WIDE Technical-Report in 2007

wija: an Open and Extensible Messaging Platform on the Internet

wide-tr-ideon-wija-platform-00.pdf

WIDE Project : http://www.wide.ad.jp/

wija: an Open and Extensible Messaging Platform on the Internet

Kenji Saito[1]

[1]Research Institute for Digital Media and Content, Keio University

1 Introduction

1.1 Purpose of Development

Our purpose is to assure freedom of experimentation for researchers, software developers and anyone in pursuit of problems.

Openness, autonomy, extensibility, and resulted dependability/sustainability are the basic properties of the Internet, which we would like to enhance. Although IP provides global reachability as a foundation for freely proposing, testing and deploying new services, it is not an easy tool for creation to be used by general public. We are to provide an open platform on top of TCP/IP, or another Internet over IP, at a level closer to human beings and human relations, so that anyone can participate in creation of new communication on the Internet.

For this purpose, we have developed *wija* as a platform for experimenting with all possibilities of communication on the Internet. This statement is not an exaggeration. The basic semantics of IP is that a message is reachable to a destination specified by an identifier/locator. A communication protocol implementing this semantics should be able to cover all possibilities of the Internet. This is why we have chosen to use an instant messaging protocol in our development, which implements the basic semantics of the Internet at a human level.

1.2 Mission Statement

We have set forth the following mission statement:

M-1: Messaging Platform

We design *wija* to be a 1) cross-platform, 2) internationalized and 3) interoperable messaging software, 4) with means for extending its functionalities.

The need for interoperability would require *wija* to conform to available centralized messaging protocols. But centralization deprives users and developers of freedom of innovation. Centralization also have negative effects on dependability and sustainability, because when a server is down, there is nothing outsiders of the domain can do for recovering the services.

The design of *wija* should shift toward decentralization.

M-2: End-to-Endness

We design *wija* to be end-to-end oriented, so that problems in new applications are not to be solved by adding new features to servers, but by communication among clients.

In the end, it should eliminate all dependencies on servers for maximal autonomy, dependability and sustainability. This makes safe communication among clients particularly important.

M-3: Security

We design *wija* to be integrated with a foundation for secure communication, which must provide end-to-end public key cryptography. *wija* should provide easy-to-use human interface for handling keys and certificates, and encrypting/signing messages.

M-4: Freedom

We develop *wija* as a free software, without restrictions for further innovation by any parties.

1.3 Challenges

There are several challenges for completing the above missions:

C-1: Pathway to P2P (peer-to-peer)

One challenge is to set the pathway in the development for future elimination of dependencies on servers.

In addition to moving from centralized to more P2P-oriented protocols, we should also consider how we can eliminate dependencies on the web in distributing software and providing support for users and developers.

C-2: User Friendliness

Another challenge is to design an easy-to-use integration with the foundation for secure communication so that anyone can handle public key cryptography.

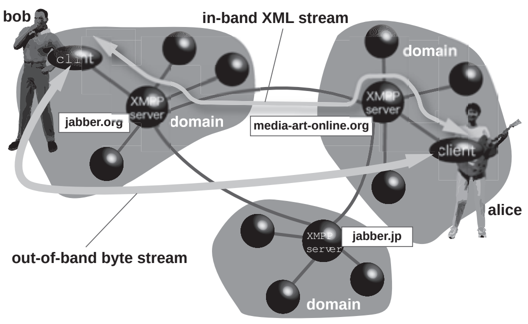


Figure 1: Overview of XMPP communication

2 Background – Jabber/XMPP

2.1 History

Jabber/XMPP is a set of open protocols for instant messaging and presence sharing.

It was first proposed as *Jabber* protocol in 1998. The first version of an open source Jabber server, *jabberd*, was released in year 2000. JSF (Jabber Software Foundation) was established in 2001 to maintain the specifications and to support their enhancements. In 2004, the core protocols of Jabber were standardized by IETF[46] as Extensible Messaging and Presence Protocol (hence the name Jabber/XMPP).

Jabber/XMPP has been applied to many instant messaging systems including those developed by Apple, FedEx, Google, Oracle and Sun Microsystems.

2.2 Characteristics

The characteristics of Jabber/XMPP in comparison with other instant messaging systems are as follows:

1. *Openness*

   Specifications of Jabber/XMPP are open to public. Anyone can implement the protocols for free. Readers can find lists of clients and servers at [19].

2. *Autonomy*

   The network of Jabber/XMPP, as illustrated in Fig. 1, is close to that of SMTP[33] (Simple Mail Transfer Protocol); mail servers can be set up by anyone without permissions from any parties once a domain name is registered, and anyone can freely join the network of the global electronic mail system. Likewise, XMPP servers can be set up without any permissions, and anyone can freely join the global network of the instant messaging and presence sharing system.

3. *Extensibility*

   Jabber/XMPP is an extensible set of protocols whose data descriptions are based on XML[6]. New features can be incorporated without breaking the existing system. Extended protocols can be submitted to XSF (XMPP Standards Foundation) as an XEP (XMPP Extension Protocols) for standardization. An XEP is approved as a standard after feedbacks from implementers and reviews from XSF.

2.3 Communication Mechanism

In Jabber/XMPP, a peer is identified by a Jabber ID of the following form: $\boxed{user@domain/resource}$

A *user* name defines a user in a domain. A *domain* name is effectively the host name of the server to which the user is connected with a client. A *resource* name distinguishes a communication session from others by the same user in the same domain.

Fig. 1 illustrates the communication mechanism of Jabber/XMPP.

Like SMTP, clients need to talk to a server in order to have their messages reach the clients on the other ends (*in-band XML stream*). It implies that Jabber/XMPP has such a weakness that communication becomes impossible when either of the servers in between is down. Therefore, it is our intention that it will be replaced by a more P2P oriented technology in the future to achieve the level of dependability and sustainability required for our purposes.

