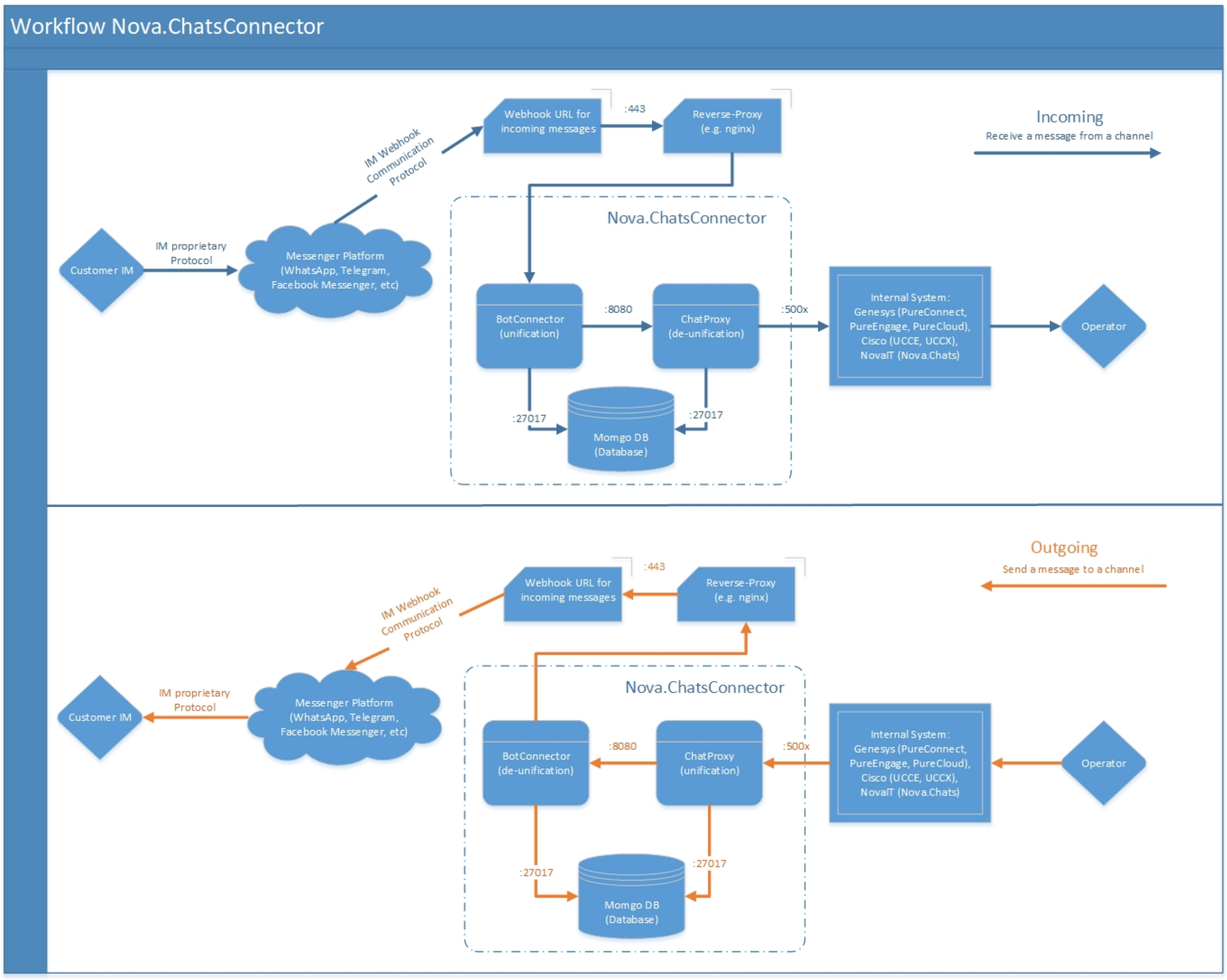
Products Family

Components

Architecture


Workflow Nova.ChatsConnector
Incoming
Receive a message from a channel
Customer IM
IM proprietary Protocol
Messenger Platform (WhatsApp, Telegram, Facebook Messenger, etc)
IM Webhook Communication Protocol
Webhook URL for incoming messages
:443
Reverse-Proxy (e.g. nginx)
Nova.ChatsConnector
BotConnector (unification)
:8080
ChatProxy (de-unification)
:500x
:27017
Momgo DB (Database)
:27017
Internal System: Genesys (PureConnect, PureEngage, PureCloud), Cisco (UCCE, UCCX), NovaIT (Nova.Chats)
Operator
Outgoing
Send a message to a channel
Customer IM
IM proprietary Protocol
Messenger Platform (WhatsApp, Telegram, Facebook Messenger, etc)
IM Webhook Communication Protocol
Webhook URL for incoming messages
:443
Reverse-Proxy (e.g. nginx)
Nova.ChatsConnector
BotConnector (de-unification)
:8080
ChatProxy (unification)
:500x
:27017
Momgo DB (Database)
:27017
Internal System: Genesys (PureConnect, PureEngage, PureCloud), Cisco (UCCE, UCCX), NovaIT (Nova.Chats)
Operator

Installation

Disable SE Linux

1. Check status `# sestatus`
2. `# sudo setenforce 0`
3. Open the `/etc/selinux/config` file and set the `SELINUX` mod to `disabled`
4. Restart and check

MongoDB Installation

1. Add MongoDB repo `# vi /etc/yum.repos.d/mongodb.repo`

   ```
    [MongoDB]
    name=MongoDB Repository
    baseurl=http://repo.mongodb.org/yum/redhat/$releasever/mongodb-
   org/4.2/$basearch/
    gpgcheck=1
    enabled=1
    gpgkey=https://www.mongodb.org/static/pgp/server-4.2.asc
   ```

2. Install daemon `# yum install mongodb-org`

3. Start daemon `# systemctl start mongod.service`

4. Check TCP Port 27017 is listened `netstat -anp | grep 27017` and try to connect through Robo 3T Client (ssh tunnel needed)

Nginx Installation

1. Install `# yum install nginx`

2. Updating Diffie-Hellman Parameters

   If you test your server using the SSL Labs Server Test now, it will only get a B grade due to weak Diffie-Hellman parameters. This effects the security of the initial key exchange between our server and its users. We can fix this by creating a new dhparam.pem file and adding it to our server block.

   ```
   # sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
   ```

3. Create and modify config per site:

   ```
   # vim /etc/nginx/conf.d/newsite.conf

   server {
   server_name  newsite.com;
   listen 443 ssl;
   ```

```
    ssl_protocols TLSv1.1 TLSv1.2;
    ssl_certificate "/etc/pki/nginx/..";
    ssl_certificate_key "/etc/pki/nginx/..";
    ssl_dhparam "/etc/ssl/certs/dhparam.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    access_log  /var/log/nginx/newsite.com.acces.log  main;

    # Nova.ChatsConnector BotConnector Webhooks
    location /webhook {
    proxy_set_header    Host $host;
    proxy_set_header    X-Real-IP $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto $scheme;
    proxy_pass          http://127.0.0.1:8080;
    proxy_read_timeout  30;
    }

    # Nova.ChatsConnector BotConnector Download Files
    location / {
        root              /var/lib/nginx/nova.chatsconnector;
        try_files         $uri @defaulthome;

        expires           max;
        #access_log        off;
    }

    # Nova.ChatsConnector BotConnector Default Home
    location @defaulthome {
        root              /usr/share/nginx/html;
        try_files         $uri $uri/index.html $uri.html index.html =404;

        expires           max;
    }

        error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
    }

    server {
        server_name  newsite.com;
        listen       80;
        return 301 https://$host$request_uri;
    }
```

