NOOXY Service Framework

Started by Yves Chen, 10, Mar, 2018

Document Overview

1. Orientation
2. Architecture
3. serverside modure
4. clientside modure
5. Service, ServiceSocket and API
6. Activities and ActivitySocket(Client socket)

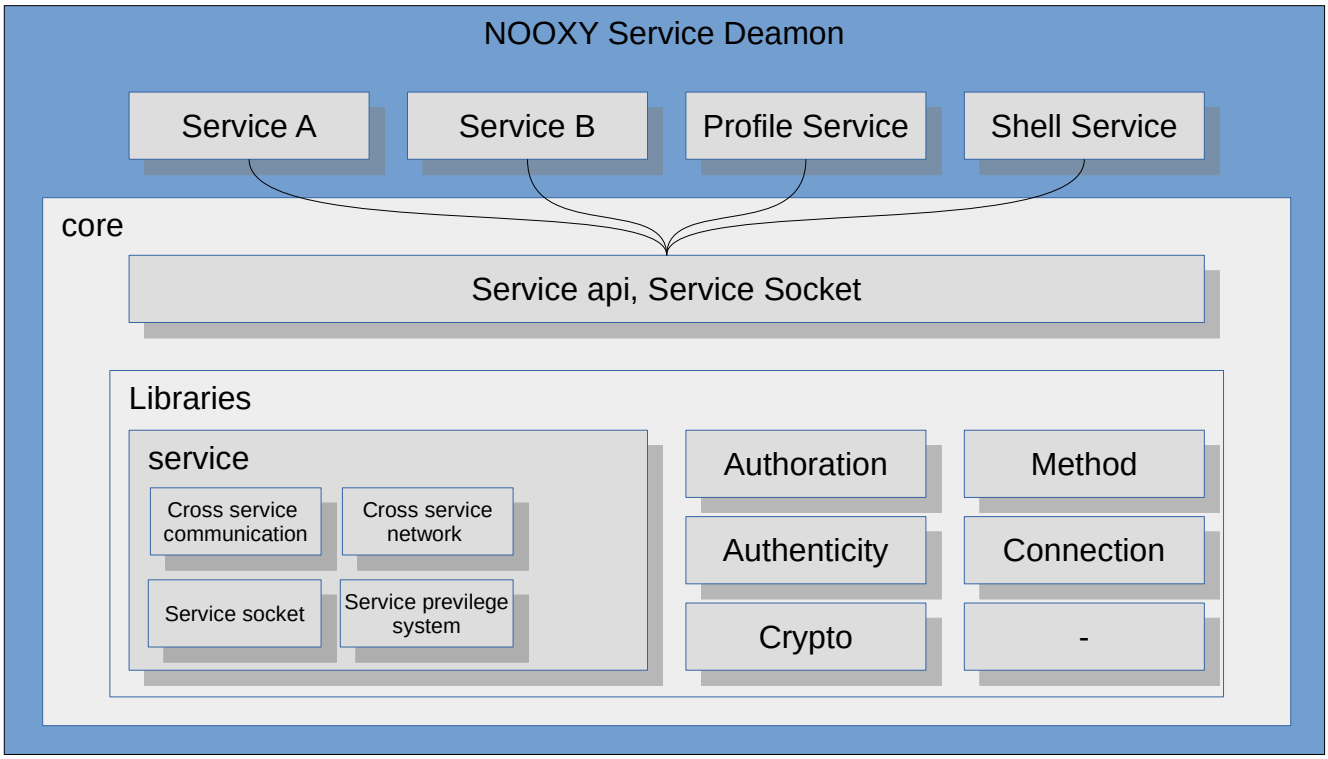
Orientation

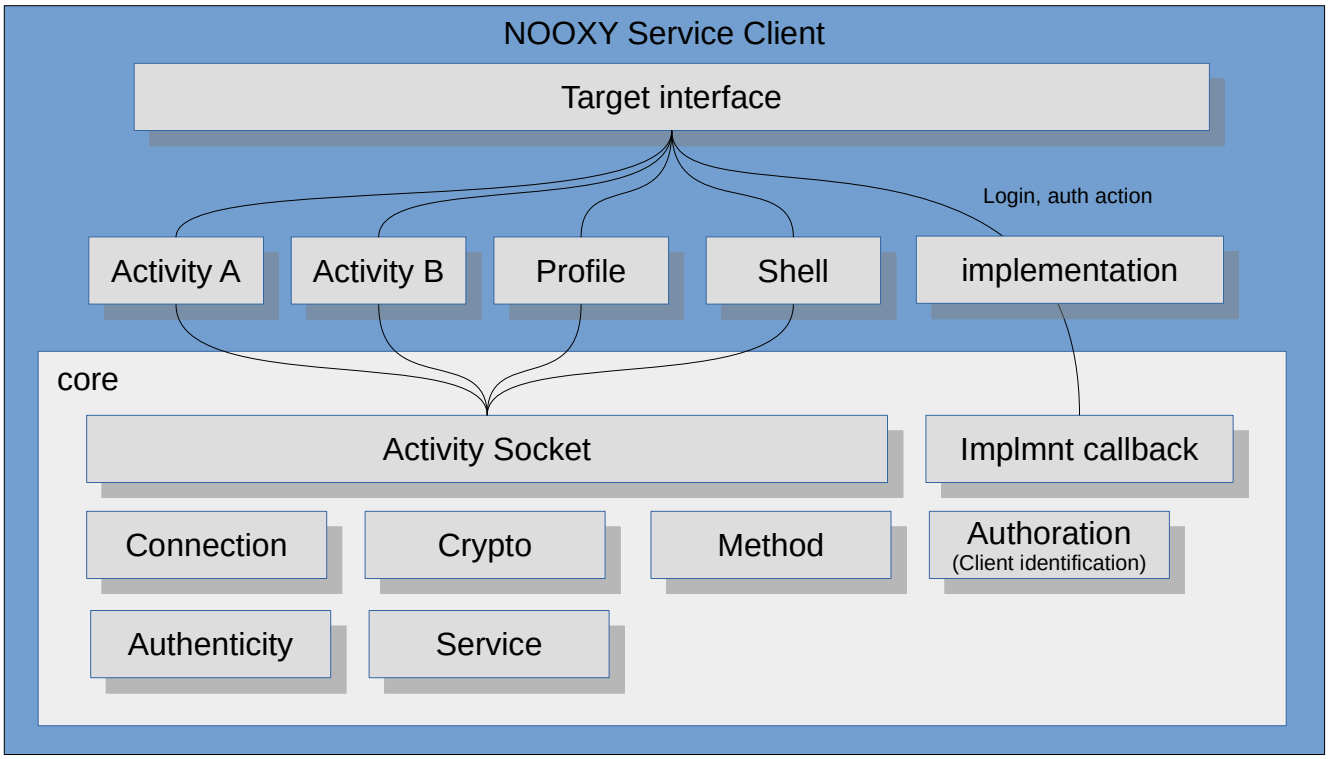
NOOXY Service Framework Orientation

1. User Orientation
2. Server client structure
3. Authoriation system
4. Modurable(base on service)
5. lightweight
6. “Everything based on service” sturcture

