Loyalty API

Revisions

Revision	Author	Date	Comments
1	cenee	16/03/2016	Translated document
2	jbbinard	15/04/2016	Purse info documentation
3	vlaine	20/04/2016	Add money information on loyalty member Fix send event paragraph
4	cenee	02/05/2016	Updated the “buy” event type example
5	vlaine	13/05/2016	Added immediate benefit
6	jbbinard	19/05/2016	Added read only valid immediate benefits
7	vlaine	24/05/2016	Added cash receipt fields details
8	vlaine	02/06/2016	Added activationDate property on person
9	vlaine	21/06/2016	Added UTC information on all dates and times

Table of contents

This document describes the programming interface, or API, of Digitaleo's platform for managing loyalty. This API allows you to create a loyalty program and to configure it with rules for earning points, offers to be converted, and vouchers to be used by customers.

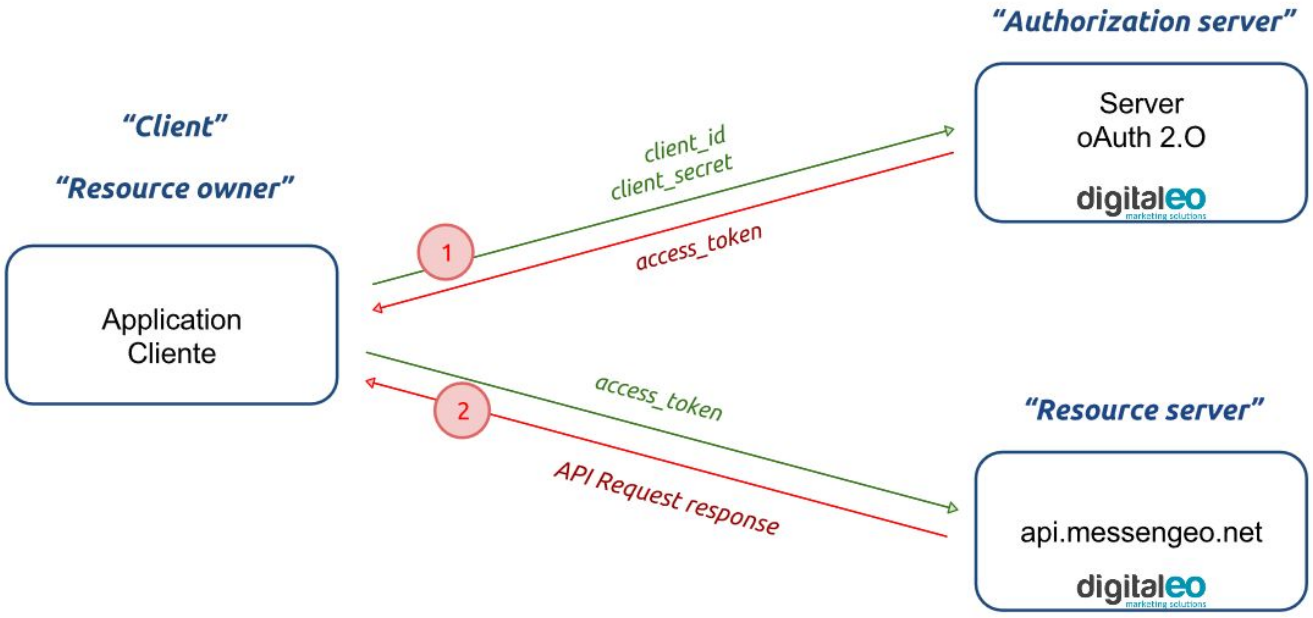
1. Overview

This API is RESTfull. It is based on various resources of which the details are provided further on in this document.

The purpose of this first section is to help you understand the various types of calls to our APIs, regardless of the resource.

1.1. Authentication

Authentication to our APIs is based on the oAuth 2.0 protocol. Each call to our APIs has to contain an access_token that the client application will have requested beforehand from the Digitaleo authorization server:



1.1.1. Retrieving the application ids

To retrieve a client_id and a client_secret, you must declare an application in the Digitaleo platform.
For this,

- Connect to app.digitaleo.com
- Click on the Parameters menu
- Go to the API tab

1.1.2. Retrieve an authentication token

The client must perform a POST request with the following parameters:

- grant_type: The value must be "client_credentials" for this type of authorization
- client_id: The id of the application (client)
- client_secret: The secret key of the application (client)

Note: The client_id and client_secret will be sent to you.