4. Modify standart config:

```
    # vim /etc/nginx/nginx.conf

    user nginx;
    worker_processes auto;
    error_log /var/log/nginx/error.log;
    pid /run/nginx.pid;

    # Load dynamic modules. See /usr/share/nginx/README.dynamic.
    include /usr/share/nginx/modules/*.conf;

    events {
        worker_connections 1024;
    }

    http {
        log_format  main  '$remote_addr - $remote_user [$time_local]
  "$request" '
                          '$status $body_bytes_sent "$http_referer" '
                          '"$http_user_agent" "$http_x_forwarded_for"';

        access_log  /var/log/nginx/access.log  main;
        # Security
        server_tokens off;

        sendfile            on;
        tcp_nopush          on;
        tcp_nodelay         on;
        keepalive_timeout   65;
        types_hash_max_size 2048;

        # Optimization
        gzip  on;
        gzip_types application/javascript image/* text/css;
        gunzip on;

        include             /etc/nginx/mime.types;
        default_type        application/octet-stream;

        # Load modular configuration files from the /etc/nginx/conf.d
  directory.
        # See http://nginx.org/en/docs/ngx_core_module.html#include
        # for more information.
        include /etc/nginx/conf.d/*.conf;
    }
```

5. Validate nginx configuration

```
    # nginx -t
```

6. Check nginx functionality.

Node.JS Installation

1. Install *Stable* Release

```
 # yum install -y gcc-c++ make
 # curl -sL https://rpm.nodesource.com/setup_12.x | sudo -E bash -
 # sudo yum install nodejs
```

2. Check versions

```
 # node -v
 # npm -v
```

.Net CORE Installation (PureConnect Only)

Prerequisites 2.0.x Core (Microsoft article)

Install 2.0.x Core (Microsoft article)

1. Add Repos and dependencies

```
 # rpm -Uvh https://packages.microsoft.com/config/rhel/7/packages-microsoft-
prod.rpm
 # yum install dotnet-sdk-2.0.3

============================================================================
===================================
 Package                          Arch            Version
Repository                             Size

============================================================================
===================================
 Installing:
 dotnet-sdk-2.0.3                 x86_64          2.0.3-1
```

```
packages-microsoft-com-prod           86 M
 Installing for dependencies:
 aspnetcore-store-2.0.0         x86_64          2.0.0-1
packages-microsoft-com-prod           24 M
 aspnetcore-store-2.0.3         x86_64          2.0.3-1
packages-microsoft-com-prod          7.9 M
 dotnet-host                    x86_64          2.2.5-1
packages-microsoft-com-prod           44 k
 dotnet-hostfxr-2.0.3           x86_64          2.0.3-1
packages-microsoft-com-prod          182 k
 dotnet-runtime-2.0.3           x86_64          2.0.3-1
packages-microsoft-com-prod           24 M

 Transaction Summary

============================================================================
===================================
 Install  1 Package (+5 Dependent packages)
```

2. Validate Installation
   Try to RUN PureConnect IWT ChatProxy

NovaChatsConnector Components Installation

All of Packages install from RPM.

```
    # rpm -ivh ${nova-component}.rpm
```

Update for all Packages install from RPM. Before updating backup is recommended.

```
    # rpm -Uvh ${nova-component}.rpm
```

Administration

Chatproxy manages by `systemd`. Some commands:

Operation	Command
status	`systemctl status ${daemon}.service`
restart	`systemctl restart ${daemon}.service`
stop	`systemctl stop ${daemon}.service`
start	`systemctl start ${daemon}.service`
view service logs	`journalctl -xe -u ${daemon}.service`

Operation	Command
check net use	`netstat -anp | grep LISTEN`

Maintenance and Support

Setup Components

Nova.BotConnector

- *Introduction*

  Nova.BotConnector allows you to connect your bot to multiple messaging channels.

  - ***Core*** component. Provides routing messages from outside to the another components and back. Works as a Linux daemon behind Reverse-proxy.
  - The ***Core*** is requested by Webhook from outside (from Messengers).
  - The ***Core*** is requested by `POST | PUT | DELETE` requests from inside (e.g. `curl`, another components).

- *Installation*

  1. Install Component

```
  # rpm -ivh $HOME/nova-botconnector-2020_R1-1.x86_64.rpm
```

  2. Modify and enable service

```
  # systemctl enable nova-botconnector.service
  # vim /etc/systemd/system/multi-user.target.wants/nova-
botconnector.service

  [Unit]
  Description=Nova.BotConnector service
  After=network.target mongodb.service

  [Service]
  ExecStart=/usr/bin/node /opt/nova-botconnector/src/index.js
  WorkingDirectory=/opt/nova-botconnector
  Restart=on-failure
  StandardOutput=syslog
  StandardError=syslog
  SyslogIdentifier=botconnector
  User=botconnector
  Group=botconnector
```

```
Environment=NODE_ENV=production

[Install]
WantedBy=multi-user.target
```

The last option `Environment` in `[Service]` section means what config service have to use Production or Development config file.

3. Configure your Nova.BotConnector

```
#vim /opt/nova-botconnector/conf/production.js
```

Section	Parameter	Value	Explanation
`db`	`uri`	*URL*	connection to Mongo. Replica-set Supported
`db`	`options`	*JSON*	default
`server`	`port`	*int*	application bind port
`server`	`host`	*string*	application bind IP
`server`	`base_url`	*URL*	base URL for webhook, must be FQDN
`server`	`cleanOlder`	*int*	remove conversations older than x day

```
module.exports = {
db: {
    uri: 'mongodb://127.0.0.1:27017/nova-botconnector',
    debugMode: false,
    options: {
    useNewUrlParser: true,
    reconnectTries: Number.MAX_VALUE,
    reconnectInterval: 1000,
    poolSize: 10,
    connectTimeoutMS: 5000,
    family: 4,
    keepAlive: true,
    keepAliveInitialDelay: 300000,
    },
},
server: {
    port: '8080',
    host: '127.0.0.1',
    base_url: 'https://yourservername.com',
    cleanOlder: 7
},
logging: { // log4js-node configuration
```

```
        appenders: {
            file: {
                type: 'file',
                filename: '/var/log/nova-botconnector/nova-botconnector.log',
                maxLogSize: 52428800, // 50 MB
                backups: 5,
                compress: true,
                keepFileExt: true
            }
        },
        categories: {
            default: {appenders: ['file'], level: 'debug'}
        }
    },
    }
```

4. Start your service

```
    # systemctl start nova-botconnector
    # systemctl status nova-botconnector
```

5. Create New Connector Instance of Nova.BotConnector

- *Administration*

1. New Connector Instance

You have to decide what TCP Port will be used and set into request

```
    # curl -X POST 'http://localhost:8080/connectors' --data
  'url=http://localhost:5000'
```

You will receive response with UUID `${CONNECTOR_ID}` of new Connector. Remember this value (or found in mongodb).

