

OPENSPIME ARCHITECTURE



- 0. INTRODUCTION 3**
- 0.0. OVERVIEW 3**
- 0.1. COPYRIGHT..... 3**
- 0.2. PERMISSIONS..... 3**
- 0.3. DISCLAIMER OF WARRANTY 3**
- 0.4. LIMITATION OF LIABILITY 3**
- 0.5. RELATION TO XMPP..... 4**
- 0.6. STATUS OF THE DOCUMENT..... 4**
- 0.7. TERMINOLOGY 4**

- 1. OPENSPIKE ARCHITECTURE 5**
- 1.0. OVERVIEW 5**
- 1.1. FOREWORD ON XMPP AND IDs..... 5**
 - 1.1.0. XMPP..... 5
 - 1.1.1. OSID 6
- 1.2. ELEMENTS EXPLAINED 6**
 - 1.2.0. SPIMES..... 6
 - 1.2.1. WIRELESS/BLUETOOTH/OTHER BRIDGES..... 6
 - 1.2.2. SCOPENODES..... 7
 - 1.2.3. SERVICES..... 8
 - 1.2.4. SPIPEGATES..... 9

- 2. SECURITY CONSIDERATIONS..... 10**

0. Introduction

This document illustrates the OpenSpime architecture and the overall data and functionality flow.

0.0. Overview

OpenSpime is an Open Services distributed network which provides all necessary functionalities to allow data collection and encrypted messaging between entities.

This document illustrates the OpenSpime architecture and assumes you are familiar with the Extensible Messaging and Presence Protocol (XMPP, <http://www.xmpp.org>).

0.1. Copyright

This OpenSpime protocol is copyright ©2008 by WideTag Inc.

0.2. Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this protocol (the "Protocol"), to make use of the Protocol without restriction, including without limitation the rights to implement the Protocol in a software program, deploy the Protocol in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Protocol, and to permit persons to whom the Protocol is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Protocol.

Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Protocol, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or WideTag Inc.

0.3. Disclaimer of Warranty

This Protocol is provided on an **"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.** In no event shall WideTag Inc or the authors of this Protocol be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the Protocol or the implementation, deployment, or other use of the Protocol.

0.4. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall WideTag Inc or any author of this Protocol be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising out of the use or inability to use the Protocol (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if WideTag Inc or such author has been advised of the possibility of such damages.

0.5. Relation to XMPP

The Extensible Messaging and Presence Protocol (XMPP) is defined in the XMPP Core (RFC 3920, <http://www.ietf.org/rfc/rfc3920.txt>) and XMPP IM (RFC 3921, <http://www.ietf.org/rfc/rfc3921.txt>) specifications contributed by the XMPP Standards Foundation to the Internet Standards Process, which is managed by the Internet Engineering Task Force in accordance with RFC 2026. Any protocol defined in this document has been developed outside the Internet Standards Process and is to be understood as an **extension to XMPP** rather than as an evolution, development, or modification of XMPP itself.

0.6. Status of the document

This document is FINAL and has been written by Roberto Ostinelli for WideTag, Inc.

0.7. Terminology

The capitalized key words:

- **MUST, MUST NOT, REQUIRED,**
- **SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED,**
- **MAY,** and **OPTIONAL,**

