KontoCloud E-Wallet Platform

iOS SDK Integration

Publication date: July 17, 2017

ContoWorks GmbH

Data classification:	NDA Confidential
Document identifier:	EC8337FF-D51E-464D-BD84-5B3CEA96CCE4

Contents

1. Add KontoCloud SDK to your app

To integrate the KontoCloud SDK libraries into your own project, you need to perform a few basic tasks to prepare your project. This tutorial assumes your IDE for application development is XCode.

Drag KontoCloudSDK.framework and KontoCloudSDK.bundle from Finder into the Frameworks group in Xcode. Ensure that you’ve checked the check box "Copy items if needed" and the radio button "Create groups".

2. KontoCloud SDK

2.1. Payment Form

2.1.1. Request authorization token

In order to show Payment Form your app should request an authorization token from the KontoCloud API by calling following API method depending on payment form mode:

Payment Form Mode	**API method**
REGISTRATION	Init Add Stored Payment Option
PAYMENT	Init Authorize

In case of your app is using KontoCloud SDK API Services see Init add stored payment option.

1.1.2. Initialize and render Payment Form

Initialize CWPaymentForm and add it as a subview to the one of your views. You should add it in viewDidLayoutSubview method so that it is correctly displayed.

```
- (void)viewDidLayoutSubviews {
    [super viewDidLayoutSubviews];
    if (!_isAlreadyLayout) {
        _isAlreadyLayout = YES;

        CWPaymentForm *paymentForm = [[CWPaymentForm alloc]
initWithFrame:_webViewContainer.bounds];
        [_webViewContainer addSubview:paymentForm];

        paymentForm.mode = FormMode_REGISTRATION;
        paymentForm.apiURL = Constants.API_URL;
        paymentForm.providerMode = ProviderMode_TEST;

        [paymentForm setOnSuccessCallback:^(NSString *paymentOptionCode,
NSString *authorizationToken, NSString *payerID){

        }];

        [_paymentForm setOnBeforeSubmitCallbackWithArg:

               ^(id<onBeforeSubmitArgInterface> args) {

         // For example, disable Submit button and show loading indicator.

        }];
        [paymentForm renderWithPaymentOptionCode:PaymentOption_VISA
authorizationToken:authToken formStyle:nil];
    }
}

- (IBAction)saveAction:(id)sender {
    [paymentForm submit];
}
```

1.1.2.1. UI Customization

The Payment Form uses CWFormStyle class to customize UI. You can specify properties such as font size, font color, etc. Any customized element on the form belongs to the class CWInputStyle. CWInputStyle has several options that you can use for customization:

Style property	**Description**
labelFontStyle	The label style.
editTextFontStyle	The style of edit text.
validationHintFontStyle	The validation hints style.

editTextBorderStyle	The field borders style.

You can use paymentFormInputStyle property of CWFormStyle class to set the style for all elements on the form. If you want to customize specific input, you can use properties of CWFormStyle listed below:

Property	**Description**
CWFormStyle.paymentFormCardNumberInputStyle	Style for card number input
CWFormStyle.paymentFormExpiryDateInputStyle	Style for expiry date input
CWFormStyle.paymentFormCardHolderInputStyle	Style for card holder input
CWFormStyle.paymentFormCvvInputStyle	Style for cvv input
CWFormStyle.paymentFormAccountHolderInputStyle	Style for account holder input
CWFormStyle.paymentFormIbanInputStyle	Style for iban input

Example of usage

```
CWFormStyle *formStyle = [CWFormStyle new];

        // set style for all inputs on the payment form

        formStyle.paymentFormInputStyle.labelFontStyle.textSize = 12;
        formStyle.paymentFormInputStyle.labelFontStyle.textColor = [UIColor
grayColor];
        formStyle.paymentFormInputStyle.labelFontStyle.textStyle =
CWTextStyle_BOLD | CWTextStyle_ITALIC;
  formStyle.paymentFormInputStyle.editTextBorderStyle.bottomBorderStyle.size
= 2;
 formStyle.paymentFormInputStyle.editTextBorderStyle.bottomBorderStyle.color
= [UIColor colorWithWhite:0 alpha:0.5];

      // set style for a specific input
      formStyle.paymentFormCardNumberInputStyle.isPlaceholderVisible = NO;
        formStyle.paymentFormCardNumberInputStyle.labelFontStyle.fontName =
@"HelveticaNeue-Bold";

        formStyle.paymentFormCardNumberInputStyle.labelFontStyle.textSize =
20;

        formStyle.paymentFormCardNumberInputStyle.labelFontStyle.textStyle =
CWTextStyle_ITALIC;



        [paymentForm renderWithPaymentOptionCode:PaymentOption_VISA
authorizationToken:authToken formStyle:formStyle];
```

2.1.2.2 Payment form wizard

The payment form can be shown in a wizard format. I.e. you can set one specific field visible at the current step, validate it, and then show the next one.