In case with Genesys PureConnect command will be:

```
    # curl -X POST 'http://localhost:8080/connectors' --data
  'url=http://localhost:5000/api/message'
```

2. Add Channels

- ***Trough Nova Chats API (WhatsApp, Skype, Instagram)***

Before adding channel, you have to Create it on Nova.Chats

- ***Nova Chats***

```
curl -X  POST \
--data "slug=WhatsappPureConnect" \
--data "type=novachats" \
--data "token=c4436e57856ff976e9473b75f1bbe2" \
--data "isActivated=true" \
"http://localhost:8080/connectors/${CONNECTOR_ID}/channels"
```

- ***Direct API***

  - ***Telegram***

```
curl -X POST  \
--data "isActivated=true" \
--data "slug=TelegramSlug" \
--data "token=652278712:AAGI8FqQwGPa_K4Gf-smLvZXx4zbqk8Q4is" \
--data "type=telegram" \
"http://localhost:8080/connectors/${CONNECTOR_ID}/channels"
```

  - ***Viber***

```
curl -X POST  \
--data "isActivated=true" \
--data "slug=BiberSlug" \
--data "token=49674f056da7d074-c09d50d64d30605e-2e3da1df1adaa0ac"
\
--data "type=viber" \
--data "userName=NovaIT" \
"http://localhost:8080/connectors/${CONNECTOR_ID}/channels"
```

  - ***Twilio***

```
curl -X POST  \
--data "isActivated=true" \
--data "slug=WhatsAppSlug" \
--data "phoneNumber=whatsapp:+14155238886" \
--data "clientId=id" \
--data "clientSecret=secret" \
--data "type=whatsapp" \
"http://localhost:8080/connectors/${CONNECTOR_ID}/channels"
```

- ***Nexmo***

  ```
  curl -X POST  \
  --data "slug=NexmoDemo" \
  --data "isActivated=true" \
  --data "type=nexmo" \
  --data "token=8llntawuntawun0qxt" \
  --data "phoneNumber=447418342149" \
  "http://localhost:8080/connectors/${CONNECTOR_ID}/channels"
  ```

- ***Infobip***

  ```
  curl -X POST  \
  --data "slug=InfobipSlug" \
  --data "isActivated=true" \
  --data "type=infobip" \
  --data "userName=Nova-It" \
  --data "password=GRvsQfdfU22frs6U" \
  --data "phoneNumber=447491163530" \
  --data "accountKey=C6321E1DCFB76208EA8CEB665FA1D48A" \
  "http://localhost:8080/connectors/${CONNECTOR_ID}/channels"
  ```

- ***WeChat***

  ```
  curl -X POST \
  --data "slug=WeChat" \
  --data "isActivated=true" \
  --data "type=wechat" \
  --data "appId=appID" \
  --data "appSecret=secret" \
  --data "token=12345678901" \
  --data "userName=wechat" \
  "http://localhost:8080/connectors/{CONNECTOR_ID}/channels"
  ```

- ***WebChat***

  ```
  curl -X POST  \
  --data "slug=WebChat" \
  --data "isActivated=true" \
  --data "type=webchat" \
  "http://localhost:8080/connectors/${CONNECTOR_ID}/channels"
  ```

3. Send test message and view logs according to *production/development.js*

More information see in next section.

4. Connectivity checking

First of all, check that all the needed service started and successfully run.

- Nginx (or another reverse-proxy)
- BotConnector
- ChatProxy

  Also, check that bound Ports are LISTENING.

The right way to check how it works - register one of the simplest channel, Telegram, and write a few messages to bot.

After that you can check the logs:

1. First logs can be `Nginx` logs. You can see how messages arrive and request a webhook.
2. BotConnector log `$LOG/nova-botconnector.log`.
3. ChatProxy log `$LOG/nova-chatproxy.log`.

If the log doesn't give clear/detailed information - try to change Log Level, according to using Logger Specs (log4js, nlog).

UUIDs that help to track messaging.

Entity	Meaning
`ConnectorID`	First UUID in one log string, ID of Connector Instance
`conversation`	ID of conversation on kept trough all logs
`source`	type of channel

5. Some commands

ChatpRouter manages by `systemd`. Usefull commands are:

Operation	Command
check status	`systemctl status nova-botconnector.service`
restart	`systemctl restart nova-botconnector.service`
restart	`systemctl stop nova-botconnector.service` `systemctl start nova-botconnector.service`
view service logs	`journalctl -xe -u nova-botconnector.service`
check net use	`netstat -anp | grep LISTEN`

- *Maintenance and Support*

  Updating. Installing new release

  1. Stop service
  2. Copy current release with all files into another place:

     ```
     # cp /opt/nova-botconnector/ $HOME/backup
     ```

  3. Install update from by RPM

     ```
     # rpm -Uvh nova-botconnector-2020_R1-1.x86_64.rpm
     ```

  4. Compare new configs with old one, add a new setting, if they exist and copy modified configs into folder with new installation.
  5. Start service and check connectivity

Nova.ChatProxy.Genesys.PureConnect.ICWS

- *Introduction*

Nova.ChatProxy.Genesys.PureConnect allows you to connect Nova.BotConnector with Genesys PureConnect by using ICWS API.

- *Installation*

1. Install Component

   ```
   # rpm -ivh $HOME/nova-chatproxy-genesys-pureeconnect-2020_R1-1.x86_64.rpm
   ```

2. Configure your Nova.ChatProxy

   ```
   # cd /opt/nova-chatproxy-genesys-pureeconnect/config/
   # cp chatproxy.js nova-chatproxy-genesys-pureeconnect.js
   ```

   Component's Settings:

Section	Parameter	Value	Explanation
`db`	`uri`	*URL*	connection to Mongo. Replica-set Supported
`db`	`options`	*JSON*	default

Section	Parameter	Value	Explanation
`genesys`	`endpoint`	*URL*	
`chat`	`targetName`	*string*	Target Genesys Workgroup
`chat`	`firstName` `lastName`	*string*	from Who you will be see chat-session
`chat`	`additionalAttributes`	*JSON*	A collection of any additional attributes associated with the interaction. These can be standard CIC attribute names or custom attribute names to support custom handlers / integrations
`chat`	`pollingInterval`	*int*	Pooling timeout in miliseconds
`chat`	`systemMessages`	*boolean*	enable or disable
`chat`	`welcomeMessage` `waitForAgentMessage` `chatEndMessage`	*string*	enables messages from internal settings `/opt/nova-chatproxy-cisco-ece/resources/messages.json`; filename must be like Enviroment variable in SYSTEMD.
`app`	`host`	*string*	own host
`app`	`port`	*int*	own port
`app`	`endQueuedSessions`	*int*	how quickly QUEUED session have will be deleted, in *min*
`app`	`endDeadSessions`	*int*	how quickly DEAD session have will be deleted in *min*
`app`	`downloadFolder`	*string*	local directory where media will be stored
`app`	`downloadURLPrefix`	*string*	prefix for formation full URL of stored media
`app`	`downloadMaxSize`	*int*	max size of download media. in *Bytes*
`connector`	`url`	*URL*	Internal BotConnector's URL
`connector`	`connectorId`	*UUID*	The ID of Botconnector Instance what have to proccess messages from ChatProxy

```
 module.exports = {
 db: {
         uri: 'mongodb://127.0.0.1:27017/nova-chatproxy-genesys-pureconnect-
icws',
         debugMode: false,
         options: {
         useNewUrlParser: true,
         useFindAndModify: false,
         reconnectTries: Number.MAX_VALUE,
         reconnectInterval: 5000,
		poolSize: 10,
		connectTimeoutMS: 5000,
		family: 4,
		keepAlive: true,
		keepAliveInitialDelay: 300000,
		}
	},
	genesys: {
		endpoint: 'https://site.com/api/site.inin.local',
		timeout: 5000
	},
	chat: {
		targetName: 'Workgroup',
		firstName: 'Nova',
		lastName: 'ChatProxy',
		additionalAttributes: {
		chatApplication: 'Nova.ChatProxy',
		},
		pollingInterval: 2000,
		systemMessages: true,
		welcomeMessage: false,
		waitForAgentMessage: false,
		chatEndMessage: false,
	},
	app: {
		host: '127.0.0.1',
		port: 5000,
		endQueuedSessions: 3,
		endDeadSessions: 1440,
		downloadFolder: '/var/lib/nginx/downloads/',
		downloadURLPrefix: ''https://yourservername.com/',
		downloadMaxSize: 52428800 // 50 MB
	},
	connector: {
		url: 'http://127.0.0.1:8080/connectors/',
		connectorId: '01b74904-71c9-4dec-9a0c-def2115989ca',
		timeout: 5000
	},
	logging: { // log4js-node configuration
		appenders: {
		file: {
			type: 'file',
			filename: '/var/log/nova-chatproxy-genesys-
pureconnect/chatproxy.log',
			maxLogSize: 52428800, // 50 MB
			backups: 5,
			compress: true,
			keepFileExt: true
		}
		},
		categories: {
```

```
            default: {appenders: ['file'], level: 'trace'}
            }
    }
    }
```

