xDash - The convergence between backend and instant messaging

What is xDash?

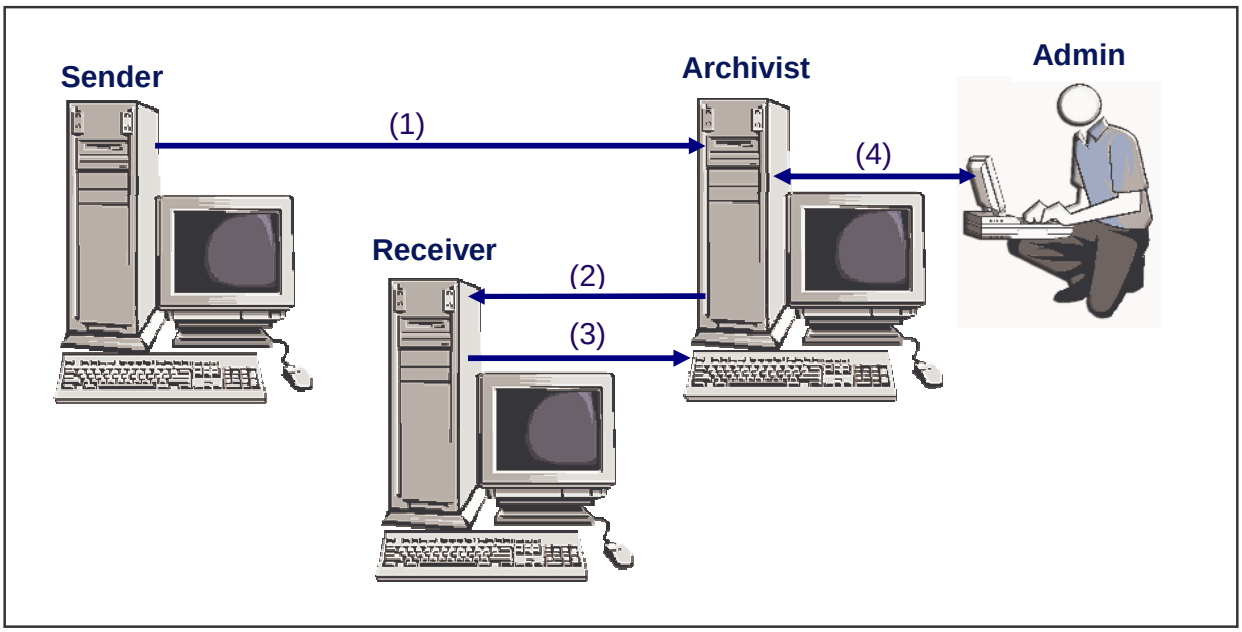
xDash is an Enterprise Service Bus (ESB) for Enterprise Application Integration (EAI). xDash is using Jabber as communication protocol and implementing the concept of convergence between backend asynchronous messaging and instant messaging.

Why should I use xDash?

You should consider using xDash when:

- You have to manage a fast integration project with little time and low budget;
- You are convinced, that integration has so many faces, that agile approach and flexible design are better than buying a “silver bullet” from big names;
- You are convinced, that integration is rather a continuous, never-ending process than a single act of some software implementation;
- You would like to concentrate on writing the integration logic in a well known environment and have no time to learn myriads of new options and APIs with little reuse;
- You need an open human readable communication protocol (XML), where you can see what is going on under the hood;
- You need a free open source multiplatform information bus for publish/subscribe integration;
- You are interested in a ground-breaking project dealing with the convergence of instant and backend asynchronous messaging.

How does the xDash work?



xDash works on the conceptual level as follows (see picture above):

- A Sender gets from someone/something data for a job and the task to start it.
- Sender sends the data to the Archivist to keep him informed, about what he has to do (1).
- Archivist sends the data to Receivers, who have to do the job (2).
- Each Receiver does his job (transforms the received data and puts where needed) and sends the operation result back to the Archivist (3).
- The Archivist tracks all information about the success of the job (4).

