"Two things form the bedrock of any open society - freedom of expression and rule of law. If you don't have those things, you don't have a free country." - Sir Salman Rushdie

Freedom of expression, especially when inspired by innocent intention, is the defining force behind open societies. This powerful force now has new wings thanks to social media. Social media grants voice to the unheard all around this ever turbulent little planet of ours and we are still only beginning to see what it can do - be it in Kiev, Moscow, Ferguson, Baltimore, Tunis, Cairo, Istanbul, Riyadh, Tehran or Gaza. You would be fooling yourself if you do not think those in the legacy halls of power would not love to suppress, subvert and abuse this grassroots force. You need not look much farther to find evidence of this than a gritty documentary named Citizenfour and a meek young man named Edward Snowden.

One of the many things that pose a serious risk to the openness of social media is its thus far deeply proprietary nature. It should be socking to anyone that there are very few or no open technical standards in the social media space. There are some good reasons for this such as industry immaturity and a very rapid pace of innovation. There are also some not very good reasons such as cutthroat competition to monetize social media and a resulting lack of interest in mutual cooperation towards openness.

Hopefully things will change as the social media industry stabilizes a bit more and is more ready for standardization. In the meanwhile community projects like Agorava have a very important part to play. In absence of any real open standards,   
Agorava can provide some level of interoperability through a common interface for the very divergent social media APIs of today - at least for Java and Java EE developers. The project is also very significant to the Java EE ecosystem on its own right. It demonstrates first-hand how CDI extensions can effectively build upon the stable core of the Java EE platform where needed while still adhering as closely as possible to the principles of interoperability and vendor-neutrality.

This book is critically important to the project. It provides great background of the social media industry, explains the core concepts important for Java social media developers, describes the so far feeble efforts towards standardization, covers the other Java social media projects, shares the motivation for the project as well as of course providing a technical overview and deep dive of the Agorava API. Given the volatility, depth and complexity of the topic, this book was no doubt a very tough one to write.

To boot I can’t think of a better person than Werner to write the book. He has demonstrated his deep personal commitment to openness, standardization, Java and Java EE over many years as well as his clear command of the social media industry and Agorava.

I hope the project and the book will prove invaluable to any Java social media developer.

Reza Rahman

Founder Java EE Guardians   
Former Java EE Evangelist @ Oracle

# Introduction

#### Ancient History

Most people assume social media to be a modern-day invention of companies like Yahoo!, Google, MySpace, Facebook and Twitter. In actual fact, social media has a history that reaches back over thousands of years.

To some the postal service was one of the earliest forms of social media. From the days of the ancient Persian postal service to the Greek Agora [1] and Roman Forum [2], the postal service served as a medium for social and political discussion. The first recorded equivalents to Wiki or “Forum posts” were literally carved into the walls of Roman Forums or similar buildings [3] Recording events like the destruction of Pompeii by Mt. Vesuvius or several parts of Rome under Nero.

The Ancient Agora of Athens acted as incubator to democracy being a favorite venue for social and political debate while doing business in ancient Greece. Until today, this worked in similar ways while the forums often turned digital. From Tunisia to Ukraine opposition movements used social media to bring democracy to places where it’s often been surprised and uncommon for a long time. [4]

#### Modern History

In 1975, shortly before the American military pulled out of Vietnam, the libertarian thinker Samuel Edward Konkin III introduced a philosophy called Agorism, a social philosophy of counter-economics and the ideas associated with that practice [5]. In a market anarchist society, law and security would be provided by market actors instead of political institutions. Agorists recognise that situation cannot develop through political reform. Instead, it shall arise as a result of market processes. Aspects of its social values can be found today in the Occupy Movement, its primary goal being to make the economic and political relations in all societies less vertically hierarchical and more flatly distributed. Or in protest parties like Beppe Grillo in Italy, Syriza in Greece (where it even took over government), Podemus in Spain or the Pirate Parties. Agorism also put forth the idea of using different currencies, something modern social media and related online communities are adopting with digital currencies like Bitcoin.

In 1966, the email was invented, followed by the arrival of CompuServe in 1969. CompuServe launched its email service in the late 80s and expanded into early 90s, before finally dissolving original CompuServe in 2009 after several takeovers.

Usenet started up in 1979 and provided its users with forums and newsgroups. Surprisingly, most of their functions have changed very little and still exist today.

Internet Relay Chat (IRC) began in 1988 and was widely used across academic networks before emerging into what is now the Internet. IRC remains quite active today, although it is largely overshadowed by other short messaging services like Twitter. There are still around 3200 IRC servers around the world with hundreds of thousands of channels available.

Based on the notion of Six Degrees of Separation [6], the first modern social network, sixdegrees.com, was founded in 1997. It allowed members to link to other members as friends, write messages or post on bulletin boards. After being sold on to another company, it was shut down in 2001 with roughly around one million users. When services like Amazon, Yahoo! and 1998 Google first came into existence, they were not considered social networks.

In 1995, Microsoft picked up on the emerging trends by launching the Microsoft Network (MSN). MSN became known as a ‘portal’, another word for personalised online community. It wasn’t surprising when many portal products, servers and vendors jumped the social bandwagon a few years later.

Both Microsoft and Yahoo! had fallen behind some of social media’s new kids on the block and beside a rather brilliant and fast move with YouTube, most other aspects of social media were also a bit underestimated by Google, until it returned via Google+ and is certainly here to stay even if it may not lead every aspect of it. Yahoo! also offers plenty of social features and has since its recent recovery taken over smaller vendors in the social networking space like Flickr or Tumblr. 1999, Brad Fitzpatrick, now also working at Google founded LiveJournal.

At the start of the 2000s, MySpace and LinkedIn (and their somewhat different standing today ten years later) to the middle of that decade, when current giants Twitter or Facebook were founded, or already growing Google acquired YouTube. In the last few years, we have seen several similar take-overs, most notably Facebook buying up Instagram for close to a billion dollars or its gigantic takeover bid for mobile messaging provider WhatsApp.

Social networks have always existed, since people first started to communicate. Technical means changed over time, and are likely to change even faster in the near future. Some services will be merged; others might become irrelevant and vanish sooner or later. Using vendor-neutral standards and frameworks, that won’t tie you into the offerings of just a single provider is crucial, unless you want to spend more time on rapidly changing APIs than actually using it for something productive. Or worse, having to switch over to another provider, if proprietary API and framework offered by a particular vendor is no longer available. While not a Social Network in the sense of Twitter of Facebook, the recently closed Bitcoin trading site Mt.Gox [7] was a good example for such dilemma. It used to be market leader in this area for a long time, thus applications and e-commerce solutions offering Bitcoin support may have used its API directly, now forced to throw those apps away or rewrite them.

If a vendor-neutral abstraction is created on top of such API, then it is much easier to cope with problems and changes like those caused by a vendor going out of business or being taken over by others.

This is where Agorava comes into play.

1. Standardisation

## Early Approaches

The theories which we encountered in the introduction, such as Six Degrees of Separation [6], influenced later concepts like Friend of a Friend (FOAF) [8]. Aspect of both can be found in most “Friend” mechanisms used by social networks today.

Rich Site Summary (RSS) [9] is an early example of a digital form of social media. Among the developers involved in the design and standardisation of RSS was Aaron Swartz, who before his untimely death in early 2013, had worked on a book about *A Programmable Web* [10] a rather down-to earth, street-smart counterpart to the typically dry academic papers one finds in this field.

Portable Contacts was created by Plaxo, a Social Address book service founded in 2002 by a group of entrepreneurs around Sean Parker. Parker had legal troubles one wouldn’t call so different from Aaron Swartz after his Napster music exchange portal was forced out of business, but fortunately talking for more than just one Social Media household name of today he coped with it differently. The idea of a portable address book or set of “Digital Business Cards” was well-intended and makes sense especially from a user’s point of view, but commercial interests of competing players each of them greedy to share date even with those who own it created many roadblocks to an otherwise good idea. Plaxo while still in business after all those years was not necessarily his greatest hit and investors kicked him out in ways not so different from what had happened to co-founder Eduardo Severin at Facebook when Parker got involved. Financially this made him rich and put muscle behind other services, e.g. Spotify. Little coincidence, it seems quite a bit like Napster, but in a more commercially more accepted way, also thanks to his fame and fortune after Facebook. Portable Contacts may have been a significant part of the social media movement but are no longer used. This is because many of the API’s used within larger companies are no longer compatible with Portable Contacts. The only remainder still in use is vCard, used by many companies and applications.

## Mashup

As the Java language played a more and more important role first on the desktop (mostly via Java Applets in the early days, or AWT/Swing standalone applications) then around the turn of the century also on the server, Sun and other companies (of those at least HP and IBM are still active, plus Sun now part of Oracle) began to standardise parts of the Java Platform in the so called Java Community Process (founded 1998) or JCP.

After being turned down or withdrawn twice, a notable standards for Social Networks, then often called “Portal” (see MSN earlier) was JSR 168 the Java Portlet Specification [11] In an ideal world it allowed portal applications to be deployed into multiple portal servers regardless of the vendor and together with other standards, especially the OASIS-defined WSRP (Web Services for Remote Portlets) raised hopes for interoperability between these applications even across different portals. After 10 years, with other technologies and languages, especially “light weight containers” on the rise, the Java Portlet standard is currently about to get a version 3.0 update (JSR 362) Also planning to incorporate some of the later Social standards where possible.

While Portlets and related standards are still mostly the domain of large scale Enterprise Application servers inspired by music or video compilations or remixes a more language and platform neutral approach called Mashup [12] started shortly after the first Portlet standard. Whether it was Yahoo Pipes, Microsoft Popfly, Google Mashup Editor, IBM Mashup Center or BEA AquaLogic [13] all of them promised interoperability and personalised widgets, with an experience similar to iGoogle. A service, which mostly due to Google+ taking personalisation and a more “Integrated Social Media” approach to the next level, was shut down on November 1 2013.

So was RSS, only a few months after Aaron Swartz’ death. Google Reader was discontinued on July 1 2013. While Swartz as last documented his unfinished book [10] advocated an open API based programmable web, most of the big providers have different plans, as numerous articles around Google Reader’s end like this one [14] express.

Another social standard has been affected by this development almost since its inception: OpenSocial [15] Developed by Google with a handful initial supporters, MySpace still the most popular social network at the time, Ning or Plaxo. At this time Portable Contacts mentioned earlier would also fit into the overall idea of easy to use Widgets and Services across a variety of social networking providers. As opposed to e.g. Version 2.0 of the Java Portlet standard which released its Public Draft around the same time.