3. Modify and enable service

```
    # cd /etc/systemd/system/
    # vim /etc/systemd/system/nova-chatproxy-genesys-pureconnect.service

    [Unit]
    Description=Genesys PureConnect using ICWS API proxy for Nova.BotConnector.
    After=network.target mongodb.service

    [Service]
    ExecStart=/usr/bin/node /opt/nova-chatproxy-genesys-
   pureconnect/src/index.js
    WorkingDirectory=/opt/nova-chatproxy-genesys-pureconnect
    Restart=on-failure
    RestartSec=30
    StandardOutput=syslog
    StandardError=syslog
    SyslogIdentifier=nova-chatproxy-genesys-pureconnect
    User=botconnector
    Group=botconnector
    Environment=CHATPROXY_INSTANCE=nova-chatproxy-genesys-pureconnect

    [Install]
    WantedBy=multi-user.target

    # systemctl enable nnova-chatproxy-genesys-pureconnect.service
    # chown botconnector.botconnector /etc/systemd/system/nova-chatproxy-
   genesys-pureconnect.service
    # chown botconnector.botconnector -R /opt/nova-chatproxy-genesys-
   pureconnect/config/
    # systemctl start nova-chatproxy-genesys-pureconnect.service
```

The last option `Environment` in `[Service]` section means what config service have to use from config files.

- *Administration*

Connectivity checking

First of all, check that the needed ChatProxy service started and successfully run.

Check bound Port is LISTEN.

Check ChatProxy log and systemd unit log (`journalctl -xe -u nova-chatproxy-genesys-pureconnect.service`)

If the log doesn't give clear/have detailed information - try to change Log Level, according to using Logger Specs (log4js).

UUIDs that help to track messaging.

Entity	Meaning
`ConnectorID`	First UUID in one log string, ID of Connector Instance
`conversation`	ID of conversation on kept trough all logs
`source`	type of channel

Some commands

ChatpRouter manages by `systemd`. Usefull commands are:

Operation	Command
check status	`systemctl status nova-chatproxy.service`
restart	`systemctl restart nova-chatproxy.service`
restart	`systemctl stop nova-chatproxy.service` `systemctl start nova-chatproxy.service`
view service logs	`journalctl -xe -u nova-chatproxy.service`
check net use	`netstat -anp | grep LISTEN`

- *Maintenance and Support*

Updating. Installing new release

1. Stop service
2. Copy current release with all files into another place:

```
# cp /opt/nova-chatproxy-genesys-pureconnect/ $HOME/backup
```

3. Install update from by RPM

```
# rpm -Uvh nova-chatproxy-genesys-pureconnect-2020_R1-1.x86_64.rpm
```

4. Compare new configs with old one, add a new setting, if they exist and copy modified configs into folder with new installation.

5. Start service and check connectivity

Nova.ChatProxy.Genesys.PureConnect.IWT

- *Introduction*

Nova.ChatProxy.Genesys.PureConnect allows you to connect Nova.BotConnector with Genesys PureConnect by using Interaction Web Tools API.

- *Installation*

Nova.ChatProxy.Genesys.PureConnect.IWT developed by .NET. RPM doesn't supported. Each brand new Instance of Chatproxy IWT copied from tar.gz.

Before deploying Instance you have to Install .Net Core.

1. Untar Nova.ChatProxy_v1.7.0.0.zip into /opt/chatProxyPureConnectIWT

2. Configure Port.
   You have to decide, what port it will be. This port only for Internal user between components.

```
            # vim /opt/chatProxyPureConnectIWT/hosting.json
            {
            "urls": "http://127.0.0.1:5000"
            }
```

3. Configure Logs.
   You have to decide where it will be. For Logging solution uses NLog and have XML structure.
   Modify `nlog variables` for defining place for logs.

```
    <?xml version="1.0" encoding="utf-8" ?>
    <nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        autoReload="true"
        internalLogLevel="Warn"
        internalLogFile="internal-nlog.txt">

    <!-- Load the ASP.NET Core plugin -->
    <extensions>
        <add assembly="NLog.Web.AspNetCore"/>
    </extensions>

    <variable name="logDirectory" value="logs/${shortdate}" />
    <variable name="logRootDirectory" value="./logs" />
    <!-- the targets to write to -->
    <targets>
        <!-- write logs to file -->
        <target xsi:type="File"
                name="allfile"
                archiveEvery ="Hour"
                createDirs="True"
                archiveFileName="${logRootDirectory}/archive/nlog-all_{##}.log"
                archiveNumbering="Rolling"
                maxArchiveFiles="5"
                fileName="${logRootDirectory}/nlog-all.log"
                layout="${longdate}|${uppercase:${level}}|${event-
properties:item=EventId.Id}|${logger}|${message} ${exception}" />

        <!-- another file log, only own logs. Uses some ASP.NET core renderers
-->
        <target xsi:type="File"
                name="ownFile-web"
                archiveEvery ="Hour"
                archiveFileName="${logRootDirectory}/archive/nlog-all_{##}.log"
                createDirs="True"
                archiveNumbering="Rolling"
                maxArchiveFiles="5"
                fileName="${logRootDirectory}/nlog-chatProxy.log"
                layout="${longdate}| ${uppercase:${level}}| ${callsite} |
${message} ${exception}" />

        <!-- write to the void aka just remove -->
        <target xsi:type="Null" name="blackhole" />
    </targets>

    <!-- rules to map from logger name to target -->
    <rules>
```

4. Configure ChatProxy

   Component's Settings:

Section	Parameter	Value	Explanation
`MongoDB`	`Client`	*URI*	connection to Mongo. Replica-set Supported
`MongoDB`	`Database`	*string*	DB name
`BotConnector`	`Url`	*URL*	from BotConnector settings. Where the BotConnector LISTEN as a Daemon

Section	Parameter	Value	Explanation
`BotConnector`	`ConnectorId`	*UUID*	To which BotConnector ChatProxy have to route messages from Genesys side
`BotConnector`	`Timeout`	*int*	how long waint response from BotConnector
`BotConnector`	`FirstHandler`	*int*	default
`BotConnector`	`DownloadFolder`	*int*	Where sended media will be stored
`BotConnector`	`DownloadURLPrefix`	*int*	Places this address for full URL in Response to Customer
`InternalConnectors`	`Connector.Url`	*string*	Main Route. Can be directly to CIC. HTTP or HTTPS
`InternalConnectors`	`Connector.UrlAlternative`	*string*	Backup Route. Can be directly to CIC. HTTP or HTTPS
`InternalConnectors`	`Connector.Workgroup`	*string*	Workgroup on Genesys side
`InternalConnectors`	`Connector.InactiveIntervalMinutes`	*int*	How long keep connection to Genesys side
`InternalConnectors`	`Connector.UserNamePattern`	*string*	What data can be transfer into Name on Genesys Side

```
{
"Configuration": {
    "MongoDB": {
    "Client": "mongodb://localhost:27017",
    "Database": "ChatProxyNovaPureConnect"
    },

    "BotConnector": {
    "Url": "http://127.0.0.1:8080",
    "ConnectorId": "786b4297-e5aa-439c-a8f7-37d8dd61c836",
    "Timeout": 10,
    "FirstHandler": "Genesys_PureConnect_Ex_Chat_Queue",
    "DownloadFolder": "/var/lib/nginx/downloads/",
    "DownloadURLPrefix": "https://novachatsconnector.demo.novait.com.ua"
    },

    "InternalConnectors": [
    {
        "Name": "Genesys_PureConnect_Ex_Chat_Queue",
        "Type": "Genesys.PureConnect.Ex",
        "Connector": {
        "Url": "https://192.168.1.1/I3Root/Server1",
        "UrlAlternative": "https://192.168.1.2/I3Root/Server1",
        "Workgroup": "W_Nova.ChatsConnector_Queue",
        "InactiveIntervalMinutes": 1,
        "UserNamePattern": "{chatid}, {senderid}, {__v}, {_id},
{participant}, {conversation}, {receivedat}, {userid}, {username},
{lastname}, {firstname}, {source}, {type}"

        }
    }
    ]
 }
 }
```