Jabber/XMPP also provides means for the clients to directly communicate (*out-of-band byte stream*), using protocols such as SOCKS5[22]. It is recommended that clients use out-of-band communication whenever a large data is involved. Still, in-band communication takes an important role in identifying the peers by their IP addresses and port numbers, which can dynamically change; because of this, Jabber/XMPP provides an excellent way to rendezvous for applications on the Internet.

3 Core Design

3.1 Primary Decisions

3.1.1 Development Language

We have chosen Java[43] as the development language for *wija*. We did it mainly for ease of implementing cross-platform and internationalized software. In addition, we hoped that popularity of the language would attract more developers.

A drawback of selecting Java for a client-side application is the necessity for users to prepare the runtime environment. Although this has never been a problem for Mac OS X[4], on which Java 2 Standard Edition is pre-installed, and increasingly less problematic for Linux[23] platforms because of GCJ[12] (The GNU Compiler for Java) and GNU Classpath[13], this has always been a concern for Windows[25]. Section 6.2.1 discusses whether this decision has affected user distribution.

3.1.2 Communication Protocol

We have chosen Jabber/XMPP as the communication protocol to start with, and then shift toward a more P2P-oriented protocol for enhanced dependability and sustainability.

Many available instant messaging services today, such as MSN[26], AIM[2], Yahoo![49] and Skype[40] are proprietary. They may become interoperable with one another, but even then, freedom of innovation will not be ensured for general public.

We believe that Jabber/XMPP is the best available choice for the instant messaging protocol for our purposes because of its openness, autonomy and extensibility as described in section 2.2. It also provides a foundation for experimenting with P2P protocols using out-of-band byte streams.

3.1.3 Policy on Conforming to Standards

Conformity is important for assuring interoperability, and attracting potential users of existing messaging systems to use *wija*. But at the same time, *wija* should be inherently experimental.

Our policy is that the end-to-end principle (M-2) should take higher priority over conformity to existing protocols, unless the protocol is too popular in practice to ignore, in which case lack of interoperability is considered more harmful.

3.1.4 Cryptographic Framework

We have chosen OpenPGP[7] as the cryptographic framework because of its end-to-endness implied from the web of trust[45], and GnuPG[44] as its implementation because it is a free software. We have designed *wija* to be integrated with GnuPG (version 1.x), and to have a GUI front-end for handling all PGP operations.

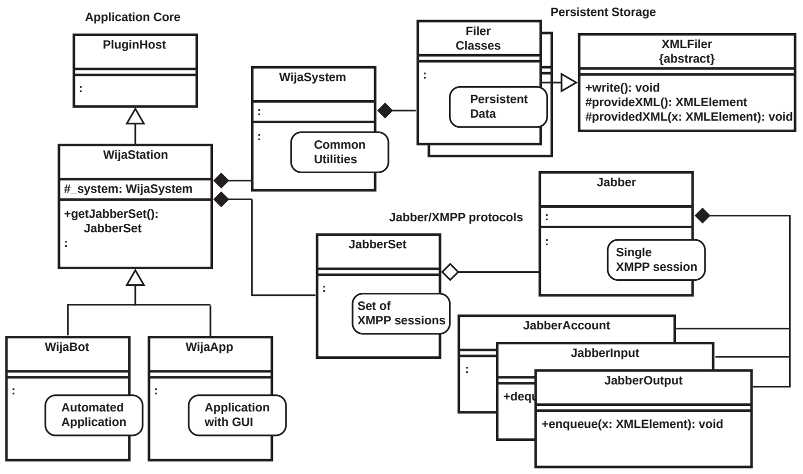
It does not mean that *wija* ignores X.509[18]. Jabber/XMPP allows communication between a client and a server, and between a server and another server encrypted by TLS[8] (Transport Layer Security). Without supporting this, *wija* would lose compatibility with popular XMPP services as the one provided from Google, Google Talk[14].

By the above described policy, we have implemented TLS connection with XMPP servers using Java security framework, so that users can log into Google Talk using *wija* (a big advantage to us for increasing the number of potential users).

By the same policy, we have designed a direct end-to-end PGP public key exchange protocol over Jabber/XMPP as described in section 4.3.2, in addition to key exchange using public key servers.

3.1.5 Distribution License

We have chosen GNU GPL[11] version 3 as the license under which *wija* is distributed.



- *PluginHost* is an abstraction of a program that can be extended by plug-ins. A reference to the *PluginHost* object is passed to each plug-in when the plug-in starts.
- *wija* (implemented as a *WijaApp* object) and *wijabot* (implemented as a *WijaBot* object) share core components with respect to communication and storage.
- Since version 0.13, *wija* supports simultaneous multiple XMPP sessions with different Jabber IDs. All accesses to XMPP protocols need to go through *JabberSet* object that provides search functions for the particular XMPP session in question.

Figure 2: Core components of *wija* in UML class diagram

3.2 Core Components

Fig. 2 is a UML[32] (Unified Modeling Language) class diagram for core components of *wija*.

We have developed software *robot* version of *wija*, which we call *wijabot*. *wija* and *wijabot* share most features including hypertext sharing (section 4.2.1) and PGP public key exchange (section 4.3.2), but *wijabot* does not have direct human interface while running. *wijabot* is extensible with plug-ins with the same architecture (section 4.1.4) as *wija* to implement necessary automation to fit the purposes. *wijabot* has been used for providing services such as SOCKS5 proxy (section 4.2.2) which do not require human intervention.

Java classes of *wija* are designed in such a way that *wija* and *wijabot* can share most of the code.

As a general rule, we have a class in the package of *wija* (*org.media_art_online.wija*) that implements a feature required in *wijabot* (e.g. *WijaStation*), and a separate class provides user interface to be used in *wija* (e.g. *WijaApp*). Compared to *wija*, only 3% of code was needed to implement *wijabot*.

