viafirma fortress

integration guide

Table of Contents

Viafirma Fortress - integration guide

This web guide is also available in PDF format:

https://doc.viafirma.com/viafirma-fortress/integration/en/documentation.pdf

Requirements

For testing Fortress services, OAuth credentials are required. Please contact fortress@viafirma.com for this purpose.

Last review: November-2024

Authenticating users / authorizing operations

End users should prove their identity to execute authentication / signature operations, by one or two factors of authentication (also called IdP: identity providers). Third-party applications are allowed to provide restrictions to this behaviour (which ones / how many of them will be used).

These factors of authentication are classified in three categories:

- Based in Knowledge (Something I Know)
- Based in Possession (Something I have)
- Based in Inherence (Something I am)

When two factors of authentication are used, factors of different categories should be used (for instance, first a knowledge factor and later a possession one).

Request authorization

During the authorization phase, end user will be redirected to a Viafirma Fortress user interface, where the user ID has to be entered, and later user should provide valid responses to one or two factors of authentication. For instance, a user can be asked a PIN and then a One-Time-Password sent to the user's mobile phone in an SMS message.

Viafirma Fortress provide a web interface for this purpose, located in the following URL:

```
{viafirma_fortress_url}/oauth2/v1/auth
```

({viafirma_fortress_url}: URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress)

The web interface will provide authentication screens for the user, using the factors of authentication recorded in the system for the specified user. This auth URL can receive some parameters which are used to prepare the authentication / authorization request:

```
{viafirma_fortress_url}/oauth2/v1/auth?
scope=profile|certificate|certificates&
state=&
redirect_uri={authorized_url_callback}&
response_type=code&
client_id={systemcode}&
user_code={usercode}
```

Param	Value	Desc
scope	profile / certificate / certificates	**profile**: used to request the authorization to access to user profile information, like name, email, etc. The following message will be shown on screen: El sistema **CRM Acme Inc.** está solicitando su autorización para: Obtener la información de su perfil **certificate**: used to request authorization to access to a specific digital certificate belonging to the end user. The following message will be printed on screen: El sistema **CRM Acme Inc.** está solicitando su autorización para: Obtener la información de uno de sus certificados **certificates**: used to request authorization to access to any of the digital certificates belonging to the end user. The following message will be printed on screen:

		
state	[string]	OPTIONAL; any value can be sent by third-party application, which will be sent back by Viafirma Fortress to the redirect_uri
redirect_uri	URL	Must be included in system_client Viafirma Fortress configuration
response_type	code	It should be `code` for webapps
client_id	[string]	third-party application code, which has been recorded in Viafirma Fortress
user_code	[string]	user code as recorded in Viafirma Fortress, for example, citizen-id, passport-id, etc.

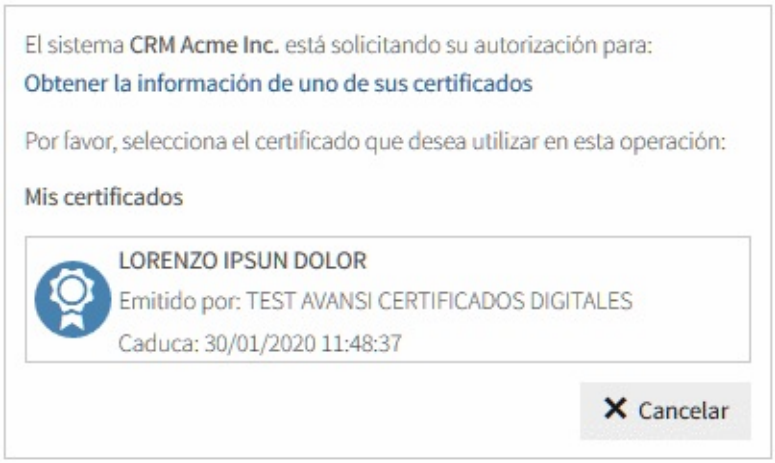
Identity providers (IDPs)

An identity provider is a factor of authentication that protects user profile and certificates information. These IDPs are enabled or disabled in Viafirma Fortress setup, and can be later associated with users. These IDPs are used by Viafirma Fortress to make end users prove their identity (for instance: entering a PIN, or password, entering an OTP code sent to an email address or a mobile phone via SMS, etc.).

In order to verify if an IDP is active, settings following this pattern: `fortress.idp.{idp_code}.active` can be checked (as included in the Installation Manual).

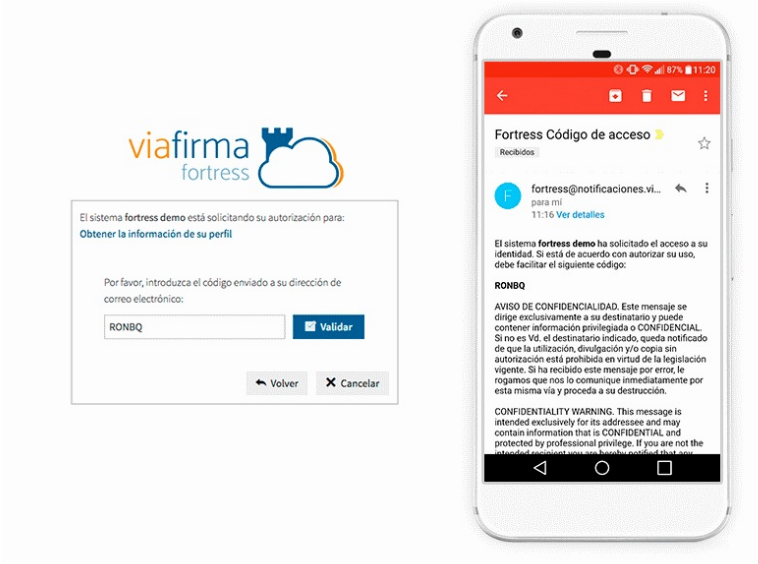
Once the user identification is successfully performed, according to the request settings (one / two factors, etc.), Viafirma Fortress considers the operation has been authorized by end user and redirects back to **authorized_url_callback**, returning control back to the third party application.

When the passed scope of authorization is **certificate** or **certificates**, after the successful identification, Viafirma Fortress will provide the end user a list of the active digital certificate/s available for selection, returning to the client application once a specific certificate has been selected by user.



Email IDP

Viafirma generates an OTP code (otp - one time password) which is sent to the user email address. User is requested to enter this code in the web interface.

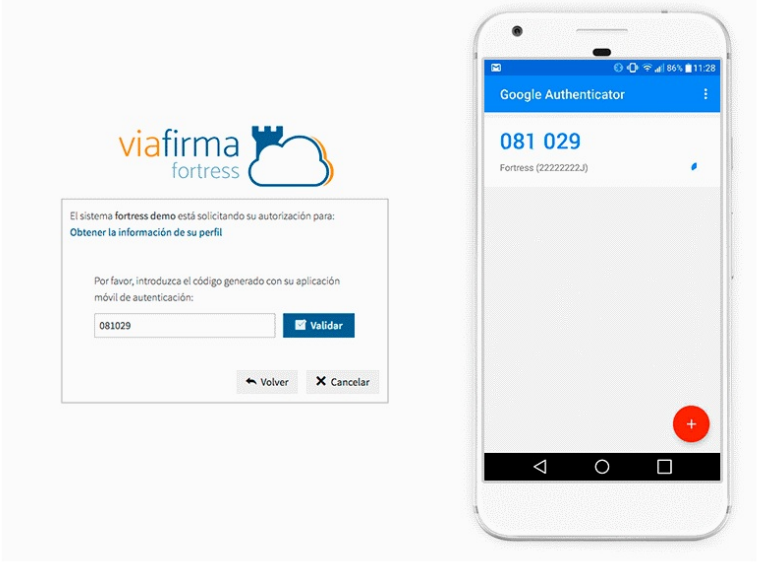


OTP SMS IDP

Viafirma generates an OTP code (otp - one time password) which is sent to the user mobile phone number via SMS. User is requested to enter this code in the web interface.

OTP (soft token) IDP

In this case, the OTP code is generated by Google Authenticator or Viafirma OTP apps, available for iOS and Android mobile devices.



LDAP IDP

User is requested to provide LDAP credentials (user/password). LDAP settings must be configured during the installation and configuration phases.

PIN IDP

Users are requested to provide the PIN (personal identification number, a positive 4-digits integer number) which was registered by them in the platform.

Password IDP

Similar to the PIN IDP, Users are requested to provide the password registered by them in the platform.

Getting the authorization/authentication process response

As explained before, once the user authorizes the operation and the identification is successfully performed, according to the request settings (one / two factors, etc.), Viafirma Fortress redirects back to **authorized_url_callback**, returning control back to the client application, including the `code` param in the redirect URL querystring:

```
{authorized_url_callback}?state=&code={authorization_code}
```

Callback sample:

```
https://example.com/response?state=&code=9a3fff39-079c-45ec-b263-7d80afb18161
```

Denied response denied or response with errors

In case the user does not authorize the request or in the event of any kind of errors, Viafirma Fortress will redirect user to **authorized_url_callback** URL including the param `error` in the querystring:

```
{authorized_url_callback}?error={error_code}&state=
```

Error callback sample:

```
http://example.com/?error=access_denied&state=
```

Getting an access token

Once the client app has received a valid authorization-code (`code`), an access token should be obtained by redeeming the authorization code:

Method: `POST`
URL: `{viafirma_fortress_url}/fortress/oauth2/v1/token`

Params:

Param	Desc
code	code returned in the authorization process (included in redirect URL)
client_id	client id (OAuth credentials).
client_secret	client secret (OAuth credentials).
redirect_uri	any authorized URL in the client application configuration
grant_type	`authorization_code` for end users authorizations and `client_credentials` for client apps authorizations.

Responses are returned using `application/json` format:

```
{
  "access_token": "1/fFAGRNJru1FTz70BzhT3Zg",
  "expires_in": 3920,
  "token_type": "Bearer",
  "user_code": "11111111H"
}
```

When **grant_type** value is `client_credentials` , no user code is returned:

```
{
  "access_token": "1/fFAGRNJru1FTz70BzhT3Zg",
  "expires_in": 3920,
  "token_type": "Bearer"
```

```
}
```

Response description:

Param	Desc
access_token	access token provided by Viafirma Fortress
expires_in	life time of access_token (in seconds). On batch signatures processes this param is null; batch procedures require `scope` = `certificate` and `signatures` = `"*"`
token_type	Type of returned token, constant value: `Bearer`
user_code	user code in Viafirma Fortress, for example, for example, citizen-id, passport-id, etc.
certificate	If `scope` = `certificate` certificate info will be returned in this field (null in other cases).

Accessing the APIs

Some API REST services related to user profile / certificates can be invoked using a valid `access_token` . Depending on the provided `scope` during the authorization phase, some API services can be called:

- SCOPE = profile
- SCOPE = certificate or certificates

for application / client methods:

- signing documents

User authentication and authorization of signature operations

The process of authentication and authorization of signature operations for a user requires the following steps:

- Client system authentication.
- Signature request
- Authentication and authorization of the request
- Execution of the signature.

Acontinuación se describen los siguientes apartados del proceso.

Client system authentication

To perform signature operations provided by Viafirma Fortress it is necessary to obtain a token associated with the client.

To do this, Viafirma Fortress offers the following Rest method, available at:

```
https://fortress.viafirma.com/fortress/oauth2/v1/token
```

This URL receives a series of parameters, which configure and prepare the signature request made by a client:

```
https://fortress.viafirma.com/fortress/oauth2/v1/token?
scope=client&
redirect_uri={url_de retorno_definido_en_viafirma_fortress}&
client_id={codigo_del_cliente_definido_en_viafirma_fortress}&
client_secret={clave_del_cliente_definido_en_viafirma_fortress}&
grant_type=client_credentials
```

Parameter	Value	Description
scope	client	For services associated with signing documents.
redirect_uri	URL	It must match one of the return URLs defined in Viafirma Fortress
client_id	Client ID defined in Viafirma Fortress	Identify the client application that made the request
client_secret	Customer key defined in Viafirma Fortress	Allows you to validate the client application that made the request
grant_type	client_credentials	Indicates that the client requests access to protected resources under his control

As a certificateRequestEntity, Viafirma Fortress will return an object in `application / json` format with the information of the access token associated with the client.

```
{
  "access_token": "1479cc2592a84cfb83c01402df613d01",
  "token_type": "Bearer",
  "expires_in": 3599
}
```

Signature request

With the client system token obtained from the previous call, the client will call the Viafirma Fortress method / signature, providing the information to be digitally signed by the user. In the next section you will find the detailed description of the signature method, as well as the parameters it receives. Once the information is processed Viafirma Fortress will return to the client system an object in `application / json` format, composed of an authorization code and an execution code

```
{
  "authCode": "124d6a9b5eaa470396a4db454780f6da",
  "exeCode": "96f1e73e5718438c8683846a2479d198"
}
```

Authentication and authorization of the request.

Once the document or the documents to be signed have been prepared, it will be necessary to authenticate the user to be able to make the signature.

