Гриф С2

Спецификация

на

Локална услуга за електронно подписване в браузър

Версия 3.35

София, Януари 2025

Управление на документа	
Подготвен от	Владимир Методиев
Прегледан от	Владимир Методиев
Одобрен от	Димитър Николов

ПРОМЕНИ НА ВЕРСИЯТА		
Версия	Промяна	Дата/Извършител
1.00	Начално създаване	26.08.2016/ВИ
1.01		
1.02	Добавено • Опции за сваляне и вдигане защитата на MS Edge • Сваляне защитата на MS Edge след инсталация	03.10.2016/ВИ
1.03	Добавено • Подписани елементи в JSON	07.10.2016/ВИ
1.04	Добавено • Проверка на подписаните елементи в JSON	10.10.2016/ВИ
1.09	Добавено • Подписване без извикване на GetSigner	01.03.2017/ВИ
1.12	Добавено • Timeout 15 минути за въвеждане на ПИН • Параметризирано е съобщението за потвърждаване на подписаните данни • Заявка за спиране на приложението	13.04.2017/ВС
1.13	Добавено • Заявка за статуса на подписваните документи от масив • Сменено лого на фирмата по време на инсталация • Скрит бутон за смяна на сертификат • Сменен прозорец за избор на сертификат при MSCapi • Добавени съобщения за различни събития по време на подписване	06.06.2017/ВС 12.07.2017/С. Славов
1.14	Добавено: • Разписване с различни видове карти • Пропърти файл за избиране на карта и нейният dll файл • Прозорец за избиране на dll в случай, че картата не присъства в пропърти файла • Премахнат е прозореца за избор на вида карта при PKCS11 подпис	18.09.2017/С. Славов

	• При инсталиране на BISS за Windows се инсталират и удостоверителните вериги на Борика • Сменена Java версия от 1.8.0_121 на 1.8.0_144	
2.15	• Добавен джоб извикващ версията на BISS през 15 минути • Оправен проблем при прочитането на сертификатите когато има включени повече от една карта • Оправен selector за филтриране на сертификати • При инсталация на Windows инсталатора изключва BISS ако има включен • Добавен параметър charset=utf-8 в хедъра на респонса • Сменена Java версия на 1.8.0_202	31.01.2019/С. Славов
2.16	• Версия пусната за клиентите	18.02.2019/С. Славов
2.18	• BISS връща съобщения за всички бутони по UI-а • Бутона „За програмата“ е преместен на предпоследна позиция отделен от останалите • В заявката за подписване е добавен параметър „additionalConfirmText“ на база, на който се показва допълнителна информация в прозореца за потвърждение на хеша • Параметъра “showAllCerts” е преименуван на “showValidCerts”, защото с другото име не беше много ясно предвид факта, че false връщаше всички, а true само валидните • Валидиране на заявката за подписване BISS ще връща грешка в зависимост от това дали е невалиден подписа или сертификата	26.06.2019/ С.Славов
2.26	• Автоматичен ъпдейт • Добавяне на нов ATR • SIEMENS картите работят с библиотека- LINUX/Mac:cvP11 WINDOWS: cmP11	01.02.2020/Г.Бажлекова
2.28	• Проверката за нова версия на BISS се прави на всеки 2 часа от стартирането на приложението • Ако има отворен прозорец със съобщение за нова версия на BISS -не се отваря нов • Ако не може да се изгради trusted верига на клиентския сертификат се връща съобщение за грешка	01.02.2020/Г. Бажлекова

2.30	• Променена версия на log4j – 2.17.1	21.02.2022/В.Методиев
3.35	• OpenSDK	27.01.2025/В.Методиев

Съдържание

1. Описание.

Приложения софтуер за електронно подписване изпълнява технология, която позволява локално полагане на електронен подпис без използване на активна компонента в Интернет браузър при Клиента. То се базира на спецификация Signature Creation Service 1.0.1 (June 30, 2015), която е допълнена с необходимите функции с цел гарантиране на работоспособността на основното Уеб-базирано приложение.

Приложния софтуер е реализиран като локален Web сървър, който позволява достъп до създаваща подписи функционалност посредством използване на защитен протокол (HTTPS).

При стартиране на Локалната услуга за електронен подпис (BISS) се проверява първият свободен порт от следните портове : 53952, 53953, 53954 и 53955. Следователно HTML приложението също трябва да сканира тези портова за намиране на BISS.

2. Системни изисквания.

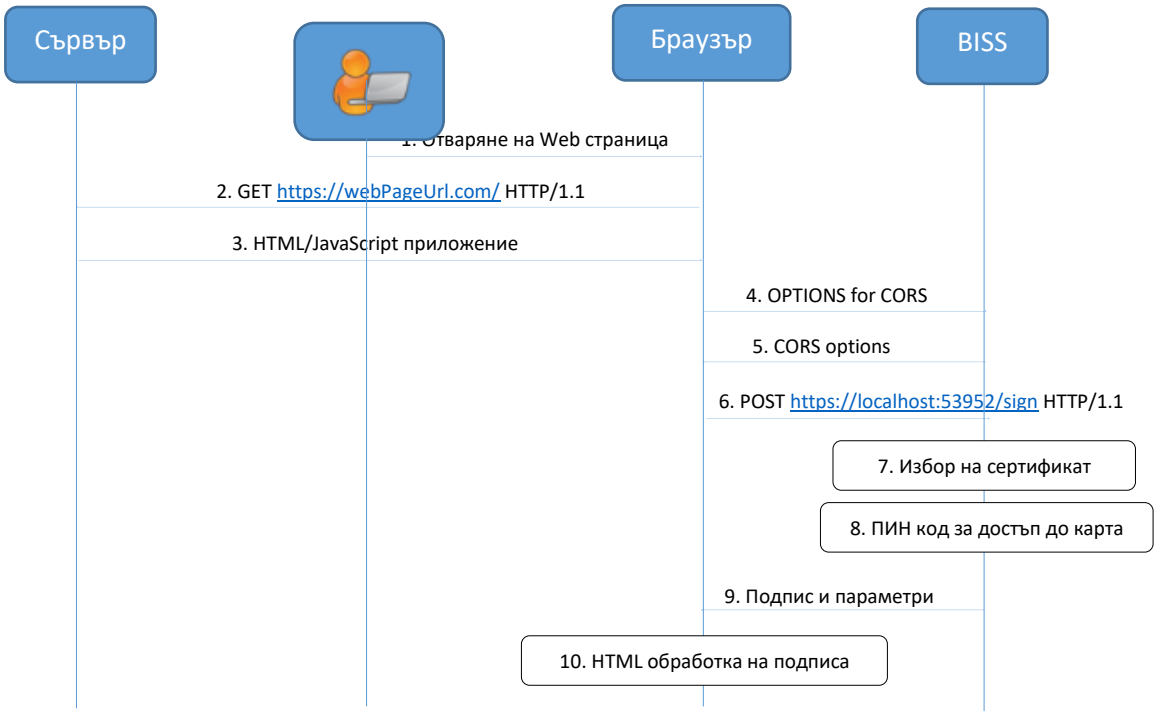
BISS може да работи на следните операционни системи:

- Windows 8/10, 32/64 bit;
- Linux Ubuntu 18.04 LTS, 20.04 LTS
- Mac OS X 10.15 Catalina

За Mac OS X е задължително да има инсталирана Java 8.