The URL for retrieving a token is as follows

```
https://oauth.messengeo.net/token
```

Example of an HTTP request

```
POST /token HTTP/1.1
Host: oauth.messengeo.net
Content-Type: application/x-www-form-urlencoded

client_id=51612c780b4dbaea8f81995beccbcfec08969d0e&
client_secret=p280edbd76d510c41990cbe5e6108c7e&
grant_type=client_credentials
```

Example of a request with Curl

```
curl https://oauth.messengeo.net/token
 -d 'client_id=51612c780b4dbaea8f81995beccbcfec08969d0e'
 -d 'client_secret=p280edbd76d510c41990cbe5e6108c7e'
 -d 'grant_type=client_credentials'
```

Return

if successful, the authorization server will return a code 200 HTTP response of which the body will contain the following JSON flow

```
{
   "access_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2...",
   "expires_in":"3600",
   "token_type":"bearer",
   "scope":"basic",
}
```

Description of the various fields:

Field	Description
access_token	The token issued by the authorization server. *Note: The size of the token can range up to 50,000 characters*
expires_in	The lifespan in seconds of the token issued
token_type	The type of token. The Digitaleo server only supports the "bearer" type
scope	The scope of the token

If one of the parameters is not correct, the authorization server will return a code 400 http response (HTTP/1.1 400 Bad Request) of which the body will contain the following json flow:

```
{
   "error":"invalid_client",
   "error_description":"The client credentials are invalid",
}
```

1.1.3. Using the authorization token (access_token)

The authorization token is sent to the API in the header of the HTTP request and more particularly in the header “Authorization: Bearer”. Note that the “Authorization: Bearer” is case-sensitive.

Example of an HTTP request

```
GET /rest/campaigns HTTP/1.1

Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.digitaleo.com
```

Example of a request with Curl

```
curl -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2..."
https://api.digitaleo.com/loyalty/v1/person/search/Doe
```

1.2. Pagination

Pagination for the results is standardized for the resources that support it. It has the form of 2 lines in the response header and of a set of parameters available when the API is called.

For example, searching for a person using the following GET call:

```
{URL_CustomerMaster}/person/search/Doe
```

Results returned in paginated form:

Response header:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.0
Link:{URL_CustomerMaster}/person/search/doe?page=2&pageSize=100&sortBy=email&sortDirection=Descending;rel="next"
```

(For the Person resource, the default sorting is on the email property)

Content of the response:

```
[
    {
        "id": "54fec93029d29f0b2421f204",
        "lastName": "Doe",
        "firstName": "John",
        "email": "john.doe@unknown.com",
        "mobile": "+33 6 00 00 00"
    },
    {
        "id": "54fec93029d29f0b2421f203",
        "lastName": "Doe",
        "firstName": "Jeanne",
        "email": "jeanne.doe@unknown.com",
        "mobile": "+33 6 00 00 01 01"
    }
]
```

If the number of results in the request exceeds the size of a results page (specified using the “pageSize” parameter), the Link:...;rel=”next” is inserted with the url and the parameters allowing the next page to be selected.

List of parameters that affect pagination:

Parameter	Function
page	Index of the page to be returned (base 1)
pageSize	The number of elements per page to be returned
sortBy	The name of the field to be used to sort the results
sortDirection	The sort direction (possible values: “Ascending” and “Descending”)

The “prev” and “next” links in the headers are inserted only if they apply. As such, there is never a “prev” link when consulting the first page of the results. Likewise, there is never a “next” link when consulting the last page.

1.3.Updating resources

To update a resource, a complete resource must be provided, i.e. with the modified and unmodified fields.

Example of updating an offer:

```
PUT {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/loyaltyoffer/{id}
{
  "name":"offer 1",
  "label":"Super offer 1",
  "description": "lorem ipsum .......",
  "mastercode": "AZERTY",
  "startdate": "2015-01-01T01:01:01",
  "enddate" : "2015-03-01T01:01:01",
  "imageuri" : "http://demo.digitaleo.com/images/demo?3456789",
  "thumbnailuri": "http://demo.digitaleo.com/images/demo?3456789&mode=thumbnailuri",
  "expirationtolerance": "7",
  "converter": {
    "minimumpoint": "1000",
    "scheduletypename":  "monthly",
    "priority": "1"
 }
}
```

Updating a resource returns HTTP code 200 if the operation was a success and the modified resource, otherwise HTTP code 304 and the error message are returned.

1.4. Return codes

The HTTP response code is contained:

- in the HTTP header,
- in the content of the response in the case of an error.

The return codes are based on the HTTP return codes:

- 2XX - The call to the API unfolded correctly
- 4XX – The call to the API has an error in its parameters.

Codes with success:

- 200 OK: everything went well
- 201 Created: Resource created
- 204 No Content: Resource updated or deleted

The error codes that you are likely to see are the following:

- 304 Not Modified: Error during updating or deleting (the resource is not modified)
- 400 Bad Request: Missing or incorrect parameter
- 401 Unauthorized: Authentication failed
- 403 Forbidden: Access to the requested location is prohibited
- 404 Not Found: Unknown method or method not indicated
- 405 Method Not Allowed: You are not authorized to use the method that you are requesting
- 414 Request-URI Too Long: Your request is too large, please shorten it
- 417 Expectation Failed: The required parameters are either missing or are incorrect
- 500 Internal Server Error: Unidentified error

For example, if the authentication token is no longer valid for the following request:

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
https://api.messengeo.net/rest/resource
```

The header of the HTTP response will be

```
< HTTP/1.1 401 Unauthorized
< Date: Fri, 06 Mar 2015 21:32:06 GMT
< Server: Apache/2.2.16 (Debian)
< X-Powered-By: PHP/5.3.3-7+squeeze15
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 46
< Content-Type: application/json
```

while the body of the HTTP response will be

```
{
   "status": 401,
   "message": "Authorization has been denied for this request."
}
```

2. Loyalty

A loyalty program defines all of the resources (rules, offers, vouchers, etc.) for one or several brands.

