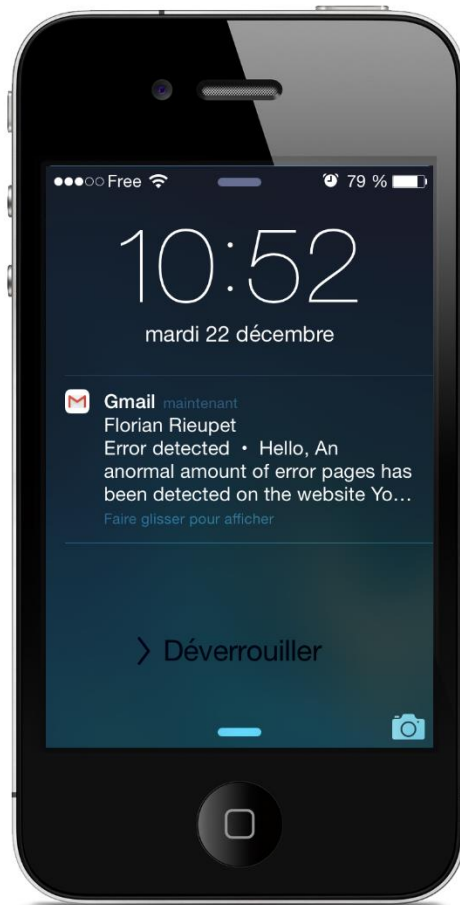


# PROGRAMMER DES ALERTES POUR LA SOLUTION AT INTERNET

Utilisation de l'API Rest AT Internet



Exemple de script

## PREAMBULE

Florian Rieupet  
Jérôme Piquet

Ce script a été créé par des personnes en dehors des projets AT Internet. Il s'agit d'une alternative proposée au système d'alertes. Elle contient des limites et est améliorable. Veuillez noter que son bon fonctionnement n'est pas de la responsabilité d'AT Internet et qu'aucun support sur cette solution ne peut être effectué via le centre support.

Cependant nous sommes preneurs de tous vos retours sur le script et nous allons essayer aussi bien que possible de répondre à vos questions sur GitHub

# Sommaire

## Table des matières

1. Pré-requis .....	4
1.1. Récupération des ressources .....	4
1.2. Installation.....	4
2. Personnalisation du code .....	5
2.1. Paramétrage appel .....	5
2.1.1. Data query .....	6
2.1.2. Code .....	7
2.2. Paramétrage mail.....	8
2.3. Pour aller plus loin .....	11

# 1. PRÉ-REQUIS

## 1.1. RECUPERATION DES RESSOURCES

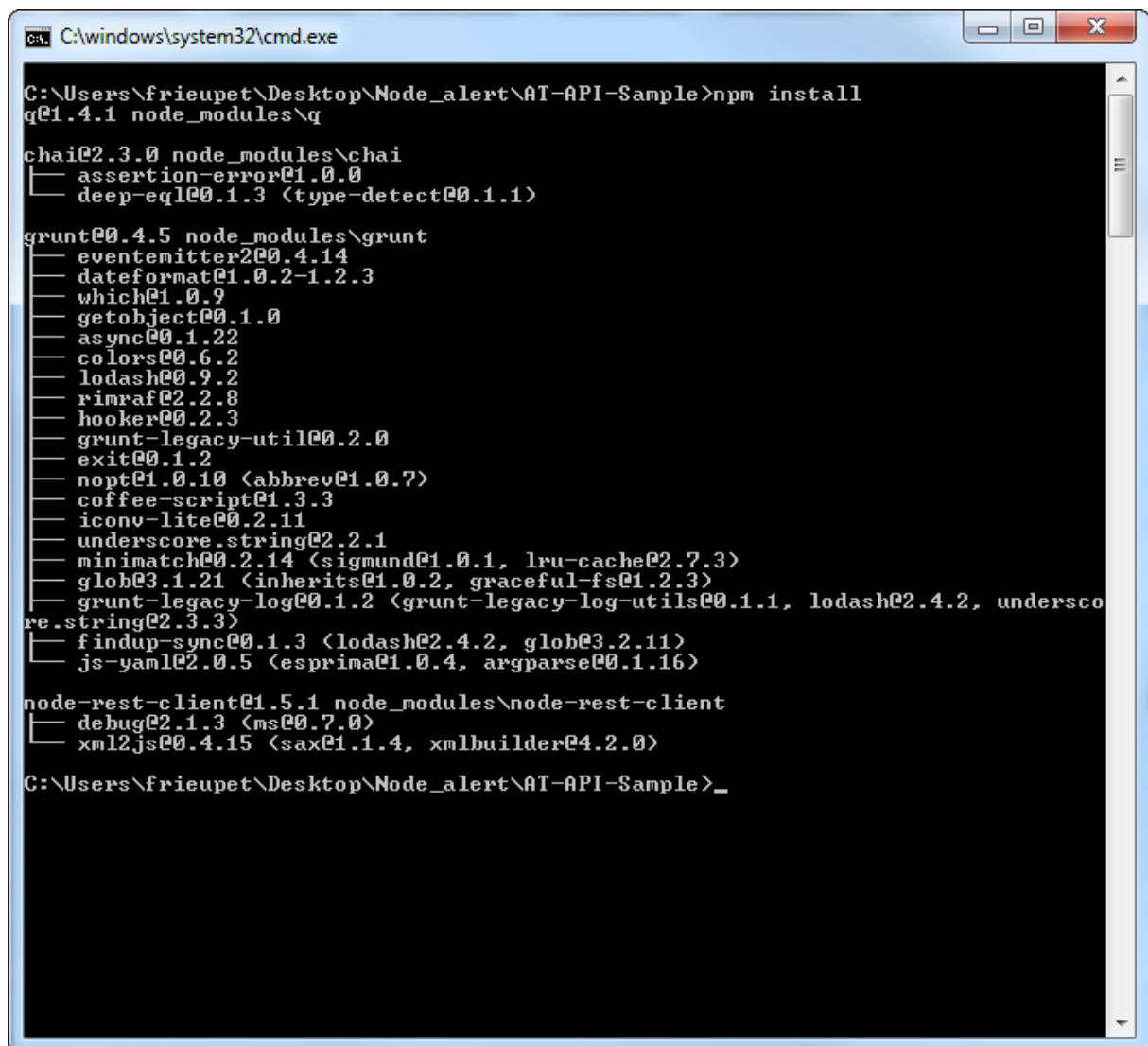
Vous pouvez récupérer toutes les ressources nécessaires pour l'utilisation de ce code via le repository sur GitHub : <https://github.com/frieupet/AT-API-Sample>

