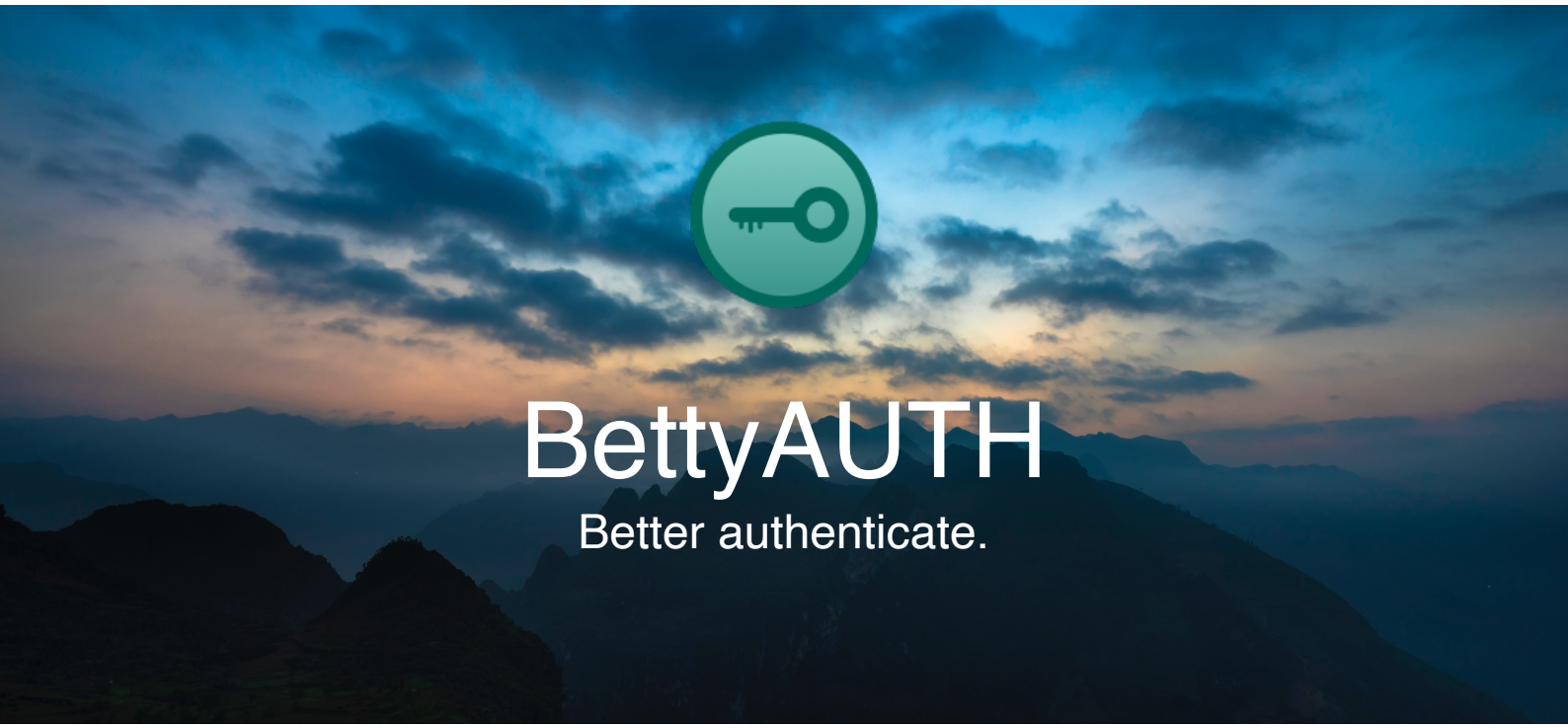
BETTY SOFT

Bettysoft AuthSDK

December 2023 | English



About this SDK

Bettysoft AuthSDK is designed to offer you an easy way to build security into your products. Simple to use, Bettysoft AuthSDK runs in websites and Apps written in Swift.
This SDK includes:

- API written in PHP
- Framework written in Swift

Setup

Get your API key & configure Management Console

Sign in to BettyAUTH Management Console with your Bettysoft Account and navigate to Licence Overview to get your API key. This key is needed to authenticate yourself on every API call. Do not share this key with anyone else!
If you are going to use the API in a PHP project and host it on a domain, you have the ability to enter your domain. Optionally you can also validate your domain.
If you want to enter and / or validate your domain, you can easily do this by following the next steps:

Enter domain:

Notice: You do not need to enter a domain or validate it. It is an extra feature and let your customers trust you even more.

1. Click BettyAUTH Settings and enter your domain in the input field. Please make sure, you also enter possible subdomains and make sure your web server uses https!

   Validate domain:

   Notice: Your domain can only be validated, if it supports https.

   a. Create a <API key>.txt-file. (Filename = API key, filetype = txt). This file can be empty.
   b. Upload the text file to your server. Make sure, it is accessible under the same domain you entered earlier.
2. Finish by clicking **Save**.

Your domain should be visible in the input field. If you validated your domain, you also see a green checkmark.

