

OPENSPIME CORE PROTOCOL AND EXTENSIONS v0.9

0. INTRODUCTION	4
0.0. OVERVIEW	4
0.1. COPYRIGHT.....	4
0.2. PERMISSIONS.....	4
0.3. DISCLAIMER OF WARRANTY	4
0.4. LIMITATION OF LIABILITY	4
0.5. RELATION TO XMPP.....	5
0.6. STATUS OF THE DOCUMENT.....	5
0.7. TERMINOLOGY	5
1. THE OPENSPIME PROTOCOL.....	6
1.0. OVERVIEW	6
1.1. THE CLAIMING CONCEPT	7
1.2. OPENSPIME CORE REFERENCE SCHEMA.....	8
1.3. ERROR RESPONSES.....	12
1.4. XML SCHEMAS	14
1.4.0. CORE PROTOCOL	14
1.4.1. CORE ERRORS.....	15
1.4.2. TRANSPORT KEY	16
1.4.3. CLAIM KEY	16
2. OPENSPIME PROTOCOL CORE EXTENSIONS	17
2.0. DATA REPORTING	17
2.0.0. DATA REPORTING – COMPLETE REFERENCE SCHEMA	17
2.0.1. DATA REPORTING – NAMESPACE REFERENCE SCHEMA	21
2.0.2. SUCCESSFUL RESPONSE	22
2.0.3. ERROR RESPONSE	22
2.0.4. XML SCHEMAS	23
2.1. CLAIM	25
2.1.0. CLAIM REQUEST – COMPLETE REFERENCE SCHEMA	25
2.1.1. CLAIM REQUEST – NAMESPACE REFERENCE SCHEMA	25
2.1.2. SUCCESSFUL CLAIM RESPONSE – COMPLETE REFERENCE SCHEMA	26
2.1.3. SUCCESSFUL CLAIM RESPONSE – NAMESPACE REFERENCE SCHEMA	27
2.1.4. ERROR RESPONSE	28
2.1.5. XML SCHEMAS	29
2.2. SPIMESEEK PROTOCOL.....	30
2.2.0. SPIMESEEK REQUEST – COMPLETE REFERENCE SCHEMA.....	31
2.2.1. SPIMESEEK LOGIC.....	33
2.2.2. SPIMESEEK REQUEST – NAMESPACE REFERENCE SCHEMA	34
2.2.3. SPIMESEEK REQUEST SUCCESSFUL ACKNOWLEDGEMENT.....	35
2.2.4. SPIMESEEK RESPONSE – COMPLETE REFERENCE SCHEMA	36



2.2.5. SPIMESEEK RESPONSE – NAMESPACE REFERENCE SCHEMA37
2.2.6. ERROR RESPONSE38
2.2.7. XML SCHEMAS39

3. APPENDIX 41

3.0. APPENDIX A: XEP-0189: PUBLIC KEY PUBLISHING..... 41

3.0.0. PUBLIC RSA KEY EXCHANGE BETWEEN ENTITIES41
3.0.1. PUBLIC RSA KEY REQUEST TO CERTIFICATION AUTHORITIES.....41
3.0.2. THE <KEYINFO/> ELEMENT FORMAT41
3.0.3. ERROR RESPONSES.....42

0. Introduction

This document illustrates the OpenSpime protocol v0.9 and its core extensions.

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 protocol v0.9 and its core extensions. It assumes that you are familiar with the Extensible Messaging and Presence Protocol (XMPP, <http://www.xmpp.org>) and with the OpenSpime architecture.

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. The OpenSpime Protocol

1.0. Overview

The OpenSpime protocol is a **set of custom XMPP protocol extensions**. The OpenSpime Core protocol defines the XMPP protocol extension that can be used to:

- **encrypt** the data content sent between two entities (end-to-end encryption);
- **digitally sign** the data content sent between two entities;
- **claim** the authority to perform operations in the name of another entity.

Note that:

- the **RSA** and **AES** algorithms SHOULD be used over the OpenSpime network for encryption and digital signature;
- the Public Key Publishing XMPP extension (XEP-0189, <http://www.xmpp.org/extensions/xep-0189.html>), SHOULD be used to manage all **public RSA key publishing**. For more information on specific OpenSpime application of this XMPP extension please refer to the Annexes.

Based on the OpenSpime Core protocol, three OpenSpime extensions have already been defined. These allow:

- **Data reporting** extension, which defines how an entity can report data to ScopeNodes;
- **Claim** extension, that defines the mechanism which authorizes an entity of the OpenSpime network to perform trustful operations in the name of another entity which has allowed it;
- **SpimeSeek** extension, which defines the process which allows seeking for other entities data across the network.

According to the XMPP protocol, the XMPP stanzas to be used as transport of the OpenSpime protocol MUST be either:

- **<iq/>** [info-query]: a **two-way request-response** mechanism. The semantics of IQ enable an entity to make a request of, and receive a response from, another entity. According to the XMPP protocol, a <iq/> stanza of type 'get' or 'set' MUST receive as response a <iq/> stanza of type 'result' or 'error', the format of which MUST comply to the XMPP core protocol as defined in RFC 3920. Therefore, when using <iq/> stanzas as transport of the OpenSpime protocol, the requesting entity MUST receive a **confirmation** for a successful operation, or a **notification** of errors provided by the core XMPP protocol (such as network errors, for instance, or the other errors defined in §9.3.3 of the RFC 3920) or custom errors.
- **<message/>**: a **one-way** communication that is being '**pushed**' from an entity to another, similar to communications that occur in emails, for instance.

