



XMPP

XmppTalk

Philippe Back
@philippeback



What is XMPP?

eXtensible Messaging and Presence Protocol



The most secure messaging standard

Battle-tested. Independent. Privacy-focused.

XMPP is the open standard for messaging and presence

XMPP powers emerging technologies like [IoT](#), [WebRTC](#), and [social](#).

It's a [living standard](#). Engineers actively extend and improve it.

No one owns XMPP. It's free and open for everyone [since 1999](#).

Millions use [XMPP software](#) daily to connect to people and services.

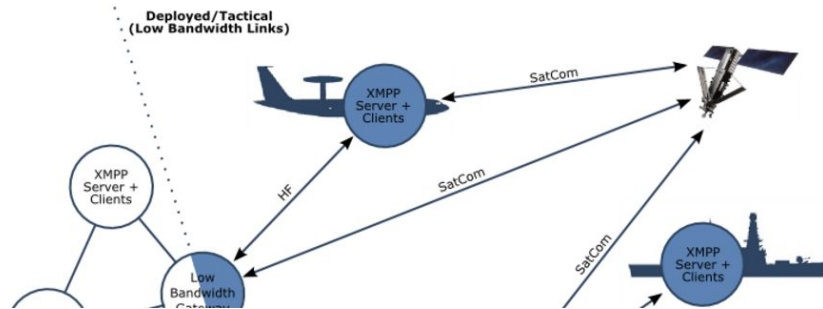
Uses of Xmpp



XMPP is the Internet Standard for Instant Messaging, Real Time Messaging, and Multi-User Chat. It provides high functionality and good security capabilities. XMPP is being increasingly adopted by NATO and National organizations, as the technology of choice for Instant Messaging.

Scenarios

Two sample scenarios are considered that show ways in which XMPP can be used, and requirements that need to be considered given the mix of communication methods likely to be available during a deployment.



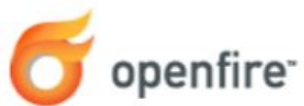
So, I wanted an XMPP client in Pharo for the Slack

Since then we moved to Discord and there is no XMPP there.

Anyway...

One needs a server

<http://files.pharo.org/media/logo/logo-flat.png>



Openfire 3.9.3
Logged in as admin - [Logout](#)

Server Users/Groups **Sessions** Group Chat Plugins

Active Sessions Tools

- ▶ Client Sessions
- Server Sessions
- Component Sessions

Client Sessions

Active Client Sessions: 2 -- Sessions per page: 15 ▼

Refresh: None ▼ (seconds)

	Name 	Resource	Node	Status	Presence	Priority	Client IP	Close Connection
1	pharo	PharoIDE	Local	Authenticated 	 Online	0	127.0.0.1	
2	phil	Gajim	Local	Authenticated 	 Online	50	127.0.0.1	

Unpack and run OpenFire (Needs some Java)

```
philippe libstrophe openfire* pharo | To release input, press Ctrl+Alt. -head> 18/05 11:54:15 ☂ 16.1°
conf lib plugins
philippeback@ubuntu:/opt/openfire$
philippeback@ubuntu:/opt/openfire$ ls
bin documentation LICENSE.html README.html
changelog.html embedded-db logs resources
conf lib plugins
philippeback@ubuntu:/opt/openfire$ cd bin
philippeback@ubuntu:/opt/openfire/bin$ ll
total 36
drwxr-xr-x 3 philippeback philippeback 4096 May 6 2014 ./
drwxr-xr-x 11 philippeback philippeback 4096 Jan 17 06:22 ../
drwxr-xr-x 3 philippeback philippeback 4096 May 6 2014 extra/
-rwxr-xr-x 1 philippeback philippeback 13388 May 6 2014 openfire*
-rwxr--r-- 1 philippeback philippeback 5034 May 6 2014 openfirectl*
philippeback@ubuntu:/opt/openfire/bin$ ./openfire start
Starting openfire
philippeback@ubuntu:/opt/openfire/bin$
```

