SimplePay API v2.1 oneclick and token (recurring) payment

Development Documentation

Payment process and development documentation for stored card payments

16.08.2020

CONTENTS

Document History

Date	Version	Change
22.02.2019	190222	Original issue
22.03.2019	190322	Inclusion of OneClick payments, taken from the general documentation
05.04.2019	190405	Expansion of the test elements, Chapter 8
03.06.2019	190603	Revision of the documentation
16.06.2019	190616	Revision before compilation
15.01.2020	200115	Adding Strong Customer Authentication (SCA, 3DS)
16.03.2020	200316	Clarifications of the strong customer authentication process - do process - dorecurring process
07.04.2020	200407	New processes - Simplified two-step card registration process
29.05.2020	200529	New processes - do - API v2 payment with a card stored using API v1 - do - API v2 payment with a card stored in OTP Webshop (Middleware, MW) service - tokenquery - query token status - tokencancel - inactivate token
16.08.2020	200816	Clarifications - Strong user authentication - Introducing threeDSReqAuthMethod New processes - Legacy oneclick and recurring card storage

1 Short Summary

1.1 SimplePay Online Payment v2.0 and v2.1

SimplePay is the online payment solution of OTP Mobil Kft. This documentation was created for the v2.1 card storage and payment based on the v2 version of the merchant API and the transaction management to be used for SimplePay payments.

The description explains three kinds of card-storage-based business logic and technical implementation:

- **oneclick**: recurring payments requiring customer presence
- **recurring**: automatically recurring payments, using tokens
- **legacy:** card storage in the OTP Bank system for both oneclick and recurring

The above two topics were previously discussed in separate documents, but in the following, this common documentation discusses all of the card storage solutions for the SimplePay system (v2.1 API).

This documentation does not address "Simple" card storage, selectable by the customer on the SimplePay payment page, as that does not require merchant-side development.

This documentation only provides information on the recurring payments mentioned above, but basic know-how and the payment process with no card storage is not discussed.

This documentation discusses strong customer authentication **(PSD2, SCA, 3DS)** only in relation to stored card payments.

Prior knowledge of the information contained in the general documentation of the SimplePay payment process is therefore crucial in interpreting the forthcoming information.

The technical solution to strong customer authentication is under active development for stored card payments. For this reason the documentation may be expanded more often by additions to descriptions of existing processes or upon introduction of new functions.

The current version of SimplePay's general technical documentation is available in the download section of the merchant admin interface (for both the sandbox and the live systems), or at the following URL:

Hungarian version: http://simplepartner.hu/download.php?target=v21dochu
English version: http://simplepartner.hu/download.php?target=v21docen

This documentation refers to the aforementioned general documentation, or assumes knowledge of it, in all sections when discussing the given topic.

1.2 Conditions of deployment

Every merchant system (web store, mobile application) is tested before deployment. In case of the application of card storage, this is done even if this function has been added to a previously tested system.

In the case of card storage, beyond the basic payment tests, the checking described in the "**Testing**" section of this documentation shall be also performed.

1.3 Definitions

Card Registration: an option of the bank card payment, during which the details of the card used for the payment will be saved in the SimplePay system. Only successfully authorised cards' data can be saved.
PCI-DSS (Payment Card Industry Data Security Standard): data security standard for bank card storage. In the SimplePay system, the cards are stored in a continuously audited environment in accordance with the PCI-DSS requirements.
cardSecret: a unique piece of data provided by the customer at the registration transaction, which only he/she knows. In the followings this must be entered every time a transaction is initiated with the saved card to prove that he/she is present.
OneClick payment: the customer initiates the payment by using the **cardSecret** on a case by case basis. It is fundamentally a convenience service because afterwards the card number is not required to be provided on the payment page.
Token: an identifier generated during the registration payment, which can be used to initiate subsequent payments with the saved card.
Recurring payment: recurring payments automatically initiated by the merchant without the cardholder's presence, using the **token**, e.g. in case of subscription based services
Legacy card storage: Card storage in the OTP Bank system for both oneclick and recurring.

1.4 OneClick payment

OneClick Payment allows customers to save their credit cards during the transaction. During the next purchase of the customer, it will not be necessary to redirect him or her to SimplePay's payment page; it will be enough to start the transaction in the background, so the customer remains on the merchant web site all this time. This card-storage-based payment is only applicable if the customer is present at the purchase and the payment transaction.

1.5 OneClick Payment transaction process

1.5.1 Payment with card registration using OneClick

The first transaction involves the registration of the customer's card. **3DS** authentication is performed on the payment page during the first card registration transaction; the merchant system is only required to send the necessary customer data in the **"start"** request.

a. On the merchant's website the customer selects the items to be purchased and the total sum to be paid is calculated.
b. The merchant enables the customer on the website to register a unique identifier in the SimplePay system (**cardSecret**). This data must be added to the start function when the transaction is initiated. The merchant system is not permitted to save the cardSecret value, it may only forward it to SimplePay. For OneClick payments to be initiated later, transactions may only be initiated when the customer has provided the cardSecret.
c. The merchant transfers the transaction data to SimplePay via the API (**start**). At this point the payment transaction is created in the SimplePay system. The system sends back a URL as a response to the data received. The merchant needs to direct the customer to this URL in the browser.

d. At the URL provided, the customer lands on the SimplePay payment page where the transaction data provided earlier are displayed. The customer can provide his/her card details on the payment page. If he/she already has a card registered in the Simple application, it can be selected to complete the payment, too.
e. After the card details have been provided the payment is authorised by the bank (authorisation).
f. Following the authorisation, the customer is redirected to the merchant's website (**back**). Here it is necessary to inform the customer about the result of the transaction based on the returned data.
g. After this, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.
h. The IPN message includes:
- the expiry date of the registered card
- the partly concealed card number, where the last 4 digits can be read

1.5.2 Payment with a registered card using OneClick (in the presence of the card holder)

It is possible to start transactions with a registered card where no card details are needed for the payment

a. On the merchant's website the customer selects the items to be purchased and the total sum to be paid is calculated.
b. On the website the merchant enables the customer to provide the unique identifier (**cardSecret**) used earlier for the registration of the card. The merchant system is not permitted to save the cardSecret value, it may only forward it to SimplePay.
c. It transfers the purchase data along with the cardSecret value in the background to SimplePay via the API (**do**). Meanwhile, the customer stays on the webpage of the webstore, no redirection is necessary.
d. The SimplePay system **sends** customer data **to the bank for 3DS authentication.**
e. If **3DS authentication by the bank is unsuccessful**,
   - Authorisation is **not** performed at this point.
   - In this case, in its **do** request the SimplePay system returns a URL to the merchant to which it may redirect the customer (**challenge**).
   - If subsequently the merchant redirects the customer to the received URL, the customer is at a place where additional customer authentication required by the card issuing bank is performed.
   - If, after the above process, authentication followed by payment are successful, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.
f. If **3DS authentication by the bank is successful,** the SimplePay system identifies the stored card, following which payment is authorised by the bank (authorisation). Thus, in case of success **there is no challenge**, therefore the payment process can run in the background.
g. The SimplePay system sends a synchronous response to the **do** request about the result of the authorisation. This point is the logical equivalent of the redirection to the merchant website in case of regular payment (**back**).

h. After this, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.

1.6 Recurring payment

Recurring payments mean when a registered card can be used to initiate multiple payments without the presence of the card holder. This payment process is useful when the merchant wants to automatically initiate payments, such as collecting monthly fees.

The recurring process and the previously discussed OneClick process are different technically as well. For this reason, it is necessary to distinguish those registered card payments when the payment takes place in the presence of the card holder (oneclick) from the ones when it takes place in the absence of the card holder (recurring).

Recurring card registration can be performed in two ways according to the merchant's business logic.

- one-step, with immediate charge (e.g. as part of a purchase)
- two-step, without charge (registration is the sole purpose)

1.7 Recurring payment transaction process

1.7.1 One-step card registration process

In this case, a valid purchase and card charge takes place as well, and as a parameter of the transaction, the merchant also requests tokens from the SimplePay system. The tokens will allow the initiation of transactions later without the customer's presence.

3DS authentication is performed on the payment page during the first card registration transaction; the merchant system is only required to send the necessary customer data in the **"start"** request.

i. On the merchant's website the customer selects the items to be purchased and the total sum to be paid is calculated.

i. Before starting the transaction, the merchant must define (with the consent of the customer) the parameters of the requested tokens as follows:
   a. number of transactions (1-24), so many separate additional payments can be initiated on the registered card
   b. upper value limit, the maximum of the transaction amount that can be initiated later with the tokens
   c. expiration date, up to this deadline it is possible to initiate payment with the generated tokens.

j. The transaction data is transferred to SimplePay via the API (**start**), along with the token parameters. At this point the payment transaction is created in the SimplePay system. The system sends back a URL as a response to the data received. The merchant needs to direct the customer to this URL in the browser.

k. The generated tokens are provided in the response to the start request. The tokens will only be active after the successful authorisation. The tokens must be stored by the merchant to be used for future payments.
l. At the URL provided in Point **c.**, the customer lands on the SimplePay payment page where the transaction data provided earlier are displayed. In addition to the details of the purchase, also the token parameters specified in Point **b**. are displayed, so the customer can verify them. The customer can enter the details of his or her card on the payment page.
m. After the card details have been provided the payment is authorised by the bank (authorisation).
n. Following the authorisation, the customer is redirected to the merchant's website (**back**). Here it is necessary to provide information on the result of the authorisation based on the returned data.
o. After this, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.
p. The IPN message includes:
    a. the expiry date of the registered card
    b. the partly concealed card number, where the last 4 digits can be read
q. The requested tokens are activated at the same time as the IPN message. The tokens can be used for later payments with the card registered for the transaction.

1.7.2 Two-step card registration process

In this case, the authorisation of a symbolic amount, e.g. 100 HUF takes place, which gets only blocked on the customer's account, and if it was successful it gets immediately released. The purpose of the request is only to register and generate the tokens. Real purchases will be initiated by the merchant with the tokens received.

3DS authentication is performed on the payment page during the first card registration transaction; the merchant system is only required to send the necessary customer data in the **"start"** request.

The two-step payment must be active in the merchant account before using this feature. Knowledge of the logic of the two-step payments described in the referenced base documentation is essential for the development of this function.

a. On the merchant's website, the customer starts a transaction with a symbolic amount, specified by the merchant (e.g. 100 HUF).
b. Before starting the transaction, the merchant must define the parameters of the tokens as follows:
    a. number of transactions (1-24), so many separate additional payments can be initiated on the registered card
    b. upper value limit, the maximum of the transaction amount that can be initiated later with the tokens
    c. expiration date, up to this deadline it is possible to initiate payment with the generated tokens.
c. The transaction data is transferred to SimplePay via the API (**start**), along with the token parameters. At this point the payment transaction is created in the SimplePay system.

The system sends back a URL as a response to the data received. The merchant needs to direct the customer to this URL in the browser.

d. The generated tokens are provided in the response to the start request. The tokens will only be active after the successful authorisation. The tokens must be stored by the merchant to be used for future payments.
e. At the URL provided in Point **c.**, the customer lands on the SimplePay payment page where the transaction data provided earlier are displayed. In addition to the details of the purchase, also the token parameters specified in Point **b**. are displayed, so the customer can verify them. The customer can enter the details of his or her card on the payment page.
f. After the card details have been provided the payment is authorised by the bank (authorisation). **In the case of a two-step payment, the specified amount will only be blocked during the authorisation, but no charge will be made. To close the transaction (charge or release), additional interaction is required from the merchant.**
g. Following the authorisation, the customer is redirected to the merchant's website (**back**). Here it is necessary to provide information on the result of the authorisation based on the returned data.
h. After this, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). The IPN message notifies about the successful authorisation here. This IPN message is optional in the case of two-step payments.
i. The IPN message includes:
    a. the expiry date of the registered card
    b. the partly concealed card number, where the last 4 digits can be read
j. The requested tokens are activated at the same time as the IPN message. The tokens can be used for later payments with the card registered for the transaction.
k. After receiving the IPN message, the release (**finish**) can be started. The finish request closes the two-step transaction. If it is initiated with the amount of 0, the full blocked amount will be available again to the cardholder. The required amount of active tokens has been created by these, without placing a real charge on the customer's card.
l. The SimplePay system sends an IPN message to the merchant after the finish request as well, which contains the final amount of the transaction, which in this case is 0. This IPN message is not optional.