1.1. The Claiming Concept

Claiming is the mechanism which authorizes an entity of the OpenSpime network to perform trustful operations in the name of another entity which has allowed it. It is similar to claiming the ownership of another entity. An entity successfully authorized to perform such operations is called an **authorized claimer**, an entity which has authorized another entity is called a **claimed entity**.

When an authorized claimer performs operations in the name of a claimed entity, it uses a **Claim Key** that was previously granted to it by the claimed entity. This key proves that it is authorized to do so.

When necessary, an authorized claimer MAY perform requests and receive responses encrypted with **its own** public RSA key, instead of the claimed entity one. This can be very useful in case scenarios such as, for instance, seeking for lost spimes.

1.2. OpenSpime Core Reference Schema

Before proceeding to a detailed description of the OpenSpime protocol and its extensions, it's essential to provide the Core OpenSpime Reference Schema in which the various elements can be located.

The **OpenSpime Core Reference Schema** is as follows (a formal description can be found in the XML Schemas section in §1.4).

```
<openspime xmlns='openspime:protocol:core' version='0.9'>
  <originator cert='{cert-servid}' osid='{originator-osid}'>
    <sign>...</sign>
    <claimkey cert='{cert-servid}' claims='{claimed-osid}'>...
  </claimkey>
  <...>
</originator>
<transport to='{recipient-osid}' content-type='{content-type}'
  transport-key='{transport-key}'>
  ...
</transport>
</openspime>
```

Node Elements are explained here below.

1.2.0.0. The `<openspime/>` element

The `<openspime/>` element is the top-parent node which qualifies the 'wrapper' namespace. This element SHOULD specify the namespace 'openspime:protocol:core', and SHOULD have the 'version' attribute set.

The `<openspime/>` element MUST be an **immediate child** of a `<iq/>` or `<message/>` stanza. There MUST be one `<openspime/>` element as top-parent node.

1.2.0.1. The `<originator/>` element

This `<originator/>` element contains identifiers and information on the originator of the message.

There MAY be one `<originator/>` element under the `<openspime/>` parent node.

This element MAY have a 'osid' attribute, which specifies the OSID of the originator of the message (aka the **originator entity**). This is useful on some occasions for which it is necessary to keep track on the OSID that first originated the message (i.e. when a message is relayed).

If the 'osid' attribute is not specified or if the element `<originator/>` does not exist, the originator entity is considered to be the entity stated in the 'from' field of the XMPP `<iq/>` or `<message/>` stanza.

This element MAY also have a 'cert' attribute, which is the **ServID** providing the originator entity's **certification services**. This ServID SHOULD be published on openspime.org to be accepted as valid certification service.

1.2.0.2. The `<sign/>` element

A `<sign/>` element contains the **digital signature** of the message contained in the `<transport/>` node. This signature is used to prove that the elements contained in the `<transport/>` node have been issued by the originator entity.

There MAY be one `<sign/>` element under the `<originator/>` parent node, in which case the 'cert' attribute of the `<originator/>` element MUST be specified.

The value of the `<sign/>` element is computed as follows.

- The **data to be signed** is the **plain text version** of the content of the <transport/> element. For instance, an example of data to be signed is:

```
<data xmlns='openspime:protocol:extension:data' version='0.9'>
  <entry>
    <date>2008-04-02T17:54:22+01:00</date>
    <exposure>outdoor</exposure>
    <lat>45.475841199050905</lat>
    <lon>9.172725677490234</lon>
    <ele unit='m'>120.0</m>
    <ppm>176.4</ppm>
  </entry>
  <entry>
    <date>2008-04-01T12:23:54+01:00</date>
    <exposure>outdoor</exposure>
    <lat>45.329189283984598</lat>
    <lon>9.154738473847384</lon>
    <ele unit='m'>102.2</m>
    <ppm>212.4</ppm>
  </entry>
</data>
```

- The XML of the above data MUST be converted to canonical form according to Canonical XML (<http://www.w3.org/TR/xml-c14n>).
- The **cryptographic hash** of this data is computed using SHA-1.
- The **resulting hash** is **encrypted** using the **private RSA key** of the **originator entity**.
- The result of the encryption is then **encoded to Base64**.

When a <sign/> element is found as child of a <originator/> element, the recipient entity SHOULD verify the signature of the originator entity. To do so, the recipient entity MUST know the public RSA key of the originator entity.

To obtain the **public RSA key** of the originator entity, the recipient entity SHOULD contact the ServID specified in the 'cert' attribute of the <originator/> element using the Public Key Publishing XMPP extension (XEP-0189). This ServID SHOULD be published on openspime.org to be accepted as valid certification service. The public RSA key of the originator entity can then be used by the recipient entity to verify the signature of the elements contained in the <transport/> node.

To avoid network overloads, it is RECOMMENDED that recipient entities keep a **cache** database of entities' public RSA keys. Should this cache database be set up, and in the case that a signature cannot be verified, a recipient entity SHOULD refresh the public RSA key of the originator entity before considering the signature invalid, since its public RSA key may have been meanwhile modified.

1.2.0.3. The <claimkey/> element

The <claimkey/> element contains the **Claim Key** of the originator entity. This Claim Key is used to prove that the originator entity is authorized to perform trustful operations in the name of a claimed entity.

This element MUST have an attribute 'claims', which specifies the **claimed entity's** OSID.

This element MUST also have an attribute 'cert', which is the **ServID** providing the claimed entity's **certification services**. This ServID SHOULD be published on openspime.org to be accepted as valid certification service.