Example of usage

```
PaymentFormElement activeElement = PaymentFormElement_CardNumber;

// Set specific payment form field visible
[paymentForm setElement:PaymentFormElement_All visible:NO];
[paymentForm setElement:activeElement visible:YES];


// Validate visible payment form element when some button touched
// If the last visible element validated then submit payment form
[paymentForm validateElement:activeElement callback:^(PaymentFormElement
element, BOOL isValid) {
// go to next wizard step

        if (isValid) {
            if (activeElement == PaymentFormElement_CVV) {
                [paymentForm submit];
            }
        }
    }];
```

2.1.2.3 Accept the Direct Debit Authorization

To add Direct Debit authorization support implement the onBeforeSubmitCallbackWithArg handler of the PaymentForm.

1. Check if Bank Account is selected and direct debit authorization is not accepted yet.

2. Prevent form from being submitted.

3. Show confirmation dialog.

4. Re-submit form if the user presses the positive button.

See full example below

```
    // payment form initialization

        __block BOOL isDirectDebitAuthorizationAccepted = NO;
        __weak typeof(_paymentForm) weakPaymentForm = _paymentForm;
        [_paymentForm
setOnBeforeSubmitCallbackWithArg:^(id<onBeforeSubmitArgInterface> args){
            NSLog(@"%@ %@", args.accountHolder, args.iban);

// Prevent form from being submitted if Bank Account is selected and direct
debit authorization is not accepted yet

            if (!isDirectDebitAuthorizationAccepted &&
[args.paymentOptionCode
isEqualToString:CWLookups.PaymentOptionCode_BankAccount]) {

// Preventing form from being submitted
                [args preventSubmit];

// Extracting user input
                NSString *accountHolder = args.accountHolder;
                NSString *iban = args.iban;

// Constructing confirmation dialog
                UIAlertController *alertController = [UIAlertController

alertControllerWithTitle:@"Accept the direct debit authorization"
                                                      message:[NSString
stringWithFormat:@"Please accept the direct debit authorization below to let
us automatically pull funds from your bank account.\n\nAccount Holder:
%@\nIBAN: %@", accountHolder, iban]

preferredStyle:UIAlertControllerStyleAlert];
                UIAlertAction *acceptAction = [UIAlertAction
actionWithTitle:@"Accept" style:UIAlertActionStyleDefault
handler:^(UIAlertAction * _Nonnull action) {
// Re-submit form after direct debit authorization is accepted
                    isDirectDebitAuthorizationAccepted = YES;
                    [weakPaymentForm submit];
                }];
                [alertController addAction:acceptAction];

                UIAlertAction *cancelAction = [UIAlertAction
actionWithTitle:@"Cancel" style:UIAlertActionStyleCancel
handler:^(UIAlertAction * _Nonnull action) {

                }];
                [alertController addAction:cancelAction];

                [wSelf presentViewController:alertController animated:YES
completion:nil];
            }

// Disable Submit button and show loading indicator

        }];

// ….
```

1.1.3. Complete storing a payment option

To complete registration/payment process your app should call the following methods of the KontoCloud API.

Payment Form Mode	**API methods**
REGISTRATION	Complete Add Stored Payment Option
PAYMENT	Complete Authorize Capture

In case of your app is using KontoCloud SDK API Services see Complete add stored payment option.