## 1.2. INSTALLATION

Le seul prérequis technique consiste en l'installation de la plateforme d'exécution JavaScript Node.js, disponible en téléchargement pour la pluparts des systèmes d'exploitation à l'adresse suivante : <https://nodejs.org/en/>

Une fois Node.js installé, et les sources récupérées via GitHub, il sera nécessaire de télécharger les dépendances via une console et la commande suivante :  
\$ npm install (Node Package Manager est installé en même temps que Node.js).

Vous devriez avoir les résultats suivants :



```
C:\windows\system32\cmd.exe

C:\Users\frieupet\Desktop\Node_alert\AT-API-Sample>npm install
q@1.4.1 node_modules\q
├─ chai@2.3.0 node_modules\chai
│   └─ assertion-error@1.0.0
│       └─ deep-eql@0.1.3 (type-detect@0.1.1)
└─ grunt@0.4.5 node_modules\grunt
    ├── eventemitter2@0.4.14
    ├── dateformat@1.0.2-1.2.3
    ├── which@1.0.9
    ├── getobject@0.1.0
    ├── async@0.1.22
    ├── colors@0.6.2
    ├── lodash@0.9.2
    ├── rimraf@2.2.8
    ├── hooker@0.2.3
    ├── grunt-legacy-util@0.2.0
    ├── exit@0.1.2
    ├── nopt@1.0.10 (abbrev@1.0.7)
    ├── coffee-script@1.3.3
    ├── iconv-lite@0.2.11
    ├── underscore.string@2.2.1
    ├── minimatch@0.2.14 (sigmund@1.0.1, lru-cache@2.7.3)