A Claim Key results from the encryption and encoding of XML data which has the following **XML Claim Key format**:

```
<claimkey xmlns='openspime:protocol:core:claimkey' version='0.9'>
  <osid>...</osid>
  <expdate>...</expdate>
</claimkey>
```

Where:

- The <osid/> element specifies the OSID of the **originator** entity (which is the claimer entity);
- The <expdate/> element specifies the expiration date of the key. The value of the <expdate/> element MUST be in the format defined in the international standard ISO 8601, more specifically:

[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]±[hh]:[mm]

A **Claim Key** is considered **valid** when:

- the OSID of the **originator** entity corresponds to the OSID contained in the <osid/> element of the Claim Key, once decoded and decrypted;
- the **date** specified in the decrypted <expdate/> element of the Claim Key is a date in the future.

The value of a <claimkey/> element is computed as follows.

- The **data to be encrypted** is an XML which corresponds to the XML Claim Key format shown here above. For instance, an example of data to be encrypted is:

```
<key xmlns='openspime:protocol:core:claimkey' version='0.9'>
  <osid>4A6B-5638-0312111D@spime.openspime.com/spime</osid>
  <expdate>2008-12-31T23:59:59+01:00</expdate>
</key>
```

- This data is **encrypted** using the **private RSA key** of the claimed entity.
- The result of the encryption is then **encoded to Base64**.

A typical **Claim Key validation** would be:

- The recipient of a message which specifies a Claim Key MUST obtain the **public RSA key** of the **claimed** entity. To do so, the recipient SHOULD contact the **certification service** specified in the 'cert' attribute of the <claimkey/> element using the Public Key Publishing XMPP extension (XEP-0189). The ServID of this certification service SHOULD be published on openspime.org to be accepted as a valid certification service.
- The obtained public RSA key of the claimed entity MUST be used by the recipient to decode and **decrypt** the Claim Key.
- the Claim Key is tested for **validity** (i.e. that the OSID specified in the <osid/> element of the Claim Key corresponds to the originator entity's OSID, and the <expdate/> element of the Claim Key specifies a date in the future).
- Additional and **custom checks** are performed depending on the OpenSpime extensions used. For instance, when seeking for a lost spime using the SpimeSeek extension, the SpimeID of the sought spime MUST correspond to the 'claims' attribute of the <claimkey/> element for a request to be accepted (i.e. only an authorized claimer can request information on a lost spime).

To avoid network overloads, it is RECOMMENDED that recipient entities keep a **cache** database of entities' public RSA keys. Should this cache database be set up, and in the case that a Claim Key cannot be decrypted, a recipient entity SHOULD refresh the public RSA key of the claimed entity before

considering the Claim Key invalid, since the claimed entity's public RSA key may have been meanwhile modified.

There MAY be one <claimkey/> elements under the <originator/> parent node.

If a <claimkey/> element is child of a <originator/> node, the recipient entity of the message MUST validate the Claim Key and provide all the additional custom checks defined by the relevant OpenSpime protocol extensions before it can reply to the originator entity.

1.2.0.4. Custom <originator/> child elements

Additionally, some additional information on the originator MAY be passed in custom child elements of the <originator/> node.

1.2.0.5. The <transport/> element

This is the container node of the OpenSpime protocol extensions (three extensions are covered in this document: Data Reporting, Claim and SpimeSeek).

This element MAY also have a 'to' attribute, which specifies the OSID of the final recipient of the message (aka the **recipient entity**). This is useful on some occasions for which it is necessary to keep track of the intended recipient of a message (i.e. when a message is relayed).

If the 'to' attribute is not specified, the recipient entity is considered to be the entity stated in the 'from' field of the XMPP <iq/> or <message/> stanza.

This element MAY have a 'content-type' attribute which specifies the format of its own content. This attribute MUST have one of the following values:

- **text/plain**: plain text;
- **x-openspime/aes-base64**: encrypted format.

If the attribute 'content-type' is omitted, its default value MUST be considered 'text-plain'.

If the content of the <transport/> element is transmitted encrypted (i.e. using the **x-openspime/aes-base64** content-type), the value of the <transport/> element is computed as follows.

- A unique **32-bytes AES key** is generated.
- The **data to be encrypted** is the **plain text version** of the content of the <transport/> element. For instance, an example of data to be encrypted is:

```
<data xmlns='openspime:protocol:extension:data' version='0.9'>
  <entry>
    <date>2008-04-02T17:54:22+01:00</date>
    <exposure>outdoor</exposure>
    <lat>45.475841199050905</lat>
    <lon>9.172725677490234</lon>
    <ele unit='m'>120.0</m>
    <ppm>176.4</ppm>
  </entry>
  <entry>
    <date>2008-04-01T12:23:54+01:00</date>
    <exposure>outdoor</exposure>
    <lat>45.329189283984598</lat>
    <lon>9.154738473847384</lon>
    <ele unit='m'>102.2</m>
    <ppm>212.4</ppm>
  </entry>
</data>
```

- This data is **encrypted** using the **AES_256_CBC** algorithm with the newly generated **AES key** (the AES cipher SHOULD be initiated with a random initialization vector).

- The result of the encryption is then **encoded to Base64**.

If the content of the <transport/> element is transmitted encrypted (i.e. using the **x-openspime/aes-base64** content-type), the <transport/> element MUST have a 'transport-key' attribute which specifies the AES key and the initialization vector used to encrypt the data.