Customers who are members of the loyalty program can earn points according to events linked to the brand.

Loyalty is covered by the “Loyalty” API:
https://api.digitaleo.com/loyalty/v1

This loyalty API uses the customer database managed by the “CustomerMaster” API:
https://api.digitaleo.com/customermaster/v1

Note: these two APIs are versioned (current versions: v1), the version number is part of the URL.

2.1 Members of the loyalty program

Members who belong to the loyalty program are persons managed by the CustomerMaster API (with last name, first name, mobile telephone number, email address, mailing address, etc.), and who have additional data that is specific to the loyalty program (loyalty card identifier, number of points, vouchers, etc.

2.2 Events

Events concern an end user as well as a brand. There are various events: in-store purchase, store check-in, activity on social networks, entering personal information, …

The events are deposited on an Azure service bus, in order to be processed by the loyalty system.

2.3 Rules for earning points

The rules for earning points allow you to reward a customer in the form of points according to his actions.

Examples of rules:
- for any in-store purchase, €1 spent earns 1 point
- a “like” on Facebook earns 50 points

2.4 Offers

A loyalty program also includes a catalog of offers. An offer is a definition of an advantage in exchange for the customer's points. Offers have validity dates in terms of their application.

Examples of offers:
- exchanging 1000 points for a €10 reduction voucher
- exchanging 2000 points for a 25% reduction voucher

Creating vouchers from offers can be:
- automatic, on the same day every month
- automatic, every time the customer earns points
- on request (from the customer or a user managing the loyalty program)

2.5 Vouchers

A voucher is an instance of an offer. It is granted to a customer in exchange for points, according to the catalog of offers available. When the voucher is granted, the customer's number of points is decremented.

The voucher is to be used in one go; it "burns" when it is used. Vouchers also have validity dates in terms of their use.

2.6 Immediate benefits

An immediate benefit is a discount which can be used directly when purchasing. The discount must be calculated by the caller and the result must be set in the cash register

2.7 Purse

The purse can be used by a member to buy somethings. It's incremented by the earn rules created by the loyalty manager.
The purse information could have a maximum defined and the caller must check it before processing payment. When the purse use transaction is created, the API will check it also.

3. Reference documentation

3.1. Persons who are members of the loyalty program

3.1.1 List of properties of the Person resource

Property	Description
Id	Identifier of the person
Email	Email
Phone	Telephone number
Mobile	Mobile telephone number
Fax	Fax number
TwitterId	Twitter id
FacebookId	Facebook id
Civility	Civility
FirstName	First name
LastName	Last name

Address1	Part 1 of the mailing address
Address2	Part 2 of the mailing address
ZipCode	Postal code
City	City
State	Province
Country	Country
Longitude	Longitude (double, can be null)
Latitude	Latitude (double, can be null)
Birthdate	Date of birth (Date, can be null)
Loyalty	Data linked to the loyalty program (cf. hereinbelow)
Store	Data linked to the reference/preferred point of sale of the customer

Description of the “Loyalty” property:

Property	Description
points	Number of points of the person
money	Amount of money earned by the person
loyaltyMemberId	Identifier of the member within the loyalty program
loyaltyProgramId	Identifier of the loyalty program that the member belongs to
virtualCardNumber	Identifier of the virtual loyalty card
physicalCardNumber	Identifier of the physical data card (can be null)
activationDate	Date when the member activated his account (can be empty or non existent)

Description of the “Store” property:

Property	Description
storeReferenceCode	Identifier of the store

3.1.2 Searching for a person

```
GET {URL_customerMaster}/person/search/{searchTerm}
```

URL parameter	Description
searchTerm	Terms to be searched for among: ● Last name ● First name ● Email address ● Mobile telephone number ● Identifier of the virtual loyalty card ● Identifier of the physical loyalty card

This action returns a list of “person summary” resources including the fields Last name, first name, email, mobile, avatar and Id of the person.

List of properties of the “person summary” resource

Property	Description
Id	Id of the person in Customer Master
FirstName	First name of the person
LastName	Last name of the person
Email	Email of the person
Mobile	Telephone number of the person
Avatar	URL of the avatar of the person

By default, the first 100 results are returned and are sorted by last name (LastName). If more results are available, the header of the response contains a "Link" value that can contain a maximum of 2 values.

