pay:smart specification

Version 1.36
February 2022

HEADQUARTERS: DIMOCO Payments GmbH
Campus 21 Businesspark Wien
Süd Europaring F15/302
2345 Brunn am Gebirge/Vienna
Austria

Tel: +43 1 33 66 888-0
Email: sales@dimoco.eu

DIMOCO Germany GmbH
Fritz-Vomfelde-Strasse 12
40547 Düsseldorf
Germany

Tel: +43 1 33 66 888-0 (AT)
Email: sales@dimoco.de

DIMOCO Greece
Olimpou 19, Nea Filothei/Marousi
15123 Athens
Greece

Tel: +30 (217) 777 4300
Email: sales@dimoco.gr

DIMOCO Hungary Kft.
Alkotás út 53.
1123 Budapest
Hungary

Tel: +36 1 611 9653
Email: sales@dimoco.hu

TABLE OF CONTENTS

1 CHANGE LOG

v1.0	initial version
v1.1	additional use case and action *periodic-status* documentation for result and status codes
v1.2	parameter *order* for action *identify* is optional now parameters *url_callback* and *url_return* are mandatory, unless a default configuration is requested
v1.3	description of digest computation enhanced token renamed to merchant password for clarification
v1.4	clarification of parameters *method* vs. *operator* new feature *status with return*
v1.5	introduction of parameter *service_category* for the feature *one subscription per category* introduction of parameter *service_name*
v1.6	additions for new payment method *CALL2PAY*
v1.7	new prompt parameter *prompt_image_args* separation of parameters *method* and *operator* against ambiguity description of additional xml result elements new action result codes added
v1.8	new parameter *manage_subscription_url_callback* introduction of parameter *redirect*
v1.9	new action *prompt* with parameter *subject* added
v1.10	explanation for server to server callback handling with retry behaviour
v1.11	chapters tightened and reorganized for easier comprehensibility
v1.12	added new action result codes 522, 523 and parameter *signifier*
v1.13	new chapter *notifications for managed resources* added
v1.14	parameter *service_name* now mandatory for action *start* and *start-subscription*
v1.15	obsolete staging environment for setup tasks removed
v1.16	new parameter *close_notification_url_callback*
v1.17	subscriptions without an immediate first payment are no longer indicated via subscription-status but an explicit flag within additional-results
v1.18	explanation of possible additional-results
v1.19	parameters and result codes for novel functionality *age-verification* added clarified that action *periodic-status* is only available via GET currently enhanced description for correct callback handling and digest calculation

v1.20	updated values of action result status for the indication of *unbilled subscription signups*
v1.21	description for notification *receive-sms-info* added explanation for *unbilled subscription signups* introduced
v1.22	meaning of status codes clarified
v1.23	description for action *refund* added deprecation of parameter *periodic*
v1.24	added parameter *landing_page*
v1.25	chapters reorganized and streamlined introduction of *advanced flows* for fine-grained enduser interaction
v1.26	clarified description of all relevant action parameters that are subject to be ignored under certain flow variants
v1.27	WAP push functionality for content delivery via parameter *prompt_content_args* added
v1.28	added new advanced flows called *on-demand content provisioning* and *content acknowledgement* added parameter *subscription_type*
v1.29	introduced feature *data with return* for optimized identify performance
v1.30	added action *status* and *customer_id-lookup*
v1.31	advanced flow *proportional capture of reserved amount* added added new action result codes 504, 525 and 526
v1.32	added parameter *shopper* new subject *fraudDetection* for action *prompt* introduced
v1.33	Oct 2020 graphical redesign of document action result code 404 added
v1.34	Nov 2020 clarified decryption procedure for feature *data with return*
v1.35	July 2021 action result codes 527 and 528 added corrected type declaration of *msisdn* and *customer_id* - string instead of number
v1.36	February 2022 action result codes 529 and 531 added *customer_id* may take up to 512 characters timestamps are returned with UTC offset

2 INTRODUCTION

pay:smart is DIMOCO's product providing a highly customizable and easy to use interface for performing payment transactions through DIMOCO's payment hub with all the attached mobile network operators as well as alternative payment methods. pay:smart is designed to require only a minimum number of interactions between the merchant's and DIMOCO's system and to automatically remediate all flow decisions based on customizable configuration properties. Charging itself may be performed by means of gateway billing with operators providing this technology or by premium SMS, respectively. Legal regulations and restrictions specified in MNO contracts are enforced by DIMOCO's platform.

2.1 Terminology

- enduser … consumer buying a merchant's product
- merchant … service provider using pay:smart for charging an enduser
- MNO … mobile network operator
- PSP … payment service provider
- MSISDN … mobile subscriber integrated services digital network number
- request_id … token used once for security measures (generated by merchant)
- reference … short-lived session correlator for a single pay:smart request (generated by DIMOCO)
- transaction … long-lived identifier for a payment, etc. (generated by DIMOCO)
- subscription … long-lived identifier for a subscription (generated by DIMOCO)

2.2 Service setup

The merchant must contact DIMOCO for the purpose of contracting and service setup. A service description and in case of subscriptions the definition of the subscription model including period, billing count, amount may be necessary. The merchant (if not already registered within the DIMOCO service infrastructure) will be assigned a merchant id and will receive a password needed for digest calculation. For every service a dedicated *order* (unique service identifier) is assigned. It is possible to set up default values for many of the optional parameters described in section 5

3 USE CASES

This section describes the use cases covered by pay:smart and implemented by means of different actions (see 6.1).

3.1 Enduser identification

An enduser surfing the web using the mobile internet connection of his/her mobile phone usually can be identified by the operators as the IP traffic is routed through their gateway systems. Some of the operators provide a stable unique identifier for an enduser named *alias* others provide the MSISDN.

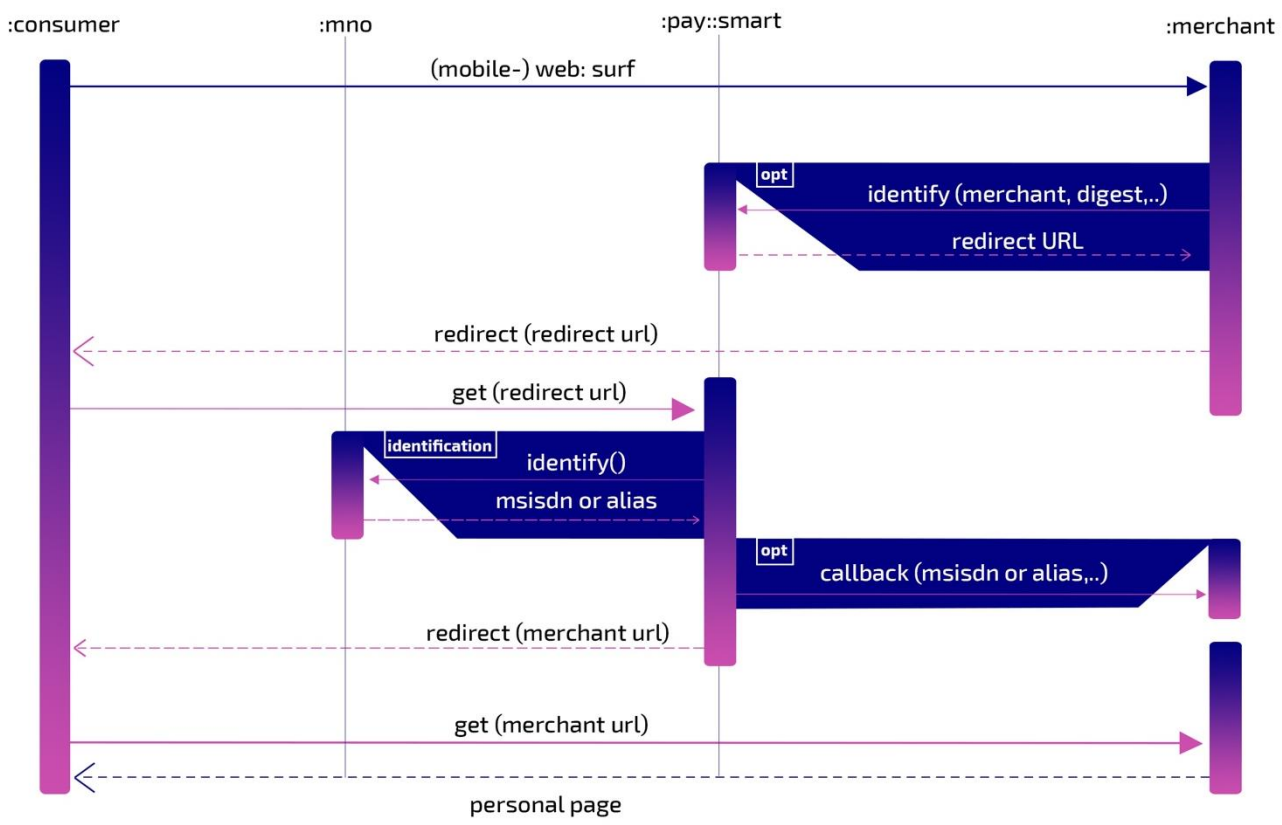

Figure 1: Flow for enduser identification

3.2 Operator lookup

Given an MSISDN and the fact that in many countries mobile phone numbers can be ported between different mobile networks the number itself does not provide a reliable way of guessing the corresponding MNO. In cases this is needed (e.g. in WEB opt-in flows) there is the possibility to query HLR databases to obtain the corresponding MNO of an MSISDN. pay:smart provides an API for retrieving this information. This is an additional feature and has to be configured explicitly.

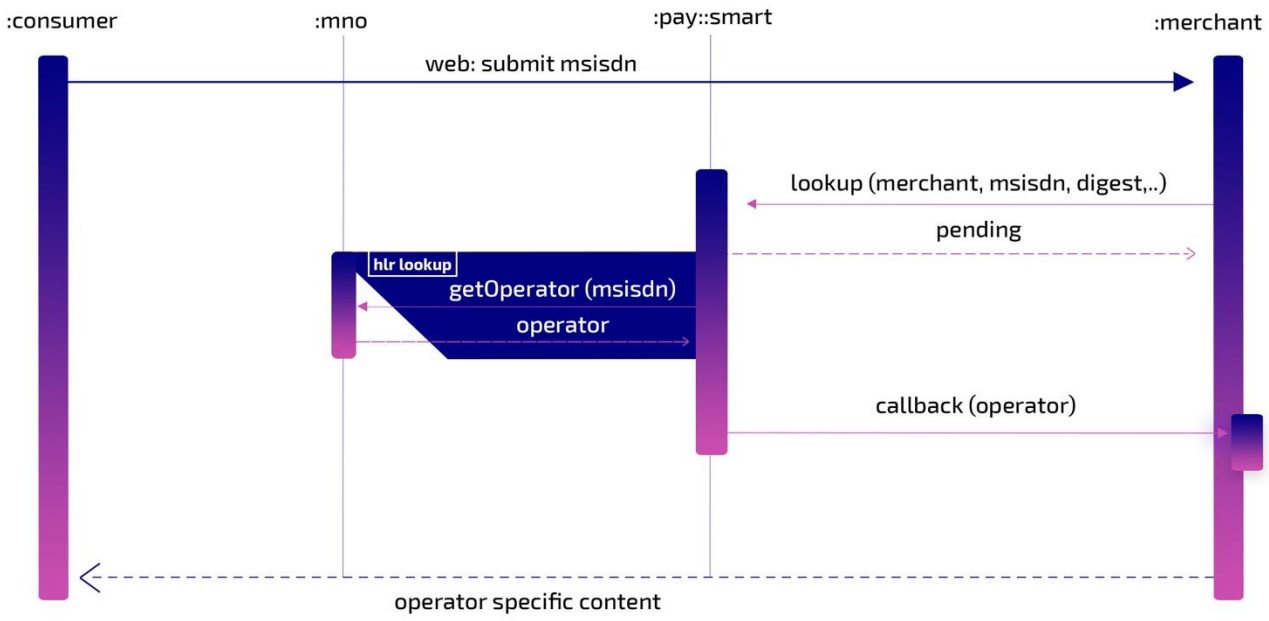

Figure 2: Flow for operator lookup

3.3 One-off payment

Whenever a single payment is requested by an enduser (for example to purchase a virtual good within an online game) the merchant must start a transaction by calling pay:smart. Depending on the provided information and corresponding configuration either a redirect URL is returned where the enduser must be redirected to, a pending status is reported, or a detailed error description is returned. pay:smart performs all steps required to gather all information needed for performing the requested charging and runs the authorization procedure if one is needed. It reports the outcome of the payment (collection of transaction data, authorization and charging) by means of a callback to the merchant's server. In case the enduser has been redirected to pay:smart at the beginning, he is redirected back to the merchant's portal *after* the callback was successfully delivered.