1.1.4. More examples

1.1.1.1. Common use case for any payment provider

```
CWPaymentForm *paymentForm = [[CWPaymentForm alloc] initWithFrame:_webViewContainer.bounds];


// Set the appropriate form mode. FormMode_REGISTRATION for register payment
option or FormMode_PAYMENT for making a payment
paymentForm.mode = FormMode_PAYMENT;

// Set the appropriate form provider mode.
paymentForm.providerMode = ProviderMode_TEST;

// Set current payment provider.
paymentForm.paymentProvider = PaymentOptionProvider_Payon;

// Pass the same redirect URL as used in the API method "Init Authorize".
This option should be set for PaymentOS, CyberSource, CyberSource with
TokenEX payment providers.
paymentForm.redirectURL = redirectUrl;

// Set KontoCloud API URL for payment form
paymentForm.apiURL = kAPIUrl;

// Attach callbacks
[_paymentForm setOnSuccessCallback:^(NSString * _Nullable paymentOptionCode,
NSString * _Nullable authorizationToken, NSString * _Nullable payerID) {
    // Complete registration/payment process (see 2.1.3)
}];

[_paymentForm setOnCancel:^(NSString * _Nullable token) {
    // Handle cancellation from paypal
}];

[_paymentForm setOnError:^(NSString * _Nullable errorMessage) {
   // Handle error
}];

// Render payment form
[paymentForm renderWithPaymentOptionCode:selectedPaymentOption authorizationToken:authorizationToken
formStyle:formStyle];
```

1.1.1.2. PayPal direct integration

Pass redirect URL and cancel URL to the Payment Form and also set the specified payment provider

```
CWPaymentForm *paymentForm = [[CWPaymentForm alloc] initWithFrame:_webViewContainer.bounds];


paymentForm.mode = FormMode_PAYMENT;
paymentForm.paymentProvider = PaymentOptionProvider_Paypal;
// Set KontoCloud API URL for payment form
paymentForm.apiURL = kAPIUrl;


// Attach callbacks
[_paymentForm setOnSuccessCallback:^(NSString * _Nullable paymentOptionCode,
NSString * _Nullable authorizationToken, NSString * _Nullable payerID) {
    // Complete registration/payment process (see 2.1.3)
}];

[_paymentForm setOnCancel:^(NSString * _Nullable token) {
    // Handle cancellation from paypal
}];

[_paymentForm setOnError:^(NSString * _Nullable errorMessage) {
    // Handle error
}];

// Render payment form
[paymentForm renderWithPaymentOptionCode:PaymentOption_PAYPAL authorizationToken:authorizationToken
formStyle:formStyle];
```

2. API Services

2.1. Initialization

Firstly, import <KontoCloudSDK/KontoCloudSDK.h>

To initialize KontoCloud SDK create an instance of *CWApiProvider* with specific parameters (*CWApiProviderOption*).

```
CWDefaultAccessTokenProvider *accessTokenProvider =
[CWDefaultAccessTokenProvider new];

// Configure API provider
CWApiProviderOption *apiProviderOptions = [CWApiProviderOption new];

apiProviderOptions.url = @"{api url}";
apiProviderOptions.programCode = @"{program code}";
apiProviderOptions.accessTokenProvider = accessTokenProvider;

CWApiProvider *apiProvider = [[CWApiProvider alloc]
initWithApiOption:apiProviderOptions];
```

1.2. Call anonymous API method

To call API method get an instance of API service using *CWApiProvider*.

List of all API services described on page 17.

```
// Getting instance of API client
CWUserAPI *userApi = apiProvider.userAPI;


// Call API method

CWCheckUserUniquenessRequest *request = [[CWCheckUserUniquenessRequest
alloc] init];
[userApi checkUserUniqueness:request userId:@"{new user id}"
completion:^(NSError *error, CWCheckUserUniquenessResponse *response) {
        if (response) {
            BOOL isUserFound = [response found];
        }
    }];
```

1.3. Authentication

To add authentication support to your app

- Implement your own *CWAccessTokenProvider*
- Use *CWDefaultAccessTokenProvider* included in the KontoCloud SDK

Then, pass an instance of *CWAccessTokenProvider* to the *CWApiProviderOption* during initialization.

API service requests *CWAccessTokenProvider* for an access token each time an API method is called.

Access token and refresh token (*CWAccessTokenResponse*) can be obtained through the

- CWAuthenticationApi.getAccessToken using user id and password
- CWAuthenticationApi.getRefreshToken using refresh token

Access tokens have a limited lifetime determined by the specified session timeout of the KontoCloud API endpoint. If an application uses an expired access token, a HTTP error 401 is returned.

To obtain a new access token without the user id and password (when access token is expiring or expired) use refresh token.

Refresh tokens are subject to strict storage requirements to ensure that they are not leaked.

Detailed authentication process described below.

