

WAMP

Concepts

A Crossbar.io **node** is a single instance of the Crossbar.io software running in a specific **node directory**

There are two types of processes running within a Crossbar.io node:

- one process **controller**
- multiple **workers**

Workers:

- Native Workers
 - Routers: WAMP routing services between WAMP clients
 - Containers: application components written in Python
- Guest Workers: arbitrary programs spawned

WAMP router

Use python **only** 2.x

prereq

```
apt-get install python-virtualenv python-dev libffi-dev \
libssl-dev libxml2-dev libxslt1-dev libyaml-dev
```

```
pip install crossbar[tls,msgpack,manhole,system]
```

ATTENZIONE con la versione 0.11.1, commentare la riga 232 di lib/lib/python2.7/site-packages/crossbar/controller/process.py

```
#ep['meta'] = e(None)
```

Create new router node

```
cd /path/
crossbar init --template default
```

.crossbar/config.json

```
{
  "controller": {
  },
  "workers": [
```

```
{
  "type": "router",
  "options": {
    "pythonpath": [".."]
  },
  "manhole": {
    "endpoint": {
      "type": "tcp",
      "port": 6022
    },
    "users": [
      {
        "user": "scipio",
        "password": "z"
      }
    ]
  },
  "realms": [
    {
      "name": "realm1",
      "roles": [
        {
          "name": "anonymous",
          "permissions": [
            {
              "uri": "*",
              "publish": true,
              "subscribe": true,
              "call": true,
              "register": true
            }
          ]
        }
      ]
    }
  ],
  "transports": [
    {
      "type": "web",
      "endpoint": {
        "type": "tcp",
        "port": 8080
      },
      "paths": {
        "ws": {
          "type": "websocket",
          "debug": false
        }
      }
    }
  ]
}
```

```
}  
]  
}
```

Start node

```
crossbar start
```

supervisor config

```
[program:wamp]  
command=/opt/wamp/lib/bin/crossbar start --cbdir /opt/wamp/.crossbar  
user=wamp  
autostart=true
```

http bridge publisher and callee

An http POST request can emit WAMP event, or we can call an RPC via http POST.

```
"transports": [  
  {  
    "type": "web",  
    "endpoint": {  
      "type": "tcp",  
      "port": 8080  
    },  
    "paths": {  
      ...  
      "push": {  
        "type": "publisher",  
        "realm": "realm1",  
        "role": "anonymous"  
      },  
      "call": {  
        "type": "caller",  
        "realm": "realm1",  
        "role": "anonymous"  
      }  
    }  
  }  
]
```

example of event publish

```
import requests  
requests.post("http://router_ip/push",  
             json={
```

```
        'topic': 'great_topic'
        'args': [some, params, to, pass, along, if, you, need,
to]
    })
```

example of event post with curl

```
curl --connect-timeout 3 \
-H "Content-Type: application/json" \
-d '{"topic": "vr.ztl.passaggi", "kwargs": {"uids": [1,2,3]}}' \
http://localhost:8081/push
```

example of RPC call with curl

```
curl --connect-timeout 3 \
-H "Content-Type: application/json" \
-d '{"procedure": "vr.ztl.permessi.get", "args": ["EC316MP"]}' \
http://127.0.0.1:8081/call
```

WSGI app (pyramid)

Create wsgi.py in same folder as setup.py

```
ini_path = '../development.ini'

from pyramid.paster import get_app, setup_logging
setup_logging(ini_path)
application = get_app(ini_path, 'main')
```

Add to “paths”

```
"/": {
    "type": "wsgi",
    "module": "wsgi",
    "object": "application"
},
```

WAMP node

Use python 3.x

```
pip install autobahn[asyncio,twisted]
```

From:

<https://wiki.csgalileo.org/> - **Galileo Labs**

Permanent link:

<https://wiki.csgalileo.org/projects/wamp/docs?rev=1443183030>

Last update: **2015/09/25 14:10**