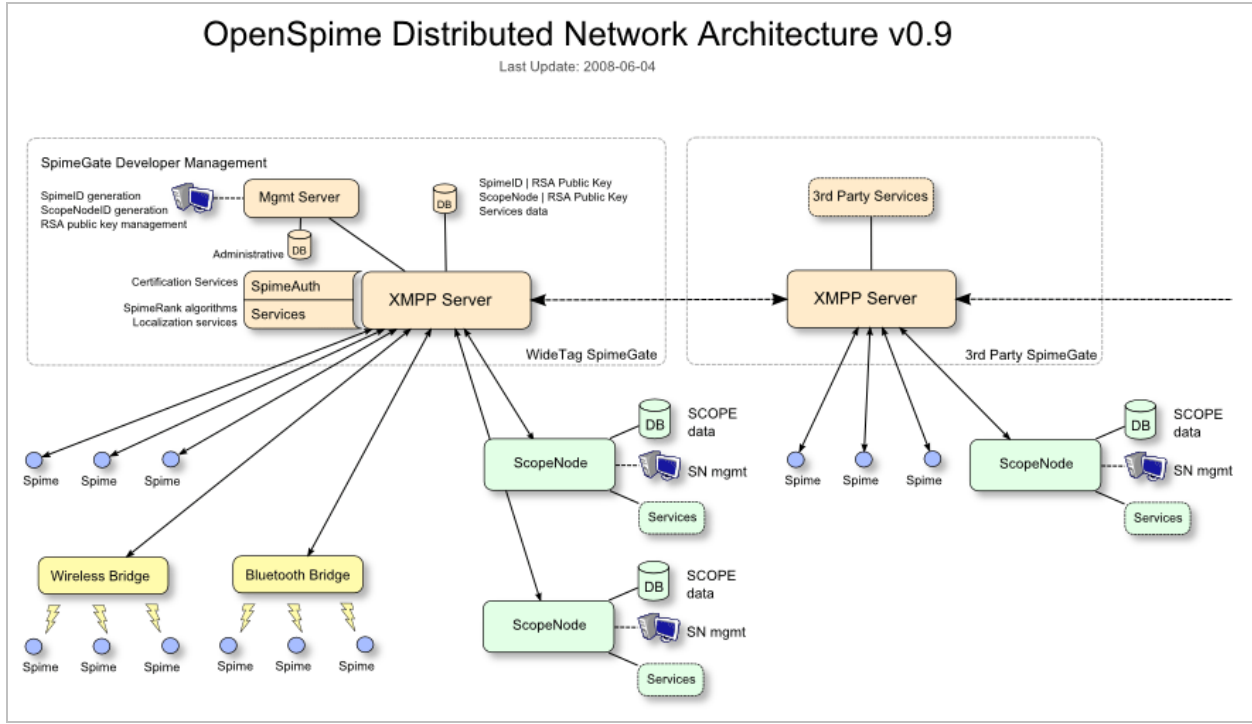
in this document are to be interpreted as described in RFC 2119 (<http://www.ietf.org/rfc/rfc2119.txt>).

Inside this document the following terms will have these specific meanings:

- **spime:** 'spime' (a contraction of the words 'space' and 'time') is a neologism coined by Bruce Sterling for a currently-theoretical object that is aware of its environment, can track its history of use and interact with the world by communicating data;
- **spime talk:** spime-to-spime messaging;
- **scope:** a reason for collecting data;
- **entity:** any application or device accessing the OpenSpime network;
- **client:** an application or system that accesses a (remote) service on another computer system known as a server by way of a network (source: Wikipedia);
- **server:** an application or device that performs services for connected clients as part of a client-server architecture. A server application, is "an application program that accepts connections in order to service requests by sending back responses" (RFC 261 HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>);
- **request:** the initial sending of a data request message from an application to another application of a network;
- **response:** the reply done by the application recipient of a request.
- **message:** a communication packet sent between a sender application or device to a recipient application or device.

1. OpenSpime Architecture

1.0. Overview



1.1. Foreword on XMPP and IDs

1.1.0. XMPP

As it can be seen from the architecture schema here above, OpenSpime relies on the Extensible Messaging and Presence Protocol (XMPP, <http://www.xmpp.org>) to handle message transport. XMPP is an open Extensible Mark-up Language (XML) protocol for near-real-time messaging, presence, and request-response services.

XMPP has been chosen mainly for these reasons:

- XMPP servers are well-known robust applications, which handle both **synchronous** and **asynchronous communication** (based on the 'presence', i.e. online status of XMPP clients);
- XMPP is based on **identity handling**, which is essential for OpenSpime functionalities;
- XMPP allows both **one-way** communication (messaging) and **two-way** communication (REST-like interaction);
- XMPP is **extensible** through the addition of XML namespaces inside top-level tags, though providing a transport wrapper for the OpenSpime protocol;
- XMPP has a series of **functionalities** we believe will be very useful in future developments of OpenSpime.