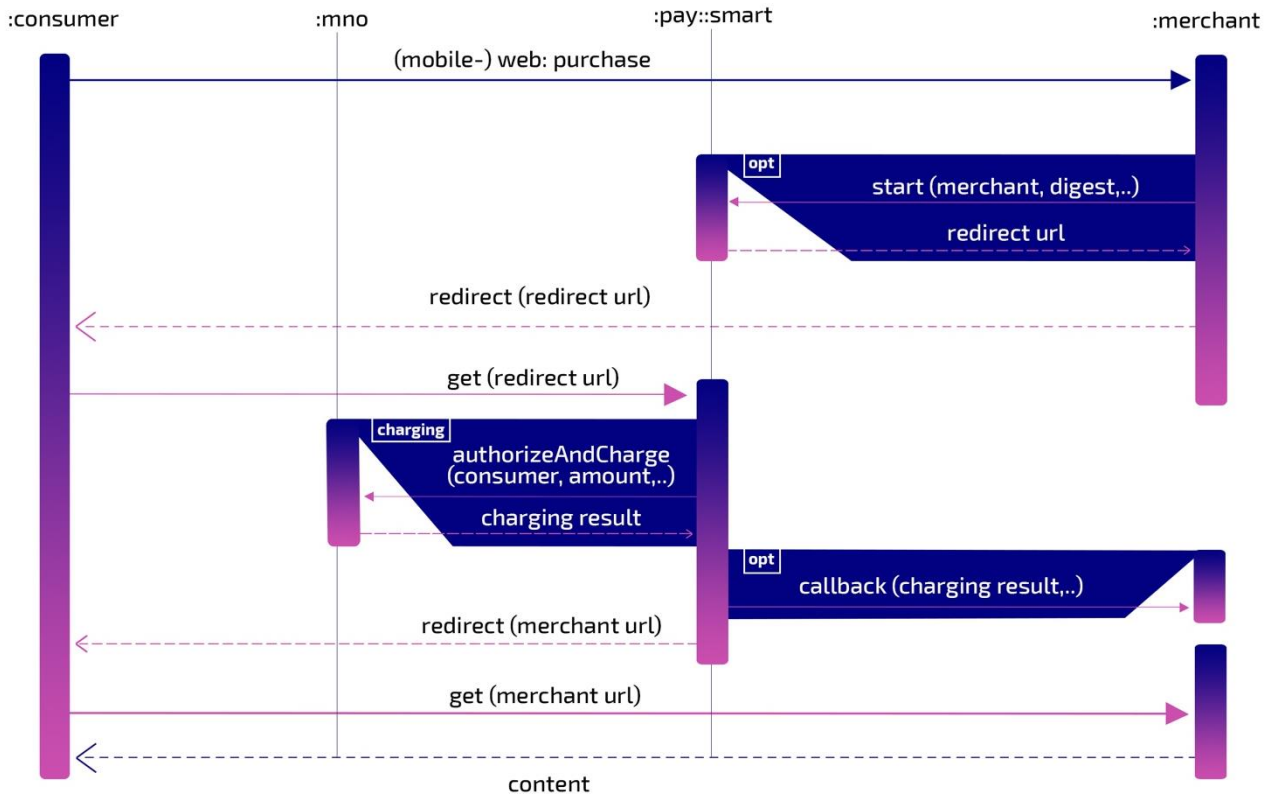

Figure 3: Flow for one-off payment

3.4 Opening a subscription

Whenever an enduser decides to sign up for a service with a subscription model on a merchant's portal the merchant has to start a subscription by calling pay:smart. The definition of the subscription model may be either provided with the initial request or can be configured on DIMOCO's side alternatively. The flow for starting the subscription is identical to the flow for performing a single one-off charging. Redirecting the enduser may be involved, various methods for retrieving subscription information and for identifying the enduser may be applied, the authorization procedure may be run, and the initial charging may be performed. Again, the outcome of the procedure is reported by means of a callback to the merchant's server and if required, the enduser is then redirected to the merchant's portal.

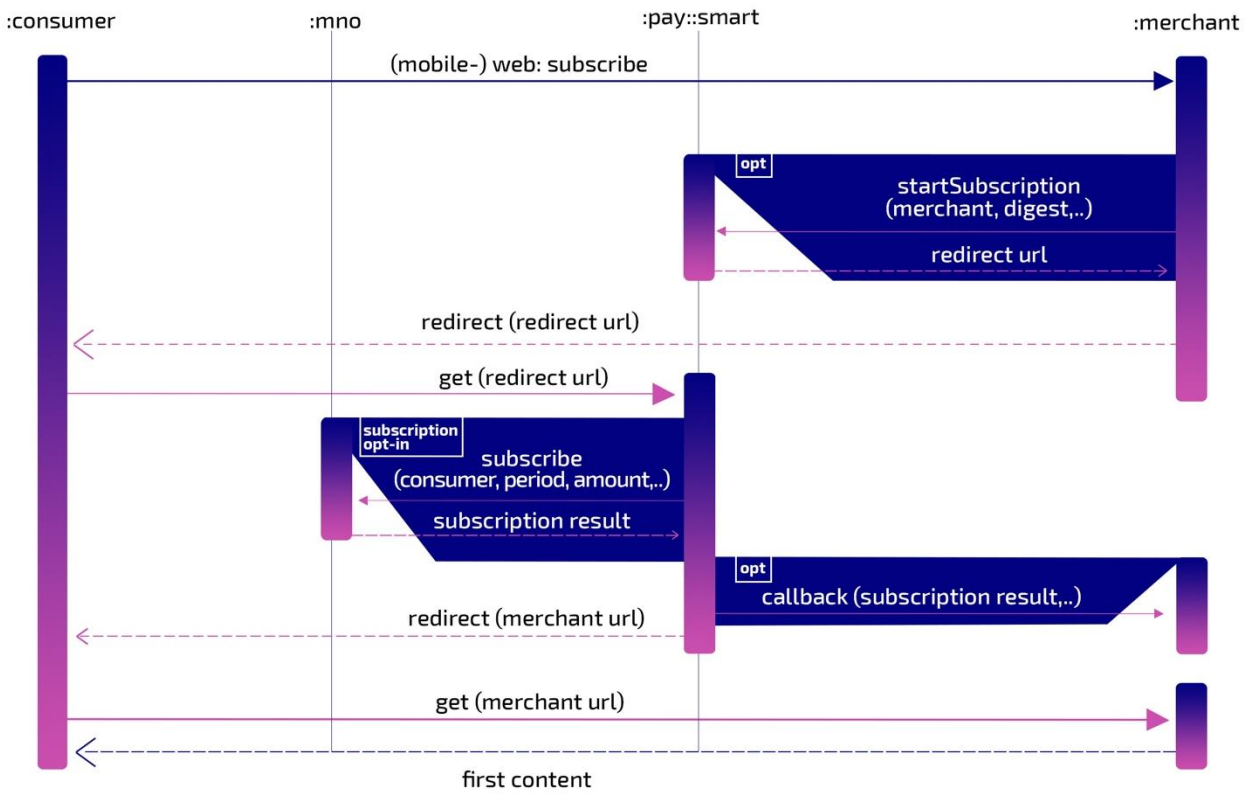

Figure 4: Flow for subscription opt-in

3.5 Renewing a subscription

Given a subscription that has been successfully opened (see the previous use case) the renewal can be performed by the merchant. Alternatively, a DIMOCO system for renewing subscriptions may be used that is called pay:periodic. In case the merchant prefers to trigger the renewal a call to the pay:smart service must be made. The result is reported by a callback to the merchant's server.

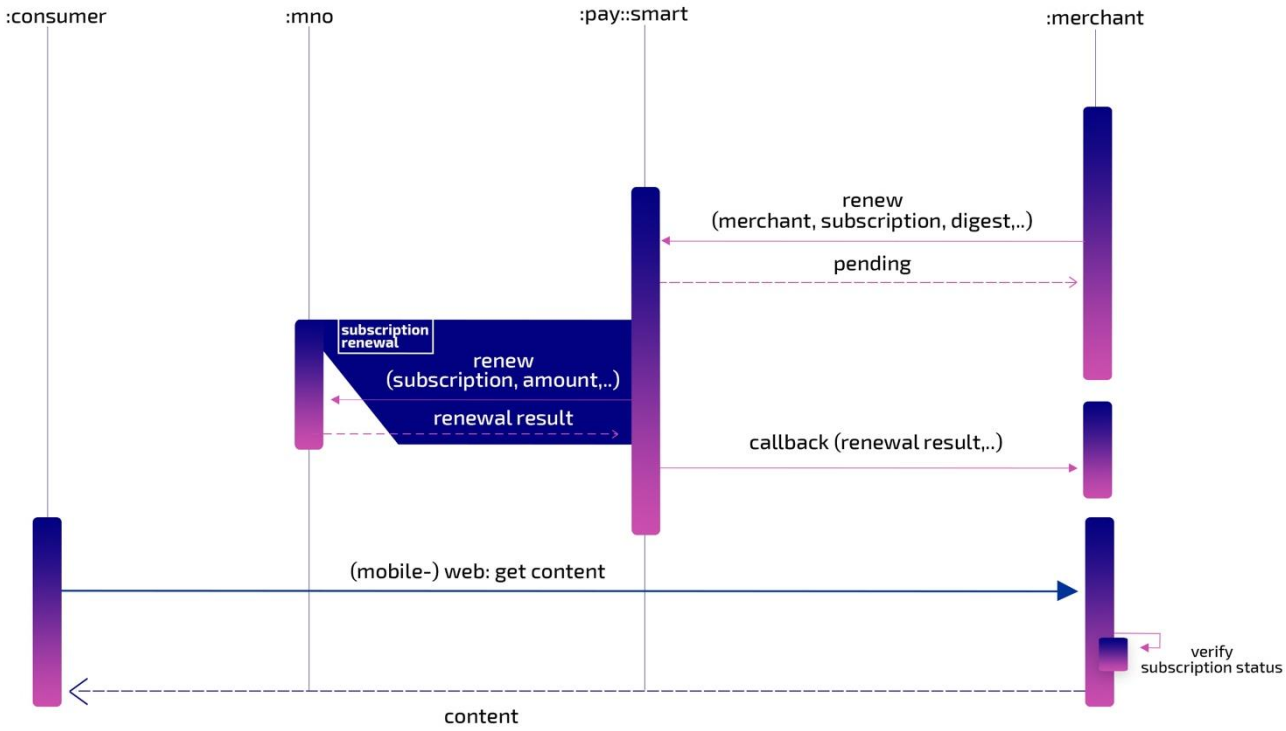

Figure 5: Flow for rebilling a subscription

3.6 Retrieving status of a subscription

A subscription that has been established via DIMOCO's platform may be queried regarding its status by means of action status.