Architecture

NOOXY Service Framework Architecture



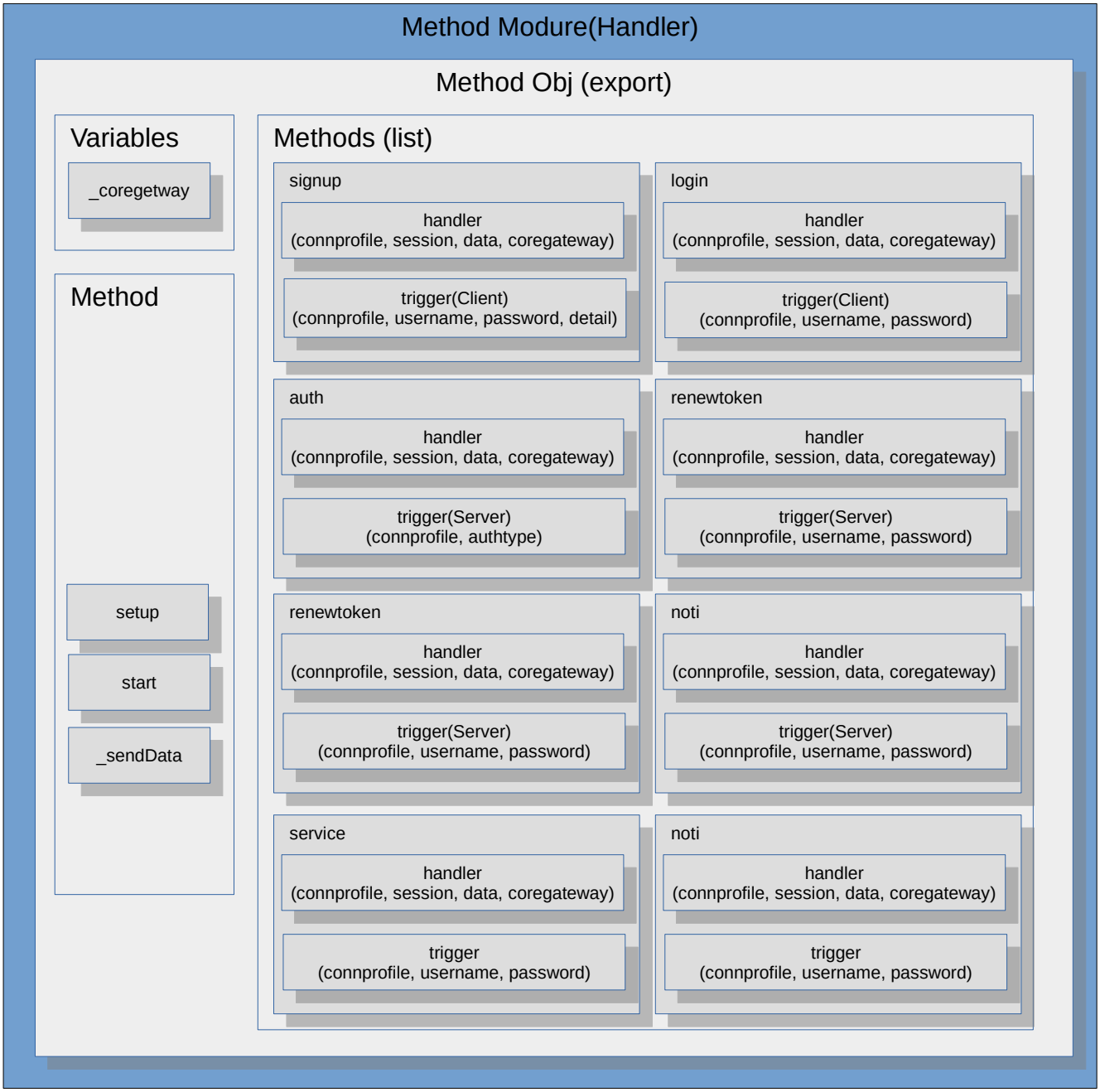


Serverside modure

Method Modure(Handler)

Objective: A pharser or a router. To pharse json between connetion. And switch, and