Внешнее API

Назначение: сервис предоставляет клиентам возможность использования удаленного КЭП

URL-путь: https://acsk.privatbank.ua/cloud/api/back

Все запросы отправляются с Header: "Content-Type: application/json". Смотреть раздел *Версионирование*

Описание

Подписанный объект

Формат данных: JSON

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	signedData		Подписанные данные	String (Base64)	Да	

Зашифрованный объект (запрос)

Формат данных: JSON

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	authData		Зашифрованный сессионный ключ	String (Base64)	Да	
	encryptedData		Зашифрованные данные	String (Base64)	Да	

Зашифрованный объект (ответ)

Формат данных: JSON

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	encryptedData		Зашифрованные данные	String (Base64)	Да	

Ошибки

Формат данных: JSON

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	errorCode		Код ошибки	int	Да	
	errorMessage		Описание ошибки	String	Да	

Получение сертификата сервиса подписей

Формат запроса GET

URL: /get-certificates

Актуальная версия: v1
Дефолтная версия: v1

Формат ответа

Формат данных: JSON

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	certificates		Список сертификатов сервиса подписей	List<String> (Base64)	Да	

Ответ успешный

Response


```
{"certificates":["MIIF9jCCBZ6gAwIBAgIUK2x9+aO...i+33Ft4lqa2lC4D9ISdKHspzDxqAhcQ5xpr3fc=="]}
```


Получение идентификатора сессии

Формат запроса POST

Формат данных: JSON

URL: /acquire-operation-id

Актуальная версия: v1
Дефолтная версия: v1

Описание запроса

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	clientId		Уникальный идентификатор PREFIX_UUID PREFIX - идентификатор системы присылающей запросы	String	Да	DIYA_550E8400-E29b-41D4-A716-446655440000

Запрос

Request

```
{"clientId":"DIYA_550E8400-E29b-41D4-A716-446655440000"}
```

Дале запрос шифруется и оформляется по схеме **"Зашифрованный объект (запрос)"**

Описание ответа

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	operationId		Идентификатор сессии	String (Base64) 88 символов	Да	

Ответ успешный

Response

```
{"operationId":"WeyMfXq32WOukwDcpHTjZa/Alyk4X+OlyDcYVtDool18Lwd6jZDi1ZOosF5/QEj5tuPrFeQQDJAAEIWDcYVtD=="}
```

В ответе получаем данные по схеме **"Зашифрованный объект (ответ)"**

После расшифрования данных получаем **"Подписанный объект"**

После проверки подписи получаем уже объект содержащий идентификатор сессии.

Запрос на формирование подписи

Формат запроса POST

Формат данных: JSON

URL: /sign

Актуальная версия: v1
Дефолтная версия: v1

Описание запроса

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	clientId		Уникальный идентификатор PREFIX_UUID PREFIX - идентификатор системы присылающей запросы	String	Да	DIYA_550E8400-E29b-41D4-A716-446655440000
	operationId		Идентификатор сессии	String (Base64) 88 символов	Да	
	time		Время формирования запроса	String (yyyy-MM-dd HH:mm:ss)	Да	2019-09-16 10:21:45

	originatorDescription		Идентификатор системы в которой был сформирован запрос	String	Да	DIYA
	operationDescription		Идентификатор операции в системе которая сформировала запрос	String	Нет	
	operationDescriptions		Список наименований документов которые будут подписаны	List<String>	Нет	["Договор залога"," Акт выполненных работ"]
	hashes		Список хешей от документов которые необходимо подписать	List<String> (Base64) * Масимальный размер списка == 40	Да	
	signatureAlgorithmName		Название алгоритма подписи	const String: **DSTU4145**	Да	
	signatureFormat		Формат запрашиваемой подписи	const String: **PKCS7**	Да	
	callbackUrl ***		URL для уведомления, сервис подписи отправит на этот адрес данные для получения подписи после того как она будет сформирована. *** При формировании callbackUrl нужно использовать только https с стандартным портом: 443	String	Нет	
	approveTimeout		Максимальное время ожидания для подтверждения заявки клиентом. Время по умолчанию 30 мин. Минимальное значение 30 мин. Максимальное 2880 мин.	int	Нет	60