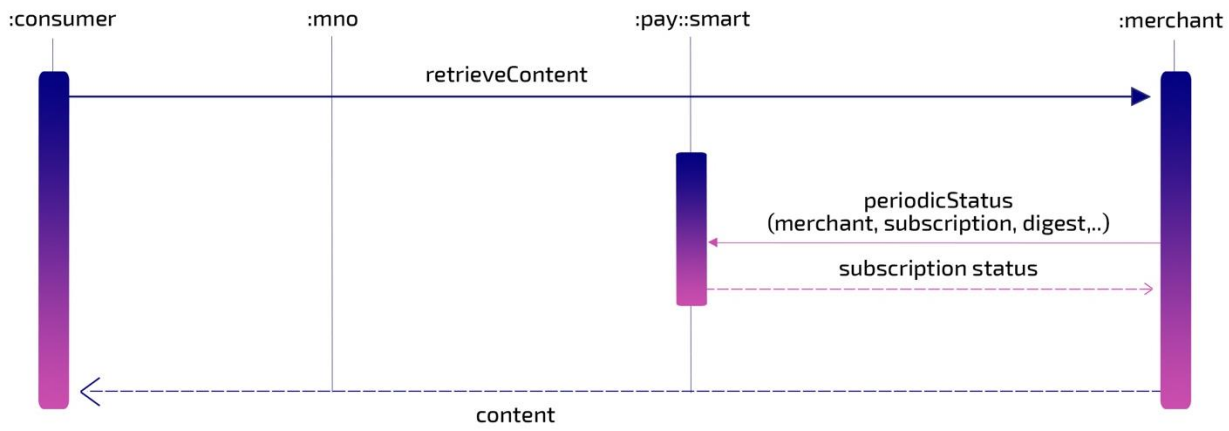

Figure 6: Flow for subscription status retrieval

3.7 Closing a subscription

A previously opened subscription may be closed in different ways. Sometimes there is a SMS channel providing the possibility to send a STOP command to the subscription service. In other cases, there are portals where endusers can view and close their subscriptions and of course a merchant usually provides the unsubscribe option on the service portal. To have the subscription closed in DIMOCO's and the PSP's systems action close-subscription must be invoked at pay:smart. In rare cases a redirect URL is returned – there are markets where an opt-out needs to be confirmed by the subscriber. As usually the result of the action is posted to the merchant's server in a callback.

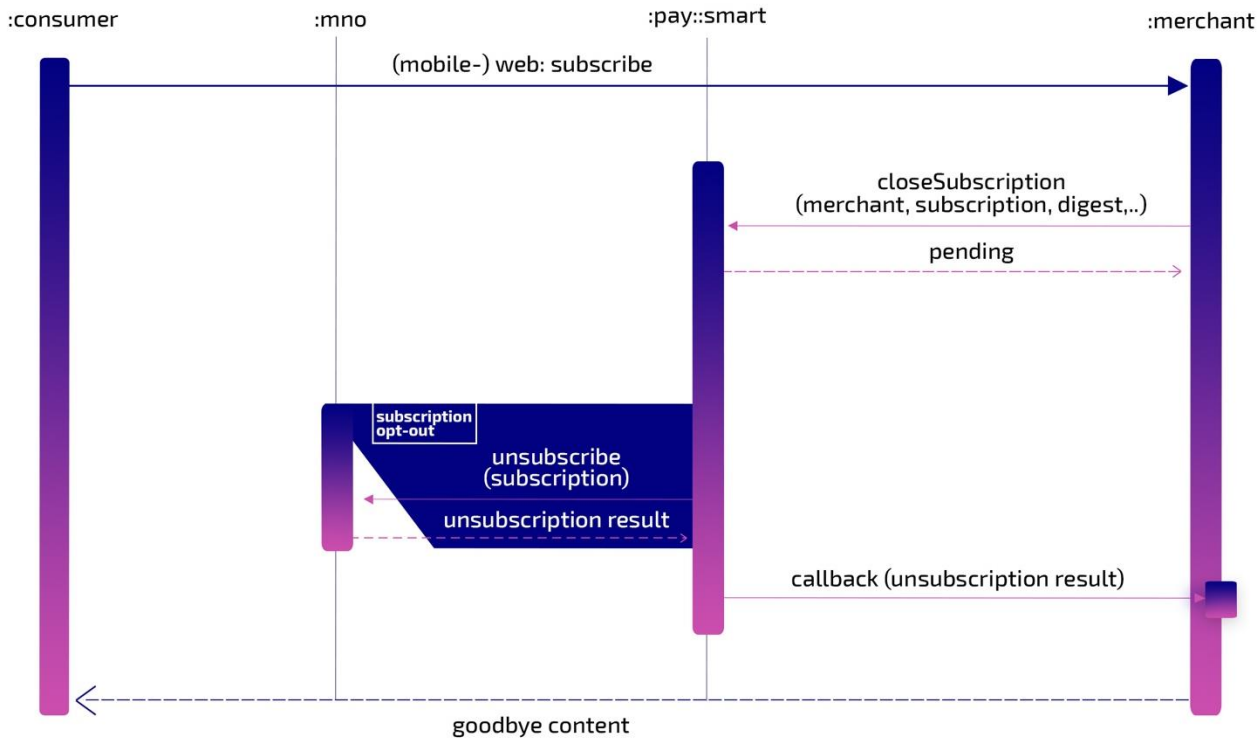

Figure 7: Flow for subscription opt-out

4 INFORMATION EXCHANGE

4.1 Communication pattern

In general, the communication pattern between the merchant's system and pay:smart is the following:

1. the merchant's system calls pay:smart to initiate an action
2. pay:smart synchronously accepts the call and optionally returns a redirect URL
    - if given, the enduser must be redirected to this URL (for cases that require identification or involve other forms of interaction like data entry, confirmation, …)
3. callbacks are issued by pay:smart to the merchant's server to report
    - necessary interim activities (only for *advanced flows* – see 8)
    - the final outcome of the action
4. if the enduser has been redirected to pay:smart he is redirected back to the merchant's portal *after* the first callback has been successfully posted to the merchant's server

4.2 Transport variants

4.2.1 Server to server

With server to server communication the exact pattern described above applies. It is the most general, reliable, secure and thus recommended way of conveying information between the merchant's system and pay:smart. As an additional benefit the enduser will experience the minimal number of redirects and will have no way of seeing possibly confidential payment details.

4.2.2 Enduser transport

The initial server to server API call can be omitted when the enduser himself transports all the necessary information. To get this working an html form or similar must be presented on the merchant's portal that contains all relevant API parameters. On submit the enduser's browser is posted directly to pay:smart where the action takes place.

Also, for returning the outcome of any action pay:smart can be configured to omit the server to server callback and let the enduser himself transport the information. This time an html form is presented by pay:smart that posts the enduser's browser on submit with all the relevant data directly back to the merchant's portal.

Both directions of enduser transport are independent of each other and can be used individually or together – thus eliminating server to server communication entirely.

Even though enduser transport is slightly simpler to implement and can save some network latency it comes with major drawbacks:

- not usable for all use cases or *advanced flows* – see 8
- enduser session gets lost on a failed request that the merchant's system cannot even detect
- less reliable data transport (e.g. no retry)
- enduser redirect must happen via POST instead of simple GET
- action parameters/results visible to enduser

NOTICE
We highly recommend the use of server to server communication and harness enduser transport only for cases that really justify its drawbacks.

4.3 API URLs

pay:smart has no restrictions on source IP addresses for incoming connections.

For callbacks the merchant's server must allow incoming connections coming from the following IP range: **91.198.93.0/24**

Callbacks contain sensitive information and can only be delivered via properly secured https URLs. An officially signed certificate needs to be in place on the merchant's server.

Server to server API URL: https://services.dimoco.at/smart/payment

Enduser transport API URL: https://services.dimoco.at/smart/userpayment

TIP
For optimal performance we advise to use persistent connections for server to server communication.

4.4 Call details

These sections deal with passing of information to/from pay:smart and the security measures involved. For a detailed description of the parameters see 5.

4.4.1 Towards pay:smart

Calls made towards pay:smart must be POST requests to one of the API URLs of section 4.3 providing the parameters in url-encoded form with content-type *application/x-www-form-urlencoded* and any contained strings must have charset *UTF-8*

This is actually the default behaviour if an html form is submitted via browser.

Example for a complete url-encoded request towards pay:smart:
merchant=678678&**order**=4711&**action**=start&**request_id**=98c6dec3-c5f0-4810-9490-e2b9f2e2d34a&**amount**=1.99&**url_callback**=https%3A%2F%2Fmerch.at%2Fcb%3Fx%3Dy&**digest=**ff98e66379b8474be66aad871230eba19245f21ac7b2c6908faf3bf7aafa98b4

The parameter *digest* is mandatory in all pay:smart requests and provides the necessary authentication primitive. It has to be calculated from the request's parameter values using the secret merchant password which was provided to you by DIMOCO during service setup.

Digest calculation formula: **hex(hmac_sha256_digest(password, payload))**

The payload is constructed by concatenation of all unencoded request parameter ***values*** (only the values!) in **alphabetical order of the parameter *names***.

If we take the example request from above with a merchant password of:
top-secret

The digest calculation would look like:
digest = hex(hmac_sha256_digest(utf8_bytes("top-secret"),
utf8_bytes("start1.99678678471198c6dec3-c5f0-4810-9490-e2b9f2e2d34ahttps://merch.at/cb?x=y")))

Notice that the parameter values are unencoded and were reordered to resemble alphabetical order of their corresponding parameter names!