Example, with a search for the term “j”

```
{URL_CustomerMaster}/person/search/j?page=2&pageSize=20&sortBy=email&sortDirectionDescending
```

Response header:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.0
Link:{URL_CustomerMaster}/person/search/j?page=1&pageSize=20&sortBy=email&sortDirection=Desc
ending;rel="prev"
Link:{URL_CustomerMaster}/person/search/j?page=3&pageSize=20&sortBy=email&sortDirection=Desc
ending;rel="next"
```

3.1.3 Searching for a person with one or more known criteria

```
GET {URL_customerMaster}/person/searchByCriteria?firstName=john&lastName=doe&email=
john.doe@test.local&mobile=06010101010
```

The URL parameters are optional, however at least one parameter must be present. If several parameters are present, the research takes them into account (research of the "AND" type)

URL parameter	Description
firstName	First name
lastName	Last name
email	email
mobile	Portable telephone number

The search returns a list of "person summary" resources including the fields Last name, first name, email, mobile, avatar and Id of the person.

List of properties of the "person summary" resource

Property	Description
Id	Id of the person in Customer Master
FirstName	First name of the person
LastName	Last name of the person
Email	Email of the person
Mobile	Telephone number of the person
Avatar	URL of the avatar of the person

By default, the first 100 results are returned and are sorted by last name (LastName). If more results are available, the header of the response contains a "Link" value that can contain a maximum of 2 values.

3.1.4 Reading a person

```
GET {URL_customerMaster}/person/{id}
```

This method returns the complete person corresponding to the identifier passed as a parameter.

URL parameter	Description
id	Id of the person to be read

3.1.5 Updating a person

```
PUT {URL_customerMaster}/person/{id}/source/{source}/brand/{brand}
```

URL parameter	Description
id	Id of the person to be updated
source	Source at the origin of the modification (in the case of a cash register system, use the "store" source)
brand	Brand at the origin of the modification (all of the brands are defined when the parameters for the loyalty program are set)

The properties that can be updated are:

Property	Description
Email	Email
Phone	Telephone number
Mobile	Mobile telephone number
Fax	Fax number
Cvility	Civility
FirstName	First name
LastName	Last name
Address1	Part 1 of the mailing address
Address2	Part 2 of the mailing address
ZipCode	Postal code
City	City
State	Province
Country	Country
Birthdate	Date of birth (Date)

3.1.6 Attaching a physical card to a person/Replacing the physical card

```
POST {URL_LOYALTY}/loyaltymember/{personId}/physicalcard/{cardNumber}
```

This requires the identifier of the person as well as the identifier of the physical card to be attached.

Property	Description
personId	Identifier of the person
cardNumber	Number of the physical loyalty card

If the loyalty account already contains a physical card, an error is returned.

To update the number of a physical loyalty card for a customer who already has a physical card attached, the following method must be used:

```
PUT {URL_Loyalty}/loyaltymember/{personId}/physicalcard/{cardNumber}
```

Property	Description
personId	Identifier of the person
cardNumber	Number of the physical loyalty card

3.2. Events

3.2.1 Structure of events

Events are described in the form of a JSON flow and have a fixed common structure as well as a variable portion according to the type of event.

The structure of an event is as follows:

```
{
        “version”: “1.0”,
        "eventType": "XXX",
        "eventSource": "XXX",
        "brand": "XXX",
        "eventDate": "",
        "personIdentifier": "",
        "eventData": {
         }
}
```

Property	Description
version	The version of the structure of the event. Must be “1.0”
eventType	Type of event. The possible values are: “updateProfile”, “storeCheckin”, “facebookLikePage”, “facebookLikePost”, “selfie”, “buy”
eventSource	The source of the event, i.e. the type of system that generated this event. The possible values are: “mobile”, “facebook”, “clienteling”, “store”, “webSite”
brand	The brand concerned by the event
eventDate	The date and time of the event. The expected format is the en-US format: YYYY/MM/dd hh:mm:ss. All dates and times must be send in UTC
personIdentifier	The identifier of the person concerned by the event. This identifier depends on the source of the event. cf.: Person Identifier table according to the source
eventData	JSON structure depending on the type of event. The presence of the property is required but the content or the absence of content depends on the type of event.

Person Identifier according to the source

Source type	Identifier type	Description
mobile	Id of the person in the CustomerMaster system	In the framework of a mobile application where the end user is authenticated and the user's profile is read from the CustomerMaster system, the id to be used in the events is the id of the profile
clienteling	Id of the person in the CustomerMaster system	In the framework of a clienteling application where the profile of the end user is read from the CustomerMaster system, the id to be used in the events is the id of the profile
webSite	Id of the person in the CustomerMaster system	In the framework of the loyalty website where the profile of the end user is read from the CustomerMaster system, the id to be used in the events is the id of the profile
store	Number of the loyalty card (virtual or physical) or id of the person in the CustomerMaster system	In the framework of a cash register system, the identifier to be used is either the loyalty card number (virtual or physical), or the id of the person in the CustomerMaster system

3.2.2 Format of the “updateProfile” event

```
{
        “version”: “1.0”,
        "eventType": "updateProfile",
        "eventSource": "clienteling", // The event source (store, webSite, etc, etc, etc)
        "brand": "DressingShop", // The brand associated to the source
        "eventDate": "2015-06-08 11:30:00", // The event date (in en-US format, UTC)
        "personIdentifier": "XXX",
        "eventData": { // The updateProfile data
            "email": "",
            "phone": "",
            "mobile": "",
            "fax": "",
            "civility": "",
            "firstName": "",
            "lastName": "",
            "address1": "",
            "address2": "",
            "zipCode": "",
            "city": "",
            "state": "",
            "country": "",
            "birthdate": "",
            "avatar":  ""
        }
}
```

3.2.3 Format of the “storeCheckin” event