While seemingly easier to use, so were exploits and security issues, often demonstrated by “script kiddies” more or less out of primary school using a few scripts to break or compromise those services and widgets. Full scale Java Enterprise servers and Portlets were and still are much harder to break into, if system administrators do their homework. And all the fuzz about Java Security or it being used to transport malware usually affects the client side.

Exactly where a vast majority of OpenSocial JavaScript code runs making it more exposed than a mostly server-side or carefully mixed use of server (Java EE or other environments) and client tier (JavaScript, etc.) See Figure 1 – N-tier Enterprise Application for different tiers of a typical web-based Enterprise Application. Java EE, but you may replace it by other languages like Python, PHP, Ruby or Scala with the layers being about the same.

While Google began to use OpenSocial in its **Orkut** service (popular in Latin America and a few other countries like India, but never really picked up in other places) other vendors initially planning to support it abandoned those plans one after the other. The likes of LinkedIn or XING especially did so for security reasons. XING as we’ll hear more on Security and Privacy about the latter, too. Neither Facebook nor Twitter, the two dominating providers from the second half of the last decade ever cared about OpenSocial. And more importantly, Google killed its own creation when it became obvious. Google+ would never support it either. A few vendors mainly from the server and large Enterprise fraction like IBM, SAP or Confluence author Atlassian still back OpenSocial.

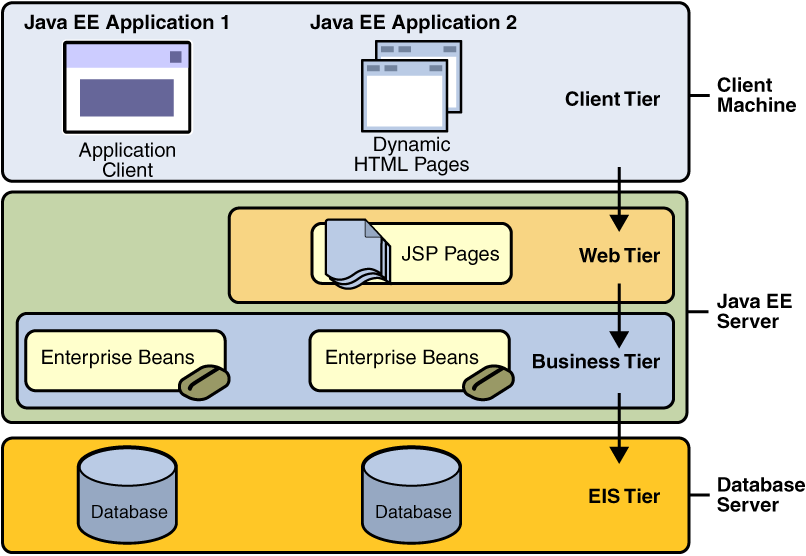


Figure 1 – N-tier Enterprise Application

Dachis Group, recently acquired by Sprinklr, sort of a “Think Tank” for Social Media and Enterprise Data Analysis is also strongly involved, though its focus goes more in directions like Big Data and NoSQL systems lately, and it openly admits like most remaining OpenSocial backers (or ThoughtWorks in its regular survey called “Radar” about technologies to watch or regard less over time) the only place for it may be the Enterprise and Intranet after all. Big Data can be seen as a Mashup, too, but it takes place primarily on the EIS or DB tier (see Figure 1) so I won’t discuss it much here. These technologies would also deserve their own book and exceed the scope of this one ;-)

Instead of a seamless Mashup of e.g. Google, Facebook, Twitter, LinkedIn, Microsoft or Yahoo! the only value of OpenSocial is now connecting HR, Sales or Accounting with each other and provide the management with some nifty “Social” Mashup of all departments and subsidiaries of a large enterprise. Making it practically useless to those who wish to see let’s say information from their Twitter feed next to the Facebook timeline. The purpose, OpenSocial was originally created for.

The latest twist in the OpenSocial story is an effort to create a W3C standard based on OpenSocial and similar concepts. What has been a hard and thorny path in the case of the Semantic Web seems almost impossible here; especially given a majority of players prefer their “walled garden” as we found earlier [14]

While W3C defined an interesting standard called semantic web, which sounds synergetic to some of OpenSocial’s ideas, semantic web is so far used only in a few isolated sectors like biology or healthcare.

Work at W3C happens at a rather slow pace, similar to (or even slower than) HTML5, for example, which started in 2008. Today it is still in candidate state (despite many vendors already proposing solutions and products being based on it). A final recommendation is expected by the end of 2014 with further updates and improvements to come two years after that, and so on. That will make it well over a decade before something gets standardised this way.

Types of Standards

Aside from their lobbying of “Basic Standards” like OpenSocial [15], Figure 2 provides a decent overview of Social Standards by Dachis Group from a series of articles [16]



Figure 2 – What's missing in Social Business Standards?

Two main categories are

1. **Higher-Order Standards**: Behaviour and Domain-Driven business oriented standards and APIs. Most of them not so clearly fleshed out by the authors of [16]
2. **Basic Standards**: Some were mentioned before, here grouped into 4 sub-categories   
    a. Contacts  
    b. Activity/Messaging  
    c. Integration/Interoperability  
    d. Security

Then you may look at the way, Social standards, frameworks and libraries communicate. By either

* Consuming information
* Analysing information
* Transforming information
* Providing information

A typical example for consuming information would be showing your status in your favourite social networks inside another application, e.g. your blog or corporate website.

Another one is if you include a map service to show directions to an event you host or participate in.

Analysing information involves some sort of calculation, ranking or benchmarking. Often consuming more than one source of information, so you can get a better picture. An example of such a service would be Klout. Where activities across multiple social networks influence your overall ranking.

Transforming information can be done either by merging and combining several data streams, a simple sum of two numbers, or if e.g. your travel booking site predicts the cost of a trip in your local currency based on conversion rates. Translation services like Google Translate would be another, more sophisticated example of transforming information.

Last but not least providing information is any service or sometimes just content your application makes available to others. Ideally via an API allowing others to consume, analyse or transform it based on their needs.

Another way of looking at Enterprise Social Networks, related standards or products according to IDC [Figure 3] would be

* Stand-a-Lone Products or services with social aspect that can be installed and used on their own. Though by the nature some, e.g. those embracing OpenSocial aim to be interoperable with others, ideally across multiple vendors
* Embedded Social facets or features within a larger product or suite. A classical examples would be Oracle Fusion Middleware, containing social functionality in the footsteps of BEA AquaLogic [13] or several other Oracle acquisitions [23] in the Social sector since then
* Speciality often Domain-specific Social Networks, e.g. for Healthcare, Fashion, Pets or any other area of interest to a significant community. You’ll hear about a range of these later in **4 Enterprise Social**.

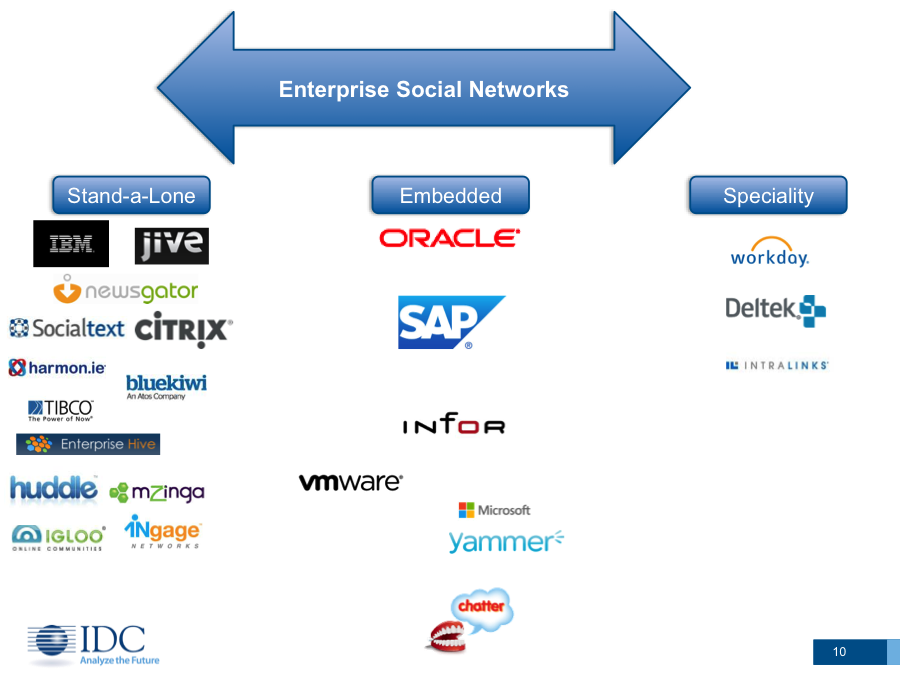


Figure 3 – Categories of Enterprise Social Networks

**OPENi** (a Social Media research group funded by the European Union) [17] tries to group social standards and APIs in a somewhat context-related manner:

* Activity API referring to a social, health, behavioural activity log as reflected in multiple cloud-based services ranging from Social, Photo and Video Sharing to Health and Location-based Services. In essence, it includes all the social and personal activities of a user, and is related with the logging activity of a device. Relevant Categories: Gaming, Health. Location-based Services, Music, Photo, Shopping, Social, Video
* Advertising and Analytics API Enabling the collection, aggregation and analysis of end-user/customer’s needs, interests and preferences based on
  + Their interaction with advertising/marketing content
  + Application behavioural data
  + Social network interactions towards enabling personalised advertising services with enhanced end user added value. Relevant Categories: Analytics, Advertising
* Location API Enabling location awareness through Check-ins, Direction, Events, Reviews and Tips. It is a strong contextual API with location, which can be extracted by a GPS sensor and can be mapped on a map. Relevant Categories: Mapping, Location-based Services, Social, Music
* Media API Bringing together photo, music and video sharing services with file transfer and syncing functionalities. It is related with the Gallery Application and the file system of a device. Relevant Categories: File Transfer & Syncing, Location-based Services, Music, News, Photo, Social, Video
* Products & Services API Embracing Payments and Shopping services. This special case of commercialization-relevant objects, requires a strong API with transactional capabilities, and enhanced security. It is related with the store application of a Phone. Relevant Categories: Location-based Services, Payments, Shopping.
* Profiles API Extrapolating information about people based on Analytics, Advertising, Contacts, Gaming, Health, Location-based Services, Messaging & Chat, Music, Social, etc. services. Typically, people can be represented in various ways, with multiple profiles. This API brings all that information together: avatars, profiles, contacts and accounts, interrelating different profiles, in different services. It is directly related to the Agenda application of a smartphone. Some relevant Categories: Analytics, Advertising, Contacts, Gaming, Health, Location-based Services, Messaging & Chat, Music, Film…
* Search API In order to find and retrieve information from the cloud-based services. It brings together existing search engines with distributed search functionalities among various platforms, and is related to the search functionality found in a desktop or phone OS.