5. Create Service

```
# cd /etc/systemd/system/
# vim /etc/systemd/system/nova-chatproxy-iwt.service

[Unit]
Description=Chat Proxy for bot connector. Customer: IWT

[Service]
WorkingDirectory=/opt/chatProxyPureConnectIWT
ExecStart=/usr/bin/dotnet /opt/chatProxyPureConnectIWT/ChatProxy.dll
Restart=always
RestartSec=10  # Restart service after 10 seconds if dotnet service crashes
SyslogIdentifier=chatproxy-pivotalgroup
User=root
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

[Install]
WantedBy=multi-user.target


# systemctl enable nova-chatproxy-iwt.service
# chown botconnector.botconnector /etc/systemd/system/nova-chatproxy-
```

```
iwt.service
 # chown botconnector.botconnector -R /opt/nova-chatproxy-iwt/config/
 # systemctl start nova-chatproxy-iwt.service
```

The last option `Environment` in `[Service]` section means what config service have to use from config files.

- *Administration*

Connectivity checking

First of all, check that the needed ChatRouter service started and successfully run.

Check bound Port is LISTEN.

Check ChatProxy log and systemd unit log (`journalctl -xe -u nova-cahtrouter.service`)

If the log doesn't give clear/have detailed information - try to change Log Level, according to using Logger Specs (nlog).

UUIDs that help to track messaging.

Entity	Meaning
`ConnectorID`	First UUID in one log string, ID of Connector Instance
`conversation`	ID of conversation on kept trough all logs
`source`	type of channel

Some commands

ChatpRouter manages by `systemd`. Usefull commands are:

Operation	Command
check status	`systemctl status nova-chatproxy.service`
restart	`systemctl restart nova-chatproxy.service`
restart	`systemctl stop nova-chatproxy.service` `systemctl start nova-chatproxy.service`
view service logs	`journalctl -xe -u nova-chatproxy.service`
check net use	`netstat -anp | grep LISTEN`

- *Maintenance and Support*

Updating. Installing new release

1. Stop service

2. Copy current release with all files into another place:

   ```
   # cp /opt/nhatProxyPureConnectIWT/ $HOME/backup
   ```

3. Install update by coping from tat.gz to working directory.

4. Compare new configs with old one, add a new setting, if they exist and copy modified configs into folder with new installation.

5. Start service and check connectivity

Nova.ChatProxy.Genesys.PureEngage

- *Introduction*

Nova.ChatProxy.Genesys.PureEngage allows you to connect Nova.BotConnector with Genesys PureEngage.

- *Installation*

1. Install Component

   ```
   # rpm -ivh $HOME/nova-chatproxy-genesys-pureengage-2020_R2-1.x86_64.rpm
   ```

2. Configure your Nova.ChatProxy

   ```
   # cd /opt/nova-chatproxy-genesys-pureegage/config/
   # cp chatproxy.js nova-chatproxy-genesys-pureengage.js
   ```

   Component's Settings:

Section	Parameter	Value	Explanation
`db`	`uri`	*URL*	connection to Mongo. Replica-set Supported
`db`	`options`	*JSON*	default
`genesys`	`endpoint`	*URL*	Path to internal or external endpoint
`chat`	`serviceName`	*string*	Service Name on Genesys endpoint (Queue)
`chat`	`firstname`	*string*	see the example below. Can be Pattern for sending channel info to end system
`chat`	`lastname`	*sstring*	see the example below. Can be Pattern for sending channel info to end system

Section	Parameter	Value	Explanation
`chat`	`attachedUserData`	*JSON*	Could be added additional custom parameters for sending into Genesys
`chat`	`memberInfo`	*JSON*	patterns, that can be send to customer side
`chat`	`welcomeMessage` `waitForAgentMessage` `chatEndMessage` `systemMessages`	*boolean*	enables messages from internal settings `/opt/nova-chatproxy-genesys-purecloud/messages.json`; filename must be like Enviroment variable in SYSTEMD
`chat`	`pollingInterval`	*int*	how frequetly endpoint will be requested (in milisec)
`app`	`host`	*string*	own host
`app`	`port`	*int*	own port
`app`	`downloadFolder`	*string*	local directory where media will be stored
`app`	`downloadURLPrefix`	*string*	prefix for formation full URL of stored media
`app`	`downloadMaxSize`	*int*	max size of download media. in *Bytes*
`connector`	`url`	*URL*	Internal BotConnector's URL
`connector`	`connectorId`	*UUID*	The ID of Botconnector Instance what have to proccess messages from ChatProxy

```
module.exports = {
db: {
        uri: 'mongodb://127.0.0.1:27017/nova-chatproxy-genesys-pureengage',
        debugMode: false,
        options: {
        useNewUrlParser: true,
        useFindAndModify: false,
        reconnectTries: Number.MAX_VALUE,
        reconnectInterval: 5000,
        poolSize: 10,
        connectTimeoutMS: 5000,
        family: 4,
        keepAlive: true,
        keepAliveInitialDelay: 300000,
        }
},
genesys: {
        endpoint: 'https://gbank.demo.genesys.com/gms_port_8010',
        timeout: 5000,
},
chat: {
        serviceName: 'customer-support',
        firstName: '{userid}',
        lastName: 'ChatProxy | {source}',
        attachedUserData: {
        pfs_id: '380672189796',
        channel: '{source}',
        },
        pollingInterval: 3000,
        welcomeMessage: false,
        waitForAgentMessage: false,
        chatEndMessage: false,
 },
 app: {
        host: '127.0.0.1',
        port: 5000,
        downloadFolder: '/var/lib/nginx/downloads/nccpengage',
        downloadURLPrefix:
'https://novachatsconnector.demo.novait.com.ua/nccpengage',
        downloadMaxSize: 52428800 // 50 MB
 },
 connector: {
        url: 'http://127.0.0.1:8080/connectors/',
        connectorId: 'd4e38610-ac6a-43f3-8222-78e44fb46512',
        timeout: 5000
 },
 logging: { // log4js-node configuration
        appenders: {
        file: {
                type: 'file',
                filename: '/var/log/nova-chatproxy-genesys-
pureengage/gdemo/nova-chatproxy.log',
                maxLogSize: 52428800, // 50 MB
                backups: 5,
                compress: true,
                keepFileExt: true
        }
        },
        categories: {
        default: {appenders: ['file'], level: 'trace'}
        }
 }
 }
```

3. Modify and enable service

```
 # cd /etc/systemd/system/
 # vim /etc/systemd/system/nova-chatproxy-genesys-pureengage.service

 [Unit]
 Description=Genesys PureEngage proxy for Nova.BotConnector
 After=network.target mongodb.service
 [Service]
 ExecStart=/usr/bin/node /opt/nova-chatproxy-genesys-pureengage/src/index.js
 WorkingDirectory=/opt/nova-chatproxy-genesys-pureengage
 Restart=on-failure
 RestartSec=30
 StandardOutput=syslog
 StandardError=syslog
 SyslogIdentifier=nova-chatproxy-genesys-pureengage
 User=botconnector
 Group=botconnector
 Environment=CHATPROXY_INSTANCE=nova-chatproxy-genesys-pureengage

 [Install]
 WantedBy=multi-user.target

 # systemctl enable nova-chatproxy-genesys-pureengage.service
 # chown botconnector.botconnector /etc/systemd/system/nova-chatproxy-
genesys-pureengage.service
 # chown botconnector.botconnector -R /opt/nnova-chatproxy-genesys-
pureengage/config/
 # systemctl start nova-chatproxy-genesys-pureengage.service
```

The last option `Environment` in `[Service]` section means what config service have to use from config files.