One needs a non Pharo client

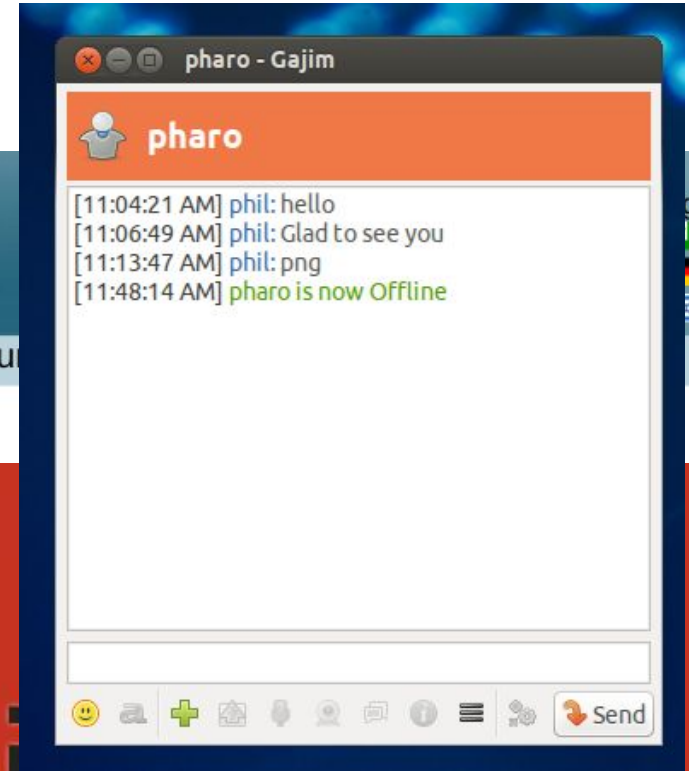


Bonjour et bienvenue sur le

Le but de Gajim est de fournir
cohabite très bien avec GNO
distribué sous la licence GNU
idées pour améliorer Gajim, e

FONCTIONNALITÉS :

- Modes de fenêtre de ch
- Support des groupes d
- transformation d'une di



Profanity

A console ba

One does not want to recode the low level

libstrophe

An XMPP library for C

libstrophe is a minimal XMPP library written in C. It has almost no external dependencies, only an XML parsing library (expat or libxml are both supported). It is designed for both POSIX and Windows systems.

Build 32 bit library for Pharo 32-bit

Use a container (LXC)

Can do it with Docker as well. No need for Docker with LXC.