While OPENi has not made any of their API available on GitHub (as their site promised in 2013 ;-) its work caught attention of e.g. W3C where OpenSocial [15] is now further discussed. So if not APIs of its own, OPENi and its concepts may help there. We’ll highlight OpenSocial in more detail when looking at **1.4.2 Apache Shindig**.

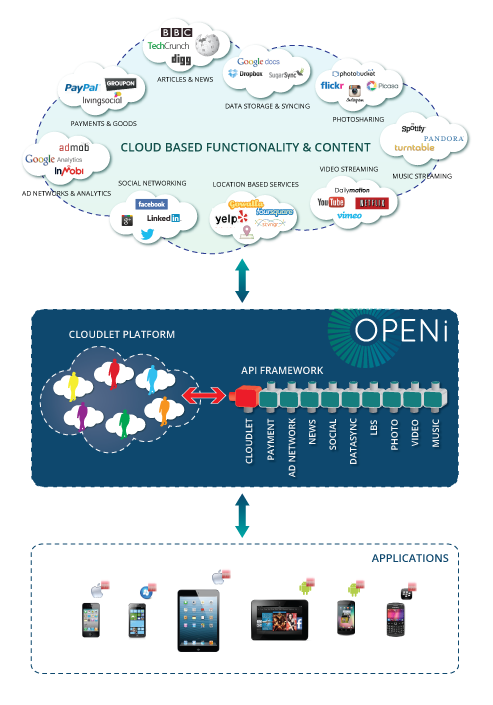


Figure 4 - OPENi concept

Apart from dissecting and categorising Social Media standards as above, these are the 3 most important things APIs and related standards have in common:

* Practically all Social Media services use REST as transmission protocol
* Most of them transmit data in JSON format, some in XML or both
* Identification & Authentication are almost always based on the OAuth protocol

You’ll notice “trivial” basic standards like REST, XML or JSON missing from Figure 2 as the authors of [16] believe they must be common knowledge by now. Along those lines I’ll also consider the reader familiar enough with them.

We’ll take a closer look at Security, Authorization, Authentication or Digital Identity in chapter 2, as well as aspects of Privacy or Data Protection. First let’s have a look at relevant Java Social Media APIs and Frameworks before and leading to Agorava.

Java Social

Following Java standards like Portlet API or various parts of Java Enterprise there have been several Java frameworks and libraries for Social Media.

* + 1. Twitter4J

Twitter4J is an unofficial Java library for the Twitter API.  
With Twitter4J, you can easily integrate Twitter calls in your Java application.

Its author, Yusuke Yamamoto used to work for Twitter in 2012.   
While he did, he was briefly meant to represent Twitter in the Social JSR Expert Group. More on that later…

It’s lightweight, mostly backward-compatible with Java 1.4, making it also easy to integrate into Mobile apps for instance. Not very surprising more or less official clones and forks or projects inspired by it are available anywhere from J2ME to Android. Twitter4J is an API Binding Library, thus primarily for consuming information. While extensions or your application could analyse or transform it, the framework itself doesn’t provide support beyond that of Twitter itself. Based on Hinchcliffe’s categorization Twitter4J could mostly be seen as a Basic API, but with aspects of Higher-Order.

At the time of writing, the official Twitter4J website (twitter4j.org) was temporarily unable to service requests due to maintenance downtime or capacity problems. Whether or not that is a signal for lack of maintenance by the author and contributors, I can’t say. I therefore point to the GitHub project page [18] which as long as e.g. Twitter itself shouldn’t contest it or force GitHub to turn it down, is likely to remain. The last update was about a month ago, some other modules had been updated within the past 3-8 months, so it may not see high activity, but seems well-maintained at least.

* + 1. Apache Shindig

Apache Shindig [19] was started relatively soon after the OpenSocial [15] efforts began. Mainly driven by Google and its Orkut at first for obvious reasons, other contributors especially IBM joined later. Due to the refocus of OpenSocial towards a “Social Intranet” none of the sites or software claimed to use it are sure to still do so. In fact, Google Partuza points to a non-existing page, hence it was shut-down like other examples we heard of earlier.

LinkedIn and most other social media providers have gradually or completely abandoned OpenSocial from its public sites, too.

An interesting approach by Google to develop OpenSocial applications inside a WYSIWYG environment of Eclipse IDE was OpenSocial Development Environment (OSDE) see [20]. It is safe to assume some was taken over by IBM and integrated into Social editions of Eclipse based development tools like RAD, etc. while the Google Code project was last updated in June 2010.

OpenSocial API specifications pointed by Apache Shindig are no longer hosted even by OpenSocial supporter Atlassian it seems, the link points to an error page. Whether or not this means Atlassian also doesn’t believe in OpenSocial and won’t use it in newer versions of its products or it’s simply a process of reviving the whole OpenSocial movement on W3C remains to be seen. In the meantime all versions of OpenSocial, especially 1.0 or 2.0 have been failures and sites like Orkut at most used the 0.7 or 0.8 versions.

According to OpenSocial supporting Dachis Group [16] Shindig as manifestation represents a Basic standard (2c). It has means of both providing and consuming information.

On September 30th, 2014 Google has **shut down** Orkut**,** making Google+ the only remaining Google reference.

Of course, Google had already stopped using Shindig in favour of its own Google Code projects. Most of them were also closed down and either point to the OpenSocial page now, or no longer exist. A little over a year later, on October 23, 2015 Apache Foundation has **retired** the Shindig project. It remains to be seen, if OpenSocial will follow its fate, but having been the de facto Reference Implementation of OpenSocial, it certainly looks like another big nail to its coffin.

* + 1. Efforts by Sun/Oracle

Around 2008, Enterprise Social Mashup driver BEA Systems [13] had just been acquired by Oracle, Java creator Sun Microsystems did not quite guess a similar fate was also around the corner for it. Based on OpenSocial, Sun created a fork or derived project from Apache Shindig named SocialSite [21].

SocialSite still counts 231 members on Java.net, but the project is inactive. One of the last commits was by Kohsuke Kawaguchi, Sun’s creator of Hudson CI server and later Jenkins after he was hired by CloudBees.

Following the acquisition of RightNow Technologies (see [23] end of 2011) maker of SaaS CRM software with Social aspects – a move clearly targeting the likes of Salesforce.com or SAP, some of the former RightNow teams and products were incorporated into Oracle’s Social Enabled Enterprise department.

JavaOne 2012 featured a Social Day I also attended with speakers like Facebook’s Developer Relations Manager James Pearce. Oracle itself presented a “one man” project written by freelance contractor still for RightNow under the codename “Project Sneaker” which just after JavaOne 2012 was published as Oracle SocialLink on Java.net [24]

It never attracted followers outside Oracle and must be considered more dead than even SocialSite now.

* + 1. DaliCore

After Oracle took over Sun, the abandoned SocialSite found its best equivalent and successor in DaliCore [22].

It was originally based on a CMS made by a Belgian start-up, mainly around BeJUG (DevoXX conferences) members like Johan Vos. DaliCMS (2006) was called a “Web 2.0 ready CMS”

Mostly an Open Source project with community features. 2008, around the same time Sun did with SocialSite, DaliCMS got integration with Shindig. Mostly JavaScript code from Shindig and Java code was used by DaliCMS.

Eventually it became more than a CMS, and the underlying modules turned into DaliCore. During his talk at JavaOne 2013 Johan unveiled the origin of the “Dali” part of the projects originally stood for “Dutch Amazon Lovers Initiative”. A tribute to Internet and Cloud pioneer Amazon.

DaliCore adds functionality common to user management, content or permissions on top of Java EE 6. With a strong focus on users and permissions. In about every project that uses DaliCore, users should be able to login with existing credentials (Facebook, Twitter, Google+, LinkedIn,...) So called Dali modules extend DaliCore. While certain connectivity exists to social providers like Facebook, these mainly serve as authentication providers to DaliCore SSO (Single Sign-On) functionality. Aside from that DaliCore aims at providing information.

Important aspects of DaliCore are:

* Mostly Social Container and CMS with partial Open Social features
* Java Persistence support via EJB, JPA or similar standards
* No clear separation of modules, especially API/Spec and Implementation, in most cases shares the same module and even package space. Making reuse and extension harder despite the attempt for that via Dali Module
* Currently seems a bit heavy, strong dependencies on the Full Java EE stack, i.e. using only its Web Profile seems hard at the moment
* GlassFish support and integration is positive where it is the preferred application server. Thanks to the Java EE standard it is not impossible, but could be harder to use with other containers
* It demonstrated scalability also under heavy usage with some customers, e.g. an e-Voting site in Belgium Unlike Apache Shindig or Sun SocialSite (which DaliCore seems like a natural successor in most cases) it is used by actual customers at least in Benelux countries as of now

After Oracle announced, it will no longer provide commercial support for GlassFish or enhanced features like Clustering (beyond PoC quality provided by a Java EE Reference Implementation, it’ll remain), Performance tuning, etc. a major argument for DaliCore vanished. Oracle recommends its WebLogic/Fusion Middleware products, or current GlassFish users will be forced to migrate to other servers if they need commercial support.

DaliCore author Johan Vos and his Belgian start-up claim, they’ll offer commercial support for their customers, maybe even try to extend that, but DaliCore has become a clear victim of this. Since early 2013 there has been no activity in the java.net repository, so in many cases, it may not work with changing API by leading social providers. Hosting a site may be less critical, but with the underlying GlassFish platform deprecated by Oracle on a commercial front, those who plan to host more than just a small, private Blog may prefer to look elsewhere.

While Johan Vos recently told us, he was still working on it, he also confirmed, this is now done in a **private** repository at BitBucket. Thus neither his statement can be verified that way, nor can DaliCore still be considered “Open Source”. At least the source is not accessible, and clients therefore at the mercy of Johan’s company. A risk that doesn’t exist for most other examples here, even if those that are only maintained by a single person or company. The website of Johan’s company, LodgON seems like it currently uses DaliCMS under the hood, but it does not mention DaliCore or suggests it could ever become Open Source again.