1.3.1. Authenticate user

Call [authenticationAPI getAccessTokenWithUserName: password: grantType: completion:] to retrieve *CWAccessTokenResponse*.

Then, configure *CWAccessTokenProvider* to return access token.

```
// Getting instance of authentication API service
CWAuthenticationAPI *authenticationAPI = apiProvider.authenticationAPI;


// Retrieving access token
[authenticationApi getAccessTokenWithUserName:@"{user id}" password:@"{user
password}" grantType:@"password" completion:^(NSError *error,
CWAccessTokenResponse *response) {

// Authenticating user
        [apiProvider.apiOption.accessTokenProvider
setAccessToken:response.access_token];

// Store token details
        NSString *refreshToken = response.refresh_token;
        NSDate *accessTokenExpireDate = [[NSDate date]
dateByAddingTimeInterval:[response.expires_in integerValue]];

    }];
```

1.1.2. Refresh user access token

Call [authenticationAPI getRefreshTokenWithToken: grantType: completion:] to retrieve new *CWAccessTokenResponse*.

Then, configure *CWAccessTokenProvider* to return new access token.

```
// Getting instance of authentication API service
CWAuthenticationAPI *authenticationAPI = apiProvider.authenticationAPI;


// Assume that the user is already authenticated

BOOL isTokenExpired = [[NSDate date] compare:accessTokenExpireDate] ==
NSOrderedAscending;
if (!isTokenExpired) {
   [authenticationApi getRefreshTokenWithToken:refresh_token
grantType:@"refresh_token" completion:^(NSError *error,
CWAccessTokenResponse *response) {

         // Refreshing user token
         [apiProvider.apiOption.accessTokenProvider
setAccessToken:response.access_token]

          // Refresh stored token details
          refreshToken = response.refresh_token;
          accessTokenExpireDate = [[NSDate date]
dateByAddingTimeInterval:[response.expires_in integerValue]];
     }];
}
```

1.4. More examples

1.1.1. Find account number

```
// Getting instance of API service
CWAccountAPI *accountAPI = apiProvider.accountAPI;

// Assume that the user is already authenticated

// Getting account number
CWFindAccountRequest *request = [[CWFindAccountRequest alloc]
initWithUserId:@"{user id}"];

[accountAPI findAccount:request completion:^(NSError *error,
CWFindAccountResponse *response) {
     NSString *accountNumber = response.accno;
}];
```

1.1.1.

1.1.2. Get account information

```
// Getting instance of API service
CWAccountAPI *accountAPI = apiProvider.accountAPI;

// Assume that the user is already authenticated

// Getting account information for authenticated user
CWGetAccountInformationRequest *request = [[CWGetAccountInformationRequest
alloc] initWithAccnoType:@"{accno type}"];

[accountAPI getAccountInformation:request accno:@"{account number}"
completion:^(NSError *error, CWGetAccountInformationResponse *response) {
     NSString *firstName = response.firstName;
     NSString *lastName = response.lastName;
}];
```

1.1.2.

1.1.3. Get payment information

```
// Getting instance of API service

CWPaymentAPI *paymentAPI = apiProvider.paymentAPI;

// Getting payment information by unique reference
CWGetPaymentInformationRequest *request = [[CWGetPaymentInformationRequest
alloc] init];
[paymentAPI getPaymentInformation:request uniqueReference:@"{unique
reference}" completion:^(NSError *error, CWGetPaymentInformationResponse
*response) {
        NSString *customerFullName = response.customerFullName;
}];
```

1.1.3.

1.1.4. Send password reset code

```
// Getting instance of API service
CWUserAPI *userAPI = apiProvider.userAPI;


CWSendPasswordResetCodeRequest *request = [[CWSendPasswordResetCodeRequest
alloc] init];
[userAPI sendPasswordResetCode:request userId:@"{user id}"
completion:^(NSError *error, CWSendPasswordResetCodeResponse *response) {

}];
```

1.1.4.

1.1.5. Init add stored payment option

```
// Getting instance of API service
CWAccountAPI *accountAPI = apiProvider.accountAPI;


CWInitAddStoredPaymentOptionRequest *request =
[[CWInitAddStoredPaymentOptionRequest alloc]
initWithAccnoType:@"{accnoType}" paymentOptionCode:@"{paymentOptionCode}"];

[accountAPI initAddStoredPaymentOption:request
accno:@"{accountNumber}"completion:^(NSError *error,
CWInitAddStoredPaymentOptionResponse *response) {

}];
```

1.1.5.