1.1.1. OSID

An OpenSpime IDentifier, aka **OSID**, is a JID (Jabber IDentifier, for more information please refer to <http://www.ietf.org/rfc/rfc3920.txt>) used on the OpenSpime network. An OSID is therefore a unique identifier on the OpenSpime network. OSIDs are spime identifiers, ScopeNode identifiers and Service identifiers (see here below).

1.2. Elements explained

1.2.0. Spimes

From an architectural point of view, spimes are XMPP clients.

1.2.0.0. SpimeID

Every spime accessing the OpenSpime architecture **MUST** be provided with a SpimeID, a unique identifier that allows to access all of the OpenSpime network functionalities.

A SpimeID is an OSID, which **MUST** be a full JID (since it is a device, it is by definition a resource). Examples of SpimeID are:

*4A6B-5638-0312111D@spime.openspime.com/spime
me@mydomain.com/spime
gateway@mynetwork.com/spime12*

Having a SpimeID allows a spime to:

- **send data** over the OpenSpime network;
- **be reachable**, since its SpimeID is its unique address;
- **digitally sign** sent data;
- **receive encrypted data**, which ensures privacy and sensible data protection;
- reach **higher levels of trust** on the data it transmits, since it is possible to check for coherence on the data sent by a known spime (thus increasing its impact on, for instance, global scopes).

1.2.0.1. RSA Key

Every spime **MAY** have its own **RSA key** (secret and public pair). The public RSA key of a spime **SHOULD** be transmitted to a SpimeGate certification service (see below §1.2.4.1). This allows:

- entities to exchange public RSA keys **via dedicated services**;
- **digital signature**, otherwise not possible.

It is up to the spime owner to communicate its key or eventual key changes, and consider automation in propagating the key change information to other entities (via, for instance, PubSub of the Public Key Publishing XMPP extension or custom spime talk).

1.2.1. Wireless/Bluetooth/Other bridges

Depending on hardware/software capabilities, some spimes may not be able to connect via **TCP/IP** directly, or have the necessary processing power needed to provide **encryption** and **XMPP communication**. Therefore, some Wireless, Bluetooth or other bridges might be needed to allow spimes to send their collected data through the OpenSpime network.

1.2.2. ScopeNodes

Data collected by spimes is transmitted via the OpenSpime architecture to a **specific ScopeNode** in charge of collecting their data.

Every ScopeNode server has its own **scope**, i.e. a reason for collecting data. Data collected using a scope **will be only available on the ScopeNode database**, and it is up to the ScopeNode owner to define the eventual policies for the aggregation and distribution of the data, or eventually develop and provide applications that use it.

The scope may be global and **public** such as CO2 earth monitoring, or **private** such as personal or company data monitoring. For this reason, **global scopes**, such as the ones related to earth monitoring, SHOULD only be provided by organizations which guarantee the respect of Open Standards policies and of best practices concerning privacy of the data. The public scopes SHOULD be published on openspime.org.

A ScopeNode MUST have **one scope**, which is defined and managed at ScopeNode level.

From an architectural point of view, ScopeNodes are XMPP clients.

1.2.2.0. SNID

Every ScopeNode has its unique ScopeNode Identifier, aka **SNID**, an OSID that allows to access all of the OpenSpime network functionalities.

A SNID MUST be a full JID (since a ScopeNode is an XMPP client, it is by definition a resource). Examples of SNID are:

775B-F2C2-2B3765F4@scopenode.openspime.com/myscope
mySNID@mydomain.com/co2

Having a SNID allows a ScopeNode to **receive data** which MAY also be **encrypted**, thus ensuring privacy and sensible data protection.

1.2.2.1. RSA Key

Every ScopeNode MAY have its own **RSA key** (secret and public pair), which is generated at ScopeNode level. The public RSA key of a ScopeNode SHOULD be transmitted to a SpimeGate certification service (see below §1.2.4.1). This allows:

- entities to exchange public RSA keys **via dedicated services**;
- **digital signature**, otherwise not possible.