1.7.3 Simplified two-step card registration process

In this case, the merchant system can already specify in advance in the "**start**" request (**onlyCardReg**) that the purpose of the transaction is only card registration, so it will not want to debit the amount of the transaction.

In the case of a two-step payment, this means that after successfully reserving the required amount, the system will also automatically unlock the full amount, so the merchant system does not need to start this process in another API request.

The two-step card registration process described above is simpler in a was that the entire "**finish**" request can be omitted.

1.7.4 Recurrent payment with a registered card

At payments with a previously generated token it is irrelevant whether the registration was one or two-step. The only important thing at this point is to have a successful authorisation taken place during the registration transaction, resulting in the generated tokens being activated. At the same time, it is selectable whether the token-initiated payment starts in a one or two-step way.

a. The time has come for a payment in the merchant system, for example, to collect the monthly fee for a regular service. This will result in the full amount to be paid.

j. The merchant selects one of the tokens stored for the customer, which can be used to charge the saved card.

k. No successful purchase could have been made with the selected **token** before.

l. The purchase **amount** must comply with the limit specified at the token request.

m. The **date** of purchase must fall within the time limit specified at the token request.

n. The merchant transfers the purchase data, supplemented by the token, to SimplePay in the background, via the API (**dorecurring**).

o. In the case of an account set up for two-step payment, the dorecurring request can **optionally** be supplemented with the **twoStep** variable.
   - if the **twoStep** variable is **not passed**, the token payment will be executed as a two-step payment according to the account setup, i.e., it must be closed with a "**finish**" request.
   - also, if the **twoStep** variable is **passed**, and its value is **true**, the token payment will be executed as a two-step payment according to the account setup, i.e., it must be closed with a "**finish**" request.
   - however, if the **twoStep** variable is **passed**, and its value is **false**, the **full sum will be charged** after a successful authorisation, **without** the need for **a "finish" request**.

- The SimplePay system **sends** customer data **to the bank for 3DS authentication.**
- If **3DS authentication by the bank is unsuccessful**,
  - Authorisation is **not** performed at this point.
  - In this case, in its **dorecurring** request the SimplePay system returns a URL to the merchant to which it may redirect the customer (**challenge**).
  - If this was a completely automatic payment request, where the customer is not present, and it can therefore not be redirected, this payment process is terminated at this point **due to unsuccessful 3DS authentication**.
  - If the customer is present during payment, and the merchant can redirect it to the received URL, the customer is at a place where additional customer authentication required by the card issuing bank is performed.
  - If, after the above process, authentication followed by payment are successful, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.

r. If **3DS authentication by the bank is successful,** the SimplePay system identifies the stored card, following which payment is authorised by the bank (authorisation).

p. The SimplePay system sends a synchronous response to the **dorecurring** request about the result of the authorisation. This point is the logical equivalent of the redirection to the merchant website in case of regular payment (**back**).

q. After this, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**).
r. In the case of a one-step payment, this concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.
s. In the case of a two-step payment, the transaction must be closed by a "**finish**" request. The exception to this is the use of the above mentioned **twoStep** variable with the value **false**.
t. The token can be used for one successful payment. In case of success, it becomes used and no more payment can be started with it.

1.7.5 Tokens

The tokens generated in the start request must be stored in the merchant system. They can be used later to initiate transactions. Tokens have the following features:

- The tokens belong to a specific successful card registration SimplePay bank card payment.
- Transactions initiated with these tokens will charge the bank card used for that specific successful payment.
- The number of tokens for a given registration cannot be expanded. If additional tokens are needed, another card registration transaction need to be started.
- The card's registration is valid while
    - not all the tokens generated for it are used up
    - **and** the expiration date specified at the generation has not passed yet
    - **and** the validity of the registered bankcard has not expired
- Of the above two dates, always the closer one is the deadline for validity.
- The SimplePay system can start the authorisation on the token request if
    - the above eligibility criteria for the registration are met
    - the amount of the transaction does not exceed the upper limit value of the token
    - no successful transaction has taken place with the token yet
    - the transaction is started the proper way technically
- For a transaction initiated with a token, the bank will make the charge if the previously registered bank card, at the given moment
    - is valid (has not expired, has not been blocked, is not lost, etc.)
    - the amount specified in the transaction is available on the bank card
    - no settings of the bank regarding the usage of the bank card are exceeded (transaction limit; daily or monthly amount or transaction number limit, etc.)
- The generated tokens are equivalent
- The tokens can be used in any order
- Each token can only be used for a separate payment.
- The tokens cannot be combined with each other
- After a successful payment, the token will be inactive regardless of the amount charged.
- Only those tokens can be used to initiate a payment that has not been used for a successfully completed transaction before.
- After an unsuccessful payment (i.e., when no charge has taken place), the token can be used again to initiate a new payment
- Tokens cannot be modified
- By deactivating a card registration (**cardcancel**), all related tokens become inactive.
- Deactivation is irreversible. Unused tokens of a deactivated registration will no longer be eligible for payment.

- Payments initiated by a certain card can be queried (**cardquery**) even after its deactivation or expiration.

1.8 Payments based on Legacy card registration

In this way, the registration and payment processes can be technically simpler, but the card is not stored in SimplePay's own system, but in OTP Bank's system.

The location of the storage is important in that SimplePay can initiate transactions for authorisation to multiple recipients for cards stored in its own system, depending on which is currently available. In the case of cards saved in the banking system, authorisation can only take place in the Bank's system when making a payment.

1.8.1 Legacy oneclick registration

The first transaction involves the registration of the customer's card. **3DS** authentication is performed on the payment page during the first card registration transaction; the merchant system is only required to send the necessary customer data in the **"start"** request.

s. On the merchant's website the customer selects the items to be purchased and the total sum to be paid is calculated.
t. The merchant indicates **legacy** card storage using the **legacyCardRegister** variable for oneclick payments.
u. The merchant transfers the transaction data to SimplePay via the API (**start**). At this point the payment transaction is created in the SimplePay system. The system sends back a URL as a response to the data received. The merchant needs to direct the customer to this URL in the browser.
v. At the URL provided, the customer lands on the SimplePay payment page where the transaction data provided earlier are displayed. The customer can provide his/her card details on the payment page.
w. After the card details have been provided the payment is authorised by the bank (authorisation).
x. Following the authorisation, the customer is redirected to the merchant's website (**back**). Here it is necessary to inform the customer about the result of the transaction based on the returned data.
y. After this, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.
z. The IPN message includes:
- the expiry date of the registered card
- the partly concealed card number, where the last 4 digits can be read

1.8.2 Legacy oneClick payment

It is possible to start transactions with a registered card where no card details are needed for the payment

u. On the merchant's website the customer selects the items to be purchased and the total sum to be paid is calculated.

v. It transfers the purchase data in the background to SimplePay via the API (**do**). Meanwhile, the customer stays on the webpage of the webstore, no redirection is necessary.
w. Along with the purchase data, the merchant also passes the **type** field, which indicates whether the customer is present at the transaction.
x. The SimplePay system **sends** customer data **to the bank for 3DS authentication.**
y. If **3DS** authentication **by the bank is unsuccessful but the customer is present**
   - Authorisation is **not** performed at this point.
   - In this case, in its **do** request the SimplePay system returns a URL to the merchant to which it may redirect the customer (**challenge**).
   - If subsequently the merchant redirects the customer to the received URL, the customer is at a place where additional customer authentication required by the card issuing bank is performed.
   - If, after the above process, authentication followed by payment are successful, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.
z. If **3DS** authentication **by the bank is unsuccessful and the customer is not present**, the transaction ends here unsuccessfully, as in this case it is not possible to start a challenge process.
aa. If **3DS authentication by the bank is successful,** the SimplePay system identifies the stored card, following which payment is authorised by the bank (authorisation). Thus, in case of success **there is no challenge**, therefore the payment process can run in the background.
bb. The SimplePay system sends a synchronous response to the **do** request about the result of the authorisation. This point is the logical equivalent of the redirection to the merchant website in case of regular payment (**back**).
cc. After this, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.

1.8.3 Legacy recurring registration

The first transaction involves the registration of the customer's card. **3DS** authentication is performed on the payment page during the first card registration transaction; the merchant system is only required to send the necessary customer data in the **"start"** request.

In this case, **onlyCadrReg** using and two-step payment is also possible as detailed in the normal **recurring** process.

aa. On the merchant's website the customer selects the items to be purchased and the total sum to be paid is calculated.
bb. The merchant indicates **legacy** card storage using the **legacyCardRegister** variable for recurring payments.
cc. The merchant transfers the transaction data to SimplePay via the API (**start**). At this point the payment transaction is created in the SimplePay system. The system sends back a

URL as a response to the data received. The merchant needs to direct the customer to this URL in the browser.

dd. At the URL provided, the customer lands on the SimplePay payment page where the transaction data provided earlier are displayed. The customer can provide his/her card details on the payment page. If he/she already has a card registered in the Simple application, it can be selected to complete the payment, too.

ee. After the card details have been provided the payment is authorised by the bank (authorisation).

ff. Following the authorisation, the customer is redirected to the merchant's website (**back**). Here it is necessary to inform the customer about the result of the transaction based on the returned data.

gg. After this, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.

hh. The IPN message includes:

- the expiry date of the registered card
- the partly concealed card number, where the last 4 digits can be read
-

1.8.4 Legacy recurring payment

a. The time has come for a payment in the merchant system, for example, to collect the monthly fee for a regular service. This will result in the full amount to be paid.
b. It transfers the purchase data in the background to SimplePay via the API (**do**).
c. Along with the purchase data, the merchant also passes the **legacyRecurring** variable to indicate that it is a recurring transaction, and the type field to indicate that the customer is not present at the transaction.
d. The SimplePay system **sends** the purchase data **to the bank**
e. If **3DS** authentication **by the bank is unsuccessful and the customer is not present**, the transaction ends here unsuccessfully, as in this case it is not possible to start a challenge process.
f. If **3DS authentication by the bank is successful,** the SimplePay system identifies the stored card, following which payment is authorised by the bank (authorisation).
g. The SimplePay system sends a synchronous response to the **do** request about the result of the authorisation. This point is the logical equivalent of the redirection to the merchant website in case of regular payment (**back**).
h. After this, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.

1.9 Downloadable Resources

The basic resources necessary for the development can be downloaded from the merchant administration interface of both the sandbox and the live systems.

The following can be found in both systems in the menu on the left, under "**Downloads**"

- technical documentation

- sample code
- logo package
- financial resources

Documentation and sample code for payments are available at the following URL:

This documentation in Hungarian:
http://simplepartner.hu/download.php?target=v21cardstoragedochu

This documentation in English:
http://simplepartner.hu/download.php?target=v21cardstoragedocen

SDK sample code for Oneclick és Recurring payments:
http://simplepartner.hu/download.php?target=v21cardstoragesdk

Base documentation in Hungarian:
http://simplepartner.hu/download.php?target=v21dochu

Base documentation in English:
http://simplepartner.hu/download.php?target=v21docen

Base SDK sample code:
http://simplepartner.hu/download.php?target=v21sdk

2 Implementation of oneclick transactions

2.1 start - oneClick card registration

OneClick Payment allows customers to save their credit cards during the transaction. During the customer's next purchase, it will not be necessary to redirect him or her to SimplePay's payment page; it will be enough to start the transaction in the background. The benefit is that the customer will not be redirected, but will stay on the merchant's website the whole time, and payment will take place as a request sent in the background.

This way, payment is much faster and more convenient for the customer, because there is no need for a redirection to SimplePay's payment page, and to re-enter the card data for each purchase.

Customers saving their cards must be users registered in the merchant's system. This service cannot be provided for non-registered users!

The merchant must enable the customer to specify a unique identifier (**cardSecret**) before starting the payment.