Запрос

Request

```
{
    "clientId":"DIYA_550E8400-E29b-41D4-A716-446655440000",
    "operationId":"WeyMfXq32WOukwDcpHTjZa/Alyk4X+OlyDcYVtDool18Lwd6jZDi1ZOosF5/QEj5tuPrFeQQDJAAEIWDcYVtD==",
    "time":"2019-09-16 10:21:45",
    "originatorDescription":"DIYA",
    "operationDescription":"QAZWSX123",
    "hashes":["yk4X+OlyDcYVtDool18Lwd6jZDi1ZOosF5/QEj5tuPr==", "Dool18Lwd6jZDi1ZOyk4X+OlyDcYVtosF5/QEj5tuPr=="],
    "signatureAlgorithmName":"DSTU4145",
    "signatureFormat":"PKCS7"
}
```

Дале запрос шифруется и оформляется по схеме **"Зашифрованный объект (запрос)"**

Описание ответа

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	status		Статус обработки запроса	int на данный момент возможны следующие статусы: 1 - операция начата	Да	

Ответ успешный

Response

```
{"status":1}
```

В ответе получаем данные по схеме **"Зашифрованный объект (ответ)"**

После расшифрования данных получаем **"Подписанный объект"**

После проверки подписи получаем уже объект содержащий статус обработки запроса.

Примечание

Если в запросе был указан параметр **callbackUrl** то после формирования подписи сервис подписей отправит на указанный URL запрос в формате:

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	clientId		Уникальный идентификатор PREFIX_UUID PREFIX - идентификатор системы присылающей запросы	String	Да	DIYA_550E8400-E29b-41D4-A716-446655440000
	operationId		Идентификатор сессии	String (Base64) 88 символов	Да	
	errorCode		Код ошибки	int	Нет	
	errorMessage		Описание ошибки	String	Нет	

Request

```
{
        "clientId":"DIYA_550E8400-E29b-41D4-A716-446655440000",
        "operationId":"WeyMfXq32WOukwDcpHTjZa/Alyk4X+OlyDcYVtDool18Lwd6jZDi1ZOosF5/QEj5tuPrFeQQDJAAEIWDcYVtD=="
}
```

Данный запрос не шифруется

Передача данных в Приват24 моб. через QR-код

Описание запроса

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	clientId		Уникальный идентификатор PREFIX_UUID PREFIX - идентификатор системы присылающей запросы	String	Да	DIYA_550E8400-E29b-41D4-A716-446655440000
	operationId		Идентификатор сессии	String (Base64) 88 символов	Да	

Формирование QR-кода

Сервис который отправил запрос на подпись должен сформировать URL:

https://www.privat24.ua/rd/kep?hash=rd/kep/{"clientId":"clientId","operationId":"operationId","action":"action"}

где:

clientId - Уникальный идентификатор PREFIX_UUID. PREFIX - идентификатор системы присылающей запросы;
operationId - Идентификатор сессии внешнего сервиса;
action - тип операции:

- auth - операция авторизации, при передачи данного типа, необходимо отобразить тексты для авторизации
- sign - операция подписи, при передачи данного типа, необходимо отобразить тексты для подписи

Пример ссылки:

https://www.privat24.ua/rd/kep?hash=rd/kep/{"clientId":"DIYA_550E8400-E29b-41D4-A716-446655440000","operationId":"WeyMfXq32WOukwDcpHTjZa/Alyk4X+OlyDcYVtDool18Lwd6jZDi1ZOosF5/QEj5tuPrFeQQDJAAEIWDcYVtD==","action":"sign"}

Сформированный URL необходимо преобразовать в QR-код и отобразить клиенту.

Запрос статуса формирования подписи

Формат запроса POST

Формат данных: JSON

URL: /sign-status

Актуальная версия: v1
Дефолтная версия: v1

Описание запроса

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных
	clientId		Уникальный идентификатор PREFIX_UUID PREFIX - идентификатор системы присылающей запросы	String	Да	DIYA_550E8400-E29b-41D4-A716-446655440000
	operationId		Идентификатор сессии	String (Base64) 88 символов	Да	

Запрос

Request

```
{
    "clientId":"DIYA_550E8400-E29b-41D4-A716-446655440000",
    "operationId":"WeyMfXq32WOukwDcpHTjZa/Alyk4X+OlyDcYVtDool18Lwd6jZDi1ZOosF5/QEj5tuPrFeQQDJAAEIWDcYVtD=="
}
```

