# WEB SCRAPING ACCOUNT SERVICE API

# 

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Author** | **Revision Number** | **Date** |
| TCSCODER | 1.0 | 22/08/2016 |
|  |  |  |
|  |  |  |

# TABLE OF CONTENTS

1. Organization of Submission
2. Architecture
3. Preconditions
4. Configuration
5. Verification

# 

# 

# **Organization of Submission**

The submission file is named as submission.zip extract the zip file, the zip will be extracted to submission root directory. This root directory will be referred to as <ROOT> in this document.

Navigate to <ROOT> and list the contents. The following files and directories are listed.

|  |  |
| --- | --- |
| Name | Description |
| app.zip | Application source code |
| docs | contains deployment guide and postman collection json files |
| README.md | Submission README file |

Extract the app.zip file, the file will be extracted to app directory.

1. **Architecture**

The application has very simple and scalable architecture.

There are two very abstract concepts in the application.

1. **Account Type**: Account type is any configurable name given to an account. Initially there is only a single account configured with the type ‘remit’.
2. **Account Handlers**: Account handlers are the corresponding handlers for the account type. Each of the account handlers must reside in /accounts directory and must expose a method *“handleRequest(query, body)”*.  
   Since an account handler may need access to the client request data, query strings and request body are parsed by the application and provided to the account handlers when application invokes *handleRequest.* Since account handlers are indenpendent of each other, they are invoked in parallel for faster processing.

The account handlers will automatically be loaded by the application and corresponding handler will be invoked.

**How account handlers are loaded ?**

The account handlers are loaded based on their type. They will be searched for in /accounts directory under <ROOT> of the application.

The name of the account handler is configured in config/default.js file with each account type.

For ex: Current configuration has only one account type configured ‘remit’

ACCOUNTS: {

remit: {

handler: ‘RemitAccountHandler.js’

}

}

NOTE: The account type must be the object properties of the ACCOUNTS object. These properties will be searched for in the application, and corresponding account handlers will be loaded for each key.

The corresponding handler for remit account type is ‘RemitAccountHandler’ in /accounts directory.

To add a new account type and handler simply add a key with account type and handler property with name of the handler and corresponding handler module in /accounts directory.

This structure has no dependency on each other and can be scaled to any number of handlers.

**How new handlers can be added ?**

Following the above architecture it is very easy to add new account type and their handlers.

To add a new account handler simply add the corresponding type on ACCOUNTS object in default.js file and create the corresponding handler in /accounts directory.

Suppose I would like to add an account type of ‘rbs’

Add a corresponding entry in default.js file for rbs and define handler and create handler module with the name RbsAccountHandler.js in /accounts directory.

ACCOUNTS: {

remit: {.....}

rbs: {

handler: ‘RbsAccountHandler.js’

}

}

1. **Preconditions**

To run the application the following prerequisites have to be installed. This guide assumes that you are using either ubuntu linux distribution or debian linux distribution or mac osx operating systems.

On both ubuntu and debian we can use apt-get utility to install packages, so it’s fairly simple

1. node.js 4.5.0 version. Visit the [official page](https://nodejs.org/download) to download installers. It is strongly recommended to use nvm to install node.  
   You can use following command to install nvm  
   curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.25.4/install.sh | bash  
   After installing nvm run nvm ls-remote to list remote versions. To install a particular version of node run nvm install 4.5.0  
   For more information visit [this](https://github.com/creationix/nvm) guide.  
   When you install node npm should automatically be installed
2. The node module dependencies are specified in package.json file. To install the node dependencies run npm install in your terminal at root of source code directory
3. Install nodemon globally to run watch task. npm install -g nodemon
4. **Configuration**

The application is using [node-config](https://github.com/lorenwest/node-config) module to manage application configuration.

The node-config module reads application configuration from the config directory. The default configuration is defined in default.js file

Below is a table which defines all configurations for the application

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **sample value** |
| LOG\_LEVEL | The log level for the application | debug |
| PORT | The port on which to start the application. Reading from environment is also supported. | process.env.PORT || 3500 |
| ERROR\_API | The error reporting API endpoint | http://private-b850a-webscrapererrorapi.apiary-mock.com/api/errors |
| ACCOUNTS | The various pluggable accounts to be used with this application.  The account type is the object key and there is a mapping of key -> object for each account type. | remit: {  handler: ‘RemitAccountHandler.js’,  login: {  url: 'http://remit-proxyservice.ap-northeast-1.elasticbeanstalk.com/proxy\_login',  auth: {  username: process.env.REMIT\_USERNAME,  password: process.env.REMIT\_PASSWORD,  },  },  details: {  url: 'http://remit-proxyservice.ap-northeast-1.elasticbeanstalk.com/MainAccountBalanceHistory.jsf',  },  } |

The REMIT\_USERNAME and REMIT\_PASSWORD has to be set as environment variable

export REMIT\_USERNAME=coder

export REMIT\_PASSWORD=uR2FpzUv

1. **Verification**

Source code quality verification.

The application uses eslint for source code quality check. run npm run lint to verify. There shouldn’t be any lint errors.



Manual Verification

To manually verify start the server using npm run start command

postman collection is included which list all the requests.

1. GET /api/accounts

Get all the accounts details.

Query parameters

fromTransDate: The start date of the transactions, should be in MM/DD/YYYY format

toTransDate: The end date of the transactions, should be in MM/DD/YYYY format

Response: You should get json response with each account type and corresponding response ( account details ) for that account type.