One needs to bind to Pharo: UnifiedFFI UFFI Binding for Libstrophe

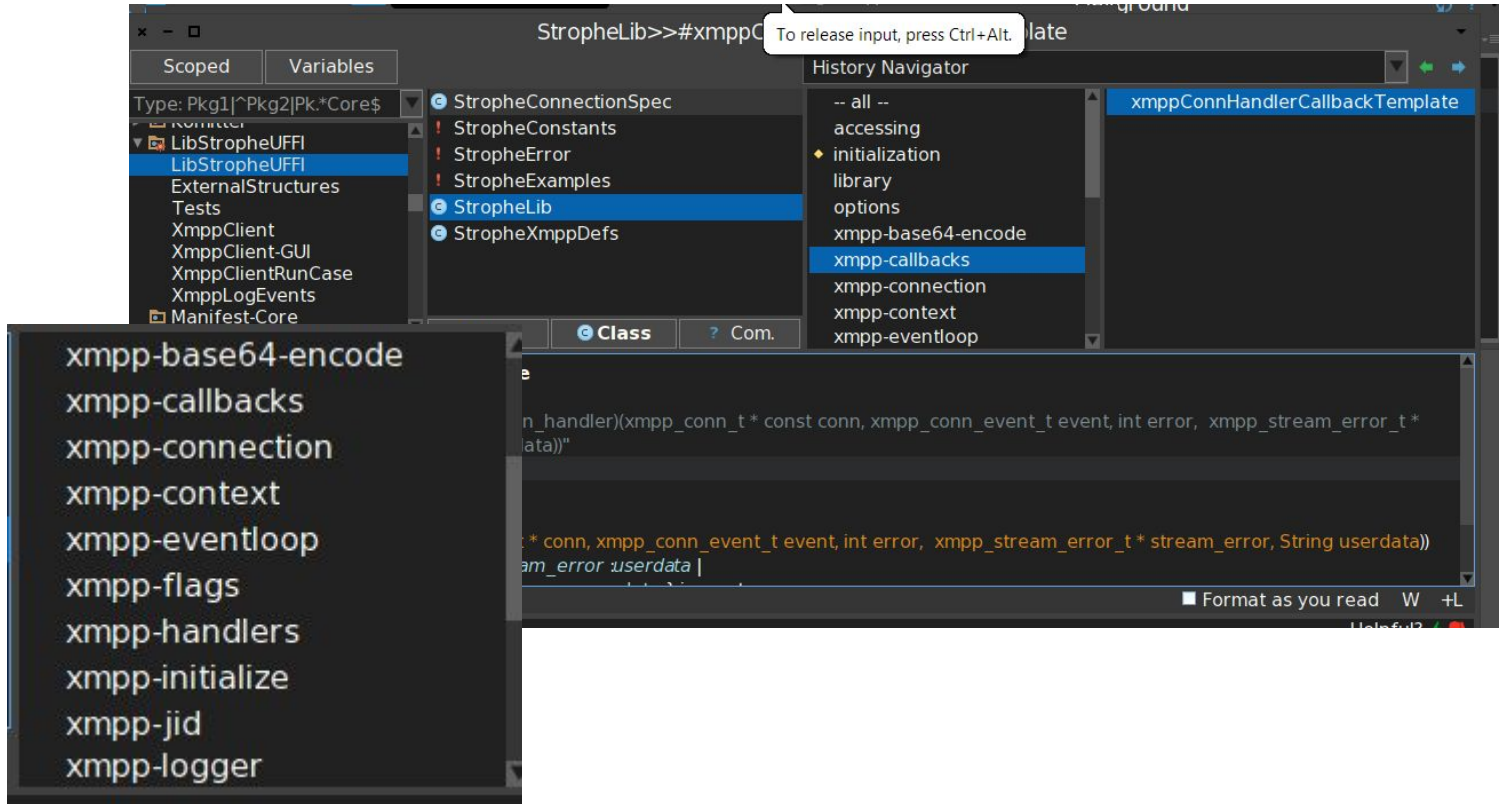
```
philippe <bstrophe openfire pharo bas... To release input, press Ctrl+Alt. leader* 18/05 11:54:37 16.1°
43 | xmpp_conn_type_t;
44 |
45 | typedef void (*xmpp_log_handler)(void * const userdata,
46 |                                 const xmpp_log_level_t level,
47 |                                 const char * const area,
48 |                                 const char * const msg);
49 |
50 | struct _xmpp_log_t {
51 |     xmpp_log_handler handler;
52 |     void *userdata;
53 | };
54 |
55 | /* return a default logger filtering at a given level */
56 | xmpp_log_t *xmpp_get_default_logger(xmpp_log_level_t level);
57 |
58 | /* connection */
59 |
60 | /* opaque connection object */
61 | typedef struct _xmpp_conn_t xmpp_conn_t;
62 | typedef struct _xmpp_stanza_t xmpp_stanza_t;
63 |
NORMAL > master > strophe.h      cpp < utf-8[unix] < 33% : 143: 1 < ! [145]tr...
```

Interesting...