The merchant must inform the customers that they will be able to start a transaction with the saved card for future oneClick payments only by entering the cardSecret.

The merchant system is not permitted to save the cardSecret value, it may only forward it to SimplePay.

CardSecret is also part of the encrypted card data that is stored. Since the cardSecret value is not stored by itself, there is no way to modify it. In case a change is required, the registered card must be deleted, and a new registration must be initiated.

Management of data relating to card registration can only take place through an encrypted SSL-certified connection.

The merchant system initiating the registration must use TLS 1.2 https protocol. Self-Sign SSL certification is not accepted.

Initiating a card registration transaction is a **start** request that is complemented by the cardSecret variable. In all other aspects (e.g. **3DS**) it is the same as the start function described earlier in the base documentation.

Hereinafter, the registration transaction is the same as described earlier, in the case of standard payment carried out in the browser.

The cardSecret is marked in red in the JSON example below. In all further examples, we will use the value "**SECRET**" as the content of **cardSecret**.

```
{
  "merchant":"PUBLICTESTHUF",
  "orderRef":"101010515408918334286",
  "customer":"v2 START Tester",
  "customerEmail":"sdk_test@otpmobil.com",
  "language":"HU",
  "currency":"HUF",
  "total":"15",
  "sdkVersion":"SimplePay_PHP_SDK_2.0.2_181002",
  "salt":"56176471ee827d50540e5907dd99f979",
  "methods":[
    "CARD"
  ],
  "invoice":{
    "name":"SimplePay V2 Tester",
    "company":"",
    "country":"hu",
    "state":"Budapest",
    "city":"Budapest",
    "zip":"1111",
    "address":"Address 1",
    "address2":"",
    "phone":"06203164978"
  },
  "timeout":"2018-10-30T09:40:33+00:00",
  "url":"http://localhost/v21/back.php",
  "cardSecret":"SECRET"
}
```


The response to the request, containing the payment link, is as follows:


```
{
  "salt":"gu3rMVuNGp5DMt5kXChmJJIQeKRZeEKA",
  "merchant":"PUBLICTESTHUF",
  "orderRef":"101010515408918334286",
  "currency":"HUF",
  "transactionId":99350378,
  "timeout":"2018-10-30T10:40:33+01:00",
  "total":15.0,
  "paymentUrl":"https://sandbox.simplepay.hu/pay/pay/pspHU/aoFtTKccntnBG6nfT2fORF2jd3iCAw1OwSn6A75b0jz9
XtjzBp"
}
```


2.2 OneClick Card registration statement

Prior to the card registration transaction, the customer must be informed, whereby **the customer must** explicitly **accept the card registration statement**. This statement does not replace the data transfer declaration, but is complementary to it in the event of a card registration transaction.

There are several options to place this statement
- in the site's own Terms and Conditions
- in the site's other documentation relating to payment
- displayed directly during card registration payments

It is not sufficient to place the declaration on the website if the customer does not encounter it and has not accepted it.

It may be accepted with a checkbox or by unambiguously indicating when the transaction is initiated that he/she accepts the declaration by initiating the payment.
In the highlighted section, you need to fill in the statement with the merchant's real information using the following method.

URL: the domain name specified in the contract, and in case of an application, the application name is also added.

2.2.1 Brochure in Hungarian

Recurring bank card payment (hereinafter: "Recurring payment") is a feature provided by SimplePay in relation to bank card acceptance, meaning that further payments may be initiated in the future without re-entering the bank card data, using the bank card data provided by the Customer during the registration transaction. One of the types of recurring payment, the so-called "ad-hoc payment" is made with the approval of the Customer during each transaction, so you have to approve all transactions in the case of each future payment. You will receive a notification of the successful payment through channels identical with conventional credit card payments in every case.

*By accepting this statement to use recurring payment, you allow to make subsequent payments made from your user account in this online store (**……[URL]……..**) without providing credit card details again after the successful registration transaction.*

Attention(!): Bank card data is handled in accordance with the card company policy. Neither the Merchant nor SimplePay has access to bank card information. The Merchant is directly responsible for any recurring payment transaction wrongly or unlawfully initiated, that is, all kind of claims against the Merchant's Payment Service Provider (SimplePay) are excluded.

I have read this brochure, I understand and accept its content.

2.2.2 Brochure in English

Recurring credit card payment (hereinafter referred to as Recurring payment) is a function included in the acceptance of credit cards provided by SimplePay meaning that in the future it is possible to make payments with credit card details provided by the Customer during the registration transaction without giving credit card details again. One of the types of recurring payment, the so-called "ad-hoc payment" is made with the approval of the Customer during each transaction, so you have to approve all transactions in the case of each future payment. You will receive a notification of the successful payment through channels identical with conventional credit card payments in every case.

*By accepting this statement to use recurring payment, you allow to make subsequent payments made from your user account in this online store (**……[URL]……..**) without providing credit card details again after the successful registration.*

Please note: the processing of credit card details is in accordance with the rules of card issuers. Neither the merchant nor SimplePay has access to the credit card data. The Merchant shall

assume direct liability for false or unauthorised recurring payments initiated by the Merchant, any claim enforcement against the Merchant's payment service provider (SimplePay) shall be unavailable.

I have read this notification, I take notice of its content and accept it.

2.3 Payment page in case of card registration

The payment page will differ from what you see during standard payment only in one aspect: in this case, the caption of the button initiating the payment will read "PAYMENT WITH CARD REGISTRATION".

2.4 ipn – in the case of a card registration

Receiving an IPN is the same as the IPN reception described earlier, because the IPN is independent of how the transaction was started.

However, at card storage, the message is expanded with the following fields:

- **expiry**: the expiry date of the registered card
- **cardMask**: the partly concealed card number, where the last 4 digits can be read


```
{
  "cardMask":"xxxx-xxxx-xxxx-4193",
  "salt":"ywj9Sn8KLKUct08EApxxQfKWgCSlZZxm",
  "orderRef":"101010515606836609132",
  "method":"CARD",
  "merchant":" PUBLICTESTHUF",
  "finishDate":"2019-06-16T13:15:05+02:00",
  "expiry":"2021-10-31T00:00:00+02:00",
  "paymentDate":"2019-06-16T13:14:21+02:00",
  "transactionId":99204917,
  "status":"FINISHED"
}
```


The IPN message contains the expiration date of the card the customer used to make the card registration payment. If this card has a shorter remaining validity than that set by the merchant at the token request, then also the tokens will only be active until that date, because it will not be possible to charge the card after its expiration.

2.5 do – initiating transaction with a saved card

The **do** request is expected by the API at the following URL:
https://sandbox.simplepay.hu/payment/v2/do

When using the "**do**" function you can initiate a payment transaction using a previously saved card. In this case, customers don't need to be redirected to the payment page because the payment is carried out with the card they previously specified and was stored in SimplePay' system. The API sends a synchronous response to requests sent with the POST method.

Before initiating a payment, the merchant enables the customer on the website to provide the unique identifier (cardSecret) registered for his or her card earlier. The transaction toward the

SimplePay API must be initiated using this data together with the SimplePay identifier of the registration transaction.

Thus, the two key data below are necessary for initiating a stored card transaction, so the system can identify the registered card.

One is the SimplePay identifier of the earlier card registration transaction (**cardId**). This piece of data can be stored in the merchant system.

The other piece of data necessary is the password (**cardSecret**) provided by the customer during the registration transaction. This data must be requested before the transaction starts, and cannot be stored.

The merchant system is not permitted to save the cardSecret value, it may only forward it to SimplePay.

2.6 do – data necessary for 3DS authentication

Due to **PSD2** compliance, cardholder identification, i.e. **3DS authentication**, is also necessary before stored card payments with the presence of the user (**Customer Initiated Transaction** or **CIT** in short).

In this case, too, the merchant's system needs to send data previously provided in the **"start"** request necessary for **3DS** authentication.

Beyond the above, initiation of the transaction requires three additional data important for 3DS:

One piece of data is the type of transaction (**type**) as to whether the customer is present or not during the initiation of the transaction. In case of a oneclick, this may only be a **"CIT"** for a **"do"** request, i.e. a transaction initiated in the presence of the customer. This piece of data is necessary for **3DS** authentication and is forwarded by the SimplePay system.

The other data are parameters of the customer's browser (**browser**). This piece of data is also part of **3DS** authentication, which is forwarded to the card issuing bank.

details of the **browser** array

- **accept**: Value of accept http header
- **agent**: Value of user-Agent http header
- **ip**: IP of browser source
- **java**: whether the browser supports java applets; in javascript: *navigator.javaEnabled()*
- **lang**: browser language; in javascript: *navigator.language*
- **color**: color depth of browser; in javascript: *screen.colorDepth*
- **height** = browser screen height; in javascript: *screen.height*
- **width** = browser screen width; in javascript: *screen.width*
- **tz** = browser time zone; in javascript: *new Date().getTimezoneOffset()*

Since the cardholder is present during the transaction (**CIT**), it is necessary to provide his/her browser parameters. If browser data are lacking, 3DS authentication by the bank may end with an error.

The **third** way of customer's registration **(threeDSReqAuthMethod**) in the merchant system is: possible values:

01: guest
02: registered with the merchant
05: registered with a third party ID (Google, Facebook, account, etc.)

In relation to a given bank card, the bank may compare data to similar data of transactions received through other online channels for the same card. For this reason, the sending of false data carries the risk that 3DS authentication ends with an error, i.e. payment may fail.

The sample code below shows the JSON string required to start the transaction.

```
{
 "salt":"911f00255b2c080dd710fdb48586f9b4",
 "orderRef":"101010515409076376148",
 "cardId":"99350681",
 "cardSecret":"SECRET",
  "type": "CIT",
  "threeDSReqAuthMethod": "02",
 "browser": {
   "accept": "*/*",
   "agent": "Chrome\/75.0.3770.142 Safari\/537.36",
   "ip": "10.0.0.50",
   "java": false,
   "lang": "en",
   "color": 24,
   "height": 1080,
   "width": 1920,
   "tz": -120
 },
  "customerEmail":"sdk_test@otpmobil.com",
  "invoice":{
    "name":"SimplePay V2 Tester",
    "company":"",
    "country":"hu",
    "state":"Budapest",
    "city":"Budapest",
    "zip":"1111",
    "address":"Address 1",
    "address2":"",
    "phone":"06203164978"
  },
 "merchant":"PUBLICTESTHUF",
 "currency":"HUF",
 "customer":"DO tester",
 "total":42,
 "sdkVersion":"SimplePay_PHP_SDK_2.0.2_181002"
}
```

2.7 do - with unsuccessful 3DS authentication

If **3DS authentication by the bank is unsuccessful**, authorisation is **not** yet performed.

The merchant's system needs to be prepared for when during the 3DS process a card issuing bank may require the cardholder to interactively identify himself/herself.

-In this case, in its **"do"** request the SimplePay system returns a URL (**redirectUrl**) to the merchant to which it may redirect the customer (**challenge**).

Reply to a **"do"** request in case of unsuccessful 3DS authentication

```
{
  "salt":"QnQtqSrVu01g331fNkXZBD2YusLzfG5U",
  "merchant":"PUBLICTESTHUF",
  "orderRef":"101010515409132276357",
  "currency":"HUF",
  "total":42.0
  "redirectUrl": "https://sandbox.simplepay.hu/pay/pay/ABCDEFG",
  "errorCodes": <errorCode>[]
}
```

2.8 do – challenge

If the merchant's system redirects the customer to the URL received in the **"redirectUrl"** field, the customer is at a place where additional customer authentication required by the card issuing bank is performed.

If authentication followed by payment are successful, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.

2.9 do – with successful 3DS authentication

After successful 3DS authentication by the bank, the SimplePay system identifies the stored card, following which the payment is authorised by the bank (authorisation). Thus, in case of success **there is no challenge**, therefore the payment process can run in the background.

The SimplePay system sends a synchronous response to the request with the result of the authorisation. This point is the logical equivalent of the redirection to the merchant website in case of regular payment (**back**).

Next, the fraud monitoring and prevention process is carried out in the background. If the system detects no issues, it sends a feedback to the website in the background (**ipn**). This concludes the payment transaction. After the IPN message has been received, the merchant can complete the order.