This 'transport-key' attribute is build as follows:

- The **data to be encrypted** is an XML with the following **XML transport key format**:

```
<transportkey xmlns='openspime:protocol:core:transportkey' version='0.9'>
  <key> ** Base64 data ** </key>
  <vint> ** Base64 data ** </vint>
</transportkey>
```

Where:

- **key** is the 32 bytes AES key, encoded in base64;
- **vint** is the random initialization vector, encoded in base64.
- This XML is **encrypted** using the **public RSA key** of the **recipient entity**, which MUST be made available to the originator entity by the recipient entity (eventually via certification services).
- The result of the encryption is then **encoded to Base64**.

There MUST be one <transport/> element under the <openspime/> parent node.

1.3. Error Responses

Any errors occurred during a data reporting operation MUST be notified via an XMPP defined condition (RFC 3920, §9.3.3) or via an application-specific <iq/> stanza error by including a properly-namespaced child in the error element of the <iq/> stanza, as defined in the XMPP core protocol (RFC 3920, §9.3.4).

The verbosity level of the response <iq/> stanza notifications depends on recipient entity implementations. However, the following condition is defined for use in stanza errors:

- <decryption-error xmlns='openspime:protocol:core:error'>
the incoming stanza was sent encrypted, though there were errors decrypting it.
- <decryption-not-enabled xmlns='openspime:protocol:core:error'>
the incoming stanza was sent encrypted, but the recipient entity is not enabled to decrypt it.
- <xml-malformed-transport-node xmlns='openspime:protocol:core:error'>
the incoming stanza has been decrypted, but the <transport/> node contains non valid xml.
- <invalid-signature xmlns='openspime:protocol:core:error'>
the incoming stanza has a signature which could not be validated.
- <signature-error-invalid-cert-auth xmlns='openspime:protocol:core:error'>
the incoming stanza has a signature certified by a certification authority not accepted by the recipient entity.
- <signature-error-public-key-corrupted xmlns='openspime:protocol:core:error'>
the incoming stanza has a signature which could not be validated because the public RSA key of the originator received from the cert authority is corrupted.
- <signature-not-enabled xmlns='openspime:protocol:core:error'>
the incoming stanza has a signature, but the recipient entity is not enabled to verify signatures.

- `<invalid-claim-key xmlns='openspime:protocol:core:error'>`
the incoming stanza has a claim key which could not be validated.
- `<claim-key-error-invalid-cert-auth xmlns='openspime:protocol:core:error'>`
the incoming stanza has a claim key certified by a certification authority not accepted by the recipient entity.
- `<claim-key-error-public-key-corrupted xmlns='openspime:protocol:core:error'>`
the incoming stanza has a claim key which could not be validated because the public RSA key of the claimer received from the cert authority is corrupted.
- `<claim-key-not-enabled xmlns='openspime:protocol:core:error'>`
the incoming stanza has a claim key, but the recipient entity is not enabled to verify claim keys.

For further information and examples of these responses, please refer to the OpenSpime Protocol Application Examples document.

1.4. XML Schemas

1.4.0. Core Protocol

Schema location: <http://openspime.org/protocol/core.xsd>

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  targetNamespace='openspime:protocol:core'
  version='0.9'
  elementFormDefault='qualified'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='openspime'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='originator' minOccurs='0'>
          <xs:complexType>
            <xs:sequence>
              <xs:element name='sign' type='xs:string' minOccurs='0' />
              <xs:element name='claimkey' minOccurs='0'>
                <xs:complexType>
                  <xs:attribute name='cert' type='xs:string' use='required' />
                  <xs:attribute name='claims' type='xs:string' use='required' />
                </xs:complexType>
              </xs:element>
              <xs:any minOccurs='0' maxOccurs='unbounded' />
            </xs:sequence>
            <xs:attribute name='cert' type='xs:string' />
            <xs:attribute name='osid' type='xs:string' />
          </xs:complexType>
        </xs:element>
        <xs:element name='transport'>
          <xs:complexType>
            <xs:sequence>
              <xs:any namespace='##other' minOccurs='0' />
            </xs:sequence>
            <xs:attribute name='to' type='xs:string' />
            <xs:attribute name='content-type'>
              <xs:simpleType>
                <xs:restriction base='xs:string'>
                  <xs:enumeration value='text/plain' />
                  <xs:enumeration value='x-openspime/rsa-base64' />
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
            <xs:attribute name='transport-key' type='xs:string' />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name='version'>
        <xs:simpleType>
          <xs:restriction base='xs:string'>
            <xs:pattern value='([0-9])+\.([0-9])+' />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

1.4.1. Core Errors

Schema location: <http://openspime.org/protocol/core/error.xsd>

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  targetNamespace='openspime:protocol:core:error'
  version='0.9'
  elementFormDefault='qualified'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='decryption-error' type='empty'/>
  <xs:element name='decryption-not-enabled' type='empty'/>
  <xs:element name='xml-malformed-transport-node' type='empty'/>
  <xs:element name='invalid-signature' type='empty'/>
  <xs:element name='signature-error-invalid-cert-auth' type='empty'/>
  <xs:element name='signature-error-public-key-corrupted' type='empty'/>
  <xs:element name='signature-not-enabled' type='empty'/>
  <xs:element name='invalid-claim-key' type='empty'/>
  <xs:element name='claim-key-error-invalid-cert-auth' type='empty'/>
  <xs:element name='claim-key-error-public-key-corrupted' type='empty'/>
  <xs:element name='claim-key-not-enabled' type='empty'/>
  <xs:group name='protocolCoreErrorGroup'>
    <xs:choice>
      <xs:element ref='decryption-error'/>
      <xs:element ref='decryption-not-enabled'/>
      <xs:element ref='xml-malformed-transport-node'/>
      <xs:element ref='invalid-signature'/>
      <xs:element ref='signature-error-invalid-cert-auth'/>
      <xs:element ref='signature-error-public-key-corrupted'/>
      <xs:element ref='signature-not-enabled'/>
      <xs:element ref='invalid-claim-key'/>
      <xs:element ref='claim-key-error-invalid-cert-auth'/>
      <xs:element ref='claim-key-error-public-key-corrupted'/>
      <xs:element ref='claim-key-not-enabled'/>
    </xs:choice>
  </xs:group>
  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