3. Процес по полагане на електронен подпис

По – долу е дадена диаграмата описваща процеса по полагане на електронен подпис. Схемата е съсредоточена върху интерфейса между HTML приложението и BISS.



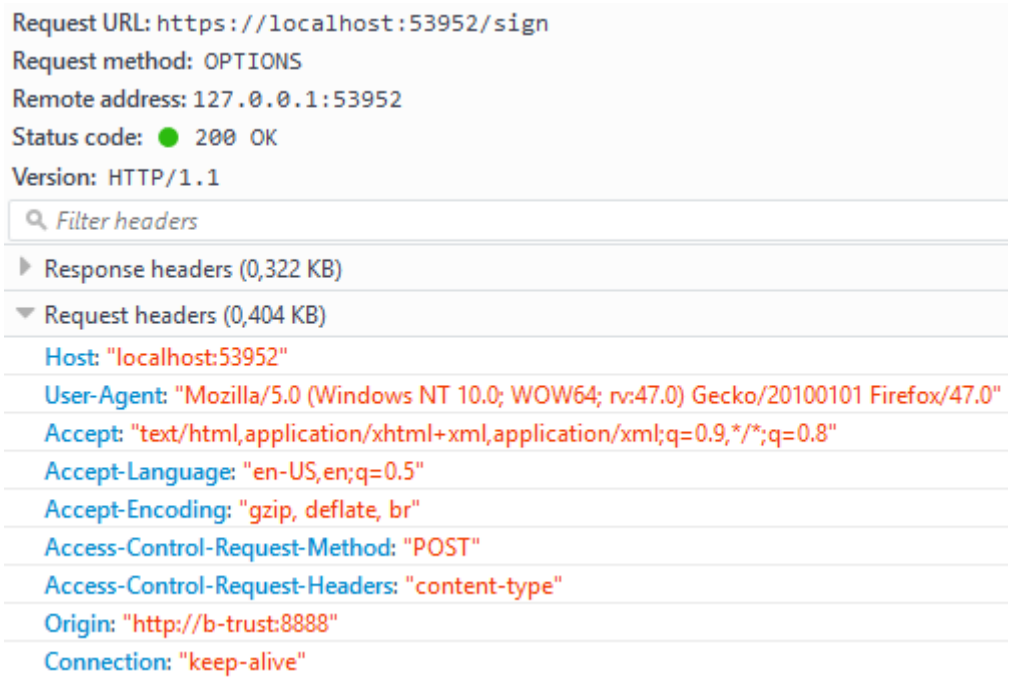
1. Потребителят въвежда в браузъра URL адреса на HTML приложението.
2. Браузъра прави защитена заявка към сървъра на HTML приложението.
3. Браузъра зарежда HTML приложението от сървъра.
4. Браузъра прави OPTION заявка към BISS за да провери дали CORS заявките са позволени.
5. BISS връща CORS параметри позволявайки заявката да бъде направена от текущото HTML приложение (идентифицирано чрез Origin параметъра).
6. HTML приложението създава заявка за създаване на подпис.
7. Потребителят избира сертификат за подписване.
8. Потребителят въвежда ПИН код за достъп до смарт картата.
9. BISS връща отговор на HTML приложението. Отговорът съдържа подписа, алгоритъма на подписване и сертификационната верига.
10. HTML приложението обработва (валидира, създаване на контейнер) подписа.

4. Локална услуга за електронно подписване BISS

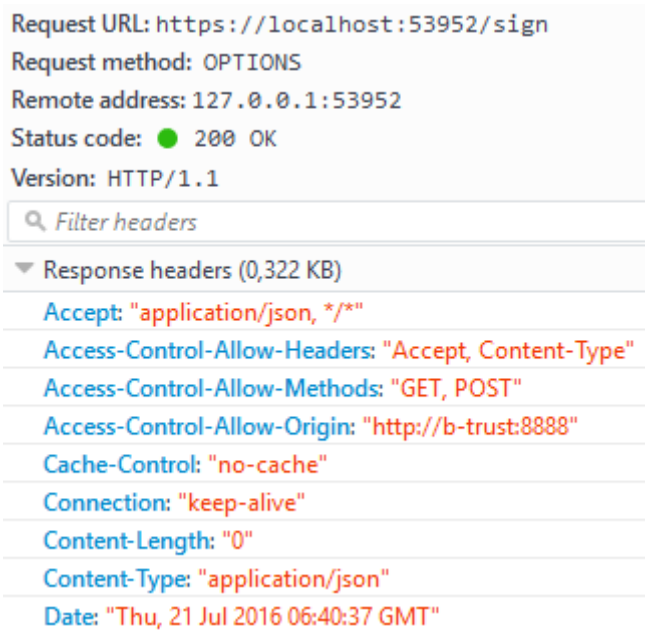
4.1 Cross-Origin Resource Sharing (CORS) проверка

Браузърите, които подържат CORS ще направят проверка дали CORS заявките са позволени да бъдат изпращани към BISS. Това се прави като се изпраща OPTION заявка към BISS указваща Origin атрибута на HTML приложението, както и позволените методи и Header атрибути.

Примерна заявка е показана по долу:



В резултат на OPTION заявката BISS трябва да върне HTTP отговор със указани CORS атрибути. Пример на такъв отговор е показан по долу:



4.2 Проверка на версията

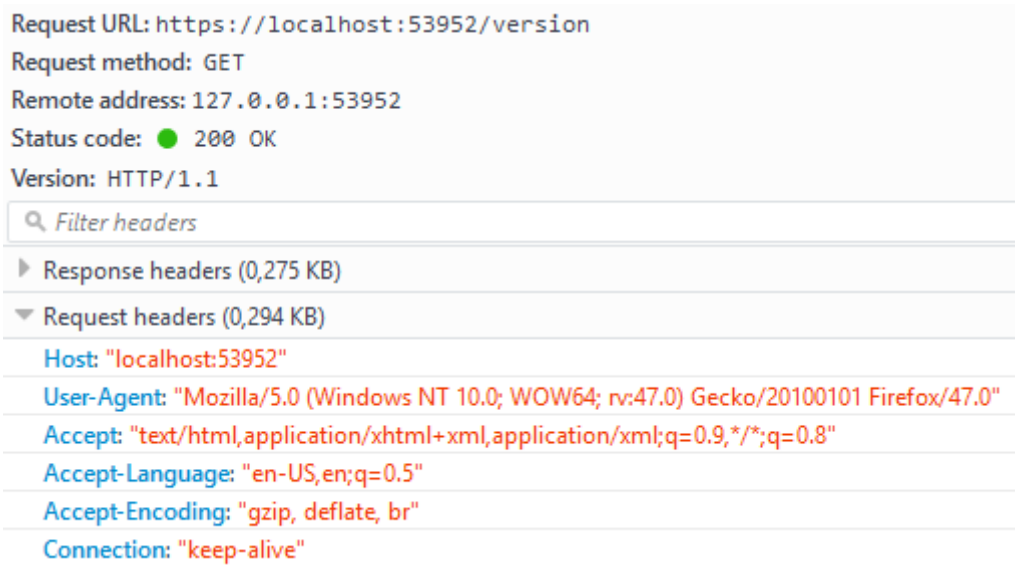
4.2.1 Запитване

HTML приложението може да изпрати запитване за версията на BISS. Запитването за версията на BISS може да бъде използвано от клиентското приложение за откриване на BISS и потвърждение, че е стартирано.

HTTP параметри:

- HTTP метод: GET
- Request-URI: /version
- Други HTTP параметри се добавят от браузъра по спецификацията на HTTP/1.1 и CORS.

Примерно запитване:



4.2.2 Отговор

BISS отговаря на HTML приложението със следните HTTP параметри:

- Status-Code: 200
- Content-Type: application/json; charset=utf-8
- Други HTTP параметри се добавят от браузъра както е по спецификацията на HTTP/1.1 и CORS.

Както и със следният JSON отговор:

- **version** (string, задължителен): версията, която BISS поддържа.
- **httpMethods** (string, задължителен): позволените HTTP методи за изпращане на заявка за подписване.
- **contentTypes** (string, задължителен): подържаните типове данни, които могат да бъдат изпратени към BISS.
- **signatureTypes** (string, задължителен): подържаните типове подписване от BISS.
- **selectorAvailable** (Boolean, задължителен): указва дали BISS подържа филтър при избора на сертификат.
- **hashAlgorithms** (string, задължителен): подържаните хеш алгоритми.

4.3 Избор на сертификат за подписване

Преди подписването, HTML приложението изпраща заявка за избор на сертификат

4.3.1 Запитване

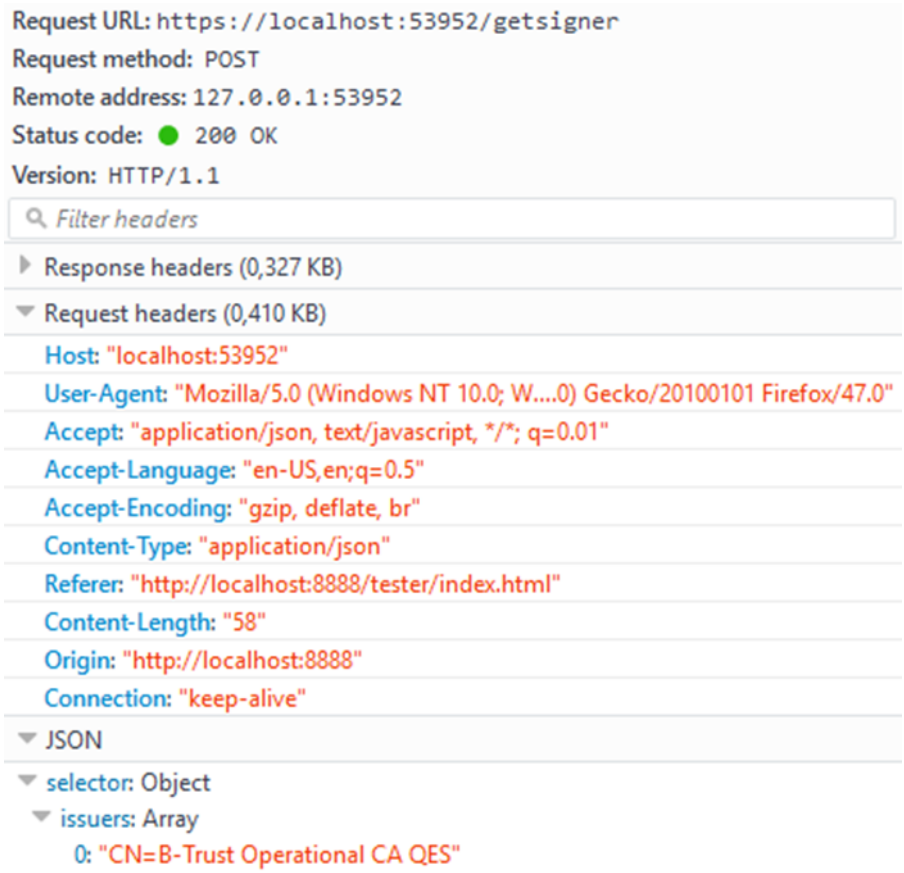
HTML параметри на заявката:

- HTTP метод: POST
- Request-URI: /getsigner
- Content-Type: application/json
- Други HTTP параметри се добавят от браузъра, както е по спецификация на HTTP/1.1 и CORS.

POST параметри на заявката:

- **Selector** (Object, по избор): съдържа параметрите за филтриране на сертификати:
  - **Issuers** (array, по избор): списък на позволени доставчици на удостоверителни услуги (ДУУ). Полето „Issuer“ задължително трябва да съответства на един от позволените ДУУ. Препоръчително е да се филтрират клиентските сертификати посредством селективния филтър „akis“ вместо “Issuers“. Големината на буквите има значение.
  - **akis** (array, по избор): списък от „authority key identifiers“ на позволените ДУУ в base64 формат. Големината на буквите има значение.
  - **keyUsages** (array, по избор): списък на позволени стойности на атрибута „key Usage“ от клиентския сертификат (“digitalSignature”, “nonRepudation”, “dataEncipherment”, “decipherOnly”, “encipherOnly”, “keyAgreement”, “keyEncipherment”, “keyCertSign”, “crlSign”). Ако има указани стойности, то клиентския сертификат трябва да ги има всички.
- **showValidCerts** (boolean, по избор): не задължителен параметър указващ дали да се показват или не невалидните сертификати. При **false** показва всички(валидни и невалидни), при **true** показва само валидните. Ако не се подаде стойността по подразбиране е **false**

Примерна заявка за избор на сертификат:



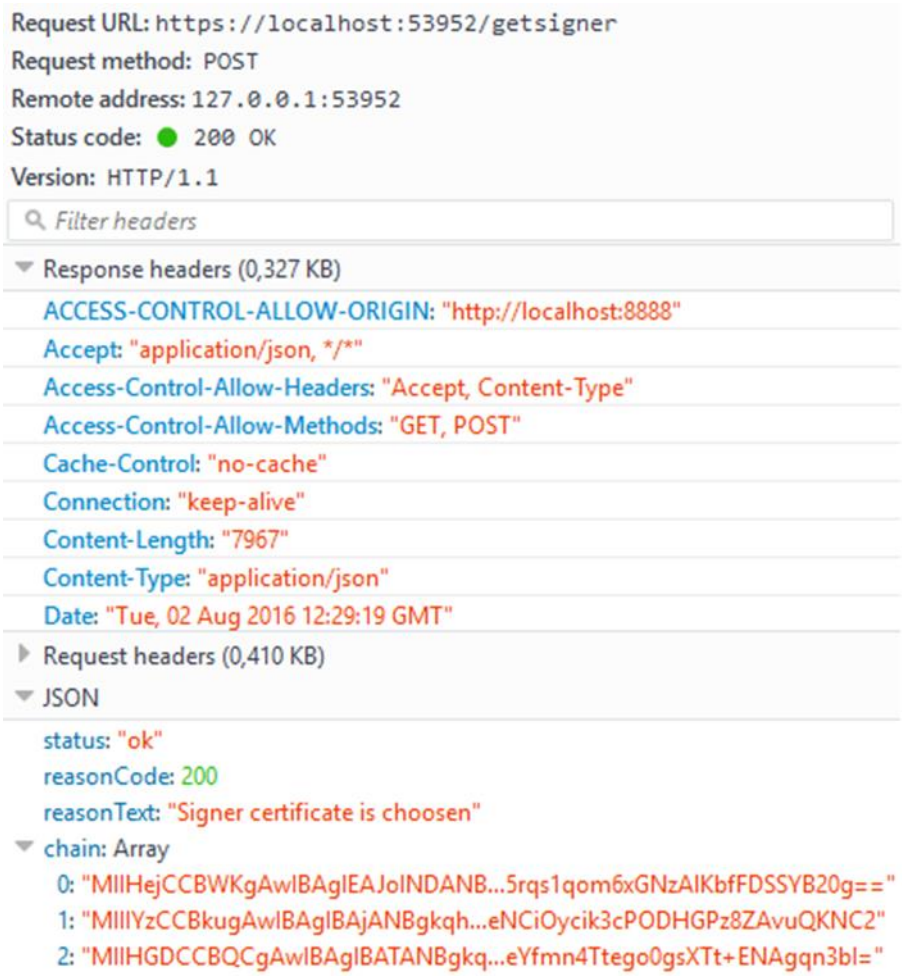
4.3.2 Отговор

BISS връща отговор със следните HTTP параметри:

- Status-code: 200
- Content-Type: application/json; charset=utf-8
- Други HTTP параметри се добавят от браузъра, както е по спецификацията на HTTP/1.1 и CORS.

Допълнителни параметри на отговора:

- **status** (string, задължителен): показва дали подписването е успешно. Поддържаните стойности са “ok” и “failed”.
- **reasonCode** (string, задължителен): съдържа кода съответстващ на статуса на отговора. Всички кодове са описани в секция 1.4.
- **reasonText** (string, задължителен): Съдържа текстово описание на статуса на отговора.
- **chain** (array, по избор): съдържа сертификационната верига на подписващия сертификат. Подписващия сертификат е на позиция 0. Всички сертификати са base64 кодирани. Наличен е ако подписването е успешно. Сертификационната верига може да не е пълна и да съдържа само подписващия сертификат.



4.4 Създаване на подпис

HTML приложението може да създава подпис чрез изпращане на заявка за подписване към BISS. В случай, че подписващият сертификат е наличен предварително, може да се извика метода за подписване без да се извиква метода /getsigner (за избор на подписващ сертификат).

4.4.1 Запитване

HTTP параметри на заявката:

- HTTP метод: POST
- Request-URI: /sign
- Content-Type: application/json
- Други HTTP параметри се добавят от браузъра, както е по спецификацията на HTTP/1.1 и CORS.

POST параметри на заявката:

- **version** (string, по избор): съдържа версията, която се подържа от HTML приложението и се очаква от BISS.
- **contents** (array, задължителен): съдържа base64 кодирани данни, които трябва да бъдат подписани.

Пример: Ако трябва да се подпише string "TestData" - Base64 кодирания string е VGVzdERhdGE=. Base64 кодирания string се подава в поле "contents": ["VGVzdERhdGE="] на заявката към BISS.

- **signedContents** (array, задължителен):. Тук се подава Base64 кодиран подпис на хешът на данните. Хешът е генериран с алгоритъм SHA-256 и се подписва с частния ключ на сървърния сертификат. Извършва се SHA256WithRSA подписване.**hashAlgorithm.** Подписът на хеша се извършва със съответния частен ключ на сървърния сертификат**.**

Пример: Ако трябва да се подпише string „TestData“, то с частния ключ на сървърния сертификат (указан в атрибут **signedContentsCert**) се подписва този **string**, в зависимост от избрания в атрибут **hashAlgorithm** алгоритъм. Тоест при избран хеш алгоритъм SHA-256 се извършва SHA256WithRSA подписване на string “TestData”:
signedContents=rsaSha256Sign("TestData")=encrypt(sha256("TestData"))
serverPrivateKey serverPrivateKey

- **signedContentsCert** (array, задължителен): съдържа base64 кодиран сертификат, който е подписал съответният елемент от атрибута **signedContents**. Този сертификат трябва да е от доверена сертификационна верига. (За MS Windows издателя на този сертификат трябва да присъства в Trusted Root Certification Authorities или Intermediate Certification Authorities). Не могат да се използват Self-Signed Certificates. Тъй като при инсталация на BISS, се инсталират сертификационните вериги на B-Trust, препоръчва се използването на сървърен сертификат (OV/DV SSL сертификат) от този доставчик. Също така за подписването може да се използва домейн сертификата на страницата, която се обръща към BISS.

- **contentType** (string): съдържа типа на данните за подписване.. Подава се стойност“data“, който показва, че параметъра “contents” съдържа информацията, която ще бъде хеширана (с указаната стойност в параметъра **hashAlgorithm**) преди да се подпише.

- **hashAlgorithm** (string, по избор): съдържа алгоритъма, който трябва да бъде използван за подписване. По подразбиране стойността е “SHA256”. Поддържаните стойности са “SHA256” и “SHA512”.

- **signatureType** (string, по избор): съдържа формата на подписване. Подава се стойност, поподразбиране стойността е “signature”.