* + 1. Spring Social

SpringSource, recently rebranded by its parent companies as Pivotal has a long history of jumping on trends and providing libraries or extensions to its Spring Framework for just about everything it seems.

Spring Social [25] is a binding library for consuming information. Although integrating OAuth with Spring Security and other parts are symptomatic of a Basic API it shows more aspects of a Higher-Order API than most other solutions out there. It is not very actively supported and updates or blog posts by Pivotal staff appear 6 to 9 months or longer after a change was made or a feature announced at a conference. This is not the strongest signal to the community, but Spring Social can be considered reasonably mature. There is evidence of some commercial adoption in products or solutions, but vendors also worry about Pivotal/VMware could stop it in favor of other more strategic parts of their big portfolio.

Open Source ecosystems like GitHub show various extensions or plugins, but not all of them implement enough of a social network or API to be useful. Some are merely a PoC and most of them have not been actively maintained for some time either.

Important aspects of Spring Social are:

* It works primarily with Spring Framework or other parts of the Spring landscape for obvious reasons
* UI frameworks other than Spring MVC are relatively harder to integrate
* Official Spring Social connectors focus mostly on Facebook, so far examples only show Single Service approach. No straightforward support of multiple services or providers
* Despite an otherwise lively Spring community, even  
  inside Pivotal/VMware there’s doubt about support  
  and activity, especially after some people left both companies. This has been confirmed by top level architects who said priorities lie elsewhere, although there are no immediate plans to abandon it completely
* However, this bares the fact of one, two at most active committers and drivers behind Spring Social. Employed by a single company.

Spring Social currently supports .NET as well. I cannot say, if there is portability of any code, but those who need to support more than one platform may appreciate a familiar API for both Java and .NET. The fact, that Pivotal recently (Q1/2015) announced it’ll drop commercial support for even larger Open Source projects like the popular Groovy / Grails ecosystem speaks for itself. There is no evidence that Pivotal plans to invest its further resources in much smaller “fish” like Spring Social either.

* + 1. Seam Social

Seam Social was part of the JBoss Seam 3 project offering social connectivity on top of Java EE standards like CDI [28]. Seam Social is a binding library for consuming information. While OAuth on top of CDI and Java EE can be considered a Basic API, Seam Social was the richest Higher-Order API before Agorava. Especially by providing CDI events for Social Behavior such as Likes, Check-Ins or updates to a user’s Timeline. Common API bindings in most cases dictated by a provider’s API definition or sometimes directly generated from it were inspired by Spring Social, but Seam Social was never a clone or fork, as it stands to Spring Social like underlying CDI and extensions do to Spring Framework.

In early 2012, Seam was stopped to be merged with Apache DeltaSpike. Which was a reason for Seam Social to move on as well…becoming Agorava [26].

* + 1. JSR 357

Between JavaOne 2011 and further conferences like DevoXX in Antwerp I initiated a series of discussions with CDI Spec Lead Pete Muir, Twitter4J author Yusuke Yamamoto or Seam Social creator Antoine Sabot-Durand to explore a possible standardisation of Social Media APIs for the Java Platform in the Java Community Process.

They thought it was an idea worth pursuing. While other APIs and projects, like DaliCore were considered possibly relevant to (especially providing information under Java EE) the strong use of CDI and rather well-structured separation of API and implementation made Seam Social the best candidate for initial codebase.

The JSR was proposed by me and Antoine as co Spec Leads, joined by companies like Red Hat/JBoss, eXo Platform or Twitter (represented by Yusuke Yamamoto) as well as JUG Chennai, DaliCore authors from BeJUG or Java EE 7 EG Member and Oracle ACE director Markus Eisele (now working at Red Hat, like Antoine). A healthy mix of corporate, individuals and JUGs one might say.

However, after IBM first voted against the JSR other EC members like SAP followed, one JUG (SouJava) voted in favor, the other (LJC) voted against it, and the creation ballot ended 8 against 5 stopping the creation of JSR 357. Sadly, although what we heard earlier about big vendors preferring “walled gardens” [14] over open standards not entirely surprising, even interested party and EG member candidate Twitter abstained. Not against it, but with a difference of just 3 votes not voting either. See [26] for an article on the subject.

IBM feels strong about OpenSocial, but as even some of its other supporters like Adjuvi’s Dion Hinchcliffe [16] explained (a bit later though) OpenSocial is a Basic Standard while the main goal of JSR 357 following Seam Social was to offer a Higher-Order API and binding to the Java Platform. Take WebSockets, among the JSRs 357 mentioned having potential for synergies. The W3C standard just along the lines of where OpenSocial hopes to go those plans were successful was defined as part of the HTML5 umbrella, while the Oracle JSR creates a Java implementation. Even on the Java Platform there have been several existing, well-adopted Open Source projects for WebSockets, nevertheless it was found worth creating a JSR, too. Similar for JSON and maybe even more striking as the de-facto Reference Implementation by JSON “father” Douglas Crockford inspired the JSON JSR, but Crockford while initially “Supporting this JSR” told me personally at Dutch Mobile Conference last year, he no longer intended to be involved. And has never stopped supporting and promoting his own libraries around it.

The fact, that first opposing IBM sold Hundreds, maybe Thousands of Social Media Software Patents to Facebook the week JSR 357 was voted down doesn’t just present a bad optic. It also makes IBM’s own support of OpenSocial a bit questionable if Social Media patents and proprietary technologies are sold to Facebook bolstering its arsenal against competitors like Twitter and others. Could be a reason behind Twitter’s “mood swing” at the last moment of the ballot. Maybe a coincidence, but Twitter4J author Yusuke Yamamoto left Twitter soon after that. Whether or not EC Members like IBM or SAP didn’t want Red Hat in an influential role in more key Open Source projects after Hibernate, CDI/Seam or Arquillian just to name a few it could have been another reason, too. Especially on that one I’d say if it was among intentions, it backfired with the creation and active development of Agorava, not just by one company (like DaliCore or Social Link;-) but a variety of contributors around the world, most of them proposed EG Members.

Murdering the heir to the Java Social “throne”, especially with a significant French participation one may say created a “French Revolution” of Java Social Frameworks named Agorava.

1. Social Security

Authorisation

Authorization, also known as access control, is the process of managing access to resources. In other words, controlling who has access to what in an application.

More formally, to authorize is to define an access policy. Examples of authorization are: Is a user allowed to look at this webpage, read or edit this information, view this form, or use this printer? Those are all decisions determining what a user may access.

* + 1. Java Authorization

Java has a security architecture that protects systems from unauthorized access by mobile/dynamic or static code. The problem is in manually determining the set of security access rights required to execute a library or application. The commonly used strategy is to execute the code, note authorization failures, allocate additional access rights, and test again. This process iterates until the code successfully runs for the required use cases. Those cases may not always cover all paths through the code, so failures can occur in deployed systems. On the other hand, a broad set of access rights may be allocated to the code to prevent authorization failures from occurring. However, this often leads to a violation of the Least Privilege design principle [27].

The Least Privilege design principle requires a minimalistic approach to granting user access rights to specific information and tools. Additionally, access rights should be time based as to limit resources access bound to the time needed to complete necessary tasks. The implications of granting access beyond this scope will allow for unnecessary access and the potential for data to be updated out of the approved context. The assigning of access rights will limit system damaging attacks from users whether they are intentional or not.

Basic Java SE Security relies on **permissions**, e.g. FilePermission to access the file system or AWTPermission to use resources like the clipboard in AWT/Swing applications.

If someone wants to allow a new type of permission for specific applications, this can be done by adding an entry in a policy file.

An example of a policy file entry granting code from “http://mysocialtv.com/” permission to watch Sky Sports 3 would be:

grant codeBase "http://mytvstation.com/" {

permission com.sky.TVPermission "sky-sports-3", "watch";

}

* + 1. Java Enterprise Edition

Security in Java EE is based on some essential terms:

* User is an individual identity defined in the identity storage. The storage can be a database, flat file or LDAP server.
* Group is a set of users with the same needs or access levels.
* Security Realm is the access channel for the application server to user and group information.
* Role is a Java EE concept of access levels. A Java EE developer specifies which roles can access which set of application functionalities. These roles are then mapped to users and groups via vendor specific configuration data.
* Principal is an identity with known credentials which can be authenticated using an authentication protocol.
* Credential contains or references information used to authenticate a principal for Java EE applications. E.g. by entering a password, or calling external services. More in “2.2 Authentication”.

Agorava relies on CDI, a Java standard called “Context and Dependency Injection”, like several other parts of Java EE from version 6 on. We consider the reader somewhat familiar with the CDI standard, if you need to learn more about it, please see [28]

You’ll remember from chapter 1.3, that almost every social service or similar API uses RESTful Web Services. REST stands for “Representational State Transfer”. JAX-RS is the Java API for RESTful Web Services (currently version 2 or JSR 339)

Here a brief example, how authorization can be done by injecting a JAX-RS SecurityContext instance using a CDI @Context annotation. The injected security context works on top of lower level HttpServletRequest API. The injected security context depends on the actual JAX-RS application deployment.

SecurityContext can be used in conjunction with sub-resource locators to return different resources based on the specific roles a user principal is included in. For example, a sub-resource locator could return a different resource if a user is a “preferred customer”

@Path("basket")

public ShoppingBasketResource get(@Context SecurityContext sc) {

if (sc.isUserInRole("PreferredCustomer") {

return new PreferredCustomerShoppingBasketResource();

} else {

return new ShoppingBasketResource();

}

}

Using SecurityContext for a Resource Selection

@Path ("resource")

@Singleton

public static class MyResource {

// Jersey will inject proxy of Security Context

@Context

SecurityContext securityContext;

@GET

public String getUserPrincipal() {

return securityContext.getUserPrincipal().getName();

}

}

Injecting SecurityContext into a singleton resource

Where a JAX-RS application is deployed in a Servlet container you can rely only on the standard Java EE Web application security offered by the Servlet container and configurable via the web.xml descriptor.

By defining <security-constraint> elements in web.xml and assigning roles which are able to access these resources. You can also define HTTP methods that are allowed to be executed. See the following example:

<security-constraint>

<web-resource-collection>

<url-pattern>/rest/admin/\*</url-pattern>

</web-resource-collection>

<auth-constraint>

<role-name>admin</role-name>

</auth-constraint>

</security-constraint>

<security-constraint>

<web-resource-collection>

<url-pattern>/rest/orders/\*</url-pattern>

</web-resource-collection>

<auth-constraint>

<role-name>customer</role-name>

</auth-constraint>

</security-constraint>

<login-config>

<auth-method>BASIC</auth-method>

<realm-name>my-default-realm</realm-name>

</login-config>

* + 1. OAuth

OAuth is an authorization protocol to delegate rights for an application to act on the behalf of a user who granted their rights without giving away login credentials like username or password.