```
{
        “version”: “1.0”,
        "eventType": "storeCheckin",
        "eventSource": "mobile",     // The event source (store, web site, etc, etc, etc)
        "brand": "DressingShop",     // The brand associated to the source
        "eventDate": "",             // The event date (in en-US format, UTC)
        "personIdentifier": "XXX",
        "eventData": {
             "userLatitude": 48.098645,
              "userLongitude": -1.699255
        }
}
```

The “userLatitude” and “userLongitude” properties are used to validate the position of the user with respect to the position of the store.

3.2.4 Format of the selfie event

```
{
    “version”: “1.0”,
    "eventType": "selfie",
    "eventSource": "mobile",   // The event source (store, web site, etc, etc, etc)
    "brand": "Dressing-Shop",  // The brand associated to the source
    "eventDate": "",           // The event date (in en-US format, UTC)
    "personIdentifier": "XXX",
    "eventData": { }
}
```

3.2.5 Format of cash register receipts

```
{
    "version": "1.1",
    "eventType": "buy",
    "eventSource": "", // The event source (store, web site, etc, etc, etc)
    "brand": "dressingshop", // The brand associated to the source
    "eventDate": "", // The event date (in en-US format, UTC)
    "personIdentifier": "", // The person identifier.
    "eventData": { // The data of the buy event
        "ticketId": "123456789",
        "physicalCardNumber" : "123456789",
        "grossAmount": "213.23", // TTC
        "immediateBenefits": [
            {
                "id": "123456789",
                "amount": "10"
            }
        ], // List of the immediate benefits chosen by the end user
        "purseTransactions": [
          "65975068"
        ],
        "store": "", // The store where the purchase has been done
        "items": [
            {
                "brand": "", // The brand
                "productLabel": "", // The product label
                "productReference": "", // The product reference
                "productFamily": "", // The product family
                "tags": [],
                "unitAmount": "123.23",
                "quantity": "1",
                "initialAmount": "123.23", // The initial amount (en-US format, TTC)
                "amount": "123.23" // The final amount of the product (en-US format, TTC)
            },
            {
                "brand": "", // The brand
                "productLabel": "", // The product label
                "productReference": "", // The product reference
                "productFamily": "", // The product family
                "tags": [],
                "unitAmount": "50.00",
                "quantity": "2",
                "initialAmount": "125.00", // The final amount (en-US format, TTC)
                "amount": "100.00" // The final amount of the product (en-US format, TTC)
            }
        ]
    }
}
```

Note: all amounts must be provided by considering the euro as the currency.

Field name	Description	Value

version	The format version.	Must be “1.1”
eventType	The event type	Must be “buy”
eventSource	The event source.	Must be “store”
brand	The brand name associated with the auth token used to send the event	string
eventDate	The event date in en-US format. All dates and times must be send in UTC	Date (2016-05-24 12:34:23)
personIdentifier	The person identifier, could be the person id (loaded from API) or the physical card number	string
ticketId	The ticket id if existing	string
physicalCardNumber	The physical card number of the person	string
grossAmount	The gross amount included the discount if any	Number in en-US format (2.6)
immediateBenefits	Json array with the immediate benefit(s) used with its id and the amount calculated from the immediate benefit data	{ “id”: “123456789”, “amount”: “24.7” }
purseTransactions	Json array with the purse transaction ids. See §3.5 for more information about purse transaction	id
store	The store name or id	string
items	Json array with the list of items	Array
items.brand	Optional The brand associated to the item	string
items.productLabel	Optional The product label as visible by the customer	string
items.productReference	Optional The product reference	string
items.productFamily	Optional The product family	string
tags	Json array with tags	String array
unitAmount	The unit amount of the item	Number in en-US format (23.98)
quantity	The quantity of the item	Integer
initialAmount	The initial amount of the item (Before any discount of immediate benefit)	Number in en-US format (24.12)
amount	The final amount paid by the customer. It will be used to calculate the earns	Number in en-US format

		(76.98)

3.2.6 Sending an event to Digitaleo

Sending an event to Digitaleo is done through the “event” resource of the CustomerMaster API. The call should be made with a valid token issued by the Digitaleo’s oAuth server.

```
POST {URL_CustomerMaster}/event/
```

The POST request content should be the event to be processed by the system.
The API respond with a HTTP status code 204 No Content

Example: Sending an event through the event resource in C#