As in the process of authentication and authorization in query operations, it is necessary that it be authenticated with 1 or 2 authentication factors. Depending on the configuration associated with the Viafirma Fortress client, Viafirma Fortress may request an authentication factor or, on the contrary, Fortress will force the user to authenticate against two authentication factors of different categories. The categories will be:

- Something I know -> Knowledge
- Something I have -> Possession
- Something that I am -> Inherence

To perform a user's authentication process, Viafirma Fortress offers a web interface, available at:

```
https://fortress.viafirma.com/fortress/oauth2/v1/auth
```

This URL receives a series of parameters, which configure and prepare the authentication and authorization request in the signing process:

```
https://fortress.viafirma.com/fortress/oauth2/v1/auth?
signature_code={codigo_autorización_de_la_firma}
scope=signature&
client_id={codigo_del_cliente_definido_en_viafirma_fortress}&
redirect_uri={url_de retorno_definido_en_viafirma_fortress}
```

Parameter	Value	Description
signature_code	Signature Authorization Code	Authorization code for the signature operation
scope	signature	**signature** : For services associated with the signing of documents
redirect_uri	URL	It must match one of the return URLs defined in Viafirma Fortress
client_id	Client ID defined in Viafirma Fortress	Identify the client application that made the request

Request user to sign

```
If the client did not report the user_code field associated with the user, in the `application / json` object that he used as
a parameter in the` / signature` method call, Viafirma Fortress will request the user code that wishes to make the signature.
```

When the user enters his user code in Fortress, Viafirma Fortress will validate it and show him the set of authentication factors in which the user is enrolled.

Viafirma Fortress will store the user once validated by at least one Authentication Factor in the browser's cookies, so as not to have to repeat the process each time the user tries to interact with Viafirma Fortress.

Authentication Factors

Viafirma Fortress, through the different authentication factors in which the user is enrolled, must ensure the identity of the user.

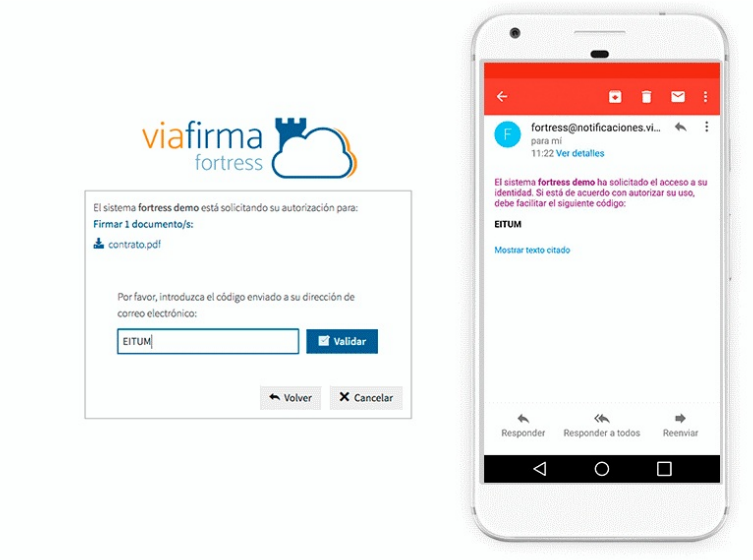
Active authentication factors can be determined during the installation of Viafirma Fortress, by modifying the values of the corresponding attributes, which follow a pattern of type `fortress.idp. {Code_of_idp} .active` (see installation manual).

During the entire document signing process, the user can see the number of documents to be signed as well as download them.

Regardless of the authentication factors selected, in case of successful authentication, it is understood that the user has authorized the operation and control will be returned to the client application, redirecting to the return URL specified in the request configuration.

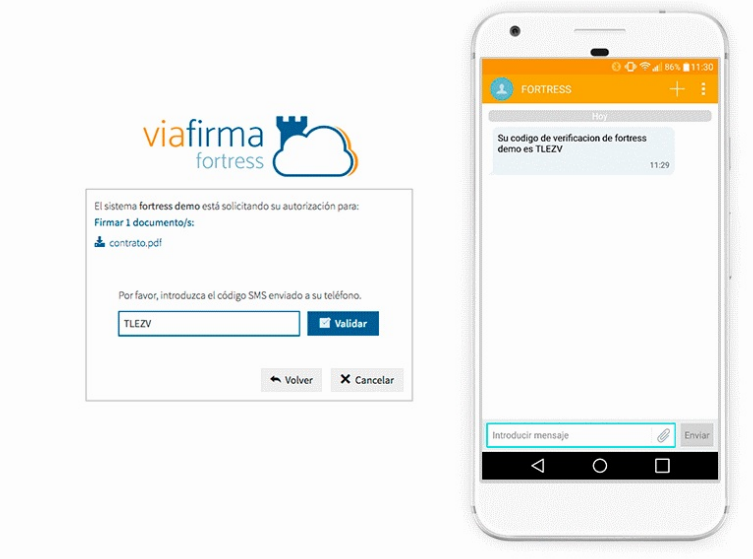
Authentication factor: Email

Aunique code is sent to the user's email, which you must enter on the authorization screen once you receive it.



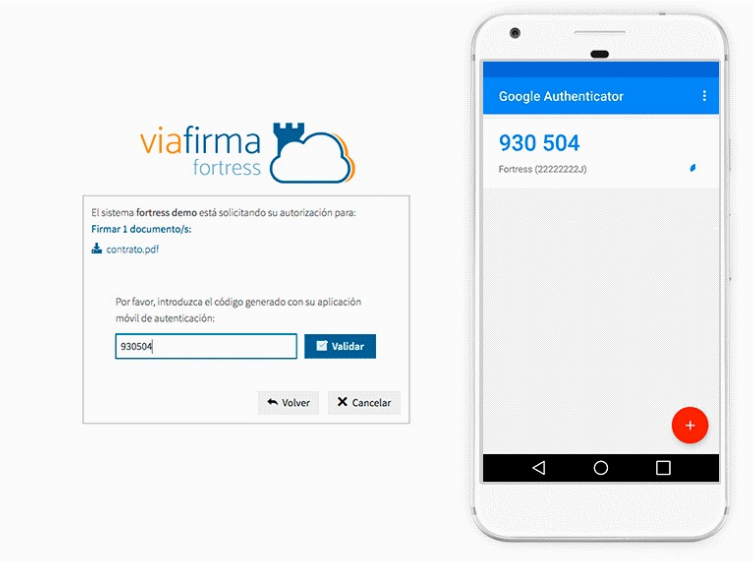
Authentication factor: SMS

An SMS with a unique code is sent to the user's mobile phone, which must be entered on the authorization screen once it is received.



Authentication factor: OTP

It is necessary to have the app (Android / IOS) that will generate a code, updated every so often. The user must enter the code in the authorization screen before the code expires.



Authentication factor: LDAP

The user's LDAP password will be requested (the configuration of the LDAP service is done during the Viafirma Fortress installation).

Authentication factor: PIN

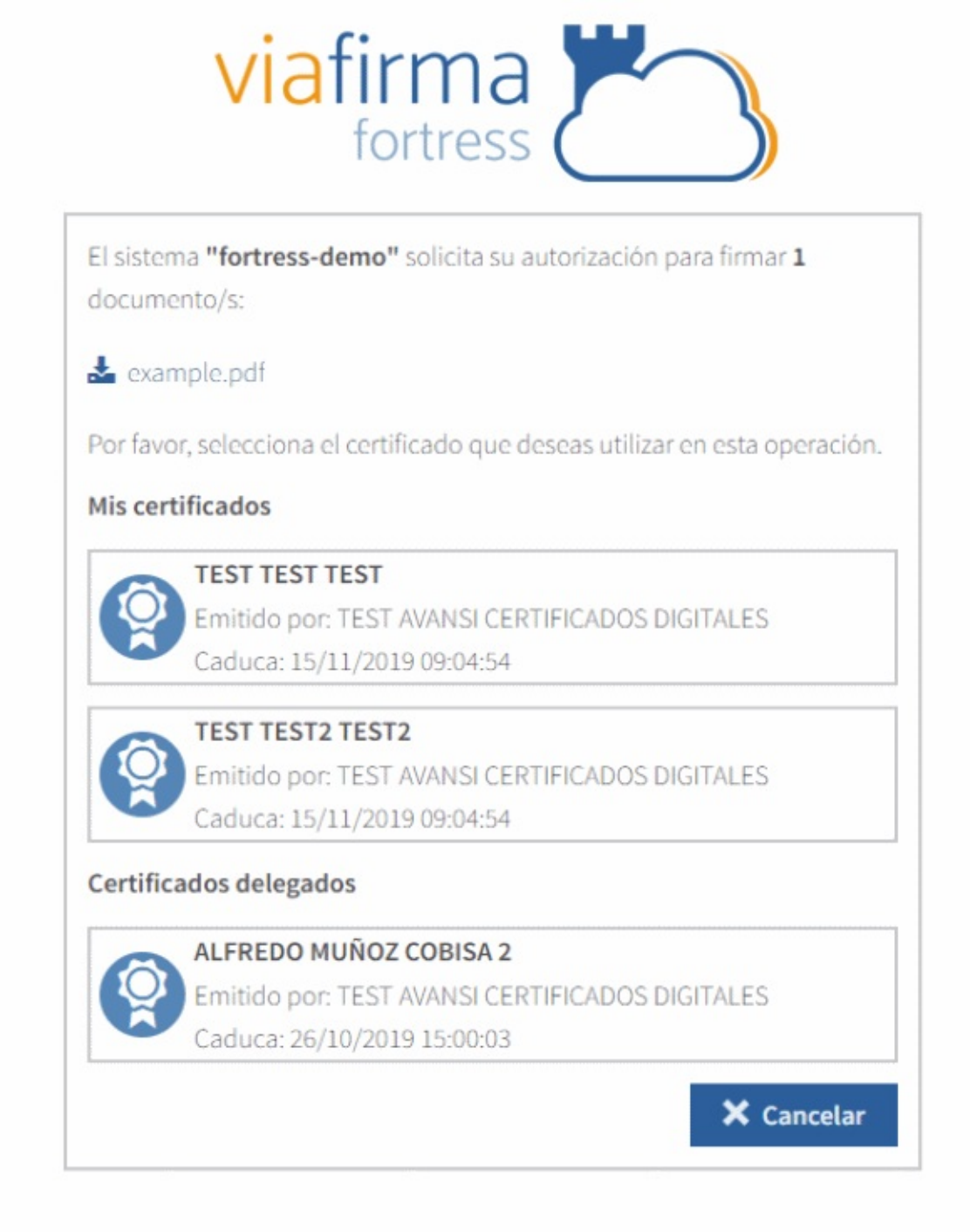
The PIN code of the user stored in Viafirma Fortress will be requested.

Authentication factor: Password

The user's password will be requested in Fortress.

Select the certificate to be used in the signature

Once the user has successfully authenticated using any of the available authentication factors, the list of delegated certificates and certificates of the user (guarded by Viafirma Fortress) will be displayed. Once the user has selected one of their certificates, control will be returned to the client application.



Execution of the signature

Finally, when the user selects a certificate, Viafirma Fortress returns the following information to the client system, to execute the signature:

- the selected certificate
- execution status
- and the date of execution

Unattended signature operations

The process to perform unattended signature operations, requires the following steps:

- Client system authentication.
- In the Viafirma Fortress backend, it is necessary to upload the certificate that will be used in the unattended signature, associated with the client system or the group.
- Signature request
- Execution of the signature.

The following sections of the process are described below.

Client system authentication

To perform signature operations provided by Viafirma Fortress it is necessary to obtain a token associated with the client.

