Table of Contents

APEX Websocket Notification

Bundle

Purpose of this software bundle was to enable all APEX developers to use modern and state of the art web features like Node, Websockets and nice looking notifications in their applications.

This bundle includes all these features and simultaneously is designed to use all of them out of the box. This means:

- Ready to go Node.js websocket server especially for notifications using socket.io
- A native PL/SQL package to send all kinds of different messages/notifications using APEX_WEB_SERVICE

- APEX plugins for all kind of events that are needed by the notification system:

    - Initialize websocket connection to server
    - Send messages and notifications to users
    - Show different styled notifications on client side

Developers don´t need to be experts in Javascript or JQuery and stuff like that (But as always, it´s not a bad skill!;)). APEX Know-How and a good knowledge of using Dynamic Actions should be enough to implement this notification bundle in your applications...

Infrastructure Diagram

Demo

A demo application is available under https://apex.danielh.de/ords/f?p=WSNOTIFY

And of course you find a APEX export (**demo_app.sql**) of it in ../apex/ folder. To use it just import the app and then go through the installation steps below. Under Shared Components --> Edit Application Definition --> Substitutions Strings, set

- **G_WS_SERVER_HOST** to the hostname or ip address of your node notification server
- **G_WS_SERVER_PORT** to the port of your node notification server
- **G_WS_SERVER_AUTHTOKEN** to your secure and random authToken of your node notification server (read further for more informations)

The demo includes all plugins and shows the most common preferences and possibilities.

Preview

Changelog

1.0.0 - Initial Release

Installation and Configuration

Installation Node.js Server

Install Node.js

It is required to have a up and running Node.js installation on your server. Either install it using a package manager, or download the latest version from Nodejs homepage...for example:

- Ubuntu:

```
apt-get install nodejs
apt-get install npm
```

- Mac OS X (Homebrew):

```
brew install nodejs
```

- Windows: Download and install it from Nodejs homepage

npm is the package manager for Node applications. npm is used to install all required packages by the Node Websocket Notification Server...

Install Notification Package

- Copy the complete folder ../node/node-notify-server to your server
- change to this directory via command line:

```
cd /path/to/node-notify-server
```

- Install all dependencies

```
npm install
```

- Start server

```
npm start
```

This should be everything to have the Notification Server up und running. To check that, you can point your web browser to http://[host-ip-of-server]:8080

There you should get a overview of all supported services by the Notification Server.

This helper pages are supported by the server:

- **Overview Services:** http://[host-ip-of-server]:8080
- **Server Status Page:** http://[host-ip-of-server]:8080/status
- **Websocket Test Client:** http://[host-ip-of-server]:8080/testclient

Configure Notification Package

You can change the default behavior of the server by editing the JSON config file ../node/node-notify-server/prefs.json

```
{
    "server": {
        "ip": "0.0.0.0", // listener ip address 0.0.0.0 for all in
        "port": "8080", // listener port
        "authUser": "", // User for HTTP basic auth, empty means n
        "authPwd": "", // Password for HTTP basic auth, empty mean
        "sslKeyPath": "", // FOR SSL: path to ssl key file (./cert
        "sslCertPath": "", // FOR SSL: path to ssl certificate fil
        "logging": true // logging to console on or off, for prod
    },
    "socket": {
        "private": true, // activate private websocket room/namesp
        "public": true, // activate public websocket room/namespac
```

```
        "authToken":"please-change-me" // authentication token, cl
    }
}
```

After changing one of these settings, please restart the Node Notification Server.