- **signerCertificateB64** (string, задължителен): Base64 кодиран избраният сертификат за подписване.

- **confirmText** (array, незадължителен): Управлява дали да се извежда диалоговия прозорец за потвърждаване на подписване на съдържанието и какъв текст да визуализира в него
  - Ако не е подадена стойност, диалоговия прозорец за потвърждаване на подписването не се визуализира
  - Ако е подаден “hash” като първи елемент от масива се визуализира част от хеш на съдържанието, което ще бъде подписано.

Ако се подаде произволен стринг като първи елемент от масива се визуализира подадения текст.

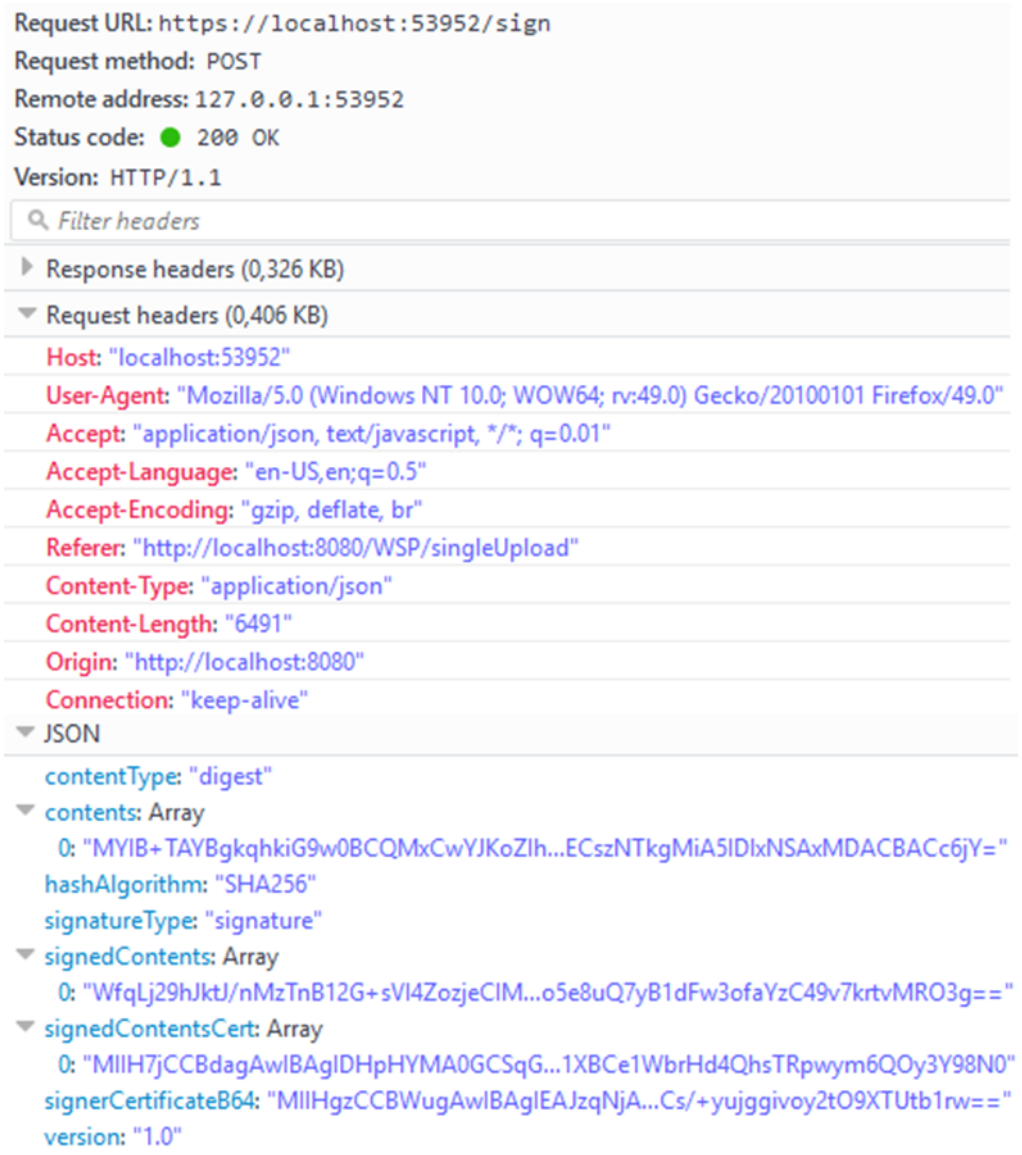
Начин за подписване на заявката с използван език Java 1.8

```
//Байтов масив на документа, който ще бъде подписан
byte[] doc = "Test".getBytes();
//Кодираме в Base64 формат и го поставяме в contents
java.lang.String docBase64 = java.util.Base64.getEncoder().encodeToString(doc);
System.out.println(docBase64);

//Вземаме хеша на документа, за подписване на заявката (актуални алгоритми са SHA-256 и SHA-512
java.security.MessageDigest md = java.security.MessageDigest.getInstance("SHA-256");
byte[] result = md.digest(doc);


//За подписване на хеша ни е нужен частният ключ по долу е показано как може да се вземе
java.lang.String keyStorePass = "xxxx";
PrivateKey privateKey = null;

// Тук поставяме пътят до .pfx хранилище за електронен подпис
java.io.FileInputStream fileInputStream = new java.io.FileInputStream("D:/Documents/SignRequest.pfx");
java.security.KeyStore ks = java.security.KeyStore.getInstance("pkcs12");
ks.load(fileInputStream, keyStorePass.toCharArray());
java.util.Enumeration<String> aliases = ks.aliases();
java.lang.String alias = null;
while (aliases.hasMoreElements()) {
    alias = aliases.nextElement();

    if (ks.isKeyEntry(alias)) {
        privateKey = (PrivateKey) ks.getKey(alias, keyStorePass.toCharArray());
        /**
         * С двата реда по долу вземаме сертификата, който е подписал заявката
         * и го поставяме в signedContentsCert параметъра на заявката за подписване
         */
        java.security.cert.X509Certificate cert = (java.security.cert.X509Certificate) ks.getCertificate(alias);
        java.lang.String certBase64 = java.util.Base64.getEncoder().encodeToString(cert.getEncoded());
        System.err.println(certBase64);
    }
}
/**
 * В тази част става подписването на заявката алоритъма в getInstance метода се задава според
 * hashAlgorithm полето от заявката (SHA256withRSA  или SHA512withRSA
 * полученият резултат от подписа се кодира в Base64 формат и се поставя в signedContents
 */
java.security.Signature signature = java.security.Signature.getInstance("SHA256withRSA");
signature.initSign(privateKey);
signature.update(result);
java.lang.String signedResult = java.util.Base64.getEncoder().encodeToString(signature.sign());
System.out.println(signedResult);
```

