Swarm Store Setup API

30/06/2014

Current implementations

The POS gateway has various setup pages for the POS systems it integrates with. Currently, these setup pages use different technologies which might make it difficult for Swarm to integrate them into their own dashboard.

OAuth pages

As some POS systems prefer OAuth authentication the gateway implements an HTML form which redirects the user to the POS provider's webpage, and a landing page where the user is redirected from the POS provider's webpage. When the user lands on the landing page the gateway receives the OAuth token as a URL parameter.

- https://pos-gateway.swarm-mobile.com/swarm/shopify.html
- https://pos-gateway.swarm-mobile.com/swarm/api/kounta.html

HTML Forms

Registration pages for some POS systems are simple HTML forms, and the gateway processes the POST arguments sent by the browser when registering the stores.

- https://pos-gateway.swarm-mobile.com/swarm/revelsystems.html

Webservice Ready

Some POS setup pages are static HTML pages using jQuery to call the gateway's store setup web service. These should be ready to be integrated into Swarm's dashboard.

- https://pos-gateway.swarm-mobile.com/swarm/lightspeedpro.html
- https://pos-gateway.swarm-mobile.com/swarm/rics.html

Webservice concept

PUT /swarm/api/shopify/account

Request:

```
{
    "code": "abcdef123456",
    "shop_url": "swarmtest.myshopify.com"
}
```

Response:

```
[
    {
        "store_id": 1,
        "name": "Shopify Shop"
    }
]
```

PUT /swarm/api/kounta/account

Request:

```
{
    "code": "abcdef123456"
}
```

Response:

```
[
    {
        "store_id": 1,
        "name": "Kounta Account"
    },
    {
        "store_id": 2,
        "name": "Second Kounta Account"
    }
]
```

PUT /swarm/api/revel/account

Request:

```
{
    "username": "testswarmmobile",
    "apikey": "abc123",
    "apisecret": "def345",
    "division": true
}
```

Response:

```
[
    {
        "store_id": 1,
        "name": "Revel Establishment"
    },
    {
        "store_id": 2,
        "name": "Second Revel Establishment"
    }
]
```

PUT /swarm/api/rics/account

Request:

```
[
    {
        "serial_num": 5555,
        "login": "swarm",
        "password": "swarm1234",
        "store_code" "10001"
    }
]
```

Response:

```
[
    {
        "store_id": 1,
        "name": "Rics #1 Store",
        "store_code": "10001"
    }
]
```

GET /swarm/api/rics/info

Request:

```
{
    "serial_num": 5555,
    "login": "swarm",
    "password": "swarm1234",
    "store_code" "10001"
}
```

Response:

```
{
    "name": "Rics #1 Store",
    "store_code": "10001"
}
```

PUT /swarm/api/lspro/account

Request:

```
{
    "user": "username",
    "token": "88888888-8888-8888-8888-888888888888"
}
```

Response:

```
[
    {
        "store_id": 1,
        "name": "Store Name"
    },
    {
        "store_id": 2,
        "name": "Second store Name"
    }
]
```

Error response

If something goes wrong, the REST service will return a JSON response with HTTP status code either

- 400 Bad Request - The request couldn't be served, e.g. because user provided invalid credentials
- 500 Internal Server Error - Catastrophic failure while serving the request

Example:

```
{
    "error_type": "bad_credentails",
    "error_message":"Error while reading remote data, Inner-Error: User  was not
found or doesn't have access"
}
```