1.4.2. Transport Key

Schema location: <http://openspime.org/protocol/core/transportkey.xsd>

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  targetNamespace=openspime:protocol:core:transportkey'
  version='0.9'
  elementFormDefault='qualified'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='transportkey'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='key' type='xs:string' />
        <xs:element name='vint' type='xs:string' />
      </xs:sequence>
      <xs:attribute name='version'>
        <xs:simpleType>
          <xs:restriction base='xs:string'>
            <xs:pattern value='([0-9])+\.([0-9])+' />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

1.4.3. Claim Key

Schema location: <http://openspime.org/protocol/core/claimkey.xsd>

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  targetNamespace='openspime:protocol:core:claimkey'
  version='0.9'
  elementFormDefault='qualified'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='key'>
    <xs:complexType>
      <xs:all>
        <xs:element name='osid' type='xs:string' />
        <xs:element name='expdate' type='xs:dateTime' />
      </xs:all>
      <xs:attribute name='version'>
        <xs:simpleType>
          <xs:restriction base='xs:string'>
            <xs:pattern value='([0-9])+\.([0-9])+' />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2. OpenSpime Protocol Core Extensions

2.0. Data Reporting

This specification defines the OpenSpime protocol extension that can be used to send data. This extension will normally be used to send collected data from spimes to ScopeNodes.

2.0.0. Data Reporting – Complete Reference Schema

The `<openspime/>` element qualified by the `openspime:protocol:core` namespace MUST be one of the following:

- an **immediate child** of a `<iq/>` stanza;
- an **immediate child** of a `<message/>` stanza;
- an item published to a **PubSub** node.

Since the semantics of a `<iq/>` stanza enable an entity to make a request of, and receive a response from the recipient, it is to be used when confirmation of receipt is requested.

Using a `<message/>` stanza instead will result in reporting data without receiving any feedback on the receipt by the final recipient.

Publishing the data reporting to a PubSub node is a way to broadcast to all the list of the node's subscribers the same data with the sending of only one message. Of course in this case too, the sender will not receive feedback of receipt by recipients.

The **Complete Reference Schema** of an OpenSpime Data Reporting message using an `<iq/>` stanza is as follows.

```
<iq from='{sender-osid}'
  to='{recipient-osid}'
  xml:lang='en'
  type='set'
  id='{iq-id}'>
  <openspime xmlns='openspime:protocol:core' version='0.9'>
    <originator cert='{cert-servid}' osid='{originator-osid}'>
      <sign>...</sign>
      <claimkey cert='{cert-servid}' claims='{claimed-osid}'>...
    </claimkey>
    <.../>
  </originator>
  <transport to='{recipient-osid}' content-type='{content-type}'
    transport-key='{transport-key}'>
    <data xmlns='openspime:protocol:extension:data' version='0.9'>
      <entry>
        <date>...</date>
        <exposure>...</exposure>
        <lat>...</lat>
        <lon>...</lon>
        <ele unit='{unit-type}'>...</ele>
        <.../>
      </entry>
    </data>
  </transport>
</openspime>
</iq>
```

Where:

- the <iq/> stanza is defined by the XMPP protocol;
- the <openspime/> namespace is defined by the OpenSpime Core Reference Schema (§1.2);
- the content of the <transport/> element is defined by the OpenSpime Data Reporting Namespace Reference Schema.

The **Complete Reference Schema** of an OpenSpime Data Reporting message using a <message/> stanza is as follows.

```
<message from='{sender-osid}'
  to='{recipient-osid}'
  xml:lang='en'
  id='{iq-id}'>
  <openspime xmlns='openspime:protocol:core' version='0.9'>
    <originator cert='{cert-servid}' osid='{originator-osid}'>
      <sign>...</sign>
      <claimkey cert='{cert-servid}' claims='{claimed-osid}'>...
    </claimkey>
    <.../>
  </originator>
  <transport to='{recipient-osid}' content-type='{content-type}'
    transport-key='{transport-key}'>
    <data xmlns='openspime:protocol:extension:data' version='0.9'>
      <entry>
        <date>...</date>
        <exposure>...</exposure>
        <lat>...</lat>
        <lon>...</lon>
        <ele unit='{unit-type}'>...</ele>
        <.../>
      </entry>
    </data>
  </transport>
</openspime>
</message>
```

Where:

- the <message/> stanza is defined by the XMPP protocol;
- the <openspime/> namespace is defined by the OpenSpime Core Reference Schema (§1.2);
- the content of the <transport/> element is defined by the OpenSpime Data Reporting Namespace Reference Schema.