It was developed by Twitter, Magnolia and Google and registered as IETF standard in April 2010 under RFC 5849.

OAuth version 2.0 is simpler to use but often criticised for having too many flavours and implementations. OAuth 2 was standardized again by IETF in October 2012 under RFC 6749 and 6750. It is not backward compatible with OAuth 1.0a.

Some notable adopters are Facebook, Amazon, Google, Salesforce.com or Microsoft.

All relevant social media APIs are based on either OAuth 1.0a or 2.0.

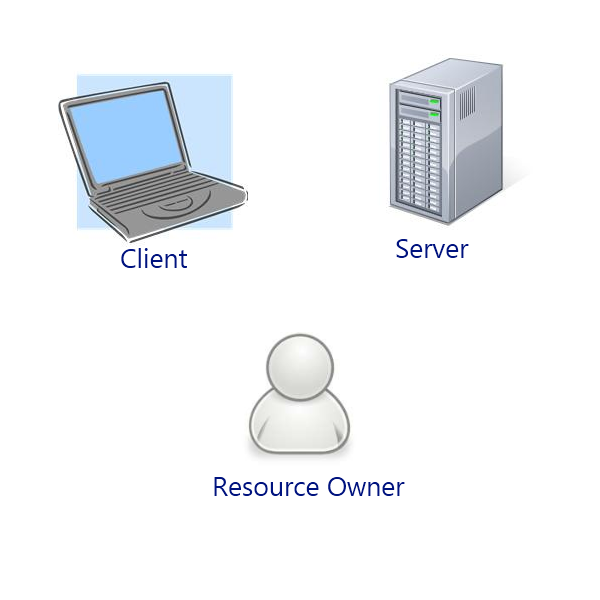


Figure 5 – OAuth parties

Figure 5 shows the actors of the OAuth flow, often also called the “OAuth Dance”.

* Resource Owner
* Client
* Server

To use OAuth, one has to create an application on the target service to have an entry point for the consumer or resource owner. Allowing access tokens to be issued to third-party clients by an authorization server, with the approval of the resource owner, or end-user. The client then uses the access token to access the protected resources hosted by the resource server.

The OAuth “Dance” has 3 major steps:

* Creation: Creating an application in the OAuth Social Media service
* Initialization: the right granting phase also called the OAuth Dance. At the end of the dance we obtain an access token (formed by a public and secret part) use in next step
* Signature: each request is signed with access token and token identifying the OAuth application that was granted the rights

Figure 6 shows the steps and authentication flow in detail for OAuth 1.0a.

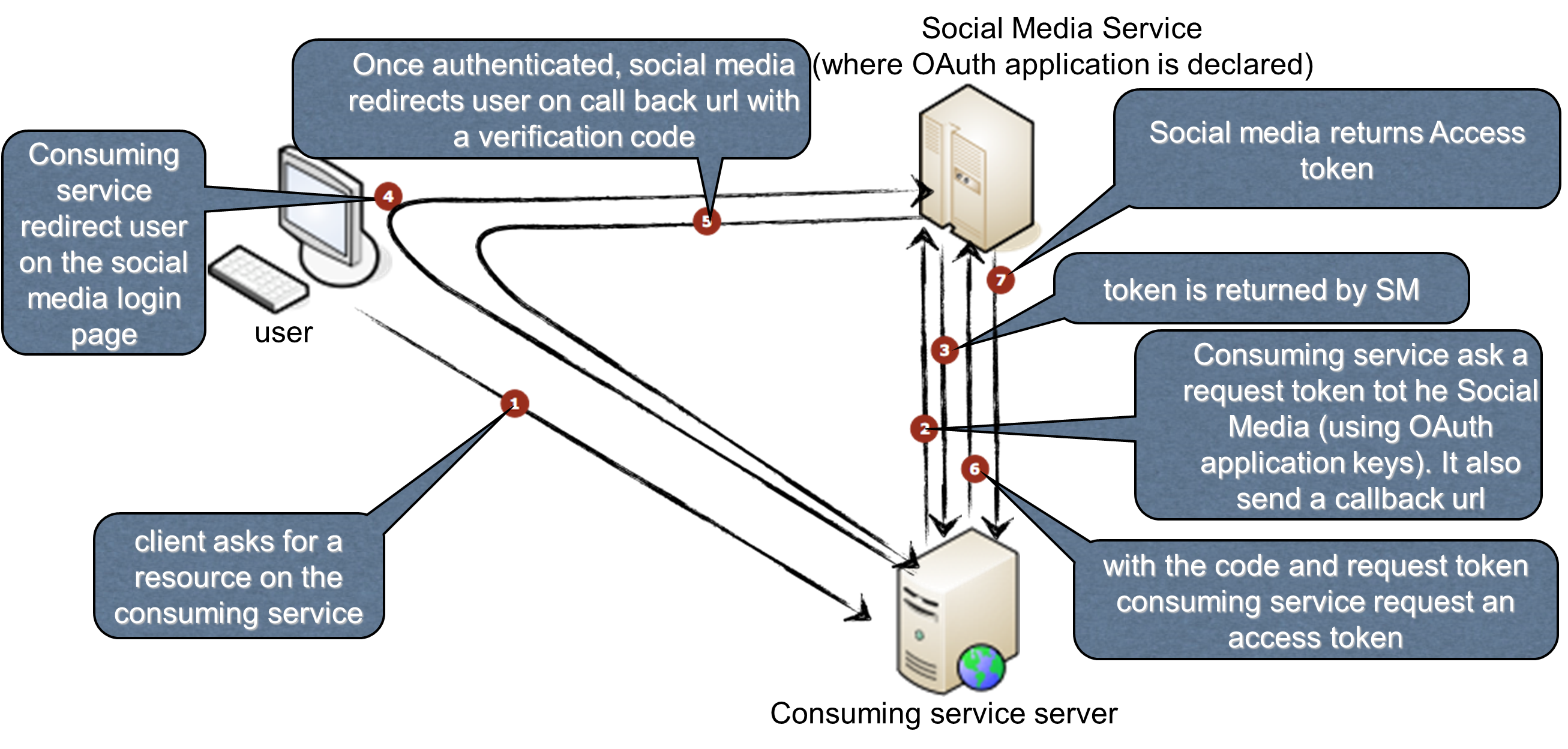


Figure 6 – OAuth 1.0a “Dance“

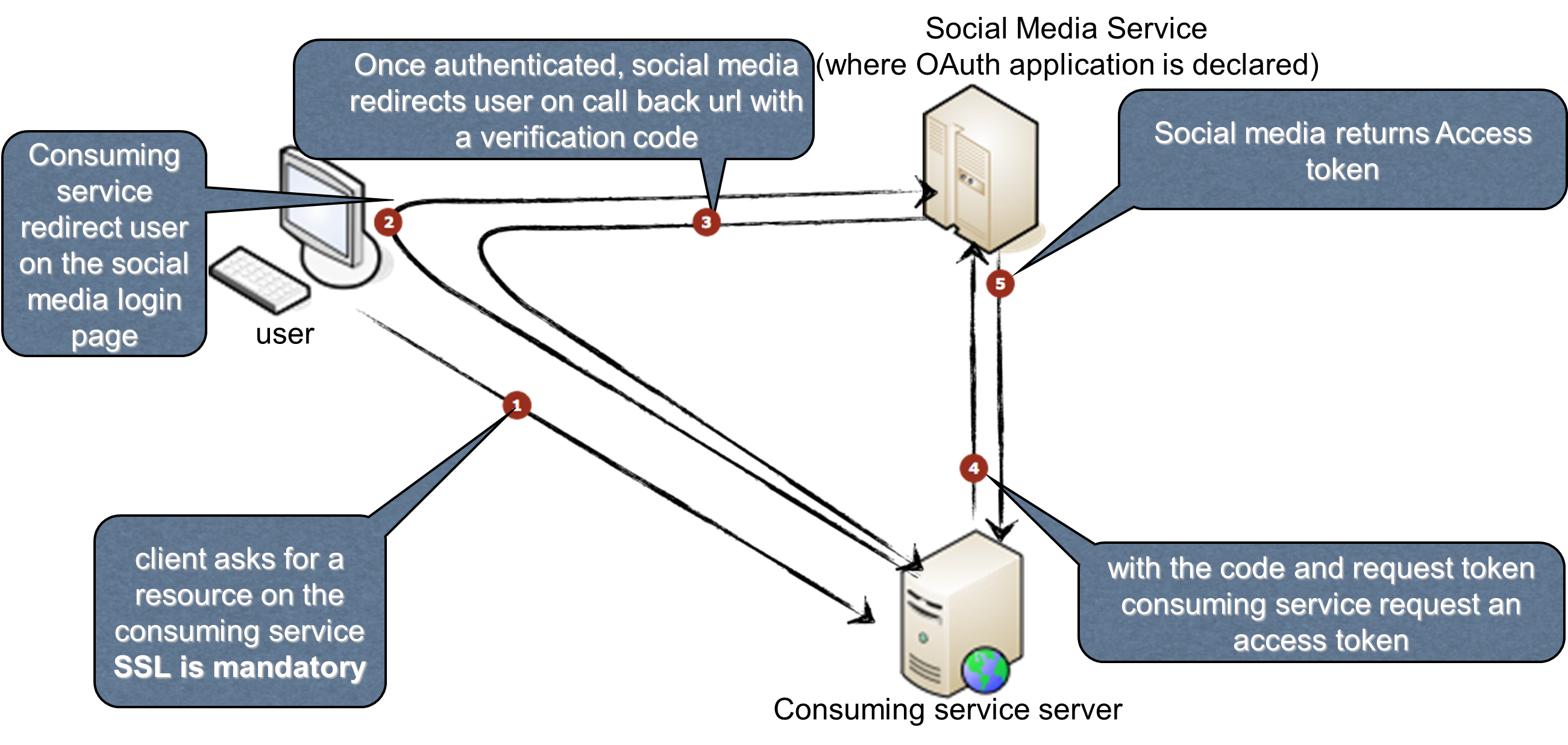


Figure 7 – OAuth 2 “Dance“

Figure 7 shows the same flow for OAuth 2. You can see the communication has been simplified, plus all exchange must now be SSL encrypted. With OAuth 1.0a this was optional and most implementing cases therefore left it out. Bearing a greater risk of interception or tampering than OAuth 2.

