MobileID InApp

Web integration guide

2020-04-02

Version 0.1

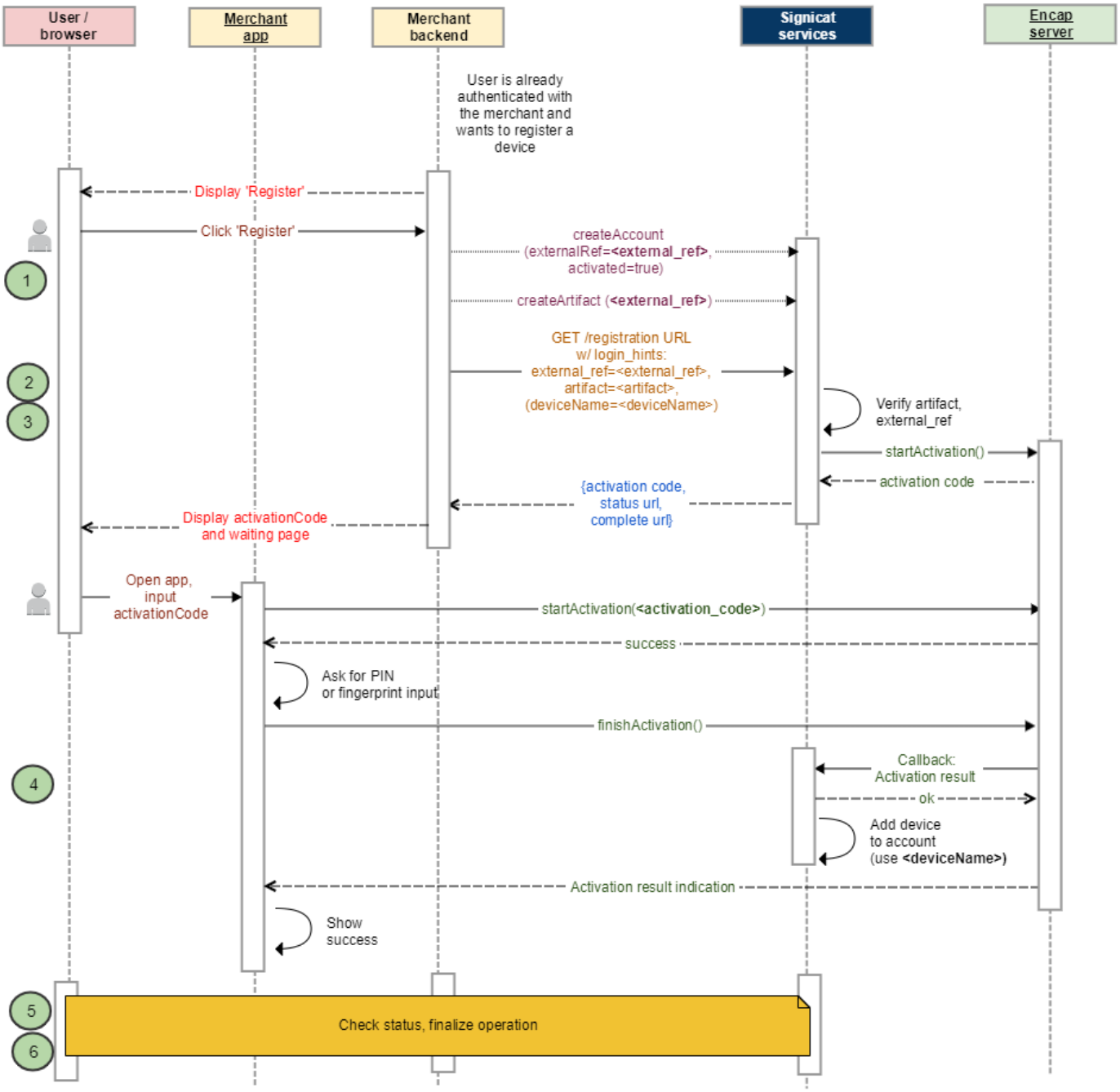
Contents

This guide illustrates how to integrate with MobileID InApp by starting the registration and authentication steps on the merchant's website.

