014/08/dtbs"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:dtbs="http://www.cryptopro.ru/schemas/2014/08/dtbs">
    <xs:element name="dtbs" type="dtbs:dtbsType" />
    <xs:complexType name="dtbsType">
        <xs:sequence>
            <xs:element ref="dtbs:row" minOccurs="0" maxOccurs="unbounded" />
```

```
        </xs:sequence>
        <xs:attribute name="unattendedSign" type="xs:boolean">
            <xs:annotation>
                <xs:documentation>
                    Если 'true', то документ подписывается без отображения и
                    подтверждения пользователем. В этом случае элементы row
                    должны отсутствовать.
                </xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>

    <xs:element name="row" type="dtbs:rowType" />
    <xs:complexType name="rowType">
        <xs:sequence>
            <xs:element name="name" type="xs:string">
                <xs:annotation>
                    <xs:documentation>
                        Название ключевого поля выжимки документа
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="value" type="xs:string">
                <xs:annotation>
                    <xs:documentation>
                        Значение ключевого поля выжимки документа
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Отображение подписываемых данных происходит, когда на Сервисе Подписи включен режим подтверждения операций. Во время создания транзакции клиентское приложение передаёт на сервер содержимое документа и его тип. Если по типу документа удалось подобрать соответствующий плагин для формирования отображаемого представления, то сервис подписи сохранит их вместе со сведениями об операции.

Центр Идентификации при подтверждении операции, используя отображаемые данные, формирует текст, отображаемый пользователю в ходе процедуру аутентификационного испытания.

Например, при формировании SMS сообщения с одноразовым паролем или отображении на Веб-интерфейсе пользователя к XML-документу применяется XSL-преобразование, преобразующее его в текст вида:

```
Name1 : Value1, Name2 : Value2, … , NameN : ValueN.
```

Пример отображаемых данных:

```
<?xml version="1.0" encoding="utf-8"?>
<dtbs xmlns="http://www.cryptopro.ru/schemas/2014/08/dtbs">
    <row>
        <name>Наименование документа</name>
        <value>Платёжное поручение</value>
    </row>
    <row>
        <name>Банк получателя</name>
        <value>АКБ "Рога и копыта"</value>
    </row>
    <row>
        <name>Счёт получателя</name>
        <value>40781032100000000000</value>
    </row>
    <row>
```

```
        <name>Сумма платежа</name>
        <value>100 RUB</value>
    </row>
</dtbs>
```

6.5. Подключение стороннего центра идентификации к серверу электронной подписи «КриптоПро DSS»

В состав СЭП «КриптоПро DSS» входит компонент Центр Идентификации. Он объединяет в себе две роли службы маркеров безопасности (Security Token Service, STS):

- Служба проверяющей стороны (Relying Part STS, RP-STS).
- Служба поставщика удостоверений (Identity Provider, IP-STS).

Все компоненты СЭП доверяют только Центру Идентификации КриптоПро DSS (ЦИ). Для подключения стороннего центра идентификации (IP-STS) необходимо установить отношение доверия с ЦИ (см. Руководство Администратора).

Подключение стороннего поставщика (IP-STS) услуг возможно в пассивном сценарии. Взаимодействие должно осуществляться согласно спецификации **WS-Federation Passive Requestor Profile**. Спецификация находится по адресу

http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fedpass/ws-fedpass.pdf

Подключения стороннего центра идентификации возможно также и в активном сценарии. Для этого предоставляются следующие конечные точки:

1. http://<hostname>/STS/Active.svc/issuedtoken
2. https://<hostname>/STS/Active.svc/issuedtoken/transport

Электронный идентификатор должен быть в формате SAML 1.1

Обязательно наличие утверждений (Claim):

- http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name, в значении которого содержится идентификатор пользователя.
- http://schemas.microsoft.com/ws/2008/06/identity/claims/role, в значении которого содержится роль пользователя. Утверждение может принимать только значения: Users, Admins.

6.6. Управление запросами и сертификатами пользователей

КриптоПро DSS позволяет операторам управлять запросами и сертификатами пользователей СЭП.

6.6.1. Получение маркера для управления пользователем

Для управления пользователем СЭП администратор должен предоставить ЦИ КриптоПро DSS маркер безопасности, содержащий помимо обязательных утверждений утверждение:

DssDelegateUserId (http://dss.cryptopro.ru/identity/claims/delegateuserid).

В нём должен содержаться идентификатор управляемого пользователя.

С маркером безопасности, полученным от ЦИ необходимо вызвать метод сервиса подписи КриптоПро DSS.

В случае, если управляемый пользователь является пользователем ЦИ КриптоПро DSS, то есть его учётные данные хранятся в БД ЦИ, администратору также необходимо получить маркер безопасности. Для этого он должен сформировать запрос на маркер (RST),

также содержащий утверждение **DssDelegateUserId** с идентификатором управляемого пользователя.

Следует отметить, что администратору, работающему в режиме управления пользователем не доступны функции по подписи и шифрованию документов, изменению пин-кода для доступа к закрытому ключу и назначение сертификата по умолчанию для пользователя.

6.6.2. «Сцепленные» учётные записи

«Сцепленная» учётная запись – это учётная запись, связанная с другой учётной записью с помощью утверждения:

DssAccountName (http://dss.cryptopro.ru/identity/claims/accountname).

ЦИ КриптоПро DSS при формировании значения утверждения DssUserId (содержащего уникальный идентификатор пользователя в БД СЭП) будет использовать не значение утверждения **Name**, а значение утверждения **DssAccountName**.

Например, с помощью «сцепленных» учётных записей можно реализовать аутентификацию устройств пользователя.

Пример:

В AD есть учетная запись *User*. Для определённого устройства пользователя заводится отдельная учётная запись *UserDevice*, которая через атрибут AD, связывается («сцепляется») с учётной записью пользователя. Пусть таким атрибутом будет атрибут Employee-Number.

В качестве значения данного атрибута задаётся идентификатор пользователя User, а в ADFS настраивается правило, которое помещает значение атрибута в утверждение DssAccountName. Дополнительно для учётной записи устройства можно настроить добавление утверждений, описывающих политику вторичной аутентификации по отношении к выполняемым действиям (см. пунк 8 Руководства администратора).

При аутентификации устройство передаёт свои учётные данные и получает от ADFS маркер, содержащий утверждение DssAccountName со значением «сцепленной» учётной записи пользователя. ЦИ КриптоПро DSS, получив такой маркер, во-первых, не запрашивает вторичную аутентификации (если сделаны соответствующие настройки), во-вторых, при формировании значения утверждения DssUserId использует не идентификатор учётной записи устройства, а идентификатор учётной записи пользователя из утверждения DssAccountName. Таким образом устройство получает возможность работать от имени пользователя.

6.7. Подтверждение произвольных операций

Возможности СЭП по вторичной аутентификации могут применять не только для подтверждения действий, требующих доступ к закрытому ключу, но и для произвольных операций.

Результатом любого подтверждения в СЭП является маркер доступ. В случае подтверждения произвольной операции в запрос на выпуск маркера вызывающая система должна включить идентификатор области использования маркера (scope). Каждой операции соответствует своя область использования.

Все области использования предварительно должны быть зарегистрированы в СЭП. У каждой области использования, требующей подтверждения, есть шаблон текстового сообщения, которое будет отображено пользователю во время аутентификации, как в вызывающей системе, так и на устройстве пользователя (в зависимости от типа аутентификации).

Шаблоны сообщений настраиваются отдельно для каждого способа аутентификации, что позволяет отправлять различный текст для различных устройств. Следует отметить, что некоторые устройства не позволяют отображать текст (например, OATH токены), для таких способов аутентификации текст может быть отображён только в вызывающей системе.

В шаблоне сообщения могут содержаться подстановочные значения (далее параметры шаблона). Значения данных параметров прикладная система обязана передать в запросе на выпуск маркер безопасности.

Формат запросов на выпуск маркера доступа отличается в зависимости от используемого интерфейса взаимодействия с ЦИ.

6.7.1. Подтверждение произвольных операций в протоколе WS-Trust

Для подтверждения операций в рамках выпуска маркера доступа по протоколу WS-Trust, вызывающая система должна дополнить запрос на выпуск маркера RequestSecuirtyToken следующими данными:

Параметр	Описание
Идентификатор области использования маркера URI: http://dss.cryptopro.ru/identity/claims/confirmationscope	В данном параметре требуется указать идентификатор области использования маркера, который был указан при её регистрации в СЭП
Параметр шаблона подтверждения URI: http://dss.cryptopro.ru/identity/claims/confirmationscopeparam	В данном параметре передаётся название и значение (разделённые символом ":") для подстановочных символов шаблона. Параметров шаблона может быть несколько в RST, повторяющиеся значения игнорируются.

Указанные параметры передаются в элементе AdditionalContext запроса на выпуск маркера.

Пример

```
            <trust:RequestSecurityToken xmlns:trust="http://docs.oasis-
open.org/ws-sx/ws-trust/200512">
                  <wsp:AppliesTo
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
                        <wsa:EndpointReference
xmlns:wsa="http://www.w3.org/2005/08/addressing">
                              <wsa:Address>https://relying-party/</wsa:Address>
                        </wsa:EndpointReference>
                  </wsp:AppliesTo>
                  <trust:KeyType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/SymmetricKey</trust:KeyType>
                  <trust:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</trust:RequestType>
                  <auth:AdditionalContext xmlns:auth="http://docs.oasis-
open.org/wsfed/authorization/200706">
                        <auth:ContextItem
Name="http://dss.cryptopro.ru/identity/claims/confirmationscope">
                              <auth:Value>test-confirmation-scope</auth:Value>
                        </auth:ContextItem>
                        <auth:ContextItem
Name="http://dss.cryptopro.ru/identity/claims/confirmationscopeparam">
                              <auth:Value>CpTime:17.01.2018
13:20:27</auth:Value>
                        </auth:ContextItem>
                  </auth:AdditionalContext>
            </trust:RequestSecurityToken>
```

В данном примере отправляется запрос на выпуск маркера, в качестве идентификатора области использования указывается **test-confirmation-scope**, в качестве параметра **CpTime**, в качестве значения параметра **17.01.2018 13:20:27**.

При использовании следующего шаблона

```
Подтверждение тестовой операции. Время {0:CpTime}
```

на устройство пользователя будет отправлено сообщение:

```
Подтверждение тестовой операции. Время 17.01.2018 13:20:27
```

ЦИ проверяет, что все указанные в шаблоне параметры переданы в запросе, и, если это не так, вернёт ошибку. В запросе может быть указана только одна область использования маркера.

6.7.2. Подтверждение произвольных операций в протоколе Oauth 2.0

В протоколе OAuth 2.0 область использования маркера указывается в authorize запросе в параметре **scope**, а подстановочные значения в параметре **dss_scope_params**.

Пример.

```
GET /STS/oauth/authorize?client_id=eea2fd3f-5c70-4d74-a594-
f1e7bf81b4d7&response_type=code&scope=openid+offline_access+dss+test-
confirmation-
scope&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto&state=0b9ffa4845ae
d3bca7fdea5728e088861e71fa70d23d1530b6a8682116ac5f49b1679f24b8788e232a289eaf6ae0
6bccd3c61376abf06cc8833e3bc9e61726ce&nonce=9ac65eef4760eb2e424631548b4bfc848ec2a
f6a0c72c69f003ce98e93470176744a9c245f8fe0cc23d67d26134f8110c0ab44e3dcfa7a8dce98f
2958bb70471&code_challenge=qGKCFb9rRn4cIpBnRw8uYiUUX0bdCEWaj3jbz919slY&code_chal
lenge_method=S256&resource=https%3A%2F%2Fsimdss.cryptopro.ru%2FSignServer%2Frest
%2Fapi&dss_scope_params=eyJDcFRpbWUiOiIxNy4wMS4yMDE4IDE0OjQ5OjU1In0 HTTP/1.1
```

Подстановочные параметры должны быть предварительно закодированы по следующему алгоритму:

1. Все параметры и их значения представить в виде JSON объекта, в котором название свойства соответствует названию параметра, а значение свойства – значению параметра.
2. Полученный JSON-объект представить, как строку в кодировке UTF-8, и преобразовать в массив байтов.
3. Полученный массив байтов преобразовать в строку с помощью алгоритма BASE64URL.

Пример

Пусть в шаблоне требуется задать параметр **CpTime** со значением **17.01.2018 14:49:55**. Соответствующий этим данным JSON объект будет выглядеть

```
{"CpTime":"17.01.2018 14:49:55"}
```

В кодировке BASE64URL

```
eyJDcFRpbWUiOiIxNy4wMS4yMDE4IDE0OjQ5OjU1In0
```

При использовании следующего шаблона

```
Подтверждение тестовой операции. Время {0:CpTime}
```

на устройство пользователя будет отправлено сообщение:

```
Подтверждение тестовой операции. Время 17.01.2018 13:20:27
```

ЦИ проверяет, что все указанные в шаблоне параметры переданы в запросе, и, если это не так, вернёт ошибку. В запросе может быть указана только одна область использования маркера.

6.7.3. Подтверждение произвольных операций в протоколе строго подтверждения

Для подтверждения произвольных операций через конечную точку StrictConfirmation необходимо передавать идентификатор области использования и параметры шаблона сообщения в запросе RequestConfirmation.

Пример:

```
POST /STS/confirmation HTTP/1.1
Authorization: Bearer eyJ0…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru

{"Resource":"https://dss.cryptopro.ru/SignServer/rest/api","ConfirmationScope":"
test-confirmation-scope","ConfirmationParams":{"CpTime":"17.01.2018 17:54:02"}}
```

В данном примере в качестве идентификатора области использования передаётся значение **test-confirmation-scope**, а в качестве значения параметра **CpTime 17.01.2018 17:54:02**.

При использовании следующего шаблона

```
Подтверждение тестовой операции. Время {0:CpTime}
```

на устройство пользователя будет отправлено сообщение:

```
Подтверждение тестовой операции. Время 17.01.2018 13:20:27
```

ЦИ проверяет, что все указанные в шаблоне параметры переданы в запросе, и, если это не так, вернёт ошибку. В запросе может быть указана только одна область использования маркера.

7. Примеры встраивания

Примеры взаимодействия с КриптоПро DSS находятся на дистрибутивном диске в папке DssSamples, либо доступны для скачивания по ссылке Примеры встраивания для разработчиков.

Структура папок DssSamples:

Externals	Служебные сборки.
Net\DSSWSTrust	Реализация расширения протокола WS-Trust. Пример демонстрирует расширение ф-ий .NET классов WSTrustChannel и WSTrustChannelFactory для поддержки двухфакторной аутентификации в КриптоПро DSS и подтверждения подписи.
Net\CSharpDssClient	Оконное приложение, пример клиента КриптоПро DSS. Пример демонстрирует использование классов DSSWSTrustChannelFactory, DSSWSTrustChannel при аутентификации на сервере подписи, получение списка сертификатов и подпись документа.
Net\HttpPostAPI	Пример веб-сайта, настроенного на взаимодействие в КриптоПро DSS через HTTP API. Демонстрирует создание запроса на сертификат и подпись документа.
Java\DSSWSTrustJava	Реализация расширения протокола WS-Trust. Пример демонстрирует расширение библиотеки Metro (Java) для поддержки двухфакторной аутентификации в КриптоПро DSS и подтверждения подписи.
Java\JavaDssClient	Оконное приложение, пример клиента КриптоПро DSS для Java платформы. Пример демонстрирует использования класса DSSWSTrustChannel при аутентификации на сервере подписи, получение списка сертификатов и подпись документа.

