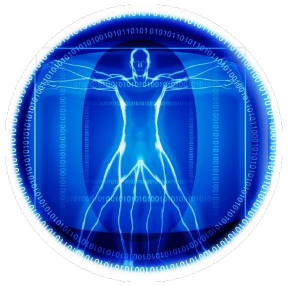
# CYBOP UDP-Kommunikation

Mirco Gatz • Patrick Westphal



## Gliederung

- Idee: UDP-Echo-Server
- CYBOL
  - `service`
    - `namespace`
    - `address`
  - `send`
    - `channel`
    - `language`
    - `receiver`
- Beispiel
  - Sender
  - Empfänger

## UDP-Echo-Server

- Idee:
  - CYBOI 1 wartet in Endlosschleife auf eingehende Daten

    → UDP/IP-Kommunikation
  - CYBOI 1 sendet empfangene Daten 1:1 wieder an den Sender zurück

  - CYBOI 2 sendet an CYBOI 1via Netzwerk bspw. „Hello CYBOI“

## UDP-Echo-Server

→ Szenario in CYBOL abbilden und eventuell auftretende Probleme/Fehler dokumentieren

## CYBOL > service

### Startup

This operation starts up the given service.

### Example

```
<part name="startup_wui" channel="inline" abstraction="operation" model="startup">
    <property name="service" channel="inline" abstraction="character" model="www"/>
    <property name="namespace" channel="inline" abstraction="character" model="inet"/>
    <property name="style" channel="inline" abstraction="character" model="stream"/>
    <property name="address" channel="inline" abstraction="character" model="any"/>
</part>
```

## CYBOL > service

### Startup

This operation starts up the given service.

### Example

```
<part name="startup_wui" channel="inline" abstraction="operation" model="startup">
    <property name="service" channel="inline" abstraction="character" model="www"/>
    <property name="namespace" channel="inline" abstraction="character" model="inet"/>
    <property name="style" channel="inline" abstraction="character" model="stream"/>
    <property name="address" channel="inline" abstraction="character" model="any"/>
</part>
```

# CYBOL > service

## Service Property

This is the service to be started up.

*required*

```
name   =  'service'
abstraction  =  'character'
model        =  'signal' | 'shell' | 'standard output' |
                'gnu_linux_console' | 'x_window_system' |
                'www' | 'cyboi'
```

## CYBOL > service

### Service Property

This is the service to be started up.

*required*

```
name   =  'service'
abstraction  =  'character'
model        =  'signal' | 'shell' | 'standard output' |
                'gnu_linux_console' | 'x_window_system' |
                'www' | 'cyboi' | 'network'
```

## CYBOL > service

### Startup

This operation starts up the given service.

### Example

```
<part name="startup_wui" channel="inline" abstraction="operation" model="startup">
    <property name="service" channel="inline" abstraction="character" model="www"/>
    <property name="namespace" channel="inline" abstraction="character" model="inet"/>
    <property name="style" channel="inline" abstraction="character" model="stream"/>
    <property name="address" channel="inline" abstraction="character" model="any"/>
</part>
```

CYBOL > `service`

## Namespace Property

The namespace of the socket.

*optional, only if service is www or cyboi*

```
name         =   'namespace'
abstraction  =   'character'
model        =   'local' | 'inet' | 'inet6' | 'ns' | 'iso' |
                 'ccitt' | 'implink' | 'route'
```

# CYBOL > service

## Namespace Property

The namespace of the socket.

*optional, only if service is www or cyboi*

name = 'namespace'

abstraction = 'character'

'ipv4' 'ipv6' (?)

model = 'local' | ~~'inet'~~ | ~~'inet6'~~ | 'ns' | 'iso' | 'ccitt' | 'implink' | 'route'

## CYBOL > service

### Startup

This operation starts up the given service.