```
string customerMasterUrl = "https://api.digitaleo.com/customermaster/v1/event";
string oAuthToken = GetOAuthToken();
string jsonContent = "{ ... }";

using (System.Net.WebClient webclient = new System.Net.WebClient())
{
        webclient.Headers.Add(System.Net.HttpRequestHeader.Authorization, “Bearer “ + oAuthToken);
        webclient.UploadString(customerMasterUrl, jsonContent);
}
```

3.3. Offers

3.3.1 List of properties of the LoyaltyOffer resource

Property	Description
$type	Type of offer (LoyaltyPercentOffer or LoyaltyClassicOffer)
Id	Identifier of the offer
Name	Name of the offer
Label	Commercial name of the offer
Terms	Terms of the offer
Description	Description of the offer
MasterCode	Master code
ImageUri	Uri of the associated visual (visual hosted in Digitaleo's content library)

ThumbnailUri	Uri of the thumbnail of the associated visual (visual hosted in Digitaleo's content library)
StartDate	Start date of the validity of the offer, in UTC
EndDate	End date of the validity of the offer in UTC
VoucherStartDate	Start date of the validity of the vouchers coming from the offer (can be null, in this case the date of issue of the voucher is used) in UTC
VoucherEndDate	End date of the validity of the voucher coming from the offer in UTC
VoucherValidityDays	Number of days of validity of the vouchers coming from the offer starting from the *VoucherStartDate* date
Converter	Defines the options that allow points to be converted into a voucher: ● MinimumPoint: Number of points required to obtain the offer ● ScheduleTypeName: name of the scheduler to use (Daily, Monthly, Instant or OnDemand) ● Priority: in the case of an auto offer (isAuto), priority for the offer with respect to the others

Automatic conversion of an offer into a voucher for a loyalty member is evaluated either once a month (ScheduleTypeName = "monthly") or once a day (ScheduleTypeName = Daily). When automatic conversion is activated, an automatic process evaluates the number of points of each loyalty member with respect to the number of points required to benefit from the offer and grants a corresponding voucher to the member if the offer is valid for this member.

In the case of an OnDemand offer, the offer is converted into a voucher following action from a human (Seller or customer, for example).
In the case of an Instant offer, the offer is converted into a voucher as soon as the customer has reached the required number of points.
If the offer is of the "LoyaltyPercentOffer" type the following additional field is returned:

Property	Description
percent	Percentage of the number of points to be credited in Euros or as an immediate discount, in exchange for all of the points.

Example of a proportional offer:
Offer defined at 2% of the points. If a customer with 1000 points wants to use it, he will obtain a €20 discount voucher in exchange for his 1000 points.

3.3.2 List of offers

This method returns the list of offers of the loyalty program, regardless of their status.

```
GET {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/loyaltyoffer
```

Paramètre URL	Description
loyaltyProgramId	Identifier of the loyalty program carrying the offers to be read

This method accepts the optional URL parameters:

URL parameter	Description	Valeur
isValid	makes it possible to retrieve only the offers that are currently valid (validity verified according to the dates)	*true* or *false*
availableWithPoints	makes it possible to retrieve the offers that require a maximum number of points specified	maximum number of points required
status	Status of the offer: Pending (in progress), Expired, NotBegun (Upcoming)	One of the values from among Pending, Expired, NotBegun
scheduleType	Type of scheduling for the offer: OnDemand, Instant, Daily or Monthly	
contextualMemberTotalPoints	Number of points to be used in order to generate the texts of proportional offers	Number of points to be used in order to generate the texts of proportional offers

Example: Reading currently-valid offers that require a maximum of 2000 points. All of the valid offers that require 2000 points **or less** will be returned:

```
GET
{URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/loyaltyoffer?isValid=true&availableWithPoints=2000
```

3.3.3 Reading an offer

This method returns the corresponding offer if it exists, with all of its properties.

```
GET {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/loyaltyoffer/{id}
```

URL parameter	Description
loyaltyProgramId	Identifier of the loyalty program carrying the offer to be read
id	Identifier of the offer to be read

3.3.4 Creating an offer

```
POST {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/loyaltyoffer
```

Paramètre URL	Description
loyaltyProgramId	Identifier of the loyalty program for which the offer is to be created

POST parameters	Type	Description
$type	string	Type of offer (LoyaltyPercentOffer or LoyaltyClassicOffer). Must be the first field of the JSON flow
Name	string (max 150 characters)	Name of the offer
Label	string (max 250 characters)	Commercial name of the offer
Terms	string	Terms of the offers
Description	string	Description of the offer
MasterCode	string	External code, for example the code of the offer in TCPOS
ImageUri	string	Uri of the associated visual (visual hosted in Digitaleo's content library)
ThumbnailUri	string	Uri of the thumbnail of the associated visual (visual hosted in Digitaleo's content library)
StartDate	DateTime	Start date of the validity of the offer in UTC
EndDate	DateTime	End date of the validity of the offer in UTC
VoucherStartDate	DateTime	Start date of the validity of the vouchers coming from the offer (can be null, in this case the start date is the date of issue of the voucher) in UTC
VoucherEndDate	DateTime	End date of the validity of the vouchers coming from the offer in UTC
VoucherValidityDays	int	Number of days of validity of the vouchers coming from the offer starting from the *VoucherStartDate* date
Converter	array	Defines the options that allow points to be converted into an offer: • MinimumPoint: Minimum number of points to obtain the offer • ScheduleTypeName: Name of the scheduler to use (Daily, Monthly, OnDemand, Instant) • Priority: Priority of the offer with respect to the others

3.3.5 Modifying an offer

```
PUT {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/loyaltyoffer/{id}
```

URL parameter	Description
loyaltyProgramId	Identifier of the loyalty program carrying the offer to be modified
id	Identifier of the offer to be modified

For the PUT parameters, refer to the parameters for creating an offer.

3.3.6 Deleting an offer