- *Administration*

Connectivity checking

First of all, check that the needed ChatProxy service started and successfully run.

Check bound Port is LISTEN.

Check ChatProxy log and systemd unit log (`journalctl -xe -u nova-chatproxy-genesys-pureengage.service`)

If the log doesn't give clear/have detailed information - try to change Log Level, according to using Logger Specs (log4js).

UUIDs that help to track messaging.

Entity	Meaning
`ConnectorID`	First UUID in one log string, ID of Connector Instance
`conversation`	ID of conversation on kept trough all logs
`source`	type of channel

Some commands

ChatpRouter manages by `systemd`. Usefull commands are:

Operation	Command
check status	`systemctl status nova-chatproxy-genesys-purecloud.service`
restart	`systemctl restart nova-chatproxy-genesys-pureengage.service`
restart	`systemctl stop nova-chatproxy-genesys-pureengage.service` `systemctl start nova-chatproxy-genesys-pureengage.service`
view service logs	`journalctl -xe -u nova-chatproxy-genesys-pureengage.service`
check net use	`netstat -anp | grep LISTEN`

- *Maintenance and Support*

Updating. Installing new release

1. Stop service
2. Copy current release with all files into another place:

   ```
   # cp /opt/nova-chatproxy-genesys-pureengage/ $HOME/backup
   ```

3. Install update from by RPM

   ```
   # rpm -Uvh nova-chatproxy-genesys-pureengage-2020_R3-1.x86_64.rpm
   ```

4. Compare new configs with old one, add a new setting, if they exist and copy modified configs into folder with new installation.
5. Start service and check connectivity

Nova.ChatProxy.Genesys.PureCloud

- *Introduction*

Nova.ChatProxy.Genesys.PureCloud allows you to connect Nova.BotConnector with Genesys PureCloud WebChat.

- *Installation*

1. Install Component

```
# rpm -ivh $HOME/nova-chatproxy-genesys-purecloud-2020_R1-1.x86_64.rpm
```

2. Configure your Nova.ChatProxy

```
# cd /opt/nova-chatproxy-genesys-purecloud/config/
# cp chatproxy.js nova-chatproxy-genesys-purecloud.js
```

Component's Settings:

Section	Parameter	Value	Explanation
`db`	`uri`	*URL*	connection to Mongo. Replica-set Supported
`db`	`options`	*JSON*	default
`genesys`	`endpoint`	*URL*	*Australia/New Zealand*, *EU Ireland*, *EU Frankfurt*, *Japan*
`genesys`	`welcomeMessage` `waitForAgentMessage` `chatEndMessage` `unsupportedContentMessage`	*boolean*	enables messages from internal settings `/opt/nova-chatproxy-genesys-purecloud/messages.json`; filename must be like Enviroment variable in SYSTEMD
`genesys`	`purecloud`	*JSON*	see the example below
`genesys`	`memberInfo`	*JSON*	patterns, that can be send to customer side
`app`	`host`	*string*	own host
`app`	`port`	*int*	own port
`app`	`endQueuedSessions`	*int*	how quickly QUEUED session have will be deleted, in *min*
`app`	`endDeadSessions`	*int*	how quickly DEAD session have will be deleted in *min*
`app`	`downloadFolder`	*string*	local directory where media will be stored
`app`	`downloadURLPrefix`	*string*	prefix for formation full URL of stored media
`app`	`downloadMaxSize`	*int*	max size of download media. in *Bytes*
`app`	`connector.url`	*URL*	Internal BotConnector's URL

Section	Parameter	Value	Explanation
`app`	`connector.connectorId`	*UUID*	The ID of Botconnector Instance what have to proccess messages from ChatProxy

```
module.exports = {
db: {
    uri: 'mongodb://127.0.0.1:27017/nova-chatproxy-genesys-purecloud',
    debugMode: false,
    options: {
        useNewUrlParser: true,
        useFindAndModify: false,
        reconnectTries: 999999,
        reconnectInterval: 5000,
        poolSize: 10,
        connectTimeoutMS: 5000,
        family: 4,
        keepAlive: true,
        keepAliveInitialDelay: 300000,
    }
},
genesys: {
    endpoint: 'https://api.mypurecloud.ie',
    timeout: 50000,
    welcomeMessage: false,
    waitForAgentMessage: true,
    chatEndMessage: false,
    unsupportedContentMessage: true,
    defaultDisplayName: 'Nova.ChatsConnector',
    purecloud: {
        organizationId: '7d3b1626-0152-4568-98e4-900512803047',
        deploymentId: '329215a2-b571-4073-8489-baf44824795d',
        routingTarget: {
            targetType: 'queue',
            targetAddress: 'ChatQueue',
        },
    memberInfo: { // can be customized
        displayName: "{firstname} {lastname}",
        lastName: "{firstname}",
        firstName: "{lastname}",
        customFields: {
            origin: '{source}',
            userid: '{userid}',
        }

        },
    },
},
app: {
        host: '127.0.0.1',
        port: 5000,
        endQueuedSessions: 1,
        endDeadSessions: 1440,
        downloadFolder: '/var/lib/nginx/downloads/tmp',
        downloadURLPrefix: 'https://yourservername.com/tmp',
        downloadMaxSize: 52428800 // 50 MB
    },
    connector: {
        url: 'http://127.0.0.1:8080/connectors/',
        connectorId: 'c019973d-f8e9-4c9c-b8ed-a688115b48f0',
        timeout: 5000
    },
    logging: { // log4js-node configuration
        appenders: {
            file: {
                type: 'file',
                filename: '/var/log/nova-chatproxy-genesys-
purecloud/nova/nova-chatproxy.log',
                maxLogSize: 52428800, // 50 MB
                backups: 5,
                compress: true,
                keepFileExt: true
            }
        },
        categories: {
            default: {appenders: ['file'], level: 'ALL'}
        }
    }
```

3. Modify and enable service

```
# cd /etc/systemd/system/
# vim /etc/systemd/system/nova-chatproxy-genesys-purecloud.service

[Unit]
Description=Genesys PureCloud proxy for Nova.BotConnector
After=network.target mongodb.service

[Service]
ExecStart=/usr/bin/node /opt/nova-chatproxy-genesys-purecloud/src/index.js
WorkingDirectory=/opt/nova-chatproxy-genesys-purecloud
Restart=on-failure
RestartSec=30
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=nova-chatproxy-genesys-purecloud
User=botconnector
Group=botconnector
```

```
 Environment=CHATPROXY_INSTANCE=nova-chatproxy-genesys-purecloud

 [Install]
 WantedBy=multi-user.target

 # systemctl enable nnova-chatproxy-genesys-purecloud.service
 # chown botconnector.botconnector /etc/systemd/system/nova-chatproxy-
genesys-purecloud.service
 # chown botconnector.botconnector -R /opt/nnova-chatproxy-genesys-
purecloud/config/
 # systemctl start nova-chatproxy-genesys-purecloud.service
```

The last option `Environment` in `[Service]` section means what config service have to use from config files.