Authentication

Like the word “Agora”, authentication is also derived from Greek terms. Originally from Greek “af̱thentikós”, it was a derivative of the noun “authentes”, “doer, master”, which was formed from “autos”, “self” and the base -hentes, “worker, doer, being”.

While by far the most popular access control for social media and APIs, OAuth is not a genuine authentication, but “pseudo-authentication”, where the application specifically requests a limited access OAuth Token (“valet key”) rather than actual credentials (like a “passport”) see Figure 8 and Figure 9.

* + 1. OpenID

The initial OpenID protocol was created 2005 by Brad Fitzpatrick at Six Apart, after he had sold his social network LiveJournal to this blog software company. The name of the company did not refer to Six Degrees of Separation [6] as it may seem, but a 6 day age difference between its married co-founders.

It was code-named Yadis (acronym for "Yet another distributed identity system") but was then officially called OpenID after David Lehn, who had also considered a similar Single Sign-On project, but lacked time donated the domain “openid.net” [29] to Six Apart.

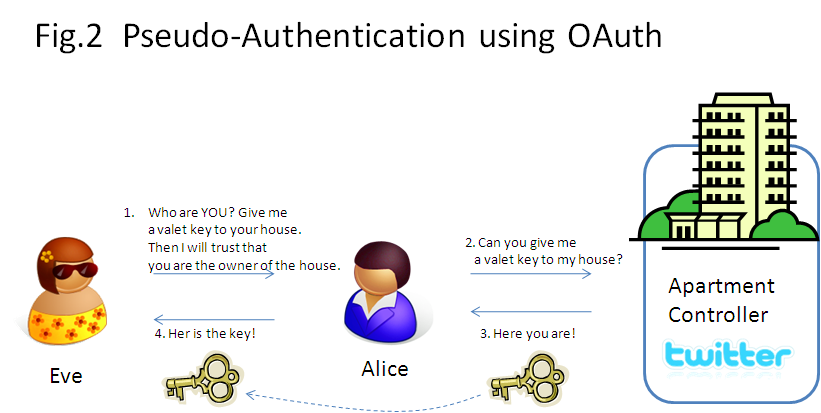


Figure 8 - Pseudo-Authentication using OAuth



Figure 9 - OpenID Authentication

Figure 10 shows the OpenID flow and its actors

* Browser
* User
* Relying Party
* Identity Provider



Figure 10 - OpenID Flow

Although OpenID has so far not been standardised by an organisation like IETF, many large Internet, Social and Telco players applied it, like AOL, Blogger, Flickr, France Telecom, Google, Hyves, LiveJournal, Microsoft, Mixi, Myspace, Novell, Orange, Sears, Sun, Telecom Italia, Telefonica O2, Universal Music Group, VeriSign, WordPress or Yahoo!

Over time others like SourceForge, PayPal or Facebook joined supporters of OpenID under the OpenID Foundation. However, in the last 2-3 years there’s been a steady decline in OpenID supporters. Facebook left OpenID, SourceForge also stopped offering an OpenID login (at the very least after Dice.com took it over from Geeknet) and in September 2013 Social Login SaaS provider Janrain announced that its service MyOpenID.com would be shut down on February 1, 2014.

So does that mean OpenID is dead? More or less in its original form, but at least still Google has high hopes for it. Strange coincidence, the initial creator of OpenID, Brad Fitzpatrick now works at Google, probably a main reason why the company remained loyal to OpenID, when most other former supporters abandoned it. OpenID Connect, Google’s new approach has however not that much in common with the old version of OpenID. It adds improved authentication and Identity Management on top of the otherwise winning combination of OAuth 2, JSON and RESTful Web Services.

OpenID Connect was released just at the end of February 2014. Beside Google, Microsoft, Salesforce.com or Deutsche Telekom adopted it so far. At Mobile World Congress 2014 in Barcelona GSMA announced plans for a mobile version “Mobile Connect”. Enough suggesting this “flavour of OAuth” may also stay for some time.

Security Frameworks

There are several Java-based security frameworks aside from base technologies like JAAS (Java Authentication and Authorization Service) supporting authorisation, authentication or both:

* Scribe
* PicketLink (by JBoss)
* Apache Shiro
* Spring Security (formerly Acegi)
* Google OAuth for Java
* Socialauth
* JSR 375

### **Scribe**

The Scribe project has built-in support for many popular services that use OAuth, such as Facebook, Twitter, Google or LinkedIn. It has very few external dependencies, mainly Apache Common Codec, allowing seamless integration into solutions without risk of dependency clash many larger and more complex solutions bear.

Calling an OAuth provider in Scribe can be as easy as the following one-liner shows:

OAuthService service **=** **new** ServiceBuilder**()**

**.**provider**(**LinkedInApi**.**class**)**

**.**apiKey**(**YOUR\_API\_KEY**)**

**.**apiSecret**(**YOUR\_API\_SECRET**)**

**.**build**();**

Until version 0.6 of Agorava, Scribe was at the heart of its OAuth functionality. From Agorava 0.7 on it has been replaced by JBoss PicketLink.

### **PicketLink**

PicketLink is an umbrella for several security related projects in the Red Hat/JBoss Middleware ecosystem. While Agorava is not a project officially created by Red Hat, it was created for this ecosystem and standards used there like CDI. Therefore supporting OAuth and Java EE Security through the means of PicketLink seemed a logical step for Agorava.

PicketLink is an Application Security Framework for Java EE applications. It provides features for authenticating users, authorizing access to business methods of your application, managing the application's users, groups, roles and permissions, etc.

These are the Top 8 Java Application Security Problems Solved by PicketLink:

1. What's the best way to add security to the application?
2. How do I authenticate and authorize users?
3. How can I control access to classes and methods?
4. How do I add Identity and Access Management (IAM) to my application?
5. How can I create a secure multi-tenancy architecture for my SaaS (Software as a Service) application?
6. How can I enable Single Sign-On based on SAML (Security Assertion Markup Language) in my application?
7. How do I add authentication and authorization to my REST layer and API?
8. How can my application authenticate users using their Facebook, Twitter or Google accounts?

Figure 11 shows an overview of PicketLink, its key components and technologies it supports. Some marked “Coming Soon” are already under development, but do not come with the GA (Generally Available or “Stable”) version of PicketLink yet.

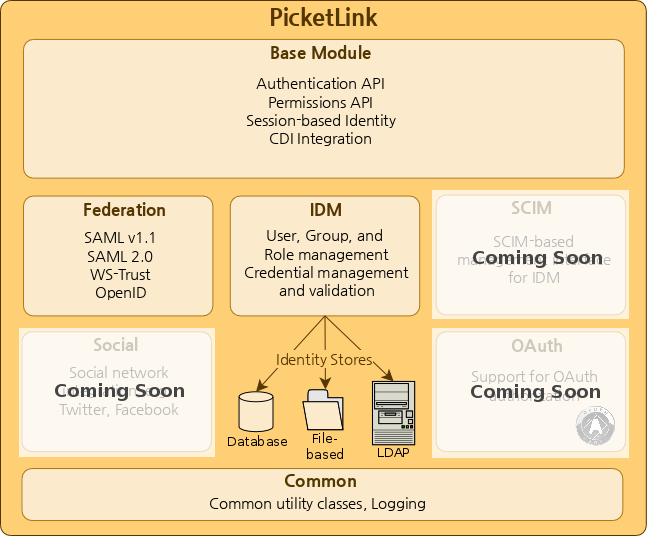


Figure 11 - PicketLink Overview

Agorava does however use mainly base functionality of PicketLink, like Authentication API or CDI Integration. Adding specialised OAuth and Social connectivity of its own.

The “dance” through the authentication flow is conducted by the PicketLink element **Authenticator**. Here an example of how to extend BaseAuthenticator

@PicketLink

public class SimpleAuthenticator extends BaseAuthenticator {

@Inject DefaultLoginCredentials credentials;

@Override

public void authenticate() {

if ("agentsmith".equals(credentials.getUserId()) &&

"matrix123".equals(credentials.getPassword())) {

setStatus(AuthenticationStatus.SUCCESS);

setAccount(new User("agentsmith"));

} else {

setStatus(AuthenticationStatus.FAILURE);

FacesContext.getCurrentInstance().addMessage(null, new FacesMessage("Authentication Failure - The username or password you provided were invalid."));

}

}

}

“3.2.1 Agorava PicketLink” will show in more detail how this is applied by Agorava Core and its PicketLink module.

### **JSR 375**

No, this is not a number-twister of “1.4.7 JSR 357”, although many security aspects 357 had planned, especially a tight integration with CDI sound quite familiar [30].

Digital Identity

* + 1. Who is No.6?

Like a question from the famous 1960s Sci-Fi series “The Prisoner” where No.6 tries to find out his own identity and that of others holding him captive both users of social media services and the systems often stumble into identity crisis.

Especially the biggest players in Social Media often face major problems when it comes to authentication and the true identity of their users.

Privacy

The term privacy comes from “private” and has Latin origin: “prīvātus”, literally means taken away from public affairs.

While ancient Romans may have been able to hide some of their affairs from public audience and famous writers of their time, and the time from something actually happening to being written down and consumed by readers was at least days, more often months or even years, it can sometimes be just the blink of an eye or click of a mouse-button today.

The revelations by whistle-blower Edward Snowden were the most extensive, but certainly not first case of privacy concerns brought into public light especially with regards to Social Networks or similar Online Services [32]. The ability to combine data from various sources and everyday life aspects into a digital picture and footprint of users was in prior decades mostly the domain of police and secret service tools like “Dragnet” (not so much the movie, though it referred to the system in a humorous way).

Not just Snowden and his skills now available to his hosts in Russia is a clear indicator, Western countries and agencies aren’t the only ones interested in such information. Many try to control; some simply block or restrict entire services, if they are not provided the same access to vital information by the social networks as intelligence agencies in their home country, usually the United States. China, Russia or Turkey, the list keeps growing of countries temporarily or permanently blocking some providers. If ownership or hosting is outsourced, not for tax but cheap workforce reasons, then some services can be affected by practices in these countries, too. One of the most popular examples we heard about earlier, because OpenID 1.0 was once created there is LiveJournal. In 2007 it was sold by Six Apart to Russian Media group SUP Media in December 2007.