In addition to general data, the **transactionId** in the response contains the SimplePay transaction identifier of the payment.

```
{
  "salt":"QnQtqSrVu01g331fNkXZBD2YusLzfG5U",
  "merchant":"PUBLICTESTHUF",
  "orderRef":"101010515409132276357",
```

```
  "currency":"HUF",
  "transactionId":99077246,
  "total":42.0
}
```

2.10 do - debiting cards stored in API v1

Merchant systems that switch from API v1 to API v2 can still use previously saved cards. Cards previously saved in SimplePay using API v1 can also be debited with API v2.

Since card registration is not performed according to the method discussed in the API v2 documentation,
but a card previously registered in API v1 inthe SimplePay system will be used, a payment using the "**do**" endpoint can be initiated as follows:

- the ID saved using the SimplePay API v1 must be used to identify the saved card (**IPN_CC_TOKEN**)
- when initiating "do", variable "cardSect" **do not** need to be sent
- when initiating "do", the presence of the customer in the "**type**" variable must be indicated in the type field
- when initiating "**do**", it is necessary to send the ID of the previous card saved

using OTP VPOS in the "**cardId**" variable (**IPN_CC_TOKEN**)

A later section of this documentation discusses **legacy** card registration and use in both oneclick and recurring processes. Cards stored using API v1 and debited using API v2 actually have a legacy card debit where the card was saved on a non-API v2.
Therefore, it is recommended to read the chapter **"Implementation for legacy card storage and stored card payments"** of this document to get to know the whole legacy process.

The following example shows the minimum sample code for this case. Of course, this needs to be supplemented with the data detailed earlier in the description of the "**do**" request in order for the 3DS process to run properly as well.

```
{
   "salt":"ed6f6279378476e1394ba3db85ad08e2",
   "orderRef":"101010515894412749644",
   "customerEmail":"sdk_test@otpmobil.com",
   "merchant":"PUBLICTESTHUF",
   "currency":"HUF",
   "cardId":"19123456",
   "type":"CIT",
   "threeDSReqAuthMethod": "02",
   "methods":[
      "CARD"
],
   "total":25,
   "sdkVersion":"SimplePay_PHP_SDK_2.0.9_200130:eef9c5a27ee155e74c65a536f194ff53"
}
```

2.10.1 Live transactions

This solution is available in the **live** system as follows:

- the card was previously saved using the live SimplePay API v1
- a "**do**" request is initiated from a live SimplePay account
- the registration ID stored in the live SimplePay system is provided in the "**cardId**" field of the "**do**" request

The SimplePay system allows a card saved using API v1 to be debited by **the same merchant account** using API v2.

In case you want to start transactions on a separate account using API v1 and API v2, you need to create an additional live account. In this case, the merchant's current and new SimplePay accounts must be linked in the SimplePay live admin system to allow transition between the two accounts in case of stored cards.

In order to create and properly configure merchant accounts, please notify your sales contact before starting this API v2 development.

2.10.2 Test transactions

Cards previously saved on a live system cannot be accessed through the sandbox system.

In the SimplePay **sandbox** system, this feature can be tested as follows:

- the card was previously saved using the sandbox SimplePay API v1
- "**do**" request is initiated from a sandbox SimplePay account
- the registration ID stored in the sandbox SimplePay system is provided in the "**cardId**" field of the "**do**" request

The SimplePay system allows a card saved using API v1 to be debited by **the same merchant account** using API v2.

In case you want to start transactions on a separate account using API v1 and API v2, you need to create a sandbox account. In this case, the merchant's current and new SimplePay accounts must be linked in the SimplePay sandbox admin system to allow transition between the two accounts in case of stored cards.

In order to create and properly configure sandbox merchant accounts, please notify your sales contact before starting this API v2 development.

2.11 do - debiting cards stored in the OTP Bank Webshop (Middleware, MW) service

Merchant systems that switch from OTP Webshop VPOS to SimplePay API v2 can still use previously saved cards. Cards previously saved in SimplePay using the OTP webshop can also be debited with API v2.

Since card registration is not performed according to the method discussed in the API v2 documentation,
but a card previously registered in the OTP system will be used, a payment using the "**do**" endpoint can be initiated as follows:

- the ID saved in the OTP Webshop service must be used to identify the saved card (**isConsumerRegistrationID**)
- when initiating "do", variable "cardSect" **do not** need to be sent
- when initiating "do", the presence of the customer in the "**type**" variable must be indicated in the type field
- when initiating "**do**", it is necessary to send the ID of the previous card savedusing OTP VPOS(**isConsumerRegistrationID**) in the "**cardId**" variable

A later section of this documentation discusses **legacy** card registration and use in both oneclick and recurring processes. In the case of cards stored using the OTP Bank Webshop and debited using the API v2, there is actually a legacy card debit where the card was not saved on the API v2.
Therefore, it is recommended to read the chapter **"Implementation for legacy card storage and stored card payments"** of this document to get to know the whole legacy process.

The following example shows the minimum sample code for this case. Of course, this needs to be supplemented with the data detailed earlier in the description of the "**do**" request in order for the 3DS process to run properly as well.

```
{
   "salt":"ed6f6279378476e1394ba3db85ad08e2",
   "orderRef":"101010515894412749644",
   "customerEmail":"sdk_test@otpmobil.com",
   "merchant":"PUBLICTESTHUF",
   "currency":"HUF",
   "cardId":"0123456789",
   "threeDSReqAuthMethod": "02",
   "methods":[
      "CARD"
],
   "total":25,
   "sdkVersion":"SimplePay_PHP_SDK_2.0.9_200130:eef9c5a27ee155e74c65a536f194ff53"
}
```

2.11.1 Live transactions

This solution is available in the **live** system as follows:

- the card was previously saved in the live OTP Webshop service
- a "**do**" request is initiated from a live SimplePay account
- the registration ID stored in the live OTP system is provided in the "**cardId**" field of the "**do**" request

In the SimplePay live admin system, the merchant's current SimplePay
account must be linked to the merchant's previous OTP VPOS service. This opens up an active passage between
the two services.

In order for the live merchant accounts to be properly configured, indicate your need when concluding the contract!

2.11.2 Test transactions

Cards previously saved on a live system cannot be accessed through the sandbox system.

In the SimplePay **sandbox** system, this feature can be tested as follows:

- card storage takes place in the sandbox environment with the technical solution of OTP Webshop
- "**do**" request is initiated from a sandbox SimplePay account
- the registration ID stored in the sandbox system is provided in the "**cardId**" field of the "**do**" request

For this test, two merchant accounts need to be created in the SimplePay sandbox (test) system:

an account configured to the use of the OTP Webshop. A card registration transaction is required using this account. Its description from the merchant's point of view can be found in the following documentation:

http://simplepartner.hu/download.php?target=otpvposhu

- **another account that is configured to the use of the API v2.** The "**do**" request has to be initiated using this account as detailed above.

In the SimplePay sandbox system, the merchant's two SimplePay accounts must be linked. This opens up an active passage between the two services.

In order for the sandbox merchant accounts to be properly configured, indicate your need when concluding the contract!

2.12 cardquery – retrieve saved card transactions

The **cardquery** request is expected by the API at the following URL:
https://sandbox.simplepay.hu/payment/v2/cardquery

Transactions initiated with registered cards can be queried from the SimplePay sytem. Queries can be sent even if the customer deletes (inactivates) his or her card.

To initiate a query, only the SimplePay identifier of the card registration transaction (**cardId**) is necessary. This piece of data can be stored in the merchant system.

The sample code below shows the JSON string required to start the transaction.

```
{
  "salt":"a97c24f6900754bdced622db157a1579",
  "cardId":"99077245",
  "merchant":"PUBLICTESTHUF",
  "history":false,
  "sdkVersion":"SimplePay_PHP_SDK_2.0.2_181002"
```

```
}
```

In the response, the expiry time of the registration can be found in the **expiry** field, which indicates the latest possible date of use based on the card's expiration date.
The **status** field contains the validity status of the registration. If its value is "**ACTIVE**", further transactions can be initiated with it.
If this value is "**DISABLED**", then no further transaction can be initiated with it, even if it hasn't expired yet.


```
{
  "salt":"S6p1PJwd5pWvMALwhflIgQPSDgzpqG2a",
  "merchant":"PUBLICTESTHUF",
  "cardId":99077245,
  "status":"ACTIVE",
  "expiry":"2021-11-01T00:00:00+01:00"
}
```


The **history** parameter with the value **true** can be used to query transactions performed with a given card registration. If the value of "history" is **false** or the parameter is not included in the query, the API returns only the above basic data about the registration.


```
{
  "salt":"37026e8109998c70b6d550a3e9b1b083",
  "cardId":"99077245",
  "history":true,
  "merchant":"PUBLICTESTHUF",
  "sdkVersion":"SimplePay_PHP_SDK_2.0.2_181002"
}
```


In the extended response, however, the transactions initiated with the given registered card are included, in addition to the above data.


```
{
  "salt":"MrAWnFQjxjPdGLeD8q6s96sbmjJuFxMs",
  "merchant":"PUBLICTESTHUF",
  "cardId":99077245,
  "status":"ACTIVE",
  "expiry":"2021-11-01T00:00:00+01:00",
  "history":[
    {
      "orderRef":"101010515409132276357",
      "transactionId":99077246,
      "status":"FINISHED",
      "paymentDate":"2018-10-30T16:27:08+01:00",
      "finishDate":"2018-10-30T16:27:23+01:00"
    },
    {
      "orderRef":"101010515409293187568",
      "transactionId":99077248,
      "status":"FINISHED",
      "paymentDate":"2018-10-30T20:55:19+01:00",
      "finishDate":"2018-10-30T20:55:31+01:00"
    }
  ]
}
```

2.13 cardcancel – deletion of a stored card

The **cardcancel** request is expected by the API at the following URL:
https://sandbox.simplepay.hu/payment/v2/cardcancel

Holders of registered cards should be enabled to delete their cards in case they no longer wish to initiate further saved card payments on the given merchant's website.

The option to delete the card cannot be subjected to any conditions. Users registered on the merchant website must be able to initiate the card deletion from their profiles. The method of deletion must be clear and at any time easily accessible to the cardholder.

The option to delete is independent of what payments have been made before or what payments should be made later with the registered card.

The deletion of the card must be done not only in the merchant system but it must be also deactivated with the **cardcancel** request in the SimplePay system.

Transaction data can be queried even after inactivation, but cardcancel will prevent any additional transactions with this registration
Inactivation is a one-time action and cannot be reversed. If the customer wants to use the card again for oneClick payments, he or she must save it again during a new card registration payment.

To initiate a deletion (deactivation) only the SimplePay identifier of the card registration transaction (**cardId**) is necessary. This piece of data can be stored in the merchant system.

The sample code below shows the JSON string required to start the transaction.

```
{
  "salt":"e0592eaa4b704a85c4c3bbb4ee83323d",
  "cardId":"99077245",
  "merchant":" PUBLICTESTHUF",
  "sdkVersion":"SimplePay_PHP_SDK_2.0.2_181002"
}
```

The value of the status will be "**DISABLED**" in the response, as a result of the request. After that, no further registration payment can be started using that **cardId** (**do**), but data can still be queried (**cardquery**).

```
{
  "salt":"4CNlGqTIwmL4VQKZjrxu9Gp9VpfZ0XX0",
  "merchant":"PUBLICTESTHUF",
  "cardId":99077245,
  "status":"DISABLED",
  "expiry":"2021-11-01T00:00:00+01:00"
}
```

2.14 tokenquery - query token status

The **tokenquery** request is expected by the API at the following URL:
https://sandbox.simplepay.hu/payment/v2/tokenquery

The API provides an opportunity to query the status of the unique token.

