Fleet Integration API

Overview

This document define interface for partners to integrate with fleetAPI.

Changes history

Date	Version	Changer	Changes
2020-03-03	1.0.0	Paweł Gamrot	Initial version
2020-03-31	1.0.1	Ireneusz Olchawski	Status endpoint excluded from authorization
2020-04-01	1.1.0	Paweł Gamrot	User must have special right: IntegrationAccess to use this API Data for status endpoint is cached
2020-04-01	1.1.0	Ireneusz Olchawski	Added IntegrationAccess to enpoints
2020-04-01	1.1.0	Ireneusz Olchawski	Added caching for status endpoint
2020-05-11	1.2.0	Paweł Gamrot	Add mode parameter to endpoint `/terminals/accesscryptogram`
2020-05-19	1.3.0	Paweł Gamrot	Add `terminalId` field to `AccessCryptogram` object
2020-06-01	1.4.0	Ireneusz Olchawski	Add multi hsm access
2020-06-08	1.4.0	Maciej Wiosło	Add `includeGitInfo` field to status endpoint
2020-12-17	1.5.0	Paweł Gamrot	Add `fleets` field to `AccessCryptogram` object

HTTP status codes

The HTTP standard provides over 70 status codes to describe the return values. We don't need them all, but there should be used at least a mount of 10.

- 200 – OK – Eyerything is working
- 201 – OK – New resource has been created
- 204 – OK – The resource was successfully deleted
- 304 – Not Modified – The client can use cached data
- 400 – Bad Request – The request was invalid or cannot be served. The exact error should be explained in the error payload. E.g. „The JSON is not valid“

- 401 – Unauthorized – The request requires an user authentication
- 403 – Forbidden – The server understood the request, but is refusing it or the access is not allowed.
- 404 – Not found – There is no resource behind the URI.
- 422 – Unprocessable Entity – Should be used if the server cannot process the enitity, e.g. if an image cannot be formatted or mandatory fields are missing in the payload.
- 500 – Internal Server Error – API developers should avoid this error. If an error occurs in the global catch blog, the stracktrace should be logged and not returned as response.

Version information

Version : 1.5.0

URI scheme

Host : fleetintegrationapi.dmt.com.pl
BasePath : /v1
Schemes : HTTPS

Tags

- Authentication : Authentication is based on user name and password\
  Refresh token should be used to renew JWT, so after authorizationToken expired refreshToken should be used in endpoint /authentication/refreshTokens to get new authorizationToken.

Produces

- `application/json`

Paths

Authenticates user and returns token data

```
POST /authentication/login
```

Parameters

Type	Name	Description	Schema
Body	**loginModel** *required*	Object with login data	Login

Responses

HTTP Code	Description	Schema
200	Login Success	Authentication
default	Error code (400, 403, 500 etc)	ResponseCode

Tags

- Authentication

Example HTTP response

Response default

```
{
  "code" : 2,
  "message" : "wrong request parameters",
  "errors" : [ "path parameter P1 not defined" ]
}
```

Refreshes user's tokens

```
GET /authentication/refreshTokens
```

Responses

HTTP Code	Description	Schema
200	Login Success	Authentication
default	Error code (400, 403, 500 etc)	ResponseCode

Tags

- Authentication

Example HTTP response

Response default

```
{
  "code" : 2,
  "message" : "wrong request parameters",
  "errors" : [ "path parameter P1 not defined" ]
}
```

Return terminal access cryptogram

```
GET /terminals/accesscryptogram
```

Parameters

Type	Name	Description	Schema	Default
Query	**mode** *optional*	Result type mode. Available values: * 1 - only AccessCryptogram.cryptogram field is returned * 2 - only AccessCryptogram.userCryptogram and AccessCryptogram.passwordCryptogram fields are returned Parameter not set means default value which is 1.	enum (1, 2)	`1`
Query	**tid** *required*	Terminal identifier assigned by the partner (TID)	string	

Responses

HTTP Code	Description	Schema
200	OK	AccessCryptogram
default	Error code (400, 403, 500 etc)	ResponseCode

Tags

- terminals

Example HTTP response

Response 200

```
{
  "terminalId" : 123,
  "cryptogram" :
"ODlmNjY3OWUtNWZkYS00NDYwLWIzNGUtNGFiNzIyMThlOGI4AAAAADEwYWJiNGZjLWZmMWMtNDUyOC
04YjhhLWJmYmI1ZGNmNDczNQAAAAA=",
  "userCryptogram" :
"ODlmNjY3OWUtNWZkYS00NDYwLWIzNGUtNGFiNzIyMThlOGI4AAAAAA==",
  "passwordCryptogram" :
"MTBhYmI0ZmMtZmYxYy00NTI4LThiOGEtYmZiYjVkY2Y0NzM1AAAAAA==",
  "fleets" : [ "UTA", "DKV" ]
}
```