trigger between different operations.

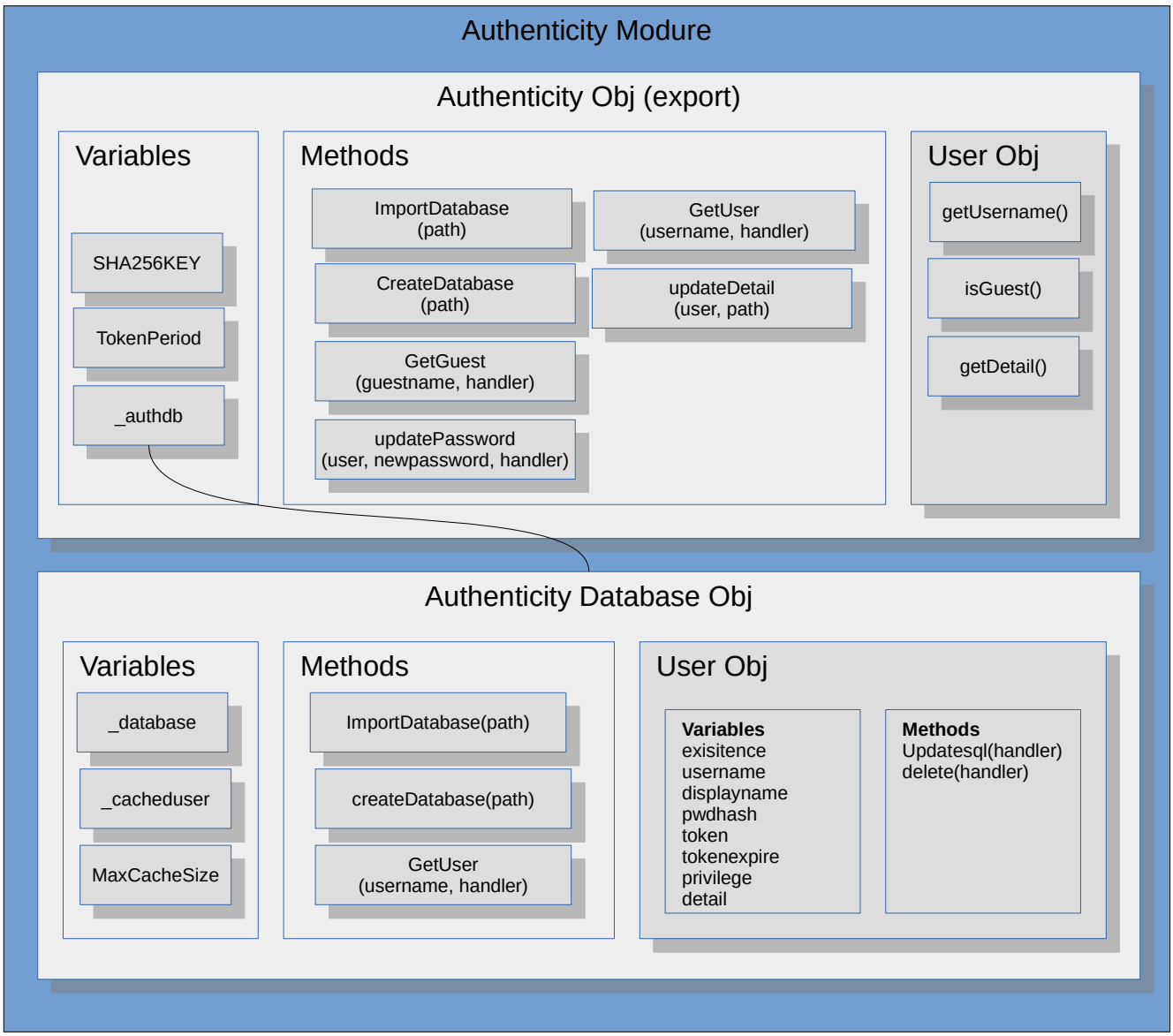
Figure:



Authenticity Modure

Objective: To interact with database, Providing Users Obj cahcing, Creating User Obj, User identification.