1.1.6. Complete add stored payment option

```
// Getting instance of API service
CWAccountAPI *accountAPI = apiProvider.accountAPI;


CWCompleteAddStoredPaymentOptionRequest *request =
[CWCompleteAddStoredPaymentOptionRequest new];
request.paymentOptionCode = @"{paymentOptionCode}";
request.useDifferentBillingAddress = NO;
request.accnoType = @"{accnoType}"
request.authorizationToken = @"{authorizationToken}";
request.expiryMonth = @"{expiryMonth}";
request.expiryYear = @"{expiryYear}";
request.carrierNumber = @"{carrierNumber}";

[accountAPI completeAddStoredPaymentOption:request accno:@"{accno}"
completion:^(NSError *error, CWCompleteAddStoredPaymentOptionResponse
*response) {

}];
```

1.1.6.

3. Push notifications

KontoCloud push notifications are delivered to the device using Firebase Cloud Messaging.

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably deliver messages.

3.1. Add Firebase and FCM SDK to your app

To add Firebase to your app you'll need a Firebase project and a Firebase configuration file for your app.

1. Create a Firebase project in the Firebase console, if you don't already have one. If you already have an existing Google project associated with your mobile app, click **Import Google Project**. Otherwise, click **Create New Project**.
2. Click **Add Firebase to your iOS app** and follow the setup steps. If you're importing an existing Google project, this may happen automatically and you can just download the config file.

3. When prompted, enter your app's bundle ID. It's important to enter the bundle ID your app is using; this can only be set when you add an app to your Firebase project.
4. At the end, you'll download a GoogleService-Info.plist file.
5. If you haven't done so already, copy this into your Xcode project root.

Learn more about Firebase
https://firebase.google.com/docs/ios/setup

1.2. Register account device

To start receiving messages you should register the Firebase token from the KontoCloud API.

```
// Retrieve the current registration token
NSString *firebaseToken = [[FIRInstanceID instanceID] token];


// Get instance of API client
CWAccountAPI *accountApi = apiProvider.accountAPI;

// Assume that the user is already authenticated

// Call register account device API method to associate the device for
notifications with an account.
CWRegisterAccountDeviceRequest *request = [[CWRegisterAccountDeviceRequest
alloc] initWithAccnoType:accnoType deviceType:deviceType
clientId:firebaseToken];

[accountAPI registerAccountDevice:request accno:accno completion:^(NSError
*error, CWResponseBase *response) {

}];
```

1.3. Unregister account device

To stop receiving messages you should unregister the Firebase token from the KontoCloud API.

```
// Retrieve the current registration token
NSString *firebaseToken = [[FIRInstanceID instanceID] token];

// Get instance of API client
CWAccountAPI *accountApi = apiProvider.accountAPI;

// Assume that the user is already authenticated

// Call unregister account device API method to remove the device from
notifications associated with an account.

CWUnregisterAccountDeviceRequest *request =
[[CWUnregisterAccountDeviceRequest alloc] initWithAccnoType:accnoType
deviceType:deviceType clientId:firebaseToken];

[accountAPI unregisterAccountDevice:request accno:accno completion:^(NSError
*error, CWResponseBase *response) {

}];
```

1.4. Receive messages

To receive messages you should implement some of the methods

```
- (void)application:(UIApplication *)application
didReceiveRemoteNotification:(NSDictionary *)userInfo {

    [self processNotificationWithDict:userInfo fetchCompletionHandler:nil];
}

- (void)application:(UIApplication *)application
didReceiveRemoteNotification:(NSDictionary *)userInfo
fetchCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler
{
    [self processNotificationWithDict:userInfo
fetchCompletionHandler:completionHandler];
}

- (void)userNotificationCenter:(UNUserNotificationCenter *)center
didReceiveNotificationResponse:(UNNotificationResponse *)response
withCompletionHandler:(void (^)())completionHandler {
    NSDictionary *userInfo = response.notification.request.content.userInfo;
    [self processNotificationWithDict:userInfo fetchCompletionHandler:nil];
}

- (void)applicationReceivedRemoteMessage:(FIRMessagingRemoteMessage
*)remoteMessage {
    if (remoteMessage.appData) {
        [self processNotificationWithDict:remoteMessage.appData
fetchCompletionHandler:nil];
    }
}

- (void)processNotificationWithDict:(NSDictionary *)userInfo
fetchCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler
{
// Process notification data parameters and show message
}
```