Дале запрос шифруется и оформляется по схеме **"Зашифрованный объект (запрос)"**

Описание ответа

Родительский тэг	Тэг	Атрибут	Описание	Формат данных	Обязательное	Пример данных

	status		Статус обработки запроса	int на данный момент возможны следующие статусы: 0 - операция зарегистрирована (под операцию был выделен идентификатор) 1 - операция начата (отправлен запрос клиенту) 2 - операция обработана (клиент подтвердил запрос) 3 - операция отклонена (клиент отклонил запрос) 4 - ошибка (при обработке запроса клиентом произошла ошибка)	Да	
signatures			Список с результатами подписания	List	Нет	
	hash		Хеш документа	String (Base64)	Да	
	signature		Подпись	String (Base64)	Нет	
	errorCode		Код ошибки	int	Нет	
	errorMessage		Текст ошибки	String	Нет	

Ответ успешный

Response

```
{
        "status":2,
        "signatures":[
                {
                        "hash":"yk4X+OlyDcYVtDool18Lwd6jZDi1ZOosF5/QEj5tuPr==",
                        "signature":"WeyMfXq32WOukwDcpHTjZa/Alyk4X+OlyDcYVtDool18Lwd6jZDi1ZOosF5
/QEj5tuPrFeQQDJAAEIWDcYVtD=="
                },
                {
                        "hash":"Dool18Lwd6jZDi1ZOyk4X+OlyDcYVtosF5/QEj5tuPr==",
                        "errorCode":504,
                        "errorMessage":"    .     "
                }
}
```

В ответе получаем данные по схеме **"Зашифрованный объект (ответ)"**

После расшифрования данных получаем **"Подписанный объект"**

После проверки подписи получаем уже объект содержащий статус обработки запроса и подпись.

Библиотека

Пример настройки библиотеки

Завантаження бібліотеки

Бібліотека EUSignJava.jar завантажується прикладними програмами засобами середовища виконання java (JRE).

Після завантаження бібліотека EUSignJava.jar автоматично розархівує необхідні для роботи криптографічні бібліотеки в каталог, що вказується прикладними програмами. Це повинен бути або каталог, в який було встановлено прикладну програму, або будь-який інший каталог, за наявності прав запису та виконання до нього.

Встановлення каталогу, в який буде разархівовуватися:

```
import com.iit.certificateAuthority.endUser.libraries.signJava.*;

public class EUSignApplication extends Application{

    public EndUser endUser;

    @Override

    public void onCreate() {

        super.onCreate();

        EndUserResourceExtractor.SetPath(this.getFilesDir().getAbsolutePath());

        EndUserResourceExtractor.SetPreLoad(true);

        endUser = new EndUser();

   }

}
```

Примітка. Встановлення каталогу, в який буде разархівовуватися програма повинно відбуватися до створення об'єкту EndUser.

Примітка. Підтримка архітектури armeabi-v7a виконується за рахунок автоматичного використання бібліотек для архітектури armeabi (для зменшення розміру програми). При підключенні в проект нативних бібліотек сторонніх виробників, необхідно використовувати лише версію для архітектури armeabi. Використання сторонніх бібліотек з архітектурою armeabi-v7a призводить до не можливості автоматичного використання бібліотек архітектури armeabi та помилок при їх завантаженні.

Настройка библиотеки

```
private void init() throws Exception {
   endUser.SetUIMode(false);
   endUser.SetLanguage(EndUser.EU_RU_LANG);
   endUser.SetCharset("UTF-8");

   try {
      endUser.Initialize();
      EndUserModeSettings modeSettings = new EndUserModeSettings(false);
      modeSettings.SetOfflineMode(false);
      endUser.SetModeSettings(modeSettings);

      EndUserProxySettings proxySettings = endUser.CreateProxySettings();
      proxySettings.SetUseProxy(false);
      proxySettings.SetAnonymous(true);
      proxySettings.SetAddress("");
      proxySettings.SetPort("");
      proxySettings.SetUser("");
      proxySettings.SetPassword("");
      proxySettings.SetSavePassword(false);
      endUser.SetProxySettings(proxySettings);

      EndUserFileStoreSettings fileStoreSettings = endUser.CreateFileStoreSettings();
      fileStoreSettings.SetPath("/home/vadym/pblib/Storage");
      fileStoreSettings.SetSaveLoadedCerts(true);
      endUser.SetFileStoreSettings(fileStoreSettings);

      EndUserCMPSettings cmpSettings = endUser.CreateCMPSettings();
      cmpSettings.SetUseCMP(true);
      cmpSettings.SetAddress("http://acsk.privatbank.ua/services/cmp/");
      cmpSettings.SetPort("80");
      cmpSettings.SetCommonName("");
      endUser.SetCMPSettings(cmpSettings);
```

```
        EndUserTSPSettings tspSettings = endUser.CreateTSPSettings();
        tspSettings.SetAddress("http://acsk.privatbank.ua/services/tsp/");
        tspSettings.SetPort("80");
        tspSettings.SetGetStamps(true);
        endUser.SetTSPSettings(tspSettings);

        EndUserOCSPSettings ocspSettings = endUser.CreateOCSPSettings();
        ocspSettings.SetAddress("http://acsk.privatbank.ua/services/ocsp/");
        ocspSettings.SetPort("80");
        ocspSettings.SetUseOCSP(true);
        endUser.SetOCSPSettings(ocspSettings);

        EndUserLDAPSettings lDAPSettings = endUser.CreateLDAPSettings();
        endUser.SetLDAPSettings(lDAPSettings);

    } catch (Exception e) {
        LOG.error("[EUSign] Initialization ERROR:", e);
    }

}
```