The **Complete Reference Schema** of an OpenSpime Data Reporting post to a XMPP PubSub service is as follows.

```
<iq from='{sender-osid}'
  to='{pubsub}'
  xml:lang='en'
  type='set'
  id='{iq-id}'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='{node-id}'>
      <item id='{item-id}'>
        <openspime xmlns='openspime:protocol:core' version='0.9'>
          <originator cert='{cert-servid}' osid='{originator-osid}'>
            <sign>...</sign>
            <claimkey cert='{cert-servid}' claims='{claimed-osid}'>...
          </claimkey>
          <.../>
        </originator>
        <transport to='{recipient-osid}' content-type='{content-type}'
          transport-key='{transport-key}'>
          <data xmlns='openspime:protocol:extension:data' version='0.9'>
            <entry>
              <date>...</date>
              <exposure>...</exposure>
              <lat>...</lat>
              <lon>...</lon>
              <ele unit='{unit-type}'>...</ele>
              <.../>
            </entry>
          </data>
        </transport>
      </openspime>
    </item>
  </publish>
</pubsub>
</iq>
```

The **Complete Reference Schema** of an OpenSpime SpimeSeek request message which is then distributed by the PubSub service to the node subscribers is as follows.

```
<message from='{sender-osid}'
  to='{pubsub}'
  xml:lang='en'
  id='{iq-id}'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='{spimeseek-node}'>
      <item id='{item-id}'>
        <openspime xmlns='openspime:protocol:core' version='0.9'>
          <originator cert='{cert-servid}' osid='{originator-osid}'>
            <sign>...</sign>
            <claimkey cert='{cert-servid}' claims='{claimed-osid}'>...
          </claimkey>
          <.../>
        </originator>
        <transport to='{recipient-osid}' content-type='{content-type}'
          transport-key='{transport-key}'>
          <data xmlns='openspime:protocol:extension:data' version='0.9'>
            <entry>
              <date>...</date>
              <exposure>...</exposure>
              <lat>...</lat>
              <lon>...</lon>
              <ele unit='{unit-type}'>...</ele>
              <.../>
            </entry>
          </data>
        </transport>
      </openspime>
    </item>
  </items>
</event>
</message>
```

Where:

- the <iq/> and <message/> stanzas are defined by the XMPP protocol;
- the <openspime/> namespace is defined by the OpenSpime Core Reference Schema (§1.2);
- the content of the <transport/> element is defined by the OpenSpime Data Reporting Namespace Reference Schema.

2.0.1. Data Reporting – Namespace Reference Schema

The `<data/>` element qualified by the `openspime:protocol:extension:data` namespace MUST be an **immediate child** of a `<transport/>` element.

The **Data Reporting** protocol extension **Reference Schema** for data transmission is as follows (a formal description can be found in the XML Schema section in §2.0.4.0).

```
<data xmlns='openspime:protocol:extension:data' version='0.9'>
  <entry>
    <date>...</date>
    <exposure>...</exposure>
    <lat>...</lat>
    <lon>...</lon>
    <ele unit='{unit-type}'>...</ele>
    <.../>
  </entry>
</data>
```

Node Elements are explained here below.

2.0.1.0. The `<data/>` element

The `<data/>` element is the node which qualifies the Data Reporting extension 'wrapper' namespace. This element SHOULD specify the namespace 'openspime:protocol:extension:data', and SHOULD have the 'version' attribute set.

There MUST be zero or one `<data/>` elements under the `<transport/>` parent node. Should a `<data/>` element exist under the `<transport/>` parent node, there SHOULD NOT be other `<transport/>` child nodes; if others were to exist, only the first child node of the `<transport/>` element SHOULD be considered by the receiving entity.

2.0.1.1. The `<entry/>` element

This is the container node for all individual data elements of a single entry. The content and format of the `<entry/>` elements are specified by the recipient entity, thus the OpenSpime Data Reporting protocol extension only specifies the optional child nodes shown in the base syntax here above.

There MAY be one or more `<entry/>` elements under the `<entries/>` parent node.

2.0.1.2. The `<date/>` element

This is the entry's data element node which specifies the date. The value of the `<date/>` element MUST be in the format defined in the international standard ISO 8601, more specifically:

[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]±[hh]:[mm]

There MAY be one `<date/>` element under a single `<entry/>` node.

2.0.1.3. The `<exposure/>` element

This is the entry's data element node which specifies the type of originator entity's exposure while collecting the data. This value provides qualitative data only. The value of this element MUST be one of the following:

- indoor
- outdoor
- liquid
- solid
- gas

- plasma
- vacuum

There MAY be one <exposure/> element under a single <entry/> node.

2.0.1.4. The <lat/> element

This is the entry's data element node which specifies the latitude GPS coordinates of the data collection. The value MUST be expressed in decimal form.

There MAY be one <lat/> element under a single <entry/> node.

2.0.1.5. The <lon/> element

This is the entry's data element node which specifies the longitude GPS coordinates of the data collection. The value MUST be expressed in decimal form.

There MAY be one <lon/> element under a single <entry/> node.

2.0.1.6. The <ele/> element

This is the entry's data element node which specifies the elevation coordinate of the data collection. This element MUST have a 'unit' attribute which MUST have one of the following values:

- **m**: for values expressed in meters;
- **f**: for values expressed in feet.

There MAY be one <ele/> element under a single <entry/> node.

2.0.1.7. Custom <entry/> child elements

As previously stated, the <entry/> node is the container node for all individual data elements of a single entry, the format of which is defined by the recipient entity. The OpenSpime Data Reporting protocol extension only specifies the optional child nodes defined here above, and total freedom is left to the recipient to define other <entry/> child nodes.

It is RECOMMENDED to define these custom elements as direct child nodes of the <entry/> element.

2.0.2. Successful Response

If a <iq/> stanza has been used to send the data, an entity notifies the **successful** reception of data via an empty <iq/> stanza.

2.0.3. Error Response

Any errors occurred during a data reporting operation MUST be notified via an XMPP defined condition (RFC 3920, §9.3.3) or via an application-specific <iq/> stanza error by including a properly-namespaced child in the error element of the <iq/> stanza, as defined in the XMPP core protocol (RFC 3920, §9.3.4).