    ├── glob@3.1.21 (inherits@1.0.2, graceful-fs@1.2.3)
    ├── grunt-legacy-log@0.1.2 (grunt-legacy-log-utils@0.1.1, lodash@2.4.2, underscore.string@2.3.3)
    └─ findup-sync@0.1.3 (lodash@2.4.2, glob@3.2.11)
        └─ js-yaml@2.0.5 (esprima@1.0.4, argparse@0.1.16)

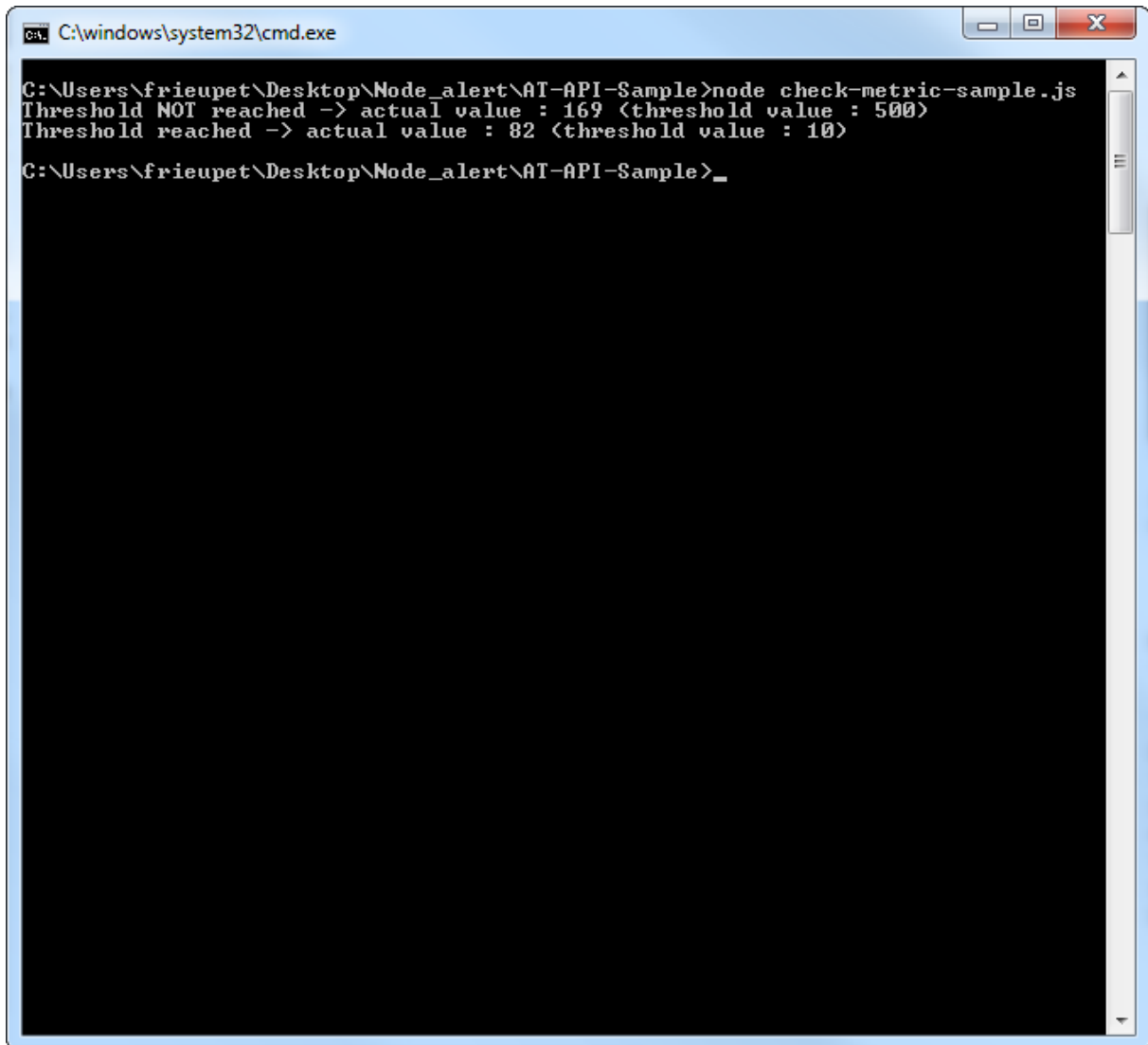
node-rest-client@1.5.1 node_modules\node-rest-client
├─ debug@2.1.3 (ms@0.7.0)
└─ xml2js@0.4.15 (sax@1.1.4, xmlbuilder@4.2.0)

C:\Users\frieupet\Desktop\Node_alert\AT-API-Sample>
```

Si vous voulez lancer le script pour vérifier son fonctionnement au cours de vos modifications il faudra utiliser la commande suivante

Node check-metric-sample.js

Vous pourrez alors visualiser les résultats comme dans mon exemple



```
C:\windows\system32\cmd.exe

C:\Users\frieupet\Desktop\Node_alert\AT-API-Sample>node check-metric-sample.js
Threshold NOT reached -> actual value : 169 <threshold value : 500>
Threshold reached -> actual value : 82 <threshold value : 10>

C:\Users\frieupet\Desktop\Node_alert\AT-API-Sample>_
```

A noter que pour la première utilisation, en raison de l'absence d'une URL valide d'appel à notre API et de vos identifiants des erreurs seront levées, pour tester le script il sera donc nécessaire de personnaliser le code dans un premier temps.