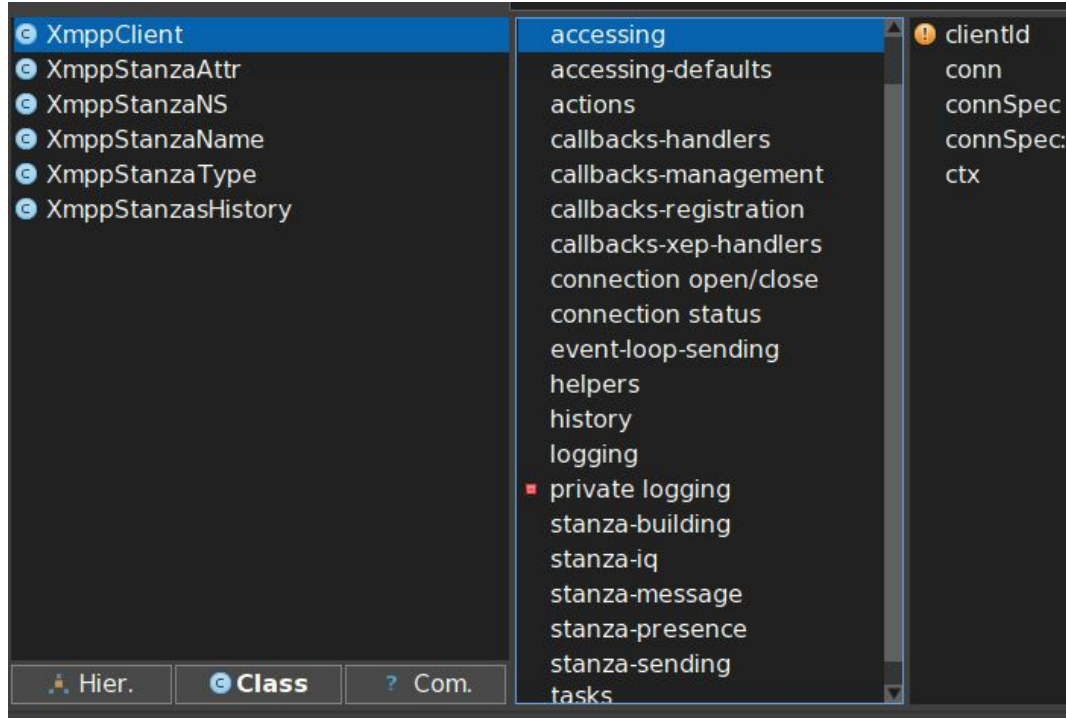
Typedefs

Constants

Callbacks



One needs a Pharo based client: XmppClient



One has to map callbacks

The image shows a screenshot of an IDE interface. At the top, a window titled "XmppClient>>#handleMessage:" is open. Below the title bar, there are tabs for "Scoped" and "Variables". The "Variables" tab is active, showing a tree view of the class hierarchy. The "XmppClient" class is selected. To the right of the tree view is a "History Navigator" showing a list of methods: "-- all --", "accessing", "accessing-defaults", "actions", "callbacks-handlers", "callbacks-management", and "callbacks-registration". The "callbacks-handlers" method is selected. Below the "History Navigator" is a list of callback methods: "handleDicoInfo:", "handleMessage:", "handlePing:", "handlePresence:", and "handlePresenceChange:". The "handleMessage:" method is selected. Below the "callbacks-handlers" method is a list of callback classes: "connHandlerCallback", "discoInfoHandlerCallback", "messageHandlerCallback", "pingHandlerCallback", "presenceChangeHandlerCallback", "presenceHandlerCallback", "timeHandlerCallback", and "versionHandlerCallback". The "connHandlerCallback" class is selected. At the bottom of the IDE, there are three buttons: "Hier.", "Class", and "Com.". The "Class" button is active.

Scoped Variables

Type: Pkg1|^Pkg2|Pk.*Core\$

- LibStropheUFFI
- ExternalStructures
- Tests
- XmppClient**
- XmppClient-GUI

XmppClient

- XmppStanzaAttr
- XmppStanzaNS
- XmppStanzaName
- XmppStanzaType
- XmppStanzasHistory

History Navigator

- all --
- accessing
- accessing-defaults
- actions
- callbacks-handlers**
- callbacks-management
- callbacks-registration

- handleDicoInfo:
- handleMessage:**
- handlePing:
- handlePresence:
- handlePresenceChange:

- connHandlerCallback**
- discoInfoHandlerCallback
- messageHandlerCallback
- pingHandlerCallback
- presenceChangeHandlerCallback
- presenceHandlerCallback
- timeHandlerCallback
- versionHandlerCallback

Hier. Class Com.

In order to implement XEPs

The screenshot shows an IDE window titled "XmppClient>>#handleMessage:". The interface is divided into several panels:

- Scopes:** Shows the current scope as "XmppClient".
- Variables:** Lists variables including XmppStanzaAttr, XmppStanzaNS, XmppStanzaName, XmppStanzaType, and XmppStanzasHistory.
- History Navigator:** A list of methods and actions, with "callbacks-xep-handlers" selected.
- Method List:** A list of methods including handleDicoInfo, handleMessage, handlePing, handlePresence, and handlePresenceChange.
- Code Editor:** Displays the implementation of the `handleMessage` method, which is a function that takes a `stanzaOpaqueObject` and returns a `stanzaOpaqueObject`. The code includes logic for handling reply text, body child, and error stanza.

```
handleMessage: stanzaOpaqueObject
| replytext bodyChild bodyStanzaOpaqueObject actualReplyStanzaType intext errorStanza from textStanzaOpaqueObject
replyStanzaOpaqueObject incomingStanzaType |
  (bodyChild := lib
    xmppStanzaGetChild: stanzaOpaqueObject
    byName: XmppStanzaName NAME_BODY) = 0
  ifTrue: [ ^ 1 ].
(errorStanza := lib
  xmppStanza: stanzaOpaqueObject
```

At the bottom of the IDE, there are status bars and a footer:

- Line 1/47 [1]
- Format as you read W +L
- Long methods ? x
- Uncommon message send ? x
- Helpful? icons

One has to see what is going on

Modeled after ZnLogEvent thing but will move to Beacon.