```
DELETE {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/loyaltyoffer/{id}
```

Paramètre URL	Description
loyaltyProgramId	Identifier of the loyalty program carrying the offer to be deleted
id	Identifier of the offer to be deleted

3.4 Vouchers

3.4.1 List of properties of the LoyaltyVoucher resource

A voucher is created using an offer, and includes a portion of the properties.

Property		Description
Id	string	Identifier of the voucher
LoyaltyOfferId	string	Id of the offer that the voucher comes from
Name	string	Name of the offer that the voucher comes from
Label	string	Commercial name of the offer that the voucher comes from
Terms	string	Terms of the offer that the voucher comes from
Description	string	Description of the offer that the voucher comes from
MasterCode	string	External code of the offer that the voucher comes from
ImageUri	string	Uri of the associated visual (visual hosted in Digitaleo's content library)
StartDate	Datetime	Start date of the validity of the voucher in UTC
EndDate	Datetime	End date of the validity of the voucher in UTC
BurnDate	Datetime	Date of use of the voucher (DateTime, null if voucher is not used) in UTC
Code	string	EAN13 code of the voucher used to generate a barcode.

3.4.2 Reading a voucher

This method returns the corresponding voucher if it exists, with all of its properties.

```
GET {URL_Loyalty}/loyaltyvoucher/{id}
```

URL parameter	Description
id	Identifier of the voucher to be used

3.4.3 List of vouchers of a member of the loyalty program

This method returns all of the vouchers of a member, regardless of their status (whether or not already used, whether or not the date is valid, etc.).

```
GET {URL_Loyalty}/loyaltyMember/{personId}/loyaltyvoucher
```

URL parameter	Description
personId	Identifier of the person

This method accepts optional URL parameters:

URL parameter	Description	Value
status	Makes it possible to retrieve only vouchers according to their status. The default value is “Usable”	● Usable ● AlreadyUsed ● Expired ● NotYetUsable

3.4.4 Creating a voucher for a member of the loyalty program

A voucher is created for a loyalty member, using an offer.

A voucher can be used if it has not already been used, and if the date of use is between the voucher's “StartDate” and “EndDate” values.

```
POST {URL_Loyalty}/loyaltyMember/{personId}/loyaltyvoucher/{loyaltyOfferId}
```

URL parameter	Description
personId	Identifier of the person
loyaltyOfferId	Identifier of the offer to be used to create a voucher

The POST call does not contain any data. The return contains the voucher created.

Result	HTTP code	JSON content
The user does not have enough points	412(PreconditionFailed)	{ "resultCode": 1, "resultText": "NotEnoughPoints" }
The offer is not usable yet	412 (PreconditionFailed)	{ "resultCode": 2, "resultText": "TooEarly" }
The offer is no longer usable	412 (PreconditionFailed)	{ "resultCode": 3, "resultText": "TooLate" }
The offer is unknown	412 (PreconditionFailed)	{ "resultCode": 5, "resultText": "UnknownOffer" }
The type of offer is not valid	412 (PreconditionFailed)	{ "resultCode": 6, "resultText": "InvalidType" }
Generic error	412 (PreconditionFailed)	{ "resultCode": 99, "resultText": "UnknowError" }

3.4.5 Using a voucher

Vouchers can only be used in one go. Once used, the “burnDate” field is populated with the date of use, and the history is filled in.

An offer can be converted into a voucher if the date of conversion is between the offer's “StartDate” and “EndDate” values. A tolerance period can be applied (value in days for “ExpirationTolerance”).

```
PUT {URL_Loyalty}/loyaltyvoucher/{voucherCode}
```

URL parameter	Description
voucherCode	Voucher code to be used to retrieve the voucher (See field “Code” of resource Voucher)

The PUT call does not contain any data.

The PUT call returns the following data:

Result	HTTP code	JSON content
Voucher used successfully	200 (Ok)	{ "resultCode": 0, "resultText": "Ok" }
The voucher is not valid yet	304 (NotModified)	{ "resultCode": 2, "resultText": "TooEarly" }
The voucher is no longer valid	304 (NotModified)	{ "resultCode": 3, "resultText": "TooLate" }
The voucher has already been used	304 (NotModified)	{ "resultCode": 4, "resultText": "AlreadyUsed" }
The voucher is not known	304 (NotModified)	{ "resultCode": 5, "resultText": "UnknownVoucher" }
Generic error	304 (NotModified)	{ "resultCode": 99, "resultText": "UnknowError" }

3.5 Purse

3.5.1 Get purse informations of a member

This method returns purse information for a member.

```
GET {URL_Loyalty}/loyaltypurse/loyaltymember/{personId}
```

URL parameter	Description
personId	Identifier of the person

List of properties of the “purse information” resource

Property	Description
availableAmount	Total amount of member’s purse
maxAmount	Maximum of usable amount of member’s purse

3.5.2 Using purse

A part or the totality of the purse amount can be used for one purchase. In order to use the purse of a member, a call to the API should be made. This call will check the purse amount asked and create a transaction in the system. The transaction id is returned by the system and must be set in the associated event of type “buy”.