The verbosity level of the response <iq/> stanza notifications depends on recipient entity implementations. However, the following condition is defined for use in stanza errors:

- <inconsistent-data-with-scope xmlns='openspime:protocol:extension:data:error'> the format of one or more <entry/> elements does not correspond to the data format accepted by the recipient entity.

For further information and examples of these responses, please refer to the OpenSpime Protocol Application Examples document.

2.0.4. XML Schemas

2.0.4.0. Data Reporting

Schema location: <http://openspime.org/protocol/extension/data.xsd>

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  targetNamespace='openspime:protocol:extension:data'
  version='0.9'
  elementFormDefault='qualified'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='data'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='entry' minOccurs='0' maxOccurs='unbounded'>
          <xs:complexType>
            <xs:sequence>
              <xs:element name='date' type='xs:dateTime' minOccurs='0' />
              <xs:element name='exposure' minOccurs='0'>
                <xs:simpleType>
                  <xs:restriction base='xs:string'>
                    <xs:enumeration value='outdoor' />
                    <xs:enumeration value='liquid' />
                    <xs:enumeration value='solid' />
                    <xs:enumeration value='gas' />
                    <xs:enumeration value='plasma' />
                    <xs:enumeration value='vacuum' />
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name='lat' type='xs:decimal' minOccurs='0' />
              <xs:element name='lon' type='xs:decimal' minOccurs='0' />
              <xs:element name='ele' minOccurs='0'>
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute name='unit'>
                        <xs:simpleType>
                          <xs:restriction base='xs:string'>
                            <xs:enumeration value='m' />
                            <xs:enumeration value='f' />
                          </xs:restriction>
                        </xs:simpleType>
                      </xs:attribute>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            <xs:any minOccurs='0' maxOccurs='unbounded' processContents="lax" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name='version'>
      <xs:simpleType>
        <xs:restriction base='xs:string'>
          <xs:pattern value='([0-9])+\.([0-9])+' />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
</xs:schema>
```

2.0.4.1. Data Reporting Errors

Schema location: <http://openspime.org/protocol/extension/data/error.xsd>

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  targetNamespace='openspime:protocol:extension:data:error'
  version='0.9'
  elementFormDefault='qualified'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='inconsistent-data-with-scope' type='empty'/>
  <xs:group name='protocolExtensionDataReportingErrorGroup'>
    <xs:choice>
      <xs:element ref='inconsistent-data-with-scope'/>
    </xs:choice>
  </xs:group>
  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

2.1. Claim

This specification defines the OpenSpime protocol extension that can be used to claim an entity.

2.1.0. Claim Request – Complete Reference Schema

The `<openspime/>` element qualified by the `openspime:protocol:core` namespace MUST be an **immediate child** of a `<iq/>` stanza.

The **Complete Reference Schema** of an OpenSpime Claim request message is as follows.

```
<iq from='{sender-osid}'
  to='{recipient-osid}'
  xml:lang='en'
  type='set'
  id='{iq-id}'>
  <openspime xmlns='openspime:protocol:core' version='0.9'>
    <originator cert='{cert-servid}' osid='{originator-osid}'>
      <sign>...</sign>
      <claimkey cert='{cert-servid}' claims='{claimed-osid}'>...
    </claimkey>
    <.../>
  </originator>
  <transport to='{recipient-osid}' content-type='{content-type}'
    transport-key='{transport-key}'>
    <claim xmlns='openspime:protocol:extension:claim' version='0.9'>
      <request to='{claimed-osid}'>
        <expdate>...</expdate>
      </request>
    </claim>
  </transport>
</openspime>
</iq>
```

Where:

- the `<iq/>` stanza is defined by the XMPP protocol;
- the `<openspime/>` namespace is defined by the OpenSpime Core Reference Schema (§1.2);
- the content of the `<transport/>` element is defined by the OpenSpime Claim Request Namespace Reference Schema.

2.1.1. Claim Request – Namespace Reference Schema

The `<claim/>` element qualified by the `openspime:protocol:extension:claim` namespace MUST be an **immediate child** of a `<transport/>` element.

The **Claim** protocol extension **Reference Schema** for a Claim request is as follows (a formal description can be found in the XML Schema section in §2.1.5.0).

```
<claim xmlns='openspime:protocol:extension:claim' version='0.9'>
  <request claims='{claimed-osid}'>
    <expdate>...</expdate>
  </request>
</claim>
```

Node Elements are explained here below.

2.1.1.0. The <claim/> element

The <claim/> element is the node which qualifies the Claim extension 'wrapper' namespace. This element SHOULD specify the namespace 'openspime:protocol:extension:claim', and SHOULD have the 'version' attribute set.

There MUST be zero or one <claim/> elements under the <transport/> parent node. Should a <claim/> element exist under the <transport/> parent node, there SHOULD NOT be other <transport/> child nodes; if others were to exist, only the first child node of the <transport/> element SHOULD be considered by the receiving entity.

2.1.1.1. The <request/> element

This node specifies the request nature of the message.

This element MUST have a 'claims' attribute which specifies the OSID of the entity which is being claimed.

There MUST be zero or one <request/> elements under the <claim/> parent node. Should a <request/> element exist under the <claim/> parent node, there SHOULD NOT be other <claim/> child nodes; if others were to exist, only the first child node of the <claim/> element SHOULD be considered by the receiving entity.

2.1.1.2. The <expdate/> element

This is the element node which specifies the **desired** expiration date of the Claim Key. There MAY be one <expdate/> element under the <request/> node.

Please note that, if an <expdate/> element has been specified, the claimed entity MAY ignore the desired expiration date request and authorize the requestor entity with a different expiration date. It is therefore RECOMMENDED that a claimer entity verifies which expiration date really is contained in a received Claim Key upon receipt.

The value of the <expdate/> element MUST be in the format defined in the international standard ISO 8601, more specifically:

[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]±[hh]:[mm]

2.1.2. Successful Claim Response – Complete Reference Schema

The <openspime/> element qualified by the *openspime:protocol:core* namespace MUST be an **immediate child** of a <iq/> stanza.

The **Complete Reference Schema** of an OpenSpime successful Claim response message is as follows.