Начин на подписване на заявката на PHP

```php
<?php
$contentType="data";
$contentToSign="test";

#Секретният ключ, с който се подписва заявката към BISS
$pkFile="cert/srv-private.key";
#Сертификата, към подписа, с който е подписана заявката за BISS
$certFile="cert/srv-certificate.crt";
#Сертификата на клиента
$userCertFile="C:/tmp/1.cer";
$fp = fopen ( $pkFile, 'r' );
$pk = fread ( $fp, filesize ( $pkFile ) );
fclose ( $fp );


$privKey = openssl_pkey_get_private($pk);
$serverCert = implode ( '', file ( $certFile, FILE_IGNORE_NEW_LINES ) );
$userCert=implode ( '', file ( $userCertFile, FILE_IGNORE_NEW_LINES ) );


$signedHash="";
$contentToSignHash = hash ( strtolower ( "sha256" ), $contentToSign, true );
openssl_sign ( $contentToSignHash, $signedHash, $privKey, OPENSSL_ALGO_SHA256);


$strippedCert = str_replace ( array (
        '\r',
        '-----BEGIN CERTIFICATE-----',
        '-----END CERTIFICATE-----'
), '', $serverCert );
$strippedUserCert = str_replace ( array (
        '\r',
        '-----BEGIN CERTIFICATE-----',
        '-----END CERTIFICATE-----'
), '', $userCert );
$requestObj=new StdClass();
$requestObj->version="1.0";
$requestObj->signatureType="signature";
$requestObj->hashAlgorithm="SHA256";


if ($contentType=="data"){
  $requestObj->contentType="data";
  $requestObj->contents=array(base64_encode($contentToSign));
}else{
    $requestObj->contentType="digest";
  $requestObj->contents=array(base64_encode($contentToSign));
}

$requestObj->signedContents=array(base64_encode($signedHash));
$requestObj->signedContentsCert=array($strippedCert);
$requestObj->signerCertificateB64=$strippedUserCert;
$requestObj->confirmText=array("hash");

echo json_encode($requestObj);
?>
```

Примерна заявка за подписване към BISS е показана по долу:



4.4.2 Отговор

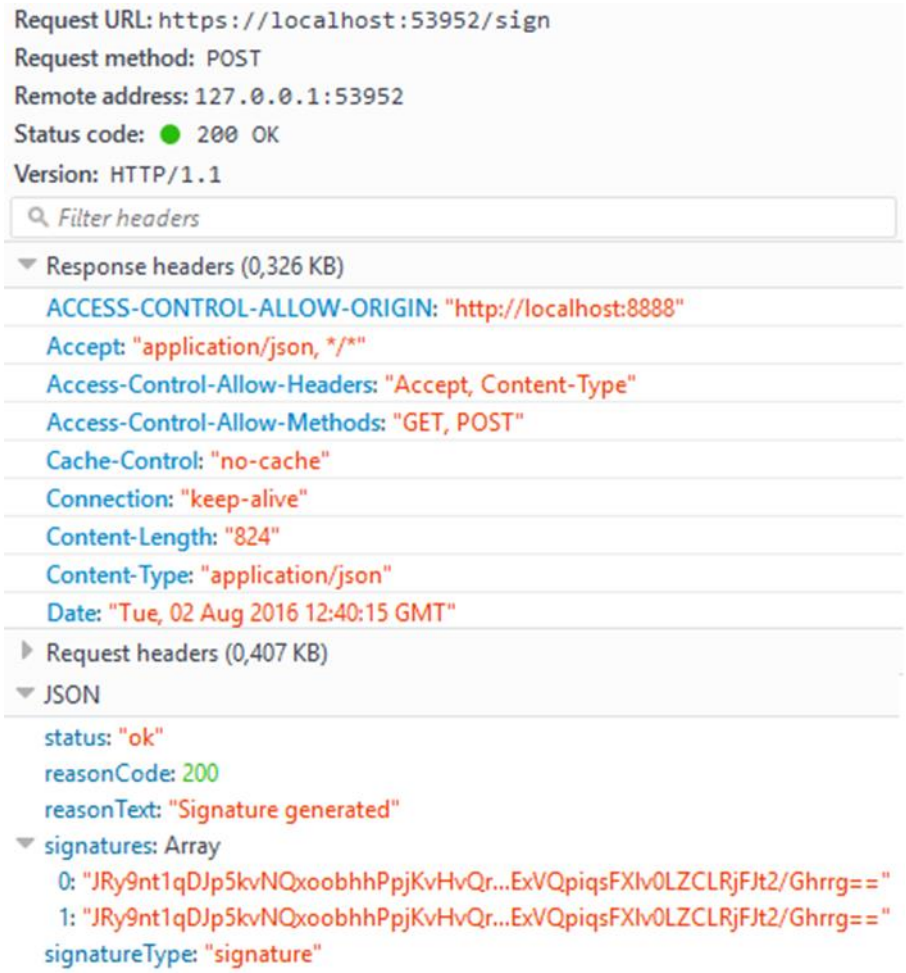
BISS връща отговор със следните HTTP параметри:

- Status-Code: 200
- Content-Type: application/json; charset=utf-8
- Други HTTP параметри се добавят от браузъра, както е по спецификацията на HTTP/1.1 и CORS.

Допълнителни параметри на отговора:

- **status** (string, задължителен): показва дали подписването е успешно. Поддържаните стойности са “ok” и “failed”.
- **reasonCode** (string, задължителен): съдържа кода съответстващ на статуса на отговора. Всички кодове са описани в секция 1.4.
- **reasonText** (string, задължителен): Съдържа текстово описание на статуса на отговора.
- **signatures** (array, по избор): съдържа base64 кодиран подпис. Наличен е ако подписването е успешно.

- **signatureType** (string, по избор): съдържа типа на подписа. Поддържаните стойности са “signature”. Наличен е ако подписването е успешно.



4.5 Статус кодове

Статус кодовете, описващи процеса на подписване, в отговора от BISS са както следва:

- 2xx – показва успешно подписване
- 4xx – показва, че подписването е прекъснало поради някаква причина
- 5xx – показва, че BISS не работи правилно или не поддържа заявената функционалност.