SSL Support:

- For test environments you can use the script ../node/node-notify-server/certs/create_cert.sh to create a self signed certificate
- For production environments please get a officially signed certificate and place key.pem and cert.pem into the certs folder

Installation Database

Database ACL

All notifications are sent through web service requests. Therefore a ACL is needed, so you are allowed to connect to this host. Here is a example script, configure it to reflect your environment...

```
DECLARE
  --
  l_filename    VARCHAR2(30) := 'ws-notify-host.xml';
  l_host_or_ip  VARCHAR2(50) := '<YOUR-NODE-WEBSOCKET-HOST>'
  l_port        NUMBER       := 8080;
  l_schema      VARCHAR2(20) := '<SCHEMA>';
  --
BEGIN
  --
  BEGIN
    dbms_network_acl_admin.drop_acl(acl => l_filename);
  EXCEPTION
```

```
    WHEN OTHERS THEN
      NULL;
  END;
  --
  dbms_network_acl_admin.create_acl(acl         => l_filename,
                                    description => 'All requests t
                                    principal   => l_schema,
                                    is_grant    => TRUE,
                                    privilege   => 'connect');
  --
  dbms_network_acl_admin.add_privilege(acl       => l_filename,
                                       principal => l_schema,
                                       is_grant  => TRUE,
                                       privilege => 'resolve');
  --
  dbms_network_acl_admin.assign_acl(acl        => l_filename,
                                    host       => l_host_or_ip,
                                    lower_port => l_port);
END;
/
```

Oracle SSL Wallet

If you configured the Node Notification Server with SSL/HTTPS support, a Oracle SSL Wallet is needed by the database to communicate with the REST-Interface for sending notifications.

To manually create a wallet, either use Oracle Wallet Manager or create the wallet with openssl utils like:

- Grab the certificate from your server node-notify-server/certs
- Create the wallet on command line

```
openssl pkcs12 -export -in cert.pem -out ewallet.p12 -nokeys
```

- Place the wallet file on your database server
- Change the wallet path and password in package specification under "Websocket REST Call defaults / security defaults"
    - **g_ssl_wallet_path:** Path of Oracle SSL wallet
    - **g_ssl_wallet_pwd:** Password of Oracle SSL wallet

Compile PL/SQL package

- Change the global variables in the package specification under "Websocket REST Call defaults" to reflect your environment
    - **g_ws_rest_host:** Hostname or IP of Node Server
    - **g_ws_rest_port:** Port of Node Server
    - **g_ws_rest_proto:** Protocol of Node Server (http or https) - if https, then "g_ssl_wallet_path" and "g_ssl_wallet_pwd" are required
    - **g_ws_basic_auth_user:** HTTP Basic Auth username of Node Server (REST-Interface)
    - **g_ws_basic_auth_pwd:** HTTP Basic Auth password of Node Server (REST-Interface)
- Connect to your database and compile the package spec and body (ws_notify_api.pks & ws_notify_api.pkb) from ../plsql folder

Installation APEX

Install Plugins

The APEX part contains 3 plugins, you can find it in ../apex/plugins folder. Just import these 3 files to your application and you are ready to go.

- **Init Websocket Notify Connection** - dynamic_action_plugin_de_danielh_initwsnotifyconnection.sql

- **Send Websocket Notify** - dynamic_action_plugin_de_danielh_sendwsnotify.sql
- **Show Websocket Notify** - dynamic_action_plugin_de_danielh_showwsnotify.sql

For a detailed description of the plugins, read further under **"Usage Section"** or import the demo app sql file to your workspace.

Usage

Node.js Server

As mentioned in the installation steps, the node notification server component consists of 3 areas:

- **REST-Interface**

Sending messages and notifications to users which are connected to the websocket interface.

- **Websocket-Interface**

Connecting and authenticating users against the node server and still more to receive live messages on client browser from server part. There exists 2 rooms/namespaces which users can subscribe to:

- **private** - For single user messages to all instances of one user (e.g. one user is logged in with 3 browsers)
- **public** - For single user messages to all instances of one user *AND* broadcasting messages to all connected clients

- **Helper pages**

Helper pages to get informations about services, status of server and a test client page to test some websocket interactions.

- **Overview Services:** http://[host-ip-of-server]:8080
- **Server Status Page:** http://[host-ip-of-server]:8080/status
- **Websocket Test Client:** http://[host-ip-of-server]:8080/testclient

General settings of the node server like IP, port, authentication, SSL support and active websocket rooms can be configured with ../node/node-notify-server/prefs.json file as mentioned above.

