rs-connectivity-service
 - The Reachback-Service team repo for WebSocket tunneling developed in Node - for any Cloud Computing environment.

Vision
- Provide seamless access to the Enterpise side resources(Database, Artifacts, files, ..etc) for any application.
- Make the Developer and Architect productive by eliminating the unnecessary work to externalize enterprize internal resources.
- Create a universal PaaS across countries and regions.

##Features Implemented
- Support ecured Websocket connection (SSL/TLS).
- Support Proxy Services.
- Support Multi-tenancy (Multi-Servers/Clients).
- Support all streaming/non-streaming TCP protocols. (ICMP, SSH, FTP, sFTP, etc.)
- Bi-Directional streaming.
- Support Client<=>Gateway<=>Server model. (new)
- Support Server-side connectivity monitoring (wakeup request)
- Support Basic/UAA/OAuth/OAuth2 Authentication.

##Work in progress
- Client/Server CLI.
- Gateway CLI.
- [Java RS Client Library](https://github.build.ge.com/212359746/phConnectivityJavaClientLib)
- Gateway as a Service/Tile in Predix CF.

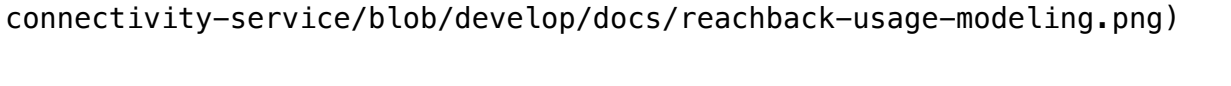
Installation

```shellscript
git clone #git-repo #path
cd #path
npm install
```

Generate a self-signed TLS/SSL Certificate with OpenSSL

```shellscript
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days XXX
```

Usage

###usage modeling


###gateway
```javascript
var RSGateway=require('./../rs-gateway');

var phs=new RSGateway({
    localPort:process.env.PORT || 9898,
    ssl:{
        key:'./cert/rs-key.pem',
        cert:'./cert/rs-cert.pem'
    },
    clients: {
        '192.168.0.100':{
            targetServerId: '192.168.0.102',
            auth:{
                type: 'oauth',//oauth,basic
                clientId: 'predix',
                clientSecret: 'hellopredix',
                authUrl: 'predix.io/oauth/token'
            }
        },
        '192.168.0.101':{
            targetServerId: '192.168.0.103',
            auth:{
                type: 'basic',
                secret: '1234'
            }
        }
    },
    servers: {
        '192.168.0.102':{
            auth:{
                type: 'oauth',//oauth,basic
                clientId: 'predix',
                clientSecret: 'hellopredix',
                authUrl: 'predix.io/oauth/token'
            }
        },
        '192.168.0.103':{
            auth:{
                type: 'basic',
                secret: '143434'
            }
        }
    },
    groups: {
        'ge-internal':['192.168.0.100','192.168.0.101'],
        'osaka-network':
['192.168.0.102','192.168.0.103','192.168.0.101'],
        'tamakura-wifi':['192.168.0.101','192.168.0.103']
    },
```

```
    keepAlive:60000, //0 Always
});

//command: DEBUG=rs:gateway node gateway

```


###client
```javascript
var RSClient=require('./../rs-client');

var phs=new RSClient({
    //proxy service. remove _ to activate
    _proxy:{
        host:'proxy-src.research.ge.com',
        port:8080
    },
    localPort:7989,
    //gatewayHost:'wss://rsgateway-perrin.run.aws-usw02-
pr.ice.predix.io',
    //gatewayPort:443,
    gatewayHost: 'wss://localhost',
    gatewayPort: 9898,
    targetServerId:'192.168.0.103',//empty/id/ip
    id: '192.168.0.101',
    secret: '1234', //Presented if requested by gateway. This would be
your the secret for the Basic auth
    oauthToken: 'rwerwerr34r34r34r' //Presented if requested by
gateway. This would be your UAA/OAuth token
});

//command: DEBUG=rs:client node client

```


###server
```javascript
var RSServer=require('./../rs-server');

var phs=new RSServer({
    //proxy service. remove _ to activate
    _proxy:{
        host:'proxy-src.research.ge.com',
        port:8080
    },

    //gatewayHost: 'wss://rsgateway-chia.run.aws-usw02-
pr.ice.predix.io',
    //gatewayPort: 443,
    gatewayHost: 'wss://localhost',
```

```
    gatewayPort: 9898,
    resourceHost: 'localhost', //no protocol prefix. this's always tcp
    resourcePort: 21,
    id: '192.168.0.103', //Reachback service credential
    secret: '143434', //Presented if requested by gateway. This would
be your the secret for the Basic auth
    oauthToken: 'rwerwerr34r34r34r' //Presented if requested by the
gateway. This would be your UAA/OAuth token
});

//command: DEBUG=rs:server node server
```

###Event handling
####gateway
"connection_accepted"
"session_create"
"session_close"
"port_error"
"port_close"
"session_join"

###server
to be cont.

###client
to be cont.

###environment variables DEBUG
####gateway
DEBUG=rs:gateway

####client
DEBUG=rs:client

####server
DEBUG=rs:server

##Reference
####Project Home:
https://github.build.ge.com/pages/212359746/ph-connectivity-node-service

####Github:
https://github.build.ge.com/pages/212359746/ph-connectivity-node-service/

####Design/Prototype:
https://github.build.ge.com/212359746/ph-tcp-tunnel/blob/master/docs/

poc-tcp-tunneling.pptx?raw=true

####Demo:

#####MongoDB
https://predix-cs-portal.run.asv-pr.ice.predix.io/mongo/listcats (List rows)

https://predix-cs-portal.run.asv-pr.ice.predix.io/mongo/addcat/:name/:type (Add row)

#####PostgresSQL
https://predix-cs-portal.run.asv-pr.ice.predix.io/pg/listbirds (List rows)

https://predix-cs-portal.run.asv-pr.ice.predix.io/pg/addbird/:name/:type (Add row)

####Release notes:
[v0.5-alpha:](https://github.build.ge.com/Reachback-Services/rs-connectivity-service/releases/tag/v0.5-alpha)
 - Proxy service fixes.
 - Add SSL/TLS for WS:// or WSS:// call on both Server/Client component.
 - README updates.
 - Usage Model Diagram updates.

[v0.4-alpha:](https://github.build.ge.com/Reachback-Services/rs-connectivity-service/releases/tag/v0.4-alpha)
 - Security enhancement.
 - OAuth/OAuth2 add-on.
 - Predix UAA compatible.
 - Multi-tendant usage enhancement.
 - Fix potential orphan connections.
 - Modulise the libraries.
 - Support Proxy settings both on client/server side.

[v0.3-alpha:](https://github.build.ge.com/Reachback-Services/rs-connectivity-service/releases/tag/v0.2-alpha) Adding the concept of three-way network connectivity (multi-clients vs. gateway vs. multi-resources/servers)
[v0.2-alpha:](https://github.build.ge.com/Reachback-Services/rs-connectivity-service/releases/tag/v0.3-alpha) In this release we introduced the concept of network connectivity between multi-clients and single on-premise server.