7.1. Взаимодействие с КриптоПро DSS на платформе .NET

Проект DSSWSTrust содержит реализацию классов DSSWSTrustServiceChannelFactory и DSSWSTrustServiceChannel, расширяющих .NET классы WSTrustServiceChannelFactory и WSTrustServiceChannel поддержкой двухфакторной аутентификации. Данные классы используются для взаимодействия с Центром Идентификации «КриптоПро DSS» и получения маркера безопасности.

В сборке DSS.Common.dll размещены классы DSSRequestSecurityToken и DSSRequestSecurityTokenResponse для работы с сообщениями RST/RSTR по спецификации WS-Federation. Данные классы наследованы от соответствующих классов из пространства имён System.IdentityModel.Protocols.WSTrust и дополнительно поддерживают процедуру User Interactive Challenge.

7.1.1. Формирование запроса на маркер безопасности

Объект класса System.IdentityModel.Protocols.WSTrust.RequestSecurityToken используется для формирования запроса на маркер безопасности.

В Таблица 103 представлены свойства класса, которые необходимо заполнить перед отправкой запроса службе.

Таблица 103. Обязательные свойства RequestSecurityToken.

Свойство	Тип данных	Значение
RequestType	string	http://schemas.microsoft.com/idfx/requesttype/issue
AppliesTo	EndpointReference	Адрес ресурса, для которого предназначен запрашиваемый маркер безопасности

Дополнительно клиентское приложение может заполнить свойство AdditionalContext. Данное свойство содержит коллекцию объектов класса ContextItem. В коллекцию можно включить выжимку документа, которая будет отображена в SMS сообщении с одноразовым паролем. Элемент ContextItem должен в этом случае иметь URI http://dss.cryptopro.ru/identity/confirmationtext.

Пример:

```
var rst = new RequestSecurityToken();
rst.RequestType = RequestTypes.Issue;
rst.KeyType = KeyTypes.Symmetric;
rst.AppliesTo = new EndpointReference(
    "http://localhost/SignServer/SignService.svc/issuedtoken");
rst.Claims.Dialect =
    "http://docs.oasis-open.org/wsfed/authorization/200706/authclaims";
rst.Claims.Add(
    new RequestClaim(
        CustomClaimTypes.DssTransactionTokenId,
        false,
        transactionID));
```

Запрос на выпуск маркера безопасности может содержать утверждения, которые должны быть добавлены в выпущенный маркер. Эти утверждения передаются через свойство Claims класса RequestSecurityToken. При наличии в коллекции утверждения http://dss.cryptopro.ru/identity/claims/dsstransactionid со значением идентификатора транзакции будет инициирован процесс подтверждения транзакции.

В ответ на запрос маркера безопасности, служба маркеров безопасности не обязана вернуть маркер. Она может потребовать у клиента дополнительные данные. Такими дополнительными данными является одноразовый пароль, отправленный пользователю.

При использовании двухфакторной аутентификации для подписи документа Центр Идентификации возвращает объект класс DSSRequestSecurityTokenResponse, содержащий элемент Challenge.

В ответ на запрос одноразового пароля, клиент отправляет DSSRequestSecurityTokenResponse, содержащий элемент ChallengeResponse.

Пример:

```
var dssRSTR = new DSSRequestSecurityTokenResponse();
dssRSTR.RequestType = RequestTypes.Issue;
dssRSTR.AppliesTo = new EndpointReference("http://localhost");
var otp = new TextChallengeResponseType[1];
string answer = Console.ReadLine();
otp[i] = new TextChallengeResponseType()
{
  RefID = stsRSTR.Challenge.TextChallenge[0].RefID
  Value = answer,
};
dssRSTR.ChallengeResponse = new InteractiveChallengeResponseType();
dssRSTR.ChallengeResponse.TextChallengeResponse = otp;
```

В данном примере одноразовый пароль считывается из консоли.

Если проверка одноразового пароля завершилась успешно, служба вернёт RequestSecurityTokenResponse содержащий запрошенный маркер безопасности. Свойство IsFinal при этом будет установлено в true. Если клиент ввёл неверный одноразовый пароль,

то RequestSecurityTokenResponse будет аналогичен самому первому ответу (поле Label в этом случае может содержать информацию об ошибке), свойство isFinal будет установлено в false.

Процедура обмена сообщениями с STS может продолжать до тех пор, пока не будет превышено количество попыток ввода одноразового пароля.

В случае успешной проверки одноразового пароля, служба выпустит маркер безопасности.

7.1.2. Взаимодействие с Центром Идентификации КриптоПро DSS

В проекте DSSWSTrust реализован класс DSSWSTrustServiceChannel для работы с Центром Идентификации. Для взаимодействия с ЦИ класс DSSWSTrustServiceChannel предоставляет метод Issue. Метод Issue отправляет запрос к Центру Идентификации на выпуск маркера безопасности.

1. SecurityToken Issue()

Данный метод формирует простейший запрос на выпуск маркера, передавая только обязательные параметры запроса. Если на Центре Идентификации включено подтверждение входа, то у пользователя будет запрошен одноразовый пароль, иначе выпуск маркера произойдёт прозрачно.

2. SecurityToken Issue(string transactionID, string confirmationText, bool formatted = false) Используется при подтверждения подписи документа. Необходимо передать следующие параметры:

- идентификатор транзакции, полученный от Сервера Подписи,
- текст подтверждения, который следует отобразить пользователю в SMS сообщении с одноразовым паролем,
- формат текста подтверждения (см. раздел 0)

3. SecurityToken Issue(RequestSecurityToken rst)

Данный метод позволяет передать на ЦИ самостоятельно сформированный запрос на выпуск маркера.

4. SecurityToken Issue(RequestSecurityToken rst, out RequestSecurityTokenResponse rstr)
5. SecurityToken Issue(RequestSecurityTokenResponse rst, out RequestSecurityTokenResponse rstr)

Метод (4) инициирует процесс получения маркера безопасности, в случае если подтверждение транзакции не требуется, он сразу вернёт объект SecurityToken, а свойство IsFinal объекта RequestSecurityTokenResponse будет иметь значение true. В случае если от пользователя требуется ввод одноразового пароля, то возвращаемый параметр rstr будет содержать свойство Challenge. Методы (4) и (5) необходимо использовать, если требуется контроль над процессом выпуска маркера безопасности.

При вызове методов (1), (2), (3) в случае использования двухфакторной аутентификации будет осуществлён вызов специальной функции обратного вызова, которая вернёт одноразовый пароль (см. пункт 4.2.4).

Для того чтобы получить экземпляр класса DSSWSTrustServiceChannel необходимо использовать фабрику DSSWSTrustServiceChannelFactory (реализован в проекте DSSWSTrust). Класс DSSWSTrustServiceChannelFactory наследован от .NET класса System.ServiceModel.Security.WSTrustServiceChannelFactory.

Данный класс служит для настройки соединения с Центром Идентификации. Он позволяет задать:

- Учётные данные для аутентификации на ЦИ (логин/пароль).

- Функцию обратного вызова для ввода одноразового пароля.

Пример:

```
var factory = new DSSWSTrustServiceChannelFactory();
factory.Credentials.UserName.UserName = "login";
factory.Credential.UserName.Password = "password";
```

У объекта DSSWSTrustServiceChannelFactory есть свойство OtpCallback, имеющее тип

```
public delegate string OtpCallbackDelegate(string message);
```

С помощью данного свойства можно задать callback-функцию, которая будет вызвана для запроса у пользователя одноразового пароля. В параметре message данной функции будет передано сообщение, которое следует отобразить пользователю.

Пример:

```
var factory = new DSSWSTrustServiceChannelFactory();
factory.OtpCallback = delegate(string message)
{
  Console.WriteLine(message);
  return Console.ReadLine()
}
```

В данном примере в качестве callback-функции используется анонимный метод, который считывает введённый пользователем одноразовый пароль из консоли.

Callback-функция будет использоваться при вызове методов Issue (1), (2), (3).

Для того чтобы создать канал для взаимодействия с ЦИ DSS необходимо вызвать метод CreateChannel().

Пример:

```
var factory = new DSSWSTrustServiceChannelFactory();
var dssStsClient = factory.CreateChannel();
dssStsClient.Issue(rst);
```

7.2. Взаимодействие с КриптоПро DSS на платформе Java

Проект DSSWSTrustJava содержит пример реализации классов для взаимодействия с Центром Идентификации и получения маркера безопасности с использованием двухфакторной аутентификации.

7.2.1. Класс DSSWSTrustChannel

Класс DSSWSTrustChannel предоставляет следующие методы для запроса маркера безопасности:

1. Token Issue()

Данный метод формирует простейший запрос на выпуск маркера. Если на Центре Идентификации включено подтверждение входа, то у пользователя будет запрошен одноразовый пароль, иначе выпуск маркера произойдёт прозрачно.

2. Token Issue(String transactionID, String confirmationText, boolean formatted)

Используется для подтверждения подписи документа. При вызове метода необходимо передать следующие параметры:

- идентификатор транзакции, полученный от Сервера Подписи,
- текст подтверждения, который следует отобразить пользователю в сообщении с одноразовым паролем,
- формат текста подтверждения (см. раздел 0).

3. Token Issue(String transactionID, String confirmationText)

Используется для подтверждения подписи документа. При вызове метода необходимо передать следующие параметры:

- идентификатор транзакции, полученный от Сервера Подписи,
- текст подтверждения, который следует отобразить пользователю в сообщении с одноразовым паролем,

Формат текста подтверждения – неформатированный.

4. Token Issue(string transactionID, string confirmationText, boolean formatted, boolean shareToken)

Аналогичен (2). Позволяет указать, следует ли добавлять выпущенный маркер в кэш для последующего использования в вызовах сервиса. Это позволяет получить маркер заранее, а затем использовать его для доступа к серверу подписи.

В методах (1), (2), (3) маркер автоматически добавляется в кэш.

7.2.2. Создание и настройка экземпляра класса DSSWSTrustChannel.

Для создания экземпляра класса **DSSWSTrustChannel** необходимо указать адрес Центра Идентификации.

Кроме того, данный класс позволяет настроить соединение и задать параметры доступа к Центру Идентификации:

- Логин/пароль для доступа к службе,
- Обработчик события запроса одноразового пароля (OtpCallback),

Пример:

```
DSSWSTrustChannel stsChannel = new
DSSWSTrustChannel("http://dss.cryptopro.ru/STS/Active.svc");
stsChannel.setCallbackHandler(new OtpCallbackHandlerImpl());
tsChannel.setUserName("login");
stsChannel.setPassword("password");
```

Обработчик OtpCallbackHandlerImpl должен реализовывать интерфейс **OtpCallbackHandler**:

```
public interface OtpCallbackHandler {
    void handle(TextInputCallback callback);
}
```

TextInputCallback содержится в пакете javax.security.auth.callback.

С помощью данного обработчика пользователю можно отобразить сообщение о вводе одноразового пароля и передать пароль в **DSSWSTrustChannel**.

7.2.3. Вызов методов Сервера Подписи

Для вызова методов сервиса необходимо получить соответствующий прокси-класс, которому передать текущие параметры конфигурации **DSSWSTrustChannel**. Это можно сделать с помощью класса **com.sun.xml.ws.security.trust.STSIssuedTokenFeature**.

Пример:

```
private static ISignService getSignServerPort(
         STSIssuedTokenConfiguration stsConfig) {
     SignService srv = new SignService();
     ISignService portToken = srv.getPort(
                new QName("http://dss.cryptopro.ru/services/2014/06/",
"WSHttpBinding_ISignService"),
                ISignService.class,
```

```
                new WebServiceFeature[]{
                    new STSIssuedTokenFeature(stsConfig)
                });
return portToken;
}
. . . . . .
ISignService port = getSignServerPort(stsChannel.getCurrentConfig());
```

Метод **getCurrentConfig** возвращает текущие параметры конфигурации **DSSWSTrustChannel**.

8. Интеграция с внешними системами

8.1. Общий сценарий использования REST API

8.1.1. Общие сведения

Данный сценарий описывает типовую последовательность действий для подписи документа в СЭП (Сервер Электронной Подписи) с использованием интерфейса взаимодействия REST, аутентификации по протоколу OAuth 2.0/OpenId Connect 1.0, без подтверждения операций.

8.1.1.1. Исходные данные

Для работы с DSS через REST API вызывающее приложение должно:

- знать адрес сервера DSS. Далее в документе и во всех примерах этот параметр будет называться **host**. Следует иметь в виду, что СЭП состоит из нескольких компонентов, каждый из которых может находится на отдельном сервере, в данном документе такой сценарий не рассматривается. Далее в примерах в качестве **host** будет использоваться dss.cryptopro.ru
- Обладать сертификатом Оператора DSS. Оператор DSS – это особенная роль, предназначенная для управления учётными записями пользователей и их сертификатами. Для выполнения некоторых действий требуется аутентификация с установлением двустороннего TLS соединения.
- Знать имена приложений-компонентов DSS. Далее эти параметры будут называться **SsAppName** и **StsAppName** для обозначения приложений Сервиса Подписи и Центра Идентификации соответственно. Таким образом, URL адрес запросов к этим компонентам будет иметь вид: https://host/SsAppName и https://host/StsAppName. По умолчанию **SsAppName** имеет значение SignServer, **StsAppName** – STS, эти же значения будут использоваться в примерах.
- Знать состав различительного имени субъекта сертификата. Различительное имя субъекта – это то, что заносится в поле «Кому выдан» сертификата.
- Знать требуемый шаблон сертификата или уметь предоставить пользователю выбор нужного шаблона. Приложение может получить от DSS перечень допустимых шаблонов.
- Знать идентификатор зарегистрированного в СЭП обработчика УЦ или уметь предоставить пользователю выбор нужного обработчика.
- Знать параметры протокола OAuth 2.0: идентификатор клиента (client_id) и адрес перенаправления (redirect_uri). Эти параметры необходимы для выполнения запросов на выпуск маркеров доступа к ресурсам СЭП.

8.1.1.2. Примеры запросов

Описания шагов снабжены примерами HTTP-запросов. Все запросы и ответы могут содержать заголовки, не указанные в примерах. Значения маркеров доступа в заголовках аутентификации приводится в урезанном виде для удобства чтения.

8.1.2. Основные сценарии взаимодействия

К основным сценариям взаимодействия относятся:

- Регистрация пользователя, назначение пароля и схемы аутентификации;
- формирование сертификата;
- подписание документа от имени пользователя;
- Удаление сертификатов пользователя;
- Удаление учётной записи пользователя.

В данном разделе приводится описание основных схем взаимодействия.

8.1.2.1. Регистрация пользователя

Исходные данные: логин пользователя, сертификат аутентификации оператора DSS.