REST-Service

The REST-Service is designed to send messages to connected websocket users. Base-URL scheme looks like this:

```
Type: GET
http://[host-ip-of-server]:[port]/notifyuser
```

- **URL-Parameter**
    - **userid** (required) - User-ID of connected user, *in APEX APP_USER is used*
    - **room** (required) - Websocket room - *valid values: private, public*
    - **type** (required) - Notification type - *valid values: info, success, warn, error*
    - **optparam** (optional) - Optional Parameter string to send any kind of information to the websocket client
- **HTTP Header-Variables**
    - **notify-title** (required) - Title of notification

- **notify-message** (required) - Message content of notification

A demo call using curl looks like this:

```
curl -H "notify-title: Test Title Text" -H "notify-message: Test M
```

PL/SQL API

The PL/SQL API consists of one package **ws_notify_api** and includes many procedures to send any kind of possible notifications over the REST-Interface. It can be used to send notifications to users via PL/SQL or inside of APEX. All web service requests are based on APEX package APEX_WEB_SERVICE.

List of global package variables

- **g_ws_rest_host** - Node Notification Server Hostname or IP
- **g_ws_rest_port** - Node Notification Server Port
- **g_ws_rest_path** - Node Notification Server REST-Service Base Path
- **g_ws_rest_proto** - Node Notification Server Protocol (http or https)
- **g_ws_rest_base_url** - Combines Protocol, Host, Port and Path
- **g_ws_basic_auth_user** - HTTP Basic Auth username of Node Server (REST-Interface)
- **g_ws_basic_auth_pwd** - HTTP Basic Auth password of Node Server (REST-Interface)
- **g_ssl_wallet_path** - If https, path to oracle wallet
- **g_ssl_wallet_pwd** - If https, password of oracle wallet

List of all procedures with all parameters

Procedure: do_rest_notify_user

Purpose: Send Websocket Notifications over REST to connected users (General sending procedure with all parameters)

Parameter:

- **i_userid** (required)
- **i_room** (required) - ("private" or "public")
- **i_type** (required) - (info, success, warn, error)
- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_user_private_info

Purpose: Send Websocket Notification to User / Room: Private / Type: Info

Parameter:

- **i_userid** (required)
- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_user_private_success

Purpose: Send Websocket Notification to User / Room: Private / Type: Success

Parameter:

- **i_userid** (required)
- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_user_private_warn

Purpose: Send Websocket Notification to User / Room: Private / Type: Warn

Parameter:

- **i_userid** (required)
- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_user_private_error

Purpose: Send Websocket Notification to User / Room: Private / Type: Error

Parameter:

- **i_userid** (required)
- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_user_public_info

Purpose: Send Websocket Notification to User / Room: Public / Type: Info

Parameter:

- **i_userid** (required)
- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_user_public_success

Purpose: Send Websocket Notification to User / Room: Public / Type: Success

Parameter:

- **i_userid** (required)
- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_user_public_warn

Purpose: Send Websocket Notification to User / Room: Public / Type: Warn

Parameter:

- **i_userid** (required)
- **i_title** (required)
- **i_message** (required)

- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_user_public_error

Purpose: Send Websocket Notification to User / Room: Public / Type: Error

Parameter:

- **i_userid** (required)
- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_all_public_info

Purpose: Send Websocket Notification to all Users / Room: Public / Type: Info

Parameter:

- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_all_public_success

Purpose: Send Websocket Notification to all Users / Room: Public / Type: Success

Parameter:

- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_all_public_warn

Purpose: Send Websocket Notification to all Users / Room: Public / Type: Warn

Parameter:

- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

Procedure: do_notify_all_public_error

Purpose: Send Websocket Notification to all Users / Room: Public / Type: Error

Parameter:

- **i_title** (required)
- **i_message** (required)
- **i_optparam** (optional) - (Optional Parameter String)

A demo call could look like this:

```
BEGIN
  ws_notify_api.do_rest_notify_user(i_userid   => 'daniel',
                                    i_room     => 'private',
                                    i_type     => 'info',
```

```
                                        i_title    => 'My test title',
                                        i_message  => 'My test message
                                        i_optparam => 'myoptionalinfo1
END;
```

APEX

As already mentioned above, the APEX part contains 3 plugins to cover all functionalities from initialization of a websocket connection, sending notifications to other connected users to show incoming notifications. All 3 plugin files are located under ../apex/plugins folder.