4 Design for Missions

4.1 Design for M-1: Messaging Platform

4.1.1 Cross-platform Support

We have successfully made *wija* run on major operating system platforms such as Linux, Mac OS X and Windows by writing the code in a platform-independent manner; differences among platforms not concealed by Java are mostly abstracted within the core components. Fig. 3 shows a screenshot of *wija* and its plug-ins on Mac OS X.

There was a challenge in providing a cross-platform mechanism to link external plug-ins with *wija*. Our determination has been that all executable files including those of external plug-ins must be able run on any platforms. Otherwise, we would need more efforts on supporting multiple platforms, and features such as secure P2P update (section 4.2.3) would have limited applications.

This linkage has been done by class loaders provided by the Java virtual machines. The mechanism requires the file name of the JAR (Java ARchive) file of a plug-in to be named as follows: $\boxed{\textit{full-name-of-the-class}\text{.jar}}$

For example, the plug-in file for *i*-WAT[38][24] is named "org.media_art_online.iwat.Iwat.jar".

Upon start-up, *wija* searches for those files under the *plugins* directory inside the program directory. When it finds one, it tries to load the class specified by the file name from the JAR file, which will result in loading all available referred classes recursively.

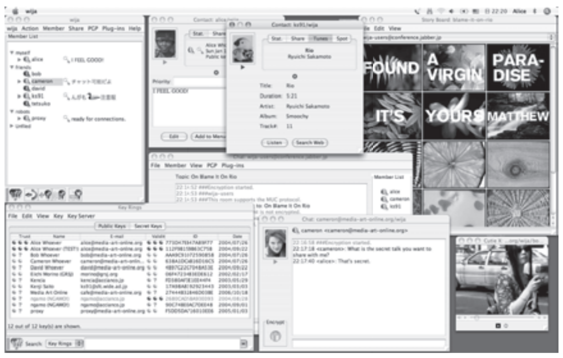


Figure 3: Screenshot of *wija* and its plug-ins

An alternative is to store the name of the class as a property of the plug-in JAR file, but our design provides simpler build process and retrieval of the class at runtime.

4.1.2 Internationalization

wija utilizes localization support of Java for internationalization. Currently all messages to users are provided in two languages: English and Japanese. Developers can add properties files to support other languages.

4.1.3 Interoperability

Table 1 shows the list of XEPs implemented in *wija* to assure interoperability with other Jabber/XMPP messaging clients.

Table 1: XEPs Implemented in *wija*

XEP #	*Name*
XEP-0020	Feature Negotiation[27]
XEP-0027	Current Jabber OpenPGP Usage[28]
XEP-0030	Service Discovery[16]
XEP-0045	Multi-User Chat[35]
XEP-0047	In-Band Bytestreams[20]
XEP-0065	SOCKS5 Bytestreams[41]
XEP-0082	Jabber Date and Time Profiles[34]
XEP-0086	Error Condition Mappings[31]
XEP-0095	Stream Initiation[29]
XEP-0096	File Transfer[30]
XEP-0115	Entity Capabilities[17]
XEP-0153	vCard-Based Avatars[36]

wija implements XEP-0115: *Entity Capabilities* so that a capability list is sent with presence from others, which is stored locally, and checked before *wija* or its plug-ins try to send any non-standard messages to the peer. In this way, *wija* can avoid obstructing communication when it is put among other clients in the instant messaging network.

Alternatively, we could use XEP-0030: *Service Discovery* to discover features supported by peers, but it would require more messages and code.

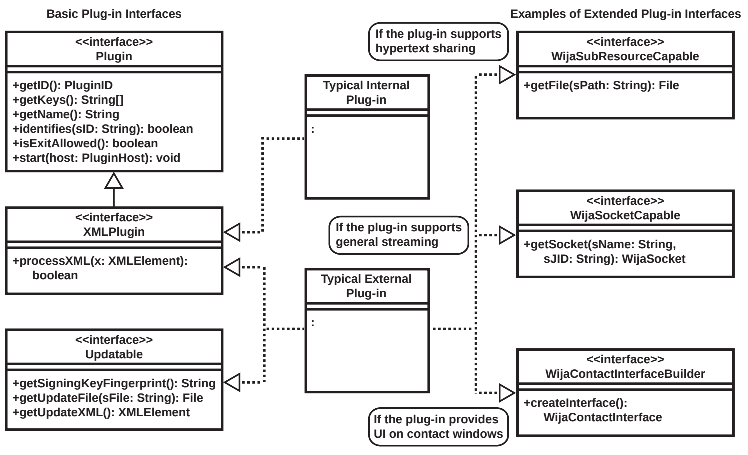
4.1.4 Means for Extension

wija allows new functionality to be added as a plug-in. Fig. 4 is a UML class diagram for plug-in APIs of *wija*.

Many XEPs, such as XEP-0045: *Multi-User Chat* and XEP-0047: *In-Band Bytestreams*, are implemented as *internal* plug-ins using the plug-in APIs, which has eased incremental development of *wija*. Those plug-ins are included in the executable file of *wija*, so that they must not implement *Updatable* that specifies external executable files to update. This is the reason why basic plug-in APIs have separated interfaces.

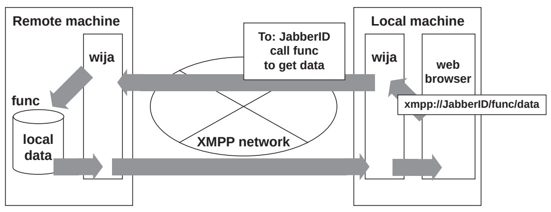
There are various extended plug-in interfaces that plug-ins may implement to provide certain types of functionalities.

Upon starting *wija*, all classes of found plug-ins are loaded first, and then the initialization code of each plug-in is called. This allows a plug-in to access public methods and fields of other plug-ins, enabling developers to utilize existing extensions for their own extensions.