Learn more about Firebase Cloud Messaging
https://firebase.google.com/docs/cloud-messaging/

1.1.1. RemoteMessage data parameters

Field	Description
accno	Recipient account number. For example you can show notifications for logged in user only (notification accno should be equal to current user account number)
uniqueReference	Unique transaction reference assigned by KontoCloud. Can be useful to determine if the transaction is initiated on the device or not. You may want to hide notifications for transactions initiated on the device
notificationMessageCode	Message code Can be useful for constructing localized messages. List of all supported notification message codes described *below*
amount	Message amount
currencyCode	Message currency code
ownerName	Message owner name

1.1.1.

1.1.2. Notification message codes

Push notifications do not contain ready to use localized message string.

To construct a localized message, use the notificationMessageCode parameter of the RemoteMessage object.

Here is a list of all supported notification message codes:

Notification Message Code	Example Message Template
LDACCT-Credit	Funds loaded successfully: {amount} {currencyCode}.
ULDACCT-Debit	Funds withdrawn successfully: {amount} {currencyCode}.
KC-CPTR-Debit	You've paid {amount} {currencyCode} to {ownerName}.
KC-RFND-Credit	You received a refund of {amount} {currencyCode} from {ownerName}.
KC-RFND-Debit	You sent a chargeback of {amount} {currencyCode} to {ownerName}.
KC-CHBK-Debit	You sent a chargeback of {amount} {currencyCode} to {ownerName}.
TFRAMNT-Credit	You received {amount} {currencyCode} from {ownerName}.
TFRAMNT-Debit	You sent {amount} {currencyCode} to {ownerName}.
AJTBAL-Credit	Your balance was adjusted by +{amount} {currencyCode}.
AJTBAL-Debit	Your balance was adjusted by -{amount} {currencyCode}.

3. Class reference

1. Payment Form

1.1. CWPaymentForm

Member	Type	Description
mode	PaymentFormMode	Sets payment form mode.
providerMode	PaymentProviderMode	Sets payment provider mode.
paymentProvider	PaymentOptionProvider	Sets payment option provider.
showStorePaymentMethod	BOOL	Set this value to true to add a checkbox to the payment form to let the customer decide whether or not to store the card details. Supported in the PAYMENT mode only.
redirectURL	NSString	The same redirect URL as used in the API method "Init Authorize"
apiURL	NSString	KontoCloud API URL

onSuccessCallback	void	Invoked when a payment form has been successfully submitted.
onFormLoaded	void	Invoked when a payment form has been successfully loaded.
onCancel(NSString * _Nullable token)	void	Used in paypal direct integration to cancel current form.
onError(NSString * _Nullable errorMessage)	void	Invoked when there has been an error during a payment form submission.
onBeforeSubmitCallbackWithArg(id<onBeforeSubmitArgInterface> args)	void	Invoked when a payment form has been validated and is going to be submitted.

renderWithPaymentOptionCode:(PaymentOptionCode) authorizationToken:(NSString *) formStyle:(CWFormStyle *)	void	Renders payment form for specified parameters. Supported payment option codes: “*BNKACCT*” – Bank Account “*VISA*” – Visa “*MSTRCRD*” – MasterCard “*PAYPAL*” – PayPal *"PAYU"* – PayU
renderWithPaymentOptions:(NSArray<NSString *> *) authorizationToken:(NSString *) formStyle:(CWFormStyle *)	void	Renders payment form for specified parameters. Supported payment option codes: “*VISA*” – Visa “*MSTRCRD*” – MasterCard If the *paymentOptionCodes* parameter contains more than one credit card code (e.g. { "VISA", "MSTRCRD" }) then payment form detects the payment option automatically based on the first four digits entered in the credit card number field.
submit	void	Validates and submits the payment form to the server. If the validation fails, the submit does not perform.

setElement:(PaymentFormElement)element visible:(BOOL)isVisible;	Void	Set specific form field visibility. Supported fields: “PaymentFormElement_All” – to set all elements visible/invisible “PaymentFormElement_CardNumber” – card number field “PaymentFormElement_CardHolder” – card holder field “PaymentFormElement_Expiry” – card expiration field “PaymentFormElement_CVV” – cvv field
validateElement:(PaymentFormElement)element callback:(void (^_Nonnull)(PaymentFormElement element, BOOL isValid))validationCallback;	void	Validate specific form field.