Response default

```
{
  "code" : 2,
  "message" : "wrong request parameters",
  "errors" : [ "path parameter P1 not defined" ]
}
```

Return service version and date and time of service start

```
GET /utilities/status
```

Description

Data for this endpoint is cached

Parameters

Type	Name	Description	Schema
Query	**includeGitInfo** *optional*	Git repository info (default is False)	string

Responses

HTTP Code	Description	Schema
200	OK	ApiStatus
default	Error code (400, 403, 500 etc)	ResponseCode

Tags

- Utilities

Example HTTP response

Response 200

```
{
  "version" : "1.2.3",
  "startupDate" : "2019-07-30T11:23:00.000+0000",
  "machineName" : "DEV-TEST1",
  "componentsStatuses" : [ {
    "name" : "DB1",
    "type" : "DATABASE",
    "status" : "OK",
    "requestDate" : "2019-07-30T12:01:00.000+0000",
    "responseTime" : 0.123
  } ],
  "git" : [ {
    "hash" : "abc1234",
    "branch" : "master"
  } ]
}
```

Response default

```
{
  "code" : 2,
  "message" : "wrong request parameters",
  "errors" : [ "path parameter P1 not defined" ]
}
```

Definitions

AccessCryptogram

Terminal access cryptogram data. Based on mode parameter from endpoint `/terminals/accesscryptogram` can be returned in two formats:

- 1 - only cryptogram field
- 2 - userCryptogram and passwordCryptogram fields

TerminalId field is allways returned.

Name	Description	Schema
cryptogram *optional*	Encrypted user and terminal password (encrypted access string in base64 format). Access string is concatenation of terminal transactionUser (filled to 40 characters with 0x00) and terminal transactionPassword (filled to 40 characters with 0x00). Encryption is done using 3DES in CBC mode with secret key (double lenght 3DES key) **Pattern** : `"^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?$"`	string (byte)
fleets *optional*	Array includes list of active fleets for terminal	< string > array
passwordCryptogram *optional*	Encrypted terminal password (encrypted password access string in base64 format). Password access string is terminal transactionPassword (filled to 40 characters with 0x00). Encryption is done using 3DES in CBC mode with secret key (double lenght 3DES key) **Pattern** : `"^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?$"`	string (byte)
terminalId *optional*	Terminal identifier assigned by the fleetAPI	integer
userCryptogram *optional*	Encrypted terminal user (encrypted user access string in base64 format). User access string is terminal transactionUser (filled to 40 characters with 0x00). Encryption is done using 3DES in CBC mode with secret key (double lenght 3DES key) **Pattern** : `"^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?$"`	string (byte)

ApiStatus

Status of the API

Name	Description	Schema
componentsStatuses *optional*		< ComponentStatus > array
git *optional*		GitInfo
machineName *optional* *read-only*	Machine name	string
startupDate *optional* *read-only*	Date and time of service start	string (date-time)
version *optional* *read-only*	Version	string

Authentication

Authentication model - response for login action

Name	Schema
authorizationToken *optional*	Token
loggedUserContext *optional*	LoggedUserContext
refreshToken *optional*	Token

ComponentStatus

Represents component status

Name	Description	Schema
name *optional* *read-only*	component name	string
requestDate *optional* *read-only*	Date and time of checking component status	string (date-time)
responseTime *optional* *read-only*	Response time	string
status *optional* *read-only*	component status (OK, FAIL)	string
type *optional* *read-only*	component type (DATABASE, API, ...)	string

GitInfo

Git repository info

Name	Description	Schema
branch *optional* *read-only*	Branch name	string
hash *optional* *read-only*	Commit hash	string

LoggedUserContext

Logged user's data

Name	Description	Schema
login *optional* *read-only*	User's login	string
userId *optional* *read-only*	User's identifier	integer

Login

User login data

Name	Schema
password *optional*	string (password)
username *optional*	string

ResponseCode

Response code.

Name	Description	Schema
code *required*	Error code. Available values: * 1 - internal error * 2 - wrong request parameters (wrong parameters in URL'u) * 3 - wrong body of request (wrong json object/method body) * 4 - invalid operation (operation cannot be performed with current context and data)	integer
errors *optional*		object
message *required*	Error description	string

Token

Authorization token's data

Name	Description	Schema
expires *optional* *read-only*	Token's expiration date	string (date-time)
token *optional* *read-only*	Token	string

Security

Bearer

For accessing the API a valid JWT token must be passed in all the queries in the 'Authorization' header.

A valid JWT token is generated by the API and returned as answer of a call to the endpoint /authentication/login giving a valid user & password.

The following syntax must be used in the 'Authorization' header :\
Bearer xxxxxx.yyyyyyy.zzzzzz

Type : apiKey
Name : Authorization
In : HEADER