Basic Plug-in Interfaces
Examples of Extended Plug-in Interfaces
<<interface>> Plugin
+getID(): PluginID
+getKeys(): String[]
+getName(): String
+identifies(sID: String): boolean
+isExitAllowed(): boolean
+start(host: PluginHost): void
<<interface>> XMLPlugin
+processXML(x: XMLElement): boolean
<<interface>> Updatable
+getSigningKeyFingerprint(): String
+getUpdateFile(sFile: String): File
+getUpdateXML(): XMLElement
Typical Internal Plug-in
Typical External Plug-in
If the plug-in supports hypertext sharing
If the plug-in supports general streaming
If the plug-in provides UI on contact windows
<<interface>> WijaSubResourceCapable
+getFile(sPath: String): File
<<interface>> WijaSocketCapable
+getSocket(sName: String, sJID: String): WijaSocket
<<interface>> WijaContactInterfaceBuilder
+createInterface(): WijaContactInterface

- It is mandatory for any *wija* plug-ins to implement *XMLPlugin*, which can take XML messages not understood by the core components.
- It is highly recommended that an external plug-in implements *Updatable*, which is called when secure P2P update of the software (section 4.2.3) is performed.
- Depending on the functionalities of a plug-in, it may implement some of extended plug-in interfaces, such as *WijaSubResourceCapable* for hypertext sharing (section 4.2.1).

Figure 4: Plug-in APIs of *wija* in UML class diagram



- *func* is either an internal or external plug-in that implements *WijaSubResourceCapable*.

Figure 5: Hypertext transfer by *wija*

4.2 Design for M-2: End-to-Endness

4.2.1 Hypertext Sharing

Hypertext sharing provides an illusion of direct retrieval of data from a client at the other end, using a web browser via HTTP[5][10] (Hypertext Transfer Protocol). This feature works even when both computers are inside their own private networks or within firewalls, using SOCKS5 proxy as described in the section to follow.

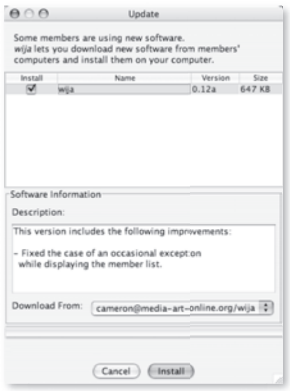
This is a basis for many plug-ins to conduct end-to-end communication with peers. We have been experimenting with the following form of URL: $\boxed{\text{xmpp://}\textit{JabberID/function/function-dependent}}$ to be fed to *wija* to initiate hypertext transfer.

Hypertext sharing is realized by running a pseudo-HTTP server locally inside *wija*, and having the web browsers of the user's choice access the server on the localhost, as illustrated in Fig. 5. The server initiates streams for file transfers, via a proxy service if necessary, and all data transfers are performed at the back as in-band or SOCKS5 byte streams.

4.2.2 Proxy Discovery

XEP-0065: *SOCKS5 Bytestreams* defines the roles of SOCKS5 proxy to intermediate two Jabber/XMPP entities which cannot directly communicate with each other because they are inside private networks or firewalls. A SOCKS5 proxy is usually implemented as a service of an XMPP domain which users must predetermine to use, but the end-to-end principle (M-2) has urged us to design otherwise for *wija*.

All *wija* clients are designed to be capable of providing a proxy service. This feature can be turned on and off by the users.


Update

Some members are using new software.
wija lets you download new software from members' computers and install them on your computer.

Install	Name	Version	Size
✓	wija	0.12a	647 KB

Software Information

Description:

This version includes the following improvements:

- Fixed the case of an occasional exception
  while displaying the member list.

Download From: cameron@media-art-online.org/wija

Figure 6: Update window

When a direct SOCKS5 connection is found impossible, *wija* searches for an available proxy service in its buddy list. This allows users to set up a proxy service on some machine using an only regular distribution of *wija*[1], and make the service available to them by adding the entity onto their buddy lists.

There is a proxy service provided by us, whose Jabber ID is "proxy@media-art-online.org", but every *wija* user has liberty to set up their own proxies, or to become one themselves.

4.2.3 Secure P2P Update

Since version 0.12, *wija* allow users to update it or its plug-ins by directly downloading the new software from the computers of their buddies if they use newer versions. The validity of the new software is automatically verified since they are digitally signed by the system's public key, which is imported the first time *wija* is started. The system's public key is stored in the internal file directory within the executable file of *wija*.

As Fig. 6 shows, the buddy from whom the software is downloaded can be selected by the user if there are multiple candidates.

Secure P2P update is performed using hypertext sharing. The update feature is implemented as an internal plug-in of *wija* that implements *WijaSubResourceCapable*. *wija* and external plug-ins register with the update plug-in so that when the update module receive a request containing the identifier of those modules it passes the message for requesting files to them. The actual transfer is performed by the hypertext transfer engine of *wija*.

It is important to log occurrences of this update for measurement purposes to find out the actual user distribution of *wija*. When the *wija*'s executable JAR file has been received from a peer, *wija* records the Jabber ID of the peer. When someone else receives the executable from the client, it sends "increment" message to the Jabber ID. Upon receiving the message, *wija* records the incremented number to its local storage. The message is forwarded to their parents if they have ones. Finally, those who have downloaded the new version directly from the web site (or the developer himself for that matter) has the approximate number of peers who are using the new version.

It is possible to record the identities of those who have downloaded the new versions, but *wija* does not do so for privacy protection reasons.