Шифрование

```
//Клиент
//cert - сертификат который получили с сервера подписей
byte[] serverCert = Base64.getDecoder().decode(cert);



//создаем сессию клиента и ключи которые используются для шифрования данных

//900 - время жизни сессии в секундах
EndUserSession clientSession = endUser.ClientDynamicKeySessionCreate(900, serverCert);



//шифруем данные клиента

//req - данные которые необходимо зашифровать
String encryptedData = endUser.SessionEncrypt(clientSession, req);



//готовим сессию для передачи на сервер
String authData = Base64.getEncoder().encodeToString(clientSession.GetData());



//расшифровать ответ сервера

byte[] decrResp = endUser.SessionDecrypt(clientSession, serverEncryptedResponse);
```

Формирование хеша

Хеш вычисляется с документа который необходимо подписать. Для вычисления хеша используется алгоритм GOST34311.

Метод вычисления хеша (**class EndUser**): **public String Hash(byte[] data) throws Exception;**

где **data** - данные который необходимо подписать

метод возвращает хеш документа в виде Base64 строки.

Библиотека JS

Настройки библиотеки

Бібліотека автоматично завантажується браузером використовуючи мову сценаріїв JavaScript.

Перед використанням бібліотека повинна бути розміщена на сервері. На web-(HTML-)сторінку, де використовується бібліотека підключається наступним чином:

```
<script type="text/JavaScript" src="euscpt.js"></script>

<script type="text/JavaScript" src="euscpm.js"></script>

<script async type="text/javascript" src="euscp.js"></script>
```

Після завантаження модуля бібліотеки буде автоматично викликана функція сповіщення про завантаження:

```
function EUSignCPModuleLoaded();
```

За замовчанням після завантаження бібліотека буде автоматично проініціалізована та буде викликана функція сповіщення про ініціалізацію:

```
function EUSignCPModuleInitialized([boolean] isInitialized);
```

Використання функцій бібліотеки можливе тільки пілся її завантаження та ініціалізаці.

Доступ до функцій бібліотеки виконується за рахунок звернення до об’єкту типу EUSignCP:

```
var endUser = EUSignCP();
```

Примітка. Бібліотека виконується в основному потоці NodeJS та блокує його. Для зменшення навантаження на процесор під час запитів до серверів АЦСК рекомендується додатково встановити модуль sleep:

```
npm install sleep
```

Примітка. Через політику безпеки web-браузерів, взаємодія з серверами ЦСК можлива лише за умови реалізації механізма кросдоменної взаємодії. Для реалізації взаємодії між доменами, сервер, на якому розміщено бібліотеку повинен, реалізовувати механізм proxy-сервісу. Бібліотека, робить синхронний запит (POST або GET) до сервера за протоколом HTTP у вигляді:

```
/proxy?address=http://ca.server.ua, де

/proxy – адреса proxy-сервісу, встановлюється функцією бібліотеки SetXMLHTTPProxyService;

?address= - параметр, що передається в запиті, містить адресу на яку потрібно перенаправити запит бібліотеки;

http://ca.server.ua – адреса на яку направлено запит бібліотеки.
```