## 2. PERSONNALISATION DU CODE

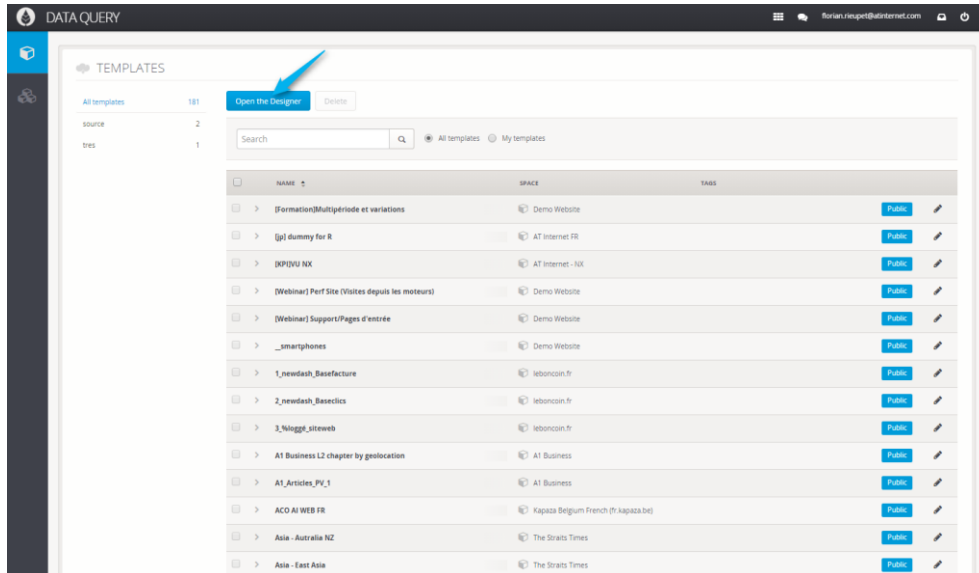
### 2.1. PARAMÉTRAGE APPEL

La première chose à faire sera d'intégrer l'appel API AT Internet que vous allez vouloir surveiller. Pour ceci le code qu'il sera nécessaire de modifier se trouve dans le fichier check-metric-sample.js et plus précisément dans la variable **queryParameters** :

```
27 // Query parameters
28 ▼ var queryParameters = {
29     "login": "",
30     "password": "",
31     "url": ""
32 };
```

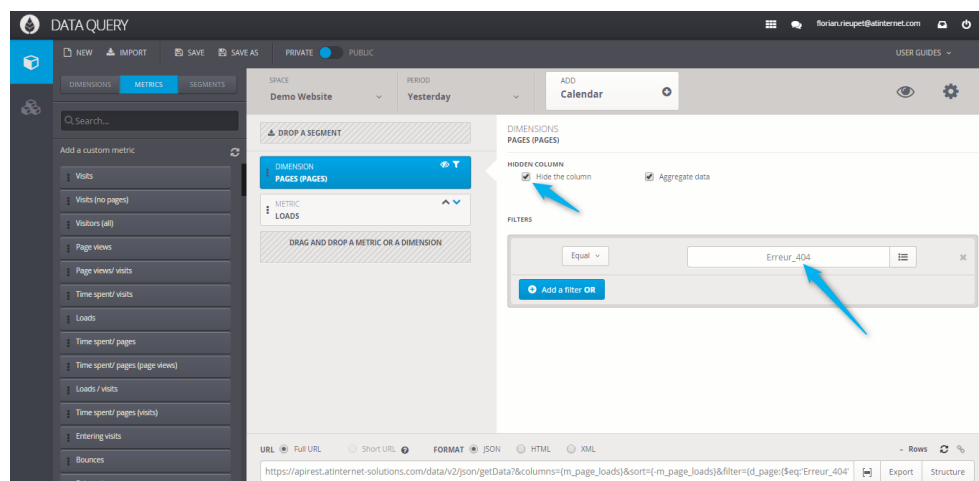
### 2.1.1. DATA QUERY

Pour créer votre appel il est nécessaire d'accéder à Data Query via votre interface AT Internet. Une fois sur la page d'accueil de Data Query cliquez sur la création d'un template