Python code

```
import hashlib, hmac, sha, urllib

ordered_params = (("action","start"),
    ("amount",1.99),
    ("merchant","678678"),
    ("order","4711"),
    ("request_id","98c6dec3-c5f0-4810-9490-e2b9f2e2d34a"),
    ("url_callback","https://merch.at/cb?x=y"))

to_sign = ""
for k,v in ordered_params: to_sign += str(v)

digest = hmac.new("top-secret".encode("utf8"), to_sign.encode("utf8"), hashlib.sha256).hexdigest()
```

PHP code

```
$ordered_params = array('action'    => 'start',
    'amount'    => 1.99,
    'merchant'  => '678678',
    'order'     => '4711',
    'request_id' => '98c6dec3-c5f0-4810-9490-e2b9f2e2d34a',
    'url_callback' => 'https://merch.at/cb?x=y');

$to_sign = '';
foreach ($ordered_params as $v)
  $to_sign .= $v;

// maybe use utf8_encode($to_sign) if your php file is encoded iso-8859-1
$digest = hash_hmac('sha256', $to_sign, 'top-secret');
```

Perl code

```
use Encode;
use Digest::SHA;

my $params = {'action'     => 'start',
      'amount'     => 1.99,
      'merchant'   => '678678',
      'order'      => '4711',
      'request_id' => '98c6dec3-c5f0-4810-9490-e2b9f2e2d34a',
      'url_callback' => 'https://merch.at/cb?x=y'};

my $to_sign = '';
foreach my $v (sort keys %$params)
   $to_sign .= $params->{$v};

my $digest = Digest::SHA::hmac_sha256_hex(encode("utf8", $to_sign),
      encode("utf8", "top-secret"));
```

Java code

```
Map<String, String> params = new TreeMap<>();
params.put("action", "start");
params.put("amount", "1.99");
params.put("merchant", "678678");
params.put("order", "4711");
params.put("request_id", "98c6dec3-c5f0-4810-9490-e2b9f2e2d34a");
params.put("url_callback", "https://merch.at/cb?x=y");

String toSign = params.values().stream().collect(Collectors.joining());

SecretKeySpec signingKey = new SecretKeySpec("top-secret".getBytes("UTF-8"), "HmacSHA256");
Mac mac = Mac.getInstance("HmacSHA256");
mac.init(signingKey);
byte[] result = mac.doFinal(toSign.getBytes("UTF-8"));

StringBuilder digest = new StringBuilder();
for (byte b : result)
   digest.append(String.format("%02x", (b & 0xff)));
```

4.4.2 Towards merchant's server

Calls towards the merchant's server are done in a way analogous to the requests towards pay:smart. They are sent via POST with a content-type of *application/x-www-form-urlencoded* and any contained strings have charset *UTF-8*

Following parameters are transmitted:

- *data* … holds the XML result document
- *digest* … holds the authentication primitive

The digest is again calculated via: **hex(hmac_sha256_digest(password, payload))**

This time the payload consists of the whole url-decoded XML result document.

TIP
If you use some kind of web framework then the url-decoding of the parameters happens automatically behind the scenes.

With an example XML result document (stripped for brevity) the calculation of the digest would look like:

```
digest = hex(hmac_sha256_digest(utf8_bytes("top-secret"),
utf8_bytes('<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<result sync="false" version="2">
  <action>start</action>
  <action_result>
    <status>0</status>
  </action_result>
  <payment_parameters>
    <order>4711</order>
  </payment_parameters>
  <transactions>
    <transaction>
      <id>999999999</id>
      <amount>1.99</amount>
      <billed_amount>1.99</billed_amount>
      <currency>EUR</currency>
      <status>5</status>
    </transaction>
  </transactions>
  <request_id>98c6dec3-c5f0-4810-9490-e2b9f2e2d34a</request_id>
  <reference>88888888-7777-6666-5555-abcdefgh1234</reference>
</result>
')))
```

Be cautious that any possible trailing line-feed symbols must not be stripped for the calculation.

If your calculated digest matches the value of the parameter *digest* you can be sure that the callback was sent by DIMOCO and was not forged by a man-in-the-middle.

NOTICE
Be aware that special characters contained in the values of the final XML document are xml-escaped. E.g. *&* is represented as *&* which is especially relevant for contained URLs. If you use an XML library for extracting values, then the xml-unescaping will happen automatically behind the scenes.

Server to server callbacks and notifications are accounted as successfully delivered only if the merchant's server answers them synchronously with http status *200*. If it cannot be delivered on the first try (http status not *200*, connection problems, etc.) it will be automatically retried for delivery until it succeeds.

4.5 Enduser return to merchant's portal

Commonly the enduser is send back to the merchant's portal either (as configured):

- via simple GET redirect *after* the XML result document was transported successfully as server to server callback (see 4.2.1) or
- via http POST request *together* with the XML result document and digest a.k.a. *callback with return* (see 4.2.2)

Parameter *url_return* given on the initiating API call (or default configured) is used as return address as is – i.e. it is normally not changed or enriched with any information. However, for specific cases some level of enrichment can be done as described in the following sections.

4.5.1 status with return

This feature can be configured for enriching *url_return* with the following core status values that reflect the final outcome. This is useful if the merchant's system cannot easily correlate the server to server callback with the following return of the enduser:

- sph-x … final status of the action with one of the following values
  - *s* … success
  - *u* … unbilled – active subscription but no/failed first payment
  - *p* … pending
  - *f* … failure
- *sph-r* … *reference* for correlation with the originating request
- *sph-s* … id of newly opened or already existing subscription
- *sph-d* … digest over the concatenated unencoded values of sph-r, sph-s and *sph-x* (computation described in 4.4.2)

WARNING
It is very important to check the validity of the parameter values with the given digest on merchant side since values transported this way are extremely easy to be forged by the enduser.

4.5.2 data with return

To gain the highest level of performance for action identify the final XML result document can be transported together with the return of the enduser as encrypted query parameter. The enduser is redirected back to the merchant's portal immediately after identification via simple GET while **no server to server callback** is send. Following parameters are added to *url_return*:

- *sph-e* … compressed and encrypted version of the final xml result
- *sph-n* … nonce necessary for decryption

Decryption procedure

1. repeat/cut your password to **16** characters – this is the decryption-key
   a. if a conversion to a byte array is necessary: use charset **UTF-8** and limit the length to exactly **16** bytes (i.e. cut excess bytes)
2. **base64-decode** the values you received in *sph-e* (payload) and *sph-n* (nonce)
   a. please ensure that these values are url-decoded beforehand (usually done by your web framework behind the scenes)

3. decrypt payload via nonce and decryption-key with the following cipher: **AES/GCM without padding**
4. gunzip the decrypted value
5. convert the decompressed value into a string with **UTF-8** charset to retrieve the original XML result document

Below you find some examples how this decryption procedure can be implemented in various programming languages.

Java code

```
String secret = <merchant password>;
String payload = Base64.getDecoder().decode(request.getParameter("sph-e"));
String nonce = Base64.getDecoder().decode(request.getParameter("sph-n"));

byte[] key = Arrays.copyOf((secret + secret + secret).getBytes("UTF-8"), 16);

Cipher ci = Cipher.getInstance("AES/GCM/NoPadding");
SecretKeySpec sc = new SecretKeySpec(key, "AES");
GCMParameterSpec gcm = new GCMParameterSpec(128, nonce);
ci.init(Cipher.DECRYPT_MODE, sc, gcm);
byte[] decrypt = ci.doFinal(payload);

GZIPInputStream gz = new GZIPInputStream(new ByteArrayInputStream(decrypt));
ByteArrayOutputStream bos = new ByteArrayOutputStream();
byte[] buffer = new byte[1024];
for (int len = 0; (len = gz.read(buffer)) >= 0;) {
   bos.write(buffer, 0, len);
}

String xmlResult = new String(bos.toByteArray(), "UTF-8");
```

PHP code

```
$secret = <merchant password>;
$payload = base64_decode($_GET['sph-e']);
$nonce = base64_decode($_GET['sph-n']);

$secret = substr($secret . $secret . $secret, 0, 16);

$ci = 'aes-128-gcm';
$encrypt = substr($payload, 0, -16);
$tag = substr($payload, -16);

$decrypt = openssl_decrypt($encrypt, $ci, $secret, OPENSSL_RAW_DATA, $nonce, $tag);

$xmlresult = gzdecode($decrypt);
```

5 ACTION PARAMETERS

The table below shows all available parameters for pay:smart API calls. The actions described in the following sections will reference their relevant parameters from this list. Many of these can have preconfigured values within the pay:smart configuration (in merchant context as well as in *order* context).

name	type	description
action	string	selects demanded use case – one of • identify • operator-lookup • customer_id-lookup • start • start-subscription • renew-subscription • status • close-subscription • prompt • refund
amount	number	Amount to be charged given as a decimal number (period as the separator). The corresponding currency is configured on DIMOCO's side as a property of *order*. For certain flow variants this parameter is ignored.
artifact	string	Instead of passing or configuring *amount*, a "virtual good" may be defined with a price per country configuration. The virtual good can be addressed with the *artifact* parameter. For certain flow variants this parameter is ignored.
channel	string	Enduser authorization technique – one of • web • wap • sms For certain flow variants this parameter is ignored.
close_notification_url_call back	string	https URL where notifications for managed subscriptions that have recently been closed shall be posted. Can be configured as a property of *order*.
country	string	ISO 3166-1 Alpha 2 code. For certain flow variants this parameter is ignored.
cp_*	string	Custom parameter – to be used whenever a merchant needs to pass any parameters that shall be present in the callback. Multiple custom parameters (with different names) may be provided.
customer_id	string	Abstract unique identification string for enduser (a.k.a. alias) that is used by some PSPs. Current max size is 512 chars. Maybe determined by DIMOCO if unknown. For certain flow variants this parameter is ignored.
digest	string	This is the authentication signature calculated from the query string and merchant password provided during service setup by DIMOCO. See 4.4.1 for a detailed description.

name	type	description
ip	string	IP address of enduser's device. For certain flow variants this parameter is ignored.
landing_page	string	could be presented to the enduser within payment-pages and will possibly be forwarded to the PSP
language	string	ISO 639-1 code of language to be used for enduser interaction. For certain flow variants this parameter is ignored.
manage_subscription_url_callback	string	https URL where notifications for managed subscriptions that have recently been renewed shall be posted. Can be configured as a property of *order*.
merchant	string	merchant_id – merchant identification provided by DIMOCO during service setup example: 23456
method	string	Payment method to be used for billing the enduser. There may be multiple possible methods applicable for the current enduser. One of • OPERATOR • ISP For certain flow variants this parameter is ignored.
msisdn	string	Enduser's MSISDN to be used for charging as defined by ITU-T E.164 – international format without leading + or *0*. Maybe determined by DIMOCO if unknown. For certain flow variants this parameter is ignored.
operator	string	MNO of the current enduser. See pay:smart operators for a complete list of operator aliases. Maybe determined by DIMOCO if unknown. For certain flow variants this parameter is ignored.
order	string	order_id – service identification provided by DIMOCO during service setup example: 123456
preemptive_content	number	when amount reservation via authorize is unsupported – 1 defines that content delivery shall happen *before* capture and *0 after*

name	type	description
prompt_content_args	JSON	Whenever content delivery is configured or requested (e.g. by means of *send_content* or with action *prompt)* the content itself can be configured or provided with the request. For SMS content the JSON object must contain a text element providing the message text in different languages (ISO 639-1 codes) like: {"text":{"de":"Willkommen im Club!", "en":"Welcome to the club!"}} and/or a url element for WAP push like: {"url":"http://merch.at/content"}
prompt_image_args	JSON	Link to additional images presented on payment pages. There can be multiple elements and the specific names depend on the actual used prompts. Contact DIMOCO for the element names of your specific use case. example with 2 images: {"album":{"pic":{"img":"http://merch.at/mozart.png","alt":"Mozart"},"desc":{"de":"Amadeus"}},"track1":{"pic":{"img":"http://merch.at/requiem.png","alt":"Requiem"},"desc":{"de":"Trauer"}}}
prompt_merchant_args	JSON	Link to merchant's logo. The JSON object shall contain a *logo* element with *img* and alt properties that will be embedded in the user prompt pages. example: {"logo":{"img":"http://merch.at/logo.jpg","alt":"Ring Store"}}
prompt_product_args	JSON	Link to product image, and product description given in multiple languages. The JSON object shall contain a pic element with *img* and *alt* properties as well as a *desc* element providing the product names in different languages. For language codes refer to ISO 639-1. example: {"pic":{"img":"http://merch.at/ring.jpg","alt":"The Ring"},"desc":{"de":"Cooler Ring","en":"Cool Ring"}}
prompt_style_args	JSON	Definition of layout/design used on payment pages. example: {"css_class":"touch", "colour_id":"1", "template_id":"5"}
prompt_url_args	JSON	Definition of administration URLs presented on payment pages. example: {"agb":{"url":"http://merch.at/agb.html"}, "impressum":{"url":"http://merch.at/impressum.html"}}
redirect	number	• 1 forces an enduser redirect to pay:smart during request processing – even if actually unneeded • *0* explicitly prevents it – e.g. meaningful for external or direct authorization via channel *sms* or on the merchant portal

name	type	description
request_id	string	short-lived unique token for every request generated by merchant (security measure)
send_content	number	• 1 enables content delivery • 0 disables it
service_category	string	Freely selectable category identifier that limits the number of active subscriptions to *one per category* – only if the *one subscription per category* feature is configured by DIMOCO. examples: gold, silver, platinum, nice, bad, …
service_name	string	name of the service as it will possibly be presented to the enduser within payment-pages, invoice lines, etc.
shopper	string	Enduser's id as defined by the merchant. E.g. username, email, SSN, member-id, …
signifier	string	This parameter is used to supply relevant external information for the following transaction. E.g. a sms reference-id from a preceding activity.
subject	string	accompanying type necessary if action *prompt* is used – currently supported • content • fraudDetection
subscription	string	subscription id – uniquely identifies a subscription
subscription_type	string	Demands a certain special type for a subscription to be opened – currently supported • trial The usage of this parameter must explicitly be allowed by DIMOCO or else your request will be rejected.
transaction	string	transaction id – uniquely identifies a transaction. For certain flow variants this parameter is ignored.
url_callback	string	https URL where the transaction result shall be posted. May be configured as a property of *order*.
url_return	string	http(s) URL where the enduser shall be redirected *after* the callback has been delivered. May be configured as a property of *order*.