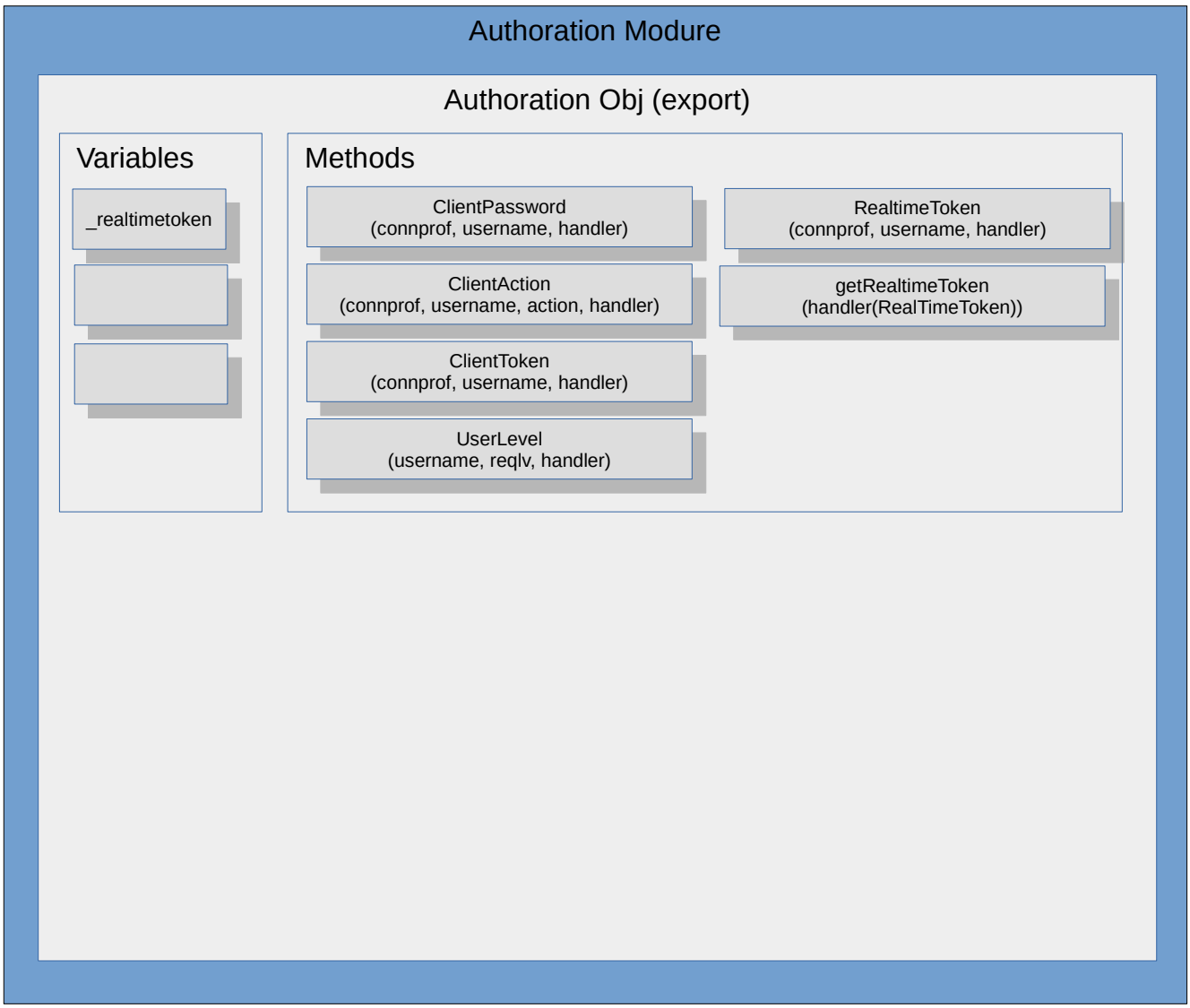
Figure:



Authoration Modure

Objective: To provide function to take authorative actions. Confirming the sensitive data or opearation is permitted.