To do this, Viafirma Fortress offers the following Rest method, available at:

```
{viafirma_fortress_url}/oauth2/v1/token
```

Where:

- `viafirma_fortress_url` : Base URL of the Viafirma Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress

This URL receives a series of parameters, which configure and prepare the signature request made by a client:

```
{viafirma_fortress_url}/oauth2/v1/token?
scope=client&
redirect_uri={url_de retorno_definido_en_viafirma_fortress}&
client_id={codigo_del_cliente_definido_en_viafirma_fortress}&
client_secret={clave_del_cliente_definido_en_viafirma_fortress}&
grant_type=client_credentials
```

Parameter	Value	Description
scope	client	For services associated with signing documents.
redirect_uri	URL	It must match one of the return URLs defined in Viafirma Fortress
client_id	Client ID defined in Viafirma Fortress	Identify the client application that made the request
client_secret	Customer key defined in Viafirma Fortress	allows you to validate the client application that made the request
grant_type	client_credentials	Indicates that the client requests access to protected resources under his control

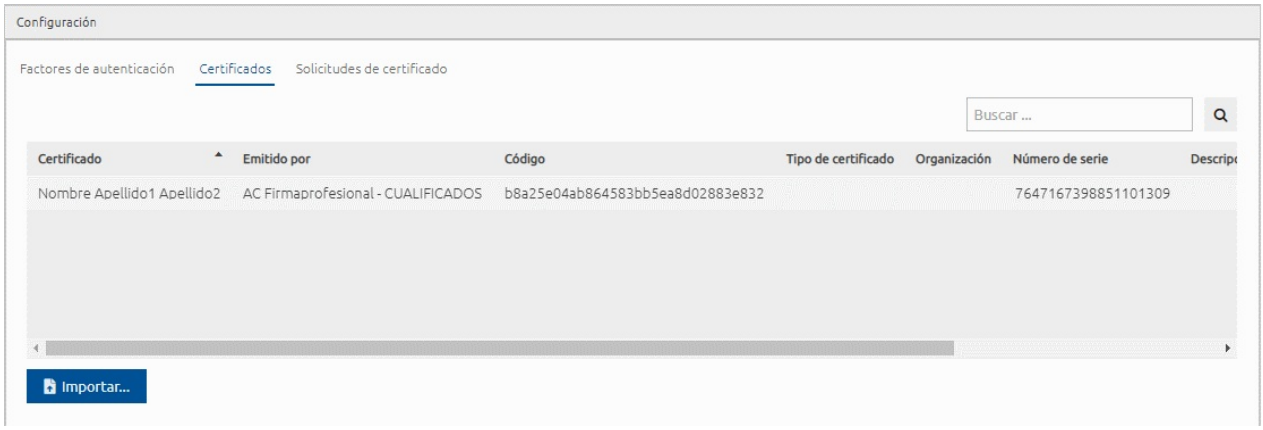
As a certificateRequestEntity, Viafirma Fortress will return an object in `application / json` format with the information of the access token associated with the client.

```
{
  "access_token": "1479cc2592a84cfb83c01402df613d01",
  "token_type": "Bearer",
  "expires_in": 3599
}
```

Host the certificate that will be used in the process, in Viafirma Fortress

Viafirma Fortress, you must manage the certificates that will be used in the unattended signature process at the Client System level or at the Group level. To manage the certificates at the client or group level, it will be necessary:

- Access the backend with a global or group administrator user
- Access the administration of your client systems or groups
- Access the detail of the client system or the group that will host the certificate used in the unattended signing process
- In the configuration section, clic on the **Certificates** tab to check the available certificates
- Press import to upload a certificate in .p12 format.
- If the platform is configured to request certificates from an embedded registration entity, you can request a new certificate.




```
Note:
The value indicated in the "Code" column is important, this value will be used in the request for an unattended signature.
```

Signature Request

With the client system token obtained from the previous call, the client will call the Viafirma Fortress method / signature, providing the information to be digitally signed unattended.

In the next section you will find the detailed description of the signature method, as well as the parameters it receives.

Once the information is processed Viafirma Fortress will return to the client system an object in `application / json` format, composed of an authorization code and an execution code:

```
{
  "authCode": "124d6a9b5eaa470396a4db454780f6da",
  "exeCode": "96f1e73e5718438c8683846a2479d198"
}
```

Execution of the signature

Finally, when the user selects a certificate, Viafirma Fortress returns the following information to the client system, to execute the signature:

- the selected certificate
- execution status
- and the date of execution

Signature extension operations

The signature extension operations process for a user requires the following steps:

- Client system authentication.
- Make the signature extension request

The following sections of the process are described below.

Client system authentication

To perform signature operations provided by Viafirma Fortress, it is necessary to obtain a token associated with the client.

To do this, Viafirma Fortress offers the following Rest method , available at:

```
{viafirma_fortress_url}/oauth2/v1/token
```

Where:

- `viafirma_fortress_url` : Base URL of the Viafirma Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress

This URL receives a series of parameters, which configure and prepare the Signature request made by a client:

```
{viafirma_fortress_url}/oauth2/v1/token?
scope = client&
redirect_uri={ url_returned_defined_in_viafirma_fortress}&
client_id={client_code_defined_in_viafirma_fortress}&
client_secret={client_key_defined_in_viafirma_fortress}&
grant_type=client_credentials
```

Parameter	Value	Description
scope	client	For services associated with document signing.
redirect_uri	URL	It must match one of the return URLs defined in Viafirma Fortress
client_id	Client ID	It is defined in Viafirma Fortress and identifies the client application that is making the request
client_secret	Client key	Allows the client application to validate that the request has been made
grant_type	client_credentials	Indicates that the client requests access to protected resources under its control

As a result, Viafirma Fortress will return an object in `application / json` format with the access token information associated with the client.

```
{
    "access_token": "1479cc2592a84cfb83c01402df613d01",
    "token_type": "Bearer",
    "expires_in": 3599
}
```

Signature extension request

With the client system token obtained from the previous call, the client will call the /extend method of Viafirma Fortress, providing it with the necessary information to extend the signature of a document previously digitally signed by the user.

In the next section you will find a detailed description of the extend method, as well as the parameters it receives.

Once the information has been processed, Viafirma Fortress will return to the client system an object in `application/json` format , composed of a reference and the base 64 bytes of the signed document.


```
{
"ref": "d8e3d98dc20e46188fd067df28048934",
"bytesB64": "MIMBKM8GCSqGSIb3DQEHAqCDASi/MIMBKLoCAQUxDzANBglghkgBZQMEAgEFADCC1QsGCSqGSIb3DQEHAaCC1PwEgtT4JVBERi0xLjMKJcTl8uXrp
..."
}
```

Viafirma Fortress API

Viafirma Fortress basically manages users (identities) information and certificates controlled by these users. The Fortress API exposes this information to third party application. To access these services, an **Access Token** is required to authorize API requests, as explained at the following link:

get Access Token

Postman Collections

If you already have access credentials to our Sanbdox environment, you can use the following postman resources to test the API. These collections include the basic use cases with which you can start your integration.

- Sandbox configuration environment

Signature Operations

- Postman Fortress Signature API Collection

Authentication operations

- Postman Fortress User Authentication API Collection

Monitoring API

Monitoring connection to Viafirma Fortress

This method we can validate the communication with Viafirma Fortress instance.

REST service specs:

Method: `GET`
URL: `{viafirma_fortress_url}/api/v1/ping`

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress

Example:

Method `GET`
URL: `{viafirma_fortress_url}/api/v1/ping`

Service response

This service will return: `200 OK` if there is communication with the Viafirma Fortress instance.

Service errors

If there is no connection to the Viafirma Fortress instance, a communication error will occur.

Error code	Error
not_found	If there is communication with the instance, but this version of Viafirma Fortress has not implemented this method.(HTTP Status: 404)

User profile API

An **Access Token** is required to authorize all API requests, as explained at the following link:

get Access Token

GET USER PROFILE

Get user profile and active certificates.

REST service specs:

Method: `GET`
URL: `{viafirma_fortress_url}/api/v1/user/{user_code}`

Security:

```
Authorization: Bearer {access_token}
```

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `user_code` : user unique identifier, for example `11111111H`

Sample Request

Method: `GET`
URL: `https://fortress.viafirma.com/fortress/api/v1/user/11111111H`

Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Sample Response

Response in `application/json` format:


```
{
  "code": "11111111H",
  "name": "JHON DOE",
  "email": "jhondoe@example.com",
  "mobile": "+34666666666",
  "lastAccess": 1501590523833,
  "role": "ROLE_USER",
  "certificates": [
    {
      "code": "226ffa94-1f0f-4c43-98aa-c7c8e4ccf657",
      "name": "Sample Certificate 01",
      "description": "Lorem ipsum dolor sit amet",
      "dateIssued": 1492432672000,
      "dateExpired": 1555504674000,
      "serialNumber": "1250978750360690486",
      "issuer": "Certificate Authority info",
      "subject": "SERIALNUMBER=11111111H, GIVENNAME=JHON, SURNAME=DOE, C=ES"
    },
    {
      "code": "014e684e-4751-4850-853c-c90802385a78",
      "name": "Sample Certificate 02",
      "description": "Lorem ipsum dolor sit amet",
      "dateIssued": 1492432671000,
```

```
      "dateExpired": 1555504674000,
      "serialNumber": "1250978750360690486",
      "issuer": "Certificate Authority info",
      "subject": "SERIALNUMBER=11111111H, GIVENNAME=JHON, SURNAME=DOE, C=ES"
    }
  ]
}
```

Where:

Param	Type	Desc
code	*string*	Usercode
name	*string*	fullname
email	*string*	email
mobile	*string*	mobile number with country prefix, Ex. +34600100200
lastAccess	*long*	datetime of last login
role	*string*	role
certificates	*array*	list of digital certificates

API Errors

Errors are returned using `application/json` format:

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	Error code
error_description	*string*	Error description

Errors:

Error code	Error
invalid_token	invalid `access_token` (HTTP Status: 401)
user_not_found	incorrect or inactive user (HTTP Status: 404)

GET USER STATUS

This service is used to retrieve information about functional operations that are allowed for a user (for instance, if user can sign, has any active digital certificate, etc.).

REST service specs:

Method: `GET`
URL: `{viafirma_fortress_url}/api/v1/user/{user_code}/status`

Security:

```
Authorization: Bearer {access_token}
```

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `user_code` : user unique identifier, for example `11111111H`

Note: a user is identified in the platform by a unique code such as id-citizen, email, passport-id, etc.

Sample Request

Method: `GET`
URL: `https://fortress.viafirma.com/fortress/api/v1/user/11111111H/status`

Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Sample response

Response in `application/json` format:

```
{
  "sign": true,
  "auth": true
}
```

Where:

Param	Type	Desc
sign	*boolean*	true if user is allowed to sign with certificate
auth	*boolean*	true if user is allowed to authenticate with certificate

API Errors

Errors are returned using `application/json` format:

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	Error code
error_description	*string*	Error description

Errors:

Error code	Error
invalid_token	invalid `access_token` (HTTP Status: 401)
user_not_found	incorrect or inactive user (HTTP Status: 404)

User and client digital certificates API

An **Access Token** is required to authorize all API requests, as explained at the following link:

get Access Token

Retrieve all certificates belonging to a user

Returns a list of active digital certificates for a specific user.

REST service specs:

Method: `GET`
URL: `{viafirma_fortress_url}/api/v1/user/{user_code}/certificate`

Security:

```
Authorization: Bearer {access_token}
```

where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `user_code` : user unique identifier, for example `11111111H`

Note: a user is identified in the platform by a unique code such as id-citizen, email, passport-id, etc.

Sample Request

Method `GET`
URL: `https://fortress.viafirma.com/fortress/api/v1/user/sample_user/certificate`

Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Sample Response

Response in `application/json` format:


```
[
  {
    "code": "226ffa94-1f0f-4c43-98aa-c7c8e4ccf657",
    "name": "Sample Certificate 01",
    "description": "Lorem ipsum dolor sit amet",
    "dateIssued": 1492432672000,
    "dateExpired": 1555504674000,
    "serialNumber": "1250978750360690486",
    "issuer": "Certificate Authority info",
    "subject": "SERIALNUMBER=11111111H, GIVENNAME=JHON, SURNAME=DOE, C=ES",
    "pem": "MIIGsTCCBZmgAwIBAgIQESeGCdXLzw9XurB4LNd0BjANBgkq..."
  },
  {
    "code": "014e684e-4751-4850-853c-c90802385a78",
    "name": "Sample Certificate 02",
    "description": "Lorem ipsum dolor sit amet",
    "dateIssued": 1492517893000,
    "dateExpired": 1555504678000,
    "serialNumber": "40963192733519241 61",
    "issuer": "Certificate Authority info",
    "subject": "SERIALNUMBER=11111111H, GIVENNAME=JHON, SURNAME=DOE, C=ES",
```

```
    "pem": "MIIFTDCCBDSgAwIBAgIIHZer06chPs4wDQYJKoZIhvcNAQEFB..."
  },
  {
    "code": "024v694e-4899-4876-863f-j91872310e70",
    "name": "Sample Certificate 03",
    "description": "Lorem ipsum dolor sit amet",
    "dateIssued": 1493432678000,
    "dateExpired": 1556504679000,
    "serialNumber": "2046339272352914110",
    "issuer": "Certificate Authority info",
    "subject": "SERIALNUMBER=11111111H, GIVENNAME=JHON, SURNAME=DOE, C=ES",
    "pem": "MIIGnTCCBYWgAwIBAgIQTuF2zDNK0C5XVqAhuNMuHjANBgkqhk..."
  }
]
```


where:

Param	Type	Desc
code	*string*	Digital certificate unique code
name	*string*	Name
description	*string*	Description
dateIssued	*string*	Date issued in milliseconds format
dateExpired	*string*	Date expired in milliseconds format
serialNumber	*string*	Serial number
issuer	*string*	Issuer (Certificate Authority)
subject	*string*	Subject
pem	*string*	Public key in PEM format

API Errors

Errors are returned using `application/json` format:

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	error description
error_description	*string*	error description

Errors:

Error code	Error
invalid_token	invalid `access_token` (HTTP Status: 401)
user_not_found	incorrect or inactive user (HTTP Status: 404)

Get information about a specific user certificate

Available only for active certificates.