6 API FOR TRIGGERING ACTIONS

The following actions represent the main API of pay:smart and must explicitly be triggered by the merchant's system when demand arises. For every action the relevant request parameters are given and the contents of the synchronous response and final callback are described.

6.1 Actions

6.1.1 identify

Action *identify* is used to determine the MSISDN or abstract alias and/or the operator of an enduser.

Request parameters (see 5):

name	mandatory?
action	yes
cp_*	no
digest	yes
merchant	yes
order	yes, unless a default has been configured on service setup
request_id	yes
url_callback	yes, unless a default has been configured on service setup
url_return	yes, unless a default has been configured on service setup

The response XML document contains the following elements:

xpath	type	description
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/redirect/**url**	string	URL where the enduser shall be redirected (only present with status 3)
/result/action_result/**status**	number	• 1 … failure • 3 … redirect required • 4 … validation failed
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	request_id provided with the API request

The XML document contained in the data form field of the callback POST has the following elements:

xpath	type	description
/result/**action**	string	the action this result refers to
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/**detail_psp**	string	error code and/or error description from PSP

/result/action_result/**status**	number	• 0 … success • 1 … failure
/result/additional_results/additional_result/**key**	string	name of additional dynamic return parameter – see 6.2
/result/additional_results/additional_result/**value**	string	value of additional dynamic return parameter – see 6.2
/result/custom_parameters/custom_parameter/**key**	string	name of custom parameter passed to the API call
/result/custom_parameters/custom_parameter/**value**	string	value of custom parameter passed to the API call
/result/customer/**country**	string	ISO 3166-1 Alpha 2 code
/result/customer/**id**	string	enduser's id aka alias
/result/customer/**ip**	string	internet address of enduser's device
/result/customer/**language**	string	ISO 639-1 code
/result/customer/**msisdn**	string	enduser's MSISDN
/result/customer/**operator**	string	enduser's operator – see pay:smart operators for a listing
/result/payment_parameters/**channel**	string	recommended technique for enduser authorization – one of • web • wap • sms
/result/payment_parameters/**method**	string	recommended payment method – one of • OPERATOR • ISP
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request
/result/transactions/transaction/**id**	string	id of identify transaction – reusable for follow up payment

6.1.2 operator-lookup

Action operator-lookup is used to determine an enduser's MNO from his MSISDN.

Request parameters (see 5):

name	mandatory?
action	yes
cp_*	no
digest	yes
merchant	yes
msisdn	yes
order	yes, unless a default has been configured on service setup
redirect	no
request_id	yes
url_callback	yes, unless a default has been configured on service setup
url_return	no, unless redirect=1 is used and no default has been configured on service setup

The response XML document contains the following elements:

xpath	type	description
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/redirect/**url**	string	URL where the enduser shall be redirected (only present with status 3)
/result/action_result/**status**	number	• 1 … failure • 3 … redirect required • 4 … validation failed • 5 … pending – result will be reported in callback
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request

The XML document contained in the data form field of the callback POST has the following elements:

xpath	type	description
/result/**action**	string	the action this result refers to
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/**status**	number	• 0 … success • 1 … failure
/result/custom_parameters/custom_parameter/**key**	string	name of custom parameter passed to the API call
/result/custom_parameters/custom_parameter/**value**	string	value of custom parameter passed to the API call
/result/customer/**country**	string	ISO 3166-1 Alpha 2 code
/result/customer/**id**	string	enduser's id aka alias
/result/customer/**language**	string	ISO 639-1 code
/result/customer/**msisdn**	string	enduser's MSISDN
/result/customer/**operator**	string	enduser's operator – see pay:smart operators for a listing
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request

6.1.3 start

Action start is used to initiate a one-off payment.

Request parameters (see 5):

name	mandatory?
action	yes
amount	no
artifact	no
channel	no
country	no

cp_*	no
customer_id	no
digest	yes
ip	no
landing_page	no
language	no
merchant	yes
method	no
msisdn	no
operator	no
order	yes, unless a default has been configured on service setup
preemptive_content	no
prompt_content_args	no
prompt_image_args	no
prompt_merchant_args	no
prompt_product_args	no
prompt_url_args	no
redirect	no
request_id	yes
shopper	no in general, yes for iGaming
send_content	no
service_name	yes
transaction	no
url_callback	yes, unless a default has been configured on service setup
url_return	yes, unless a default has been configured on service setup or redirect=0 is used

The response XML document contains the following elements:

xpath	type	description
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/redirect/**url**	string	URL where the enduser shall be redirected (only present with status 3)
/result/action_result/**status**	number	• 1 … failure • 3 … redirect required • 4 … validation failed • 5 … pending – result will be reported in callback
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request

The XML document contained in the data form field of the callback POST has the following elements:

xpath	type	description
/result/**action**	string	the action this result refers to
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/**detail_psp**	string	error code and/or error description from PSP
/result/action_result/**status**	number	• 0 … success • 1 … failure
/result/additional_results/additional_result/**key**	string	name of additional dynamic return parameter – see 6.2
/result/additional_results/additional_result/**value**	string	value of additional dynamic return parameter – see 6.2
/result/custom_parameters/custom_parameter/**key**	string	name of custom parameter passed to the API call
/result/custom_parameters/custom_parameter/**value**	string	value of custom parameter passed to the API call
/result/customer/**country**	string	ISO 3166-1 Alpha 2 code
/result/customer/**id**	string	enduser's id aka alias
/result/customer/**ip**	string	IP address of enduser's device
/result/customer/**language**	string	ISO 639-1 code
/result/customer/**msisdn**	string	enduser's MSISDN
/result/customer/**operator**	string	enduser's operator – see pay:smart operators for a listing
/result/payment_parameters/**channel**	string	enduser authorization technique – one of • web • wap • sms
/result/payment_parameters/**method**	string	payment method used – one of • OPERATOR • ISP
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request
/result/transactions/transaction/**amount**	number	requested transaction amount
/result/transactions/transaction/**billed_amount**	number	actual billed transaction amount
/result/transactions/transaction/**currency**	string	currency of billing transaction (ISO 4271 alphabetic code)
/result/transactions/transaction/**id**	string	id of billing transaction
/result/transactions/transaction/sms_message/**id**	string	id of billing SMS

/result/transactions/transaction/**status**	number	status of the payment transaction – see 10.1

6.1.4 start-subscription

Action *start-subscription* is used for signing up an enduser to a subscription service.

Typically, a successful signup implies that the enduser was billed once initially – on a failing initial payment the signup would also fail.

Nevertheless, with feature *unbilled subscription signups* it is also possible to successfully open a subscription – if configured for your *order* and supported by the PSP – even in case there is no initial payment involved or the enduser has not enough balance to fulfil it. These kinds of signups will be indicated in the XML document of the final callback via:

xpath	value	description
/result/action_result/**status**	2	subscription is active but there was no initial billing, or it failed

Request parameters (see 5):

name	mandatory?
action	yes
amount	no
artifact	no
channel	no
close_notification_url_callback	no
country	no
cp_*	no
customer_id	no
digest	yes
ip	no
landing_page	no
language	no
manage_subscription_url_callback	no
merchant	yes
method	no
msisdn	no
operator	no
order	yes, unless a default has been configured on service setup
preemptive_content	no
prompt_content_args	no
prompt_image_args	no
prompt_merchant_args	no
prompt_product_args	no
prompt_url_args	no
redirect	no
request_id	yes
shopper	no in general, yes for iGaming
send_content	no

service_category	no
service_name	yes
subscription_type	no
transaction	no
url_callback	yes, unless a default has been configured on service setup
url_return	yes, unless a default has been configured on service setup or redirect=0 is used

The response XML document contains the following elements:

xpath	type	description
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/redirect/**url**	string	URL where the enduser shall be redirected (only present with status 3)
/result/action_result/**status**	number	• 1 … failure • 3 … redirect required • 4 … validation failed • 5 … pending – result will be reported in callback
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request

The XML document contained in the data form field of the callback POST has the following elements:

xpath	type	description
/result/**action**	string	the action this result refers to
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/**detail**_psp	string	error code and/or error description from PSP
/result/action_result/**status**	number	• 0 … success – active subscription with successful first payment • 2 … unbilled – active subscription but no/failed first payment • 1 … failure
/result/additional_results/additional_result/**key**	string	name of additional dynamic return parameter – see 6.2
/result/additional_results/additional_result/**value**	string	value of additional dynamic return parameter – see 6.2
/result/custom_parameters/custom_parameter/**key**	string	name of custom parameter passed to the API call

/result/custom_parameters/custom_parameter/**value**	string	value of custom parameter passed to the API call
/result/customer/**country**	string	ISO 3166-1 Alpha 2 code
/result/customer/**id**	string	enduser's id aka alias
/result/customer/**ip**	string	IP address of enduser's device
/result/customer/**language**	string	ISO 639-1 code
/result/customer/**msisdn**	string	enduser's MSISDN
/result/customer/**operator**	string	enduser's operator – see pay:smart operators for a listing
/result/payment_parameters/**channel**	string	enduser authorization technique – one of • web • wap • sms
/result/payment_parameters/**method**	string	payment method used – one of • OPERATOR • ISP
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request
/result/subscription/definition/**amount**	number	amount to be charged per charging event (decimal number using period for separator)
/result/subscription/definition/**currency**	string	currency of subscription (ISO 4271 alphabetic code)
/result/subscription/definition/**event_count**	number	number of charging events per subscription period
/result/subscription/definition/**period_length**	number	number of subscription period units defining a subscription period
/result/subscription/definition/**period_type**	string	subscription period units: *month*, *week*, *day*
/result/subscription/**id**	string	subscription id
/result/subscription/**status**	number	subscription status – see 10.2
/result/transactions/transaction/**amount**	number	requested transaction amount
/result/transactions/transaction/**billed_amount**	number	actual billed transaction amount
/result/transactions/transaction/**currency**	string	currency of billing transaction (ISO 4271 alphabetic code)
/result/transactions/transaction/**id**	string	id of billing transaction
/result/transactions/transaction/sms_message/**id**	string	id of billing SMS
/result/transactions/transaction/**status**	number	status of the payment transaction – see 10.1

/result/transactions/transaction/**subscription_id**	string	subscription id this transaction belongs to

6.1.5 renew-subscription

Action renew-subscription is used for explicitly rebilling a previously successfully opened subscription in accordance with the underlying subscription's definition (subscription period, number of charging events during the period and amount charged).