### Example

```
<part name="startup_wui" channel="inline" abstraction="operation" model="startup">
    <property name="service" channel="inline" abstraction="character" model="www"/>
    <property name="namespace" channel="inline" abstraction="character" model="inet"/>
    <property name="style" channel="inline" abstraction="character" model="stream"/>
    <property name="address" channel="inline" abstraction="character" model="any"/>
</part>
```

# CYBOL > service

## Address Property

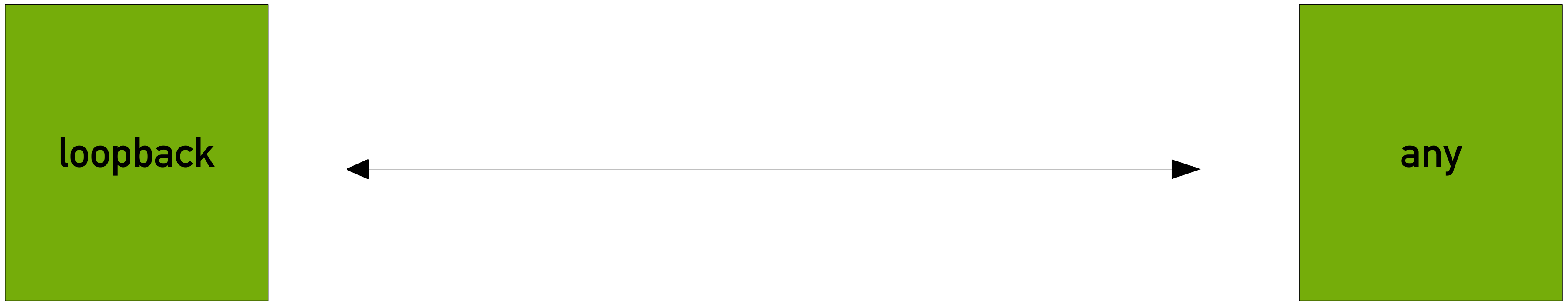
This is the address of hosts communicating with this system via socket.

*optional, only if service is www or cyboi*

```
name          =  'address'
abstraction   =  'character'
model         =  'loopback' | 'any'
```

# CYBOL > service

## Address Property



## CYBOL > send

### Send

This operation is able to send a message via textual, graphical or web user interface, or to the file system or also as shell output directly.

### Example

```
<part name="send_menu" channel="inline" abstraction="operation" model="send">
    <property name="channel" channel="inline" abstraction="character"
              model="gnu_linux_console"/>
    <property name="language" channel="inline" abstraction="character" model="tui"/>
    <property name="message" channel="inline" abstraction="knowledge" model=".app.tui"/>
    <property name="area" channel="inline" abstraction="knowledge" model=".app.tui.menu"/>
    <property name="clean" channel="inline" abstraction="boolean" model="true"/>
</part>
```

## CYBOL > send

Send

This operation is able to send a message via textual, graphical or web user interface, or to the file system or also as shell output directly.

Example

```
<part name="send_menu" channel="inline" abstraction="operation" model="send">
    <property name="channel" channel="inline" abstraction="character"
              model="gnu_linux_console"/>
    <property name="language" channel="inline" abstraction="character" model="tui"/>
    <property name="message" channel="inline" abstraction="knowledge" model=".app.tui"/>
    <property name="area" channel="inline" abstraction="knowledge" model=".app.tui.menu"/>
    <property name="clean" channel="inline" abstraction="boolean" model="true"/>
</part>
```

## CYBOL > send

### Channel Property

The channel via which to send the message.

required

```
name          =   'channel'
abstraction   =   'character'
model         =   'inline' | 'file' | 'standard_output' |
                  'gnu_linux_console' | 'x_window_system' |
                  'http'
```

## CYBOL > send

**Channel Property**

The channel via which to send the message.

required

```
name          =   'channel'
abstraction   =   'character'
model         =   'inline' | 'file' | 'standard_output' |
                  'gnu_linux_console' | 'x_window_system' |
                  'http' | 'cyboi' | 'network'
```

## CYBOL > send

**Send**

This operation is able to send a message via textual, graphical or web user interface, or to the file system or also as shell output directly.

**Example**

```
<part name="send_menu" channel="inline" abstraction="operation" model="send">
    <property name="channel" channel="inline" abstraction="character"
              model="gnu_linux_console"/>
    <property name="language" channel="inline" abstraction="character" model="tui"/>
    <property name="message" channel="inline" abstraction="knowledge" model=".app.tui"/>
    <property name="area" channel="inline" abstraction="knowledge" model=".app.tui.menu"/>
    <property name="clean" channel="inline" abstraction="boolean" model="true"/>
</part>
```

## CYBOL > send

### Language Property

The language into which to encode the message before sending it.