It is up to the ScopeNode owner to communicate its key or eventual key changes, and consider automation in propagating the key change information to the relevant spimes (via, for instance, PubSub of the Public Key Publishing XMPP extension or custom spime talk).

1.2.2.2. ScopeNode Management

ScopeNodes might provide interfaces allowing to manage:

- data collection **format**;
- **scopes**;
- **RSA** key pair **generation**;
- **SpimeID authorization** (i.e. which spimes are allowed to send data to the ScopeNode).

1.2.2.3. Security and Privacy issues

Security and Privacy of data transmitted through the OpenSpime architecture is therefore enforced by three factors:

- **data** is kept on the **ScopeNodes** only;
- **data** sent to a ScopeNode MAY be **encrypted** using the ScopeNode public RSA key, in which case this data cannot be interpreted by any other OpenSpime network entity;
- Many ScopeNodes are distributed as **Open Source** code, though avoiding risk for unwanted malicious software components.

1.2.3. Services

Services provided on the OpenSpime network MAY be **transparent**, i.e. be contacted directly by other entities, in which case they will have their own OSID.

From an architectural point of view, services are XMPP clients, dedicated to providing services to other OpenSpime network entities.

1.2.3.0. ServID

Every transparent service MUST be provided with a ServID, a unique identifier that allows to access all of the OpenSpime network functionalities.

A ServID is an OSID, which MAY be a full JID. Examples of ServID are:

services@spime.openspime.com/cert
myservice@mydomain.com
myservice.mynetwork.com
services@mynetwork.com/service12

Having a ServID allows a service to:

- **be reachable**, since its SpimeID is its unique address;
- **digitally sign** sent data;
- **receive encrypted data**, which ensures privacy and sensible data protection.

1.2.3.1. RSA Key

Every transparent service MAY have its own **RSA key** (secret and public pair). The public RSA key of a service SHOULD be transmitted to a SpimeGate certification service (see below §1.2.4.1). This allows:

- entities to exchange public RSA keys **via dedicated services**;
- **digital signature**, otherwise not possible.

It is up to the service owner to communicate its key or eventual key changes.

1.2.4. SpimeGates

SpimeGates are entirely based on XMPP servers, and provide therefore the **backbone** of the OpenSpime network. They additionally MAY provide a series of services, such as **certification** or **custom services**.

1.2.4.0. SpimeGate's XMPP Server

Entities authenticate on their XMPP server via their OSID (which are JID), in order to send their collected data, to spime talk, or provide various services.

Every SpimeGate MUST have its own XMPP server. SpimeGates therefore:

- connect to other SpimeGates through their respective XMPP servers;
- have spimes, ScopeNodes and services as XMPP clients.

1.2.4.1. SpimeGate's Custom Services

SpimeGate's Custom Services MAY also be provided. Examples are:

- **Digital Signature:** data transmitted by spimes and communication over the OpenSpime network MAY be digitally signed. A SpimeGate certification service MAY provide a means to publish public RSA keys for any OpenSpime network entity, via the Public Key Publishing XMPP extension (XEP-0189). The list of authorized SpimeGates providing certification services SHOULD be published on openspime.org.
- **Ranking:** algorithms may be developed to rank spimes activity (aka SpimeRank). The business logic behind these algorithms considerably varies depending on the nature of spime activities.

1.2.4.2. SpimeGate's Management Server

A server with its own web interface, allowing to:

- generate and manage **SpimeID**;
- generate and manage **SNID**;
- generate and manage **ServID**;
- for SpimeGates providing certification services, manage **entities' public RSA keys** (and optionally provide the online generation of the secret/public RSA key pair, though the secret key SHOULD NOT be kept on SpimeGate databases).

2. Security considerations

Implementations of the OpenSpime protocol should be strongly aware of the security issues related to (not exhaustively):

- storing **private RSA keys** (on entity applications, or in certification services);
- using the **RSA algorithm**:
 - weak **pseudo-random** number generation;
 - maximum **key life**.