REST service specs:

Method: `GET`
URL: `{viafirma_fortress_url}/api/v1/user/{user_code}/certificate/{certificate_code}`

Security:

```
Authorization: Bearer {access_token}
```

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `user_code` : user unique identifier, for example `11111111H`
- `certificate_code` : unique code of the digital certificate requested

Sample Request

Method: `GET`
URL: `https://fortress.viafirma.com/fortress/api/v1/user/sample_user/certificate/226ffa94-1f0f-4c43-98aa-c7c8e4ccf657`

Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Sample Response

Response in `application/json` format:


```
[
  {
    "code": "226ffa94-1f0f-4c43-98aa-c7c8e4ccf657",
    "name": "Sample Certificate 01",
    "description": "Lorem ipsum dolor sit amet",
    "dateIssued": 1492432672000,
    "dateExpired": 1555504674000,
    "serialNumber": "1250978750360690486",
    "issuer": "Certificate Authority info",
    "subject": "SERIALNUMBER=11111111H, GIVENNAME=JHON, SURNAME=DOE, C=ES",
    "pem": "MIIGsTCCBZmgAwIBAgIQESeGCdXLzw9XurB4LNd0BjANBgkq..."
  }
]
```


where:

Param	Type	Desc
code	*string*	Digital certificate unique code
name	*string*	Name
description	*string*	Description
dateIssued	*string*	Date issued in milliseconds format
dateExpired	*string*	Date expired in milliseconds format
serialNumber	*string*	Serial number
issuer	*string*	Issuer (Certificate Authority)
subject	*string*	Subject
pem	*string*	Public key in PEM format

API errors

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	error description
error_description	*string*	error description

Errors:

Error code	Error
invalid_token	invalid `access_token` (HTTP Status: 401)
user_not_found	incorrect or inactive user (HTTP Status: 404)
certificate_not_found	incorrect or inactive digital certificate (HTTP Status: 404)

Retrieve all certificates belonging to a system client

Returns a list of active digital certificates for a specific system client.

REST service specs:

Method: `GET`
URL: `{viafirma_fortress_url}/api/v1/client/{client_id}/certificate`

Security:

```
Authorization: Bearer {access_token}
```

where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `client_id` : System client unique identifier, for example `sample_client`

Sample Request

Method `GET`
URL: `https://fortress.viafirma.com/fortress/api/v1/client/sample_user/certificate`

Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Sample Response

Response in `application/json` format:

```
[
  {
    "code": "226ffa94-1f0f-4c43-98aa-c7c8e4ccf657",
    "name": "Sample Certificate 01",
```

```
    "description": "Lorem ipsum dolor sit amet",
    "dateIssued": 1492432672000,
    "dateExpired": 1555504674000,
    "serialNumber": "1250978750360690486",
    "issuer": "Certificate Authority info",
    "subject": "SERIALNUMBER=11111111H, GIVENNAME=JHON, SURNAME=DOE, C=ES",
    "pem": "MIIGsTCCBZmgAwIBAgIQESeGCdXLzw9XurB4LNd0BjANBgkq...",
    "delegated": false,
    "level": "MEDIUM"
  },
  {
    "code": "014e684e-4751-4850-853c-c90802385a78",
    "name": "Sample Certificate 02",
    "description": "Lorem ipsum dolor sit amet",
    "dateIssued": 1492517893000,
    "dateExpired": 1555504678000,
    "serialNumber": "40963192733519241 61",
    "issuer": "Certificate Authority info",
    "subject": "SERIALNUMBER=11111111H, GIVENNAME=JHON, SURNAME=DOE, C=ES",
    "pem": "MIIFTDCCBDSgAwIBAgIIHZer06chPs4wDQYJKoZIhvcNAQEFB...",
    "delegated": false,
    "level": "MEDIUM"
  },
  {
    "code": "024v694e-4899-4876-863f-j91872310e70",
    "name": "Sample Certificate 03",
    "description": "Lorem ipsum dolor sit amet",
    "dateIssued": 1493432678000,
    "dateExpired": 1556504679000,
    "serialNumber": "20463392723529141 10",
    "issuer": "Certificate Authority info",
    "subject": "SERIALNUMBER=11111111H, GIVENNAME=JHON, SURNAME=DOE, C=ES",
    "pem": "MIIGnTCCBYWgAwIBAgIQTuF2zDNK0C5XVqAhuNMuHjANBgkqhk...",
    "delegated": false,
    "level": "MEDIUM"
  }
]
```


where:

Param	Type	Desc
code	*string*	Digital certificate unique code
name	*string*	Name
description	*string*	Description
dateIssued	*string*	Date issued in milliseconds format
dateExpired	*string*	Date expired in milliseconds format
serialNumber	*string*	Serial number
issuer	*string*	Issuer (Certificate Authority)
subject	*string*	Subject
issuerMap	*object*	Issuer attributes
subjectMap	*object*	Subject attributes
delegated	*boolean*	Delegated certificate indicator
pem	*string*	Public key in PEM format
level	*string*	Certificate protection level

API Errors

Errors are returned using `application/json` format:

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	error description
error_description	*string*	error description

Errors:

Error code	Error
invalid_token	invalid `access_token` (HTTP Status: 401)
client_not_found	incorrect or inactive client (HTTP Status: 404)

Get information about a specific client certificate

Available only for active certificates.

REST service specs:

Method: `GET`
URL: `{viafirma_fortress_url}/api/v1/client/{client_id}/certificate/{certificate_code}`

Security:

```
Authorization: Bearer {access_token}
```

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `client_id` : client unique identifier
- `certificate_code` : unique code of the digital certificate requested

Sample Request

Method: `GET`
URL: `https://fortress.viafirma.com/fortress/api/v1/client/sample_client/certificate/226ffa94-1f0f-4c43-98aa-c7c8e4ccf657`
Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Sample Response

Response in `application/json` format:

```
{
  "code": "08d87ff2ed124a8bb7b323cbfb889e9e",
  "dateIssued": 1555495728000,
  "dateExpired": 1618567728000,
  "serialNumber": "228897951488527728794",
  "issuer": "C=DO, L=WWW.AVANSI.COM.DO, O=AVANSI S.R.L. - RNC 130222509, CN=TESTAVANSI CERTIFICADOS DIGITALES ",
```

```
  "subject": "OID.1.3.6.1.4.1.27395.8.1=CERTIFICADO DE PERSONA INDIVIDUAL, CN=LUCAS MORA PRIETO, SERIALNUMBER = 94967442 M, GI
VENNAME = LUCAS, SURNAME = MORA PRIETO, C = DO ",
  "issuerMap": {
    "C": "DO",
    "CN": "TEST AVANSI CERTIFICADOS DIGITALES",
    "L": "WWW.AVANSI.COM.DO",
    "O": "AVANSI S.R.L. - RNC 130222509"
  },
  "subjectMap": {
    "SURNAME": "MORA PRIETO",
    "C": "DO",
    "SERIALNUMBER": "94967442M",
    "1.3.6.1.4.1.27395.8.1": "CERTIFICADO DE PERSONA INDIVIDUAL",
    "CN": "LUCAS MORA PRIETO",
    "GIVENNAME": "LUCAS"
  },
  "pem": "MIIFWjCCBEKgAwIBAgI...",
  "delegated": false,
  "level": "MEDIUM"
}
```


where:

Param	Type	Desc
code	*string*	Digital certificate unique code
name	*string*	Name
description	*string*	Description
dateIssued	*string*	Date issued in milliseconds format
dateExpired	*string*	Date expired in milliseconds format
serialNumber	*string*	Serial number
issuer	*string*	Issuer (Certificate Authority)
subject	*string*	Subject
issuerMap	*object*	Issuer attributes
subjectMap	*object*	Subject attributes
delegated	*boolean*	Delegated certificate indicator
pem	*string*	Public key in PEM format
level	*string*	Certificate protection level

API errors

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	error description
error_description	*string*	error description

Errors:

Error code	Error
invalid_token	invalid `access_token` (HTTP Status: 401)
client_not_found	incorrect or inactive client (HTTP Status: 404)
certificate_not_found	incorrect or inactive digital certificate (HTTP Status: 404)

Signing of new client certificates

This service allows registering a new certificate and associating it with a client system.

REST service specs:

Method: `POST`
URL: `{viafirma_fortress_url}/api/v1/client/{client_id}/certificate

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `client_id` : client unique identifier

Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Example:

Method `POST`
URL: `https://fortress.viafirma.com/fortress/api/v1/client/sample_client/certificate` Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Service parameters

This service receives by parameters the configuration of the certificate to be signed:

The parameters that are received (in `application / json` format) have the following form:

```
{
  "keystore": "MIIZXwIBAzCCGRgGCSq...",
  "password": "123456"
}
```

Where:

Parameter	Type	Description
code	*string*	[OPTIONAL] Code to associate the certificate, if not reported Fortress generates one
description	*string*	[OPTIONAL] Description associated with the certificate
keystore	*string*	Content of keystore in PKCS#12 format encoded in Base64
password	*string*	Password of the keystore
alias	*string*	[OPTIONAL] Alias of the certificate within the keystore, only required if the keystore stores more than one certificate

Service response

The response of this service will be given (in `application / json` format) with the certificate data in the same format as the query service of a certificate of a client system.

```
{
  "code": "08d87ff2ed124a8bb7b323cbfb889e9e",
  "dateIssued": 1555495728000,
  "dateExpired": 1618567728000,
  "serialNumber": "228897951488527728794",
  "issuer": "C=DO, L=WWW.AVANSI.COM.DO, O=AVANSI S.R.L. - RNC 130222509, CN=TESTAVANSI CERTIFICADOS DIGITALES ",
  "subject": "OID.1.3.6.1.4.1.27395.8.1=CERTIFICADO DE PERSONA INDIVIDUAL, CN=LUCAS MORA PRIETO, SERIALNUMBER = 94967442 M, GI
VENNAME = LUCAS, SURNAME = MORA PRIETO, C = DO ",
  "issuerMap": {
    "C": "DO",
    "CN": "TEST AVANSI CERTIFICADOS DIGITALES",
    "L": "WWW.AVANSI.COM.DO",
    "O": "AVANSI S.R.L. - RNC 130222509"
  },
  "subjectMap": {
    "SURNAME": "MORA PRIETO",
    "C": "DO",
    "SERIALNUMBER": "94967442M",
    "1.3.6.1.4.1.27395.8.1": "CERTIFICADO DE PERSONA INDIVIDUAL",
    "CN": "LUCAS MORA PRIETO",
    "GIVENNAME": "LUCAS"
  },
  "pem": "MIIFWjCCBEKgAwIBAgI...",
  "delegated": false,
  "level": "MEDIUM"
}
```


where:

Param	Type	Desc
code	*string*	Digital certificate unique code
name	*string*	Name
description	*string*	Description
dateIssued	*string*	Date issued in milliseconds format
dateExpired	*string*	Date expired in milliseconds format
serialNumber	*string*	Serial number
issuer	*string*	Issuer (Certificate Authority)
subject	*string*	Subject
issuerMap	*object*	Issuer attributes
subjectMap	*object*	Subject attributes
delegated	*boolean*	Delegated certificate indicator
pem	*string*	Public key in PEM format
level	*string*	Certificate protection level

API errors

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc

error	*string*	error description
error_description	*string*	error description

Errors:

Error code	Error
invalid_token	invalid `access_token` (HTTP Status: 401)
client_not_found	incorrect or inactive client (HTTP Status: 404)
invalid_keystore	The keystore is not in PKCS#12 format or the password is incorrect (HTTP Status: 404)
invalid_alias	The certificate with the specified alias was not found within the keystore, or there are several certificates and the alias has not been specified (HTTP Status: 404)
certificate_already_exists	The certificate is already associated with the client system (HTTP Status: 404)
expired_certificate	The certificate has expired (HTTP Status: 404)
revoked_certificate	The certificate is revoked (HTTP Status: 404)
not_trusted_certificate	Some of the certificates in the chain can not be found in the trust store (HTTP Status: 404)
certificate_validation	An error occurred while validating the certificate (HTTP Status: 404)

Deleting client certificates

This service allows you to eliminate certificates associated with a client system.