*required*

```
name          =  'language'
abstraction   =  'character'
model         =  'tui' | 'gui' | 'wui'
```

## CYBOL > send

### Language Property

```
trunk/examples$ grep -R "language" * | grep -v svn
addition/run.cybol: <property name="language" channel="inline" abstraction="text/plain" model="integer"/>
[...]
addition_static_model/run.cybol: <property name="language" channel="inline" abstraction="text/plain"
model="text/cybol"/>
[...]
counter/run.cybol: <property name="language" channel="inline" abstraction="text/plain" model="text/model-
diagram"/>
[...]
helloworld/startup.cybol: <property name="language" channel="inline" abstraction="text/plain"
model="text/plain"/>
http_communication/logic/send_wui_index.cybol: <property name="language" channel="inline"
abstraction="text/plain" model="text/html"/>
[...]
http_communication/logic/handler/handle_www_service.cybol: <property name="language" channel="inline"
abstraction="text/plain" model="message/http-request"/>
```

## CYBOL > send

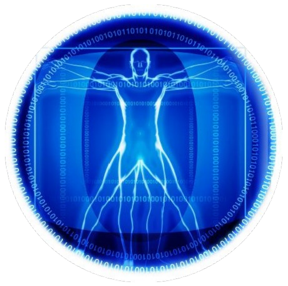
### Receiver Property

The name of the system receiving the message.

*required*

```
name          =   'receiver'
abstraction   =   'character'
model         =   name of receiving system
```

?



## Beispiel > service

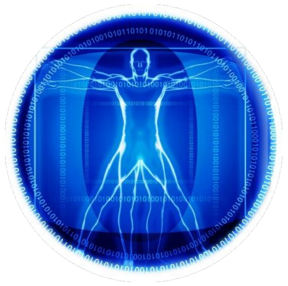
```
<part name="startup_udp" channel="inline" abstraction="operation/plain" model="startup">
    <property name="service" channel="inline" abstraction="text/plain" model="cyboi"/>
    <property name="namespace" channel="inline" abstraction="text/plain" model="ipv4"/>
    <property name="style" channel="inline" abstraction="text/plain" model="datagram"/>
    <property name="address" channel="inline" abstraction="text/plain" model="any"/>
</part>
```

## Beispiel > service

```
root@chaos:examples# nmap -sU -p 1970-1975 localhost

Starting Nmap 5.21 ( http://nmap.org ) at 2011-04-26 17:00 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000025s latency).
PORT     STATE         SERVICE
1970/udp closed        unknown
1971/udp open|filtered unknown
1972/udp closed        unknown
1973/udp closed        unknown
1974/udp closed        unknown
1975/udp closed        unknown

Nmap done: 1 IP address (1 host up) scanned in 1.35 seconds
```



## Beispiel > receive

```
<part name="read" channel="inline" abstraction="operation/plain" model="receive">

    <property name="channel" channel="inline" abstraction="text/plain" model="cyboi"/>

    <property name="language" channel="inline" abstraction="text/plain"
            model="text/plain"/>

    <property name="model" channel="inline" abstraction="path/knowledge"
            model=".udp_communication.msg"/>

    <property name="mode" channel="inline" abstraction="text/plain" model="server">

    <property name="style" channel="inline" abstraction="text/plain" model="datagram"/>

</part>
```

## Beispiel > `receive`

```
...
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
TEST ps: 0
...
```

kein Empfang

## Beispiel > send

```
<part name="write" channel="inline" abstraction="operation/plain" model="send">

    <property name="channel" channel="inline" abstraction="text/plain" model="cyboi"/>

    <property name="language" channel="inline" abstraction="text/plain"
             model="text/plain"/>

    <property name="namespace" channel="inline" abstraction="text/plain" model="ipv4"/>

    <property name="style" channel="inline" abstraction="text/plain" model="datagram"/>

    <property name="mode" channel="inline" abstraction="text/plain" model="client"/>

    <property name="message" channel="inline" abstraction="path/knowledge"
             model=".udp_communication.send_msg"/>
</part>
```

## Beispiel > send

```
examples$ ../bin/cyboi --knowledge udp_sender/run.cybol
TEST: startup socket bind s: 3
TEST: send socket sn: 2
TEST: send socket an: 2
TEST: send socket st: 2
Speicherzugriffsfehler
```