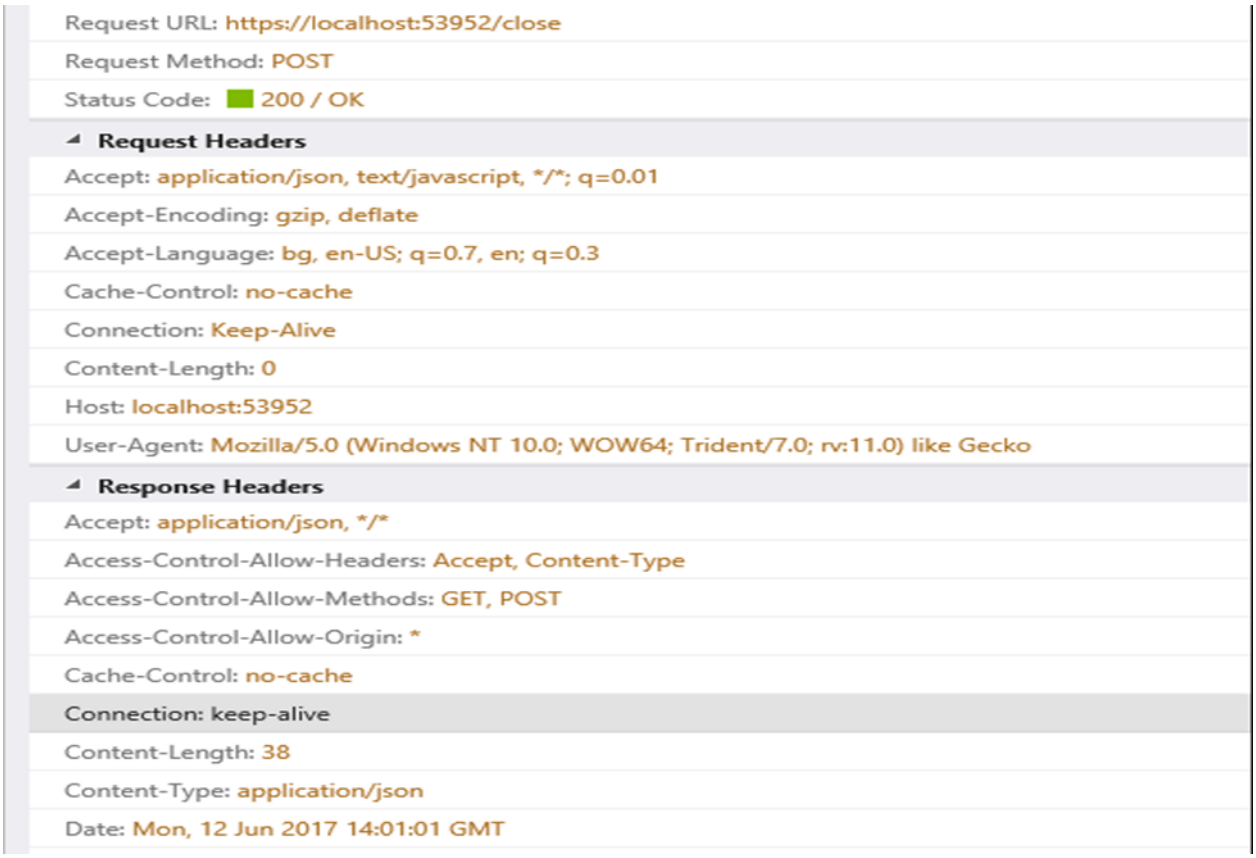
Възможните кодове са описани в следната таблица:

reasonCode	Текст	Описание
200	Успех	Подписването е успешно
400	Лоша заявка	Лоша заявка, например: • Непознат адрес (Request-URI) • Лошо форматиране на JSON

		• Липсващи задължителни параметри на заявката • Параметъра съдържа не поддържана стойност
401	Непозволен	Подписването е отказано от потребителя: • Не е избран сертификат за подписване • Потребителският сертификат няма заявените “key Usage” атрибути (грешен ПИН, Блокиран ПИН) • Потребителя няма сертификати отговарящи на филтъра в заявката
403	Забранен	Заявката да подписване е отказана, заради: • Друга заявка е в процент на обработка • Заявката за подписване е направена от непознат източник (не е използван HTTPS протокол за отваряне на HTML приложението изискващо подпис)
413	Заявката е твърде голяма	Размера на заявката е твърде голяма
441	Вътрешна грешка	Удостоверителната верига на доставчика на сертификата не е намерена.
500	Вътрешна грешка	BISS не работи правилно. Възможна причина е неправилното конфигуриране
501	Не се подържа	Заявената функционалност не се поддържа или не е налична

4.6 Заявка за спиране на приложението

Заявката се използва за спиране на приложението.



4.7 Заявка за статуса на подписваните документи от масив

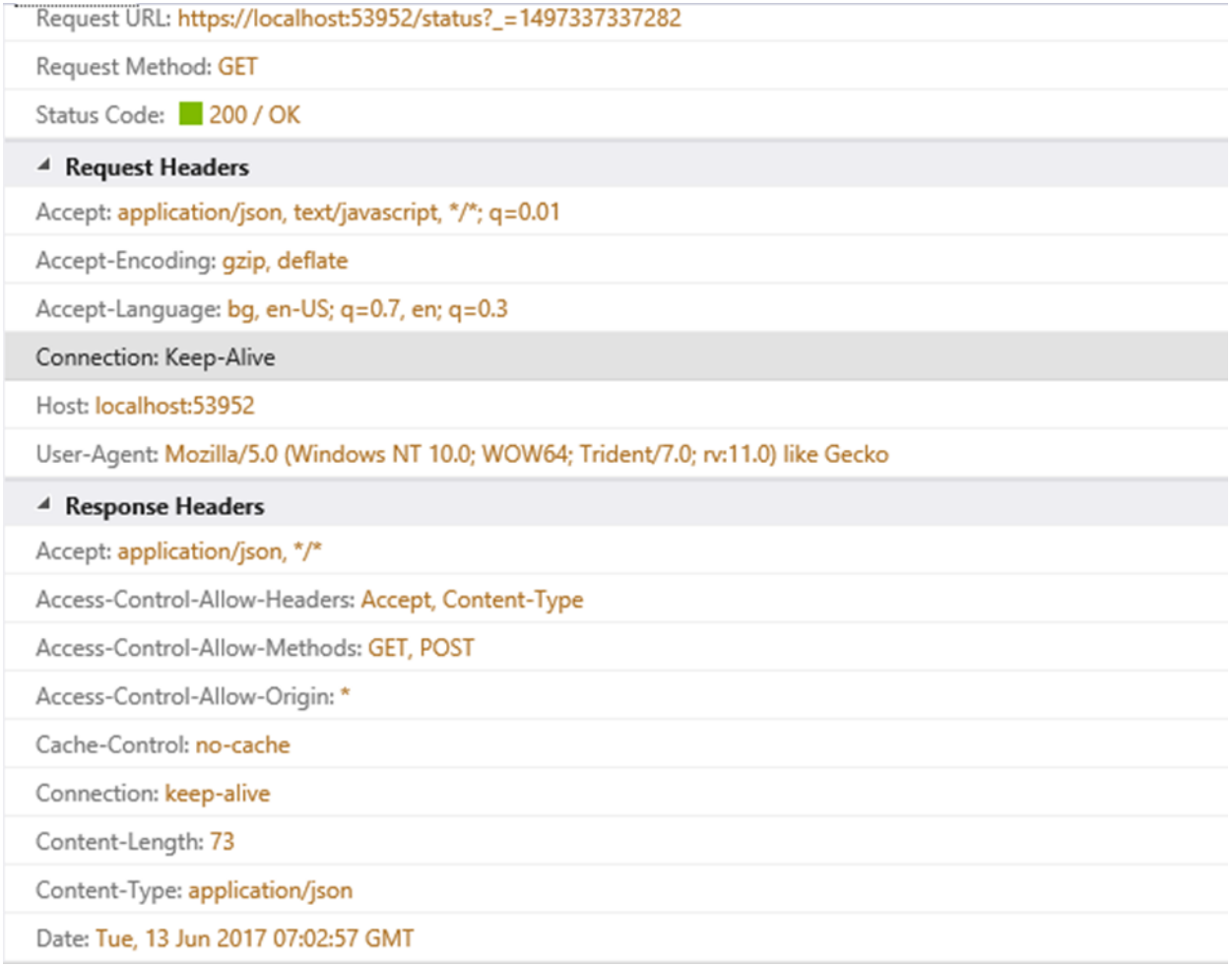
Заявката служи за следене на текущия статус на подписваните документи от масив