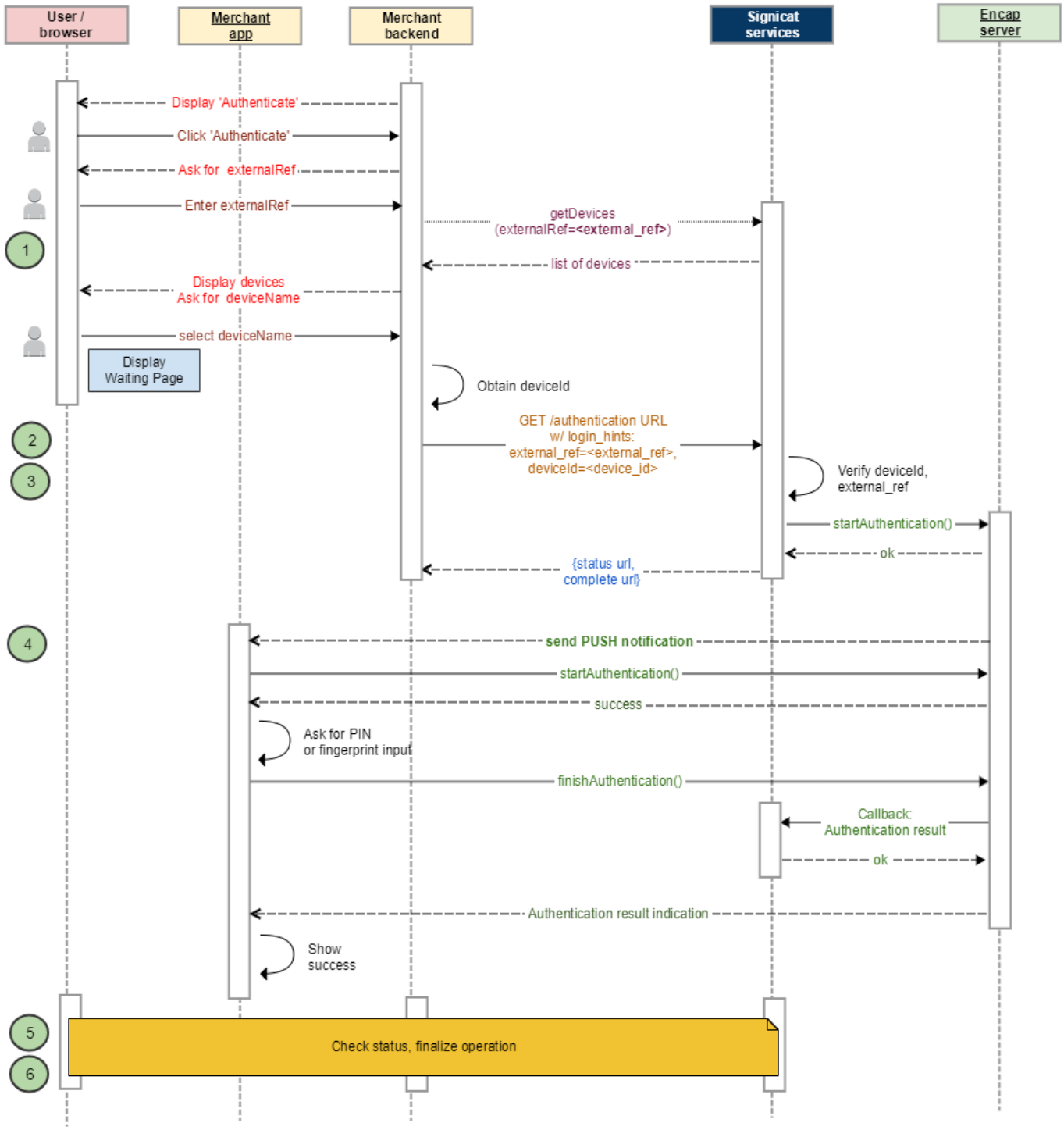
1 Integration process

1.1 Sequence diagrams for registration and authentication

1.1.1 *Registration*



1.1.2 *Authentication*



1.2 Detailed description of registration and authentication

The following table provides step-by-step instructions on the integration process. The numbering in the table corresponds to the numbered steps in the sequence diagrams above.

	Registration	**Authentication**

1. Initiate operation on merchant server	The user must already be authenticated by the merchant backend before he can register his device. He starts the registration process by clicking the **Register** button that is displayed in the browser.
	
	The merchant backend generates `externalRef` and `deviceName`, if needed.
	
	Account and artifact creation
	In order to carry out registration, an account with Signicat needs to be created. The merchant backend uses two SOAP WS calls:
	
	• One call for account creation
	  ○ It is important that the flag `activated` is true
	  ○ It is possible to add additional attributes to the account at the time of creation
	• One call for creation of `artifact` that has to be passed to Signicat's registration service when operation is initiated
		In order to start authentication process, the user must provide his `externalRef` and the name of the device he wants to use for authentication.
		