1.2. onSuccessCallback

Member	**Type**	**Description**
paymentOptionCode	NSString	It will contain a payment option code if payment form has been successfully submitted Supported paymentOptionCode values: “*BNKACCT*” – Bank Account “*VISA*” – Visa “*MSTRCRD*” – MasterCard
authorizationToken	NSString	It will contain an authorization token if payment form has been successfully submitted
additionalParam	NSString	It will contain an additional parameter, wich is used for CWCompleteAuthorizeRequest for example in paypal direct integration or for cybersource payment provider.

1.3.

1.3.

1.4. onBeforeSubmitCallbackWithArg(id<onBeforeSubmitArgInterface> args)

Member	**Type**	**Description**
accountHolder	NSString	Contains account holder if selected payment option is Bank account. Otherwise, it will be nil.
iban	NSString	Contains iban if selected payment option is Bank account. Otherwise, it will be nil.
paymentOptionCode	NSString	Contains selected payment option
originArgs	NSDictinonary	Contains all data sent by the payment form
preventSubmit()	void	Prevents form from being submitted.

1.5.

1.5. PaymentFormElement

Member	**Type**	**Description**
PaymentFormElement_CardNumber	int	Payment form card number field
PaymentFormElement_CardHolder	int	Payment form card holder field
PaymentFormElement_Expiry	int	Payment form card expiration field
PaymentFormElement_CVV	int	Payment form card cvv field
PaymentFormElement_All	int	Used to set all payment form elements visible/unvisible

1.6.

1.6. PaymentOptionProvider

Member	**Type**	**Description**
PaymentOptionProvider _Payon	NSUInteger	Returns Payon payment option provider. Supported payment options: VISA, MSTRCRD, BNKACCT, PAYPAL
PaymentOptionProvider _PaymentOS	NSUInteger	Returns PaymentOS payment option provider. Supported payment options: PAYU
PaymentOptionProvider _Paypal	NSUInteger	Returns PayPal payment option provider. Supported payment options: PAYPAL
PaymentOptionProvider _Cybersource	NSUInteger	Returns Cybersource payment option provider. Supported payment options: VISA, MSTRCRD, AMEX, MSTRO, DISCOVER
PaymentOptionProvider _CybersourceWithToken Ex	NSUInteger	Returns CybersourceWithTokenEx payment option provider. Supported payment options: VISA, MSTRCRD, AMEX, MSTRO, DISCOVER
PaymentOptionProvider _VestaWithTokenEx	NSUInteger	Returns VestaWithTokenEx payment option provider. Supported payment options: VISA, MSTRCRD, AMEX, MSTRO, DISCOVER
PaymentOptionProvider _Sepa	NSUInteger	Returns Sepa payment option provider. Supported payment options: BNKACCT
PaymentOptionProvider _PayonWithPCIProxy	NSUInteger	Returns PayonWithPCIProxy payment option provider. Supported payment options: VISA, MSTRCRD, AMEX, MSTRO

1.7. PaymentFormMode

Member	**Type**	**Description**
FormMode_UNDEFINED	int	The default value. Means that this property is not assigned and it must be assigned.
FormMode_REGISTRATION	int	Payment option registration mode. Use this mode to add a stored payment option to an e-wallet account.
FormMode_PAYMENT	int	Payment option for payment mode. Use this mode to make a payment to an e-wallet account.

1.8. PaymentProviderMode

Member	Type	Description
ProviderMode_UNDEFINED	int	The default value. Means that this property is not assigned and it must be assigned.
ProviderMode_TEST	int	Payment provider test mode. Use this mode for development and testing.
ProviderMode_LIVE	int	Payment provider live mode. Use this mode for production.

1.9.

1.9. Styles

This section contains the list of available styles, attributes supported by them and their predefined values.