4.7.1 Запитване

HTTP параметри:

- HTTP метод: GET
- Request-URI: /status
- Други HTTP параметри се добавят от браузъра по спецификацията на HTTP/1.1 и CORS.

Примерно запитване:



4.7.2 Отговор

Отговорът е в JSON формат, подобен на примера показан по-долу:
{"status":100,"statusCode":"PROCESSING","totalCount":20,"currentCount":1}, където

Status	Описание	statusCode
- 100	Грешка	ERROR
0	Не е започнало изпълнение	NOT STARTED
100	В процес на изпълнение	PROCESSING
200	Обработката е приключила	DONE

- totalCount: общия брой елементи от масива за подписване
- currentCount: брой подписани до момента елементи от масива

5. Извикване на локалната услуга за електронно подписване

5.2 Javascript код за избиране на сертификат за подписване

```
function chooseSignerCertificate() {

var port = $("#port").val();

var request = {
selector: {}
};

console.log(request);

if (keyusage!=null) {
request.selector.keyusages = new Array();
request.selector.keyusages.push(keyusage);
}
if (issuer!=null) {
request.selector.issuers = new Array();
request.selector.issuers.push(issuer);
}
if (thumbprint!=null){
request.selector.thumbprint=thumbprint;
}

var payload = null;
var querystr = "";
if (method=="POST") {
payload = JSON.stringify(request);
} else {
querystr =  "?version="+request.version;
}

var url = "https://localhost:" + port +"/getsigner" + querystr;

$.ajax({
url: url,
type: method,
crossDomain: true,
data: payload,
contentType: "application/json",
dataType: "json"
})
.done(function (response) {
console.log(response);
if (response.status=="ok") {
var div = $("#result");
$("#signerCertificate").html(response.chain[0])
for (var i=0;i<response.chain.length;i++)
div.append("<p><pre>Certificate#"+i+":\n"+rowlize(response.chain[i])+"</pre
></p>");
response.content = data;
response.contentType = type;
validate(response);
} else
$("#result").html("<p><span class='label label-
danger'>"+response.reasonCode+"</span> "+response.reasonText+"</p>");
```

```
$("#sign").removeAttr("disabled");
})
.fail(function () {
console.log("failed");
$("#result").html("<p><b>Failed!</b> Most likely the SCS server is not
running or browser's configuration policy does not allow cross origin
requests.  For instance, IE 10 and 11 do not allow requests to localhost
unless the origin server is listed as trusted: see IE Settings > Internet
Options > Security tab > click Trusted sites > click Sites buttons.</p>");
$("#sign").removeAttr("disabled");
});
}
```

5.3 Javascript код за избиране на сертификат за подписване

За извикване на BISS се използват показаните по - долу JavaScript кодове.

```
function sign(datas,sigalg) {


var port = $("#port").val();

$("#sign").attr("disabled","disabled");

var request = {
signerCertificateB64: $("#signerCertificate").html(),
contents: datas,
contentType: type,
hashAlgorithm: sigalg,
signatureType: "signature",
version: "1.0"
};

console.log(request);

/*
request.selector.validate = validation;

if (keyusage!=null) {
request.selector.keyusages = new Array();
request.selector.keyusages.push(keyusage);
}
if (issuer!=null) {
request.selector.issuers = new Array();
request.selector.issuers.push(issuer);
}
if (thumbprint!=null){
request.selector.thumbprint=thumbprint;
}
*/

var payload = null;
var querystr = "";
if (method=="POST") {
payload = JSON.stringify(request);
} else {
querystr =  "?version="+request.version;
```

```
querystr += "&contentType="+escape(request.contentType);
querystr += "&hashAlgorithm="+escape(request.hashAlgorithm);
querystr += "&signatureType="+escape(request.signatureType);
querystr += "&content="+escape(request.content);
}

var url = "https://localhost:" + port +"/sign" + querystr;

$.ajax({
url: url,
type: method,
crossDomain: true,
data: payload,
contentType: "application/json",
dataType: "json"
})
.done(function (response) {
console.log(response);
if (response.status=="ok") {
var div = $("#result");
div.html("<p><b>"+response.reasonText+"</b> <font
id=\"signature_result\"><span class='label label-warning'>validating
signature on server...</span></font></p>");
for (step = 0; step < response.signatures.length; step++) {
div.append("<p><pre>Signature
("+response.signatureAlgorithm+"):\n"+rowlize(response.signatures[step])+"<
/pre></p>");
}
//for (var i=0;i<response.chain.length;i++)
//
div.append("<p><pre>Certificate#"+i+":\n"+rowlize(response.chain[i])+"</pre
></p>");
response.content = data;
response.contentType = type;
validate(response);
} else
$("#result").html("<p><span class='label label-
danger'>"+response.reasonCode+"</span> "+response.reasonText+"</p>");
$("#sign").removeAttr("disabled");
})
.fail(function () {
console.log("failed");
$("#result").html("<p><b>Failed!</b> Most likely the SCS server is not
running or browser's configuration policy does not allow cross origin
requests.  For instance, IE 10 and 11 do not allow requests to localhost
unless the origin server is listed as trusted: see IE Settings > Internet
Options > Security tab > click Trusted sites > click Sites buttons.</p>");
$("#sign").removeAttr("disabled");
});

}
```