After 2009 most of its production was also moved to Russia and employees fired in Silicon Valley. Although privately owned, SUP Media is clearly controlled and also censored by Russian secret services. For example in 2007 Russian blogger Savva Terentyev was accused of fomenting social hatred to the staff of the Ministry of Internal Affairs and sentenced to one year probation due to his comment in the blog of a local journalist. Not to mention recent murders like that of regime-critical Boris Nemtsov in Moscow only hours after he had announced to disclose information on Russia’s involvement in Ukraine. Ironically, even in one of the most tightly censored countries like Russia, there is a tiny glimpse of hope, Social media if not bring real reforms may at least help finding clues to his murder [33]

While the argument of “National Security” and trying to catch terrorists or severe criminals online was used to justify mass surveillance on a global scale, critical bloggers or journalists often risk not just their privacy but lives if their identity is unveiled to people and agencies they criticise.

While efforts like the “Vorratsdatenspeicherung” by European hosting providers and other internet or mobile companies mean, there isn’t total privacy in Europe either, most EU countries, especially Germany have rather strong privacy and data protection laws compared to the United States and many other parts of the world. A few countries like Iceland (not currently in EU, but EFTA) have even stronger data protection laws, as well as many Scandinavian countries.

Social networks genuinely based in these countries, e.g. German XING, but also Deutsche Telekom and other internet companies from Germany like GMX/1&1 claim, their data is safe from direct access by NSA or other spies. Although local and international enquiries will be executed by these if probable cause exists against a user of serious offenses like murder, terror, organised crime or child pornography, neither of these services are likely to conduct mass-screenings of every single user on a daily basis like NSA and other agencies do in the US and other countries. Without a warrant to access records or pre-emptively snoop into communication most European users should feel safe if the provider and its servers are all genuinely based in Europe and data isn’t sent via connections abroad.

In addition to that, services like XING also offer much finer grained levels of access permissions and control by its users. Where it is possible to sometimes restrict e.g. who can see each individual data item. Let’s say you want to show your name and day of birth, but not the year or your age to everyone visiting your profile, or allow only your friends to see your phone number, this and more can be controlled by systems like XING.

This was also among the main reasons, why XING abandoned OpenSocial, because it cannot offer such fine grained access control or in some cases won’t even provide any, but instead show everything to everyone or at least all data to all of your friends.

As opposed to that, Agorava can leverage security mechanisms and access control by individual social networks. And being able to connect to selected services, even multiple of them in a single session, you can decide where you place what information and access it from the provider you best trust to handle it with the necessary care.

1. Agorava

Overview

Agorava [34] is a Social Framework, providing CDI modules, contexts and extensions to interact with major social networks or similar services exposing APIs. It offers these services to Java and Java Enterprise applications and eases mixing them in the spirit of mashups [12]

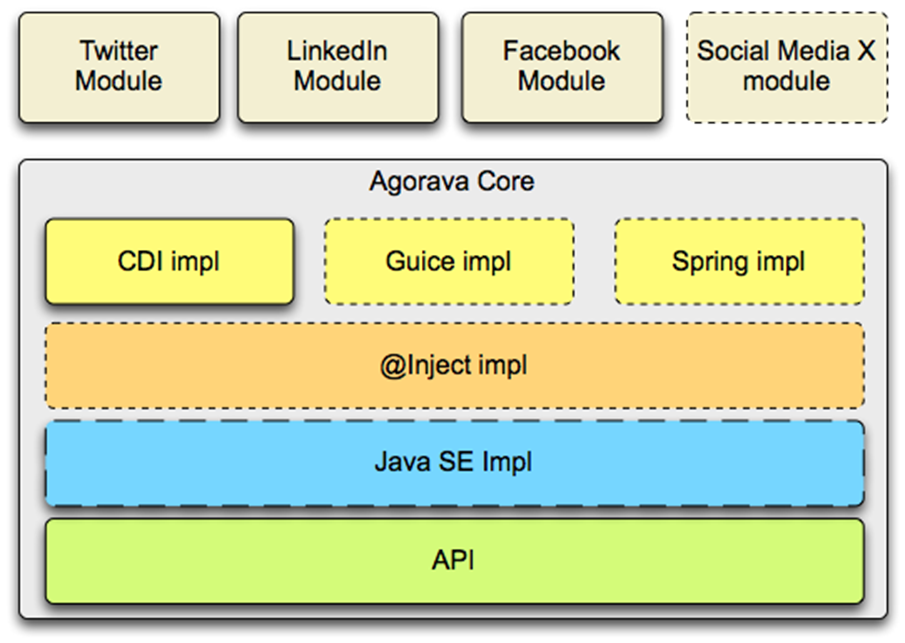


Figure 12 - Agorava Macro Architecture

These are the key characteristics and features of Agorava

* A generic portable REST client API
* A generic API to handle OAuth 1.0a and 2.0 services
* A generic API to work with JSON serialization and de-serialization
* A generic identification API to retrieve basic user information from a Social Service (as Identity Provider)
* A multiservice manager API allowing to deal with multiple OAuth applications and sessions in the same application
* Currently out of the box, Agorava connects to:
* Twitter
* Facebook
* LinkedIn
* And offers an easy way to extend it by creating a new modules for any other service or API required

Agorava is independent of CDI implementation and fully portable between Java EE 6 (minimum) and Servlet environments enhanced by CDI. It can be also used with CDI in Java SE in a Desktop or SE Embedded/Mobile application. It has been fully tested with the CDI RI implementation (JBoss Weld). Open Web Beans and Caucho Resin are also supported since the 0.5 release.

Agorava is often explained as a sort of “Java Connector Architecture for Social”. And indeed, there are striking similarities between the idea behind Agorava and what Java Connector Architecture (JCA) does under the Java EE umbrella.

Java EE Connector Architecture defines a standard architecture for connecting the Java EE platform to heterogeneous Enterprise Information Systems (EIS) including transaction processing systems such as CICS Transaction Server and Enterprise Resource Planning (ERP) systems like SAP.

Especially the so called Common Client Interface (CCI), which defined a standard client API allowing applications to access multiple resource adapters.



Figure 13– JCA Common Client Interface

One must add, JCA is a fairly old part of the Java EE umbrella, last updated with EE 6. It received a Maintenance Release about a year ago, also with some CDI support, but compared to some of the new, often Cloud and API based ERP solutions along the lines of Salesforce.com, it seems a bit outdated and is also among the lesser known and used parts of the Java EE ecosystem.

Though the JSR 357 proposal did not explicitly phrase it the “JCA way”, it is also possible, some of the supporters and JCA Experts (beside Oracle that was mostly IBM or SAP, most other companies now dominating in the Cloud didn’t even exist when JCA was originally designed) felt the little “dinosaur” they fostered could be threatened by a new standard like a Java Social Connector API.

Core

Agorava Core includes common, social networking services that are independent of a particular service provider. Offering features such as:

* OAuth connectors to authenticate against OAuth providers
* Support for generic authentication and user profile management
* A Social Multi-Account Service supporting
  + Multiple Social Network Connections
  + Multiple Sessions for the same Social Network provider, e.g. connected to multiple user accounts or queries at the same time

Agorava Core also provides implementations to these services. The main implementation is based on CDI and DeltaSpike right now, but most services have been generalised using JSR 330 (Dependency Injection) which allows more lightweight Java SE alternatives like Google Guice, Dagger or the SE version of the CDI Implementation Weld.

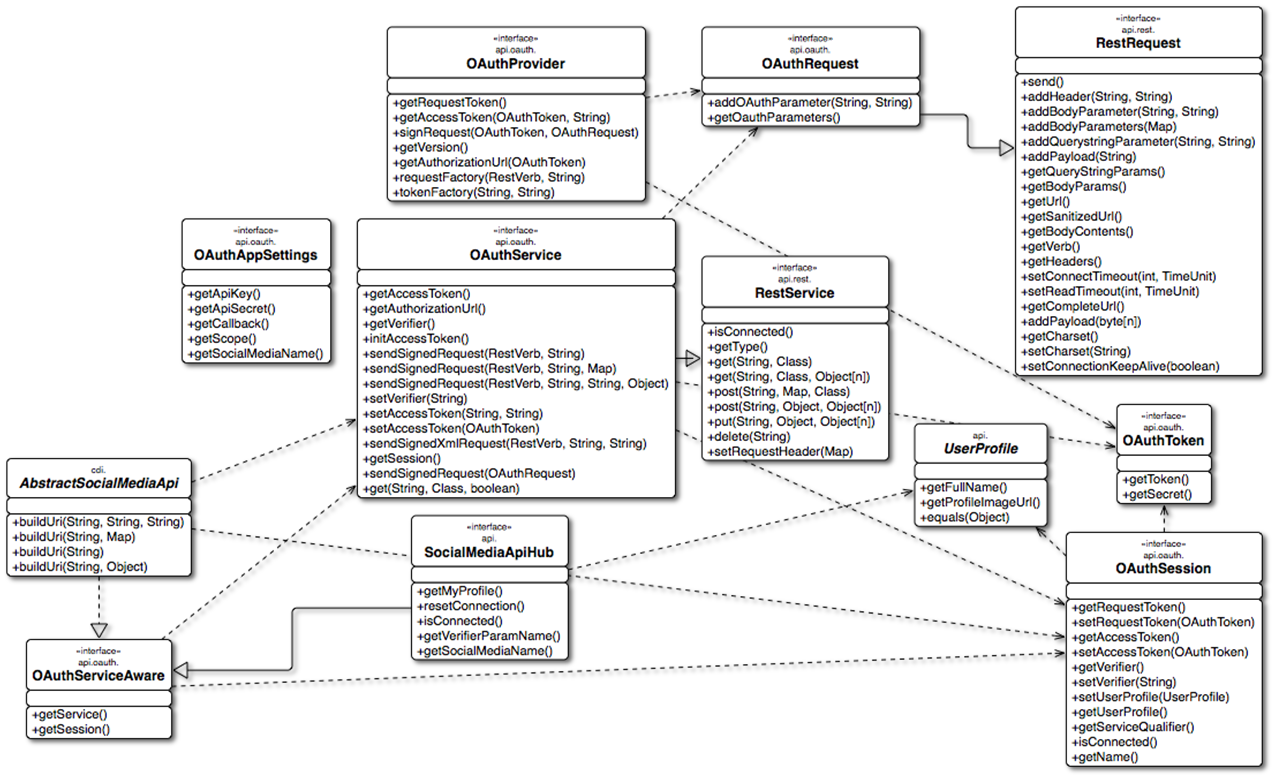


Figure 14 - Agorava Core API Architecture

* + 1. Agorava PicketLink

Working together with OAuth functionality and user profile management, Agorava Core also contains a security sub-system based on PicketLink.

