

#37 HTTP Notification Provider Contest

github: <https://github.com/Arseny271/FreeTON-HttpNotificationProviderContest>

email: arsendmitr@gmail.com

telegram: @successful17k

api docs: <https://api.events.ton.arsen12.ru/>

statistics page: <https://events.ton.arsen12.ru/>

This document contains instructions for deploying and configuring the provider and consumer of notifications in Free TON.

1. Provider

Dependencies: MongoDB, NodeJs

- Copy the provider folder to your server.
- Generate a keypair and write them into the secret/provider.keys.json. This keys will be used to sign the sent notifications
- Edit configs in configs folder:

provider-info.json - Provider information

config-mongo.json - Contains database url

config-kafka.json - Connection parameters for kafka

config-server.json –

host, port - server api parameters

max_concurrency - maximum number of simultaneously processed messages

first_attempt_timeout - message timeout on first send (if 0, no timeout)

next_attempts_timeout - message timeout on subsequent send

max_attempts - maximum number of retry attempts to send

retry_time_base - the base of the number to calculate the delay time between retransmissions. Time between dispatches is calculated as n^x , where n is

retry_time_base and x is a number of unsuccessful attempts.

- Run node init.js to initialize the database.
- Run node index.js to start the message provider

2. Consumer

The consumer is an intermediate layer, he receives messages from the provider, decrypts them and forwards them to another address in accordance with the specified rules. In fact, it would be more correct to call this module a router

For the convenience of users, the consumer has an admin panel. It simplifies consumer configuration and allows you to view the last received messages in decrypted form.

Dependencies: MongoDB, NodeJs

- Copy the consumer folder to your server
- Edit configs in configs folder:

config-mongo.json - Contains database url

config-server.json –

receiverPort, receiverEndpoint - port and url for receiving notify from providers

adminPanelPort - Port on which the admin panel will run

- Run node init.js to initialize database.
- Run node index.js to start the consumer.

3. Configuring a provider to receive notifications

- Start the debot and send it a callback url (id = arsen12). In response, the bot will send a hash and a secret.

The screenshot shows a Telegram chat interface with a bot. The bot has sent a message with a button labeled "Choose your provider:". The user has clicked this button, and the bot has responded with a blue button containing the text "github.com/Arseny271/FreeTON-HttpNotificationProviderContest, ID = arsen12". The user has clicked this button, and the bot has responded with a message "Enter data for this provider" followed by a blue button containing the URL "https://callback.ton.arsen12.ru/events". The user has clicked this button, and the bot has responded with a message containing a "Hash:" and a "Secret:". The hash is "91f1d870209660c87a64f11765ee5617e7d4fa23b153b6629b237a80a4d5eb95" and the secret is "b979ddfe2dc999c748e48834baad3130de0cc33b7a9f28c6582d6f00ffb74eb1". The bot has also sent a message "Use this secret to verify ownership of the server. Then repeat the call to the bot." and another message "Keep this secret, you may need it later." At the bottom, there is a button labeled "Available actions:".

Choose your provider:

github.com/Arseny271/FreeTON-HttpNotificationProviderContest, ID = arsen12

Enter data for this provider

<https://callback.ton.arsen12.ru/events>

Hash:
91f1d870209660c87a64f11765ee5617e7d4fa23b153b6629b237a80a4d5eb95

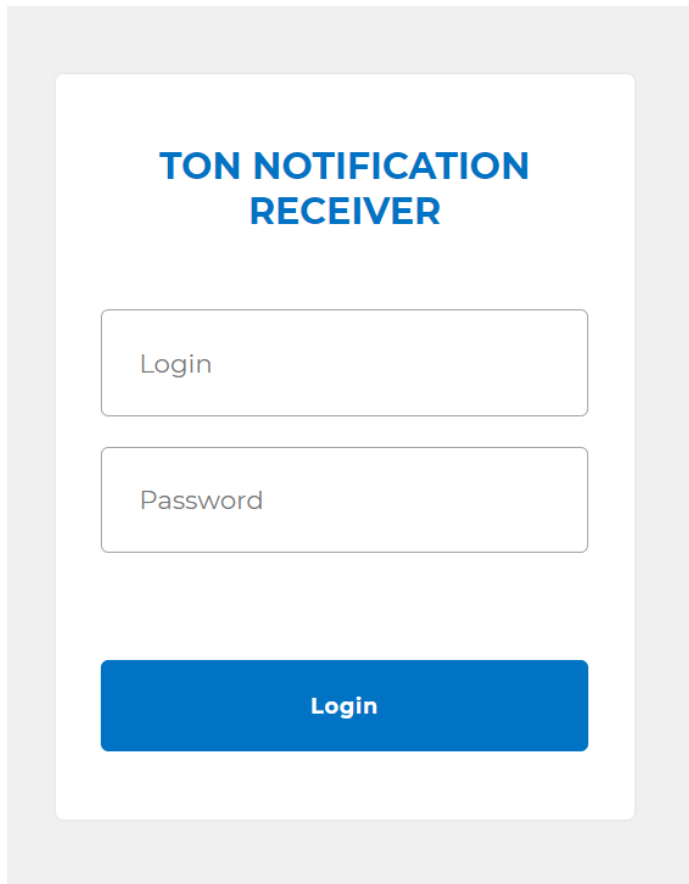
Secret:
b979ddfe2dc999c748e48834baad3130de0cc33b7a9f28c6582d6f00ffb74eb1

Use this secret to verify ownership of the server. Then repeat the call to the bot.

Keep this secret, you may need it later.

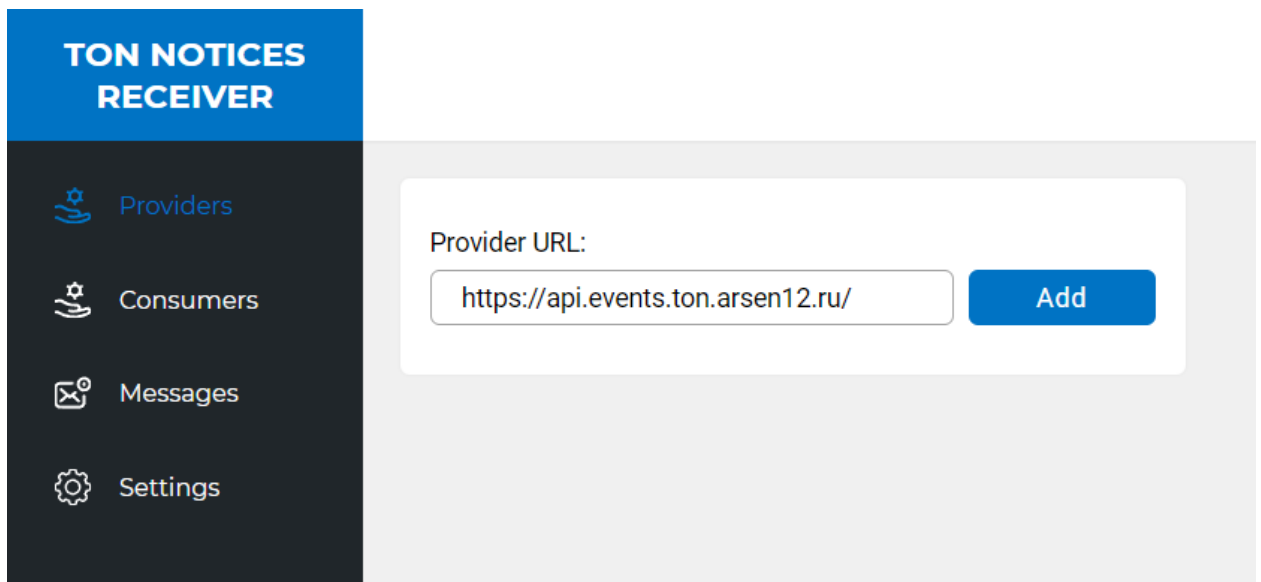
Available actions:

- Open your consumer admin panel (default login and password **admin admin**).



The screenshot shows a login interface for the 'TON NOTIFICATION RECEIVER'. It features a white card with a blue header containing the text 'TON NOTIFICATION RECEIVER'. Below the header, there are two input fields: the first is labeled 'Login' and the second is labeled 'Password'. At the bottom of the card is a blue button with the text 'Login' in white.

- On the providers tab, add arsen12 (url: <https://api.events.ton.arsen12.ru/>) to the list of providers.