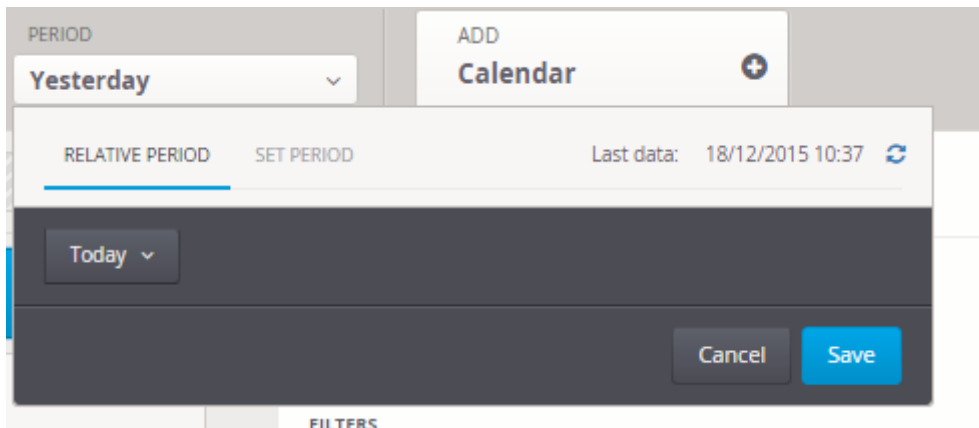


Créez alors votre requête en sélectionnant les dimensions et métriques que vous voulez utiliser. Dans mon exemple, je veux suivre mes pages 404 : si elles atteignent un seuil trop important je voudrais recevoir un mail. Je vais donc avoir besoin de la dimension page avec un filtre sur le nom de cette dernière pour ne m'intéresser uniquement aux pages 404.

Une fois mes dimensions et métriques glissées/déposées il me faudra cliquer sur la dimension pages pour rajouter le filtre. A noter qu'il est aussi nécessaire de cacher la colonne. **Le script attends une unique valeur pour fonctionner**, il nous faut donc faire en sorte de cocher sur toutes nos dimensions la case permettant de cacher la colonne pour que le résultat de la requête soit uniquement la métrique.



Une fois votre requête créée vous pouvez modifier la période pour la mettre en temps réel.



Il est désormais temps de récupérer l'URL en bas de l'interface Data Query, pensez à bien sélectionner le format JSON. Vous pouvez à ce moment-là tester votre requête via votre navigateur et **vérifier qu'uniquement une valeur est présente**. Pour une lecture plus simple, tester votre requête au format HTML, le résultat devrait être le suivant :

Loads
3

A noter que vous pouvez aller plus loin dans la personnalisation de l'appel API. Dans la requête que l'on vient de récupérer les données sont sur toute la journée, par contre on pourrait imaginer s'intéresser uniquement aux 5 dernières minutes pour des résultats plus précis. Vous pouvez télécharger le [guide utilisateur ici](#) pour apprendre à vous servir de l'API de façon plus poussée.

### 2.1.2. CODE

Une fois la requête API récupérée il est nécessaire de modifier le code pour insérer la configuration nécessaire. Copier l'URL obtenue à l'étape précédente dans le paramètre url.

Profitez-en pour rajouter votre login AT Internet ainsi que votre mot de passe (ou mieux, celui d'un compte avec des droits limités, créé pour l'occasion) dans les paramètres **login** et **password**. Votre code devrait donner le résultat suivant :

```
27 // Query parameters
28 var queryParameters = {
29   "login": "login_atinternet@domain.com",|
30   "password": "password_AT_Internet",
31   "url": "https://apiest.atinternet-solutions.com/data/v2/html/getData?&columns={m_page_loads}&sort={-m_page_loads}" +
32   "&filter={d_page:{&seq:'404_error'}}&space={s:554331}&period={R:{D:'0'}}&max-results=50&page-num=1"
33 };
```

Maintenant que l'appel est effectué dans votre script, il est important de déterminer le seuil que vous considérez comme critique et à partir duquel vous voulez recevoir une alerte. Il est défini dans la variable **threshold**. Dans mon exemple je veux recevoir une alerte dès que j'ai dépassé les 10 chargements de ma page 404\_error

```
38     var threshold = 10;
```

La dernière partie du code vous permet de déterminer ce que vous voulez faire si le seuil est atteint.

Si la requête s'exécute sans erreur, alors le traitement « **then** » de la promesse JavaScript (en réponse à l'exécution du code asynchrone de la requête http) va s'exécuter. En cas d'erreur, le traitement du « **catch** » sera exécuté (cela peut être, par exemple, la résultante d'une erreur http, ou de l'impossibilité de lire le résultat de la requête).

Dans cette section « **then** », le branchement « **if** » déterminera l'action à effectuer si le seuil est atteint, et le branchement « **else** » l'action à faire si le seuil n'est pas atteint. Selon vos besoins il faudra donc modifier cette section de code.

```
35 AThelper.getScalarValue(queryParameters)
36   .then(function (result) {
37
38       var threshold = 10;
39
40       //compare with threshold
41       if (result >= threshold) {
42           transporter.sendMail(mailOptions, function(error, info){
43               if(error){
44                   return console.log(error);
45               }
46           console.log('Message sent: ' + info.response);
47       });
48   });
49   }
50   else {
51       console.log("Threshold NOT reached -> actual value : %s (threshold value : %s)", result, threshold);
52   }
53
54 })
55 .catch(function (error) {
56     console.log("An error has occurred : %s", error.message);
57 });
58
```

Vous êtes libres sur cette partie de choisir les actions à effectuer. Dans mon exemple, si le seuil est atteint je m'envoie un mail pour être averti, et si le seuil n'est pas atteint j'affiche uniquement le résultat dans ma console.