What is the job of an xDash administrator?

- An administrator plans, develops and deploys the solution.
- Administrator is responsible for checking Archivist's work to see, if everything is running smoothly. He can also start corrective actions, if something went wrong.

What is the core of the xDash?

xDash is at basis every sort of an integration project, which sticks to the following rules:

- Communication over Jabber protocol;
- The extension of the message type to type='job' for all relevant messages;
- The message sending schema, in which always participate a message sender, a message receiver and a message archivist ("integration chain").
- For each sort of event, triggering a Sender, there is only one integration chain defined on the server by a set of Jabber IDs and forwarding rules ("Who knows whom" schema).
- The client software of Sender and Receiver knows nobody but the Archivist.
- The jabber server forwards Sender's messages sent to the Archivist, additionally to the Receivers.

Where is machine, where is software and where is a human?

- Sender is client software usually installed on a machine like web server, where triggering events happen. It detects events in applications and runs without an interference with humans. The development of the event detection is done by the xDash administrator.
- Receiver is client software installed somewhere close to a system like ERP or even on the same machine. It consumes data from the Sender's job and does the action processing, where needed. It runs without an interference with humans. The development of the interaction with a system, like an ERP or directory, is done by the xDash administrator.
- Archivist is client software installed on the same machine as Jabber server or close to it. It tracks everything what happens to Sender and Receivers and runs without an interference with humans.
- Administrative Tools are a set of software utilities installed usually on the same machine as the xDash database. They allow administrator sorting messages and purging the database.

"Who knows whom" schema explained

The publish/subscribe schema is implemented on Jabber server and in the client software. It is assumed, that:

- Jabber server is security hardened and if necessary a HA jabber cluster build;

- Sender's, Receiver's, Archivist's client configuration should give as little clue as possible about the integration chain architecture;
- Messages, foreign to the chain, should be stopped on the way to Sender, Receiver and Archivist as early as possible.

Following rules are implemented on the client side:

- Sender knows only Archivist;
- Receiver knows only Archivist;
- Archivist knows nobody but sends back a message entry confirmation, if a valid Jabber ID exists and message type is 'job'.

Following rules are implemented on the server by defining on Archivist's account appropriate Jabber XML forwarding rules for incoming messages:

- If a message is from Sender forward it to Receivers;
- Continue message delivery to the Archivist.

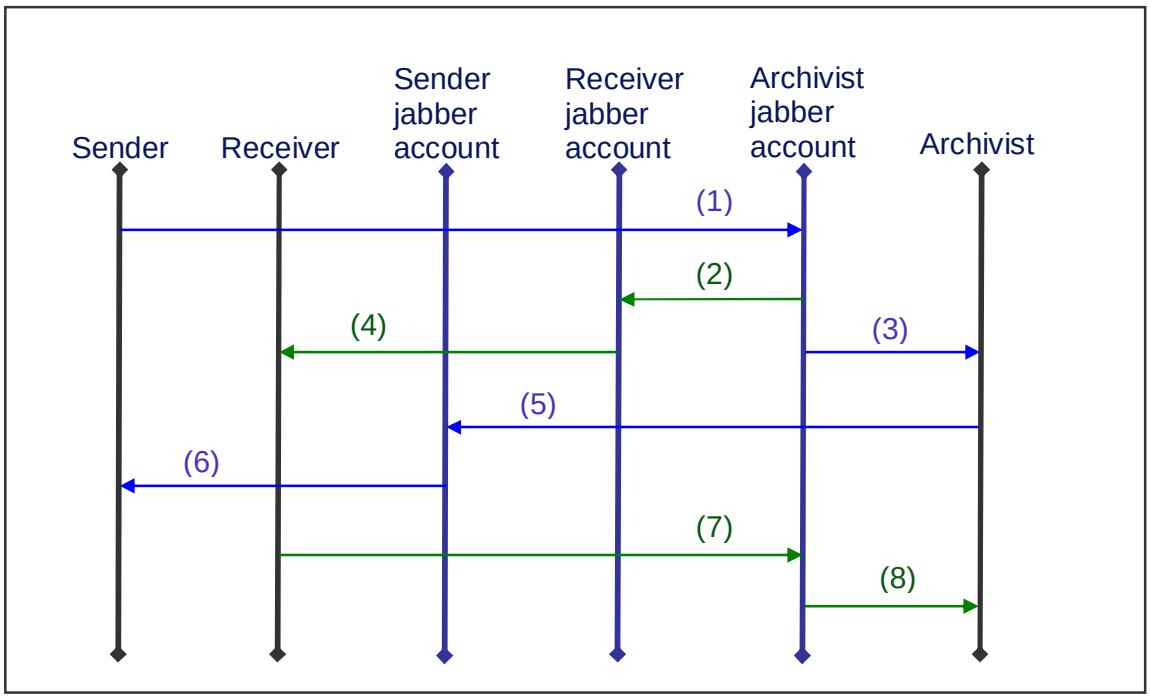
Survival strategy of the xDash integration architecture