REST service specs:

Method: `DELETE`
URL: `{viafirma_fortress_url}/api/v1/client/{client_id}/certificate/{certificate_code}

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `client_id` : client unique identifier
- `certificate_code` : unique code of the digital certificate requested
  Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Example:

Method `DELETE`
URL: `https://fortress.viafirma.com/fortress/api/v1/client/sample_client/certificate/08d87ff2ed124a8bb7b323cbfb889e9e Security Header:` Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Service response

The response of this service will be given (in `application / json` format) with the certificate data in the same format as the query service of a certificate of a client system.

```
{
  "code": "08d87ff2ed124a8bb7b323cbfb889e9e",
  "dateIssued": 1555495728000,
  "dateExpired": 1618567728000,
  "serialNumber": "228897951488527728794",
  "issuer": "C=DO, L=WWW.AVANSI.COM.DO, O=AVANSI S.R.L. - RNC 130222509, CN=TESTAVANSI CERTIFICADOS DIGITALES ",
```

```
  "subject": "OID.1.3.6.1.4.1.27395.8.1=CERTIFICADO DE PERSONA INDIVIDUAL, CN=LUCAS MORA PRIETO, SERIALNUMBER = 94967442 M, GI
VENNAME = LUCAS, SURNAME = MORA PRIETO, C = DO ",
  "issuerMap": {
    "C": "DO",
    "CN": "TEST AVANSI CERTIFICADOS DIGITALES",
    "L": "WWW.AVANSI.COM.DO",
    "O": "AVANSI S.R.L. - RNC 130222509"
  },
  "subjectMap": {
    "SURNAME": "MORA PRIETO",
    "C": "DO",
    "SERIALNUMBER": "94967442M",
    "1.3.6.1.4.1.27395.8.1": "CERTIFICADO DE PERSONA INDIVIDUAL",
    "CN": "LUCAS MORA PRIETO",
    "GIVENNAME": "LUCAS"
  },
  "pem": "MIIFWjCCBEKgAwIBAgI...",
  "delegated": false,
  "level": "MEDIUM"
}
```


where:

Param	Type	Desc
code	*string*	Digital certificate unique code
name	*string*	Name
description	*string*	Description
dateIssued	*string*	Date issued in milliseconds format
dateExpired	*string*	Date expired in milliseconds format
serialNumber	*string*	Serial number
issuer	*string*	Issuer (Certificate Authority)
subject	*string*	Subject
issuerMap	*object*	Issuer attributes
subjectMap	*object*	Subject attributes
delegated	*boolean*	Delegated certificate indicator
pem	*string*	Public key in PEM format
level	*string*	Certificate protection level

API errors

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	error description
error_description	*string*	error description

Errors:

Error code	Error
invalid_token	invalid `access_token` (HTTP Status: 401)
client_not_found	incorrect or inactive client (HTTP Status: 404)
certificate_not_found	incorrect or inactive digital certificate (HTTP Status: 404)

Signing API

An **Access Token** is required to authorize all API requests, as explained at the following link:

get Access Token

DIGITAL SIGNATURE REQUEST

REST service specs:

Method: `POST`
URL: `{viafirma_fortress_url}/api/v1/signature`

Security:

```
Authorization: Bearer {access_token}
```

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress

Sample Request

Method: `POST`
URL: `{viafirma_fortress_url}/api/v1/signature`

Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Request Params

The request body contains information such as signature format, document to be signed, etc.

`application/json` format is used:

```
{
  "signatureConfigurations": [
    {
      "ref": "#1",
      "document": {
        "bytesB64": "JVBERi0xLjMKJcTl8uXrp/Og0MTGCjQ...",
        "name": "contract.pdf"
      },
      "signatureType": "PADES_LTA",
      "signatureAlgorithm": "RSA_SHA256",
      "packaging": "ENVELOPED",
      "reason": " test PAdES signature ",
      "padesConfiguration": {
        "stamper": {
          "csvPath": "http://<someURL>/v#",
          "logoB64": "iVBORw0KGgoAAAANSUhEUgAAAWYAAABsCAYAAABZyhj...",
          "page": 1,
          "type": "QR_BARCODE128",
          "xAxis": 80,
          "yAxis": 700
        }
      }
    }
  ]
}
```

Note: params for `padesConfiguration`, `xadesConfiguration`, `tsa` and `policy` are described later.

Where:

Param	Type	Desc
userCode	*string*	OPTIONAL, used to to specify the signer user. If null, user will be requested to authenticate before signing.
certificateCode	*string*	OPTIONAL, used to specify which certificate will be used to sign. If null, user will be requested to select any of the active certificates belonging to the user.
certificatePassword	*string*	OPTIONAL, used to specify which certificate password will be used to sign, this field is only allowed for the unattended signature.
multifactorAuth	*boolean*	OPTIONAL, if true , forces the use of 2 authentication factors.
async	*string*	Set value to `true` for asynchronous execution.
callbackUrl	*string*	Fortress will do a POST with the final status to the specified URL after a asynchronous execution.
certificateFilter	*string*	Attributes by which to filter the certificate. Enter which filters you want the certificate to meet to be used.
signatureConfigurations/document/name	*string*	Name of the document to be signed
signatureConfigurations/document/bytesB64	*string*	Document to be signed (Base64)
signatureConfigurations/document/url	*string*	URL of document to be signed
signatureConfigurations/signatureType	*string*	Signature format: - `CADES_B` - `CADES_T` - `CADES_LT` - `CADES_LTA` - `PADES_B` - `PADES_T` - `PADES_LT` - `PADES_LTA` - `XADES_B` - `XADES_T` - `XADES_LT` - `XADES_LTA` - `PKCS1`
signatureConfigurations/signatureAlgorithm	*string*	signature algorithm: - `RSA_SHA1` - `RSA_SHA224` - `RSA_SHA256` - `RSA_SHA384` - `RSA_SHA512`
signatureConfigurations/packaging	*string*	signature type: - `ENVELOPED` - `ENVELOPING` - `DETACHED`
signatureConfigurations/reason	*string*	OPTIONAL, signature reason
signatureConfigurations/location	*string*	OPTIONAL, signature location
signatureConfigurations/ref	*string*	OPTIONAL, if present, his value will be returned in the signature certificateRequestEntity.

Si se informa una URL y la firma se realiza de forma asíncrona, al finalizar la firma Fortress realiza una petición POST a dicha URL con el estado final de ejecución. |

Configuración de los filtros de certificados

Esta configuración hace que a la hora de firmar, el usuario solo pueda firmar con los certificados que cumplan todos los requisitos.

```
{
  "certificateFilter": {
    "issuer.cn": [
      "AC FNMT Usuarios"
    ],
    "subject.cn": [
      "ZAMORANO DE EJEMPLO JOSE LUIS - 71121212M"
    ],
    "subject.serialnumber": [
        "99999999R",
        "71121212M"
    ],
    "oid": [
      "2.5.29.14",
      "2.5.29.15"
    ]
  }
}
```

Podemos filtrar de varias formas.

Parámetro	Tipo	Descripción
oid	*List - String*	List of oids that the certificate must have to be valid
serialnumber	*List - String*	List of serialnumber that the certificate must have to be valid
issuer.C	*Single list - String*	(Only one is allowed, starting with version 6.2.5 of Viafirma Fortress, this restriction is removed) CountryName -> ES
issuer.OU	*Single list - String*	(Only one is allowed, starting with version 6.2.5 of Viafirma Fortress, this restriction is removed) OrganizationalUnit -> Ceres
issuer.CN	*Single list - String*	(Only one is allowed, starting with version 6.2.5 of Viafirma Fortress, this restriction is removed) CommonName -> AC FNMT Usuarios
issuer.O	*Single list - String*	(Only one is allowed, starting with version 6.2.5 of Viafirma Fortress, this restriction is removed) Organization -> FNMT-RCM
subject.SURNAME	*Single list - String*	(Only one is allowed, starting with version 6.2.5 of Viafirma Fortress, this restriction is removed) Surname -> ZAMORANO DE EJEMPLO
subject.C	*Single list - String*	(Only one is allowed, starting with version 6.2.5 of Viafirma Fortress, this restriction is removed) CountryName -> ES
subject.SERIALNUMBER	*Single list - String*	(Only one is allowed, starting with version 6.2.5 of Viafirma Fortress, this restriction is removed) Serial number -> IDCES-71121212M
subject.CN	*Single list - String*	(Only one is allowed, starting with version 6.2.5 of Viafirma Fortress, this restriction is removed) CommonName -> ZAMORANO DE EJEMPLO JOSE LUIS - 71121212M

subject.GIVENNAME	*Single list - String*	(Only one is allowed, starting with version 6.2.5 of Viafirma Fortress, this restriction is removed) Givenname -> JOSE LUIS

PAdES Configuration

Params only applicable to `signatureType` PAdES (PAdES B, PAdES T, PAdES LT, PAdES LTA).

```
"padesConfiguration": {
    "stamper": { }
}
```

The stamper object is optional, and it defines a visual stamp associated with the signature PAdES.

```
{
  "stamper": {
    "csvPath": "https://sandbox.viafirma.com/fortress/v#",
    "imageB64": "JVBERi0xLjMKJcTl8uXlRU9GC...",
    "logoB64": "JVBERi0xLjMKJcTl8uXlRU9GC...",
    "page": 1,
    "rotation": "ROTATE_90",
    "textLine1": "Sample line 1",
    "textLine2": "Sample line 2",
    "textLine3": "Sample line 3",
    "type": "QR_BARCODE128",
    "xAxis": 100,
    "yAxis": 100
  }
}
```

Param	Type	Desc
stamper/csvPath	*string*	public URL for validating signed documents
stamper/xAxis	*int*	Stamper position on PDF; X-coordinates
stamper/yAxis	*int*	Stamper position on PDF; Y-coordinates
stamper/width	*int*	OPTIONAL. Stamper width
stamper/height	*int*	OPTIONAL. Stamper height
stamper/imageB64	*string*	Stamper watermark (Base64)
stamper/imageUrl	*string*	Stamper watermark (URL)
stamper/logoB64	*string*	Logo to be printed (Base64)
stamper/page	*int*	Page number where stamper will be embedded. Value `-1` for last page, `0` for all pages.
stamper/rotation	*string*	OPTIONAL. Rotation degrees: - `ROTATE_90` - `ROTATE_270`
stamper/textLine1	*string*	OPTIONAL. Text included in the stamper (line 1).
stamper/textLine2	*string*	OPTIONAL. Text included in the stamper (line 2).
stamper/textLine3	*string*	OPTIONAL. Text included in the stamper (line 3).
		Stamper type: - `PDF417` - `QR_BARCODE128` - `QR` - `BARCODE128` - `IMAGE` - `TEXT`

stamper/type	*string*	- QR_NO_TEXT - QR_SCALED - CUSTOM_TEXT - QR_REDUCED - CSV - CSV_QR - IMAGE_TEXT - DEFAULT
stamper/timeZoneId	*string*	Set the Time Zone. for stamper date to be printed

XAdES Configuration

Params only applicable to `signatureType` XAdES (XAdES B, XAdES T, XAdES LT, XAdES LTA)

```
{
  "signedInfoCanonicalizationMethod": "http://www.w3.org/TR/2001/REC-xml-c14n-20010315",
  "signedPropertiesCanonicalizationMethod": "http://www.w3.org/TR/2001/REC-xml-c14n-20010315",
  "xPathLocationString": "//book[@id='bk101-1']",
  "claimedSignerRoles": [
    "role1",
    "role2"
  ],
  "transformAlgorithms": [
    "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"
  ],
  "dssReferenceUri": "http://dsa-reference.example.com/"
}
```

Where:

Param	Type	Desc
signedInfoCanonicalizationMethod	*string*	Canonicalization Method of node `signedInfo`
signedPropertiesCanonicalizationMethod	*string*	Canonicalization Method of node `signedProperties`
xPathLocationString	*string*	XPath of ID node (XML) to be signed
claimedSignerRoles	*array*	Signer role
transformAlgorithms	*array*	Transform Algorithm of signed node: - `"http://www.w3.org/TR/2001/REC-xml-c14n-20010315"` - `"http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"` - `"http://www.w3.org/2001/10/xml-exc-c14n#"` - `"http://www.w3.org/2001/10/xml-exc-c14n#WithComments"` - `"http://www.w3.org/2006/12/xml-c14n11"` - `"http://www.w3.org/2006/12/xml-c14n11#WithComments"` - `"http://santuario.apache.org/c14n/physical"`
dssReferenceUri	*string*	ID node (XML) to be signed

TSA Configuration

TSAconfiguration is mandatory if a signature format that requires timestamp is used:

```
{
  "url": "http://tsa.example.com/",
  "user": "tsa_user",
  "password": "tsa_pass",
  "type": "USER",
  "certificateCode": "tsa_certificate_code"
}
```

Param	Type	Desc

type	*string*	Authentication type: `USER` `CERTIFICATE` `CERTIFICATE_TLS` of `URL` (if authentication is not required)
user	*string*	OPTIONAL. Only when `USER` `type` is used
password	*string*	OPTIONAL. Only when `USER` or `CERTIFICATE` or `CERTIFICATE_TLS` `type` is used
url	*string*	TSA url
certificateCode	*string*	OPTIONAL. Only when `CERTIFICATE` or `CERTIFICATE_TLS` `type` is used

POLICIES Configuration

Only applicable to XAdES EPES format; a Signature Policy can be defined:

```
{
  "id": "102039485-10283757-102837575",
  "description": "Sample policy",
  "digestAlgorithm": "SHA256",
  "digestValueB64": "JVBERi0xLjMKJcTl8uXlRU9GC",
  "url" : "https://sample/lorem_ipsum_dolor_sit_amet.pdf",
  "contentHintsDescription": "Lorem ipsum dolor sit amet",
  "contentHintsType": "Lorem ipsum dolor sit amet"
}
```

<<<<<<< HEAD | Param | Type | Desc | |----------|---------|----------| | **id** | *string* | Policy id | | **description** | *string* | Policy description | | **digestAlgorithm** | *string* | Cipher Algorithm:

- `SHA1`
- `SHA224`
- `SHA256`
- `SHA384`
- `SHA512`
- `RIPEMD160`
- `MD2`
- `MD5` | | **digestValueB64** | *string* | Policy Digest value (Base64) | | **contentHintsDescription** | *string* | Help Description |

| contentHintsType | *string* | Help content type |

Param	Type	Desc
id	*string*	Policy id
description	*string*	Policy description
digestAlgorithm	*string*	Cipher Algorithm: - `SHA1` - `SHA224` - `SHA256` - `SHA384` - `SHA512` - `RIPEMD160` - `MD2` - `MD5`
digestValueB64	*string*	Policy Digest value (Base64)
url	*string*	The SpURI (signature policy qualifier). The spURI qualifier will contain a URL value where a copy of the signing policy document can be obtained.
contentHintsDescription	*string*	Help Description
contentHintsType	*string*	Help content type

```
26367b52051e0e30d23d28b90480e0e025b5537d
```

Response

Response in `application/json` format:

```
{
  "authCode": "1aeb979ddcf247e9ad46ee73e19a326d",
  "exeCode": "f116305e7f7c44f3a29385028c5374ba"
}
```

Where:

Param	Type	Desc
authCode	*string*	Authorization code
exeCode	*string*	Execution code

API Errors

Errors are returned using `application/json` format:

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	Error code
error_description	*string*	Error description

Errors:

Error code	Error desc
invalid_request	Bad request. Incorrect of insufficient request params. (HTTP Status: 400)
invalid_token	Invalid `access_token` (HTTP Status: 401)
user_not_found	Incorrect or inactive user (HTTP Status: 404)

USER AUTHENTICATION AND CERTIFICATE SELECTION

Please review the following User Authentication and Signature Authorization section to find more details.