Request parameters (see 5):

name	mandatory?
action	yes
cp_*	no
digest	yes
merchant	yes
order	yes, unless a default has been configured on service setup
preemptive_content	no
prompt_content_args	no
redirect	no
request_id	yes
send_content	no
subscription	yes
url_callback	yes, unless a default has been configured on service setup
url_return	no, unless redirect=1 is used and no default has been configured on service setup

The response XML document contains the following elements:

xpath	type	description
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/redirect/**url**	string	URL where the enduser shall be redirected (only present with status 3)
/result/action_result/**status**	number	• 1 … failure • 3 … redirect required • 4 … validation failed • 5 … pending – result will be reported in callback
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	request_id provided with the API request

The XML document contained in the data form field of the callback POST has the following elements:

xpath	type	description
/result/**action**	string	the action this result refers to
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status

/result/action_result/**detail_psp**	string	error code and/or error description from PSP
/result/action_result/**status**	number	• 0 … success • 1 … failure
/result/additional_results/additional_result/**key**	string	name of additional dynamic return parameter – see 6.2
/result/additional_results/additional_result/**value**	string	value of additional dynamic return parameter – see 6.2
/result/custom_parameters/custom_parameter/**key**	string	name of custom parameter passed to the API call
/result/custom_parameters/custom_parameter/**value**	string	value of custom parameter passed to the API call
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request
/result/subscription/definition/**amount**	number	amount to be charged per charging event (decimal number using period for separator)
/result/subscription/definition/**currency**	string	currency of subscription (ISO 4271 alphabetic code)
/result/subscription/definition/**event_count**	number	number of charging events per subscription period
/result/subscription/definition/**length**	number	number of subscription period units defining a subscription period
/result/subscription/definition/**type**	string	subscription period units: month, week, day
/result/subscription/**id**	string	subscription id
/result/subscription/**status**	number	subscription status – see 10.2
/result/transactions/transaction/**amount**	number	requested transaction amount
/result/transactions/transaction/**billed_amount**	number	actual billed transaction amount
/result/transactions/transaction/**currency**	string	currency of billing transaction (ISO 4271 alphabetic code)
/result/transactions/transaction/**id**	string	id of billing transaction
/result/transactions/transaction/sms_message/**id**	string	id of billing SMS
/result/transactions/transaction/**status**	number	status of the payment transaction – see 10.1
/result/transactions/transaction/**subscription_id**	string	subscription id this transaction belongs to

6.1.6 close-subscription

Action close-subscription is used whenever a previously opened subscription within pay:smart needs to be closed.

Request parameters (see 5):

name	mandatory?
action	yes
cp_*	no
digest	yes
merchant	yes
order	yes, unless a default has been configured on service setup
redirect	no

request_id	yes
subscription	yes
url_callback	yes, unless a default has been configured on service setup
url_return	no, unless *redirect=1* is used and no default has been configured on service setup

The response XML document contains the following elements:

xpath	type	description
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/redirect/**url**	string	URL where the enduser shall be redirected (only present with status 3)
/result/action_result/**status**	number	• 1 … failure • 3 … redirect required • 4 … validation failed • 5 … pending – result will be reported in callback
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request

The XML document contained in the data form field of the callback POST has the following elements:

xpath	type	description
/result/**action**	string	the action this result refers to
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/**detail_psp**	string	error code and/or error description from PSP
/result/action_result/**status**	number	• 0 … success • 1 … failure
/result/additional_results/additional_result/**key**	string	name of additional dynamic return parameter – see 6.2
/result/additional_results/additional_result/**value**	string	value of additional dynamic return parameter – see 6.2
/result/custom_parameters/custom_parameter/**key**	string	name of custom parameter passed to the API call
/result/custom_parameters/custom_parameter/**value**	string	value of custom parameter passed to the API call
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request
/result/subscription/definition/**amount**	number	amount to be charged per charging event (decimal number using period for separator)
/result/subscription/definition/**currency**	string	currency of subscription (ISO 4271 alphabetic code)
/result/subscription/definition/**event_count**	number	number of charging events per subscription period

/result/subscription/definition/**length**	number	number of subscription period units defining a subscription period
/result/subscription/definition/**type**	string	subscription period units: *month*, *week*, *day*
/result/subscription/**id**	string	subscription id
/result/subscription/**status**	number	subscription status – see 10.2

6.1.7 status

Action status is used for retrieving the current status of a transaction or subscription. This action also works for subscriptions under pay:periodic management (was formerly called *periodic-status*).

There is usually no callback for this action as the synchronous answer contains all information.

Request parameters (see 5):

name	mandatory?
action	yes
digest	yes
merchant	yes
request_id	yes
subscription	no, if transaction is given
transaction	no, if subscription is given

The response XML document contains the following elements:

xpath	type	description
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/**status**	number	• 0 … success • 1 … failure
/result/additional_results/additional_result[key="last_capture_try"]/value	timestamp with UTC offset	timestamp of latest renewal attempt
/result/additional_results/additional_result[key="**last_successful_capture**"]/value	timestamp with UTC offset	timestamp of latest successful renewal
/result/additional_results/additional_result[key="**next_renewal**"]/value	timestamp with UTC offset	timestamp of next scheduled renewal
/result/additional_results/additional_result[key="**subscription_enddate**"]/value	timestamp with UTC offset	timestamp of subscription cancelation
/result/additional_results/additional_result[key="**subscription_startdate**"]/value	timestamp with UTC offset	timestamp of subscription start
/result/customer/**country**	string	ISO 3166-1 Alpha 2 code
/result/customer/**id**	string	enduser's id aka alias
/result/customer/**language**	string	ISO 639-1 code
/result/customer/**msisdn**	string	enduser's MSISDN
/result/customer/**operator**	string	enduser's operator – see pay:smart operators for a listing

/result/payment_parameters/**channel**	string	enduser authorization technique – one of • web • wap • sms
/result/payment_parameters/**method**	string	payment method used – one of • OPERATOR • ISP
/result/payment_parameters/**order**	string	service id that has been passed initially via parameter *order* or has been automatically derived from configuration
/result/**request_id**	string	*request_id* provided with the API request
/result/subscription/definition/**amount**	number	amount to be charged per charging event (decimal number using period for separator)
/result/subscription/definition/**currency**	string	currency of subscription (ISO 4271 alphabetic code)
/result/subscription/definition/**event_count**	number	number of charging events per subscription period
/result/subscription/definition/**length**	number	number of subscription period units defining a subscription period
/result/subscription/definition/**type**	string	subscription period units: *month*, *week*, *day*
/result/subscription/**status**	number	subscription status – see 10.2
/result/transactions/transaction/**amount**	number	requested transaction amount
/result/transactions/transaction/**billed_amount**	number	actual billed transaction amount
/result/transactions/transaction/**currency**	string	currency of billing transaction (ISO 4271 alphabetic code)
/result/transactions/transaction/**id**	string	id of billing transaction
/result/transactions/transaction/sms_message/**id**	string	id of billing SMS
/result/transactions/transaction/**status**	number	status of the payment transaction – see 10.1
/result/transactions/transaction/**subscription_id**	string	subscription id this transaction belongs to

6.1.8 prompt

Action prompt is used to exchange information with the enduser. Most often this means that a simple free SMS is delivered. A typically use-case is the decoupled delivery of content that was billed before via *renew-subscription* or an automatically managed subscription, but in general this action can be used for whatever reason.

Note
Based on the flow, for one-off payments/initially started subscriptions, content may be more easily delivered combined via parameter *send_content* upon action *start/start_subscription*.

Request parameters (see 5):

name	mandatory?
action	yes
country	no
cp_*	no
customer_id	no
digest	yes
ip	no
language	no
merchant	yes
msisdn	no
operator	no
order	yes, unless a default has been configured on service setup
prompt_content_args	no
redirect	no
request_id	yes
subject	yes
subscription	no
transaction	no
url_callback	yes, unless a default has been configured on service setup
url_return	yes, unless a default has been configured on service setup or *redirect=0* is used

The response XML document contains the following elements:

xpath	type	description
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/redirect/**url**	string	URL where the enduser shall be redirected (only present with status 3)
/result/action_result/**status**	number	• 1 … failure • 3 … redirect required • 4 … validation failed • 5 … pending – result will be reported in callback
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request

The XML document contained in the data form field of the callback POST has the following elements:

xpath	type	description
/result/**action**	string	the action this result refers to
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status

/result/action_result/**detail_psp**	string	error code and/or error description from PSP
/result/action_result/**status**	number	• 0 … success • 1 … failure
/result/additional_results/additional_result/**key**	string	name of additional dynamic return parameter – see 6.2
/result/additional_results/additional_result/**value**	string	value of additional dynamic return parameter – see 6.2
/result/custom_parameters/custom_parameter/**key**	string	name of custom parameter passed to the API call
/result/custom_parameters/custom_parameter/**value**	string	value of custom parameter passed to the API call
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request
/result/transactions/transaction/**id**	string	id of transaction
/result/transactions/transaction/**status**	number	status of the transaction – see 10.1

6.1.9 refund

Action refund is used to pay back the billed amount of a successful payment initiated e.g. via action start or renew-subscription.

Request parameters (see 5):

name	mandatory?
action	yes
cp_*	no
digest	yes
merchant	yes
order	yes, unless a default has been configured on service setup
redirect	no
request_id	yes
transaction	yes
url_callback	yes, unless a default has been configured on service setup
url_return	no, unless redirect=1 is used and no default has been configured on service setup

The response XML document contains the following elements:

xpath	type	description
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/redirect/**url**	string	URL where the enduser shall be redirected (only present with status 3)
/result/action_result/**status**	number	• 1 … failure • 3 … redirect required

		• 4 … validation failed • 5 … pending – result will be reported in callback
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request

The XML document contained in the data form field of the callback POST has the following elements:

xpath	type	description
/result/**action**	string	the action this result refers to
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/**detail_psp**	string	error code and/or error description from PSP
/result/action_result/**status**	number	• 0 … success • 1 … failure
/result/custom_parameters/custom_parameter/**key**	string	name of custom parameter passed to the API call
/result/custom_parameters/custom_parameter/**value**	string	value of custom parameter passed to the API call
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request
/result/subscription/definition/**amount**	number	amount to be charged per charging event (decimal number using period for separator)
/result/subscription/definition/**currency**	string	currency of subscription (ISO 4271 alphabetic code)
/result/subscription/definition/**event_count**	number	number of charging events per subscription period
/result/subscription/definition/**length**	number	number of subscription period units defining a subscription period
/result/subscription/definition/**type**	string	subscription period units: *month*, *week*, *day*
/result/subscription/**id**	string	subscription id
/result/subscription/**status**	number	subscription status – see 10.2
/result/transactions/transaction/**amount**	number	requested transaction amount
/result/transactions/transaction/**billed_amount**	number	actual billed transaction amount
/result/transactions/transaction/**currency**	string	currency of billing transaction (ISO 4271 alphabetic code)
/result/transactions/transaction/**id**	string	id of billing transaction
/result/transactions/transaction/sms_message/**id**	string	id of billing SMS
/result/transactions/transaction/**status**	number	status of the payment transaction – see 10.1
/result/transactions/transaction/**subscription_id**	string	subscription id this transaction belongs to

6.1.10 customer_id-lookup

Action *customer_id-lookup* is used to determine an enduser's abstract alias from his MSISDN.