		Enter externalRef
		When the user provides the `externalRef`, the first call toward the merchant backend is executed. The goal of this call is to fetch all available devices suitable for authentication for the supplied `externalRef`.
		
		The list of devices is then presented to the user.
		
		Select device
		The user selects the device and initiates the authentication process by clicking the **Authenticate** button. A new request is sent to the merchant backend (the `deviceName`, together with the previously entered `externalRef`), and made available to the merchant backend.
		
		Based on the supplied `deviceName`, the merchant backend obtains the `deviceId` (normally, it caches `deviceName`/`deviceId` pairs when fetching all available devices for the user).

2. Generate URL	The merchant backend constructs the operation URL as shown in the MobileID InApp web integration guide URL construction guide.	
3. Initiate operation on Signicat's server	The merchant backend executes an HTTP GET request with the URL constructed previously. See the normal response in the URL construction guide linked to above. ***Note***: To be able to perform the subsequent requests, you must keep the cookies you receive and make these available for subsequent requests. **Response error example** `{ "completeUrl": "https://id.signicat.com/...", "status": "ERROR", "error": { "code": "urn:signicat:error:idp:ACCESS_DENIED", "message": "Access denied. Wrong credentials." }, ... }`	

	Registration	Authentication

	• If an error occurs during initialization, you will receive a status indicating this, and an error object will be present. Upon error, if you choose to make a GET request towards the `completeUrl`, you will get
	
	`error=access_denied&`
	`error_description=The Resource Owner did not complete the login.`
	`urn:signicat:error:idp:ACCESS_DENIED; Access denied. Wrong credentials.`	

4. Execute operation toward Encap	If the status was "OK", the merchant backend will respond to the browser with the `activationCode` received from Encap (the `activationCode` is displayed in the browser). The user switches to the merchant app and enters the displayed `activationCode`. The merchant app continues the registration process: This involves the regular **startActivation()** / **finishActivation()** calls towards the Encap Client API. Immediately after the `activationCode` is displayed in the browser, the merchant backend starts polling toward Signicat, awaiting the status of the operation.	If the status was "OK", the merchant app will receive a PUSH message with information about the required authentication. The user switches to the Merchant App and starts the authentication process by entering the PIN. The authentication process continues in the merchant app: This involves the regular **startAuthentication()** / **finishAuthentication()** calls towards the Encap Client API. Immediately after the `deviceName` is displayed in the browser, the merchant backend starts polling toward Signicat, awaiting the status of the operation.
5. Check process status	The client (browser) may execute polling calls to the merchant backend using the status URL from steps 1-3, which executes a call to Signicat. This can be executed (periodically at pre-configured intervals) until the received result is `COMPLETED`.	
6. Get result of the process — Finalize operation	When the status from the previous call is `COMPLETED`, the client executes a finalizing call to the merchant backend that again uses the received `completeUrl` and executes a call to Signicat. Signicat then sends an `authorization_`code to the merchant backend which carries out the regular OIDC `authorization_code` sequence of steps to obtain the device information. See the MobileID InApp integration guide - Finalize operation guide for details.	

2 URL construction guide

2.1 Requests and responses

2.1.1 *Registration*

Registration OIDC request	Registration response

Without PKCE
`GET <SIGNICAT_AUTHORIZATION_ENDPOINT>?`
`response_type=code&`
`scope=openid+profile+mobileid&`
`client_id=<CUSTOMER_CLIENT_ID>&`
`redirect_uri=<CUSTOMER_REDIRECT_URI>&`
`state=<CUSTOMER_REG_METHOD_NAME:STATE_IDENTIFIER>&`
`acr_values= urn:signicat:oidc:method:<CUSTOMER_REG_METHOD_NAME>&`
`login_hint=deviceName-<DEVICE_NAME>&`
`login_hint=artifact-<ARTIFACT>&`
`login_hint=externalRef-<ACCOUNT_NAME>`
 	
With PKCE
`GET <SIGNICAT_AUTHORIZATION_ENDPOINT>?`
`response_type=code&`
`scope=openid+profile+mobileid&`
`client_id=<CUSTOMER_CLIENT_ID>&`
`redirect_uri=<CUSTOMER_REDIRECT_URI>&`
`code_challenge=<CODE_CHALLENGE>&`
`code_challenge_method=S256&`
`state=<CUSTOMER_REG_METHOD_NAME:STATE_IDENTIFIER>&`
`acr_values= urn:signicat:oidc:method:<CUSTOMER_REG_METHOD_NAME>&`
`login_hint=deviceName-<DEVICE_NAME>&`
`login_hint=artifact-<ARTIFACT>&`
`login_hint=externalRef-<ACCOUNT_NAME>`
	`{`
	`"status":"<STATUS>",`
	`"activationCode":"<ACTIVATION_CODE>",`
	`"statusUrl":"<STATUS_URL>",`
	`"completeUrl":"<COMPLETE_URL>"`
	`}`