Регистрация пользователя выполняется от имени Оператора DSS. Оператор аутентифицируется в DSS по сертификату.

Для того чтобы зарегистрировать пользователя необходимо выполнить следующую последовательность действий:

1. Создать нового пользователя в DSS.
2. Сбросить пароль.
3. Назначить схему аутентификации.

8.1.2.1.1. Создание нового пользователя

Для создания нового пользователя необходимо отправить запрос следующего вида на конечную точку StsAppName/ums/user

Запрос

```
POST /STS/ums/user HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 34

{"Login":"my-dss-test-1488312836"}
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 38

"20948431-2f7e-4825-b723-d56ea134298c"
```

В ответе содержится уникальный идентификатор пользователя DSS.

8.1.2.1.2. Сброс пароля

Для задания пароля пользователю требуется выполнить операцию сброса пароля. В результате выполнения этой операции сервис создаст новый случайный пароль, назначит его пользователю и вернёт вызывающему приложению в HTTP запросе.

Запрос

```
POST /STS/ums/user/df068d79-613c-44dc-811e-d804a570aa60/password HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Date: Tue, 14 Feb 2017 20:13:56 GMT
Content-Length: 8

"R$rif43L"
```

В ответе сервер возвращается назначенный пользователю пароль.

8.1.2.1.3. Назначение схемы аутентификации

Схема аутентификации – это последовательность методов аутентификации, которые должны пройти пользователь для получения маркеров доступа.

В данном документе рассматривается случай, при котором пользователи аутентифицируются только по логину и паролю.

Для назначения схемы требуется отправить следующий запрос.

Запрос

```
POST /STS/ums/user/20948431-2f7e-4825-b723-
d56ea134298c/authmethod/password?level=1 HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 0
```

8.1.2.2. Создание нового сертификата

Исходные данные: логин пользователя, сертификат аутентификации оператора DSS, компоненты имени пользователя, идентификатор шаблона сертификата, идентификатор клиента OAuth 2.0, адрес перенаправления.

В примерах в качестве идентификатора клиента используется значение wpf.hybrid, в качестве адреса перенаправления - urn:ietf:wg:oauth:2.0:oob:auto.

Важно! Идентификатор клиента Oauth 2.0 и адрес перенаправления назначаются на этапе регистрации сторонней системы в DSS. Их значения используются в дальнейшем в качестве значений параметров Oauth-запросов client_id и redirect_uri соответственно.

Регистрация пользователя выполняется от имени Оператора DSS. Оператор аутентифицируется в DSS по сертификату.

Для того чтобы зарегистрировать пользователя необходимо выполнить следующую последовательность действий:

1. Получить маркер доступа OAuth для Оператора.
2. Получить маркер доступа для делегирования.
3. Сформировать запрос на сертификат.

8.1.2.2.1. Получение маркера доступа

Для получения маркера доступа требуется отправить следующий запрос к конечной точке /StsAppName/oauth/authorize/certificate. Клиент должен предоставить сертификат Оператора DSS для аутентификации.

Запрос

```
GET
https://dss.cryptopro.ru/STS/oauth/authorize/certificate?client_id=wpf.hybrid&re
sponse_type=code&scope=openid+dss&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Ao
ob%3Aauto HTTP/1.1
Host: dss.cryptopro.ru
```

Ответ

```
HTTP/1.1 302 Found
Location: urn:ietf:wg:oauth:2.0:oob:auto?code=7d921a45fab293ed1185863f3a662c88
Date: Tue, 28 Feb 2017 20:14:48 GMT
Content-Length: 0
```

Согласно сценарию с кодом авторизации, клиент должен извлечь из заголовка Location значение кода авторизации. После этого требуется отправить запрос для получения маркера безопасности.

Клиент авторизуются с помощью Basic аутентификации. Если клиенту выдавался только идентификатор и не выдавался секрет (такие клиенты называются публичными), то заголовок Basic аутентификации должен формироваться от следующих данных:

```
client_id:
```

Например, для client_id равноного wpf.hybrid данное значение будет равно

```
wpf.hybrid:
```

А заголовок авторизации соответственно:

```
d3BmLmh5YnJpZDo=
```

Запрос:

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Accept: application/json
Authorization: Basic d3BmLmh5YnJpZDpzZWNyZXQ=
Content-Type: application/x-www-form-urlencoded
Host: dss.cryptopro.ru
Content-Length: 266
grant_type=authorization_code&code=7d921a45fab293ed1185863f3a662c88&redirect_uri
=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto
```

Ответ:

```
HTTP/1.1 200 OK
Content-Length: 5152
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:48 GMT

{"access_token":"eyJ… ","expires_in":300}
```

Из полученного JSON-объекта следует извлечь **access_token**.

Публичные клиенты также могут добавлять свой идентификатор не в заголовок Basic аутентификации, а передавать непосредственно через параметр запроса **client_id**.

Запрос:

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Accept: application/json
Authorization: Basic d3BmLmh5YnJpZDpzZWNyZXQ=
Content-Type: application/x-www-form-urlencoded
Host: dss.cryptopro.ru
Content-Length: 266
grant_type=authorization_code&code=7d921a45fab293ed1185863f3a662c88&redirect_uri
=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto&client_id=wpf.hybrid
```

8.1.2.2.2. Получение маркера доступа для делегирования

Для обращений к сервису подписи Оператором от имени Пользователя необходимо получить маркер доступа для делегирования. Указанная процедура описана в стандарте

https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-07.

Для этого клиент отправляет следующий запрос.

Запрос:

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: dss.cryptopro.ru
Content-Length: 3217

actor_token=ey…&actor_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-
type%3Ajwt&subject_token=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJ1bmlxdWVfbmFtZSI
6Im15LWRzcy10ZXN0LTE0ODgzMTI4MzYiLCJuYmYiOjE0ODgzMTI4ODksImV4cCI6MTQ4ODMxNjQ4OSw
iaWF0IjoxNDg4MzEyODg5fQ.&subject_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-
type%3Ajwt&grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Atoken-
exchange&client_id=wpf.hybrid
```

Параметры запроса:

- actor_token – маркер доступа Оператора (получен на шаге 2.2.1),

- actor_token_type – тип маркера доступа, должен иметь значение
- “urn:ietf:params:oauth:token-type:jwt”.
- subject_token_type – тип маркера доступа, должен иметь значение
- “urn:ietf:params:oauth:token-type:jwt”.
- subject_token – неподписанный JWT-токен, содержащий логин управляемого пользователя. В декодированном виде токен из примера имеет следующий вид:

```
{
  "alg": "none",
  "typ": "JWT"
}.
{
  "unique_name": "my-dss-test-1488312836",
  "nbf": 1488312889,
  "exp": 1488316489,
  "iat": 1488312889
}.
```

Пример кодирования JWT-токенов доступен по ссылке https://jwt.io/introduction/.

Первая часть (до точки) называется header, вторая – payload. Чтобы получить закодированное значение необходимо выполнить следующее преобразование:

```
Base64UrlEncode(Utf8GetBytes(header)) + “.” +
Base64UrlEncode(Utf8GetBytes(payload)) + “.”
```

Символ “.” в конце получившегося значения является обязательным.

Следует отметить, что при кодировании JWT используется преобразование Base64Url, отличающееся от Base64 преобразования. Условно это преобразование можно представить следующим образом:

```
Base64Url(x) := Base64(x).Split(‘=’)[0].Replace(‘+’, ’-’).Replace(‘/’, ’_’)
```

здесь функция **Split(x)**, разбивает строку на части ([i] означает взятие i–ой части), используя символ разделитель **x**, функция **Replace(x,y)** заменяет все вхождения символа **x** на символ **y**.

Для компактности перед применением преобразования из входной строки могут быть удалены переносы и пробелы:

```
{"alg":"none","typ": "JWT"}.{"unique_name": "my-dss-test-
1488312836","nbf":1488312889,"exp":1488316489,"iat":1488312889}.
```

В указанном примере header всегда будет иметь постоянное значение:

```
eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0
```

Содержимое payload постоянно изменяется. В данном примере оно равно:

```
eyJ1bmlxdWVfbmFtZSI6Im15LWRzcy10ZXN0LTE0OTA1OTUwNTkiLCJuYmYiOjE0OTA1OTUwNjUsImV4
cCI6MTQ5MDU5ODY2NSwiaWF0IjoxNDkwNTk1MDY1fQ
```

В **ответ** сервер вернёт новый маркер доступа.

```
HTTP/1.1 200 OK
Content-Length: 3098
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:48 GMT

{"access_token":"eyJ0…","expires_in":300,"token_type":"Bearer"}
```

8.1.2.2.3. Создание нового сертификата

Для создания сертификата требуется отправить запрос следующего вида конечной точке SsAppName/rest/api/requests:

Запрос

```
POST https://dss.cryptopro.ru /SignServer/rest/api/requests HTTP/1.1
```

```
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhb…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 186
Expect: 100-continue

{"AuthorityId":”1”,"DistinguishedName":{"2.5.4.3":"my-dss-test-
1488449283"},"Parameters":{"TemplateOid":"1.2.643.2.2.50.1.9.3222149.8143850.120
77221.13733360.27687.63267"}}
```

Идентификатор удостоверяющего центра, шаблоны сертификатов можно получить из политики сервиса подписи (отправив GET запрос на адрес https://host/SsAppName/rest/api/policy).

DistinguishedName представляет собой различительное имя субъекта в виде набора компонентов {oid:value}.

Ответ

```
HTTP/1.1 200 OK
Content-Length: 705
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:50 GMT

{"Subject":null,"CertificateType":"ServerSide","Base64Request":"MIIBQjCB8gIBADAh
MR8wHQYDVQQDExZteS1kc3MtdGVzdC0xNDg4MzEyODM2MGMwHAYGKoUDAgITMBIGByqFAwICJAAGByqF
AwICHgEDQwAEQNLhRrbT4cZK+vI89nRo3yNRPu9lrWOh9NhwBbahnAc48+9TVoky6pOcLo5qGPjmIYxr
KcEjJ2cg4xE339aEEe2gZTBjBgorBgEEAYI3AgEOMVUwUzALBgNVHQ8EBAMCBPAwHQYDVR0OBBYEFF7q
MqepXSXaALSAD/fmg5hJOWntMCUGA1UdJQQeMBwGByqFAwICIgIGByqFAwICIgYGCCsGAQUFBwMCMAgG
BiqFAwICAwNBAD+clm41e7m+7Yo34+LoCqzga620mtFmiTTFRBHzLiTaWRWm4wNydLhvtvyb+bZD73MV
n/aRsAX5RBwAQLKyUaI=","CertificateAuthorityID":1,"CADisplayName":null,"DistName"
:"CN=my-dss-test-
1488312836","Status":"ACCEPTED","ID":64121,"CARequestID":"471520","CertificateID
":6464,"RequestType":"Certificate","CspID":null}
```

В полученном JSON поле CertificateID указывает на идентификатор выпущенного сертификата (если статус запроса ACCEPTED).

8.1.2.2.4. Назначение сертификата по умолчанию

Для того чтобы назначить сертификат по умолчанию необходимо отправить на конечную точку SsAppName/rest/api/certificates/certificateId/default запрос следующего вида

Запрос:

```
POST /SignServer/rest/api/certificates/6119/default HTTP/1.1
Authorization: Bearer eyJ0eXAiOi…
Content-Type: application/x-www-form-urlencoded
Host: dss.cryptopro.ru
Content-Length: 12

Default=True
```

В URL адресе запроса передаётся идентификатор сертификата, который требуется назначить сертификатом по умолчанию (в примере это 6119).

Ответ:

```
HTTP/1.1 200 OK
Content-Length: 0
Date: Thu, 27 Apr 2017 06:27:04 GMT
```

8.1.2.3. Подписание документа

Исходные данные: логин пользователя, идентификатор сертификата, идентификатор клиента OAuth 2.0, адрес перенаправления.

В примерах в качестве идентификатора клиента используется значение wpf.hybrid.

Для того чтобы подписать документ необходимо выполнить следующую последовательность действий:

1. Получить маркер доступа OAuth для Пользователя.
2. Отправить запрос на подпись.

8.1.2.3.1. Получение маркера доступа

Получить маркер доступа можно, отправив запрос на конечную точку /StsAppName/oauth/token

Запрос

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: sga10.cp.ru
Content-Length: 82

grant_type=password&username=my-dss-test-
1488312836&password=p#4fd1L0&client_id=wpf.hybrid
```

Параметр password содержит пароль пользователя, параметр username имеет значение логина пользователя.

Ответ

```
HTTP/1.1 200 OK
Content-Length: 2435
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:50 GMT

{"access_token":"eyJ0eXAiOi…","expires_in":300}
```

8.1.2.3.2. Подпись документа

Для подписи документа отправляется следующий запрос на конечную точку /SsAppName/rest/api/documents

Запрос

```
POST /SignServer/rest/api/documents HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 108

{“Content”:“AX/dfse…”,”Signature”:{
"Type":"CMS","CertificateId":6119,“Parameters”:{“IsDetached”:”True”}}}
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 3682
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:15:17 GMT

"MBAXvf…”
```

В случае успешного выполнения операции возвращается подпись документа.

8.1.2.3.3. Формирование соподписи

В CMS сообщение можно добавить подпись. Такую подпись называют соподпись или параллельная подпись.

Для формирования соподписи требуется сформировать запрос следующего вида и отправить его на конечную точку /SsAppName/rest/api/documents

Запрос

```
POST /SignServer/rest/api/documents HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 3543

{"Content":"MIIJ…","Signature":{"Type":"CMS","CertificateId":6119,"Parameters":{
"IsDetached":"True","OriginalDocument":"AQI=","CmsSignatureType":"CoSign"}}}
```

Content – содержимое CMS-сообщения, в которое требуется добавить подпись.

Параметры

Type – формат CMS подписи. Формат добавляемой подписи может отличаться от форматов уже добавленных в сообщение подписей. Например, первая подпись может быть в формате CAdES-BES, вторая в формате CAdES-X Long Type 1.

IsDetached – тип подписи присоединённая/отсоединённая.

OriginalDocument – содержимое подписываемого документа. Если первая подпись была отсоединённой, то для добавления последующих подписей **требуется** наличие исходного документа.

CmsSignatureType – тип CMS подписи. Для добавления соподписи значение этого параметра должно быть **CoSign**

CertificateId – идентификатор сертификата, на ключе которого требуется осуществить подпись.

Ответ

```
HTTP/1.1 200 OK
Content-Length: 3682
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:15:17 GMT

"MBAXvf…”
```

8.1.2.4. Удаление запросов и сертификатов пользователя

Исходные данные: логин пользователя, сертификат аутентификации оператора.

Перед выполнением данной операции необходимо получит маркер доступа согласно процедуре, описанной в пунктах 3.2.1 – 3.2.2.

Для удаления всех запросов требуется отправить следующий DELETE запрос на конечную точку SsAppName/rest/api/requests.

```
DELETE /SignServer/rest/api/certificates/0 HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJ…
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 0
Date: Thu, 16 Mar 2017 08:44:51 GMT
```

Для удаления всех сертификатов требуется отправить следующий DELETE запрос на конечную точку SsAppName/rest/api/certificates.