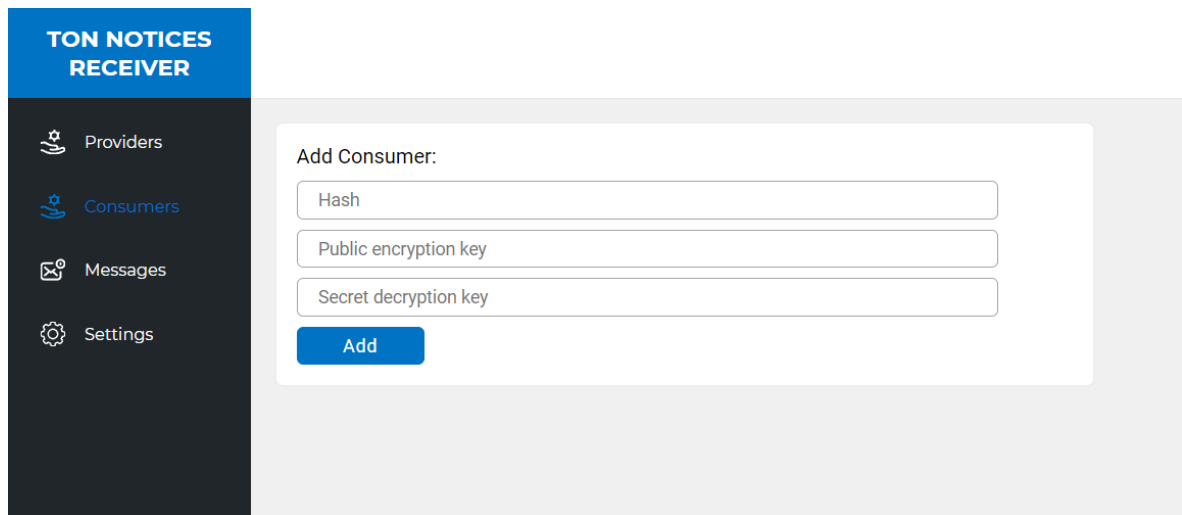
The screenshot shows the 'TON NOTICES RECEIVER' interface. On the left is a dark sidebar with a blue header 'TON NOTICES RECEIVER' and four menu items: 'Providers' (with a gear icon), 'Consumers' (with a gear icon), 'Messages' (with an envelope icon), and 'Settings' (with a gear icon). The main area is light gray and contains a white card with the label 'Provider URL:'. Below the label is an input field containing the URL 'https://api.events.ton.arsen12.ru/' and a blue 'Add' button.

- Open the consumers tab and create a consumer with the following parameters:

Hash - the hash that the debot told you

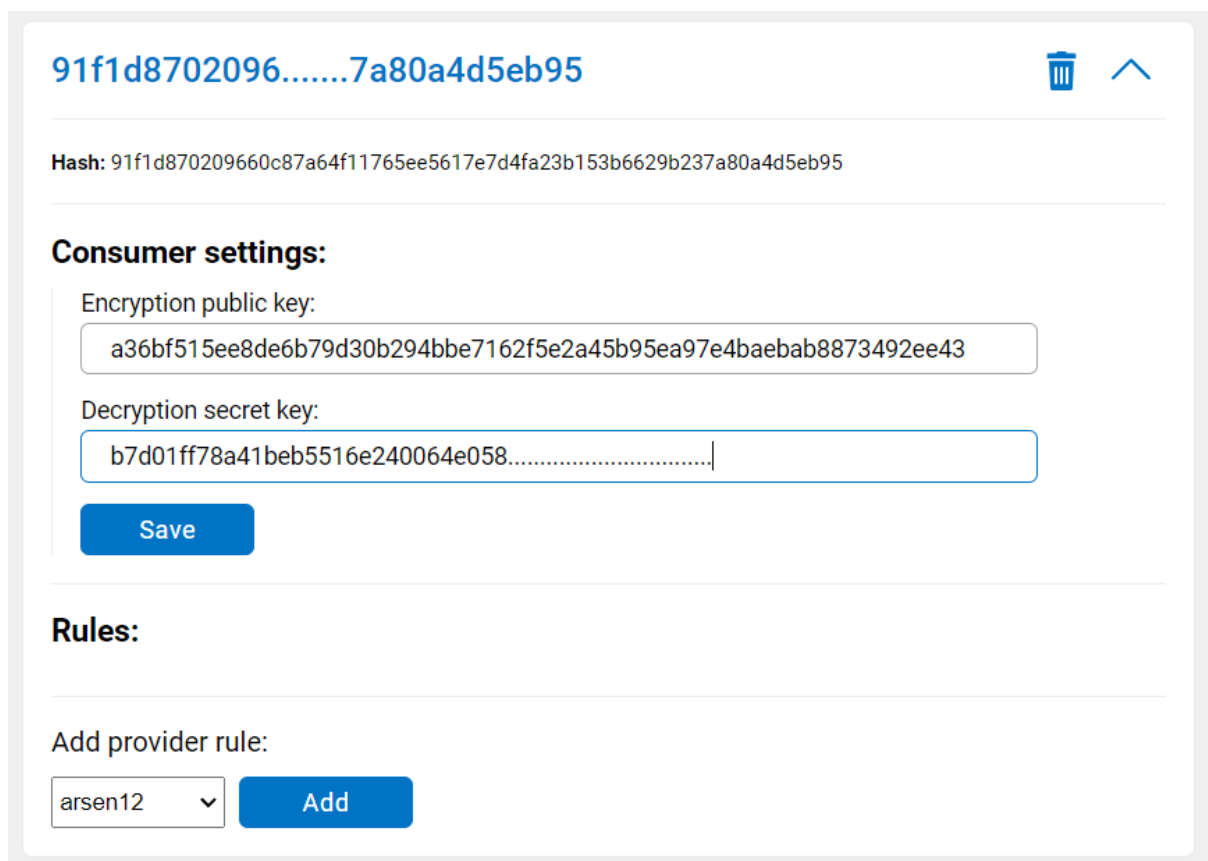
Public encryption key – The public key of the service. At the moment it is a constant: a36bf515ee8de6b79d30b294bbe7162f5e2a45b95ea97e4baebab8873492ee43