As promised in 2.3.2, here the specialised Agorava version of the PicketLink BaseAuthenticator:

@Generic

public class AgoravaAuthenticator extends BaseAuthenticator {

@InjectWithQualifier

OAuthAppSettings settings;

@Inject

DefaultLoginCredentials credentials;

@Inject

@DeltaSpike

Instance<HttpServletResponse> response;

@Inject

OAuthLifeCycleService lifeCycleService;

@Override

public void authenticate() {

if (lifeCycleService.getCurrentSession().isConnected()) {

OAuthSession session = lifeCycleService.getCurrentSession();

UserProfile userProfile = session.getUserProfile();

credentials.setCredential(session.getAccessToken());

setStatus(AuthenticationStatus.SUCCESS);

User user = new User(userProfile.getId());

user.setFirstName(userProfile.getFullName());

setAccount(user);

} else {

String authorizationUrl = lifeCycleService.startDanceFor(settings.getQualifier());

try {

response.get().sendRedirect(authorizationUrl);

} catch (IOException e) {

e.printStackTrace();

}

credentials.setStatus(Status.IN\_PROGRESS);

setStatus(AuthenticationStatus.DEFERRED);

}

}

}

Connectors

Agorava currently comes with default connectors to:

* Facebook
* LinkedIn
* Twitter

In addition to that, 3rd party Agorava Connectors were created for services like:

* EmpireAvenue
* GitHub
* Google+
* Instragram
* Meetup
* StackOverflow
* TripIt
* WordPress
* XING
* Yammer
  + 1. Anatomy of a Connector

Socializer

Socializer is a real life demonstration Web App for Agorava.

It allows you to connect to various social Media, and see your timeline and post update.

Socializer was since ported from Java Server Faces to the Apache Wicket Web Framework and more recently AngularJS.

Both ports are currently still rather experimental and therefore not official parts of the Agorava ecosystem.

* + 1. Integrating a connector

Adding a new connector to Socializer is quite easy. We’ll take the XING connector as example how to do it.

After cloning <https://github.com/agorava/agorava-socializer> the following lines in the Socializer **pom.xml**

<!-- XING -->

<dependency>

<groupId>org.agorava</groupId> <artifactId>agorava-xing-api</artifactId>

<version>0.3.0-SNAPSHOT</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.agorava</groupId>

<artifactId>agorava-xing-cdi</artifactId>

<version>0.3.0-SNAPSHOT</version>

<scope>runtime</scope>

</dependency>

Define dependencies for XING API and CDI Implementation.

Next we add the app settings for XING to CDI producer class

**org.agorava.socializer.SettingsProducer.java**

@ApplicationScoped

@Produces

@Xing

@OAuthApplication(params = {@Param(name = OAuthAppSettingsBuilder.**PREFIX**, value = "xing")})

public OAuthAppSettings **xingSettings**;

Backed by the properties file

**src/main/resources/local/agorava.properties**

xing.apiKey=<your API Key>

xing.apiSecret=<your API Secret>

Both **API key** and **API secret** have to be defined for a local Socializer application in the social media account of your choice, see <https://dev.xing.com/applications/dashboard> for XING. Most proper services offer a „Test key“ or development key like XING does.

Unfortunately some, most notably Facebook ignore the need of developers right now. Facebook’s Graph API in the recent version (v2.5) also contains a bug still requesting a deprecated call.

**xing.xhtml**

<ui:composition xmlns="http://www.w3.org/1999/xhtml"

xmlns:ui="http://java.sun.com/jsf/facelets"

xmlns:f="http://java.sun.com/jsf/core"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:p="http://primefaces.org/ui">

<h:panelGroup id="xingtl" layout="block">

<div class="well">

<h6>What's on your mind?</h6>

<p>

<p:inputTextarea value="#{socialClient.status}" styleClass="post-area" />

</p>

</div>

</h:panelGroup>

</ui:composition>

1. Enterprise Social

There are numerous social media websites either for general purpose or focus groups and special communities, including former alcoholics, music fans, developers, expats, mothers, students, teachers, government employees, people who are into knitting or BDSM. The „Fifty Shades of Grey“ books and films will probably add even more „shades“ to such a list ;-) [35]

Advertising and Analytics

Location

Media

Products & Services

Profiles

Specialties

* + 1. Software Development
    2. Health and Fitness

Bibliography

1. Agora – Wikipedia: <http://en.wikipedia.org/wiki/Agora>
2. Forum – Wikipedia: <http://en.wikipedia.org/wiki/Forum_(Roman)>
3. History of Information – Wikis / Forum Posts: <http://www.historyofinformation.com/expanded.php?category=Social+Media+%2F+Social+Networks+%2F+Wikis>
4. The Role of Social Media in Political Mobilisation, dissertation by Madeline Storck, 20 December 2011 <http://www.culturaldiplomacy.org/academy/content/pdf/participant-papers/2012-02-bifef/The_Role_of_Social_Media_in_Political_Mobilisation_-_Madeline_Storck.pdf>
5. Agorism – Wikipedia: <http://en.wikipedia.org/wiki/Agorism>
6. Six Degrees – Wikipedia: <http://en.wikipedia.org/wiki/Six_degrees>
7. Reuters: Mt. Gox files for bankruptcy <http://www.reuters.com/article/2014/02/28/us-bitcoin-mtgox-bankruptcy-idUSBREA1R0FX20140228>
8. Friend of a Friend – Wikipedia: <http://en.wikipedia.org/wiki/FOAF_(software)>
9. RSS – Wikipedia: <http://en.wikipedia.org/wiki/RSS>
10. A Programmable Web: An Unﬁnished Work by Aaron Swartz, Copyright © 2013 by Morgan & Claypool <http://upload.wikimedia.org/wikipedia/commons/3/3f/Aaron_Swartz_s_A_Programmable_Web_An_Unfinished_Work.pdf>
11. JSR 168 – Portlet Specification: <http://www.jcp.org/en/jsr/detail?id=168>
12. Mashup – Wikipedia: <http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)>
13. BEA AquaLogic: <http://blog.programmableweb.com/2007/07/17/bea-aqualogic-targets-enterprise-mashups/>
14. Google Reader’s shutdown, the rise of walled gardens and the future of the open web – GIGAOM: <http://gigaom.com/2013/07/03/google-readers-shutdown-the-rise-of-walled-gardens-and-the-future-of-the-open-web/>
15. OpenSocial – Wikipedia: <http://en.wikipedia.org/wiki/OpenSocial>
16. Are Universal Social Engagement Standards Possible? – Information Week by Dion Hinchcliffe: <http://www.informationweek.com/social-business/social_analytics/are-universal-social-engagement-standard/240147172?pgno=1>
17. OPENi: <http://www.openi-ict.eu/>
18. Twitter4J – GitHub: <http://github.com/yusuke/twitter4j>
19. Apache Shindig: <http://shindig.apache.org/>
20. OSDE – Google Code: <https://code.google.com/p/opensocial-development-environment/>
21. SocialSite – Java.net: <http://socialsite.java.net>
22. DaliCore – Java.net: <http://dalicore.java.net>
23. Acquisitions by Oracle – Wikipedia: <http://en.wikipedia.org/wiki/List_of_acquisitions_by_Oracle>
24. Oracle Social Link – Java.net: <http://sociallink.java.net>
25. Spring Social: <http://springsource.com/social>
26. The Java Social Media API is dead: Long Live Agorava: <http://jaxenter.com/the-java-social-media-api-is-dead-long-live-agorava-43362.html>
27. Principle of least privilege – Wikipedia: <http://en.wikipedia.org/wiki/Principle_of_least_privilege>
28. Context and Dependency Injection for Java EE: <http://cdi-spec.org/>
29. OpenID: <http://openid.net/>
30. JSR 375 – JavaTM EE Security API: <http://www.jcp.org/en/jsr/detail?id=375>
31. TechCrunch – Twitter Suddenly Locks Thousands of Accounts: <http://techcrunch.com/2015/02/10/twitter-account-locked/>
32. Privacy concerns with social networking – Wikipedia: <http://en.wikipedia.org/wiki/Privacy_concerns_with_social_networking_services>
33. Can Social Media Solve Boris Nemtsov’s Murder? <https://socialmedianyu.wordpress.com/2015/03/03/can-social-media-solve-boris-nemtsovs-murder/>
34. Agorava Project: <http://agorava.org>
35. List of social networking websites – Wikipedia: <http://en.wikipedia.org/wiki/List_of_social_networking_websites>

About the Author

Werner Keil is DevOps Build Manager, Java and .NET expert at a print and media world leader after working as Scrum Master and Eclipse/Java Consultant for a leading vendor of Real-time/Embedded systems, DevOps Build Manager at Maersk, Agile Coach, Principal Consultant and Distinct Architect for a Financial Services company and an Agile consultancy. Helping Global 500 Enterprises across industries like Mobile/Telco, Web 2.0, Finance, Travel/Logistics, Automotive, Healthcare, Environment & Public Services, as well as IT vendors like Oracle or IBM. Among his earlier clients was Sony where he designed and implemented micro-format based tags its online music portals.

He has worked for more than 25 years as developer, project manager, software architect, analyst and consultant on leading-edge technologies for Banking, Insurance, Telco/Mobile, Media and Public sector. Werner develops enterprise systems using Java/JEE, Oracle, IBM or Microsoft, does Web design and development using Adobe, Ajax/JavaScript, dynamic or functional languages.

Besides working for major companies, he runs his own creative and talent agency Creative Arts & Technologies and in his spare time, runs and supports open-source projects, writes song lyrics, novels, screenplays, technical books and articles.

Werner was among the first committers to Sysdeo Tomcat Launcher, a predecessor of Eclipse WTP, especially designing its UI. He is Committer at Apache Foundation, Eclipse Foundation, Babel Language Champion, UOMo Project Lead an active member of the Java Community Process, in JSRs like 321 (Trusted Java), 331 (CP), 344 (JSF 2.2), 358/364 (JCP.next), 350 (Java State), 351 (Java Identity), 354 (Money and Currency), 360/361 (ME Embedded), 362 (Portlet 3), 363 (Unit-API, also Spec Lead), 365 (CDI 2), 366 (Java EE 8), 374 (JSON-P 1.1), 375 (JavaTM EE Security API), 377 (Desktop/Embedded Application Framework) and the only Individual Member of the Executive Committee outside the United States.