Запрос

```
DELETE /SignServer/rest/api/certificates/0 HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJ…
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 0
```

```
Date: Thu, 16 Mar 2017 08:44:51 GMT
```

Для аутентификации при выполнении обоих запросов используется маркер доступа. В примерах он приведён в урезанном виде для удобства чтения.

8.1.2.5. Удаление учётной записи пользователя

Исходные данные: уникальный идентификатор пользователя, сертификат аутентификации оператора DSS.

Действие выполняется Оператором.

Важно! Удаление пользователя не приводит к удалению его запросов на сертификат и самих сертификатов. Удаление этих объектов должно выполняться отдельно.

Для удаления пользователя приложение должно сформировать HTTP DELETE запрос следующего вида

Запрос

```
DELETE /STS/ums/user/9cf94797-1d02-4076-94fe-284a64e83582 HTTP/1.1
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
```

Если пользователь с заданным **id** не найден, сервер **не** вернёт ошибку.

8.2. REST API и аутентификация при помощи апплета на SIM-карте

8.2.1. Общие сведения

Данное руководство описывает типовую последовательность действий для подписи документа с использованием интерфейса взаимодействия REST, аутентификации по протоколу OAuth 2.0/OpenId Connect 1.0, подтверждения операций с помощью апплета, расположенного на SIM-карте.

8.2.1.1. Исходные данные

Для работы с DSS через REST API вызывающее приложение должно:

- знать адрес сервера DSS. Далее в документе и во всех примерах этот параметр будет называться host. Следует иметь в виду, что СЭП состоит из нескольких компонентов, каждый из которых может находится на отдельном сервере, в данном документе такой сценарий не рассматривается. Далее в примерах в качестве host будет использоваться dss.cryptopro.ru
- Обладать сертификатом Оператора DSS. Оператор DSS – это особенная роль, предназначенная для управления учётными записями пользователей и их сертификатами. Для выполнения некоторых действий требуется аутентификация с установлением двустороннего TLS соединения.
- Знать имена приложений-компонентов DSS. Далее эти параметры будут называться SsAppName и StsAppName для обозначения приложений Сервиса Подписи и Центра Идентификации соответственно. Таким образом, URL адрес запросов к этим компонентам будет иметь вид: https://host/SsAppName и https://host/StsAppName.

По умолчанию SsAppName имеет значение SignServer, StsAppName – STS, эти же значения будут использоваться в примерах.

- Знать состав различительного имени субъекта сертификата. Различительное имя субъекта – это то, что попадает в поле «Кому выдан» сертификата.
- Знать требуемый шаблон сертификата или уметь предоставить пользователю выбор нужного шаблона. Приложение может получить от DSS перечень допустимых шаблонов.
- Знать параметры протокола OAuth 2.0: идентификатор клиента (client_id) и адрес перенаправления (redirect_uri). Эти параметры необходимы для выполнения запросов на выпуск маркеров доступа к ресурсам СЭП.

8.2.1.2. Примеры запросов

Описания шагов снабжены примерами HTTP-запросов. Все запросы и ответы могут содержать заголовки, не указанные в примерах. Значения маркеров доступа в заголовках аутентификации приводится в урезанном виде для удобства чтения.

8.2.1.3. Сервис уведомлений

Операции с апплетом, установленным на SIM-карте, являются асинхронными. После отправки запроса на выполнение операции, вызывающее приложение должно ожидать получение результата в виде callback'а.

Для получения уведомлений о статусе и результатах операций, запрошенных в DSS приложение должно реализовывать SOAP-сервис уведомлений. Данный сервис должен реализовывать один метод **Notify**, принимающий на вход объект со сведениями о результате операции **DssOperationResultMessage**.

Таблица 104 содержит описание полей данного объекта.

Таблица 104. Описание полей объекта DssOperationResultMessage

Поле	Тип	Описание
OperationCode	int	Код операции. Обязательный параметр.
ResultCode	int	Результат операции. Обязательный параметр.
UserIdentifier	string	Идентификатор пользователя, для которого была выполнена операция. Обязательный параметр. Идентификатором пользователя может быть: Номер телефона (msisdn) Логин пользователя Email Требование к идентификатору пользователя: значение должно быть уникально в рамках Центра Идентификации, которому принадлежит пользователь.
ServiceIdentifier	string	Идентификатор экземпляра ЦИ, отправившего сообщение.
AdditionalData	Dictionary<int, string>	Дополнительные сведения о результатах операции. В данном поле могут передаваться дополнительные данные специфичные для каждой из операций. Для всех операций в данном поле передаётся transactionId.

Таблица 105 содержит описание кодов операций.

Таблица 105. Коды операций

Код операции	Описание
100	Активация апплета на SIM-карте
101	Запрос статуса активации апплета на SIM-карте
102	Аутентификация пользователя

Таблица 106 содержит описание кодов результатов операций.

Таблица 106. Коды результатов операций

Код операции	Описание
0	Операция завершилась успешно
1	Внутренняя ошибка
1000	Пользователь отказался от активации апплета
1001	Исчерпан лимит ввода кода активации

Код операции	Описание
1002	Пользователь отказался от активации, по причине невозможности придумать PIN
1020	Пользователь отказался от авторизации.
1021	Истекло время ожидание авторизации
1022	Приложение пользователя заблокировано из-за неверного PIN
1023	Апплет не активирован

WSDL-описание сервиса:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsa10="http://www.w3.org/2005/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy"
xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:tns="http://dss.cryptopro.ru/sim/services/2015/11/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" name="NotificationService"
targetNamespace="http://dss.cryptopro.ru/sim/services/2015/11/">
  <wsdl:types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://dss.cryptopro.ru/sim/services/2015/11/"
elementFormDefault="qualified">
      <xs:import
namespace="http://schemas.microsoft.com/2003/10/Serialization/Arrays"/>
      <xs:element name="Notify">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="opResult" type="tns:DssOperationResultMessage"
nillable="true" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:complexType name="DssOperationResultMessage">
        <xs:sequence>
          <xs:element name="OperationCode" type="xs:int"/>
          <xs:element name="ResultCode" type="xs:int"/>
          <xs:element name="UserIdentifier" type="xs:string" nillable="true"/>
          <xs:element name="ServiceIdentifier" type="xs:string" nillable="true"
minOccurs="0"/>
          <xs:element
xmlns:q1="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
name="AdditionalData" type="q1:ArrayOfKeyValueOfintstring" nillable="true"
minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="DssOperationResultMessage"
type="tns:DssOperationResultMessage" nillable="true"/>
      <xs:element name="NotifyResponse">
        <xs:complexType>
          <xs:sequence/>
        </xs:complexType>
      </xs:element>
    </xs:schema>
    <xs:schema xmlns:tns="http://schemas.microsoft.com/2003/10/Serialization/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/"
elementFormDefault="qualified" attributeFormDefault="qualified">
      <xs:element name="anyType" type="xs:anyType" nillable="true"/>
      <xs:element name="anyURI" type="xs:anyURI" nillable="true"/>
      <xs:element name="base64Binary" type="xs:base64Binary" nillable="true"/>
      <xs:element name="boolean" type="xs:boolean" nillable="true"/>
      <xs:element name="byte" type="xs:byte" nillable="true"/>
```

```
      <xs:element name="dateTime" type="xs:dateTime" nillable="true"/>
      <xs:element name="decimal" type="xs:decimal" nillable="true"/>
      <xs:element name="double" type="xs:double" nillable="true"/>
      <xs:element name="float" type="xs:float" nillable="true"/>
      <xs:element name="int" type="xs:int" nillable="true"/>
      <xs:element name="long" type="xs:long" nillable="true"/>
      <xs:element name="QName" type="xs:QName" nillable="true"/>
      <xs:element name="short" type="xs:short" nillable="true"/>
      <xs:element name="string" type="xs:string" nillable="true"/>
      <xs:element name="unsignedByte" type="xs:unsignedByte" nillable="true"/>
      <xs:element name="unsignedInt" type="xs:unsignedInt" nillable="true"/>
      <xs:element name="unsignedLong" type="xs:unsignedLong" nillable="true"/>
      <xs:element name="unsignedShort" type="xs:unsignedShort" nillable="true"/>
      <xs:element name="char" type="tns:char" nillable="true"/>
      <xs:simpleType name="char">
        <xs:restriction base="xs:int"/>
      </xs:simpleType>
      <xs:element name="duration" type="tns:duration" nillable="true"/>
      <xs:simpleType name="duration">
        <xs:restriction base="xs:duration">
          <xs:pattern value="\-?P(\d*D)?(T(\d*H)?(\d*M)?(\d*(\.\d*)?S)?)?"/>
          <xs:minInclusive value="-P10675199DT2H48M5.4775808S"/>
          <xs:maxInclusive value="P10675199DT2H48M5.4775807S"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:element name="guid" type="tns:guid" nillable="true"/>
      <xs:simpleType name="guid">
        <xs:restriction base="xs:string">
          <xs:pattern value="[\da-fA-F]{8}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-
F]{4}-[\da-fA-F]{12}"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:attribute name="FactoryType" type="xs:QName"/>
      <xs:attribute name="Id" type="xs:ID"/>
      <xs:attribute name="Ref" type="xs:IDREF"/>
    </xs:schema>
    <xs:schema
xmlns:tns="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
elementFormDefault="qualified">
      <xs:complexType name="ArrayOfKeyValueOfintstring">
        <xs:annotation>
          <xs:appinfo>
            <IsDictionary
xmlns="http://schemas.microsoft.com/2003/10/Serialization/">true</IsDictionary>
          </xs:appinfo>
        </xs:annotation>
        <xs:sequence>
          <xs:element name="KeyValueOfintstring" minOccurs="0"
maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Key" type="xs:int"/>
                <xs:element name="Value" type="xs:string" nillable="true"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="ArrayOfKeyValueOfintstring"
type="tns:ArrayOfKeyValueOfintstring" nillable="true"/>
    </xs:schema>
```

```
  </wsdl:types>
  <wsdl:message name="INotificationService_Notify_InputMessage">
    <wsdl:part name="parameters" element="tns:Notify"/>
  </wsdl:message>
  <wsdl:message name="INotificationService_Notify_OutputMessage">
    <wsdl:part name="parameters" element="tns:NotifyResponse"/>
  </wsdl:message>
  <wsdl:portType name="INotificationService">
    <wsdl:operation name="Notify">
      <wsdl:input message="tns:INotificationService_Notify_InputMessage"
wsaw:Action="http://dss.cryptopro.ru/sim/services/2015/11/INotificationService/N
otify"/>
      <wsdl:output message="tns:INotificationService_Notify_OutputMessage"
wsaw:Action="http://dss.cryptopro.ru/sim/services/2015/11/INotificationService/N
otifyResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="BasicHttpBinding_INotificationService"
type="tns:INotificationService">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Notify">
      <soap:operation style="document"
soapAction="http://dss.cryptopro.ru/sim/services/2015/11/INotificationService/No
tify"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="NotificationService">
    <wsdl:port name="BasicHttpBinding_INotificationService"
binding="tns:BasicHttpBinding_INotificationService">
      <soap:address
location="http://localhost:8733/Design_Time_Addresses/NotificationService/Notifi
cationService/"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

8.2.2. Основные сценарии взаимодействия

К основным сценариям взаимодействия относятся:

- Регистрация пользователя, назначение SIM-карты в качестве аутентификатора;
- активация SIM-карты;
- формирование сертификата;
- подписание документа от имени пользователя;
- Удаление сертификатов пользователя;
- Удаление учётной записи пользователя;
- Поиск учётной записи пользователя.

В данном разделе приводится описание основных схем взаимодействия.

8.2.2.1. Регистрация пользователя

Исходные данные: логин пользователя (обычно номер мобильного телефона), идентификатор SIM-карты (ICCID) сертификат аутентификации оператора DSS.

Регистрация пользователя выполняется от имени Оператора DSS. Оператор аутентифицируется в DSS по сертификату.

Для того чтобы зарегистрировать пользователя необходимо выполнить следующую последовательность действий:

1. Создать нового пользователя в DSS.
2. Назначить ему SIM-карту.
3. Назначить схему аутентификации.
4. Назначить операции, требующие дополнительного подтверждения.

8.2.2.1.1. Создание нового пользователя

Для создания нового пользователя необходимо отправить запрос следующего вида на конечную точку StsAppName/ums/user

Запрос

```
POST /STS/ums/user HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 34

{"PhoneNumber":"79998887766"}
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 38

"20948431-2f7e-4825-b723-d56ea134298c"
```

В ответе содержится уникальный идентификатор пользователя DSS. В случае с аутентификацией с использованием SIM-карты логином пользователя является MSISDN.

8.2.2.1.2. Назначение SIM-карты

Для назначения SIM-карты необходимо отправить следующий запрос

Запрос

```
POST /STS/ums/user/20948431-2f7e-4825-b723-d56ea134298c/simauth HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 49

{"IccId":"89701010061787918181"}
```

Ответ

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 0
```

8.2.2.1.3. Назначение схемы аутентификации

Схема аутентификации – это последовательность методов аутентификации, которые должен пройти пользователь для получения маркеров доступа.

В случае с аутентификацией с использованием SIM-карты схема состоит из двух шагов: на первом шаге аутентификации отсутствует (используются вырожденный метод аутентификации «Только идентификация»), на втором шаге применяется метод аутентификации «С помощью SIM-карты».

Для назначения схемы требуется отправить следующие два запроса.

Запрос

```
POST /STS/ums/user/20948431-2f7e-4825-b723-
d56ea134298c/authmethod/idonly?level=0 HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 0
```

Запрос

```
POST /STS/ums/user/20948431-2f7e-4825-b723-
d56ea134298c/authmethod/SimAuth?level=1 HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 0
```

8.2.2.1.4. Назначение операций, требующих подтверждения

Требование подтверждать операции с использованием процедуры многофакторной аутентификации может распространяться на различные операции с закрытым ключом (подпись документа, расшифрование документа, создание запроса на сертификат и т.д.).

Для назначения перечня операций, требующих подтверждения необходимо сформировать и отправить на сервер следующий запрос:

Запрос

```
POST /STS/ums/user/e59661bc-e792-491c-b331-991a51ee08cb/operationpolicy HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 16

["SignDocument"]
```

В запросе указывается JSON массив, содержащий идентификаторы операций, требующих подтверждения (перечень операций см. в руководстве разработчика).

В случае аутентификации с помощью SIM-карты подтверждается, как правило, только операция подписи (SignDocument).

Ответ

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 0
```

8.2.2.2. Активация апплета на SIM-карте

Для активации апплета необходимо

1. Получить код активации.
2. Отправить запрос на активацию.
3. Получить результат операции активации.

8.2.2.2.1. Получение кода активации

Для получения кода активации необходимо выполнить следующий запрос:

Запрос

```
GET STS/ums/user/cfd9375c-1509-4d72-a619-c5ae820f5626/simauth HTTP/1.1
```

```
Host: dss.cryptopro.ru
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 16 Jun 2017 07:15:56 GMT
Content-Length: 144

{"IccId":"89701010061787918181","ProfileId":"0e2c9db5-5a9e-4b3a-a88d-
728386a40523","PhoneNumber":null,"ActivationCode":”12345678”,"LastKnownStatus":0
}
```

В поле **ActivationCode** содержится код активации апплета.