Пример настройка библиотеки для шифрования

```
console.log("Creating new endUser");
const endUser = EUSignCP();

console.log("Setting proxy handler");
endUser.SetXMLHTTPProxyService("/proxyHandler");

endUser.SetErrorMessageLanguage(EU_UA_LANG);
endUser.SetCharset("UTF-8");
endUser.SetJavaStringCompliant(true);

console.log("Setting ModeSettings");
const modeSettings = endUser.CreateModeSettings();
modeSettings.SetOfflineMode(false);
endUser.SetModeSettings(modeSettings);

console.log("Setting ProxySettings");
const proxySettings = endUser.CreateProxySettings();
proxySettings.SetUseProxy(false);
proxySettings.SetAnonymous(true);
proxySettings.SetAddress("");
proxySettings.SetPort("");
proxySettings.SetUser("");
proxySettings.SetPassword("");
proxySettings.SetSavePassword(false);
endUser.SetProxySettings(proxySettings);
```

```
console.log("Setting FileStoreSettings");
const fileStoreSettings = endUser.CreateFileStoreSettings();
fileStoreSettings.SetPath("/cert");
fileStoreSettings.SetSaveLoadedCerts(true);
endUser.SetFileStoreSettings(fileStoreSettings);

console.log("Setting CMPSettings");
const cmpSettings = endUser.CreateCMPSettings();
cmpSettings.SetUseCMP(true);
cmpSettings.SetAddress("http://acsk.privatbank.ua/services/cmp/");
cmpSettings.SetPort("80");
cmpSettings.SetCommonName("");
endUser.SetCMPSettings(cmpSettings);

console.log("Setting TSPSettings");
const tspSettings = endUser.CreateTSPSettings();
tspSettings.SetAddress("http://acsk.privatbank.ua/services/tsp/");
tspSettings.SetPort("80");
tspSettings.SetGetStamps(true);
endUser.SetTSPSettings(tspSettings);

console.log("Setting OCSPSettings");
const ocspSettings = endUser.CreateOCSPSettings();
ocspSettings.SetAddress("http://acsk.privatbank.ua/services/ocsp/");
ocspSettings.SetPort("80");
ocspSettings.SetUseOCSP(true);
endUser.SetOCSPSettings(ocspSettings);

console.log("Setting LDAPSettings");
const lDAPSettings = endUser.CreateLDAPSettings();
endUser.SetLDAPSettings(lDAPSettings);
```

Шифрование

```
console.log("Getting server certificate");
const serverCertBase64 = "MIIF4DCCBYigAwIBAgIUK2x9+aOJHaEEAAAA...";
const serverCertBytes = clientEndUser.Base64Decode(serverCertBase64);

console.log("Creating client session");
const sessionExpireTime = 900;
const endUserClientSession = endUser.ClientDynamicKeySessionCreate(sessionExpireTime, serverCertBytes);

console.log("Encrypting data");
const dataToEncrypt = "Hello World";
const dataToEncryptBytes = endUser.StringToArray(dataToEncrypt);
const encryptedDataBytes = endUser.SessionEncrypt(endUserClientSession, dataToEncryptBytes);
const encryptedDataBase64 = endUser.Base64Encode(encryptedDataBytes);
console.log("Encrypted data", encryptedDataBase64);

console.log("Getting authData");
const authDataBytes = endUserClientSession.GetData();
const authDataBase64 = endUser.Base64Encode(authDataBytes);
console.log("AuthData", authDataBase64);

console.log("Sending authDataBase64 and encryptedDataBase64 to server");
const encryptedServerResponseBase64 = sendEncryptedRequest(authDataBase64, encryptedDataBase64)
console.log("Encrypted server response", encryptedServerResponseBase64);

console.log("Decrypting server response on client side");
const decryptResponseBytes = endUser.SessionDecrypt(endUserClientSession, encryptedServerResponseBytes);
const decryptedResponse = endUser.ArrayToString(decryptResponseBytes);
console.log("Decrypted response", decryptedResponse);
```

Формирование хеша

```
console.log("Creating hash Base64");
const dataToHash = "Hello World";
const dataToHashBytes = endUser.StringToArray(dataToHash);
const hash = endUser.HashData(dataToHashBytes, true);
console.log("Hash Base64", hash)
```

Примеры использования

https://git.privatbank.ua/cdp/iit.libs/-/tree/master/Examples/JS

Формирование запросов

URL		Шифрование	Подпись тех. Ключом
/get-certificates	Запрос	-	-
	Ответ	-	-
/acquire-operation-id	Запрос	+	-
	Ответ	+	+
/sign	Запрос	+	-
	Ответ	+	+
/sign-status	Запрос	+	-
	Ответ	+	+

История изменения

Дата версии	Запрос на ИТ-изменение	Задача на ИТ-изменение	Задача на тестирование	Отчет о тестировании