The screenshot shows the 'Xmpp Log' application interface. It is divided into three main panes:

- Left Pane (Log List):** A table with columns 'Time' and 'Announcement'. The selected row is:

Time	Announcement
2017-05-18T11:06:38.072	2017-05-18 11:06:3
- Middle Pane (Variable Inspector):** A table with columns 'Variable' and 'Value'. The selected variable is 'stanzalInfo', which is a 'Dictionary [4 items]'. Other variables include 'self', 'clientId', 'id', and 'timestamp'. Below the table is a text preview of the selected dictionary's content:

```
"2017-05-18 11:06:38 048 defaultClient
stanza received a
Dictionary[#direction->#received
#stanzald...not-found
xmlns=""urn:ietf:params:xml:ns:xmpp-stanzas""
/></error></message>' #stanzaType->'error'
]"
```
- Right Pane (Dictionary View):** A table with columns 'Key' and 'Value'. It shows the structure of the dictionary:

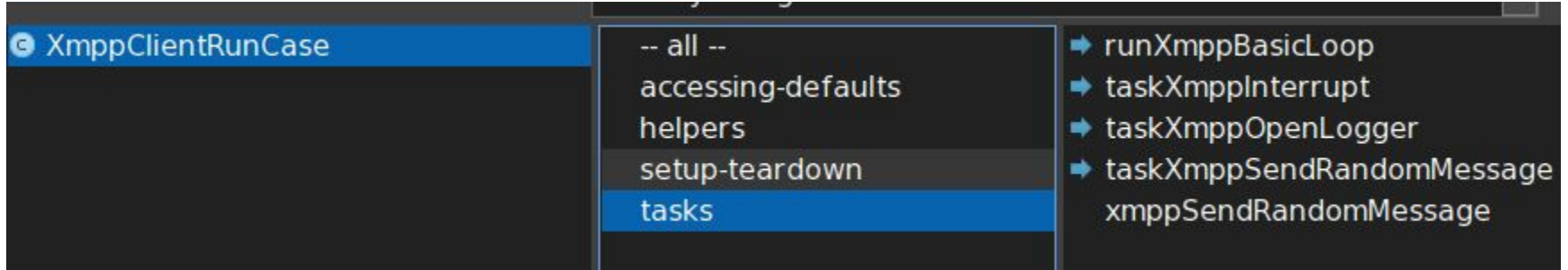
Key	Value
#direction	#received
#stanzald	nil
#stanzaString	'<message type="err
#stanzaType	'error'

Raw stuff can also be logged !console) and one can thus see the data stream

```
tp://jabber.org/features/compress"><method>zlib</method></compression><bind xmlns="urn:ietf:params:xml:ns:xmpp-bind"/><session xmlns="urn:ietf:params:xml:ns:xmpp-session"/></features>
conn DEBUG SENT: <iq id="_xmpp_bind1" type="set"><bind xmlns="urn:ietf:params:xml:ns:xmpp-bind"><resource>PharoIDE</resource></bind></iq>
xmpp DEBUG RECV: <iq id="_xmpp_bind1" to="vmfractal/f1bdd050" type="result"><bind xmlns="urn:ietf:params:xml:ns:xmpp-bind"><jid>pharo@vmfractal/PharoIDE</jid></bind></iq>
xmpp DEBUG Bind successful.
conn DEBUG SENT: <iq id="_xmpp_session1" type="set"><session xmlns="urn:ietf:params:xml:ns:xmpp-session"/></iq>
xmpp DEBUG RECV: <iq id="_xmpp_session1" to="pharo@vmfractal/PharoIDE" type="result"/>
xmpp DEBUG Session establishment successful.
conn DEBUG SENT: <presence/>
xmpp DEBUG RECV: <presence to="pharo@vmfractal/PharoIDE" from="pharo@vmfractal/PharoIDE"/>
conn DEBUG SENT: </stream:stream>
2017-05-18T11:42:10.296709+02:00 -- Disconnected
event DEBUG Stopping event loop.
2017-05-18T11:42:10.298184+02:00 -- Stop for context done
conn DEBUG Can't reset connected object.
2017-05-18T11:42:10.298564+02:00 -- Connection released
2017-05-18T11:42:10.29864+02:00 -- Context Freed
2017-05-18T11:42:10.298694+02:00 -- Shutdown done
```


It is annoying to use test cases for interactive exploration

Using RunCase approach



One needs a GUI, one day

Using Spec

Leveraging the announcements

Target is have a one to one chat and a MUC (Multi User Chat)

Conclusion

Pharo proved to be suitable for this

Very easy to implement callbacks

Promising for new XEPs

Stable

Easy to debug

Horizons

Have a couple XMPP servers in the cloud

Communicate right from the IDE