8.2.2.2.2. Активация апплета

Активация апплета – асинхронная операция: после отправки в DSS запроса на активацию приложение ожидает возврат результата выполнения операции в виде callback'а (см. раздел 8.2.1.3).

Запрос

```
POST /STS/ums/user/d6ac27f0-7f05-4ed1-a57c-7c3b85c1af3e/simauth/message/activate
HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: sga10.cp.ru
Content-Length: 0
```

В ответе содержится идентификатор транзакции.

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 16 Jun 2017 12:44:25 GMT
Content-Length: 9

"8225867"
```

8.2.2.3. Создание нового сертификата

Исходные данные: логин пользователя, сертификат аутентификации оператора DSS, компоненты имени пользователя, идентификатор шаблона сертификата, идентификатор клиента OAuth 2.0, адрес перенаправления.

В примерах в качестве идентификатора клиента используется значение wpf.hybrid, в качестве адреса перенаправления - urn:ietf:wg:oauth:2.0:oob:auto.

Важно! Идентификатор клиента Oauth 2.0 и адрес перенаправления назначаются на этапе регистрации сторонней системы в DSS. Их значения используются в дальнейшем в качестве значений параметров Oauth-запросов client_id и redirect_uri соответственно.

Регистрация пользователя выполняется от имени Оператора DSS. Оператор аутентифицируется в DSS по сертификату.

Для того чтобы зарегистрировать пользователя необходимо выполнить следующую последовательность действий:

1. Получить маркер доступа OAuth для Оператора.
2. Получить маркер доступа для делегирования.
3. Сформировать запрос на сертификат.

8.2.2.3.1. Получение маркера доступа

Для получения маркера доступа требуется отправить следующий запрос к конечной точке /StsAppName/oauth/authorize/certificate. Клиент должен предоставить сертификат Оператора DSS для аутентификации.

Запрос

```
GET
https://dss.cryptopro.ru/STS/oauth/authorize/certificate?client_id=wpf.hybrid&re
sponse_type=code&scope=openid+dss&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Ao
ob%3Aauto HTTP/1.1
Host: dss.cryptopro.ru
```

Ответ

```
HTTP/1.1 302 Found
Location: urn:ietf:wg:oauth:2.0:oob:auto?code=7d921a45fab293ed1185863f3a662c88
Date: Tue, 28 Feb 2017 20:14:48 GMT
Content-Length: 0
```

Согласно сценарию с кодом авторизации, клиент должен извлечь из заголовка Location значение кода авторизации. После этого требуется отправить запрос для получения маркера безопасности.

Клиент авторизуются с помощью Basic аутентификации. Если клиенту выдавался только идентификатор и не выдавался секрет (такие клиенты называются публичными), то заголовок Basic аутентификации должен формироваться от следующих данных:

```
client_id:
```

Например, для client_id равноного wpf.hybrid данное значение будет равно

```
wpf.hybrid:
```

А заголовок авторизации соответственно:

```
d3BmLmh5YnJpZDo=
```

Запрос

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Accept: application/json
Authorization: Basic d3BmLmh5YnJpZDpzZWNyZXQ=
Content-Type: application/x-www-form-urlencoded
Host: dss.cryptopro.ru
Content-Length: 266
grant_type=authorization_code&code=7d921a45fab293ed1185863f3a662c88&redirect_uri
=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto&resource=https%3A%2F%2Fdss.cryptopro
.ru%2FSignServer%2Frest
```

В параметре resource передаётся идентификатор сервиса ресурсов, для которого выпускается маркер. Всегда имеет значение вида https://host/SsAppName/rest.

Ответ

```
HTTP/1.1 200 OK
Content-Length: 5152
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:48 GMT

{"access_token":"eyJ… ","expires_in":300}
```

Из полученного JSON-объекта следует извлечь **access_token**.

Публичные клиенты также могут добавлять свой идентификатор не в заголовок Basic аутентификации, а передавать непосредственно через параметр запроса **client_id**.

Запрос

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Accept: application/json
Authorization: Basic d3BmLmh5YnJpZDpzZWNyZXQ=
Content-Type: application/x-www-form-urlencoded
Host: dss.cryptopro.ru
Content-Length: 266
grant_type=authorization_code&code=7d921a45fab293ed1185863f3a662c88&redirect_uri
=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto&client_id=wpf.hybrid&resource=https%
3A%2F%2Fdss.cryptopro.ru%2FSignServer%2Frest
```

8.2.2.3.2. Получение маркера доступа для делегирования

Для обращений к сервису подписи Оператором от имени Пользователя необходимо получить маркер доступа для делегирования. Указанная процедура описана в документе

https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-07.

Для этого клиент отправляет следующий запрос.

Запрос:

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: dss.cryptopro.ru
Content-Length: 3217

actor_token=ey…&actor_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-
type%3Ajwt&subject_token=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJ1bmlxdWVfbmFtZSI
6Im15LWRzcy10ZXN0LTE0ODgzMTI4MzYiLCJuYmYiOjE0ODgzMTI4ODksImV4cCI6MTQ4ODMxNjQ4OSw
iaWF0IjoxNDg4MzEyODg5fQ.&subject_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-
type%3Ajwt&grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Atoken-
exchange&client_id=wpf.hybrid&resource=https%3A%2F%2Fdss.cryptopro.ru%2FSignServ
er%2Frest
```

Параметры запроса:

- resource – идентификатор сервиса ресурсов, для доступа к которому будет использоваться полученный маркер. Имеет значение вида https://host/SsAppName/rest
- actor_token – маркер доступа Оператора (получен на шаге 8.2.2.3.1),
- actor_token_type – тип маркера доступа, должен иметь значение
- “urn:ietf:params:oauth:token-type:jwt”.
- subject_token_type – тип маркера доступа, должен иметь значение
- “urn:ietf:params:oauth:token-type:jwt”.
- subject_token – неподписанный JWT-токен, содержащий логин управляемого пользователя. В декодированном виде токен из примера имеет следующий вид:

```
{
  "alg": "none",
  "typ": "JWT"
}.
{
  "unique_name": "my-dss-test-1488312836",
  "nbf": 1488312889,
  "exp": 1488316489,
  "iat": 1488312889
}.
```

Пример кодирования JWT-токенов доступен по ссылке https://jwt.io/introduction/.

Первая часть (до точки) называется header, вторая – payload. Чтобы получить закодированное значение необходимо выполнить следующее преобразование:

```
Base64UrlEncode(Utf8GetBytes(header)) + “.” +
Base64UrlEncode(Utf8GetBytes(payload)) + “.”
```

Символ “.” в конце получившегося значения является обязательным.

Следует отметить, что при кодировании JWT используется преобразование Base64Url, отличающееся от Base64 преобразования. Условно это преобразование можно представить следующим образом:

```
Base64Url(x) := Base64(x).Split(‘=’)[0].Replace(‘+’, ’-’).Replace(‘/’, ’_’)
```

здесь функция **Split(x)**, разбивает строку на части ([i] означает взятие i–ой части), используя символ разделитель **x**, функция **Replace(x,y)** заменяет все вхождения символа **x** на символ **y**.

Для компактности перед применением преобразования из входной строки могут быть удалены переносы и пробелы:

```
{"alg":"none","typ": "JWT"}.{"unique_name": "my-dss-test-
1488312836","nbf":1488312889,"exp":1488316489,"iat":1488312889}.
```

В указанном примере header всегда будет иметь постоянное значение:

```
eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0
```

Содержимое payload постоянно изменяется. В данном примере оно равно:

```
eyJ1bmlxdWVfbmFtZSI6Im15LWRzcy10ZXN0LTE0OTA1OTUwNTkiLCJuYmYiOjE0OTA1OTUwNjUsImV4
cCI6MTQ5MDU5ODY2NSwiaWF0IjoxNDkwNTk1MDY1fQ
```

В **ответ** сервер вернёт новый маркер доступа.

```
HTTP/1.1 200 OK
Content-Length: 3098
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:48 GMT

{"access_token":"eyJ0…","expires_in":300,"token_type":"Bearer"}
```

8.2.2.3.3. Создание нового сертификата

Для создания сертификата требуется отправить запрос следующего вида конечной точке SsAppName/rest/api/requests:

Запрос:

```
POST https://dss.cryptopro.ru /SignServer/rest/api/requests HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhb…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 186
Expect: 100-continue

{"AuthorityId":”1”,"DistinguishedName":{"2.5.4.3":"my-dss-test-
1488449283"},"Parameters":{"TemplateOid":"1.2.643.2.2.50.1.9.3222149.8143850.120
77221.13733360.27687.63267"}}
```

Идентификатор удостоверяющего центра, шаблоны сертификатов можно получить из политики сервиса подписи (отправив GET запрос на адрес https://host/SsAppName/rest/api/policy).

DistinguishedName представляет собой различительное имя субъекта в виде набора компонентов {oid:value}.

Ответ:

```
HTTP/1.1 200 OK
Content-Length: 705
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:50 GMT

{"Subject":null,"CertificateType":"ServerSide","Base64Request":"MIIBQjCB8gIBADAh
MR8wHQYDVQQDExZteS1kc3MtdGVzdC0xNDg4MzEyODM2MGMwHAYGKoUDAgITMBIGByqFAwICJAAGByqF
AwICHgEDQwAEQNLhRrbT4cZK+vI89nRo3yNRPu9lrWOh9NhwBbahnAc48+9TVoky6pOcLo5qGPjmIYxr
KcEjJ2cg4xE339aEEe2gZTBjBgorBgEEAYI3AgEOMVUwUzALBgNVHQ8EBAMCBPAwHQYDVR0OBBYEFF7q
MqepXSXaALSAD/fmg5hJOWntMCUGA1UdJQQeMBwGByqFAwICIgIGByqFAwICIgYGCCsGAQUFBwMCMAgG
BiqFAwICAwNBAD+clm41e7m+7Yo34+LoCqzga620mtFmiTTFRBHzLiTaWRWm4wNydLhvtvyb+bZD73MV
n/aRsAX5RBwAQLKyUaI=","CertificateAuthorityID":1,"CADisplayName":null,"DistName"
:"CN=my-dss-test-
1488312836","Status":"ACCEPTED","ID":64121,"CARequestID":"471520","CertificateID
":6464,"RequestType":"Certificate","CspID":null}
```

В полученном JSON поле CertificateID указывает на идентификатор выпущенного сертификата.

8.2.2.4. Подписание документа

Исходные данные: логин пользователя, идентификатор сертификата, идентификатор клиента OAuth 2.0, адрес перенаправления.

В примерах в качестве идентификатора клиента используется значение wpf.hybrid, в качестве адреса перенаправления - urn:ietf:wg:oauth:2.0:oob:auto.

Для того чтобы зарегистрировать пользователя необходимо выполнить следующую последовательность действий:

1. Получить маркер доступа OAuth для Пользователя.
2. Сформировать транзакцию подписи и подтвердить её.
3. Отправить запрос на подпись.

8.2.2.4.1. Получение маркера доступа

Получить маркер доступа можно, отправив запрос на конечную точку /StsAppName/oauth/token

Запрос

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: sga10.cp.ru
Content-Length: 82

grant_type=password&username=my-dss-test-
1488312836&password=&client_id=wpf.hybrid&resource=https%3A%2F%2Fdss.cryptopro.r
u%2FSignServer%2Frest%2Fapi
```

Параметр password должен быть пустым, параметр username имеет значение логина пользователя (указывался на этапе регистрации).

Ответ

```
HTTP/1.1 200 OK
Content-Length: 2435
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:50 GMT

{"access_token":"eyJ0eXAiOi…","expires_in":300}
```

8.2.2.4.2. Формирование и подтверждение транзакции

Для формирования транзакции требуется отправить следующие данные на конечную точку /SsAppName/rest/api/transactions

Для аутентификации используется маркер доступа с предыдущего шага.

Запрос

```
POST https://dss.cryptopro.ru/SignServer/rest/api/transactions HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJh…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 241

{"OperationCode":”SignDocument”,"Parameters":[{"Name":"DocumentInfo","Value":"do
cumentname.pdf"},{"Name":"SignatureType","Value":"CMS"},{"Name":"CertificateID",
"Value":"6464"},{"Name":"DocumentType","Value":"mydsssample"}],"Document":"AQI="
}
```

Параметры данного запроса следующие:

- OperationCode – Обязательный параметр, определяющий тип подтверждаемый операции. Всегда должен быть равен SignDocument.
- DocumentInfo – Обязательный параметр, содержащий информация о подписываемом документе. В значении данного параметра можно передать, например, название документа.

- SignatureType – Обязательный параметр, определяющий тип подписи. Поддерживаемые типы подписи см. в руководстве разработчика.
- CertificateID – Обязательный параметр, определяющий идентификатор сертификата, который будет использован для подписи документа.
- DocumentType – Необязательный параметр, определяющий тип документа. На основе типа документа DSS может сформировать выжимку документа, которая затем будет отображена в мобильном приложении см. пункт XXX.

В поле Document передаётся содержимое документа в BASE64.

Ответ

```
HTTP/1.1 200 OK
Content-Length: 38
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:50 GMT

"e60744eb-e872-4759-b18e-c9305604adcb"
```

В ответ сервер возвращает **идентификатор транзакции на Сервисе Подписи**. С помощью него необходимо сформировать запрос на подтверждение операции.

Запрос

```
POST https://dss.cryptopro.ru/STS/confirmation HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJ…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 117

{"Resource":"https://dss.cryptopro.ru/SignServer/rest","TransactionTokenId":"e60
744eb-e872-4759-b18e-c9305604adcb"}
```

В поле TransactionTokenId передаётся полученный в предыдущем ответе идентификатор транзакции, в поле Resource всегда должно быть значение вида https://host/SsAppName/rest.

Ответ

```
HTTP/1.1 200 OK
Content-Length: 9446
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:50 GMT

{"Challenge":{"Title":{"Value":"Подтвердите операцию на
телефоне"},"TextChallenge":[{"AuthnMethod":"http://dss.cryptopro.ru/identity/aut
henticationmethod/simauth","RefID":"14949057","Label":"Подпись документа.
Тестовая подпись 1497854248. Тип подписи: GOST3410. Сертификат:
75580250371.","MaxLenSpecified":false,"HideTextSpecified":false}],"ContextData":
{"RefID":"14949057"}},"IsFinal":false,"IsError":false}
```

Ответ сервера содержит идентификатор транзакции-подтверждения RefId.

Идентификатор транзакции на Сервисе Подписи и идентификатор транзакции подтверждения являются разными идентификаторами. Идентификатор транзакции Сервиса Подписи используется *только один* раз при формировании первого запроса на подтверждение.

После получения уведомления о завершении операции приложению необходимо сформировать следующий запрос:

Запрос

```
POST https://dss.cryptopro.ru/STS/confirmation HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGc…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 171
```

```
{"Resource":"https://dss.cryptopro.ru/SignServer/rest","ChallengeResponse":{"Tex
tChallengeResponse":[{"RefId":"fdd05368-acf1-4372-bea2-b51a77ea1923"}]}}
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 2657
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:15:17 GMT