1.9.1. CWFormStyle

Member	Type	Description
labelFontStyle *(deprecated)*	CWFontStyle	Labels text style on the payment from.
editTextFontStyle *(deprecated)*	CWFontStyle	Text fields text style on the payment form.
editTextBorderStyle *(deprecated)*	CWBorderStyle	Text fields border style.
validationHintFontStyle *(deprecated)*	CWFontStyle	Hint labels text style on the payment form.
isPlaceholderVisible *(deprecated)*	BOOL	Defines visibility of placeholders. Default value is true.
paymentFormInputStyle	CWInputStyle	Input style for all inputs on the payment form.
paymentFormCardNumberInputStyl e	CWInputStyle	Input style for card number input on the payment form.
paymentFormExpiryDateInputStyle	CWInputStyle	Input style for expiry date input on the payment form.
paymentFormCardHolderInputStyle	CWInputStyle	Input style for card holder input on the payment form.

paymentFormCvvInputStyle	CWInputStyle	Input style for cvv/cvc input on the payment form.
paymentFormShowCVVHint	BOOL	If set to true then the credit card form will display a hint on where the CVV is located. Default value is false.

1.9.2.

1.1.2. CWInputStyle

Member	**Type**	**Description**
labelFontStyle	CWFontStyle	Labels text style on the payment from.
editTextFontStyle	CWFontStyle	Text fields text style on the payment form.
editTextBorderStyle	CWBorderStyle	Text fields border style.
validationHintFontStyle	CWFontStyle	Hint labels text style on the payment form.
isPlaceholderVisible	BOOL	Defines visibility of placeholders. Default value is true.

1.1.3.

1.1.3. CWFontStyle

Attribute Name	**Type**	**Value**
textSize	float	14
textColor	UIColor	[UIColor black]
textStyle	CWTextStyle	CWTextStyle_REGULAR
fontName	NSString	@"HelveticaNeue"

1.1.4.

1.1.4. CWTextStyle

Member	**Type**	**Description**

CWTextStyle_REGULAR	int	Regular text style
CWTextStyle_BOLD	int	Bold text style
CWTextStyle_ITALIC	int	Italic text style

1.1.5.

1.1.5. CWBorderStyle

Member	**Type**	**Description**
leftBorderStyle	CWBorderSideStyle	Left border style
rightBorderStyle	CWBorderSideStyle	Right border style
topBorderStyle	CWBorderSideStyle	Top border style
bottomBorderStyle	CWBorderSideStyle	Bottom border style

1.1.6.

1.1.6. CWBorderSideStyle

Attribute Name	**Type**	**Value**
sizer	float	1
color	UIColor	[UIColor black]

1.1.7.

2. API Services

2.1. CWAccessTokenResponse

Member	Type	Description
access_token	NSString	Access token. Provide this token with *AccessTokenProvider* to authenticate requests. See *Authenticate* user
expires_in	NSNumber	Access token expires in seconds.
refresh_token	NSString	Refresh token. Special kind of token that is used to authenticate a user without them needing to re-authenticate. See *Refresh user access token*

2.2.

1.2. CWAccessTokenProvider

Member	Type	Description
getAccessToken	NSString	Returns access token for request. If accessToken is null – API requests will be anonymous.
setAccessToken:	void	Set access token for request

1.3.

1.3. CWApiProvider

Member	Type	Description
accountAPI	CWAccountAPI	Returns account API service
authenticationAPI	CWAuthenticationAPI	Returns authentication API service
paymentAPI	CWPaymentAPI	Returns payment API service
settingsAPI	CWSettingsAPI	Returns settings API service
userAPI	CWUserAPI	Returns user API service

1.4.

1.4. CWApiProviderOption

Member	**Type**	**Description**
accessTokenProvider	*id <CWAccessTokenProvider>*	An instance of the the class which implement protocol *CWAccessTokenProvider*. If accessTokenProvider is null all requests are anonymous. Default implementation: *CWDefaultAccessTokenProvider*
partnerReferencePrefix	NSString	Partner reference prefix for all requests.
programCode	NSString	Program code
url	NSString	KontoCloud API URL.

1.5.

3. SDK Constants

You can use CWLookups class to get some of the sdk constants.

Member	**Type**	**Description**
SDKVersion	NSString	Current SDK version
PaymentOptionCode_MasterCard	NSString	MasterCard payment option code
PaymentOptionCode_Visa	NSString	Visa payment option code
PaymentOptionCode_BankAccount	NSString	Bank account payment option code
PaymentOptionCode_PayU	NSString	PayU payment option code
PaymentOptionCode_Discover	NSString	Discover payment option code
PaymentOptionCode_Amex	NSString	America express payment option code
PaymentOptionCode_Maestro	NSString	Maestro payment option code