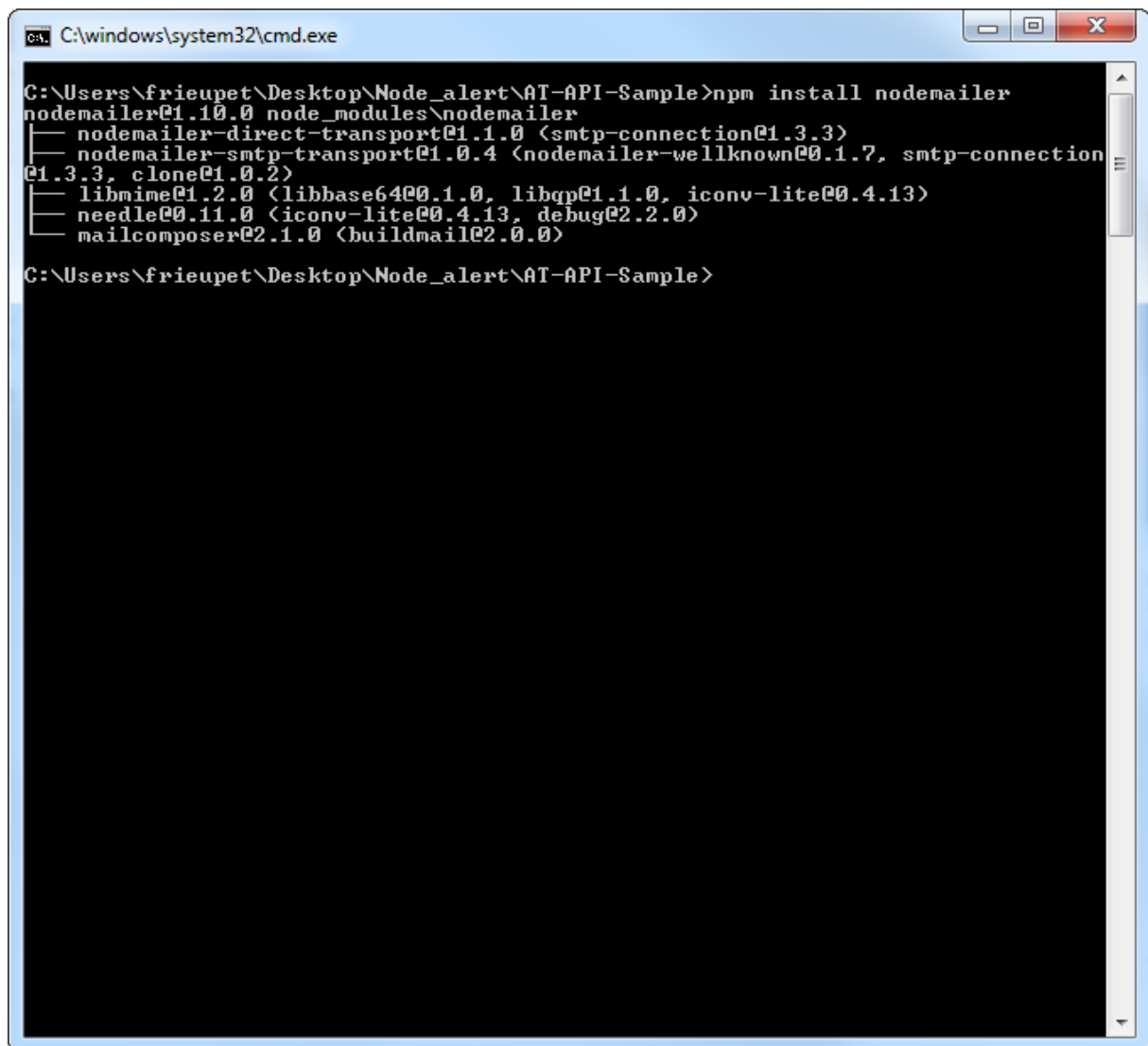
## 2.2. PARAMÉTRAGE MAIL

Dans cette dernière partie, l'objectif est de vous expliquer comment paramétrer l'envoi d'un mail si le seuil est atteint.