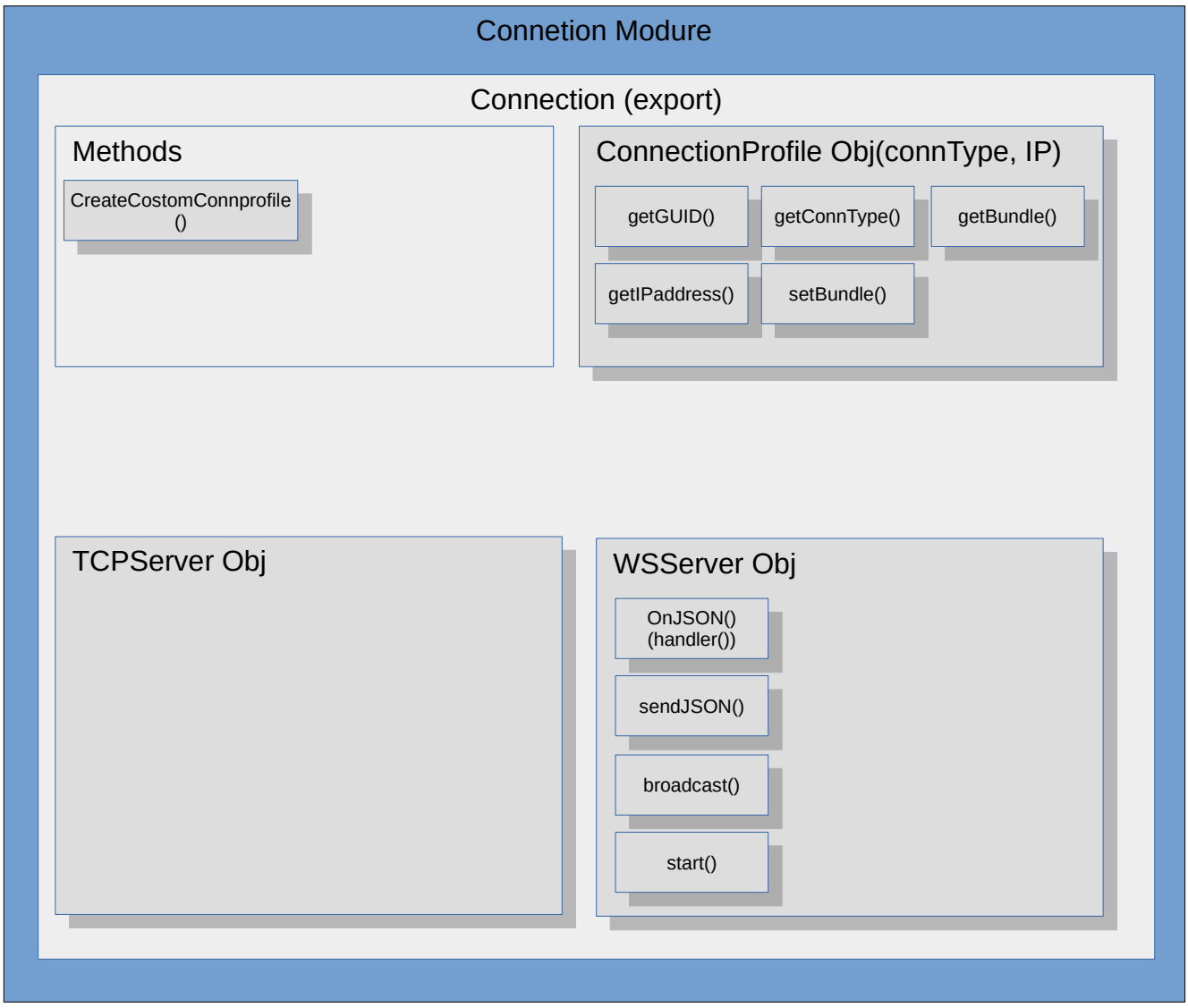
Figure:



Connetion Modure

Objective: Create a interface to get communication with remote device.

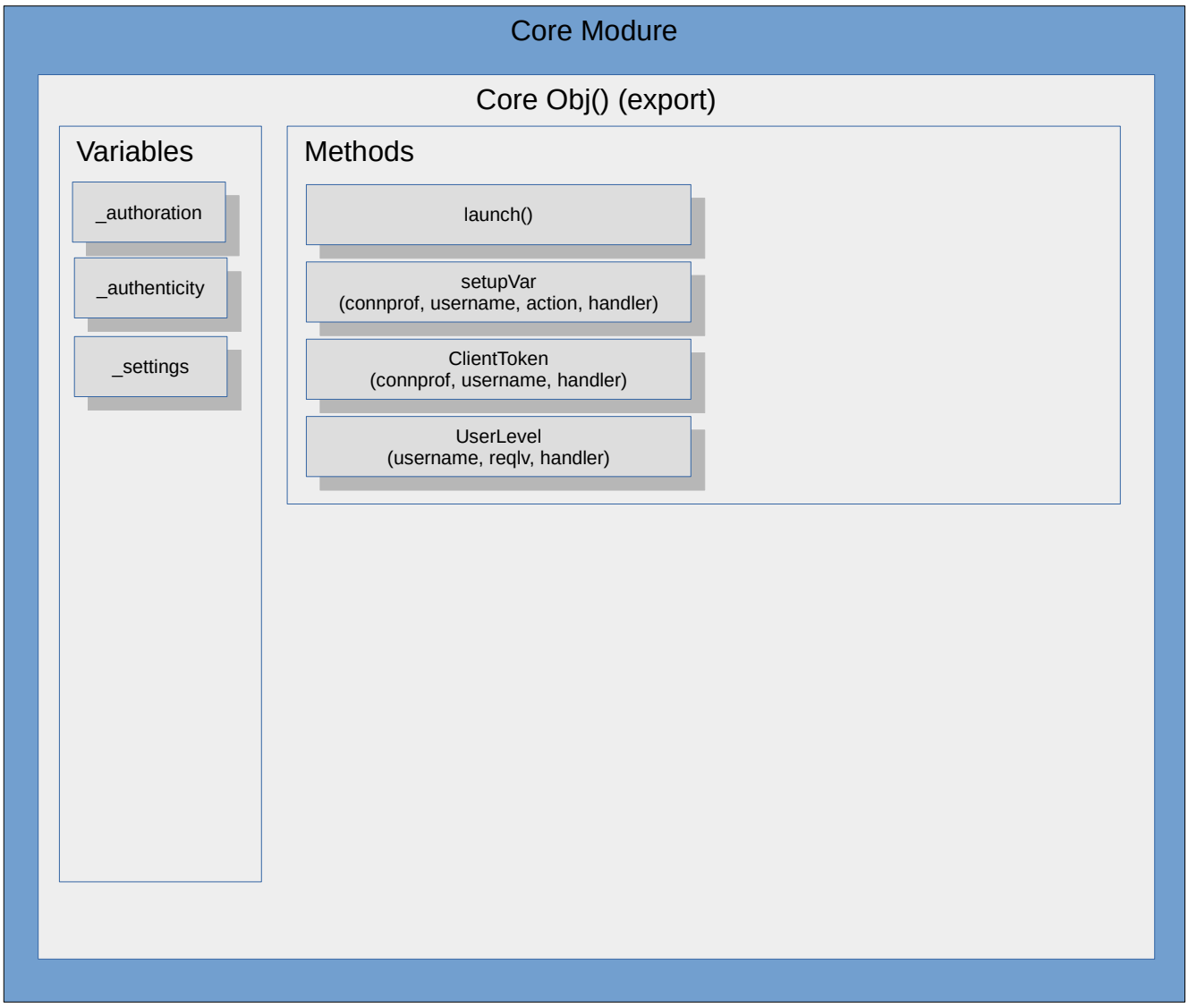
Figure:



Core 1

Objective: provide functions for runtime use, glue

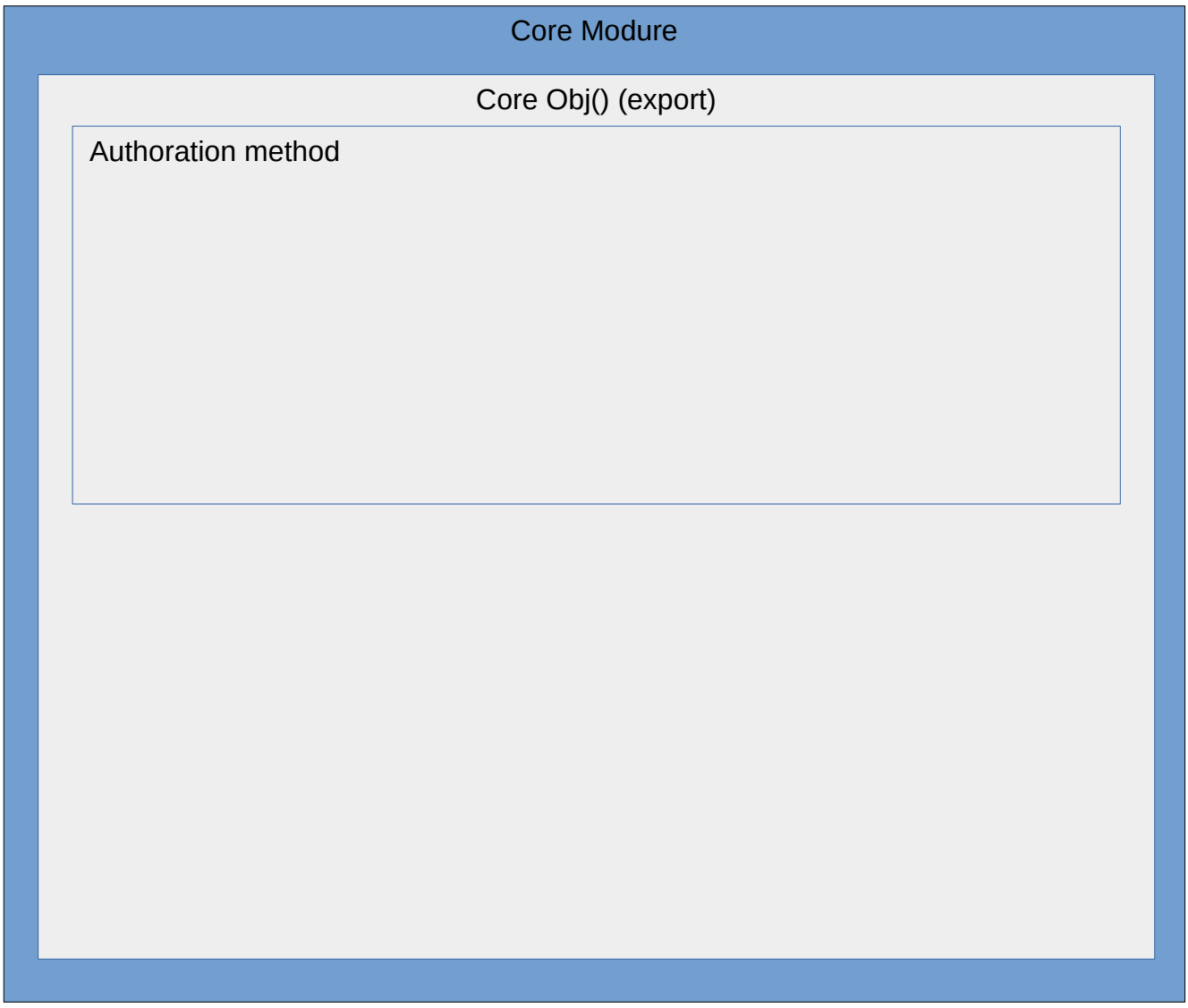
Figure:



Core 2

Objective: provide functions for runtime use, glue

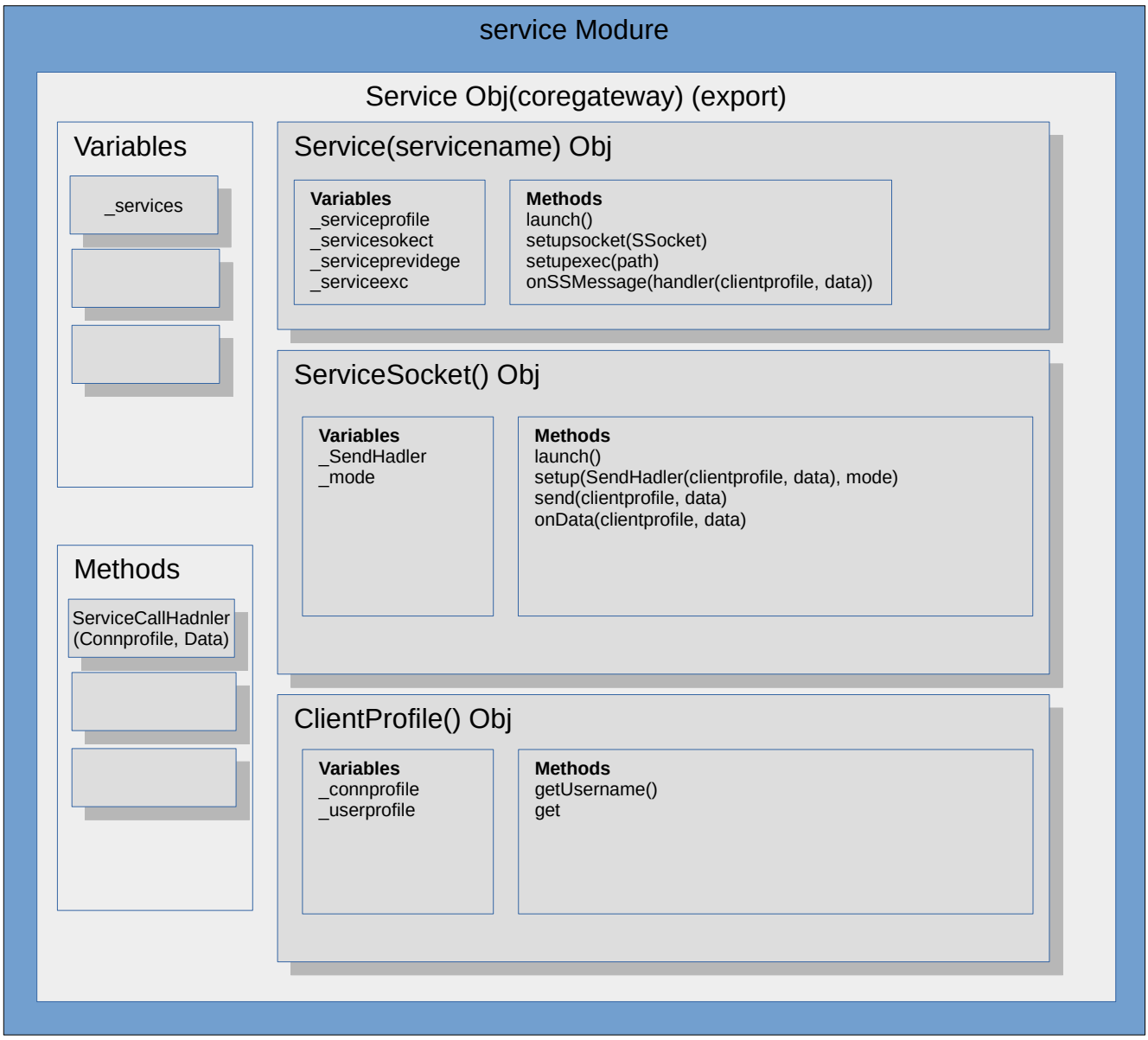
Figure:



Service Modure 1

Objective: provide and mange service api, and route the messages on internet

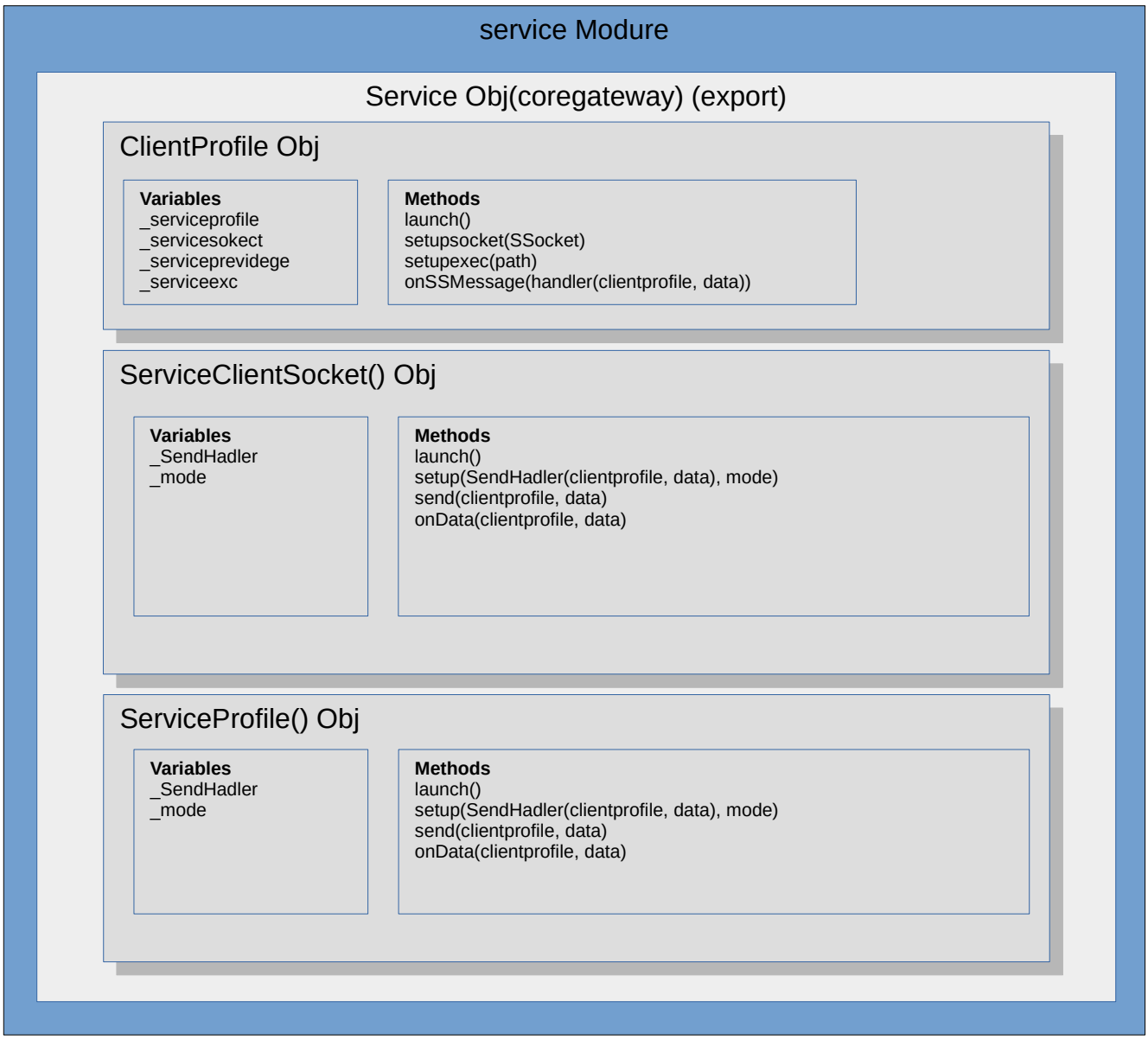
Figure:



Service Modure 2

Objective: provide and mange service api, and route the messages on internet

Figure:



Clientside modure

Service, Servicesocket and API

Explaination of how service work

Once the core of the NSF is started.
The core of NSF will navigate the directories of “services” directory which is under the root of NSF files. And in that directory it will exist a file called “entry.js”. The figure below can help you understand the concept.

```
------|--(NSd(NOOXY Service deamon))-- …
      |--(services)--|--(services_A)--|--(entry.js)
      |              |--(services_B)--|--(entry.js)
      |--(service_files)-- ...
      |--(launch.js)
```

After the core finish navigating the directories under “services”. It will call the entry.js and call it’s function “start()” and pass API parameter in to start() function. Below show how the “entry.js” file might be.

In entry.js

```
function start(api) {
    let ss = api.Service.ServiceSocket
    ss.onMessage = function(ConnProfile, Message) {
        // do somthing
    }

    ss.sendMessage(ConnProfile, “NSF is cool!”);
    // do something with api
}

module.exports = start
```

Service API list

Api.Service.ServiceSocket.onMessage
Api.Service.ServiceSocket.sendMessage
Api.Service.ServiceSocket.onBytes
Api.Service.ServiceSocket.sendBytes
Api.Service.ActivitySocket.onMessage
Api.Service.ActivitySocket.sendMessage
Api.Service.ActivitySocket.onBytes
Api.Service.ActivitySocket.sendBytes
Api.Authoration.Authby.ClientPassword
Api.Authoration.Authby.ClientAction
Api.Authoration.Authby.ClientToken
Api.System.shutdown
Api.System.restart