Digital Signature Api Documentation

Toci

August 19, 2015

Contents

1 Indtroduction **2**

1.1 API Resources 2

1.2 Error handling 2

2 Signing data **3**

3 Verifying data **4**

4 Appendix **5**

1 Indtroduction

1.1 API Resources

- Resource: /api/sign
  Resource destination: This resource allows to sign in data using sent *.pfx file with private key.
- Resource: /api/verify
  Resource destination: This resource allows to verify signed data using signature and certificate with public key (*.cer).

1.2 Error handling

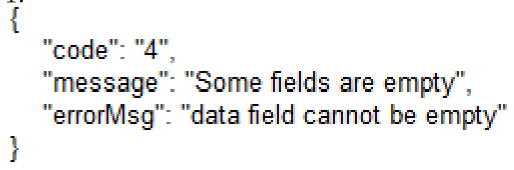
The error handling is unified in the entire API, and the error list is created bitwise. We use following error codes:

- 0 - Ok
- 4 - Missing parameters
- 32 - Wrong data
- 64 - Wrong base64 input

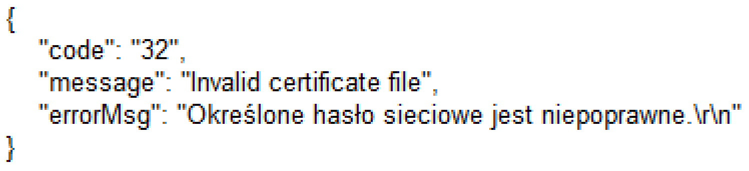
There are three types of errors:

- Some fields are empty - appears when some parameter is null
- Invalid certificate file - appears when read certificate bytes are wrong or password is incorrect(unfortunately C library exceptions doesn't split those two errors)
- Invalid base64String - when conversion from base64 to data filed

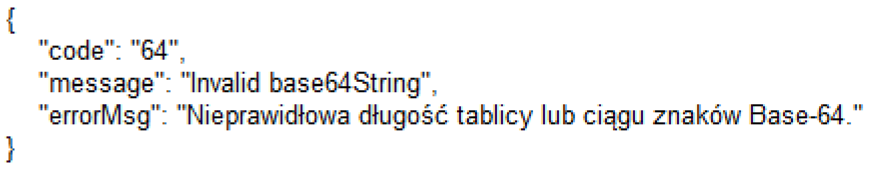
Error message format:

1.


```
{
  "code": "4",
  "message": "Some fields are empty",
  "errorMsg": "data field cannot be empty"
}
```

2.


```
{
  "code": "32",
  "message": "Invalid certificate file",
  "errorMsg": "Określone hasło sieciowe jest niepoprawne.\r\n"
}
```

3.


```
{
  "code": "64",
  "message": "Invalid base64String",
  "errorMsg": "Nieprawidłowa długość tablicy lub ciągu znaków Base-64."
}
```

2 Signing data

Example usage:

POST /api/sign

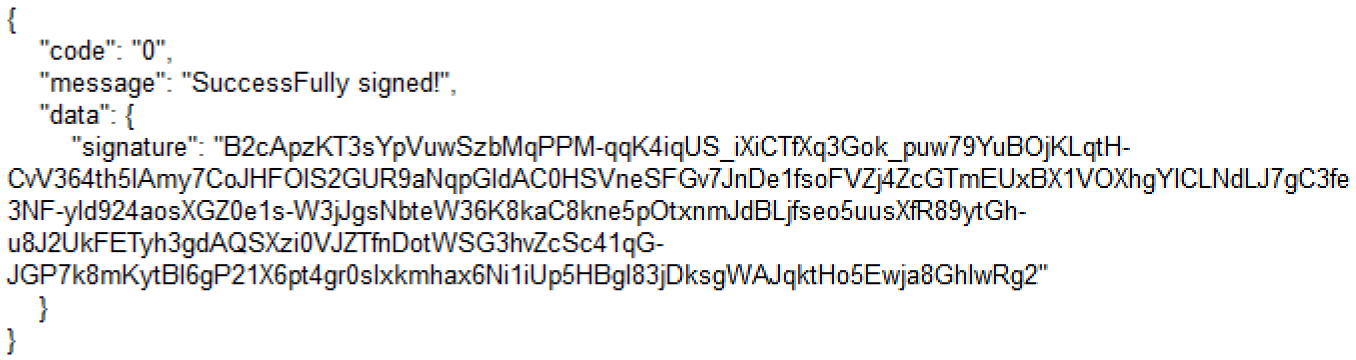
Accepted request formats:

- application/x-www-form-urlencoded
- application/json

Request body:

- data - contains base64 string with data to sign
- cert - certificate file in base64 string
- password - password to open the certificate

Response formats: JSON Example JSON Response structure:



```
{
    "code": "0",
    "message": "SuccessFully signed!",
    "data": {
        "signature": "B2cApzKT3sYpVuwSzbMqPPM-qqK4iqUS_iXiCTfXq3Gok_puw79YuBOjKLqtH-
CvV364th5lAmy7CoJHFOIS2GUR9aNqpGldAC0HSVneSFGv7JnDe1fsoFVZj4ZcGTmEUxBX1VOXhgYICLNdLJ7gC3fe
3NF-yld924aosXGZ0e1s-W3jJgsNbteW36K8kaC8kne5pOtxnmJdBLjfseo5uusXfR89ytGh-
u8J2UkFETyh3gdAQSXzi0VJZTfnDotWSG3hvZcSc41qG-
JGP7k8mKytBl6gP21X6pt4gr0slxkmhax6Ni1iUp5HBgl83jDksgWAJqktHo5Ewja8GhlwRg2"
    }
}
```

3 Verifying data

Example usage:

POST /api/verify

Accepted request formats:

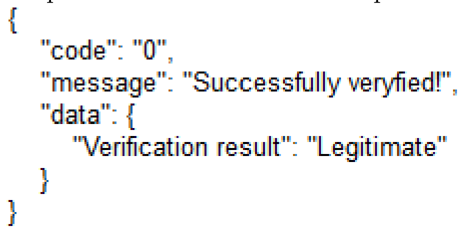
- application/x-www-form-urlencoded
- application/json

Request body:

- data - contains base64 string with data to verify
- signature - signature of data (produced by /api/sign)
- password - password to open the certificate

Response:string specifying if verification succeeded.

Response formats: JSON Example JSON Response structure:



```
{
    "code": "0",
    "message": "Successfully veryfied!",
    "data": {
        "Verification result": "Legitimate"
    }
}
```

Examples of full requests and responses are placed in Appendix.

4 Appendix

Example sign request :
Request format:

- application/x-www-form-urlencoded

  You can find the example in the attached sign-x-www-form-urlencoded.txt file.

- application/json

  You can find the example in the attached sign-json.txt file.

Example verify request :
Request format:

- application/x-www-form-urlencoded

  You can find the example in the attached verify-x-www-form-urlencoded.txt file.

- application/json

  You can find the example in the attached verify-json.txt file.