SIGNATURE EXECUTION

REST service specs:

Method: `POST`
URL: `{viafirma_fortress_url}/api/v1/signature/{executionCode}/execute`

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `executionCode` : authorization code returned by signature request method

Sample Request

Method: `POST`
URL: `{viafirma_fortress_url}/api/v1/signature/f116305e7f7c44f3a29385028c5374ba/execute`

Params

The request body must be an empty JSON:

```
{
}
```

Sample Response

```
[
  {
    "bytesB64": "a910b000d4f1a2b...",
    "signatureCode": "e2470412-33cc-467a-b357-880fe621920f",
    "mimeType": "application/pdf",
    "ref": "#1"
  },
  ...
]
```

Where:

Param	Type	Desc
bytesB64	*string*	Signed document (Base64). In case of asynchronous signature it will be empty and the signed document must be downloaded using the obtained `signatureCode` .
signatureCode	*string*	Signature ID
mimeType	*string*	MIME type
ref	*string*	It have the same value if present in the signature request

API Errors

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	Error code
error_description	*string*	Error description

Errors:

Error code	Error desc
invalid_request	Bad request. Incorrect on insufficient params. (HTTP Status: 400)

invalid_token	Invalid `access_token` (HTTP Status: 401)
user_not_found	Invalid or inactive user (HTTP Status: 404)
certificate_not_found	The certificate is not inactive or authorized, or has not been found (HTTP Status: 404)
signature_error	Problems found in signature process (HTTP Status: 500)

DOWNLOAD SIGNED DOCUMENT

Download signed document.

Service specs:

Method: `GET`
URL: `{viafirma_fortress_url}/api/v1/signature/download/{signature_code}`

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `signature_code` : signature code returned by fortress in the signature execution service

Sample Request

Method: `GET`
URL: `{viafirma_fortress_url}/api/v1/signature/download/C0XJ-XOAK-OF1O-TYJ7-S164-3197-3571-05`

Response

Document in `application/octet-stream` format.

API Errors

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	Error code
error_description	*string*	Error description

Errors:

Error code	Error desc
document_not_found	Signed document ID not found (HTTP Status: 404).

API: Methods related to unattended signature with certificate

```
Last update: November 13, 2024
```

The unattended signature process in Viafirma Fortress will consist of the processes of:

- Client authentication
- Signature request
- Signature execution
- Getting the signed document/s

In the following sections we will describe the methods available in Viafirma Fortress, associated with signature operations:

Note: To access these methods it is necessary to have an **access token**(`access_token`) obtained through an authentication and authorization request with a `scope` of type `client` and a `grant_type` of type `client_credentials` , [for which you must follow the steps indicated in this section of the documentation] (../../auth/README.md).

Unattended signature request

Use of the service

Method: `POST`
URL: `{viafirma_fortress_ url}/api/v1/signature`

Where:

- `viafirma_fortress_url` : Base URL of the Viafirma Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress

Additionally, the access token (`access_token`) must be included in the HTTP header of the `POST` request as follows:

```
Authorization: Bearer {access_token}
```

Example:

Method: `POST`
URL: `{viafirma_fortress_ url}/api/v1/signature`
Request header : `Authorization : Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Service parameters

This service receives through parameters the configuration of the signature used for each document to be signed, which indicates, among other things, the type of signature to be made, the document to be signed...

The parameters that are received (in `application/json` format) have the following form:

```
{
  "certificateCode":"b8a25e04ab864583bb5ea8d02883e832",
  "signatureConfigurations": [
  {
    "ref":"#1",
    "document": {
      "bytesB64":"JVBERi0xLjMKJcTl8uXrp/Og0MTGCjQ...",
```

```
      "name":"contract.pdf"
    },
    "signatureType":"PADES_B",
    "signatureAlgorithm":"RSA_SHA256",
    "packaging":"ENVELOPED",
    "padesConfiguration": {
      "stamper": {
        "csvPath":"http://localhost:7080/fortress/v#",
        "logoB64":"iVBORw0KGgoAAAANSUhEUgAAAWYAAABsCAYAAABZyhj...",
        "page": 1,
        "type":"QR_BARCODE128",
        "xAxis": 80,
        "yAxis": 700
      }
    }
  }
]
}
```

Note: Details of the `padesConfiguration`, `xadesConfiguration`, `tsa` and `policy` parameters are shown below.

Where:

Parameter	Type	Description
certificateCode	*string*	Code of the certificate to be used in the signature, it must be consulted in the**Certificates**tab of the**Configuration**section of the details of the client system or group where the certificate is hosted
certificatePassword	*string*	Contains the password of the certificate, this field is only allowed for unattended signing.
async	*string*	If set to `true` the call to the signature execution service is made asynchronously.
callbackUrl	*string*	If a URL is reported and the signing is performed asynchronously, upon completion of the signing Fortress makes a POST request to said URL with the final execution status.
signatureConfigurations/document/name	*string*	Name of the document to sign
signatureConfigurations/document/bytesB64	*string*	Document to sign, encoded in Base64
signatureConfigurations/document/url	*string*	URL of the document to sign
signatureConfigurations/signatureType	*string*	Signature type. Available values: - CADES_B - CADES_T - CADES_LT - CADES_LTA - PADES_B - PADES_T - PADES_LT - PADES_LTA - XADES_B - XADES_T - XADES_LT - XADES_LTA - PKCS1
signatureConfigurations/signatureAlgorithm	*string*	Algorithm that will be used to encrypt the signature. Available values: - `RSA_SHA1` - `RSA_SHA224` - `RSA_SHA256` - `RSA_SHA384` - `RSA_SHA512`
		Signature wrapper. Available values:

signatureConfigurations/packaging	*string*	- ENVELOPED - ENVELOPING - DETACHED
signatureConfigurations/reason	*string*	OPTIONAL, reason for signing
signatureConfigurations/location	*string*	OPTIONAL, location of the signature
signatureConfigurations/ref	*string*	If reported, the same value will be returned in the signature result.

PAdES configuration

This setting only applies for signatures whose `signatureType` is of type PAdES (PAdES B, PAdES T, PAdES LT, PAdES LTA).

```
"padesConfiguration": {
"stamper": { }
}
```

The stamper object is optional, and serves to define a visual stamp associated with the PAdES signature .

```
{
  "stamper": {
    "csvPath":"https://fortress.viafirma.com/fortress/v#",
    "imageB64":"JVBERi0xLjMKJcTl8uXlRU9GC...",
    "logoB64":"JVBERi0xLjMKJcTl8uXlRU9GC...",
    "page": 1,
    "rotation":"ROTATE_90",
    "textLine1":"Sample line 1",
    "textLine2":"Sample line 2",
    "textLine3":"Sample line 3",
    "type":"QR_BARCODE128",
    "xAxis": 100,
    "yAxis": 100,
    "timeZoneId":"America/Santo_Domingo"
  }
}
```

Parameter	Type	Description
stamper/csvPath	*string*	Signature Verification URL
stamper/xAxis	*int*	Position (on the X axis) of the signature stamp
stamper/yAxis	*int*	Position (on the Y axis) of the signature stamp
stamper/imageB64	*string*	Background image of signature stamp
stamper/logoB64	*string*	Signature Seal Logo (will be painted on the bottom right side of the seal)
stamper/page	*int*	Page where the seal will be painted. You can use the value `-1` to indicate the last page or the value `0` to paint the stamp on all pages of the document
stamper/rotation	*string*	Seal rotation. If informed, the seal will rotate the indicated degrees. Possible values: - `ROTATE_90` - `ROTATE_270`
stamper/textLine1	*string*	First textual line to be painted in the content of the signature seal
stamper/textLine2	*string*	Second textual line to be painted in the content of the signature stamp
stamper/textLine3	*string*	Third textual line to be painted in the content of the signature seal
stamper/type	*string*	Stamp type. Available values: - `PDF417` - `QR_BARCODE128` - `QR`

		- `BARCODE128` - `IMAGE` - `TEXT`
stamper/timeZoneId	*string*	String that will correspond to the standard list of time zones.

Depending on the `type` chosen, the stamp will have pre-established dimensions (in pixels):

- `PDF417` : 300x130
- `QR_BARCODE128` : 600x100
- `QR` : 450x50
- `BARCODE128` : 550x70
- `IMAGE` : Maintains the dimensions of the image specified in `imageB64`
- `TEXT` : 400x50

`timeZoneId` value is NOT reported in the API call, we will apply the following criteria:

- In case of unattended signature, if the Client System belongs to a group that has it informed, we apply the one of the first group that has it informed, if not we apply the one configured by default in the system.

XAdES Configuration

This setting only applies for signatures whose `signatureType` is of type XAdES (XAdES B, XAdES T, XAdES LT, XAdES LTA)

```
{
"signedInfoCanonicalizationMethod":"http://www.w3.org/TR/2001/REC-xml-c14n-20010315",
"signedPropertiesCanonicalizationMethod":"http://www.w3.org/TR/2001/REC-xml-c14n-20010315",
"xPathLocationString":"//book[@id='bk101-1']",
"claimedSignerRoles": [
   "role1",
   "role2"
],
"transformAlgorithms": [
"http://www.w3.org/TR/2001/REC-xml-c14n-20010315"
  ],
"dssReferenceUri":"http://dsa-reference.example.com/"
}
```

Where:

Parameter	Type	Description
signedInfoCanonicalizationMethod	*string*	Node canonicalization method `signedInfo`
signedPropertiesCanonicalizationMethod	*string*	Canonicalization method of node `signedProperties`
xPathLocationString	*string*	Selector of the node under which the signature will be inserted, in XPath format
claimedSignerRoles	*array*	Roles of the signatory
transformAlgorithms	*array*	Transformation algorithms for nodes. Possible values: - `"http://www.w3.org/TR/2001/REC-xml-c14n-20010315"` - `"http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"` - `"http://www.w3.org/2001/10/xml-exc- c14n#"` - `"http://www.w3.org/2001/10/xml-exc-c14n#WithComments"` - `"http://www.w3.org/2006/12/xml-c14n11"` - `"http://www.w3.org/2006/12/xml-c14n11#WithComments"` - `"http://santuario.apache.org/c14n/physical"`
dssReferenceUri	*string*	Identifier of the node to sign

TSA Settings

For signature types that include time stamping, the TSAsettings must be reported.

```
{
  "url":"http://tsa.example.com/",
  "user":"tsa_user",
  "password":"tsa_pass",
  "type":"USER",
  "certificateCode":"tsa_certificate_code"
}
```

Parameter	Type	Description
type	*string*	TSAtype. If it requires authentication with user and password , we will use the value `USER` , if it requires authentication with a certificate `CERTIFICATE` , if it requires authentication with a TSL certificate `CERTIFICATE_TLS` , if not, the value `URL`
user	*string*	User for TSAauthentication (only for `type` with value `USER`)
password	*string*	Password for TSAauthentication (for `type` with value `USER` or `CERTIFICATE_TLS`)
url	*string*	TSAURL
certificateCode	*string*	Certificate code for TSAauthentication (for `type` with value `CERTIFICATE` or `CERTIFICATE_TLS`)

Policy settings

So that the signature has a policy and can be considered EPES type, we can define its values with this configuration.

```
{
  "id":"102039485-10283757-102837575",
  "description":"Sample policies",
  "digestAlgorithm":"SHA256",
  "digestValueB64":"JVBERi0xLjMKJcTl8uXlRU9GC",
  "url" : "https://sample/lorem_ipsum_dolor_sit_amet.pdf",
  "contentHintsDescription":"Lorem ipsum dolor sit amet",
  "contentHintsType":"Lorem ipsum dolor sit amet"
}
```

Parameter	Type	Description
id	*string*	Policy identifier
description	*string*	Policy Description
digestAlgorithm	*string*	Encryption algorithm. Possible values: - `SHA1` - `SHA224` - `SHA256` - `SHA384` - `SHA512` - `RIPEMD160` - `MD2` - `MD5`
digestValueB64	*string*	Value (Base64 encoded)
url	*string*	The SpURI (signature policy qualifier). The spURI qualifier will contain a URL value where a copy of the signing policy document can be obtained.
contentHintsDescription	*string*	Help Description
contentHintsType	*string*	Type of help

Service response

The response from this service will be given (in `application/json` format) as follows:

```
{
  "authCode":"1aeb979ddcf247e9ad46ee73e19a326d",
  "exeCode":"f116305e7f7c44f3a29385028c5374ba"
}
```

Where:

Parameter	Type	Description
authCode	*string*	Authorization code
exeCode	*string*	Execution code

Service errors

The errors returned by the service (returned in `application/json` format) look like this:

```
{
  "error":"error_code",
  "error_description":"error_description"
}
```

Where:

Parameter	Type	Description
error	*string*	Error code
error_description	*string*	Error Description

Possible mistakes:

Error code	Error
invalid_request	Wrong request. Some of the input parameters are not correct. (HTTP Status : 400)
invalid_token	The `access_token` used is not correct (HTTP Status : 401)

Signature execution

With this method we will sign the documents associated with the signature preparation operation.