- *Administration*

Connectivity checking

First of all, check that the needed ChatProxy service started and successfully run.

Check bound Port is LISTEN.

Check ChatProxy log and systemd unit log (`journalctl -xe -u nova-chatproxy-genesys-purecloud.service`)

If the log doesn't give clear/have detailed information - try to change Log Level, according to using Logger Specs (log4js).

UUIDs that help to track messaging.

Entity	Meaning
`ConnectorID`	First UUID in one log string, ID of Connector Instance
`conversation`	ID of conversation on kept trough all logs
`source`	type of channel

Some commands

ChatpRouter manages by `systemd`. Usefull commands are:

Operation	Command
check status	`systemctl status nova-chatproxy-genesys-purecloud.service`
restart	`systemctl restart nova-chatproxy-genesys-purecloud.service`
restart	`systemctl stop nova-chatproxy-genesys-purecloud.service` `systemctl start nova-chatproxy-genesys-purecloud.service`

Operation	Command
view service logs	`journalctl -xe -u nova-chatproxy-genesys-purecloud.service`
check net use	`netstat -anp | grep LISTEN`

- *Maintenance and Support*

Updating. Installing new release

1. Stop service
2. Copy current release with all files into another place:

```
# cp /opt/nova-chatproxy-genesys-purecloud/ $HOME/backup
```

3. Install update from by RPM

```
# rpm -Uvh nova-chatproxy-genesys-purecloud-2020_R1-1.x86_64.rpm
```

4. Compare new configs with old one, add a new setting, if they exist and copy modified configs into folder with new installation.
5. Start service and check connectivity

Nova.ChatProxy.Cisco.ECE

- *Introduction*

Nova.ChatProxy.Cisco.ECE allows you to connect Nova.BotConnector with Cisco Enterprise Chat and Email (ECE).

- *Installation*

1. Install Component

```
# rpm -ivh $HOME/nova-chatproxy-cisco-ece-2020_R1-1.x86_64.rpm
```

2. Configure your Nova.ChatProxy

```
# cd /opt/nova-chatproxy-cisco-ece/config/
# cp chatproxy.js nova-chatproxy-cisco-ece.js
```

Component's Settings:

Section	Parameter	Value	Explanation
`db`	`uri`	*URL*	connection to Mongo. Replica-set Supported
`db`	`options`	*JSON*	default
`egain`	`corsHost`	*URL*	FQDN for chat
`chat`	`entryPointId`	*int*	Entry Point ID from Cisco settings
`chat`	`locale`	*string*	settings from Cisco Chat settings
`chat`	`templateName`	*string*	settings from Cisco Chat settings
`chat`	`firstName` `lastName`	*string*	from Who you will be see chat-session
`chat`	`systemMessages`	*boolean*	enable or disable
`chat`	`welcomeMessage` `waitForAgentMessage` `chatEndMessage`	*string*	enables messages from internal settings `/opt/nova-chatproxy-cisco-ece/resources/messages.json`; filename must be like Enviroment variable in SYSTEMD.
`app`	`host`	*string*	own host
`app`	`port`	*int*	own port
`app`	`cleanChatOlder`	*int*	
`connector`	`url`	*URL*	Nova.BotConnector Internal URL
`connector`	`connectorId`	*UUID*	Nova.BotConnector ID in MongoDB, where we will be connect channels

Tune main config:

```
# vim /opt/nova-chatproxy-cisco-ece/config/nova-chatproxy-cisco-ece.js

module.exports = {
db: {
    uri: 'mongodb://127.0.0.1:27017/nova-chatproxy-cisco-ece,
    debugMode: false,
    options: {
        useNewUrlParser: true,
        poolSize: 10,
        connectTimeoutMS: 5000,
        useUnifiedTopology: true,
        useFindAndModify: false,
        family: 4,
        keepAlive: true,
        keepAliveInitialDelay: 300000
    }
},
egain: {
    corsHost: 'https://chat.com/system',
    isDevelopmentModeOn: false,
    eGainContextPath: './',
    isDebugOn: false,
    chatPauseInSec: 30
},
chat: {
    entryPointId: 1004,
    locale: 'ru_RU',
    templateName: 'aqua-al-ukc',
    firstName: 'Nova',
    lastName: 'ChatProxy',
    systemMessages: true,
    welcomeMessage: false,
    waitForAgentMessage: false,
    chatEndMessage: false
},
app: {
    host: '127.0.0.1',
    port: 8888,
    cleanChatOlder: 24
},
connector: {
    url: 'http://127.0.0.1:8080/connectors/',
    connectorId: '7d56d98b-b28b-4b22-a875-5d9b434fc2a7',
    timeout: 5000
},
logging: { // log4js-node configuration
    appenders: {
        file: {
            type: 'file',
            filename: '/var/log/nova-chatproxy-cisco-ece/chatproxy.log',
            maxLogSize: 52428800, // 50 MB
            backups: 5,
            compress: true,
            keepFileExt: true
        }
    },
    categories: {
        default: {appenders: ['file'], level: 'debug'}
    }
}
}
```

3. Modify and enable service

```
 # cd /etc/systemd/system/multi-user.target.wants
 # cp /etc/systemd/system/multi-user.target.wants/nova-chatproxy-cisco-
ece.service ../
 # vim /etc/systemd/system/nova-chatproxy-cisco-ece.service

 [Unit]
 Description=Cisco ECE proxy for Nova.BotConnector. UKC_Help.
 After=network.target mongodb.service

 [Service]
 ExecStart=/usr/bin/node /opt/nova-chatproxy-cisco-ece/src/index.js
 WorkingDirectory=/opt/nova-chatproxy-cisco-ece
 Restart=on-failure
 RestartSec=30
 StandardOutput=syslog
 StandardError=syslog
 SyslogIdentifier=nova-chatproxy-cisco-ece
 User=botconnector
 Group=botconnector
 Environment=CHATPROXY_INSTANCE=nova-chatproxy-cisco-ece

 [Install]
 WantedBy=multi-user.target

 # systemctl enable nova-chatproxy-cisco-ece.service
 # chown botconnector.botconnector /etc/systemd/system/nova-chatproxy-cisco-
ece.service
 # chown botconnector.botconnector -R /opt/nova-chatproxy-cisco-ece/config/
 # systemctl start nova-chatproxy-cisco-ece.service
```

The last option `Environment` in `[Service]` section means what config service have to use from config files.

- *Administration*

Connectivity checking

First of all, check that the needed ChatProxy service started and successfully run.

Check bound Port is LISTEN.

Check ChatProxy log and systemd unit log (`journalctl -xe -u nova-chatproxy-cisco-ece.service`)

If the log doesn't give clear/have detailed information - try to change Log Level, according to using Logger Specs (log4js).

UUIDs that help to track messaging.

Entity	Meaning
`ConnectorID`	First UUID in one log string, ID of Connector Instance
`conversation`	ID of conversation on kept trough all logs
`source`	type of channel

Some commands

Chatproxy manages by `systemd`. Usefull commands are:

Operation	Command
check status	`systemctl status nova-chatproxy-cisco-ece.service`
restart	`systemctl restart nova-chatproxy-cisco-ece.service`
restart	`systemctl stop nova-chatproxy-cisco-ece.service` `systemctl start nova-chatproxy-cisco-ece.service`
view service logs	`journalctl -xe -u nova-chatproxy-cisco-ece.service`
check net use	`netstat -anp | grep LISTEN`

- *Maintenance and Support*

Updating. Installing new release

1. Stop service
2. Copy current release with all files into another place:

   ```
   # cp /opt/nova-chatproxy-cisco-ece/ $HOME/backup
   ```

3. Install update from by RPM

   ```
   # rpm -Uvh nova-chatproxy-cisco-ece-2020_R1-1.x86_64.rpm
   ```

4. Compare new configs with old one, add a new setting, if they exist and copy modified configs into folder with new installation.
5. Start service and check connectivity

Nova.ChatRouter.Omilia

- *Introduction*

Nova.ChatRouter allows you to connect Nova.BotConnector with Omilia Solution or Googele Dialog Flow solution.