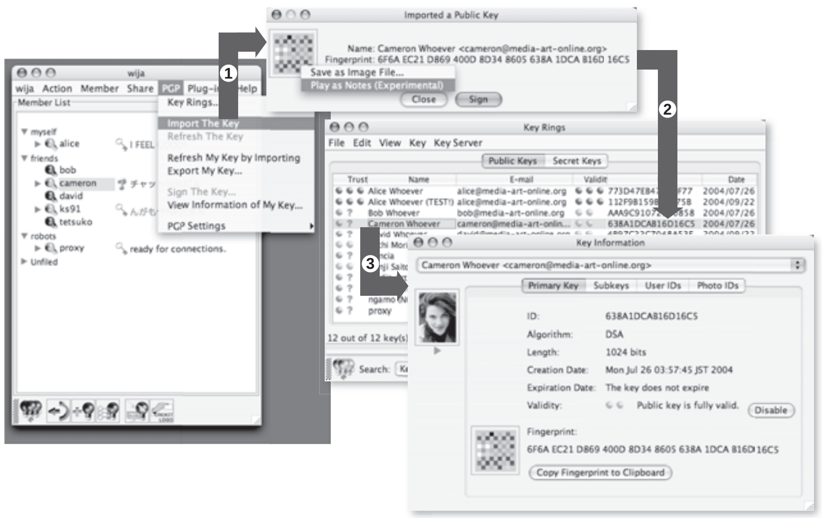
4.3 Design for M-3: Security

4.3.1 Integration with GnuPG

For providing cryptographic foundation, *wija* is designed to cover most functions of *gpg* command (version 1.4.x) in graphical ways, as well as the features defined in XEP-0027: *Current Jabber OpenPGP Usage*.

Supported GnuPG features include signatures to files and verification of them, encrypting files and decrypting them, importing and exporting public keys, searching in local key rings and in key servers, generation of key pairs, signature to public keys, generation of revocation certificates, changing passphrases, deletion of public or secret keys, browsing information on keys including photo IDs, adding and deleting user IDs and photo IDs, and sending public keys to and receiving them from key servers.

[1] Although usually *wijabot* (section 3.2) is used for the purpose of setting up a SOCKS5 proxy.



Name: Cameron Whoever <cameron@media-art-online.org>
Fingerprint: 6F6A EC21 D869 400D 8D34 8605 638A 1DCA 816D 16C5

Trust	Name	E-mail	Validit		Date
● ● ●	Alice Whoever	alice@media-art-online.org	● ● ●	773D47E84[illegible]F77	2004/07/26
● ● ●	Alice Whoever (TEST!)	alice@media-art-online.org	● ● ●	112F98159[illegible]758	2004/09/22
● ?	Bob Whoever	bob@media-art-online.org	● ●	AA9C9107[illegible]858	2004/07/26
● ?	Cameron Whoever	cameron@media-art-onlin...	● ●	638A1DCAB16D16C5	2004/07/26

Cameron Whoever <cameron@media-art-online.org>

ID: 638A1DCAB16D16C5
Algorithm: DSA
Length: 1024 bits
Creation Date: Mon Jul 26 03:57:45 JST 2004
Expiration Date: The key does not expire
Validity: ● ● Public key is fully valid.

Fingerprint:
6F6A EC21 D869 400D 8D34 8605 638A 1DCA B16D 16C5

1. A user can import public keys of others transferred via XMPP.
2. The imported keys are signed after checking their fingerprints. The keys appear in the visual representation of the key ring.
3. By double-clicking on the key, the detail information of the key is displayed.

Figure 7: Integration with GnuPG

4.3.2 Public Key Exchange

By the end-to-end principle (M-2), *wija* is designed to have a replacement feature for the function of key servers; public key exchange can be made through Jabber/XMPP in-band XML streams.

Fig. 7 shows how this feature is integrated. *wija* tracks bindings between public keys and Jabber IDs, which are initiated when users generate a new key pair. Or if *wija* finds just one key in user's secret key ring, a binding is created automatically. Users can import public keys of others by selecting peers on their buddy lists.

Managing a web of trust is hard, especially for novices of using PGP. *wija* provides some experimental ways to represent key fingerprints as illustrated in Fig. 7, as color and sound patterns, to ease the burden of fingerprint checking.

4.3.3 Encapsulation of GnuPG Features

The functionalities of GnuPG are accessed from *wija* by calling *gpg* command through *exec()* method of the Java runtime module. The functionalities are encapsulated as a Java class named *org.media_art_online.gnupg.GnuPG*, so that plug-in writers need not to be bothered by the detail of using GnuPG. The package *org.media_art_online.gnupg* provides Java classes to abstract GnuPG data structures such as keys, user IDs and photo IDs.

4.4 Design for M-4: Freedom

This mission is about providing freedom of innovation to developers, which we have been challenging through practices described in section 5.

The pluggable architecture has made *wija* extensible by allowing anyone to add new functionality, but there is another reason for the design: retractability. Because *wija* is experimental in its nature, some features may not be found socially fit. If a feature is seen that way, in light of copyright laws, for example, the plug-in is safely discarded from the system without affecting the entire platform of *wija*, so that other experiments can go on.

5 Practice

5.1 Development Environment

5.1.1 Build Toolkit

wija's developer team has been using Perforce[42] for concurrent version management, and Apache Ant[3] for building software.

We have configured a machine to perform nightly builds so that any files added locally and missing from the Perforce depot or any discrepancies among modules, especially ones between *wija* and its plug-ins, are detected the next morning.

5.1.2 Documentation Toolkit

Documentation on the web are expressed in what we call *OmniDocument* format which we have developed as an extension of RD for Ruby language. By using the format, not only it is easier to write, but also the resulted web pages are ensured to be displayable by the built-in browsing functionality of Java runtime.

The help pages of *wija* have been generated using the toolkit.

5.2 Public and Developer Relations

5.2.1 Publicities

We have been advertising *wija* via a number of media.