Use of the service

Method: `POST` URL: `{viafirma_fortress_ url}/api/v1/signature/{executionCode}/execute`

Additionally, the access token (`access_token`) must be included in the HTTP header of the `POST` request as follows:

```
Authorization: Bearer {access_token}
```

Where:

- `viafirma_fortress_url` : Base URL of the Viafirma Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `executionCode` : Execution code returned by the signature method

Example:

Method: `POST`
URL: `{viafirma_fortress_ url}/api/v1/signature/f116305e7f7c44f3a29385028c5374ba/execute`
Request header: `Authorization : Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Service parameters

This service receives by parameters the execution code executionCode , resulting from the signature preparation procedure, with this code we will obtain the information associated with each signing operation.

It must be provided (in `application/json` format) empty:

```
{
}
```

Service Response

The response from this service will be given (in `application/json` format) as follows:

```
[
  {
    "bytesB64":"a910b000d4f1a2b...",
    "signatureCode":"e2470412-33cc-467a-b357-880fe621920f",
    "mimeType":"application/pdf",
    "ref":"#1"
  },
...
]
```

Where:

Parameter	Type	Description
bytesB64	*string*	Signed document, Base64 encoded
signatureCode	*string*	Signature identifier
mimeType	*string*	MIME type of the signed document
ref	*string*	The same value is returned if it was reported in the signature request

Service errors

The errors returned by the service (returned in `application/json` format) look like this:

```
{
  "error":"error_code",
  "error_description":"error_description"
}
```

Where:

Parameter	Type	Description
error	*string*	Error code
error_description	*string*	Error Description

Possible mistakes:

Error code	Error
	Wrong request. Some of the input parameters are not correct. (HTTP Status : 400)

	Wrong request. Some of the input parameters are not correct. (HTTP Status : 400)
invalid_token	The `access_token` used is not correct (HTTP Status : 401)
certificate_not_found	The certificate you want to sign with is not correct or is not active (HTTP Status : 404)
invalid*certificate* password	The certificate password is incorrect (HTTP Status: 401)
locked_certificate	The certificate is blocked (HTTP Status : 401)
signature_error	Error during signing (HTTP Status : 500)

Download signed document

With this method we can download a signed document using a signature identifier.

Use of the service

Method: `GET` URL: `{viafirma_fortress_ url}/api/v1/signature/download/{signature_code}`

Additionally, the access token (`access_token`) must be included in the HTTP header of the `GET` request as follows:

```
Authorization: Bearer {access_token}
```

Where:

- `viafirma_fortress_url` : Base URL of the Viafirma Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress
- `signature_code` : Code of the signature from which you want to download the document

Example:

Method: `GET` URL: `{viafirma_fortress_ url}/api/v1/signature/download/C0XJ-XOAK-OF1O-TYJ7-S164-3197-3571-05`

Service Response

The response from this service will be given (in `application/octet-stream` format)

Service errors

The errors returned by the service (returned in `application/json` format) look like this:

```
{
  "error":"error_code",
  "error_description":"error_description"
}
```

Where:

Parameter	Type	Description
error	*string*	Error code
error_description	*string*	Error Description

Possible mistakes:

Error code	Error

document_not_found	No document found for the provided signature ID (HTTP Status : 404)

Extend Signature API

An **Access Token** is required to authorize all API requests, as explained at the following link:

get Access Token

DIGITAL EXTEND SIGNATURE REQUEST

REST service specs:

Method: `POST`
URL: `{viafirma_fortress_url}/api/v1/signature/extend`

Security:

```
Authorization: Bearer {access_token}
```

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress

Sample Request

Method: `POST`
URL: `{viafirma_fortress_url}/api/v1/signature/extend`

Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Request Params

The request body contains information such as signature format, document to be signed, etc.

`application/json` format is used:

```
{
  "extendSignatureConfigurations": [
    {
      "document": {
        "bytesB64": "JVBERi0xLjMKJcTl8uXrp/Og0MTGCjQ...",
        "name": "contract.pdf"
      },
      "signatureType": "PADES_LTA",
      "signatureAlgorithm": "RSA_SHA256",
      "packaging": "ENVELOPED",
      "padesConfiguration": {
        "stamper": {
          "csvPath": "http://<someURL>/v#",
          "logoB64": "iVBORw0KGgoAAAANSUhEUgAAAWYAAABsCAYAAABZyhj...",
          "page": 1,
          "type": "QR_BARCODE128",
          "xAxis": 80,
          "yAxis": 700
        }
      },
      "tsa": {
        "type": "URL",
        "url": "https://testservices.viafirma.com/via-tsa/tsa"
      }
    }
```

```
  ]
}
```

Note: params for `padesConfiguration`, `xadesConfiguration`, `tsa` and `policy` are described later.

Where:

Param	Type	Desc
userCode	*string*	OPTIONAL, used to to specify the signer user. If null, user will be requested to authenticate before signing.
document/bytesB64	*string*	Document to be signed (Base64)
signatureType	*string*	Signature format: - `CADES_T` - `CADES_LT` - `CADES_LTA` - `PADES_T` - `PADES_LT` - `PADES_LTA` - `XADES_T` - `XADES_LT` - `XADES_LTA` - `PKCS1`
signatureAlgorithm	*string*	signature algorithm: - `RSA_SHA1` - `RSA_SHA224` - `RSA_SHA256` - `RSA_SHA384` - `RSA_SHA512`
packaging	*string*	signature type: - `ENVELOPED` - `ENVELOPING` - `DETACHED`

PAdES Configuration

Params only applicable to `signatureType` PAdES (PAdES T, PAdES LT, PAdES LTA).

```
"padesConfiguration": {
    "stamper": { }
}
```

The stamper object is optional, and it defines a visual stamp associated with the signature PAdES.

```
{
  "stamper": {
    "csvPath": "https://sandbox.viafirma.com/fortress/v#",
    "imageB64": "JVBERi0xLjMKJcTl8uXlRU9GC...",
    "logoB64": "JVBERi0xLjMKJcTl8uXlRU9GC...",
    "page": 1,
    "rotation": "ROTATE_90",
    "textLine1": "Sample line 1",
    "textLine2": "Sample line 2",
    "textLine3": "Sample line 3",
    "type": "QR_BARCODE128",
    "xAxis": 100,
    "yAxis": 100
  }
}
```

Param	Type	Desc

stamper/csvPath	*string*	public URL for validating signed documents
stamper/xAxis	*int*	Stamper position on PDF; X-coordinates
stamper/yAxis	*int*	Stamper position on PDF; Y-coordinates
stamper/imageB64	*string*	Stamper watermark (Base64)
stamper/imageUrl	*string*	Stamper watermark (URL)
stamper/logoB64	*string*	Logo to be printed (Base64)
stamper/page	*int*	Page number where stamper will be embedded. Value `-1` for last page, `0` for all pages.
stamper/rotation	*string*	OPTIONAL. Rotation degrees: - `ROTATE_90` - `ROTATE_270`
stamper/textLine1	*string*	OPTIONAL. Text included in the stamper (line 1).
stamper/textLine2	*string*	OPTIONAL. Text included in the stamper (line 2).
stamper/textLine3	*string*	OPTIONAL. Text included in the stamper (line 3).
stamper/type	*string*	Stamper type: - `PDF417` - `QR_BARCODE128` - `QR` - `BARCODE128` - `IMAGE` - `TEXT` - `QR_NO_TEXT` - `QR_SCALED` - `CUSTOM_TEXT` - `QR_REDUCED` - `CSV` - `CSV_QR` - `IMAGE_TEXT` - `DEFAULT`
stamper/timeZoneId	*string*	Set the Time Zone. for stamper date to be printed

XAdES Configuration

Params only applicable to `signatureType` XAdES (XAdES B, XAdES T, XAdES LT, XAdES LTA)

```
{
  "signedInfoCanonicalizationMethod": "http://www.w3.org/TR/2001/REC-xml-c14n-20010315",
  "signedPropertiesCanonicalizationMethod": "http://www.w3.org/TR/2001/REC-xml-c14n-20010315",
  "xPathLocationString": "//book[@id='bk101-1']",
  "claimedSignerRoles": [
    "role1",
    "role2"
  ],
  "transformAlgorithms": [
    "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"
  ],
  "dssReferenceUri": "http://dsa-reference.example.com/"
}
```

Where:

Param	Type	Desc
signedInfoCanonicalizationMethod	*string*	Canonicalization Method of node `signedInfo`
signedPropertiesCanonicalizationMethod	*string*	Canonicalization Method of node `signedProperties`
xPathLocationString	*string*	XPath of ID node (XML) to be signed

claimedSignerRoles	*array*	Signer role
transformAlgorithms	*array*	Transform Algorithm of signed node: - `"http://www.w3.org/TR/2001/REC-xml-c14n-20010315"` - `"http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"` - `"http://www.w3.org/2001/10/xml-exc-c14n#"` - `"http://www.w3.org/2001/10/xml-exc-c14n#WithComments"` - `"http://www.w3.org/2006/12/xml-c14n11"` - `"http://www.w3.org/2006/12/xml-c14n11#WithComments"` - `"http://santuario.apache.org/c14n/physical"`
dssReferenceUri	*string*	ID node (XML) to be signed

TSA Configuration

TSA configuration is mandatory if a signature format that requires timestamp is used:

```
{
  "url": "http://tsa.example.com/",
  "user": "tsa_user",
  "password": "tsa_pass",
  "type": "USER",
  "certificateCode": "tsa_certificate_code"
}
```

Param	Type	Desc
type	*string*	Authentication type: `USER` `CERTIFICATE` `CERTIFICATE_TLS` of `URL` (if authentication is not required)
user	*string*	OPTIONAL. Only when `USER` `type` is used
password	*string*	OPTIONAL. Only when `USER` or `CERTIFICATE` or `CERTIFICATE_TLS` `type` is used
url	*string*	TSA url
certificateCode	*string*	OPTIONAL. Only when `CERTIFICATE` or `CERTIFICATE_TLS` `type` is used

POLICIES Configuration

Only applicable to XAdES EPES format; a Signature Policy can be defined:

```
{
  "id": "102039485-10283757-102837575",
  "description": "Sample policy",
  "digestAlgorithm": "SHA256",
  "digestValueB64": "JVBERi0xLjMKJcTl8uXlRU9GC",
  "contentHintsDescription": "Lorem ipsum dolor sit amet",
  "contentHintsType": "Lorem ipsum dolor sit amet"
}
```

Param	Type	Desc
id	*string*	Policy id
description	*string*	Policy description
digestAlgorithm	*string*	Cipher Algorithm: - `SHA1` - `SHA224` - `SHA256` - `SHA384` - `SHA512` - `RIPEMD160`

		- `MD2` - `MD5`
digestValueB64	*string*	Policy Digest value (Base64)
contentHintsDescription	*string*	Help Description
contentHintsType	*string*	Help content type

Response

Response in `application/json` format:

```
{
  "ref": "d8e3d98dc20e46188fd067df28048934",
  "bytesB64": "MIMBKM8GCSqGSIb3DQEHAqCDASi/MIMBKLoCAQUxDzANBglghkgBZQMEAgEFADCC1QsGCSqGSIb3DQEHAaCC1PwEgtT4JVBERi0xLjMKJcTl8uXrp..."
}
```

Where:

Param	Type	Desc
ref	*string*	reference code
bytesB64	*string*	Extend signed document (Base64)

API Errors

Errors are returned using `application/json` format:

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	Error code
error_description	*string*	Error description

Errors:

Error code	Error desc
invalid_request	Bad request. Incorrect of insufficient request params. (HTTP Status: 400)
invalid_token	Invalid `access_token` (HTTP Status: 401)
user_not_found	Incorrect or inactive user (HTTP Status: 404)

Encrypt / Decrypt API

```
Last revision: april 05th 2021
```

An **Access Token** is required to authorize all API requests, as explained at the following link:

Get Client Access Token

Encrypt / Decrypt request

Encrypt / Decrypt the request bytes with RSAusing client or group certificate.