The token becomes active when a successful authorisation occurs during the card registration transaction. The token becomes inactive when

- a successful transaction occurs using it
- the merchant inactivates the token (**tokencancel** request)
- the merchant inactivates the card (**cardcancel** request)
- the card expires and automatically becomes inactive in the SimplePay system

The following data are required to be sent in the request.
In the "**token**" field, enter the specific token generated during the "**start**" request whose status you want to query.

```
{
   "token":"SPT2N8MLJ9P2VJ4FUC3X224UIED627RTQKORI3PMKPP6PVD7UEF99JF7NFNIGP5X",
   "merchant":"LRMIPSMHUF",
   "salt":"2480a8fb1efe5042d8d9ce8c22a957f4",
   "sdkVersion":"SimplePay_PHP_SDK_2.0.9_200130:3d383a4becf19088c05cc871fab5b1d7"
}
```

Answer to the tokenquery request

```
{
   "salt":"pXCFBePh3JVLCcf5BFf9JfsbBn0mqFiK",
   "merchant":"LRMIPSMHUF",
   "token":"SPT2N8MLJ9P2VJ4FUC3X224UIED627RTQKORI3PMKPP6PVD7UEF99JF7NFNIGP5X",
   "status":"active",
   "expiry":"2020-12-01T17:00:00+01:00"
}
```

In the response, the status field contains the current status of the given token.
Possible values:

- active: it is possible to initiate a transaction with the given token
- used: the given token has already been used for a successful payment
- canceled: the given token has been inactivated (with a **tokencancel** request) or the registration fee

2.15 tokencancel - inactivate token

The **tokencancel** request is expected by the API at the following URL:
https://sandbox.simplepay.hu/payment/v2/tokencancel

The API provides an opportunity to inactivate the unique token.

The following data are required to be sent in the request.
In the "**token**" field, enter the specific token generated during the "**start**" request that should be inactivated.

```
{
   "token":"SPT2N8MLJ9P2VJ4FUC3X224UIED627RTQKORI3PMKPP6PVD7UEF99JF7NFNIGP5X",
   "merchant":"LRMIPSMHUF",
   "salt":"2480a8fb1efe5042d8d9ce8c22a957f4",
   "sdkVersion":"SimplePay_PHP_SDK_2.0.9_200130:3d383a4becf19088c05cc871fab5b1d7"
}
```


Answer to the tokencancel request


```
{
   "salt":"DdH4qbkIcqpfCKsItpkGK5UfhE1sH9PX",
   "merchant":"LRMIPSMHUF",
   "token":"SPT2N8MLJ9P2VJ4FUC3X224UIED627RTQKORI3PMKPP6PVD7UEF99JF7NFNIGP5X",
   "status":"cancelled",
   "expiry":"2022-11-01T00:00:00+01:00"
}
```


In the response, the status field contains the current status of the given token, which is cancelled in this case.

3 Usage of PHP SDK for oneclick transactions

Setup and usage of the PHP SDK is discussed in detail in the base documentation. To use the card storage functions (**oneclick/recurring**), the basic SDK has to be supplemented with the following downloadable sample code kit:
http://simplepartner.hu/download.php?target=v21cardstoragesdk

Contents of the supplementary kit have to be extracted to the SDK main directory.

The package also contains the items needed for **oneclick** and **recurring** card storage.
Of these, the followings are not required for the **oneclick** discussed in this chapter. Their presence does not cause any problem, but they can be deleted as well.

- recurring folder
- dorecurring.php
- src/SimplePayV21TokenData.php

Compared to the basic setup, no additional steps are required.

3.1 start – card storage

The **start** function can be also used to initiate a card registration payment.

The operating logic of the start request for a card registration process can be found in Chapter **start - oneClick card registration**. Hereinafter, its implementation using PHP SDK is described in this chapter.

Sample code in the SDK:

- **indexstorage.html** selection of the card storage mode
- **startstorage.php** execution of the card storage

Initiating a oneclick card registration transaction is a **start** request that is complemented by the **cardSecret** variable described in Chapter **2.1**. In all other aspects, it is the same as the "start" function described in the base documentation.

The sample code also contains sample code required for initiating a oneclick or a recurring process.

In the sample code the variable cardSecret is loaded with a preset value (SECRET) for oneclick. ***Instead of this***, you need to ask the customer for this information and send it in the cardSecret variable.

```
$trx->addData('cardSecret', 'SECRET');
```

3.2 ipn – in the case of a card registration

Receiving an IPN is the same as the IPN reception described earlier, because the IPN is independent of how the transaction was started.

However, at card storage, the message is expanded with the following fields:

- **expiry**: the expiry date of the registered card
- **cardMask**: the partly concealed card number, where the last 4 digits can be read

3.3 do – payment by a stored card

Using the **do** function, you can initiate a **oneclick** transaction using a registered card.

The operating logic of the do request can be found in Chapter **do – initiating transaction with a saved card**. Hereinafter, its implementation using PHP SDK is described in this chapter.

Sample code in the SDK: **do.php**
For transactions started with a registered card, the src/SimplePayV21CardStorage.php file is also required in addition to the file (src/SimplePayV21.php) containing the basic SDK classes. This includes additional functions needed for card storage.

```
//Import config data
require_once 'src/config.php';

//Import SimplePayment class
require_once 'src/SimplePayV21.php';

//Import SimplePayV21CardStorage class
require_once 'src/SimplePayV21CardStorage.php';

$trx = new SimplePayDo;

$trx->addConfig($config);

//account
$trx->addConfigData('merchantAccount', $_REQUEST['merchant']);

//OneClick transaction
$trx->addData('cardId', $_REQUEST['cardId']);

//cardSecret
$trx->addData('cardSecret', 'SECRET');

//threeDSReqAuthMethod
$trx->addData('threeDSReqAuthMethod', '02');

//add merchant transaction ID
$trx->addData('orderRef', '101010515537887919364');

// METHODS
$trx->addData('methods', array('CARD'));

// add currency
$trx->addData('currency', 'HUF');

//ORDER PRICE/TOTAL
$trx->addData('total', '42');
```

```
//customer's name
$trx->addData('customer', 'DO tester');

//customer's email
$trx->addData('customerEmail', 'sdk_test@otpmobil.com');

//start do
$trx->runDo();
```

Response to the request

```
Array
(
    [responseBody] => {"salt":"3odfGqZG5SbvvKSuWx0f9gggWpyajRYZ","merchant":"LRMIPSMHUF","orderRef":"10
10105155378879193 64","currency":"HUF","transactionId":99130180,"total":42.0}
    [responseSignature] => 2H9d7XPObSakFABMihoNmn08sSmty4HgoIxkXxu5Qu249lWoU3pmW9fwy7JjNd23
    [responseSignatureValid] => 1
    [salt] => 3odfGqZG5SbvvKSuWx0f9gggWpyajRYZ
    [merchant] => PUBLICTESTHUF
    [orderRef] => 101010515537887919364
    [currency] => HUF
    [transactionId] => 99130180
    [total] => 42
)
```

3.4 do - load cards stored in other services (API v1 or OTP Webshop)

Requesting API v2 **do** is suitable for debiting cards previously stored using **API v1** or the **OTP Bank Webshop VPOS**.

Detailed information on debiting cards saved on other systems can be found in the following sections of the present documentation:
API v1: do - API v2 payment with a card stored using API v1
OTP Webshop: do - API v2 payment with a card stored in OTP Webshop (Middleware, MW) service

For a **do** request initiated with SDK must be modified as follows:

- when initiating "do", variable "cardSect" **do not** need to be sent
- when initiating "**do**", it is necessary to send the ID of the card previously registered using API v1 or OTP VPOS in the "**cardId**" variable

For API v1, the value of **IPN_CC_TOKEN** must be sent in the **cardId** field.
In the case ofOTP Webshop VPOS the value of**isConsumerRegistrationID** has to be sent in the **cardId** field.

```
$trx->addData('cardId', 'abc123');
```

It is not necessary to change the other parameters of the **do** request.

3.5 cardquery – querying a stored card

With the **cardquery** function you can query the transactions initiated using a registered card.

The operating logic of the cardquery request can be found in Chapter 2.12. Hereinafter, its implementation using PHP SDK is described in this chapter.

Sample code in the SDK: **cardquery.php**

```
//Import config data
require_once 'src/config.php';

//Import SimplePayment
require_once 'src/SimplePayV21.php';

//Import SimplePayV21CardStorage
require_once 'src/SimplePayV21CardStorage.php';

$trx = new SimplePayCardQuery;

//config
$trx->addConfig($config);

//account
$trx->addConfigData('merchantAccount', 'PUBLICTESTHUF');

//card ID
$trx->addData('cardId', '99123456');

$trx->runCardQuery();
```

Response to the request

```
Array
(
    [responseBody] => {"salt":"AtIpjA808vdA4kZadQFOXEvbOGAvdLtK","merchant":"LRMIPSMHUF","cardId":99130
179,"status":"ACTIVE","expiry":"2021-11-01T00:00:00+01:00"}
    [responseSignature] => xuMWvPKd5rj61hZYJ7MmkPb7/UYp2sqYlgi0iRl0g0j3ItskUYvPt+c+y1RHIojm
    [responseSignatureValid] => 1
    [salt] => AtIpjA808vdA4kZadQFOXEvbOGAvdLtK
    [merchant] => PUBLICTESTHUF
    [cardId] => 99130179
    [status] => ACTIVE
    [expiry] => 2021-11-01T00:00:00+01:00
)
```

The request can be supplemented with the "**history**" variable, which is used to retrieve payments from the system that were made with the stored card.

```
$trx->addData('history', true);
```

In this case, the response is supplemented by the array "history" containing the transactions.

```
Array
```

```
(
    [responseBody] => {"salt":"BsRupCoHXHSS0zCK11kFUVQ0SAOglXQx","merchant":"LRMIPSMHUF","cardId":99130
179,"status":"ACTIVE","expiry":"2021-11-01T00:00:00+01:00","history":[{"orderRef":"10101051553788791936
4","transactionId":99130180,"status":"FINISHED","paymentDate":"2019-03-28T16:59:52+01:00","finishDate":
"2019-03-28T17:00:03+01:00"}]}
    [responseSignature] => +Npu87rISjLb/UJFdLhEJhiyDEamkQxjonahHTL8XFI4mOEVKrM3a78viEqokHXv
    [responseSignatureValid] => 1
    [salt] => BsRupCoHXHSS0zCK11kFUVQ0SAOglXQx
    [merchant] => PUBLICTESTHUF
    [cardId] => 99130179
    [status] => ACTIVE
    [expiry] => 2021-11-01T00:00:00+01:00
    [history] => Array
        (
            [0] => Array
                (
                    [orderRef] => 10101051553788791936 4
                    [transactionId] => 99130180
                    [status] => FINISHED
                    [paymentDate] => 2019-03-28T16:59:52+01:00
                    [finishDate] => 2019-03-28T17:00:03+01:00
                )

        )

)
```

3.6 cardcancel – deletion of a stored card

Using the **cardcancel** function, you can deactivate a registered card.

The operating logic of the cardcancel request can be found in Chapter 2.13. Hereinafter, its implementation using PHP SDK is described in this chapter.
Sample code in the SDK: **cardcancel.php**

```php
//Import config data
require_once 'src/config.php';

//Import SimplePayment class
require_once 'src/SimplePayV21.php';

//Import SimplePayV21CardStorage
require_once 'src/SimplePayV21CardStorage.php';

$trx = new SimplePayCardCancel;

//config
$trx->addConfig($config);

//account
$trx->addConfigData('merchantAccount', 'PUBLICTESTHUF');

//add card ID
$trx->addData('cardId', '99123456');

//start cancel
$trx->runCardCancel();
```

Response to the request

```
Array
(
    [responseBody] => {"salt":"ATGG2rd6mZF0FzDpC0SG6lezldHEmUcS","merchant":" PUBLICTESTHUF ","cardId":
99123456,"status":"DISABLED","expiry":"2022-11-01T00:00:00+01:00"}
    [responseSignature] => JTYj5JWN8Ozmc/n61yqWGo4PSHcNrclsAxqLfuF/NG06DOAmMwQ0Sjqutx0sbGHA
    [responseSignatureValid] => 1
    [salt] => ATGG2rd6mZF0FzDpC0SG6lezldHEmUcS
    [merchant] => PUBLICTESTHUF
    [cardId] => 99123456
    [status] => DISABLED
    [expiry] => 2022-11-01T00:00:00+01:00
)
```

In the response, the status of the card registration needed to become DISABLED (marked in red) as the result of the request.