We have set up an official web site of *wija* for general users, and a Wiki site[47] for developers has also been made open to public.

wija is among the lists of Jabber/XMPP clients at the JSF web site and an entry in Wikipedia[48]. It has its own entry in Wikipedia at the following URL:

- http://en.wikipedia.org/wiki/wija

We have introduced *wija* in special articles in the January 2006 issue of the monthly JavaWorld[37] and in the October 2006 issue of the quarterly UNIX magazine[39], both in Japan.

5.2.2 Communities

As for more community oriented activities, we have set up community pages for *wija* at SNS (Social Networking Service) sites *mixi*[9] and *GREE*[15]. There is a *wija* developers mailing list for discussions.

Since February 2007, we have been using SourceForge.net for development management and software distribution at the following project page:

- http://sourceforge.net/projects/wija/

5.3 Public Releases of wija

5.3.1 Releases

wija has been available at the following URL:

- http://www.media-art-online.org/wija/

Table 2 is the chronological list of releases of *wija*. Statistical analysis on the publicity of the recent versions is found in section 6.2.

Table 2: Chronological list of *wija* releases

Date	*Version*	*Description*
Jun 14, 2004	version 0.03	Pre-public release version.
Jun 14, 2004	version 0.04	Initial public release.
Jun 21, 2004	version 0.05	Improved data exchange.
Jul 15, 2004	version 0.06	Many improvements for usability.
Sep 29, 2004	version 0.07	Key rings, photo IDs and Spot-lite.
Jan 3, 2005	version 0.08	Hypertext sharing and Tunes.
Feb 2, 2005	version 0.09	Many improvements for usability.
Apr 5, 2005	version 0.10	Variance over time (*i*-WAT) and PaNIC storyboard.
Dec 6, 2005	version 0.11	GnuPG 1.4.x support and new barter currencies.
Jan 3, 2007	version 0.12	MSN/Google account support, automatic update, etc.
Sep 8, 2007	version 0.13	Simultaneous multiple sessions, Overlay GHC (initial release), etc.

We have also been distributing *wija* from SourceForge.net since version 0.12.

5.3.2 Lessons Learned

These public releases have been made as we learn by the feedbacks from users.

Supports for key rings management and photo IDs (version 0.07), for example, resulted from our observation that users had difficulty in handling public keys.

Variance over time feature of *i*-WAT and PaNIC storyboard (version 0.10), new barter currencies (version 0.11), MSN/Google account support (version 0.12), and simultaneous multiple sessions (version 0.13) resulted from explicit requests from users that are now managed on our SourceForge.net project pages.

6 Results and Evaluation

6.1 Effects on People

6.1.1 Effects to Research Activities

Most existing plug-ins of *wija* have been required from researchers' needs, and were quickly implemented on *wija* to help their research activities.

So far, *wija* and plug-ins have helped one Ph.D. degree (*i*-WAT), 3 masters degrees (Tunes, POCOMZ and related proto-typing) and one bachelor's degree (PaNIC). Spot-*lite* and Cutie X were also proposed at first as undergraduate projects.

We believe that for the researchers, *wija* has been expanding their universe of what they can do.

6.1.2 Relations with Users

Because of *i*-WAT, *wija* has already been in use by many of the WAT System community members as the reference platform of the currency exchange.

Many of such users are not specialized in computing, which makes them perfect for returning feedbacks on the usability of *wija*. Many improvements as described in section 5.3.2 owe feedbacks from them.

6.2 Statistics

6.2.1 Web Server Access Logs

Fig. 8 shows the frequencies of successful downloads of *wija* during the period between December 2005 and January 20, 2007[2].

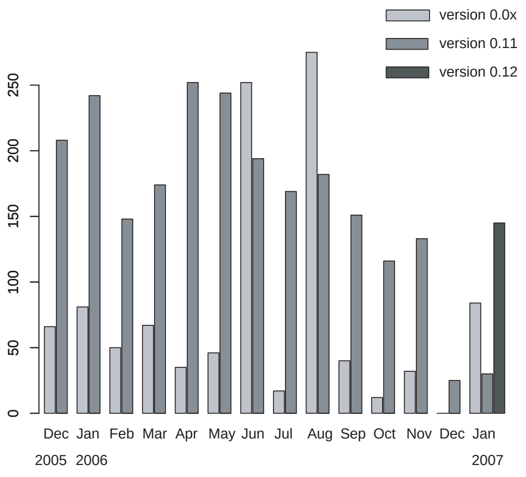
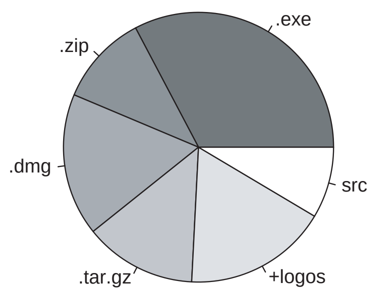


Figure 8: Successful downloads of *wija* Dec/2005 ∼ Jan/2007

The number of downloads in January 2007 shows that transition to the succeeding version has been successful. However, there remain mysterious downloads of prior versions of *wija*, especially in June and August 2006. Further investigating the server record has shown that many downloads were initiated from the same IP addresses in a very short periods, suggesting that those cases are accesses from some kind of software robots.

[2]The data for December 2006 has been mostly lost because of an accident resulted from replacing the web server.



wija011.exe	781
wija011.zip	261
wija011.dmg	408
wija011.tar.gz	321
wija011x-logos0yy.{exe|zip}	412
wija011src.tar.gz	204
Total	2,387

Figure 9: Successful downloads of *wija* version 0.11 (∼ Jan/20/2007)

Fig. 9 shows the number of successful downloads of *wija* version 0.11 for different platforms.

After a little over one year from its release, *wija* version 0.11 had 2,387 successful downloads. 44% (.exe + .zip) of them were for Windows platform, followed by 17% each for distribution of LOGOS and for Mac OS X. About 9% were downloads of the source code.

Choice of Java as development language did not seem to affect the user distribution as much as we feared.

6.2.2 XMPP Server Records

Some information can be collected from the XMPP server we have been operating.