{"AccessToken":"eyJ0eXAiOi…","IsFinal":true,"IsError":false}
```

Если транзакция подтверждена, то IsFinal будет равен true, а в поле AccessToken будет содержаться маркер доступа для выполнения операции, иначе IsFinal будет равен false.

8.2.2.4.3. Подпись документа

Для подписи документа отправляется следующий запрос на конечную точку /SsAppName/rest/api/documents

Запрос

```
POST https://dss.cryptopro.ru/SignServer/rest/api/documents HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 108

{}
```

Следует отметить, что содержимое документа в запросе не передаётся (в данном примере передаётся пустой JSON объект), так как оно уже было передано на сервер при создании транзакции. Если же для подписи требуется передать какие-то параметры, то они должны быть переданы в запросе повторно:

```
{Signature: {“Parameters”:{“Detached”:”true”}}}
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 3682
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:15:17 GMT

"MBAXvf…”
```

В случае успешного выполнения операции возвращается подпись документа.

8.2.2.4.4. Отображение подписанного документа

При использовании подтверждения операции подписи через myDss существует возможность показать пользователю в мобильном приложении некоторые сведения об операции. Сведения об операции помимо информации о сертификате и типе создаваемой подписи могут содержать информацию о документе. Информация о документе бывает трёх типов:

1. Неформатированные сведения о документе.
2. Форматированные сведения о документе.

Тип (1) можно использовать для передачи названия документа или названия подписываемого файла. Информация данного типа передаётся в запросе на создание транзакции (см. пункт 8.2.2.4.2) в значении параметра DocumentInfo:

```
{"OperationCode":”SignDocument”,"Parameters":[{"Name":"DocumentInfo","Value":"do
cumentname.pdf"},{"Name":"SignatureType","Value":"CMS"},{"Name":"CertificateID",
"Value":"6464"},{"Name":"DocumentType","Value":"simsample"}],"Document":"AQI="}
```

Тип (2) представляет собой данные в формате DTBS (Data To Be Signed, см. соответствующий раздел руководства разработчика). Для их получения на сервере DSS должны быть настроены специальные конвертеры, которые преобразуют содержимое документа в DTBS. Нужный конвертер подбирается на основе типа документа, передаваемого при создании транзакции в значении параметра DocumentType.

По умолчанию, в DSS присутствует только конвертер-заглушка, осуществляющий преобразование «1 к 1». То есть на вход операции подписи (в параметре Document) должно быть изначально передано содержимое в формате DTBS. DocumentType при этом должен быть равен dtbs.

Пример dtbs документа

```
<?xml version="1.0" encoding="utf-8"?><dtbs
xmlns="http://www.cryptopro.ru/schemas/2014/08/dtbs"><row><name>Номер счёта
получателя</name><value>1490690623</value></row><row><name>Номер счёта
отправителя</name><value>1490690633</value></row><row><name>Сумма</name><value>1
00</value></row></dtbs>
```

Пример dtbs документа после BASE64 преобразования (считая, что исходная строка в UTF-8):

```
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz48ZHRicyB4bWxucz0iaHR0cDovL3d3
dy5jcnlwdG9wcm8ucnUvc2NoZW1hcy8yMDE0LzA4L2R0YnMiPjxyb3c+PG5hbWU+0J3QvtC80LXRgCDR
gdGH0ZHRgtCwINC/0L7Qu9GD0YfQsNGC0LXQu9GPPC9uYW1lPjx2YWx1ZT4xNDkwNjkwNjIzPC92YWx1
ZT48L3Jvdz48cm93PjxuYW1lPtCd0L7QvNC10YAg0YHRh9GR0YLQsCDQvtGC0L/RgNCw0LLQuNGC0LXQ
u9GPPC9uYW1lPjx2YWx1ZT4xNDkwNjkwNjMzPC92YWx1ZT48L3Jvdz48cm93PjxuYW1lPtCh0YPQvNC8
0LA8L25hbWU+PHZhbHVlPjEwMDwvdmFsdWU+PC9yb3c+PC9kdGJzPg==
```

Пример запроса на создание транзакции

```
{"OperationCode":”SignDocument”,"Parameters":[{"Name":"DocumentInfo","Value":"do
cumentname.pdf"},{"Name":"SignatureType","Value":"CMS"},{"Name":"CertificateID",
"Value":"6464"},{"Name":"DocumentType","Value":"dtbs"}],"Document":"
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz48ZHRicyB4bWxucz0iaHR0cDovL3d3
dy5jcnlwdG9wcm8ucnUvc2NoZW1hcy8yMDE0LzA4L2R0YnMiPjxyb3c+PG5hbWU+0J3QvtC80LXRgCDR
gdGH0ZHRgtCwINC/0L7Qu9GD0YfQsNGC0LXQu9GPPC9uYW1lPjx2YWx1ZT4xNDkwNjkwNjIzPC92YWx1
ZT48L3Jvdz48cm93PjxuYW1lPtCd0L7QvNC10YAg0YHRh9GR0YLQsCDQvtGC0L/RgNCw0LLQuNGC0LXQ
u9GPPC9uYW1lPjx2YWx1ZT4xNDkwNjkwNjMzPC92YWx1ZT48L3Jvdz48cm93PjxuYW1lPtCh0YPQvNC8
0LA8L25hbWU+PHZhbHVlPjEwMDwvdmFsdWU+PC9yb3c+PC9kdGJzPg=="}
```

8.2.2.5. Поиск пользователя

Исходные данные: логин пользователя, сертификат аутентификации оператора DSS.

Действие выполняется Оператором.

Запрос

```
GET /STS/ums/user/?type=login&value=my-dss-test-1489649228 HTTP/1.1
Host: dss.cryptopro.ru
```

В запросе клиентское приложение должно передать логин пользователя.

В ответ на этот запрос сервер сформирует следующий результат.

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Thu, 16 Mar 2017 07:29:35 GMT
Content-Length: 344

{"UserId":"31c7f78d-1a41-44a1-85c0-5c7bb0ee3eea","Login":"my-dss-test-
1489649228","PhoneNumber":null,"Email":null,"PhoneConfirmed":false,"EmailConfirm
ed":false,"DisplayName":null,"DistinguishName":null,"AccountLocked":false,"Group
":"Default","CreationDate":"2017-03-
16T10:27:11.577","LockoutDate":null,"LastLoginDate":"2017-03-16T10:27:11.577"}
```

В теле запроса содержится информация о пользователя в виде JSON-объекта.

Уникальный идентификатор пользователя представляет собой значение свойства **UserId** полученного объекта.

Если по заданному логину не удалось найти пользователя, сервер вернёт HTTP-ответ со статусом 404 (NOT FOUND).

Ответ

```
HTTP/1.1 404 Not Found
```

8.2.2.6. Удаление запросов и сертификатов пользователя

Исходные данные: логин пользователя, сертификат аутентификации оператора.

Перед выполнением данной операции необходимо получит маркер доступа согласно процедуре, описанной в пунктах 2.2.1 – 2.2.2.

Для удаления всех запросов требуется отправить следующий DELETE запрос на конечную точку SsAppName/rest/api/requests.

```
DELETE /SignServer/rest/api/certificates/0 HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJ…
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 0
Date: Thu, 16 Mar 2017 08:44:51 GMT
```

Для удаления всех сертификатов требуется отправить следующий DELETE запрос на конечную точку SsAppName/rest/api/certificates.

Запрос

```
DELETE /SignServer/rest/api/certificates/0 HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJ…
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 0
Date: Thu, 16 Mar 2017 08:44:51 GMT
```

Для аутентификации при выполнении обоих запросов используется маркер доступа. В примерах он приведён в урезанном виде для удобства чтения.

8.2.2.7. Удаление учётной записи пользователя

Исходные данные: уникальный идентификатор пользователя, сертификат аутентификации оператора DSS.

Действие выполняется Оператором.

Важно! Удаление пользователя не приводит к удалению его запросов на сертификат и самих сертификатов. Удаление этих объектов должно выполняться отдельно.

Для удаления пользователя приложение должно сформировать HTTP DELETE запрос следующего вида

Запрос

```
DELETE /STS/ums/user/9cf94797-1d02-4076-94fe-284a64e83582 HTTP/1.1
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
```

Если пользователь с заданным **id** не найден, сервер **не** вернёт ошибку.

8.3. Интеграция с модулем аутентификации myDSS

8.3.1. Общие сведения

Данный раздел описывает типовую последовательность действий для подписи документа с использованием интерфейса взаимодействия REST, аутентификации по протоколу OAuth 2.0/OpenId Connect 1.0, подтверждения операций с помощью приложения myDSS.

8.3.1.1. Исходные данные

Для работы с DSS через REST API вызывающее приложение должно:

- знать адрес сервера DSS. Далее в документе и во всех примерах этот параметр будет называться host. Следует иметь в виду, что СЭП состоит из нескольких компонентов, каждый из которых может находится на отдельном сервере, в данном документе такой сценарий не рассматривается. Далее в примерах в качестве host будет использоваться dss.cryptopro.ru
- Обладать сертификатом Оператора DSS. Оператор DSS – это особенная роль, предназначенная для управления учётными записями пользователей и их сертификатами. Для выполнения некоторых действий требуется аутентификация с установлением двустороннего TLS соединения.
- Знать имена приложений-компонентов DSS. Далее эти параметры будут называться SsAppName и StsAppName для обозначения приложений Сервиса Подписи и Центра Идентификации соответственно. Таким образом, URL адрес запросов к этим компонентам будет иметь вид: https://host/SsAppName и https://host/StsAppName.

По умолчанию SsAppName имеет значение SignServer, StsAppName – STS, эти же значения будут использоваться в примерах.

- Знать состав различительного имени субъекта сертификата. Различительное имя субъекта – это то, что попадает в поле «Кому выдан» сертификата.
- Знать требуемый шаблон сертификата или уметь предоставить пользователю выбор нужного шаблона. Приложение может получить от DSS перечень допустимых шаблонов.
- Знать параметры протокола OAuth 2.0: идентификатор клиента (client_id) и адрес перенаправления (redirect_uri). Эти параметры необходимы для выполнения запросов на выпуск маркеров доступа к ресурсам СЭП.

8.3.1.2. Примеры запросов

Описания шагов снабжены примерами HTTP-запросов. Все запросы и ответы могут содержать заголовки, не указанные в примерах. Значения маркеров доступа в заголовках аутентификации приводится в урезанном виде для удобства чтения.

8.3.2. Основные сценарии взаимодействия

К основным сценариям взаимодействия относятся:

- Регистрация пользователя, назначение myDss в качестве аутентификатора;
- формирование сертификата;
- подписание документа от имени пользователя;
- смена ключа myDss;
- Удаление сертификатов пользователя;
- Удаление учётной записи пользователя;
- Поиск учётной записи пользователя.

В данном разделе приводится описание основных схем взаимодействия.

8.3.2.1. Регистрация пользователя

Исходные данные: логин пользователя, сертификат аутентификации оператора DSS, контактные данные пользователя (адрес электронной почты или номер мобильного телефона).

Регистрация пользователя выполняется от имени Оператора DSS. Оператор аутентифицируется в DSS по сертификату.

Для того чтобы зарегистрировать пользователя необходимо выполнить следующую последовательность действий:

1. Создать нового пользователя в DSS.
2. Назначить ему мобильный аутентификатор myDss.
3. Назначить схему аутентификации.
4. Назначить операции, требующие дополнительного подтверждения.

8.3.2.1.1. Создание нового пользователя

Для создания нового пользователя необходимо отправить запрос следующего вида на конечную точку StsAppName/ums/user

Запрос

```
POST /STS/ums/user HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 34

{"Login":"my-dss-test-1488312836"}
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 38

"20948431-2f7e-4825-b723-d56ea134298c"
```

В ответе содержится уникальный идентификатор пользователя DSS.

8.3.2.1.2. Назначение мобильного аутентификатора

Для назначения мобильного аутентификатора требуется отправить контактные данные пользователя. Эти данные будут использованы для передачи *второй части* ключа.

Запрос

```
POST /STS/ums/user/20948431-2f7e-4825-b723-d56ea134298c/mobileauth HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 69

{"UserContactInfo":"79150000000","UserContactInfoType":"PhoneNumber"}
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Expires: -1
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 7683
{"ExternalUserId":"e59b4978-b171-4ea3-a3ff-
aac6a3b4f3b3","QrCode":"R0l…","KeyExpirationTime":"2018-02-28T00:00:00"}
```

В ответе сервер возвращает JSON-объект, содержащий QR-код, который следует отобразить пользователю.

8.3.2.1.3. Назначение схемы аутентификации

Схема аутентификации – это последовательность методов аутентификации, которые должны пройти пользователь для получения маркеров доступа.

В случае с myDss схема состоит из двух шагов: на первом шаге аутентификации отсутствует (используются вырожденный метод аутентификации «Только идентификация»), на втором шаге применяется метод аутентификации «С помощью мобильного приложения».

Для назначения схемы требуется отправить следующий два запроса.

Запрос

```
POST /STS/ums/user/20948431-2f7e-4825-b723-
d56ea134298c/authmethod/idonly?level=0 HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 0
```

Запрос

```
POST /STS/ums/user/20948431-2f7e-4825-b723-
d56ea134298c/authmethod/MobileAuth?level=1 HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 0
```

8.3.2.1.4. Назначение операций, требующих подтверждения

Требование подтверждать операции с использованием процедуры многофакторной аутентификации может распространяться на различные операции с закрытым ключом (подпись документа, расшифрование документа, создание запроса на сертификат и т.д.).

Для назначения перечня операций, требующих подтверждения необходимо сформировать и отправить на сервер следующий запрос:

Запрос

```
POST /STS/ums/user/e59661bc-e792-491c-b331-991a51ee08cb/operationpolicy HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 16

["SignDocument"]
```

В запросе указывается JSON массив, содержащий идентификаторы операций, требующих подтверждения (перечень операций см. в руководстве разработчика).

В случае с myDss подтверждается, как правило, только операция подписи (SignDocument).