Il faut dans un premier temps installer le module de mail que l'on voudra utiliser. J'ai dans mon cas utilisé nodemailer qui est très simple à utiliser. Cependant vous pouvez utiliser d'autres modules selon ce que vous voulez faire une fois l'alerte levée, on peut imaginer recevoir un SMS ou utiliser une API. Pour installer nodemailer la commande à utiliser est

```
$npm install nodemailer
```





```
C:\windows\system32\cmd.exe

C:\Users\frieupet\Desktop\Node_alert\AT-API-Sample>npm install nodemailer
nodemailer@1.10.0 node_modules\nodemailer
├── nodemailer-direct-transport@1.1.0 (smtp-connection@1.3.3)
├── nodemailer-smtp-transport@1.0.4 (nodemailer-wellknown@0.1.7, smtp-connection@1.3.3, clone@1.0.2)
├── libmime@1.2.0 (libbase64@0.1.0, libqp@1.1.0, iconv-lite@0.4.13)
├── needle@0.11.0 (iconv-lite@0.4.13, debug@2.2.0)
└── mailcomposer@2.1.0 (buildmail@2.0.0)

C:\Users\frieupet\Desktop\Node_alert\AT-API-Sample>
```

Une fois l'installation terminée il est nécessaire d'effectuer et de paramétrer nodemailer, vous pouvez consulter le repository complet de ce projet sur GitHub :

<https://github.com/andris9/Nodemailer#possible-transport-methods>

Vous pourrez ainsi aller encore plus loin et trouver exactement comment le paramétrer si vos besoins sont différents des miens.

Dans une première variable que j'ai nommé **transporter** on va configurer nodemailer

```
6 // create reusable transporter object using SMTP transport
7 var transporter = nodemailer.createTransport({
8   service: 'Gmail',
9   auth: {
10     user: 'florian.atinternet@gmail.com',
11     pass: 'password'
12   }
13 });
```

Dans mon cas j'utilise le service gmail, pour ceci il faut accepter dans vos paramètres google que des applications moins sécurisées peuvent utiliser votre service.

<https://www.google.com/settings/security/lesssecureapps>

Dans **user** et **pass** il faudra renseigner vos identifiants de votre compte gmail. Bien sûr si vous n'avez pas gmail mais un service d'envoi de mail, vous pouvez configurer celui-ci à cet endroit, je vous renvoie vers la documentation de nodemailer dans GitHub pour pouvoir l'effectuer.

Dans une seconde variable que j'ai nommé **mailOptions** vous allez paramétrer vos informations d'envoi cette fois-ci.

```
18 // setup e-mail data with unicode symbols
19 var mailOptions = {
20   from: 'Florian Rieupet <florian.atinternet@gmail.com>', // sender address
21   to: 'florian.rieupet@atinternet.com', // list of receivers
22   subject: 'Hello', // Subject line
23   text: 'Error detected', // plaintext body
24   html: 'Hello<br> An anormal amount of error pages has been detected on the website.' // html body
25 };
```

**from** : Il correspond à l'adresse qui va envoyer le mail, ce sera probablement la même adresse que l'étape précédente si vous avez choisi la solution gmail.

**to** : cela va vous permettre de paramétrer à quelle adresse vous voulez envoyer le mail. Vous pouvez renseigner plusieurs adresses en les séparant par des virgules.