4 Implementation of recurring transactions

The transaction data must be sent by POST method in a JSON string as described in the referenced documentation's "**General message format**" chapter.

Request authentication is discussed in the referenced documentation's "**Validating messages**" chapter.

4.1 start – with recurring card registration

In the case of recurring payments, the merchant's system receives a specified number of tokens from the SimplePay system during card registration. In subsequent payments, these tokens can be used to initiate transactions.

This enables the automation of payments, initiating them without the presence of the user.

Customers saving their cards must be users registered in the merchant's system. This service cannot be provided for non-registered users!

The merchant must inform the customer before the payment is started that his or her card will be stored. In addition, the merchant must also inform the customer on the following details of the tokens to be generated for the stored card:

- how many tokens will be created
- the upper limit of payment amounts these can be used for in the future
- what will be the expiration date of the tokens, i.e., how long they can be used

Management of data relating to card registration can only take place through an encrypted SSL-certified connection.

The merchant system initiating the registration must use TLS 1.2 https protocol. Self-signed certification is not accepted.

A card registration transaction is initiated by a **start** request that is complemented by the variable **recurring** containing the parameters (number of tokens, maximum amount, expiration date) of the requested tokens, as follows:

- **times**: the number of times the merchant wants to charge the customer's card. A corresponding number of tokens will be generated as a result of the request
- **until**: the end date of the tokens' validity
- **maxAmount**: the upper limit of the amount each transaction can be initiated with

In all other aspects (e.g. **3DS**), the registration transaction is identical to the normal **start** function previously described in the aforementioned documentation. In the JSON sample below, the supplementation related to the recurring function is highlighted in red.

Request initiation

```
"merchant":"PUBLICTESTHUF",
"orderRef":"12700115510197967678",
```

```
  "customer":"v2 SimplePay Test",
  "customerEmail":"sdk_test@otpmobil.com",
  "language":"HU",
  "currency":"HUF",
  "sdkVersion":"SimplePayV2.1_PHP_SDK_2.0.4_180221",
  "salt":"c71e720d4a4f36ce6858cae698923cb1",
  "methods":[
    "CARD"
  ],
  "recurring":{
    "times":3,
    "until":"2020-12-01T18:00:00+02:00",
    "maxAmount":50
  },
  "threeDSReqAuthMethod": "02",
  "total":50,
  "invoice":{
    "name":"SimplePay V2 Tester",
    "company":"",
    "country":"hu",
    "state":"Budapest",
    "city":"Budapest",
    "zip":"1111",
    "address":"Address 1",
    "address2":"",
    "phone":"06203164978"
  },
  "timeout":"2019-02-24T15:59:56+01:00",
  "url":"http://localhost/v21/backrecurring.php",
}
```


Response to the request


```
{
  "salt":"jHseNFc16LUms0MFpHg0rkNQRgNNlbTd",
  "merchant":"PUBLICTESTHUF",
  "orderRef":"12700115510207958612",
  "currency":"HUF",
  "transactionId":99123339,
  "timeout":"2019-02-24T16:16:35+01:00",
  "total":25.0,
  "paymentUrl":"https://tsecurepay.simplepay.hu/pay/pay/pspHU/MyVrQUHeja9LQT6OwbuNItmTTpsbkQ4WwSNcMm8vA
X9KhbRmAH",
  "tokens":[
    "SPTB7NI5GE2WNM55DPKIA27H2QEVCVSVACQCS2VHCW9PVTTIWOVMWPLUSCAQ22SV",
    "SPTDAJMDGTOGMIAR6OGQU23IVBWFDCWL7XM542UUS2GJTH7K8CVQ6CE739BA763O",
    "SPT4CM2QVQ9GN2XNW2HJQ22TTONVDLIJVF4KO2RIQX94NJBR8GVLDO5526DA5E5F"
  ]
}
```


The generated tokens can be found in the tokens field of the response. At this point the tokens are not active yet. To activate the tokens, the transaction must be started as previously described at the start function. If the transaction will be successful, the tokens will become active at the same time as the IPN message arrives.

4.2 Two-step start - recurring with card registration

The technical possibility of performing two-stage transactions can be requested at the conclusion of the contract, and also can be required later at SimplePay's IT support. Once this

is done, the merchant system can indicate by the value of the **twoStep** variable for each transaction whether it wants to initiate a one-step or a two-step payment.

In the event that the purpose of the transaction is only card registration, so the merchant does not want to charge the amount of the transaction, the process can be simplified in the case of a two-step account. In this case, the merchant system can already specify in advance in the "**start**" request with the variable "**onlyCardReg**" that the purpose of the transaction is only card registration, so it will not want to debit the amount of the transaction.

```
"onlyCardReg":true,
```

In the case of a two-step payment, this means that after successfully reserving the required amount, the system will also automatically unlock the full amount, so the merchant system does not need to start a "**finish**" request in another API request.

4.3 Recurring Card registration statement

Prior to the card registration transaction, the customer must be informed, whereby the customer must explicitly accept the card registration statement. This statement does not replace the data transfer declaration, but is complementary to it in the event of a card registration transaction.

There are several options to place this statement

- in the site's own Terms and Conditions
- in the site's other documentation relating to payment
- displayed directly during card registration payments

It is not sufficient to place the declaration on the website if the customer does not encounter it and has not accepted it.

It may be accepted with a checkbox or by unambiguously indicating when the transaction is initiated that he/she accepts the declaration by initiating the payment. In the highlighted section, you need to fill in the statement with the merchant's real information using the following method.

URL: the domain name specified in the contract, and in case of an application, the application name is also added.

4.3.1 Brochure in Hungarian

Recurring bank card payment (hereinafter: "Recurring payment") is a feature provided by SimplePay in relation to bank card acceptance, meaning that further payments may be

initiated in the future without re-entering the bank card data, using the bank card data provided by the Customer during the registration transaction.

By accepting this Statement about Recurring Payment, you agree that after the successful registration, in this webshop (……[URL]……..) without providing credit card details and you allow for the Merchant to make the payment without your transactional approval.

Attention(!): Bank card data is handled in accordance with the card company policy. Neither the Merchant nor SimplePay has access to bank card information.

The Merchant is directly responsible for any recurring payment transaction wrongly or unlawfully initiated, that is, all kind of claims against the Merchant's Payment Service Provider (SimplePay) are excluded.
I have read this brochure, I understand and accept its content.

4.3.2 Brochure in English

Recurring credit card payment (hereinafter referred to as Recurring payment) is a function included in the acceptance of credit cards provided by SimplePay meaning that in the future it is possible to make payments with credit card details provided by the Customer during the registration transaction without giving credit card details again.
By accepting this statement to use Recurring payment you allow to make subsequent payments made from your user account in this online store (……[URL]……..) without providing credit card details and you allow for the Merchant to make the payment without your transactional approval.

Please note: the processing of credit card details is in accordance with the rules of card issuers. Neither the merchant nor SimplePay has access to the credit card data.

The Merchant shall assume direct liability for false or unauthorised recurring payments initiated by the Merchant, any claim enforcement against the Merchant's payment service provider (SimplePay) shall be unavailable.
I have read this notification, I take notice of its content and accept it

4.4 Payment page

The payment page differs from what you see in normal payments as follows:

- in this case, the caption of the button initiating the payment will read "PAYMENT WITH CARD REGISTRATION"
- the parameters of the tokens requested by the start communication (number of tokens, maximum amount and expiration date) are displayed

4.5 Back

Following the authorisation, the customer is redirected to the merchant's website as previously described in the referenced documentation. In the case of recurring payments, there is no difference in the parameters being sent and their processing from those described above.

4.6 ipn – in the case of a card registration

Receiving an IPN is the same as the IPN reception described earlier, because the IPN is independent of how the transaction was started.

However, at card storage, the message is expanded with the following fields:

- **expiry**: the expiry date of the registered card
- **cardMask**: the partly concealed card number, where the last 4 digits can be read


```
{
  "cardMask":"xxxx-xxxx-xxxx-4193",
  "salt":"ywj9Sn8KLKUct08EApxxQfKWgCSlZZxm",
  "orderRef":"101010515606836609132",
  "method":"CARD",
  "merchant":" PUBLICTESTHUF",
  "finishDate":"2019-06-16T13:15:05+02:00",
  "expiry":"2021-10-31T00:00:00+02:00",
  "paymentDate":"2019-06-16T13:14:21+02:00",
  "transactionId":99204917,
  "status":"FINISHED"
}
```


The IPN message contains the expiration date of the card the customer used to make the card registration payment. If this card has a shorter remaining validity than that set by the merchant at the token request, then also the tokens will only be active until that date, because it will not be possible to charge the card after its expiration.

4.7 dorecurring – payment initiation using a token

The API expects the **dorecurring** request at the following URL:
https://sandbox.simplepay.hu/payment/v2/dorecurring

By using the "**dorecurring**" function you can initiate a payment transaction using a previously saved card. In this case, customers don't need to be redirected to the payment page because the payment is carried out with the card they previously specified and was stored in SimplePay' system. The API sends a synchronous response to requests sent with the POST method.

The request is basically the same as the "**do**" request used for "**oneclick**" payments. The only difference is that one of the previously generated tokens should be sent in the variable "**token**" instead of the "**cardSecret**" variable.

4.8 dorecurring – data necessary for 3DS authentication

Due to **PSD2** compliance, cardholder identification, i.e. **3DS authentication**, is also necessary before stored card payments **without the presence of the customer** initiated by the merchant (**Merchant Initiated Transaction** or **MIT** in short).

When initiating a transaction, it is necessary to send one of the previously generated **tokens**, which has not been used yet

Beyond the above, initiation of the transaction requires two additional data important for **3DS**:

One piece of data is the type of transaction (**type**), which is important as to whether the customer is present or not during the initiation of the transaction. In the case of token payment in the "**dorecurring**" request, it can be "**MIT**" because the transaction was initiated **without the presence of the customer** (recurring)
This piece of data is necessary for **3DS** authentication and is forwarded by the SimplePay system.

The second way of customer's registration **(threeDSReqAuthMethod**) in the merchant system is: possible values:

01: guest
02: registered with the merchant
05: registered with a third party ID (Google, Facebook, account, etc.)

In relation to a given bank card, the bank may compare data to similar data of transactions received through other online channels for the same card. For this reason, the sending of false data carries the risk that 3DS authentication ends with an error, i.e. payment may fail.

Request initiation in relation to **"MIT"**

```
{
  "salt":"776748178d05236f40839a4ecd0755dc",
  "orderRef":"101010515510244751578",
  "customerEmail":"sdk_test@otpmobil.com",
  "merchant":"PUBLICTESTHUF",
  "currency":"HUF",
  "customer":"Dorecurring tester",
  "token":"SPTB7NI5GE2WNM55DPKIA27H2QEVCVSVACQCS2VHCW9PVTTIWOVMWPLUSCAQ22SV",
  "type": "MIT",
  "threeDSReqAuthMethod": "02",
  "methods":[
    "CARD"
  ],
  "total":42,
  "invoice":{
    "name":"SimplePay V2 Tester",
    "company":"",
    "country":"hu",
    "state":"Budapest",
    "city":"Budapest",
    "zip":"1111",
    "address":"Address 1",
    "address2":"",
    "phone":"06203164978"
  },
  "sdkVersion":"SimplePay_PHP_SDK_2.0.4_180221:"
}
```

4.9 cardquery – retrieve stored card transactions

With the **cardquery** request you can query the transactions initiated using a card registered in the system. This is done the same way it is described at the oneclick payment in Chapter "cardquery – retrieve saved card transactions".

4.10 cardcancel – deletion of a stored card

With the **cardcancel** request you can delete (deactivate) a card registered in the system. This is done the same way it is described at the oneclick payment in Chapter "cardcancel – deletion of a saved card".

5 Usage of PHP SDK for recurring transactions

Setup and usage of the PHP SDK is discussed in detail in the base documentation. To use the card storage functions (**oneclick/recurring**), the basic SDK has to be supplemented with the following downloadable sample code kit:
http://simplepartner.hu/download.php?target=v21cardstoragesdk

Contents of the supplementary kit have to be extracted to the SDK main directory.