Ответ

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 0
```

8.3.2.2. Создание нового сертификата

Исходные данные: логин пользователя, сертификат аутентификации оператора DSS, компоненты имени пользователя, идентификатор шаблона сертификата), идентификатор клиента OAuth 2.0, адрес перенаправления.

В примерах в качестве идентификатора клиента используется значение wpf.hybrid, в качестве адреса перенаправления - urn:ietf:wg:oauth:2.0:oob:auto.

Важно! Идентификатор клиента Oauth 2.0 и адрес перенаправления назначаются на этапе регистрации сторонней системы в DSS. Их значения используются в дальнейшем в качестве значений параметров Oauth-запросов client_id и redirect_uri соответственно.

Регистрация пользователя выполняется от имени Оператора DSS. Оператор аутентифицируется в DSS по сертификату.

Для того чтобы зарегистрировать пользователя необходимо выполнить следующую последовательность действий:

1. Получить маркер доступа OAuth для Оператора.
2. Получить маркер доступа для делегирования.
3. Сформировать запрос на сертификат.

8.3.2.2.1. Получение маркера доступа

Для получения маркера доступа требуется отправить следующий запрос к конечной точке /StsAppName/oauth/authorize/certificate. Клиент должен предоставить сертификат Оператора DSS для аутентификации.

Запрос:

```
GET
https://dss.cryptopro.ru/STS/oauth/authorize/certificate?client_id=wpf.hybrid&re
sponse_type=code&scope=openid+dss&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Ao
ob%3Aauto HTTP/1.1
Host: dss.cryptopro.ru
```

Ответ:

```
HTTP/1.1 302 Found
Location: urn:ietf:wg:oauth:2.0:oob:auto?code=7d921a45fab293ed1185863f3a662c88
Date: Tue, 28 Feb 2017 20:14:48 GMT
Content-Length: 0
```

Согласно сценарию с кодом авторизации, клиент должен извлечь из заголовка Location значение кода авторизации. После этого требуется отправить запрос для получения маркера безопасности.

Клиент авторизуются с помощью Basic аутентификации. Если клиенту выдавался только идентификатор и не выдавался секрет (такие клиенты называются публичными), то заголовок Basic аутентификации должен формироваться от следующих данных:

```
client_id:
```

Например, для client_id равноного wpf.hybrid данное значение будет равно

```
wpf.hybrid:
```

А заголовок авторизации соответственно:

```
d3BmLmh5YnJpZDo=
```

Запрос:

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Accept: application/json
Authorization: Basic d3BmLmh5YnJpZDpzZWNyZXQ=
Content-Type: application/x-www-form-urlencoded
Host: dss.cryptopro.ru
Content-Length: 266
grant_type=authorization_code&code=7d921a45fab293ed1185863f3a662c88&redirect_uri
=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto
```

Ответ:

```
HTTP/1.1 200 OK
Content-Length: 5152
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:48 GMT

{"access_token":"eyJ… ","expires_in":300}
```

Из полученного JSON-объекта следует извлечь **access_token**.

Публичные клиенты также могут добавлять свой идентификатор не в заголовок Basic аутентификации, а передавать непосредственно через параметр запроса **client_id**.

Запрос

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Accept: application/json
Authorization: Basic d3BmLmh5YnJpZDpzZWNyZXQ=
Content-Type: application/x-www-form-urlencoded
Host: dss.cryptopro.ru
Content-Length: 266
grant_type=authorization_code&code=7d921a45fab293ed1185863f3a662c88&redirect_uri
=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob%3Aauto&client_id=wpf.hybrid
```

8.3.2.2.2. Получение маркера доступа для делегирования

Для обращений к сервису подписи Оператором от имени Пользователя необходимо получить маркер доступа для делегирования. Указанная процедура описана в стандарте https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-07.

Для этого клиент отправляет следующий запрос.

Запрос

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: dss.cryptopro.ru
Content-Length: 3217

actor_token=ey…&actor_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-
type%3Ajwt&subject_token=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJ1bmlxdWVfbmFtZSI
6Im15LWRzcy10ZXN0LTE0ODgzMTI4MzYiLCJuYmYiOjE0ODgzMTI4ODksImV4cCI6MTQ4ODMxNjQ4OSw
iaWF0IjoxNDg4MzEyODg5fQ.&subject_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-
type%3Ajwt&grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Atoken-
exchange&client_id=wpf.hybrid
```

Параметры запроса:

- actor_token – маркер доступа Оператора (получен на шаге 2.2.1),
- actor_token_type – тип маркера доступа, должен иметь значение
- “urn:ietf:params:oauth:token-type:jwt”.
- subject_token_type – тип маркера доступа, должен иметь значение
- “urn:ietf:params:oauth:token-type:jwt”.
- subject_token – неподписанный JWT-токен, содержащий логин управляемого пользователя. В декодированном виде токен из примера имеет следующий вид:

```
{
  "alg": "none",
  "typ": "JWT"
}.
{
  "unique_name": "my-dss-test-1488312836",
  "nbf": 1488312889,
  "exp": 1488316489,
  "iat": 1488312889
}.
```

Пример кодирования JWT-токенов доступен по ссылке https://jwt.io/introduction/.

Первая часть (до точки) называется header, вторая – payload. Чтобы получить закодированное значение необходимо выполнить следующее преобразование:

```
Base64UrlEncode(Utf8GetBytes(header)) + “.” +
Base64UrlEncode(Utf8GetBytes(payload)) + “.”
```

Символ “.” в конце получившегося значения является обязательным.

Следует отметить, что при кодировании JWT используется преобразование Base64Url, отличающееся от Base64 преобразования. Условно это преобразование можно представить следующим образом:

```
Base64Url(x) := Base64(x).Split(‘=’)[0].Replace(‘+’, ’-’).Replace(‘/’, ’_’)
```

здесь функция **Split(x)**, разбивает строку на части ([i] означает взятие i–ой части), используя символ разделитель **x**, функция **Replace(x,y)** заменяет все вхождения символа **x** на символ **y**.

Для компактности перед применением преобразования из входной строки могут быть удалены переносы и пробелы:

```
{"alg":"none","typ": "JWT"}.{"unique_name": "my-dss-test-
1488312836","nbf":1488312889,"exp":1488316489,"iat":1488312889}.
```

В указанном примере header всегда будет иметь постоянное значение:

```
eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0
```

Содержимое payload постоянно изменяется. В данном примере оно равно:

```
eyJ1bmlxdWVfbmFtZSI6Im15LWRzcy10ZXN0LTE0OTA1OTUwNTkiLCJuYmYiOjE0OTA1OTUwNjUsImV4
cCI6MTQ5MDU5ODY2NSwiaWF0IjoxNDkwNTk1MDY1fQ
```

В **ответ** сервер вернёт новый маркер доступа.

```
HTTP/1.1 200 OK
Content-Length: 3098
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:48 GMT

{"access_token":"eyJ0…","expires_in":300,"token_type":"Bearer"}
```

8.3.2.2.3. Создание нового сертификата

Для создания сертификата требуется отправить запрос следующего вида конечной точке SsAppName/rest/api/requests:

Запрос

```
POST https://dss.cryptopro.ru /SignServer/rest/api/requests HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhb…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 186
Expect: 100-continue

{"AuthorityId":”1”,"DistinguishedName":{"2.5.4.3":"my-dss-test-
1488449283"},"Parameters":{"TemplateOid":"1.2.643.2.2.50.1.9.3222149.8143850.120
77221.13733360.27687.63267"}}
```

Идентификатор удостоверяющего центра, шаблоны сертификатов можно получить из политики сервиса подписи (отправив GET запрос на адрес https://host/SsAppName/rest/api/policy).

DistinguishedName представляет собой различительное имя субъекта в виде набора компонентов {oid:value}.

Ответ

```
HTTP/1.1 200 OK
```

```
Content-Length: 705
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:50 GMT

{"Subject":null,"CertificateType":"ServerSide","Base64Request":"MIIBQjCB8gIBADAh
MR8wHQYDVQQDExZteS1kc3MtdGVzdC0xNDg4MzEyODM2MGMwHAYGKoUDAgITMBIGByqFAwICJAAGByqF
AwICHgEDQwAEQNLhRrbT4cZK+vI89nRo3yNRPu9lrWOh9NhwBbahnAc48+9TVoky6pOcLo5qGPjmIYxr
KcEjJ2cg4xE339aEEe2gZTBjBgorBgEEAYI3AgEOMVUwUzALBgNVHQ8EBAMCBPAwHQYDVR0OBBYEFF7q
MqepXSXaALSAD/fmg5hJOWntMCUGA1UdJQQeMBwGByqFAwICIgIGByqFAwICIgYGCCsGAQUFBwMCMAgG
BiqFAwICAwNBAD+clm41e7m+7Yo34+LoCqzga620mtFmiTTFRBHzLiTaWRWm4wNydLhvtvyb+bZD73MV
n/aRsAX5RBwAQLKyUaI=","CertificateAuthorityID":1,"CADisplayName":null,"DistName"
:"CN=my-dss-test-
1488312836","Status":"ACCEPTED","ID":64121,"CARequestID":"471520","CertificateID
":6464,"RequestType":"Certificate","CspID":null}
```

В полученном JSON поле CertificateID указывает на идентификатор выпущенного сертификата.

8.3.2.3. Подписание документа

Исходные данные: логин пользователя, идентификатор сертификата, идентификатор клиента OAuth 2.0, адрес перенаправления.

В примерах в качестве идентификатора клиента используется значение wpf.hybrid, в качестве адреса перенаправления - urn:ietf:wg:oauth:2.0:oob:auto.

Для того чтобы зарегистрировать пользователя необходимо выполнить следующую последовательность действий:

1. Получить маркер доступа OAuth для Пользователя.
2. Сформировать транзакцию подписи и подтвердить её.
3. Отправить запрос на подпись.

8.3.2.3.1. Получение маркера доступа

Получить маркер доступа можно, отправив запрос на конечную точку /StsAppName/oauth/token

Запрос

```
POST https://dss.cryptopro.ru/STS/oauth/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: sga10.cp.ru
Content-Length: 82

grant_type=password&username=my-dss-test-
1488312836&password=&client_id=wpf.hybrid
```

Параметр password должен быть пустым, параметр username имеет значение логина пользователя (указывался на этапе регистрации).

Ответ

```
HTTP/1.1 200 OK
Content-Length: 2435
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:50 GMT

{"access_token":"eyJ0eXAiOi…","expires_in":300}
```

8.3.2.3.2. Формирование и подтверждение транзакции

Для формирования транзакции требуется отправить следующие данные на конечную точку /SsAppName/rest/api/transactions

Для аутентификации используется маркер доступа с предыдущего шага.

Запрос

```
POST https://dss.cryptopro.ru/SignServer/rest/api/transactions HTTP/1.1
```

```
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJh…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 241

{"OperationCode":”SignDocument”,"Parameters":[{"Name":"DocumentInfo","Value":"do
cumentname.pdf"},{"Name":"SignatureType","Value":"CMS"},{"Name":"CertificateID",
"Value":"6464"},{"Name":"DocumentType","Value":"mydsssample"}],"Document":"AQI="
}
```

Параметры данного запроса следующие:

- OperationCode – Обязательный параметр, определяющий тип подтверждаемый операции. Всегда должен быть равен SignDocument.
- DocumentInfo – Обязательный параметр, содержащий информация о подписываемом документе. В значении данного параметра можно передать, например, название документа.
- SignatureType – Обязательный параметр, определяющий тип подписи. Поддерживаемые типы подписи см. в руководстве разработчика.
- CertificateID – Обязательный параметр, определяющий идентификатор сертификата, который будет использован для подписи документа.
- DocumentType – Необязательный параметр, определяющий тип документа. На основе типа документа DSS может сформировать выжимку документа, которая затем будет отображена в мобильном приложении см. пункт XXX.

В поле Document передаётся содержимое документа в BASE64.

Ответ

```
HTTP/1.1 200 OK
Content-Length: 38
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:50 GMT

"e60744eb-e872-4759-b18e-c9305604adcb"
```

В ответ сервер возвращает **идентификатор транзакции на Сервисе Подписи**. С помощью него необходимо сформировать запрос на подтверждение операции.

Запрос

```
POST https://dss.cryptopro.ru/STS/confirmation HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJ…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 117

{"Resource":"https://dss.cryptopro.ru/SignServer/rest/api","TransactionTokenId":
"e60744eb-e872-4759-b18e-c9305604adcb"}
```

В поле TransactionTokenId передаётся полученный в предыдущем ответе идентификатор транзакции, в поле Resource всегда должно быть значение вида https://host/SsAppName/rest/api.

Ответ

```
HTTP/1.1 200 OK
Content-Length: 9446
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:14:50 GMT

{"Challenge":{"Title":{"Value":"Подтвердите операцию на устройстве с помощью
приложения."},"TextChallenge":[{"Image":{"MimeType":"image/gif","Value":"R0l…”
AuthnMethod":"http://dss.cryptopro.ru/identity/authenticationmethod/mobile","Ref
ID":"fdd05368-acf1-4372-bea2-
b51a77ea1923","MaxLenSpecified":false,"HideTextSpecified":false}],"ContextData":
```

```
{"RefID":"fdd05368-acf1-4372-bea2-
b51a77ea1923"}},"IsFinal":false,"IsError":false}
```

Ответ сервера содержит QR-код для offline-подтверждения и **идентификатор транзакции-подтверждения** RefId.

Идентификатор транзакции на Сервисе Подписи и идентификатор транзакции подтверждения являются разными идентификаторами. Идентификатор транзакции на сервисе подписи используется *только один* раз при формировании первого запроса на подтверждение.

Для подтверждения транзакции следует отправить запрос

Запрос

```
POST https://dss.cryptopro.ru/STS/confirmation HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGc…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 171

{"Resource":"https://sga10.cp.ru/DssSignServer/rest/api","ChallengeResponse":{"T
extChallengeResponse":[{"RefId":"fdd05368-acf1-4372-bea2-
b51a77ea1923","Value":"049609"}]}}
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 2657
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:15:17 GMT

{"AccessToken":"eyJ0eXAiOi…","IsFinal":true,"IsError":false}
```

Если транзакция подтверждена, то IsFinal будет равен true, а в поле AccessToken будет содержаться маркер доступа для выполнения операции, иначе IsFinal будет равен false.

Если в запросе не указывать Value, то сервис подписи вернёт текущий статус транзакции (IsFinal = true – Транзакция подтверждена, иначе не подтверждена).

8.3.2.3.3. Подпись документа

Для подписи документа отправляется следующий запрос на конечную точку /SsAppName/rest/api/documents

Запрос

```
POST https://dss.cryptopro.ru/SignServer/rest/api/documents HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI…
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 108

{}
```

Следует отметить, что содержимое документа в запросе не передаётся (в данном примере передаётся пустой JSON объект), так как оно уже было передано на сервер при создании транзакции. Если же для подписи требуется передать какие-то параметры, то они должны быть переданы в запросе повторно:

```
{Signature: {"Parameters":{"Detached":"true"}}}
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 3682
Content-Type: application/json; charset=utf-8
Date: Tue, 28 Feb 2017 20:15:17 GMT

"MBAXvf…"
```

В случае успешного выполнения операции возвращается подпись документа.

8.3.2.3.4. Отображение подписанного документа

При использовании подтверждения операции подписи через myDss существует возможность показать пользователю в мобильном приложении некоторые сведения об операции. Сведения об операции помимо информации о сертификате и типе создаваемой подписи могут содержать информацию о документе. Информация о документе бывает трёх типов:

1. Неформатированные сведения о документе.
2. Форматированные сведения о документе.
3. Содержимое PDF документа.

Тип (1) можно использовать для передачи названия документа или названия подписываемого файла. Информация данного типа передаётся в запросе на создание транзакции (см. пункт 8.2.2.4.2) в значении параметра DocumentInfo:

```
{"OperationCode":”SignDocument”,"Parameters":[{"Name":"DocumentInfo","Value":"do
cumentname.pdf"},{"Name":"SignatureType","Value":"CMS"},{"Name":"CertificateID",
"Value":"6464"},{"Name":"DocumentType","Value":"mydsssample"}],"Document":"AQI="
}
```

Тип (2) представляет собой данные в формате DTBS (Data To Be Signed, см. соответствующий раздел руководства разработчика). Для их получения на сервере DSS должны быть настроены специальные конвертеры, которые преобразуют содержимое документа в DTBS. Нужный конвертер подбирается на основе типа документа, передаваемого при создании транзакции в значении параметра DocumentType.