- "The show must go on!" – the foremost aim of the integration project is to provide reliable communication between applications, close to the real time from the integrated applications point of view. The data, which should be published to other applications, has to be processed as soon as possible, even at the cost of processing the same data again. This means, that the working strategy of xDash must be fault tolerant, also to the internal problems.
- Lazy evaluation –xDash assumes that in 99% cases everything goes well and does things only when really needed. Processing information like data extraction and transformation is done where the information is used – on the receiver.
- Storing only as much information as needed – In case something goes wrong only as much information should be kept persistent as it is needed to reconstruct and relaunch an action.

How the xDash survival strategy is implemented?

- Each Sender, Receiver, Archivist are registered with the operating system watching them constantly and restarting if killed by accident or internal error.
- Each Sender, Receiver, Archivist tries to reconnect to the server, when the network connection is lost.
- Sender takes a job from the waiting queue and sends for tracking to the Archivist. The Archivist's forwarding rules on the jabber server forward the message to the Receivers prior to the delivery to the Archivist (even if the Archivist has some processing problems, the integration still works). After successful serialisation, the Archivist sends the acknowledgment back to the Sender. The Sender deletes the job from the waiting queue. If something wrong happens, the job stays in the queue and waits for another trial.
- Each message has a unique key as the thread. The integration scripts in "onMessage" event handler on Receivers should be written so that they are tolerant to the same job coming double (this is an assumption valid for any network application).
- Administrative Tools are nothing more than a set of utilities, which can be used manually or in an automation script. They allow tracking and reconstructing of actions and data, even if not all messages reached their destinations.

A detailed look how xDash works – all pieces together



- Sender detects a new job in the waiting queue and sends to Archivist's account a job notification (1);
- The Archivist's account checks, if the message is from Sender and forwards it immediately to the Receiver's account (2);
- The Archivist account delivers the message to the Archivist (3);
- The Archivist does serialisation and sends back to the Sender the OK-acknowledgment (5);
- If the Archivist OK message comes (6), the Sender moves to the next job;
- The Receiver's account delivers the message to the Receiver (4);
- The Receiver extracts the message and processes it. The result is sent to the Archivist's account (7);
- The Archivist's account checks, if the message is from Receiver or Sender and delivers it to the Archivist for serialisation (8).

What is your decision?

- The choice of the implementation of the Jabber server;
- The choice of the implementation for Sender, Receiver, Archivist and Administrative Tools;

- The choice of the Integration Planning Kit.

What can you find at the moment at the http://xdash.jabberstudio.org?

- Reference implementation for Sender, Receiver, Archivist and Administrative Tools in Perl. The Sender starts a job on dropping a file in a directory, the name of the file is taken as the job thread. All the Perl code is self-explanatory.
- Integration Planning Kit based on MS Excel;
- Documentation, how to do an integration project with the help of xDash;
- Inspirations for further development, for everybody who would like to participate in the xDash project.