Include API in your project

To work properly, you have to upload the file *bs_authsdk.php* to your server and include it to every page that uses BettyAUTH API functionality.

```
include 'bs_authsdk.php';
```

Include framework in your project

Drag the Xcode file in your project and link the framework in the project settings. Then import the code by using:

```
import BS_AuthSDK
```

Using the API

Send a request to the server

1. Create a new object of the class *BS_AuthSDK*. With the method *sendRequest()* you are able to send requests to the BettyAUTH-server.

```
$api_call = new BS_AuthSDK();
```

2. Send your request to the server.
   a. The method *sendRequest(string $method, string $url, $data)* needs the following three parameters. The first one specifies the method that is used to send the request. BettyAUTH offers POST, PUT and DELETE as the three standard request methods.

   Second parameter is *$url* as described in the following table.

url type: string	***Description***
id	Calls the ID class to get necessary variables and methods.
tan	Calls the TAN class to get necessary variables and methods.

The third parameter *$data* is a json encoded string, that stores additional information for the request in form of key-value-pairs (kvp). The kvp *apiKey* (which stores your apiKey) and *ip* (that specifies the clients ip address) are needed in every request you send to the API server.

In addition to these two you need the following kvp described in the table below.

method type: string	***url*** type: string	***data (key:value)***			***Description***
		action type: string	***id*** type: integer	***tan*** type: integer	
POST	**id**	create	–	–	Creates a new ID to store in your database.
Per user in your database you have to create exactly one ID to let BettyAUTH work correctly for this user.					
PUT	**tan**	–	needed	–	Generates a new tan for the given *id*.
POST	**tan**	–	needed	needed	Checks the given *tan* for the given *id*

A request to generate a new TAN for the id 12345678 would then look like the following. You have to replace the apiKey with your personal apiKey.

```
$api_call = new BS_AuthSDK("");
$result = $api_call->sendRequest("PUT","tan",
 json_encode(array(
      "apiKey" => "ABCD123…4567890",
      "id" => 12345678,
      "ip" => $_SERVER["REMOTE_ADDR"]
 )));
```

b. Every call of *sendRequest()* returns a http response code (*status*), *data* and a *message*. You can access these attributes by calling the equivalent BS_AuthSDK class-methods *status()*, *data()* and *message()*. The following table explains the different response codes and shows which data and messages are responded.

method type: string	***url*** type: string	***action*** type: string	***response***			***Description***
			status	***data***	***message***	
POST	**id**	**create**	200	id type:integer	–	ID successfully created.
			400	–	Unable to create ID	ID was not created or invalid request.
			401	–	Authentication error	API authentication error.
			403	–	Access denied	Calling IP address is blocked by the system.
PUT	**tan**	**–**	201	–	–	Successfull operation.
			400	–	Unable to generate TAN	TAN was not created or invalid request.
			401	–	Authentication error	Authetication error.
			403	–	Access denied	Calling IP address is blocked by the system.

method type: string	***url*** type: string	***action*** type: string	***response***			***Description***
			status	***data***	***message***	
POST	**tan**	**–**	200	–	–	Successfull operation.
			400	–	–	Invalid request.
			401	–	Authentication error	Wrong tan or API authentication error.
			403	–	Access denied	Calling IP address is blocked by the system.

Using the Framework

Send a request to the server

1. Create a new object of the class *BS_AuthSDK*. This class implements several methods to offer full BettyAUTH functionality in your product. The constructor of this class gets your API key as a parameter.

```
let apiCall = BS_AuthSDK("API_KEY")
```

2. Send your request to the server.
   a. To function, every request with the implemented class methods needs parameters. In table 2.2 you can see which parameters are needed for the specific method.
   b. Parameters needed for API calls.

method	**parameters**		***Description***
	id	***tan***	
createID()	-	-	Creates a new ID to store in your database.
Per user in your database you have to create exactly one ID to let BettyAUTH work correctly for this user.			
generateTan()	needed	-	Generates a new tan for the given *id*.
checkTan()	needed	needed	Checks the given *tan* for the given *id*

All parameters are of type *string* and given to the method as a parameter or parameter list.

A request to create a TAN for the id 12345678 could look like the following.

```
apiCall->generateTan("12345678")
```

   c. The API uses completion handlers to give feedback to the user, that means that every Class-method call end in a success or failure response.

Also, every API call sends a http response with possible data and message. The following table explains the different response codes and shows which data and messages are responded.

method	*response*			*Description*
	code	*data*	*message*	
createID()	100	–	–	Calling IP address is blocked by the system.
	200	id	–	ID successfully created.
	400	–	Unable to create ID	ID was not created or invalid request.
	401	–	Authentication error	API authentication error.
generateTan()	100	–	–	Calling IP address is blocked by the system.
	201	–	–	Successfull operation.
	400	–	Unable to create TAN	TAN was not created or invalid request.
	401	–	Authentication error	Authetication error.
checkTan()	100	–	–	Calling IP address is blocked by the system.
	200	–	–	Successfull operation.
	400	–	–	Invalid request.
	401	–	Authentication error	Wrong tan or API authentication error.

Include certificate in your product

To show your customers data security is important to you and that you protect their data with BettyAUTH, you can include the "Protected with BettyAUTH"-certificate in your product.

Please make sure, you are always using the current certificate. You can find it here: https://bettyauth.de/seal.png