The number of XMPP users at media-art-online.org, which we believe can approximate the number of *wija* users, is 335 as of January 22, 2007.

6.2.3 P2P Update Log

The number from the XMPP server can be misleading because users can easily start and stop using *wija*. Substantial number of users recorded on the server may not be using *wija* currently.

To obtain more accurate measurement, we have been investigating a way to use the logging of secure P2P update. Soon after the release of *wija* version 0.12 on January 3, 2007, we have built a minor update called version 0.12a, and have been distributing the version only through the secure P2P update.

We discovered that 47 users have successfully downloaded and used version 0.12a before another minor update on April 26, 2007. We believe that it approximates the size of the user cluster of *wija* around us.

6.3 Evaluation

6.3.1 Evaluation of Challenges

Our view is that although challenges remain in many ways, our efforts so far have been successful as follows:

C-1: Pathway to P2P

In many development items, we have successfully eliminated necessity of XMPP servers supporting certain features.

Elimination of necessity of web servers is beginning. Distribution of the software is partially done in a P2P way.

C-2: User Friendliness

In large part, this remains to be a challenge.

However, GnuPG integration seems to be appreciated by users; to our best knowledge, *wija* is the only software written in Java (thus naturally cross-platform) that provides GUI frontend for GnuPG. Some novel approaches to enhance usability have been experimented such as direct end-to-end exchange of public keys and audio/visual representation of fingerprints. As a casual observation, many non-technical users are signing their presences and using encrypted messages, which seems to be indicating that the design has been successful.

6.3.2 Evaluation of Missions

Our view is that except the above remaining challenges, our missions have been complete as described below.

M-1: Messaging Platform

wija is a platform that runs on multiple OS platforms. It is internationalized in such a way that every message to user is provided in both English and Japanese, and it provides means for incorporating additional languages. It has been helping innovations as a tool for researchers and students pursuing their new applications of the Internet.

M-2: End-to-Endness

Most features, except vCard sharing (including avatars) that is too popular among Jabber/XMPP clients to design otherwise, do not require specific functionality of XMPP servers.

We have also eliminated the need for public key servers.

M-3: Security

wija is integrated with GnuPG, and provides means for signing and encrypting messages, as well as safely updating the software by downloading them from peers.

M-4: Freedom

wija has been distributed from our site under GNU GPL version 3, and third-party sites are also redistributing *wija*.

Secure P2P update uses the system's public key to verify the new software, which might sound as if we are the only ones who can develop *wija* and external plug-ins. But the key is stored within the original executable file a user trusted to download by other means. Therefore, this does not restrict freedom of developers distributing modified versions of *wija* with their own system's public keys, and allowing users to maintain the line of software by the secure P2P update.

7 Related Work

There are a lot of messaging clients which support Jabber/XMPP, but we only refer to Psi[21] and Adium X[1], for their cross-platform support and extensibility, respectively.

There are surprisingly small number, if not none at all, of projects that make use of extensibility of Jabber/XMPP protocols other than *wija*.

7.1 Psi

Psi is a popular Jabber/XMPP messaging client which supports multiple platforms. Although Psi is based on the same set of protocols, the policy is quite different from that of *wija*, as quoted below.

"The goal of the Psi project is to create a powerful, yet easy-to-use Jabber/XMPP client that tries to strictly adhere to the XMPP drafts and Jabber JEPs[3]. This means that in most cases, Psi will not implement a feature unless there is an accepted standard for it in the Jabber community."

7.2 Adium X

Adium X is a popular messaging client for Mac OS X that supports multiple protocols through use of libgaim library.

Like *wija*, Adium X has its plug-in architecture, and many of the essential features of the software are provided by plug-ins.

To our best knowledge, Adium X is not known as a platform for extension of what we can do over the Internet, except Gizmo, an IP telephony plug-in.

[3]XEPs were formerly known as JEPs, standing for Jabber Extension Proposals.

8 Future Work

We intend to further brush up the design of *wija* so that it will be more usable and graphically pleasing. We also intend to support more protocols (via gateways). We believe that we need those improvements to attract more attentions from researchers and general public, so that *wija* will become a more valuable tool for research and for living.

Meanwhile, we intend to develop further on the P2P aspects of *wija*. We intend to put more efforts on Overlay GHC as a tool for research on declarative programming of overlay networking. The insights we will acquire from the experience will be used upon the design of *wija* to replace XMPP with P2P protocols (we will probably make those protocols switchable).

9 Conclusions

We have developed *wija*, an open and extensible messaging platform on the Internet to assure freedom of experimentation for researchers, software developers and anyone in pursuit of problems, in the form of a Jabber/XMPP client.

Although this work is still in progress, we believe that we have achieved many aspects of the original mission statement. In particular, we have avoided dependencies on servers in many scenes of our development; public key exchange, proxy discovery, and so on, not to mention the level of ease we have achieved in handling public key cryptography.

In the hope that this software project will be an asset for advancement of software science and networking, we will continue our development.

References

[1] Adium team: Adium. Hypertext document. Available electronically at http://www.adiumx.com/.

[2] AOL LLC: Instant Messenger - AIM, as of 2007. Available electronically at http://www.aim.com/.

[3] Apache Ant Project: Apache Ant. Hypertext document. Available electronically at http://ant.apache.org/.

[4] Apple Computer, Inc.: Apple - Mac OS X, as of 2007. Available electronically at http://www.apple.com/macosx/.

[5] Berners-Lee, T., Fielding, R. T., and Nielsen, H. F.: *Hypertext Transfer Protocol – HTTP/1.0*, May 1996. RFC 1945.

[6] Bray, T., Paoli, J., C.M.Sperberg-McQueen, and Maler, E.: *Extensible Markup Language (XML) 1.0 (Second Edition)*, October 2000. W3C Recommendation. Available electronically at http://www.w3.org/TR/REC-xml.