The package also contains the items needed for **oneclick** and **recurring** card storage.
Of these, the following is not required for the **recurring** payment method discussed in present chapter. Its presence does not cause any problem, but it can be deleted as well.

- do.php

You need to grant write permission to the "**recurring**" folder included in the supplementary kit because the tokens created by the SDK are stored there.

Compared to the basic SDK, no additional steps are required.

The "recurring" folder and in it the tokens stored in the file are created for demonstrational purposes. This method only serves to enable the SDK to illustrate the operational logic of recurring payment.

In live mode, for real recurring transactions, database storage is recommended instead of this storage mode.

5.1 start

Sample code in the SDK: **startstorage.php**
The request initiation can be implemented as follows:

The sample code also contains sample code required for initiating a oneclick or a recurring process.

In the case of the recurring function, the **start** function discussed above must be supplemented with the "recurring" elements needed for token generation. All other parts of the request are the same as described above.


```
{
   "salt":"db35ed9444f04179b1e665db8552d858",
   "merchant":"LRMIPSMHUF",
   "orderRef":"101010515510319702312",
   "currency":"HUF",
   "sdkVersion":"SimplePay_PHP_SDK_2.0.9_200130:dc597528c6fb3b4c6312d470ed28895e",
   "methods":[
      "CARD"
   ],
   "recurring":{
      "times":3,
      "until":"2020-12-01T18:00:00+02:00",
      "maxAmount":50
   },
   "total":"25",
   "threeDSReqAuthMethod":"02",
   "customerEmail":"sdk_test@otpmobil.com",
```

```
   "language":"HU",
   "timeout":"2020-08-16T12:49:27+00:00",
   "url":"http:\/\/payusdk.simplelabs.hu\/nandi\/v21_200814\/backstorage.php",
   "invoice":{
      "name":"SimplePay V2 Tester",
      "country":"hu",
      "state":"Budapest",
      "city":"Budapest",
      "zip":"1111",
      "address":"Address 1",
      "address2":"",
      "phone":"06203164978"
   }
}
```

The response to the request is located in the array found in the $trx->transaction variable. Within this, the array "tokens" contains the generated tokens for subsequent transactions.

```
Array
(
    [responseBody] => {"salt":"016u2eCj7v1nvP2wGGlwl7cP1n0SmRsy","merchant":"PUBLICTESTHUF","orderRef":
"101010515510319702312","currency":"HUF","transactionId":99123344,"timeout":"2019-02-24T19:22:50+01:00"
,"total":25.0,"paymentUrl":"https://tsecurepay.simplepay.hu/pay/pay/pspHU/ffWNRjXbQaZtxvZQnexbiFkzOFJnv
A8KxSJ1o1nfg0S6ejnDBG","tokens":["SPT9KF5FG8VQX944PXPM227HHFJ253O778EKS2XSQ6KX5GRQAKWELCTH9VOQXKML","SP
T96TF79WTQXG7FFN2O824RI5D25BPCWCSWK2RAJDEM72L3BGWCU6MXW35B36IU","SPT582Q3H7HQWI6GEFEW628LPVC247DAVL3QE2
S7UMC99WXW9CW84SUXLH7B5NHI"]}
    [responseSignature] => Pjeetn6DvKWLIPfq/nd30irXIIfWX7kHe3e4Fh61Ue4raXoS4dUyHT9ZwBg85GlT
    [responseSignatureValid] => 1
    [salt] => 016u2eCj7v1nvP2wGGlwl7cP1n0SmRsy
    [merchant] => PUBLICTESTHUF
    [orderRef] => 101010515510319702312
    [currency] => HUF
    [transactionId] => 99123344
    [timeout] => 2019-02-24T19:22:50+01:00
    [total] => 50
    [paymentUrl] => https://sandbox.simplepay.hu/pay/pay/pspHU/ffWNRjXbQaZtxvZQnexbiFkzOFJnvA8KxSJ1o1nf
g0S6ejnDBG
    [tokens] => Array
        (
            [0] => SPT9KF5FG8VQX944PXPM227HHFJ253O778EKS2XSQ6KX5GRQAKWELCTH9VOQXKML
            [1] => SPT96TF79WTQXG7FFN2O824RI5D25BPCWCSWK2RAJDEM72L3BGWCU6MXW35B36IU
            [2] => SPT582Q3H7HQWI6GEFEW628LPVC247DAVL3QE2S7UMC99WXW9CW84SUXLH7B5NHI
        )
)
```

5.2 back - communication

The information on the merchant's web page is the same as the back information described earlier. There is no need to display additional information here for card registration.

5.3 ipn – in the case of a card registration

Receiving an IPN is the same as the IPN reception described earlier, because the IPN is independent of how the transaction was started.

However, at card storage, the message is expanded with the following fields:

- **expiry**: the expiry date of the registered card
- **cardMask**: the partly concealed card number, where the last 4 digits can be read

5.4 dorecurring

Sample code in the SDK: **dorecurring.php**
The request initiation can be implemented as follows:

Dorecurring is a "**do**" request discussed earlier, with the difference that the value of one of the registered tokens must be sent in the variable called "**token**" instead of "**cardSecret**". The API sends a synchronous response to the request.

Request

```
//Import config data
require_once 'src/config.php';
require_once 'src/SimplePayV21.php';

// new SimplePayDoRecurring instance
$trx = new SimplePayDoRecurring;

//add config data
$trx->addConfig($config);

//token
$trx->addData('token', 'SPT9KF5FG8VQX944PXPM227HHFJ253O778EKS2XSQ6KX5GRQAKWELCTH9VOQXKML');

//add merchant transaction ID
$trx->addData('orderRef', '101010515510244751578');

//threeDSReqAuthMethod
$trx->addData('threeDSReqAuthMethod', '02');

// METHODS
$trx->addData('methods', array('CARD'));

// add currency
$trx->addData('currency', 'HUF');

//ORDER PRICE/TOTAL
$trx->addData('total', '42');

//customer's name
$trx->addData('customer', 'Dorecurring tester');

//customer's email
$trx->addData('customerEmail', 'sdk_test@otpmobil.com');

//start dorecurring
$result = $trx->runDo();
```

Response

```
Array
(
    [responseBody] => {"salt":"j7Bge2EZoFH8ygegvdgz42ieEDa6LG4T","merchant":" PUBLICTESTHUF ","orderRef
":"101010515510244751578","currency":"HUF","transactionId":99172598,"total":42.0}
    [responseSignatureValid] => 1
    [salt] => j7Bge2EZoFH8ygegvdgz42ieEDa6LG4T
    [merchant] => PUBLICTESTHUF
    [orderRef] => 101010515510244751578
    [currency] => HUF
    [transactionId] => 99172598
    [total] => 42
)
```

5.5 cardquery

The implementation of the cardquery request using the SDK is the same as described earlier for oneclick card storage.

5.6 cardcancel

The implementation of the cardcancel request using the SDK is the same as described earlier for oneclick card storage.

6 Implementation for legacy card storage and stored card payments

In this way, the registration and payment processes can be technically simpler, but the card is not stored in SimplePay's own system, but in OTP Bank's system.

The location of the storage is important in that SimplePay can initiate transactions for authorisation to multiple recipients for cards stored in its own system, depending on which is currently available. In the case of cards saved in the banking system, authorisation can only take place in the Bank's system when making a payment.

When registering a card or registering a customer in the merchant system, a card registration statement has to be displayed and accepted.
Depending on the method of use, it is necessary to apply one of the two statements in this documentation that covers the trader process.

- oneclick card registration statement
- recurring card registration statement

The transaction data must be sent by POST method in a JSON string as described in the referenced documentation's "**General message format**" chapter.

Request authentication is discussed in the referenced documentation's "**Validating messages**" chapter.

6.1 start – with legacy oneclick card registration

Cards stored during Legacy card registration will be used for payments where the customer is present or not and the merchant can initiate a one-time payment with the card.
Compared to legacy recurring, the emphasis here is on one-time pay, while in case of recurrence, regular repetition will be the key term.

A one-time payment can be made with or without the customer's presence and active participation.

When the customer is present (Customer Initiated Transaction, **CIT**), the stored card can be used as a kind of convenience service, so when the customer presses the payment start button on the merchant side, the merchant system calls SimplePay in the background and the payment transaction occurs there.
The advantage of this is that the buyer will not be redirected to the SimplePay payment page and will not have to provide any additional information, but the payment can be made with only one click.

When the customer is not present (Merchant Initiated Transaction, **MIT**), the merchant can initiate payments with the stored card. This could be the case, for example, when several micro transactions take place where no real payment is initiated immediately, but the trading system initiates the payment with the aggregate amount to SimplePay once a day.

Customers saving their cards must be users registered in the merchant's system. This service cannot be provided for non-registered users!

In this case, the **legacyCardRegister** variable with a value of **true** must be added for the start request.

In all other aspects (e.g. **3DS**), the registration transaction is identical to the normal **start** function previously described in the aforementioned documentation. In the JSON sample above, the supplementation related to the legacy oneclick function is highlighted in red.

Request initiation


```
  {
   "salt":"348e88ac5605ab6434421300e6a260f9",
   "merchant":"LRMIPSMHUF",
   "orderRef":"101010515975209557334",
   "currency":"HUF",
   "sdkVersion":"SimplePay_PHP_SDK_2.0.9_200130:dc597528c6fb3b4c6312d470ed28895e",
   "methods":[
      "CARD"
   ],
   "legacyCardRegister":true,
   "total":"25",
   "threeDSReqAuthMethod":"02",
   "customerEmail":"sdk_test@otpmobil.com",
   "language":"HU",
   "timeout":"2020-08-15T19:59:15+00:00",
   "url":"http:\/\/payusdk.simplelabs.hu\/nandi\/v21_200814\/backstorage.php",
   "invoice":{
      "name":"SimplePay V2 Tester",
      "country":"hu",
      "state":"Budapest",
      "city":"Budapest",
      "zip":"1111",
      "address":"Address 1",
      "address2":"",
      "phone":"06203164978"
   }
}
```


Response to the request


```
{
   "salt":"EULqEyvPShJfb7lfQFT1ykDZLtasktR7",
   "merchant":"LRMIPSMHUF",
   "orderRef":"101010515975209557334",
   "currency":"HUF",
   "transactionId":500068753,
   "timeout":"2020-08-15T21:59:15+02:00",
   "total":25.0,
   "paymentUrl":"https://sandbox.simplepay.hu/pay/pay/pspHU/P1tH6yhbj684031MsmGZVDmzWv07FAZRwShwJrOe8BeDbygmCo"
}
```


The "**transactionId**" got back in the response is the ID of the created transaction. This will later be the unique data identifying the stored card, which will be used as a "**cardId**" to initiate stored card payments.

After successful authorisation of the transaction created in the start request, the saved card will be active to make further payments.

6.2 do – legacy recurring payment

A card saved for legacy oneclick purpose can be debited with a "**do**" request. The operating logic of the do request can be found in Chapter "**do – initiating transaction with a saved card**".

The only difference compared to the "**do**" call described earlier is that **it is not necessary to send** the **cardSecret** variable, as it was not included in the **start** request either.

If the customer is present, the stored card transaction can be initiated as follows:
In this case, the value of the **type** will be **CIT** and the **browser** data have to be sent.
The previously saved card is indicated by the registration transaction ID specified in the **cardId** field.