**subject** : ce sera le sujet de votre mail

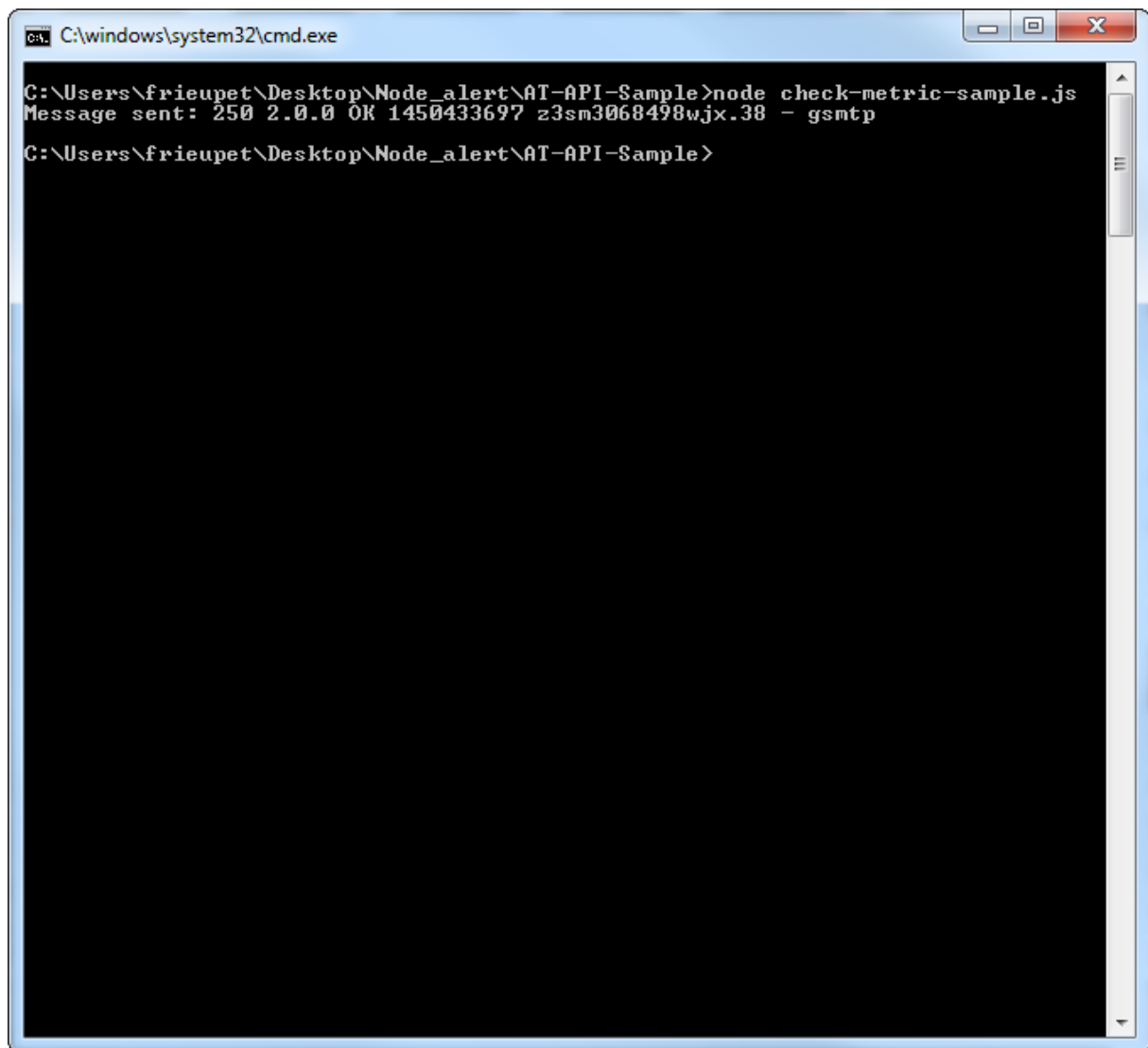
**text** : il s'agit du contenu du mail si le code HTML n'est pas exécuté

**html** : il s'agit du contenu de votre mail

La configuration de nodemailer est maintenant terminée, et votre application va pouvoir envoyer des mails. La dernière étape est donc de créer l'envoi de ce mail dans votre application au moment où vous le désirez. Dans mon cas, dès que le seuil est atteint je veux recevoir mon alerte, je vais donc pouvoir insérer l'envoi de l'email dans le code que nous avons créé au point précédent.

```
35 ATHelper.getScalarValue(queryParameters)
36 .then(function (result) {
37
38     var threshold = 10;
39
40     //compare with threshold
41     if (result >= threshold) {
42         transporter.sendMail(mailOptions, function(error, info){
43             if(error){
44                 return console.log(error);
45             }
46             console.log('Message sent: ' + info.response);
47         });
48     }
49     else {
50         console.log("Threshold NOT reached -> actual value : %s (threshold value : %s)", result, threshold);
51     }
52
53 })
54
55 .catch(function (error) {
56     console.log("An error has occurred : %s", error.message);
57 });
58
```

J'utilise la fonction d'envoi de mail fournie dans nodemailer dans le cas où la vérification me renvoie que le seuil est trop important. Dans ce cas-là j'affiche aussi dans ma console que le message a été envoyé. Si je lance l'application que nous venons de créer, si mon seuil est atteint j'obtiens alors le résultat suivant (et un mail dans ma boîte) :



```
C:\windows\system32\cmd.exe
C:\Users\frieupet\Desktop\Node_alert\AT-API-Sample>node check-metric-sample.js
Message sent: 250 2.0.0 OK 1450433697 z3sm3068498wjx.38 - gsmt
C:\Users\frieupet\Desktop\Node_alert\AT-API-Sample>
```

### 2.3. POUR ALLER PLUS LOIN

Maintenant que le script a été créé vous allez pouvoir le placer sur un serveur et l'automatiser pour qu'il se lance à intervalle régulier afin d'avoir votre système d'alerte fonctionnel. AT Internet travaille actuellement sur une application de live alerting qui facilitera et permettra d'automatiser la gestion des alertes. Cette fonctionnalité est prévue pour fin 2016.