Request parameters (see 5):

name	mandatory?
action	yes
cp_*	no
digest	yes
merchant	yes
msisdn	yes
order	yes, unless a default has been configured on service setup
redirect	no
request_id	yes
url_callback	yes, unless a default has been configured on service setup
url_return	no, unless *redirect=1* is used and no default has been configured on service setup

The response XML document contains the following elements:

xpath	type	description
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/redirect/**url**	string	URL where the enduser shall be redirected (only present with status 3)
/result/action_result/**status**	number	• 1 … failure • 3 … redirect required • 4 … validation failed • 5 … pending – result will be reported in callback
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request

The XML document contained in the data form field of the callback POST has the following elements:

xpath	type	description
/result/**action**	string	the action this result refers to
/result/action_result/**code**	number	additional code of the reported status providing further details – see 9
/result/action_result/**detail**	string	verbose description of the status
/result/action_result/**status**	number	• 0 … success • 1 … failure
/result/custom_parameters/custom_parameter/**key**	string	name of custom parameter passed to the API call
/result/custom_parameters/custom_parameter/**value**	string	value of custom parameter passed to the API call
/result/customer/**country**	string	ISO 3166-1 Alpha 2 code

/result/customer/**id**	string	enduser's id aka alias
/result/customer/**language**	string	ISO 639-1 code
/result/customer/**msisdn**	string	enduser's MSISDN
/result/customer/**operator**	string	enduser's operator – see pay:smart operators for a listing
/result/**reference**	string	correlation id for matching callback with initiating API call
/result/**request_id**	string	*request_id* provided with the API request

6.2 Primary additional results

6.2.1 subscription_terminated

This additional result can be returned with value *true* for any action which is related to a subscription. It indicates that the subscription is closed now and is no longer usable for billing the enduser.

6.2.2 trial_subscription

Indicates with a value of *true*, that the subscription at hand has a special type. These subscriptions have their first payment attempt not during signup but deferred after some trial period has passed. The first payment attempt will happen during a usual renewal if the subscription was not closed meanwhile.

6.2.3 low_money

This flag is *true* for subscriptions that could not be billed initially during signup (e.g. because of insufficient balance) but can still be kept for renewal attempts.

6.2.4 avs_registered

When this entry is returned with value *true* it means that there was a successfully executed age-verification registration for the current enduser. Note that this activity was maybe subject to charge.

6.2.5 notification

This additional result is added to event responses with value true to clearly distinguish notifications from usual callbacks. See 7 for a detailed explanation of notifications and their accompanying secondary additional results.

6.2.6 activity_required

This entry will be present with value true to indicate the necessity of further activities to fulfil the transaction. Section 8 gives details about all possible activities and their corresponding secondary additional results.

6.2.7 proportional_capture

Indicates with a value of true that not the whole reserved amount was billed but only a requested proportional part of it. The corresponding values will be given in the transaction's section with amount (reserved amount) and *billed_amount* (actual billed amount).

7 NOTIFICATIONS FOR MANAGED RESOURCES

For some PSPs resp. use cases the management of a subscription, payment, SMS, etc. is handled directly by the PSP or DIMOCO. This is unlike to the case where the merchant manages those resources via explicitly triggering required actions like *start-subscription*, *renew-subscription and close-subscription.*

All automatically happening events for a managed resource like opt-in, renew and opt-out are reported to the merchant via so-called *notifications* that have **no** prior merchant request. This is in contrast to the manually triggered use cases where every *callback* from pay:smart **always** has a corresponding request from the merchant (correlated via unique identifier *reference* – see 2.1).

WARNING
It is very important that notifications are always verified for their authenticity by the merchant's server via the accompanied digest. Otherwise an attacker could generate any number of forged notifications that look perfectly valid!

The following notifications are delivered to statically defined URLs that can be configured for a specific *order* on service setup. Renewal- and close-notification URLs can also be provided dynamically with action *start-subscription* via parameters *manage_subscription_url_callback* and *close_notification_url_callback* (see 5).

All notifications are commonly indicated via:

xpath	value	description
/result/additional_results/additional_result[key="**notification**"]/value	true	outcome of externally triggered event without prior merchant request

7.1 start-subscription

This notification indicates that an enduser has been signed in to a subscription service externally – without a preceding explicit *start-subscription* request from the merchant. E.g. SMS opt-in for a service that was announced via advertisement.

7.2 renew-subscription

The outcome of every automatically performed billing event within a subscription service is announced to the merchant's system via this notification. Note that this can also happen for subscriptions that have been opened explicitly via action *start-subscription*.

7.3 close-subscription

This notification is triggered when an enduser was unsubscribed from a subscription service externally – without a preceding explicit *close-subscription* request from the merchant.

7.4 start

The outcome of every one-off payment that has been initiated externally is announced to the merchant's system via this kind of notification.

7.5 receive-sms-info

For every free SMS received from an enduser that can be correlated to a specifically configured *order* this notification will be generated.

In the response XML document, the following values are available:

xpath	type	description
/result/additional_results/additional_result[key="**signifier**"]/value	string	reference of received message – maybe needed for later API calls
/result/additional_results/additional_result[key="**sms_service**"]/value	string	service number where SMS was received
/result/additional_results/additional_result[key="**sms_text**"]/value	string	content of received SMS
/result/customer/**country**	string	ISO 3166-1 Alpha 2 code
/result/customer/**msisdn**	string	enduser's MSISDN
/result/customer/**operator**	string	enduser's operator – see pay:smart operators for a listing
/result/payment_parameters/**order**	string	correlated *order*

8 ADVANCED FLOWS

NOTICE
Almost all available pay:smart use cases can be handled solely with the standard communication flows as outlined in section 3. That means for a fully functional and complete pay:smart integration the advanced flows described below typically do **not** need to be implemented. You will be informed explicitly by DIMOCO if the implementation of an advanced flow is necessary for your integration.

For some use cases a more fine-grained integration of pay:smart to the merchant's portal is desirable. The additional flexibility gained comes at the cost of increased complexity and should be taken into account.

Advanced flows require the merchant's system to perform additional activities usually involving the enduser while a payment is still in progress. The type of activity that must take place is reported via an interim callback to the merchant's server. If the enduser was initially redirected to pay:smart, he is returned to the merchant's portal a*fter* the successful delivery of the callback.

Interim callbacks commonly indicate a necessary activity via:

xpath	value	description
/result/action_result/**status**	5	action pending (still ongoing)
/result/additional_results/additional_result[key="**activity_required**"]/value	true	merchant's system must perform an activity to complete the action

Flow-specific additional values in the result are explained individually below.

8.1 Early enduser return

In very rare cases payment flows may take a significant amount of time at the PSP until the final outcome becomes available. Potentially there is a meaningful activity that the enduser can do during this delay on the merchant's portal. This advanced flow enables the early return of the enduser to the merchant's portal even though the final callback was not delivered yet.

Additional values within interim callback:

xpath	value	description
/result/additional_results/additional_result[key="**activity**"]/value	delay-enduser	keep enduser busy until arrival of the final result callback (comes with a significant delay!)

8.2 SMS from enduser

Some use cases have the requirement that the enduser must manually send an SMS with a predefined text to a given service number. This advanced flow enables the merchant's system itself

to ask the enduser for this SMS as pay:smart would normally do. Thus, no redirect of the enduser to pay:smart is necessary.

Additional values within interim callback:

xpath	value	description
/result/additional_results/additional_result[key="**activity**"]/value	request-sms	request enduser to manually send an SMS
/result/additional_results/additional_result[key="**sms_text**"]/value	<text>	text that must be contained in SMS
/result/additional_results/additional_result[key="**sms_service**"]/value	<service-number>	service number where SMS must be sent to

After the SMS was submitted a follow-up callback will inform about the final outcome of the payment.

8.3 TAN entry on merchant's portal

For confirming a payment via TAN pay:smart usually displays a page to the enduser where he must submit the TAN previously sent to his mobile. With this advanced flow the merchant's system itself can ask for the TAN and submit it on behalf of the enduser to pay:smart. Thus, no redirect of the enduser to pay:smart is necessary.

Additional values within interim callback:

xpath	value	description
/result/additional_results/additional_result[key="**activity**"]/value	request-tan	TAN must be queried from enduser
/result/additional_results/additional_result[key="**response_url**"]/value	<URL>	URL where the TAN must be submitted to
/result/transactions/transaction/**id**	<id>	transaction id that must be submitted alongside

After the enduser entered the TAN on the merchant's portal it must then be submitted via server to server call to pay:smart as explained in 4.4.1 **but** to the URL specified in the interim callback. The following parameters apply:

- *result* … indicates if the enduser entered the TAN or aborted
  - 0 … TAN was given by enduser
  - 1 … enduser cancelled the TAN entry or a timeout happened (parameter tan can be omitted in this case)
- *tan* … as supplied by the enduser
- *transaction* … as given in the interim callback
- *digest* … over the concatenated unencoded values of *result*, *tan* (if any) and *transaction* (computation described in 4.4.1)

The synchronous response to this call contains a flat JSON document delivered with content-type application/json and the following elements:

key	type	description
accepted	boolean	• *true* … request was accepted, TAN will be checked, and result is delivered via callback • *false* … request could not be accepted due to a technical failure – see *detail*
detail	string	optional human readable explanation of the outcome

The asynchronous callback delivered afterwards is either a final callback about the outcome of the payment or again a *request-tan* interim callback demanding another TAN entry attempt.

8.4 On-demand content provisioning

When the content to deliver cannot be given upfront with the initiation of a payment (e.g. because it is expensive to create) then this flow can help. It will ask the merchant for the content via interim callback at the latest moment possible during the payment.

Additional values within interim callback:

xpath	value	description
/result/additional_results/additional_result[key="**activity**"]/value	provide-content	payment was confirmed and is ready to be (or was) captured so content must be given now to pay:smart for delivery
/result/additional_results/additional_result[key="**response_url**"]/value	<URL>	URL where the content must be given to
/result/transactions/transaction/**id**	<id>	transaction id that must be submitted alongside

The content must be submitted via server to server call to pay:smart as explained in 4.4.1 **but** to the URL specified in the interim callback. The following parameters apply:

- *result* … indicates if the content can be provided
  - 0 … given content shall be delivered
  - 1 … no content can be given, and payment shall be cancelled (parameter *prompt_content_args* can be omitted in this case)
- *prompt_content_args* … content provided as JSON document like e.g. {"client_login":"xyz"}
- *transaction* … as given in the interim callback
- *digest* … over the concatenated unencoded values of *prompt_content_args* (if any), *result* and *transaction* (computation described in 4.4.1)

The synchronous response to this call contains a flat JSON document delivered with content-type *application/json* and the following elements:

key	type	Description
accepted	boolean	• *true* … request was accepted, content will be delivered • *false* … request could not be accepted due to a technical failure – see *detail*
detail	string	optional human readable explanation of the outcome

The asynchronous callback delivered afterwards will indicate the final outcome of the payment.