REST service specs:

Method: `POST`
URL: `{viafirma_fortress_url}/api/v1/encrypt`

Security:

```
Authorization: Bearer {access_token}
```

Where:

- `viafirma_fortress_url` : URL of the Fortress implementation, for example https://sandbox.viafirma.com/fortress or https://fortress.viafirma.com/fortress

Sample Request

Method: `POST`
URL: `{viafirma_fortress_url}/api/v1/encrypt`

Security Header: `Authorization: Bearer 0b79bab50daca910b000d4f1a2b675d604257e42`

Request Params

The request body contains the encryption / decryption request information `application/json` format is used:

```
{
  "certificateCode": "580fe337eba1483683290cbbf94982a3",
  "mode": "ENCRYPT",
  "bytesB64": "dGVzdA=="
}
```

Where:

Param	Type	Desc
certificateCode	*string*	Used to specify which client or group certificate will be chosen to encrypt / decrypt. Certificate public key will be used to encrypt and private key to decrypt.
mode	*string*	Select encrypt or decrypt mode. The allowed values are `ENCRYPT` , `DECRYPT` .
bytesB64	*string*	Content to encrypt / decrypt

Response

Response in `application/json` format:

```
{
  "bytesB64": "ggj5mRTVh3FKAz4wf2EmaX7Zfr...=="
}
```

Where:

Param	Type	Desc
bytesB64	*string*	Encryption / decryption certificateRequestEntity (Base 64)

API Errors

Errors are returned using `application/json` format:

```
{
  "error": "error_code",
  "error_description": "error_description"
}
```

Where:

Param	Type	Desc
error	*string*	Error code
error_description	*string*	Error description

Errors:

Error code	Error desc
invalid_request	Bad request. Incorrect of insufficient request params. (HTTP Status: 400)
invalid_token	Invalid `access_token` (HTTP Status: 401)
certificate_not_found	Incorrect or inactive certificate code (HTTP Status: 404)

Quick integration examples

Note: All references to Base64-encoded files or documents are truncated to make this documentation easier to read.

User authentication, query operations

The third application"SAMPLE APP"wants to authenticate a user to query the user data, whose code is `sample_user`.

Previous requirements:

- The application must be registered as a client system in Viafirma Fortress
- You must have been provided with a `client_id`. In this example it will be `sample_app`
- You must have been provided with a `client_secret`. In this example it will be `12345`
- You must have an allowed return URL configured: `http://www.example.com/auth`

When the"SAMPLE APP"application wants to authenticate the user against Viafirma Fortress, it will redirect the user to a URL:

```
{viafirma_fortress_url}/oauth2/v1/auth?
scope=profile&
state=&
redirect_uri=http://www.example.com/auth&
response_type=code&
client_id=sample_app&
user_code=sample_user
```

In this URL the user will be presented with the different Viafirma Fortress authentication factors in which they are enrolled. You will use one of them to authenticate and authorize the operation. Once the process is finished, Viafirma Fortress will return control to the"SAMPLE APP"application, redirecting to the return URL: `http://www.example.com/auth?state=&code=e2470412-33cc-467a-b357-880fe621920f`

This URL will be sent as a URL parameter the value of the **authorization code**, with which you can request an **access token**with which to operate (e.g. obtain information about the user's status).

To obtain this **access token**, the"SAMPLE APP"application will make a request to Viafirma Fortress:

- Method: `POST`
- URL: `https://fortress.viafirma.com/fortress/oauth2/v1/token`
- Parameters:
  - `code`: Whose value is the authorization code previously obtained: `"e2470412-33cc-467a-b357-880fe621920f"`
  - `client_id`: Whose value is the one determined in Viafirma Fortress for the application"SAMPLE APP": `"sample_app"`
  - `client_secret`: Whose value is the one determined in Viafirma Fortress for the application"SAMPLE APP": `"12345"`
  - `redirect_uri`: whose value is the return URL for which the authorization request was made: `"http://www.example.com/auth"`
  - `grant_type`: This value is fixed: `"authorization_code"`

The result of this `POST` request will be:

```
{
 "access_token":"1/fFAGRNJru1FTz70BzhT3Zg",
 "expires_in": 3920,
 "token_type":"Bearer"
}
```

Once these values are obtained, we can consider that the user has been correctly authenticated. We can also use the value of `access_token` to perform query operations on the Viafirma Fortress API (for example, obtain the user's status, the certificates of a user "**scope=CERTIFICATES**" or the detail of a certificate "**scope=CERTIFICATE**").

Signing a PDF document

The third application"SAMPLE APP"wants the user `sample_user` to sign a PDF document.

Previous requirements:

- The application must be registered as a client system in Viafirma Fortress
- You must have been provided with a `client_id` . In this example it will be `sample_app`
- You must have been provided with a `client_secret` . In this example it will be `12345`
- You must have an allowed return URL configured: `http://www.example.com/sign`

Get client token

At the time when the"SAMPLE APP"application wants to start the PDF document signing operation, it must obtain a client system token.

To obtain this **access token**, the"SAMPLE APP"application will make a request to Viafirma Fortress:

- Method: `POST`
- URL: `https://fortress.viafirma.com/fortress/oauth2/v1/token`
- Parameters:
  - `client_id` : Whose value is the one determined in Viafirma Fortress for the application "SAMPLE APP": `"sample_app"`
  - `client_secret` : Whose value is the one determined in Viafirma Fortress for the application "SAMPLE APP": `"12345"`
  - `redirect_uri` : whose value is the return URL for which the authorization request was made: `"http://www.example.com/auth/response"`
  - `grant_type` : This value is fixed: `"client_credentials"`

```
https://fortress.viafirma.com/fortress/oauth2/v1/token?
grant_type=client_credentials&
redirect_uri=http://www.example.com/auth/response&
client_id=sample_app&
client_secret =12345
```

The result of this `POST` request will be:

```
{
  "access_token":"666b3b58ecb54db784e2eafdfc66e113",
  "expires_in": 3920,
   "token_type":"Bearer"
}
```

Signature Request

With the access_token resulting from the call, the client system will call the signature method :

Method: `POST` URL: `https://fortress.viafirma.com/fortress/api/v1/signature` Request header : `Authorization : Bearer 666b3b58ecb54db784e2eafdfc66e113`

```
{
  "userCode":"abcde",
  "redirectUri":"http://localhost:8080/fortress-demo/sign",
  "signatureConfigurations": [
```

```
    {
      "signatureType":"PADES_B",
      "signatureAlgorithm":"RSA_SHA256",
      "packaging":"ENVELOPED",
      "document": {
        "name":"example.pdf",
        "bytesB64":"JVBERi0xLjMKJcTl8uXrp/Og0MTGCjQgMCBvYmoKPDwgL0xlbmd0aC..."
      },
      "padesConfiguration": {
        "stamper": {
          "csvPath":"http://localhost:7080/fortress/v#",
          "logoB64":"iVBORw0KGgoAAAANSUhEUgA...",
          "page": 1,
          "type":"QR_BARCODE128",
          "xAxis": 80,
          "yAxis": 700
        }
      }
    }
  ]
}
```

In the body of the method the system must include a json with the following format:

- userCode : user code
- redirectUri : Uri where you should redirect the operation once it is finished
- signatureConfigurations : for each document to be signed, the document, the type of signature and the signing policies must be indicated.

The result of this `POST` request will be:

```
{
  "authCode":"d8e3d98dc20e46188fd067df28048934",
  "exeCode":"cae2c9fe4f4b41888d42ac18a88096a2"
}
```

Signature Request Authorization

When the"SAMPLE APP"application wants to begin the signing operation of the PDF document, it will redirect the user to a URL to authorize the signing operation and select the certificate to use:

```
https://sandbox.viafirma.com/fortress/oauth2/v1/auth?signature_code=7b3e77ad2aef4e479c2ae39f497cfe0c&scope=signature&client_id
=fortress-dem&redirect_uri=https%3A%2F%2Fsandbox.viafirma.com%2fortress-demo%2Fsign%2Fresponse
```

In this URL the user will be presented with the different Viafirma Fortress authentication factors in which they are enrolled. You will use one of them to authenticate and authorize the signing operation. Once authenticated, you will be shown the different certificates that Viafirma Fortress is holding for this user, so you can select which one you want to sign with.

Execute Signature

Once these values are obtained, we can consider that the user has been correctly authenticated and has authorized the signing operation, so the signing service can be called. To do this, a request is made to Viafirma Fortress, including the access token and certificate identifier obtained in the previous step:

- HTTP method: `POST`
- URL: `https://fortress.viafirma.com/fortress/api/v1/signature/cae2c9fe4f4b41888d42ac18a88096a2/execute` Request header : `Authorization : Bearer 666b3b58ecb54db784e2eafdfc66e113`

The response of this service will be:

```
{
  "documentB64":"LjMKJcTl8u...",
  "mimeType":"application/pdf",
  "signatureCode":"TFOR-TRES-SOAK-OF1O-TXFR-5151-8007-9109-77"
}
```


In the `documentB64` attribute we will have the signed document (encoded in Base64), and in `signatureCode` the signature identifier.

Signature Extension

With the access_token resulting from the call, the client system will call the extend method:

Method: `POST` URL: `https://fortress.viafirma.com/fortress/api/v1/signature/extend` Request header : `Authorization : Bearer 666b3b58ecb54db784e2eafdfc66e113`

```
{
  "extendSignatureConfigurations": [
    {
      "document": {
        "bytesB64": "JVBERi0xLjMKJcTl8uXrp/Og0MTGCjQ...",
        "name": "contrato.pdf"
      },
      "signatureType": "PADES_LTA",
      "signatureAlgorithm": "RSA_SHA256",
      "packaging": "ENVELOPED",
      "tsa": {
        "type": "URL",
        "url": "https://testservices.viafirma.com/via-tsa/tsa"
      }
    }
  ]
}
```

In the body of the method the system must include a json with the following format:

- userCode : user code
- redirectUri : Uri where you should redirect the operation once it is finished
- extendSignatureConfigurations : for each document to be signed, the document, the type of signature and the signing policies must be indicated.

The result of this `POST` request will be:


```
{
"ref":"d8e3d98dc20e46188fd067df28048934",
"bytesB64":"MIMBKM8GCSqGSIb3DQEHAqCDASi/MIMBKLoCAQUxDzANBglghkgBZQMEAgEFADCC1QsGCSqGSIb3DQEHAaCC1PwEgtT4JVBERi0xLjMKJcTl8uXrp.
.."
}
```

Sample application

A sample Fortress client application is provided to help developers to integrate third-party apps with **Viafirma Fortress**. This basic web application shows how to use the main API services:

- Retrieve user information (to check if it has active certificates and IDPs).
- User authentication.
- Authorization request to sign a document.
- PAdES and XAdES signature.
- Batch signature.

Source code

Prerequisites

- JDK 1.7+
- Maven 3.0+

Request credentials from the commercial department:

- `client_id`. In this example it will be `sample_app`
- `client_secret`. In this example it will be `12345`

Quickstart

The sample application is based on Spring Boot and includes an embedded Tomcat server to simplify the execution.

- The app can be imported in any IDE and execute `com.viafirma.fortress.demo.FortressDemoApplication` class.
- Once imported, you must configure the credentials provided in the fortress-demo.properties file located in /src/main/resource, for example:

  ```
  fortress.demo.api.url=https://sandbox.viafirma.com/fortress/
  fortress.demo.api.client_id=sample_app
  fortress.demo.api.client_secret=12345
  ```

  Once set up, you will be able to:
  - Start the application with the command `mvn spring-boot:run`.
  - Compile the application with `mvn clean package` and deploy the WAR to a Tomcat container or run it directly:

```
java -jar target/viafirma-fortress-demo.war
```

Testing the application

Once the application has started, the main page is accesible in any web browser:

```
http://localhost:8080/fortress-demo/
```

it is also available in the sandbox environment:

```
https://sandbox.viafirma.com/fortress-demo/
```

Login screen

The first interface (login screen) is just used to get Fortress user code. Please enter a valid user code (for instance, `12345678Z`) and any password (it will not be checked). If the user code does not exist, user will not be able to check authentication or digital signature with a centralized digital certificate.

Main screen

If the entered user has certificates / IDPs associated, authentication and signing buttons are enabled, with different options to sign documents: PDF (PAdES), XML (XAdES) or batch signature.

User authentication

Test user `12345678Z` can be used, with PIN IDP `1234` and OTP (soft token) IDP based on Google Authenticator, scanning the attached QR Code:

Viafirma Fortress Desktop - Windows client

```
Rev: 2019-01-14
```

Viafirma Fortress Desktop client (available for Windows 7-8-10), the centralized certificates can be used for authentication and signature operations in external websites and applications, as if the certificates were locally installed.

Production version

The production version is available in the following URL:

- https://fortress.viafirma.com/

Other environments

Developers usually use the application in different environments for testing purposes. There is a version which allows users to modify the Fortress endpoint. It is available in the following URLs:

- Viafirma Fortress Desktop for DEVELOPERS 64 bits
- Viafirma Fortress Desktop for DEVELOPERS 32 bits