Secret decryption key - The secret key that the debot told you when the contract was deployed



The screenshot shows the 'TON NOTICES RECEIVER' interface. On the left is a dark sidebar with a menu containing 'Providers', 'Consumers', 'Messages', and 'Settings'. The 'Consumers' option is highlighted. The main area is titled 'Add Consumer:' and contains three input fields: 'Hash', 'Public encryption key', and 'Secret decryption key'. Below these fields is a blue 'Add' button.

- After successful creation, you will see something like this. Select the provider ID and click add rule



The screenshot shows the settings page for a consumer. At the top, the provider ID '91f1d8702096.....7a80a4d5eb95' is displayed with a trash icon and an upward arrow. Below this, the 'Hash' is shown as '91f1d870209660c87a64f11765ee5617e7d4fa23b153b6629b237a80a4d5eb95'. The 'Consumer settings:' section includes 'Encryption public key' (a36bf515ee8de6b79d30b294bbe7162f5e2a45b95ea97e4baebab8873492ee43) and 'Decryption secret key' (b7d01ff78a41beb5516e240064e058.....). A blue 'Save' button is at the bottom of this section. The 'Rules:' section has a heading 'Add provider rule:' followed by a dropdown menu showing 'arsen12' and a blue 'Add' button.

Secret is used to prove ownership of the server. Indicate the secret that debot told you.

Forward url is the address to which the decrypted notification will be sent

Rules:

arsen12:

Secret:

Forward url:

Save

Remove

After creating the rule, write the debot in turn (instead of the url callback, you can send any non-empty text), if you specified the correct secret in the provider's settings, the debot will inform you that the callback has been successfully registered, now you will receive notifications about events in the network

Send callbackUrl | deviceToken to provider

Choose your provider:

github.com/Arseny271/FreeTON-
HttpNotificationProviderContest, ID =
arsen12

Enter data for this provider

14515131513

Callback registered successfully

4. About https support

To support https, it is proposed to use nginx with similar configs.

```
server {
    server_name debot.events.ton.arsen12.ru api.events.ton.arsen12.ru;
    location / {
        proxy_pass http://localhost:8000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    listen 443 ssl;
    ssl_certificate /etc/letsencrypt/live/debot.events.ton.arsen12.ru/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/debot.events.ton.arsen12.ru/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
```

5. About performance

Despite the fact that the program is written in js and the single-threaded nature of this language, the code tries to use all processor cores using multiprocessing. This is important because each notification is signed with a secret key, which is computationally expensive and single-threaded processing may not provide the desired performance.