8.5 Content acknowledgement

This flow allows the merchant to fulfil content delivery by his own means while the payment process itself is still handled by pay:smart. To enable this functionality an interim callback will be issued at the correct time during the payment indicating to the merchant that the content shall now be shipped. An acknowledging server to server call from the merchant afterwards will tell pay:smart to finalize or cancel the payment.

Additional values within interim callback:

xpath	value	description
/result/additional_results/additional_result[key="**activity**"]/value	ship-content	payment was confirmed and is ready to be (or was) captured so content must be shipped now by the merchant and acknowledged to pay:smart afterwards
/result/additional_results/additional_result[key="**response_url**"]/value	<URL>	URL where the acknowledgement must be submitted to
/result/transactions/transaction/**id**	<id>	transaction id that must be submitted alongside

The acknowledgement must be submitted via server to server call to pay:smart as explained in 4.4.1 **but** to the URL specified in the interim callback. The following parameters apply:

- *result* … indicates outcome of content delivery by merchant
  - 0 … successful content shipment
  - 1 … no content was shipped, and payment shall be cancelled
- *transaction* … as given in the interim callback
- *digest* … over the concatenated unencoded values of *result* and *transaction* (computation described in 4.4.1)

The synchronous response to this call contains a flat JSON document delivered with content-type *application/json* and the following elements:

key	type	Description
accepted	boolean	• *true* … acknowledgement was accepted • *false* … acknowledgement could not be accepted due to a technical failure – see *detail*
detail	string	optional human readable explanation of the outcome

The asynchronous callback delivered afterwards will indicate the final outcome of the payment.

8.6 Proportional capture of reserved amount

If the amount for a payment depends on the actual volume of content consumed by the enduser, this is the appropriate flow. It works for PSPs where a maximum amount can be reserved upfront, and later, when the volume of consumption is known, the proportional amount can be captured. After the enduser confirmed the payment and the maximum amount was successfully reserved an interim callback will be issued to the merchant that the content can now safely be consumed by the enduser. A server to server call from the merchant afterwards will tell pay:smart to finalize the payment with a proportional amount or to cancel it, e.g. if no consumption happened at all.

This flow is only available for one-off payments via action *start*. Parameter *amount* of action *start* specifies the maximum amount that shall be reserved.

Additional values within interim callback:

xpath	value	description
/result/additional_results/additional_result[key="**activity**"]/value	capture-amount	payment was confirmed and maximum amount was reserved; content can be consumed now, and proportional amount must be sent to pay:smart afterwards
/result/additional_results/additional_result[key="**response_url**"]/value	<URL>	URL where the amount must be submitted to
/result/transactions/transaction/**id**	<id>	transaction id that must be submitted alongside

The proportional amount must be submitted via server to server call to pay:smart as explained in 4.4.1 **but** to the URL specified in the interim callback. The following parameters apply:

- *result* … indicates outcome of content consumption
  - 0 … content was (fully or partially) consumed
  - 1 … no content was consumed, and payment shall be cancelled (parameter amount can be omitted in this case)
- *amount* … proportional amount that shall be captured (>0, <=max. amount)
- *transaction* … as given in the interim callback
- *digest* … over the concatenated unencoded values of *amount* (if any), *result* and *transaction* (computation described in 4.4.1)

The synchronous response to this call contains a flat JSON document delivered with content-type *application/json* and the following elements:

key	type	Description
accepted	boolean	• *true* … request was accepted, proportional amount will be captured • *false* … request could not be accepted due to a technical failure – see *detail*
detail	string	optional human readable explanation of the outcome

The asynchronous callback delivered afterwards will indicate the final outcome of the payment. In case an amount smaller than the maximum amount was captured, the following elements will apply:

xpath	value	Description
/result/action_result/**status**	0	action is declared successful also for a proportional capture
/result/additional_results/additional_result[key="**proportional_capture**"]/value	true	actual indicator that only a proportional amount was captured (see 6.2.7)
/result/transactions/transaction/**amount**	<maximum amount>	indicates the reserved maximum amount
/result/transactions/transaction/**billed_amount**	<billed amount>	actual billed transaction amount

9 ACTION RESULT CODES

Following result codes are used for indicating the reason of a failed action.

Important: Additional codes can be added with future versions of pay:smart.

name	code	description
FEATURE_UNAVAILABLE	4	e.g. sending content disallowed
ERROR_REQUEST_NO_DATA	100	request contains no (valid) data
ERROR_REQUEST_INVALID	101	invalid data: parsing or validation failed
ERROR_REQUEST_INVALID_VALUE	102	invalid/missing parameter
PARAM_MISSING_FORMAL	103	missing parameter formally declared
ERROR_REQUEST_CLIENT_AUTHORIZATION	111	merchant unauthorized, e.g. wrong credentials
ERROR_REQUEST_USER_EXISTS	121	enduser already enlisted/registered
ERROR_REQUEST_USER_AGE_VERIFICATION_REQUIRED	123	age of enduser must have been verified
ERROR_REQUEST_USER_AVS_REGISTRATION_REQUIRED	124	enduser must have been registered for age-verification
ERROR_REQUEST_ACCESS_DENIED	125	client not allowed to get this data
ERROR_REQUEST_MERCHANT_ORDER_AUTHORIZATION	126	*order* does not belong to merchant
ERROR_REQUEST_USER_TOO_MANY_DEVICES	128	shopper attempted to use too many devices for payment
ERROR_REQUEST_PAYMENT_TRANSACTION_INVALID	131	payment transaction not known
ERROR_REQUEST_ORDER_INVALID	133	*order* not known or data invalid/incomplete
ERROR_REQUEST_URL_MISSING	135	URLs missing
ERROR_REQUEST_SUBSCRIPTION_UNKNOWN	138	subscription not known
ERROR_REQUEST_PIN_INVALID	141	pin for web payment invalid
ERROR_REQUEST_CHANNEL_INVALID	142	authorization channel invalid/not applicable for this request
ERROR_REQUEST_CONCURRENCY	144	illegal concurrent or duplicate request
ERROR_INACTIVE_PAYMENT_METHOD	150	PSP is disabled in general or for the given *order*
ERROR_INACTIVE_MERCHANT	152	merchant is disabled
ERROR_INACTIVE_ORDER	153	*order* is disabled
ERROR_INACTIVE_USER	154	enduser is disabled
ERROR_LIMIT_PSP	206	enduser has reached limit or cover insufficient at PSP for this amount
ERROR_LIMIT_PSP_TRANSACTION	207	enduser has reached limit per transaction at PSP for this amount
ERROR_ACTION_IMPLEMENTATION	300	internal error
ERROR_ACTION_STATUS	301	requested action invalid since transaction has the wrong state
ERROR_ACTION_MUST_REDIRECT	302	only in combination with status 3 (redirect required)
ERROR_ACTION_UNSUPPORTED	303	constellation of *action*, *channel*, etc. not available for this PSP
ERROR_ACTION_LOAD_EXCEEDED	304	load boundaries exceeded
ERROR_ACTION_TECHNICAL_TIMEOUT	309	a technical timeout occurred
ERROR_AMOUNT_GENERAL	400	money or currency related error

name	code	description
ERROR_AMOUNT_INVALID	401	amount is wrong
ERROR_AMOUNT_NOT_ALLOWED	404	given amount is not allowed
ERROR_PSP_UNAVAILABLE	500	PSP system unavailable
ERROR_PSP_COMMUNICATION	501	communication failed
ERROR_PSP_SETUP	502	configuration invalid
ERROR_PSP_OTHER_ERROR	503	more specific error not available
ERROR_PSP_LOAD_EXCEEDED	504	too many requests towards PSP
ERROR_PSP_CUSTOMER_UNKNOWN	510	not a customer of PSP
ERROR_PSP_CUSTOMER_CANNOT_PAY	511	customer not activated
ERROR_PSP_GOODS_NOT_ALLOWED	512	this type of goods cannot be sold to this customer
ERROR_PSP_PAYMENT_INFO_INVALID	513	missing/invalid data like MSISDN
ERROR_PSP_CUSTOMER_CANCEL	515	enduser cancelled payment
ERROR_PSP_SUBSCRIPTION_ERROR	516	subscription unknown, invalid, etc.
ERROR_PSP_SERVICE_UNKNOWN	517	service number not configured etc.
ERROR_PSP_CONCURRENT_AUTHORIZATION	518	capture or cancel all previous authorized transactions before starting a new transaction
ERROR_PSP_CUSTOMER_TIMEOUT	519	enduser did not react (click buy/cancel, answer sms etc.)
ERROR_PSP_CUSTOMER_PAY_ACTIVATION	520	customer not registered for payment or payment not activated
ERROR_PSP_SUBSCRIPTION_CLOSED	521	subscription exists but is (now) closed
ERROR_PSP_CUSTOMER_NO_BALANCE	522	prepaid customer has insufficient, or no money left
ERROR_PSP_ALREADY_SUBSCRIBED	523	duplicate subscription not possible for same service
ERROR_PSP_UNSUCCESSFUL	525	activity failed for an unspecific but harmless reason at PSP
ERROR_PSP_FRAUD_DETECTED	526	fraudulent activity prevented
ERROR_PSP_KYC_MISMATCH	527	enduser KYC check failed (e.g. via GSMA mobile connect)
ERROR_PSP_RESERVATION_ERROR	528	reservation prematurely aborted by PSP
ERROR_PSP_MVNO_NOT_SUPPORTED	529	mobile virtual network operators cannot be used
ERROR_PSP_CUSTOMER_RESTART	530	enduser demanded restart of payment
ERROR_PSP_PREPAID_NOT_ALLOWED	531	prepaid contracts cannot be used for current use-case
ERROR_PSP_PARTIAL_BILLING	535	only a partial amount instead of the whole requested amount was billed
ERROR_PSP_AGE_VERIFICATION	540	customer cannot get this (adult) content since age is not verified
ERROR_PSP_SMS_DLR_FAILED	550	sms was not delivered to handset
ERROR_PSP_SMS_EXPIRED	551	sms expired and was not delivered
ERROR_SUBSCRIPTION_OTHER_ERROR	700	general error during subscription handling – usually configuration or implementation related
ERROR_SUBSCRIPTION_RETRY_ABSOLUTE_LIMIT	701	absolute number of unsuccessful retries for subscription reached – it must thus be closed
ERROR_SUBSCRIPTION_TIME_LIMIT	702	a time-based charging limit has been reached (e.g. max 3 chargings per day) –

name	code	description
		renewals should be spread over whole period
ERROR_SUBSCRIPTION_RETRY_TIME_LIMIT	703	subscription has been retried too often and further retries are not allowed at the moment
ERROR_SUBSCRIPTION_PSP_LIMIT	704	PSP specific limit has been reached and further chargings are not allowed at the moment
ERROR_SUBSCRIPTION_PERIOD_LIMIT	705	subscription has already been charged completely for the current period

10 STATUS CODES

10.1 Transaction status

value	description
-1	transaction prepared
0	transaction in progress
1	
2	
3	
4	transaction successful
5	
6	transaction refunded

10.2 Subscription status

value	description
-1	subscription activation failed
0	subscription activation in progress, renewals not allowed yet
1	
2	
3	subscription active, renewals allowed
4	
5	subscription terminated, renewals no longer allowed

REFERENCES

We will gladly answer any questions you have about our products and services.

DIMOCO Payments GmbH
Campus 21 Businesspark Wien Süd
Europaring F15/302
2345 Brunn am Gebirge/Vienna
Austria

Tel: +43 1 33 66 888-0
Fax: +43 1 33 66 888-9000
Email: sales@dimoco.eu

[pay:smart operators]: https://services.dimoco.at/operators/list/current