2.1.2 *Authentication*

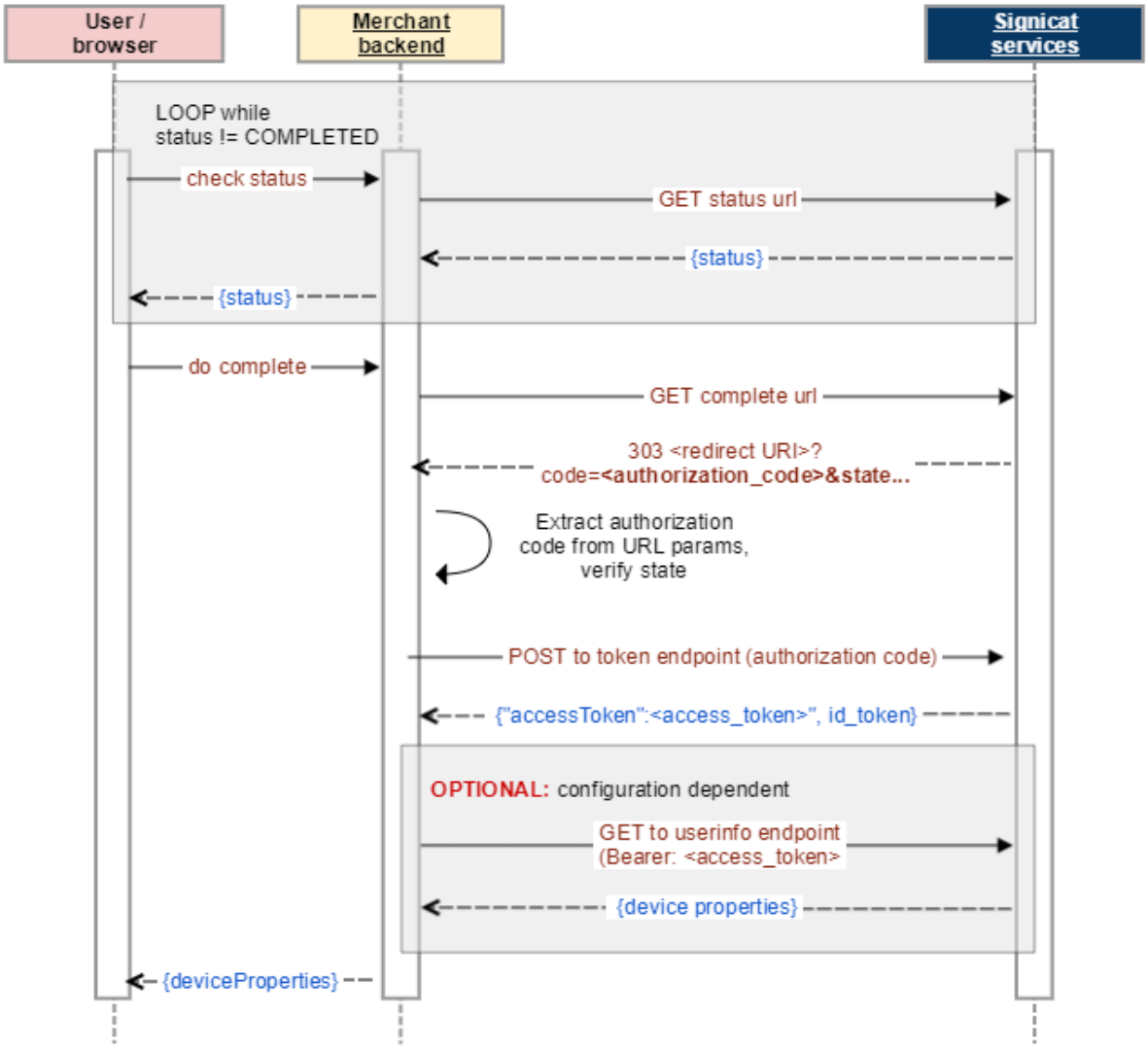
Authentication OIDC request	Authentication response