- *Installation*

1. Install Component

```
# rpm -ivh $HOME/nova-chatrouter-2020_R1-1.x86_64.rpm
```

2. Configure ChatRouter

```
# cd /opt/nova-chatrouter/config/
# cp chatrouter.js nova-chatrouter-omilia.js
```

Component's Settings:

Section	Parameter	Value	Explanation
`db`	`uri`	*URL*	connection to Mongo. Replica-set Supported
`db`	`options`	*JSON*	default
`app`	`host`	*string*	own host
`app`	`port`	*int*	own port
`app`	`endDeadSessions`	*int*	how frequently need to delete ended session. Recomenend value 24 hours
`app`	`endBasedOnChatProxySessions`	*boolean*	Chatrouter must end session bassing on session state of ChatProxy or not
`omilia`	`enabled`	*boolean*	enabled or disabled
`omilia`	`url`	*URL*	URL to Omilia host
`omilia`	`application`	*string*	Tne name of Omilia's Application handles chat
`omilia`	`sessionRefreshTimeout`	*int*	in miliseconds, how long will be *NoInput* timeout.
`omilia`	`attachHistory`	*boolean*	send Omilia's dialog history

Section	Parameter	Value	Explanation
`omilia`	`surveyEnabled`	*boolean*	enable Application *after* ChatProxy Session
`omilia`	`surveyApplicatonId`	*string*	The name of Omilia's Application handles chat *after* ChatProxy Session (i.e. questionnaire)
`omilia`	`drtviewer`	*JSON*	connection parameters for *omilia.attachHistory*. Required only if *attachHistory* is enabled.
`omilia`	`sendGreetingMessage`	*boolean*	first message from Omilia after session creation
`omilia`	`ignoreFailure`	*boolean*	session will be transferred to ChatProxy if Omilia fails
`dialogflow`	`enabled`	*boolean*	enabled or disabled
`dialogflow`	`projectId`	*string*	The project ID from the Google Developer's Console
`dialogflow`	`languageCode`	*string*	The language to retrieve training phrases, parameters and rich messages for
`dialogflow`	`endIntent`	*string*	The Intent indicates about ending Chat Session
`dialogflow`	`transferIntent`	*string*	The Intent indicates about starting ChatProxy Session
`dialogflow`	`ignoreFailure`	*boolean*	
`chatproxy`	`url`	*URL*	Internal ChatProxy's URL
`botconnector`	`url`	*URL*	Internal BotConnector's URL
`botconnector`	`connectorId`	*UUID*	The ID of Botconnector Instance what have to proccess messages from ChatRouter

Tune main config:

```
# vim /opt/nova-chatrouter/config/nova-chatrouter-omilia.js


module.exports = {
db: {
uri: 'mongodb://127.0.0.1:27017/nova-chatrouter-omilia',
debugMode: false,
options: {
    useNewUrlParser: true,
    useFindAndModify: false,
    useUnifiedTopology: true,
    reconnectTries: 999999,
    reconnectInterval: 5000,
    poolSize: 10,
    connectTimeoutMS: 5000,
    family: 4,
    keepAlive: true,
    keepAliveInitialDelay: 300000,
    bufferMaxEntries: 0
}
},
app: {
    host: '127.0.0.1',
    port: 6789,
    endDeadSessions: 1440,
    endBasedOnChatProxySessions: false
},
omilia: {
    enabled: false,
    url: 'https://192.168.1.1:8443/',
    applicationId: 'TEST_APP',
    sessionRefreshTimeout: 30000,
    attachHistory: true,
    surveyEnabled: true,
    surveyApplicatonId: 'Demo_Feedback',
    drtviewer: { // r
        url: 'https://192.168.1.1',
        username: 'drtviewer',
        password: 'drtviewer',
        clientSecret: '28ac0167-22c9-422d-acea-00cdf1bbcc16'
    },
    sendGreetingMessage: true,
    ignoreFailure: true,
    timeout: 5000
},
dialogflow: {
    enabled: true,
    projectId: 'chatrouterdemoapp-nxnlhq',
    languageCode: 'en-EN',
    endIntent: 'No_Help_Not_Needed',
    transferIntent: 'Operator',
    ignoreFailure: true,
    timeout: 10000
},
chatproxy: {
    url: 'http://127.0.0.1:5000/',
    timeout: 5000
},
botconnector: {
    url: 'http://127.0.0.1:8080/connectors/',
    connectorId: '3be1660b-0b70-49b4-964a-bc6667fd6f4f',
    timeout: 5000
},
logging: {
    appenders: {
        file: {
            type: 'file',
            filename: '/var/log/nova-chatrouter/chatrouter.log',
            maxLogSize: 52428800,
            backups: 5,
            compress: true,
            keepFileExt: true
        }
    },
    categories: {
        default: {appenders: ['file'], level: 'debug'}
    }
}
}
```

3. Modify and enable service

```
 # cd /etc/systemd/system/
 # vim /etc/systemd/system/nova-chatrouter-omilia.service

 [Unit]
 Description=Chat routing application.

 [Service]
 ExecStart=/usr/bin/node /opt/nova-chatrouter/src/index.js
 WorkingDirectory=/opt/nova-chatrouter
 Restart=on-failure
 RestartSec=30
 StandardOutput=syslog
 StandardError=syslog
 SyslogIdentifier=nova-chatrouter-omilia
 User=botconnector
 Group=botconnector
 Environment=CHATROUTER_INSTANCE=nova-chatrouter-omilia

 [Install]
 WantedBy=multi-user.target

 # systemctl enable nova-chatrouter-omilia.service
 # chown botconnector.botconnector /etc/systemd/system/nova-chatrouter-
omilia.service
 # chown botconnector.botconnector -R /etc/systemd/system/nova-
```

```
chatrouter/config/
 # systemctl start nova-chatrouter-omilia.service
```

The last option `Environment` in `[Service]` section means what config service have to use from config files.

- *Administration*

Connectivity checking

First of all, check that the needed ChatRouter service started and successfully run.

Check bound Port is LISTEN.

Check ChatProxy log and systemd unit log (`journalctl -xe -u nova-cahtrouter.service`)

If the log doesn't give clear/have detailed information - try to change Log Level, according to using Logger Specs (log4js).

UUIDs that help to track messaging.

Entity	Meaning
`ConnectorID`	First UUID in one log string, ID of Connector Instance
`conversation`	ID of conversation on kept trough all logs
`source`	type of channel

Some commands

ChatpRouter manages by `systemd`. Usefull commands are:

Operation	Command
check status	`systemctl status nova-chatrouter.service`
restart	`systemctl restart nova-chatrouter.service`
restart	`systemctl stop nova-chatrouter.service` `systemctl start nova-chatrouter.service`
view service logs	`journalctl -xe -u nova-chatrouter.service`
check net use	`netstat -anp | grep LISTEN`

- *Maintenance and Support*

Updating. Installing new release

1. Stop service

2. Copy current release with all files into another place:

```
    # cp /opt/nova-chatrouter/ $HOME/backup
```

3. Install update from by RPM

```
    # rpm -Uvh nova-chatrouter-2020_R1-1.x86_64.rpm
```

4. Compare new configs with old one, add a new setting, if they exist and copy modified configs into folder with new installation.

5. Start service and check connectivity