As the customer is present, the **challenge** process can also be applicable. The detailed description can be found in the following chapters regarding "do" communication

- **"do - with unsuccessful 3DS authentication"**
- **"do – challenge"**

Due to the possibility of the **challenge**, the "**url**" field can be sent in this case as well.


```
{
   "salt":"101f2e79fa37ecf489acbb13245f1f19",
   "orderRef":"101010515975236696434",
   "customerEmail":"sdk_test@otpmobil.com",
   "merchant":"LRMIPSMHUF",
   "currency":"HUF",
   "methods":[
      "CARD"
   ],
   "total":5,
   "cardId":"500068760",
   "threeDSReqAuthMethod":"02",
   "type":"CIT",
   "browser":{
      "accept":"text\/html,application\/xhtml+xml,application\/xml;q=0.9,image\/webp,image\/apng,*\/*;q
=0.8,application\/signed-exchange;v=b3;q=0.9",
      "agent":"Mozilla\/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit\/537.36 (KHTML, like Gecko) Chrom
e\/84.0.4147.105 Safari\/537.36",
      "ip":"94.199.53.96",
      "java":false,
      "lang":"hu-HU",
      "color":24,
      "height":1280,
      "width":720,
      "tz":-120
   },
   "url":"http:\/\/payusdk.simplelabs.hu\/nandi\/v21_200814\/back.php",
   "sdkVersion":"SimplePay_PHP_SDK_2.0.9_200130:dc597528c6fb3b4c6312d470ed28895e"
}
```

The response to the request is the same as the output of "**do**" described above.


```
{
   "salt":"anORX5M2d66mWskKWWcoqkqMtF58wYk7",
   "merchant":"LRMIPSMHUF",
   "orderRef":"101010515975236696434",
   "currency":"HUF",
   "transactionId":500068761,
   "total":5.0
}
```


In that case, if the card issuing bank requests additional interactive customer identification for the payment, the system will return the URL to which the customer should be redirected in the "redirectUrl" field in the synchronous response of the "**do**" as described earlier in the **challenge** process.


```
{
  "salt":"QnQtqSrVu01g331fNkXZBD2YusLzfG5U",
  "merchant":"PUBLICTESTHUF",
  "orderRef":"101010515975236696434",
  "currency":"HUF",
  "total":5.0
  "redirectUrl": "https://sandbox.simplepay.hu/pay/pay/ABCDEFG",
  "errorCodes": <errorCode>[]
}
```


If the customer is not present, the stored card transaction can be initiated as follows:
In this case, the value of **type** will be **MIT**. As the customer is not present and the interactive challenge process is not possible, there is no need for **browser** data or sending the **url.**
The previously saved card is indicated by the registration transaction ID specified in the **cardId** field.


```
{
   "salt":"101f2e79fa37ecf489acbb13245f1f19",
   "orderRef":"101010515975236696434",
   "customerEmail":"sdk_test@otpmobil.com",
   "merchant":"LRMIPSMHUF",
   "currency":"HUF",
   "methods":[
      "CARD"
   ],
   "total":5,
   "cardId":"500068760",
   "threeDSReqAuthMethod":"02",
   "type":"MIT",
   "sdkVersion":"SimplePay_PHP_SDK_2.0.9_200130:dc597528c6fb3b4c6312d470ed28895e"
}
```

The response to the request is the same as the output of "**do**" described above.


```
{
   "salt":"anORX5M2d66mWskKWWcoqkqMtF58wYk7",
   "merchant":"LRMIPSMHUF",
   "orderRef":"101010515975236696434",
   "currency":"HUF",
   "transactionId":500068761,
   "total":5.0
}
```


6.3 start – with legacy recurring card registration

Cards stored during Legacy recurring card registration will be used for payments where the customer is not present and the merchant continuously charges the card at some regular time interval.
The emphasis here is on regularity, on debiting at equal intervals. Intervals can be monthly, weekly, daily etc., according to the merchant's business model.

Customers saving their cards must be users registered in the merchant's system. This service cannot be provided for non-registered users!

In this case, the **legacyRecurring** variable with a value of **true** must be added for the start request.

In all other aspects (e.g. **3DS**), the registration transaction is identical to the normal **start** function previously described in the aforementioned documentation. In the JSON sample above, the supplementation related to the legacy oneclick function is highlighted in red.

Request initiation


```
{
 "salt":"348e88ac5605ab6434421300e6a260f9",
 "merchant":"LRMIPSMHUF",
 "orderRef":"101010515975209557334",
 "currency":"HUF",
 "sdkVersion":"SimplePay_PHP_SDK_2.0.9_200130:dc597528c6fb3b4c6312d470ed28895e",
 "methods":[
    "CARD"
 ],
 "legacyRecurring":true,
 "total":"25",
 "threeDSReqAuthMethod":"02",
 "customerEmail":"sdk_test@otpmobil.com",
 "language":"HU",
 "timeout":"2020-08-15T19:59:15+00:00",
 "url":"http:\/\/payusdk.simplelabs.hu\/nandi\/v21_200814\/backstorage.php",
 "invoice":{
    "name":"SimplePay V2 Tester",
    "country":"hu",
    "state":"Budapest",
    "city":"Budapest",
    "zip":"1111",
```

```
      "address":"Address 1",
      "address2":"",
      "phone":"06203164978"
   }
}
```

Response to the request

```
{
   "salt":"EULqEyvPShJfb7lfQFT1ykDZLtasktR7",
   "merchant":"LRMIPSMHUF",
   "orderRef":"101010515975209557334",
   "currency":"HUF",
   "transactionId":500068753,
   "timeout":"2020-08-15T21:59:15+02:00",
   "total":25.0,
   "paymentUrl":"https://sandbox.simplepay.hu/pay/pay/pspHU/P1tH6yhbj684031MsmGZVDmzWv07FAZRwShwJrOe8BeDbygmCo"
}
```

The "**transactionId**" got back in the response is the ID of the created transaction. This will later be the unique data identifying the stored card, which will be used as a "**cardId**" to initiate stored card payments.

After successful authorisation of the transaction created in the start request, the saved card will be active to make further payments.

6.4 do – legacy recurring payment

Since the customer is not present and a recurring payment will be made, in this case the value of **type** will be **REC**. As the customer is not present and the interactive challenge process is not possible, there is no need for **browser** data or sending the **url.**
The previously saved card is indicated by the registration transaction ID specified in the **cardId** field.

As the card was saved for recurring purposes, the **legacyRecurring** variable with a value of **true** must also be sent in the "do" request.

```
{
   "salt":"101f2e79fa37ecf489acbb13245f1f19",
   "orderRef":"101010515975236696434",
   "customerEmail":"sdk_test@otpmobil.com",
   "merchant":"LRMIPSMHUF",
   "currency":"HUF",
   "methods":[
      "CARD"
   ],
   "total":5,
   "cardId":"500068760",
   "legacyRecurring":true,
   "threeDSReqAuthMethod":"02",
   "type":"REC",
   "sdkVersion":"SimplePay_PHP_SDK_2.0.9_200130:dc597528c6fb3b4c6312d470ed28895e"
}
```

The response to the request is the same as the output of "**do**" described above.


```
{
   "salt":"anORX5M2d66mWskKWWcoqkqMtF58wYk7",
   "merchant":"LRMIPSMHUF",
   "orderRef":"101010515975236696434",
   "currency":"HUF",
   "transactionId":500068761,
   "total":5.0
}
```

7 Use of PHP SDK for legacy card storage and stored card transactions

The sample code for SDK will also be updated for the documentation. Based on that, this chapter on the use of the sample code will be updated until 24.08.2020.

8 Logos and information pages

In the event of a registration payment, the SimplePay logo must be displayed in a permanently visible section of the payment acceptor's site (e.g. in the footer), or during the payment method selection for the transaction.
The logo must be placed and displayed as described in the referenced documentation's "**Logos and information pages**" chapter.

9 Data Transfer Declaration

A data transfer declaration is only necessary in the event of a card registration transaction.

As the merchant transfers the order/customer data to a third party, the customer must explicitly accept the data transfer declaration.
The declaration has to be displayed, and accepted by customers, as described in the referenced documentation's "**Data Transfer Declaration**" chapter.

The data transfer declaration is not replaced by the previously specified declaration regarding card storage.

10 Testing

Every merchant system (webstore, mobile application) is tested by SimplePay before deployment.

Card storage tests are supplementary to those described in the referenced documentation's "**Testing**" chapter. It necessary to test the following elements even if the card storage function has been implemented as an enhancement of a previously deployed – and tested – merchant system.

10.1 General test elements for card storage

The following test elements are valid for each card storage mode.

10.1.1 SSL certification

- The merchant system uses an appropriate SSL certification

10.1.2 Registered user

- Bank cards may only be saved by registered users of the merchant's website

10.1.3 Data Transfer Declaration

- The necessary declaration appears as described, before the registration transaction is started, or during registration
- The declaration, or the document containing it, must be accepted by the customer
- The declaration is filled in with the merchant's details

10.1.4 Information on card registration

- The customer is properly informed that his or her card will be registered during the payment transaction
- The customer is informed about why his or her card will be registered and what payments will it be used for in the future

10.1.5 Successful registration transaction

- The registration transaction runs appropriately until the end
- The necessary information is displayed as described above

10.1.6 Multiple registrations

- If the event of more than one registration, the registrations are displayed in a distinguishable, non-ambiguous way for the customer.
- The customer can choose which registration to use for the payment.

10.1.7 Deleting and deactivating a stored card

- The deletion of the saved card can be accomplished from the merchant system (website or mobile application), as described above. The deletion (deactivation) must be carried out in the SimplePay system (**cardcancel**).
- If there is more than one registered card, the customer can choose which registered card to delete from the merchant system

10.2 Test elements for oneclick card storage

The following test elements supplement those described in the section "**General test elements for card storage**" above, in the event of **oneclick** payments.

10.2.1 Information in the event of oneclick card registration

- The customer is informed about what **cardSecret** is used for during registration and subsequent payments

10.2.2 Card storage declaration

- The necessary **oneclick** card storage declaration is displayed and mandatorily accepted by the customer before starting the transaction, as described above.
- The declaration does not replace, but supplements, the data transfer declaration displayed previously.

10.2.3 Initiating a oneclick transaction with a registered card

- Using the **cardSecret**, as described above, a payment can be initiated from the merchant website with the registered card (**do**).

10.3 Test elements for recurring card storage

The following test elements supplement those described in the section "**General test elements for card storage**" above, in the event of **recurring** payments.

10.3.1 Information in the event of recurring card registration

- The customer is informed about the parameters the tokens will be generated with
    - how many tokens
    - for what amount
    - expiry date

10.3.2 Card storage declaration

- The necessary **recurring** card storage declaration is displayed and mandatorily accepted by the customer before starting the transaction, as described above.
- The declaration does not replace, but supplements, the data transfer declaration displayed previously.

10.3.3 Initiating a recurring transaction with a registered card

- using the token, a payment can be initiated with the registered card (**dorecurring**)

10.4 Test elements for legacy card storage

The following test elements supplement those described in the section "**General test elements for card storage**" above, in the event of **legacy** payments.

10.4.1 Information in the event of oneclick card registration

- The customer is informed about his or her car will be stored for future payments

10.4.2 Card storage declaration

- The necessary **oneclick** or **recurring** card storage declaration is displayed and mandatorily accepted by the customer before starting the transaction, as described above.
- The declaration does not replace, but supplements, the data transfer declaration displayed previously.

10.4.3 initiating a legacy oneclick transaction with a registered card

- if the legacy payments are implemented in **oneclick** logic, as described above, a payment can be initiated from the merchant website with the registered card (**do**).

10.4.4 initiating a legacy recurring transaction with a registered card

- if the legacy payments are implemented in **recurring** logic, as described above, a payment can be initiated from the merchant website with the registered card (**do**).

11 Support

For further information or technical support, please contact us at itsupport@otpmobil.com.
For faster processing, please provide us the data identifying the problem or your inquiry.

Transaction
If you have questions regarding transactions, please provide us the **SimplePay** identifier of the payment. The **identifier is composed of eight digits** in **1xxxxxxx** format in sandbox mode and in **6xxxxxxx** format for live transactions.

Interface
Whether the transaction was started using V1 or V2 API.

Merchant account
Regarding technical configurations, please provide the identifier of the **merchant's account** within the SimplePay system. The identifier is the account's **MERCHANT value**.

Payment system
The system your inquiry relates to. **Sandbox system** exclusively for tests, and **Live system** for live payment transactions.

Deployment
In case of deployment tests, please contact us via itsupport@otpmobil.com, providing the following:

- under which contracted domain name the testable payment was developed
- which account is in use (MERCHANT)
- where to access the testable system