Without PKCE
`GET <SIGNICAT_AUTHORIZATION_ENDPOINT>?`
`response_type=code&`
`scope=openid+profile+mobileid&`
`client_id=<CUSTOMER_CLIENT_ID>&`
`redirect_uri=<CUSTOMER_REDIRECT_URI>&`
`state=<CUSTOMER_AUTH_METHOD_NAME:STATE_IDENTIFIER>&`
`acr_values= urn:signicat:oidc:method:<CUSTOMER_AUTH_METHOD_NAME>&`
`login_hint=deviceId-<DEVICE_ID>&`
`login_hint=externalRef-<ACCOUNT_NAME>`
With PKCE
`GET <SIGNICAT_AUTHORIZATION_ENDPOINT>?`
`response_type=code&`
`scope=openid+profile+mobileid&`
`client_id=<CUSTOMER_CLIENT_ID>&`
`redirect_uri=<CUSTOMER_REDIRECT_URI>&`
`code_challenge=<CODE_CHALLENGE>&`
`code_challenge_method=S256&`
`state=<CUSTOMER_AUTH_METHOD_NAME:STATE_IDENTIFIER>&`
`acr_values= urn:signicat:oidc:method:<CUSTOMER_AUTH_METHOD_NAME>&`
`login_hint=deviceId-<DEVICE_ID>&`
`login_hint=externalRef-<ACCOUNT_NAME>`
	`{`
	`"status":"<STATUS>",`
	`"statusUrl":"<STATUS_URL>",`
	`"completeUrl":"<COMPLETE_URL>"`
	`}`

2.2 Parameters

Parameter	Description
`STATE_IDENTIFIER`	Random text used together with `CUSTOMER_REG_METHOD_NAME` to uniquely identify the ongoing registration session in the merchant's backend. The session state can be compared when callback/ redirect data is received from Signicat.
`ACTIVATION_CODE`	Code to be used with Encap.
`STATUS_URL`	URL (towards Signicat's server) that is used to get the status of the ongoing operation.
`COMPLETE_URL`	URL (towards Signicat's server) that is used to signal the completion of the transaction. This will need to be used when the merchant's app gets notification from the MobileID App that the registration is done.
`DEVICE_ID`	Device ID
`CODE_CHALLENGE`	PKCE Code Challenge. Base64UrlEncoded SHA256 of the the value for `CODE_VERIFIER` (to be used later when the authentication code is exchanged for `access_token`)
`CODE_CHALLENGE_METHOD`	PKCE Code Challenge Method. Recommended: S256

3 Finalize operation



Note: This operation is carried out in the same way regardless of whether the operation in question is **registration** or **authentication**.

3.1 Check status

Note: This is just one implementation possibility. It is possible to execute the implementation in different ways.

While the operation (registration, authentication) is ongoing between the merchant app and Signicat, the client (browser) may execute polling calls to the merchant backend using the previously received status URL, which executes a call to Signicat.

This can be executed (periodically at pre-configured intervals) until the received result is `COMPLETED`.

Request	Response
`GET <STATUS_URL>`	`{ "status": "PENDING"/"COMPLETED" }`

3.2 Complete operation

Request	Response	Comment
`GET <COMPLETE_URL>`	`AUTHORIZATION_CODE`	Signicat's server sends an authorization code to the `CUSTOMER_REDIRECT_URL`.

`POST <SIGNICAT_TOKEN_ENDPOINT>`
`HTTP/1.1`
`Content-Type: application/json`
`Authorization: Basic <CUSTOMER_BASIC_AUTH_HEADER>`
`#body`
`client_id=<CUSTOMER_CLIENT_ID>&`
`redirect_uri=<CUSTOMER_REDIRECT_URI>&`
`grant_type=authorization_code&`
`code=<AUTHORIZATION_CODE>`
	`{ "access_token": "<ACCESS_TOKEN>", "token_type": "Bearer", ... }`	The authorization code is exchanged for an access token, id token, and optionally refresh token.

[OPTIONAL]
`GET <SIGNICAT_USERINFO_ENDPOINT>`
`HTTP/1.1`
`Content-Type: application/json`
`Authorization: Bearer <ACCESS_TOKEN>`
	For registration:
	`{ "sub": "<SUBJECT>", "name": "<EXTERNAL_REF>" ... }`
	
	For authentication:
	`{ "sub": "<SUBJECT>", "externalRef": "<EXTERNAL_REF>", "deviceName": "<DEVICE_NAME>", ... }`
		Additional information (such as data on the authenticated user) can be retrieved from Signicat's OIDC server using the /userinfo endpoint.