[7] Callas, J., Donnerhacke, L., Finney, H., and Thayer, R.: *OpenPGP Message Format*, November 1998. RFC 2440.

[8] Dierks, T. and Allen, C.: *The TLS Protocol Version 1.0*, January 1999. RFC 2246.

[9] eMercury, Inc.: Social Networking Site [mixi], since 1995. Available electronically at http://mixi.jp/ *(in Japanese)*.

[10] Fielding, R. T., Gettys, J., Mogul, J. C., Nielsen, H. F., Masinter, L., Leach, P. J., and Berners-Lee, T.: *Hypertext Transfer Protocol – HTTP/1.1*, June 1999. RFC 2616.

[11] Free Software Foundation, Inc.: GNU General Public License, 1991. Hypertext document. Available electronically at http://www.gnu.org/licenses/ gpl.html.

[12] Free Software Foundation, Inc.: GCJ: The GNU Compiler for Java - GNU Project, as of 2007. Available electronically at http://gcc.gnu.org/java/.

[13] Free Software Foundation, Inc.: GNU Classpath - GNU Project, as of 2007. Available electronically at http://www.gnu.org/software/classpath/.

[14] Google, Inc.: Google Talk, as of 2007. Available electronically at http://www.google.com/talk/.

[15] GREE: Home - GREE, since 2004. Available electronically at http://gree.jp/ *(in Japanese)*.

[16] Hildebrand, J., Millard, P., Eatmon, R., and Saint-Andre, P.: *XEP-0030: Service Discovery*, March 2005.

[17] Hildebrand, J. and Saint-Andre, P.: *XEP-0115: Entity Capabilities*, October 2004.

[18] Housley, R., Ford, W., Polk, T., and Solo, D.: *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, January 1999. RFC 2459.

[19] Jabber Software Foundation: Jabber: Open Instant Messaging and a Whole Lot More, Powered by XMPP, since 1999. Available electronically at http://www.jabber.org/.

[20] Karneges, J.: *XEP-0047: In-Band Bytestreams (IBB)*, December 2003.

[21] Karneges, J. and Psi Team: Psi Jabber Client. Hypertext document. Available electronically at http://psi-im.org/.

[22] Leech, M.: *SOCKS Protocol Version 5*, March 1996. RFC 1928.

[23] Linux Online, Inc.: The Linux Home Page at Linux Online, as of 2007. Available electronically at http://www.linux.org/.

[24] Media Art Online: i-WAT. Hypertext document. Available electronically at http://www.media-art-online.org/iwat/.

[25] Microsoft Corporation: Microsoft Windows Family Home Page, as of 2007. Available electronically at http://www.microsoft.com/windows/.

[26] Microsoft Corporation: MSN.com, as of 2007. Available electronically at http://www.msn.com/.

[27] Millard, P.: *XEP-0020: Feature Negotiation*, May 2004.

[28] Muldowney, T.: *XEP-0027: Current Jabber OpenPGP Usage*, March 2004.

[29] Muldowney, T., Miller, M., and Eatmon, R.: *XEP-0095: Stream Initiation*, April 2004.

[30] Muldowney, T., Miller, M., and Eatmon, R.: *XEP-0096: File Transfer*, April 2004.

[31] Norris, R. and Saint-Andre, P.: *XEP-0086: Error Condition Mappings*, February 2004.

[32] OMG: *Unified Modeling Language (UML), version 2.1.1*, February 2007. OMG Specification. Available electronically at http://www.omg.org/ technology/documents/formal/uml.htm.

[33] Postel, J. B.: *Simple Mail Transfer Protocol*, August 1982. RFC 821.

[34] Saint-Andre, P.: *XEP-0082: Jabber Date and Time Profiles*, May 2003.

[35] Saint-Andre, P.: *XEP-0045: Multi-User Chat*, September 2005.

[36] Saint-Andre, P.: *XEP-0153: vCard-Based Avatars*, August 2006.

[37] Saito, K.: Examining the Charms of Jabber, an Extensible Instant Messaging Protocol, *JavaWorld*, (2006). *in Japanese.*

[38] Saito, K.: *i-WAT: The Internet WAT System – An Architecture for Maintaining Trust and Facilitating Peer-to-Peer Barter Relationships –*, PhD Thesis, Graduate School of Media and Governance, Keio University, February 2006.

[39] Saito, K.: Local Production, Local Consumption P2P Network, *UNIX magazine*, (2006). *in Japanese.*

[40] Skype Limited: Skype - internet calls, as of 2007. Available electronically at http://www.skype.com/.

[41] Smith, D., Miller, M., and Saint-Andre, P.: *XEP-0065: SOCKS5 Bytestreams*, November 2004.

[42] Software, P.: Perforce Software – The Fast Software Configuration Management System, 1996, 2005. Hypertext document. Available electronically at http://www.perforce.com/.

[43] Sun Microsystems, Inc.: Java Technology, as of 2007. Available electronically at http://java.sun.com/.

[44] The Free Software Foundation: The GNU Privacy Guard. Hypertext document. Available electronically at http://www.gnupg.org/.

[45] The Free Software Foundation: The GNU Privacy Handbook. Available electronically at http://www.gnupg.org/.

[46] The Internet Engineering Task Force: IETF Home Page. Available electronically at
http://www.ietf.org/.

[47] WIDE Project IDEON Working Group: wija Project. Hypertext document. Available electronically at http://member.wide.ad.jp/wg/ideon/?en%2F Projects%2Fwija.

[48] Wikipedia: List of Jabber clients - Wikipedia, the free encyclopedia, as of 2007. Available electronically at http://en.wikipedia.org/wiki/ List_of_Jabber_clients.

[49] Yahoo! Inc.: Yahoo! Messenger, as of 2007. Available electronically at http://messenger.yahoo.com/.

Copyright Notice