```
POST {URL_Loyalty}/loyaltypurse/loyaltymember/{personId}
```

URL parameter	Description
personId	Identifier of the person

The POST request should be made with those parameters

POST parameters	Type	Description
purseAmount	double	The amount of the purse to be used. It must be less than or equal to the availableAmount AND to the maxAmount

The POST request returns the following transaction:

POST parameters	Type	Description
id	string	The transaction id. It must be set in the “buy” event
IsPending	bool	A value indicating if the transaction has been validated by the system
amount	double	The used amount of purse
dateCreated	DateTime	The transaction creation date in UTC
dateUpdated	DateTime	The transaction update date in UTC

3.6. Immediate benefit

3.6.1 List all immediate benefits

This method returns the list of immediate benefits of the loyalty program

```
GET {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/immediatebenefit
```

Paramètre URL	Description
loyaltyProgramId	Identifier of the loyalty program carrying the immediate benefit to be read

This method accepts the optional URL parameters:

URL parameter	Description	Value
page	The page number to be returned	
pageSize	The page size	Default value is 100
sort	The field to be used to sort the immediate benefits	Allowed values are “endDate” and “startDate”
sortDirection	The sort direction to be applied	Allowed values are “ascending” and “descending”
isValidOnly	True to filter only usable immediate benefits	Default value is false

Example 1 : Reading immediate benefits with a page size of 50 items sorted by enddate:

```
GET
{URL_tropicloyalty}/loyaltyProgram/{loyaltyProgramId}/immediatebenefit?page=1&pageSize=50&sort=enddate&sortDirection=descending
```

Example 2 : Reading only valid immediate benefits :

```
GET {URL_tropicloyalty}/loyaltyProgram/{loyaltyProgramId}/immediatebenefit?isValidOnly=true
```

3.6.2 Reading an immediate benefit

This method returns the corresponding immediate benefit if it exists, with all of its properties.

```
GET {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/immediatebenefit/{id}
```

URL parameter	Description
loyaltyProgramId	Identifier of the loyalty program carrying the immediate benefit to be read
id	Identifier of the immediate benefit to be read

The returned properties are

Properties	**Type**	**Description**
Id	string	Immediate benefit id
Name	string	Name of the immediate benefit
Label	string	Commercial name of the immediate benefit
Description	string	Description of the immediate benefit
ImageUri	string	Uri of the associated visual (visual hosted in Digitaleo's content library)
ThumbnailUri	string	Uri of the thumbnail of the associated visual (visual hosted in Digitaleo's content library)
StartDate	DateTime	Start date of the validity of the immediate benefit in UTC
EndDate	DateTime	End date of the validity of the immediate benefit in UTC
PercentAmount	number	Percent of the discount
MaxAmount	number	Maximum amount of the discount
CanBeCombinedWithOtherRules	boolean	A value indicating if the rules could be evaluated when this

		immediate benefit is used.
IsPublic	boolean	A value indicating if the immediate benefit should be displayed in the clienteling and end user application
ImageUri	string	Uri of the associated visual (visual hosted in Digitaleo's content library)
ThumbnailUri	string	Uri of the thumbnail of the associated visual (visual hosted in Digitaleo's content library)

3.6.3 Creating an immediate benefit

```
POST {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/immediatebenefit
```

Paramètre URL	Description
loyaltyProgramId	Identifier of the loyalty program for which the immediate benefit is to be created

POST parameters	Type	Description
$type	string	Must be LoyaltyImmediateBenefit
Name	string	Name of the immediate benefit
Label	string	Commercial name of the immediate benefit
Description	string	Description of the immediate benefit
ImageUri	string	Uri of the associated visual (visual hosted in Digitaleo's content library)
ThumbnailUri	string	Uri of the thumbnail of the associated visual (visual hosted in Digitaleo's content library)
StartDate	DateTime	Start date of the validity of the immediate benefit in UTC
EndDate	DateTime	End date of the validity of the immediate benefit in UTC
PercentAmount	number	Percent of the discount
MaxAmount	number	Maximum amount of the discount
CanBeCombinedWithOtherRules	boolean	A value indicating if the rules could be evaluated when this immediate benefit has been used.

IsPublic	boolean	A value indicating if the immediate benefit should be displayed in the clienteling and end user application
ImageUri	string	Uri of the associated visual (visual hosted in Digitaleo's content library)
ThumbnailUri	string	Uri of the thumbnail of the associated visual (visual hosted in Digitaleo's content library)

3.6.4 Modifying an immediate benefit

```
PUT {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/immediatebenefit/{id}
```

URL parameter	Description
loyaltyProgramId	Identifier of the loyalty program carrying the immediate benefit to be modified
id	Identifier of the immediate benefit to be modified

For the PUT parameters, refer to the parameters for creating an immediate benefit.

3.6.5 Deleting an immediate benefit

```
DELETE {URL_Loyalty}/loyaltyProgram/{loyaltyProgramId}/immediatebenefit/{id}
```

Paramètre URL	Description
loyaltyProgramId	Identifier of the loyalty program carrying the immediate benefit to be deleted
id	Identifier of the immediate benefit to be deleted

Copyright