Init Websocket Notify Connection

- **Plugin File:** dynamic_action_plugin_de_danielh_initwsnotifyconnection.sql
- **Purpose:** Initialize a connection to the websocket server, for general usage over all pages of your APEX application this plugin should be located on Global Page 0 (Zero)
- **Plugin Attributes:**
    - **Use SSL** - Choose if the connection to the websocket server is secure (HTTPS) or plain (HTTP)
    - **Server Hostname or IP** - The hostname or ip address of the websocket server
    - **Server Port** - The port of the websocket server
    - **Websocket Type or Room** - The type/room of the websocket server. There are 2 possible connections: private, public
    - **Websocket Auth User-ID** - User-ID which connects / authenticates against the websocket server. Is used to identify a

user. Default: APEX APP_USER
    - **Websocket Auth-Token** - Auth-Token of the Node Notify Websocket Server. This is to increase security.
    - **Logging** - Whether to log events in the console

- **Plugin Events:**

    - **Private Websocket Connection success** - Successfully connected to private room
    - **Private Websocket Connection error** - Error connecting to private room
    - **Private Websocket Disconnected** - Websocket connection is disconnected for private room
    - **Public Websocket Connection success** - Successfully connected to public room
    - **Public Websocket Connection error** - Error connecting to public room
    - **Public Websocket Disconnected** - Websocket connection is disconnected for public room
    - **Receive Private Message** - Receiving an incoming private message
    - **Receive Public Message** - Receiving an incoming public message

---

Send Websocket Notify

- **Plugin File:** dynamic_action_plugin_de_danielh_sendwsnotify.sql

- **Purpose:** Send websocket notifications to other connected users or to all connected users

- **Plugin Attributes:**
    - **To User (User-ID)** - Item which holds informations about the User-ID or Username of the user who get´s the notification
    - **Websocket Room** - Item which holds informations about the websocket room - Valid values: "private" or "public"
    - **Notification Type** - Item which holds informations about the type of the notification - Valid values: info, success, warn, error
    - **Notification Title** - Item which holds informations about the title of the notification
    - **Notification Message** - Item which holds informations about the message content of the notification
    - **Optional Parameter** - Item which holds informations about a optional parameter - This could be any kind of string or number combination. This information can be processed on the client side
    - **Show Wait Spinner** - Show / Hide wait spinner for AJAX call
    - **Logging** - Whether to log events in the console
- **Plugin Events:**
    - **Send Notification success** - Sending a notification was successfull
    - **Send Notification error** - Error sending a notification
    - **Send Notification missing values** - Missing required parameters for sending a notification

---

Show Websocket Notify

- **Plugin File:** dynamic_action_plugin_de_danielh_showwsnotify.sql

- **Purpose:** Show notifications for incoming websocket message events. Notifications UI based on AlertifyJS

- **Plugin Attributes:**

    - **Notification Icon CSS Class** - Icon CSS class for the incoming notification - Default: fa-bell
    - **Notification Wait Time** - Time (in seconds) to wait before the notification is dismissed, a value of 0 means keep open till clicked.
    - **Notification Position** - Position of the notification message on screen
    - **Logging** - Whether to log events in the console

All other parameters of an notification object (title, message, type (info, success, warn, error), etc.) are automatically fetched from the websocket message event.

- **Plugin Events:**
    - **Private Notification clicked** - Clicked on a private notification object
    - **Public Notification clicked** - Clicked on a public notification object

License