По умолчанию, в DSS присутствует только конвертер-заглушка, осуществляющий преобразование «1 к 1». То есть на вход операции подписи (в параметре Document) должно быть изначально передано содержимое в формате DTBS. DocumentType при этом должен быть равен dtbs.

Пример dtbs документа

```
<?xml version="1.0" encoding="utf-8"?><dtbs
xmlns="http://www.cryptopro.ru/schemas/2014/08/dtbs"><row><name>Номер счёта
получателя</name><value>1490690623</value></row><row><name>Номер счёта
отправителя</name><value>1490690633</value></row><row><name>Сумма</name><value>1
00</value></row></dtbs>
```

Пример dtbs документа после BASE64 преобразования (считая, что исходная строка в UTF-8):

```
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz48ZHRicyB4bWxucz0iaHR0cDovL3d3
dy5jcnlwdG9wcm8ucnUvc2NoZW1hcy8yMDE0LzA4L2R0YnMiPjxyb3c+PG5hbWU+0J3QvtC80LXRgCDR
gdGH0ZHRgtCwINC/0L7Qu9GD0YfQsNGC0LXQu9GPPC9uYW1lPjx2YWx1ZT4xNDkwNjkwNjIzPC92YWx1
ZT48L3Jvdz48cm93PjxuYW1lPtCd0L7QvNC10YAg0YHRh9GR0YLQsCDQvtGC0L/RgNCw0LLQuNGC0LXQ
u9GPPC9uYW1lPjx2YWx1ZT4xNDkwNjkwNjMzPC92YWx1ZT48L3Jvdz48cm93PjxuYW1lPtCh0YPQvNC8
0LA8L25hbWU+PHZhbHVlPjEwMDwvdmFsdWU+PC9yb3c+PC9kdGJzPg==
```

Пример запроса на создание транзакции

```
{"OperationCode":”SignDocument”,"Parameters":[{"Name":"DocumentInfo","Value":"do
cumentname.pdf"},{"Name":"SignatureType","Value":"CMS"},{"Name":"CertificateID",
"Value":"6464"},{"Name":"DocumentType","Value":"dtbs"}],"Document":"
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz48ZHRicyB4bWxucz0iaHR0cDovL3d3
dy5jcnlwdG9wcm8ucnUvc2NoZW1hcy8yMDE0LzA4L2R0YnMiPjxyb3c+PG5hbWU+0J3QvtC80LXRgCDR
gdGH0ZHRgtCwINC/0L7Qu9GD0YfQsNGC0LXQu9GPPC9uYW1lPjx2YWx1ZT4xNDkwNjkwNjIzPC92YWx1
ZT48L3Jvdz48cm93PjxuYW1lPtCd0L7QvNC10YAg0YHRh9GR0YLQsCDQvtGC0L/RgNCw0LLQuNGC0LXQ
u9GPPC9uYW1lPjx2YWx1ZT4xNDkwNjkwNjMzPC92YWx1ZT48L3Jvdz48cm93PjxuYW1lPtCh0YPQvNC8
0LA8L25hbWU+PHZhbHVlPjEwMDwvdmFsdWU+PC9yb3c+PC9kdGJzPg=="}
```

Тип (3) используется при PDF подписи (SignatureType равен PDF). В этом случае содержимое документа (значение параметра Document) рассматривается как PDF документ и передаётся в приложение myDss в неизменном виде.

8.3.2.4. Смена ключа myDss

Исходные данные: логин пользователя, сертификат аутентификации оператора DSS, контактные данные пользователя (адрес электронной почты или номер мобильного телефона).

Действие выполняется Оператором.

Для смены ключа клиентское приложение должно отправить запрос следующего вида

Запрос

```
PATCH /STS/ums/user/20948431-2f7e-4825-b723-d56ea134298c/mobileauth HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: dss.cryptopro.ru
Content-Length: 69

{"UserContactInfo":"79150000000","UserContactInfoType":"PhoneNumber"}
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Expires: -1
Date: Tue, 28 Feb 2017 20:13:56 GMT
Content-Length: 7683
{"ExternalUserId":"e59b4978-b171-4ea3-a3ff-
aac6a3b4f3b3","QrCode":"R0l…","KeyExpirationTime":"2018-02-28T00:00:00"}
```

В ответе сервер возвращает JSON-объект, содержащий QR-код, который следует отобразить пользователю.

8.3.2.5. Поиск пользователя

Исходные данные: логин пользователя, сертификат аутентификации оператора DSS.

Действие выполняется Оператором.

Запрос

```
GET /STS/ums/user/?type=login&value=my-dss-test-1489649228 HTTP/1.1
Host: dss.cryptopro.ru
```

В запросе клиентское приложение должно передать логин пользователя.

В ответ на этот запрос сервер сформирует следующий результат.

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Thu, 16 Mar 2017 07:29:35 GMT
Content-Length: 344

{"UserId":"31c7f78d-1a41-44a1-85c0-5c7bb0ee3eea","Login":"my-dss-test-
1489649228","PhoneNumber":null,"Email":null,"PhoneConfirmed":false,"EmailConfirm
ed":false,"DisplayName":null,"DistinguishName":null,"AccountLocked":false,"Group
":"Default","CreationDate":"2017-03-
16T10:27:11.577","LockoutDate":null,"LastLoginDate":"2017-03-16T10:27:11.577"}
```

В теле запроса содержится информация о пользователя в виде JSON-объекта.

Уникальный идентификатор пользователя представляет собой значение свойства **UserId** полученного объекта.

Если по заданному логину не удалось найти пользователя, сервер вернёт HTTP-ответ со статусом 404 (NOT FOUND).

Ответ

```
HTTP/1.1 404 Not Found
```

8.3.2.6. Удаление запросов и сертификатов пользователя

Исходные данные: логин пользователя, сертификат аутентификации оператора.

Перед выполнением данной операции необходимо получит маркер доступа согласно процедуре, описанной в пунктах 2.2.1 – 2.2.2.

Для удаления всех запросов требуется отправить следующий DELETE запрос на конечную точку SsAppName/rest/api/requests.

```
DELETE /SignServer/rest/api/certificates/0 HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJ…
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 0
Date: Thu, 16 Mar 2017 08:44:51 GMT
```

Для удаления всех сертификатов требуется отправить следующий DELETE запрос на конечную точку SsAppName/rest/api/certificates.

Запрос

```
DELETE /SignServer/rest/api/certificates/0 HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJ…
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: 0
Date: Thu, 16 Mar 2017 08:44:51 GMT
```

Для аутентификации при выполнении обоих запросов используется маркер доступа. В примерах он приведён в урезанном виде для удобства чтения.

8.3.2.7. Удаление учётной записи пользователя

Исходные данные: уникальный идентификатор пользователя, сертификат аутентификации оператора DSS.

Действие выполняется Оператором.

Важно! Удаление пользователя не приводит к удалению его запросов на сертификат и самих сертификатов. Удаление этих объектов должно выполняться отдельно.

Для удаления пользователя приложение должно сформировать HTTP DELETE запрос следующего вида

Запрос

```
DELETE /STS/ums/user/9cf94797-1d02-4076-94fe-284a64e83582 HTTP/1.1
Host: dss.cryptopro.ru
Content-Length: 0
```

Ответ

```
HTTP/1.1 200 OK
```

Если пользователь с заданным **id** не найден, сервер **не** вернёт ошибку.

9. Создание и настройка модуля отправки SMS

9.1. Подключаемые модули

Все подключаемые модули, используемые в КриптоПро DSS, должны представлять собой сборку .NET и реализовывать интерфейс **IDSSPlugin**, описанный в сборке **CryptoPro.DSS.Common.dll**.

Интерфейс предоставляет метод **Initialize**, который должен инициализировать подключаемый модуль по параметрам, передаваемым на вход данного метода в виде словаря.

```
    public interface IDSSPlugin
    {
        void Initialize(IDictionary<string, string> parameters);
    }
```

9.2. Описание модуля отправки SMS

Модуль отправки коротких текстовых сообщений (SMS) должен представлять собой сборку .NET. Сборка должна содержать единственный публичный класс, реализующий интерфейс **ISmsPlugin**. Интерфейс **ISmsPlugin** описан в сборке **CryptoPro.DSS.Common.dll**.

Интерфейс предоставляет метод **SendSms**, предназначенный для отправки SMS, определён следующим образом:

```
public interface ISmsPlugin : IDSSPlugin
{
    void SendSms(string message, string phone);
}
```

Метод **SendSms** принимает следующие параметры:

- **message** – текстовое сообщение;
- **phone** – номер телефона;

Список дополнительных параметров задаётся с помощью командлета Set-DSSStsProperties (см. пункт 7.1.1 Руководства администратора)

При возникновении ошибки в процессе выполнения метода **SendSms** необходимо сгенерировать исключение **SmsException**.

9.3. Пример готового модуля отправки SMS

В данном разделе приведен пример модуля отправки коротких текстовых сообщений (SMS).

```
using System.Collections.Generic;
using System.Web.Services.Protocols;
using DSS.Common.ExternalUsage;

namespace DSS.SmsService.DevinoSms
{
    public class DevinoSmsPlugin : ISmsPlugin
    {
        public void SendSms(string message, string phone)
        {
            try
            {
                SmsService smsService = new SmsService();
                string sessionID =
```

```
                    smsService.GetSessionID(smsLogin, smsPasswd);

                Message smsMessage = new Message();
                smsMessage.Data = message;
                smsMessage.SourceAddress = sourceAddress;
                smsMessage.DestinationAddresses = new[] { phone };
                smsService.SendMessage(sessionID, smsMessage);
            }
            catch (SoapException ex)
            {
                if (ex.Message.Contains("Invalid destination address"))
                    throw new SmsException("Неверный номер телефона", ex);
                throw new SmsException(
                    "Произошла ошибка при отправке SMS", ex);
            }
        }
        public void Initialize(IDictionary<string, string> parameters)
        {
            smsLogin = parameters["login"];
            smsPasswd = parameters["password"];
            sourceAddress = parameters["sourceaddress"];
        }
    }
}
```

10. Создание и настройка модуля для преобразования форматов

Модуль для преобразования форматов должен представлять собой сборку .NET. Сборка должна содержать публичный класс, реализующий интерфейс **IDSSDocumentConverter**. Интерфейс **IDSSDocumentConverter** описан в сборке **CryptoPro.DSS.Common.dll**.

```
public interface IDSSDocumentConverter : IDSSPlugin
{
    byte[] Convert(byte[] documentContent, out DssConvertArgs args);
        Stream Convert(Stream documentContent, out DssConvertArgs args);
}
```

Интерфейс **IDSSPlugin** предоставляет метод **Initialize**, который должен инициализировать подключаемый модуль по параметрам, передаваемым на вход данного метода в виде словаря.

```
    public interface IDSSPlugin
    {
        void Initialize(IDictionary<string, string> parameters);
    }
```

Список дополнительных параметров задаётся при регистрации плагина с помощью командлетов Add-DSSFEConverterPlugin или Add-DSSSTSConverterPlugin (см. Руководства администратора).

Метод **Convert** принимает следующие параметры:

- **documentContent** – входные данные;
- **args** – объект, в который будут записаны дополнительные параметры о результатах преобразования формата;

При успешном завершении работы метод возвращает сконвертированный документ в виде массива байт или потока соответственно. Если при конвертировании произошла ошибка, необходимо вернуть null.

Класс DssConvertArgs описан в сборке CryptoPro.**DSS.Common.dll**.

```
public class DssConvertArgs
{
    public bool CanProceed;
        public OutputFormat Format;
    public Exception ConverterException;
}
public enum OutputFormat
{
      PDF,
      UNCONVERTED,
      ETOKEN_INIT_DATATABLE,
}
```

При завершении работы метода **Convert** необходимо заполнить параметр **DssConvertArgs args**:

- если плагин не может преобразовать входные данные, выставить значение **args.CanProceed** в false;
- при возникновении исключения в методе **Convert** записать его в **args.ConverterException**;
- если преобразование формата было выполнено успешно, то выставить значение **args.CanProceed** в true и заполнить значение **args.Format**.

11. Изменения SOAP-интерфейса Сервера Подписи

11.1. Изменения интерфейса в версии 1.0.1146

11.1.1. Функции

1. SignDocuments
   Добавлен в версии 1.0.1146
2. SetRequestStatus
   Добавлен в версии 1.0.1146
3. CreateRequestEx
   Добавлен в версии 1.0.1146
4. GetRevokeRequests
   Добавлен в версии 1.0.1146
5. GetRequestContent
   Добавлены параметры requestType, caID.

11.1.2. Типы данных

1. DSSSignDocumentResponse
   Добавлен в версии 1.0.1146.
2. DSSCertRequest
   Добавлены поля CertificateID, RequestType.
3. DSSRevRequest
   Добавлен в версии 1.0.1146.
4. DSSCertificate
   Добавлено поле CertificateAuthorityID.
5. DSSCAPolicy
   Добавлены поля AllowUserMode, CAType.
6. CertificateStatus
   Добавлены поля PinCode, ActiveCertID.
7. RevocationInfo
   Добавлены поля SerialNumber, UnholdAction, ApproveDate, RequestDate. Удалены поля DistName, UnholdDuration. У поля RevocationReason изменился тип.

11.1.3. Перечислимые типы данных

1. DSSRequestStatusEnum
   Удалены поля FIRST, NOT_REGISTERED. Из описания типа удалён атрибут Flags.
2. DSSCAType
   Добавлен в версии 1.0.1146.
3. CARequestTypeEnum
   Добавлен в версии 1.0.1146.
4. CertRevokeReasonEnum
   Добавлен в версии 1.0.1146.

5. RequestParams
   Добавлен в версии 1.0.1146.

11.2. Изменения интерфейса в версии 1.0.1162

11.2.1. Типы данных

1. DSSPolicy

Изменён тип поля TransactionConfirmation с bool на MfaPolicy.
Добавлено поле ActionPolicy.

2. DSSAction

Добавлен в версии 1.0.1162

11.2.2. Перечислимые типы данных

1. MfaPolicy

Добавлен в версии 1.0.1162.

2. DSSActions
3. Добавлен в версии 1.0.1162.

11.3. Изменения интерфейса в версии 1.0.1555

11.3.1. Перечислимые типы данных

1. SignatureParams

Перечисление расширено двумя параметрами PdfSignatureAppearance и PdfSignatureTemplateId.

11.4. Изменения интерфейса в версии 1.0.1777

11.4.1. Функции

1. EnhanceSignature
2. EnhanceSignature
3. EnhanceSignatureRest

11.4.2. Перечисления

1. SignatureParams
   Добавлен тип подписи CAdES-T. Добавлен параметр для включения отображаемых данных в подписанный атрибут подписи формата CMS, PDF. Добавлен параметр для создания параллельной и заверяющей CMS подписи.

11.5. Изменения интерфейса в версии 1.0.2020

11.5.1. Функции

1. Добавлена функция CreateTransactionToken.
2. Функция GetTransactionID отмечена как устаревшая.