```
<iq from='{sender-osid}'
  to='{recipient-osid}'
  xml:lang='en'
  type='result'
  id='{iq-id}'>
  <openspime xmlns='openspime:protocol:core' version='0.9'>
    <originator cert='{cert-servid}' osid='{originator-osid}'>
      <sign>...</sign>
      <claimkey cert='{cert-servid}' claims='{claimed-osid}'>...
    </claimkey>
    <.../>
  </originator>
  <transport to='{recipient-osid}' content-type='{content-type}'
    transport-key='{transport-key}'>
    <claim xmlns='openspime:protocol:extension:claim' version='0.9'>
      <response authorizes='{requestor-osid}'>
        <claimkey>...</claimkey>
      </response>
    </claim>
  </transport>
</openspime>
</iq>
```

Where:

- the <iq/> stanza is defined by the XMPP protocol;
- the <openspime/> namespace is defined by the OpenSpime Core Reference Schema (§1.2);
- the content of the <transport/> element is defined by the OpenSpime Claim Response Namespace Reference Schema.

2.1.3. Successful Claim Response – Namespace Reference Schema

The <claim/> element qualified by the *openspime:protocol:extension:claim* namespace MUST be an **immediate child of a <transport/> element**.

The **Claim** protocol extension **Reference Schema** for a successful Claim response is as follows (a formal description can be found in the XML Schema section in §2.1.5.0).

```
<claim xmlns='openspime:protocol:extension:claim' version='0.9'>
  <response authorizes='{requestor-osid}'>
    <claimkey>...</claimkey>
  </response>
</claim>
```

Node Elements are explained here below.

2.1.3.0. The <claim/> element

The <claim/> element is the node which qualifies the Claim extension 'wrapper' namespace. This element SHOULD specify the namespace 'openspime:protocol:extension:claim', and SHOULD have the 'version' attribute set.

There MUST be zero or one <claim/> elements under the <transport/> parent node. Should a <claim/> element exist under the <transport/> parent node, there SHOULD NOT be other <transport/> child

nodes; if others were to exist, only the first child node of the <transport/> element SHOULD be considered by the receiving entity.

2.1.3.1. The <response/> element

This node specifies the response nature of the message. This element MUST have a 'authorizes' attribute which specifies the OSID of the entity which becomes an authorized claimer.

There MUST be zero or one <response/> elements under the <claim/> parent node. Should a <response/> element exist under the <openspime/> parent node, there SHOULD NOT be other <claim/> child nodes; if others were to exist, only the first child node of the <claim/> element SHOULD be considered by the receiving entity.

2.1.3.2. The <claimkey/> element

The <claimkey/> element contains the **Claim Key** which empowers the claim requestor (i.e. the originator entity) to become an authorized claimer.

For more information on **Claim Keys** please refer to §1.2.0.3.

It is RECOMMENDED that an entity receiving a Claim Key verifies its contents to ensure that the received key is valid, and to acknowledge the key's expiration date.

There MUST be one <claimkey/> element under the <response/> parent node.

2.1.4. Error Response

Any errors occurred in the claiming process MUST be notified via an XMPP defined condition (RFC 3920, §9.3.3) or via an application-specific <iq/> stanza error by including a properly-namespaced child in the error element of the <iq/> stanza, as defined in the XMPP core protocol (RFC 3920, §9.3.4).

The verbosity level of the response <iq/> stanza notifications depends on entity implementations.

For further information and examples of these responses, please refer to the OpenSpime Protocol Application Examples document.

2.1.5. XML Schemas

2.1.5.0. Claim

Schema location: <http://openspime.org/protocol/extension/claim.xsd>

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  targetNamespace='openspime:protocol:extension:claim'
  version='0.9'
  elementFormDefault='qualified'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='claim'>
    <xs:complexType>
      <xs:choice>
        <xs:element name='request'>
          <xs:complexType>
            <xs:sequence>
              <xs:element name='expdate' type='xs:dateTime' minOccurs='0' />
            </xs:sequence>
            <xs:attribute name='claims' type='xs:string' use='required' />
          </xs:complexType>
        </xs:element>
        <xs:element name='response'>
          <xs:complexType>
            <xs:sequence>
              <xs:element name='claimkey' type='xs:string' />
            </xs:sequence>
            <xs:attribute name='authorizes' type='xs:string' use='required' />
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:attribute name='version'>
        <xs:simpleType>
          <xs:restriction base='xs:string'>
            <xs:pattern value='([0-9])+\.([0-9])+' />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2.2. SpimeSeek Protocol

This specification defines the OpenSpime protocol extension that can be used to seek for spime localization. Seeking for spime localization is a process which originates from a single entity and spreads across the OpenSpime network.

An entity generates a SpimeSeek request by publishing it on its SpimeGate's XMPP *spimeseek* PubSub node.

In accordance to XMPP Publish-Subscribe Extension Specifications (XEP-0060, <http://www.xmpp.org/extensions/xep-0060.html>), and in order to provide homogeneity in SpimeSeek services' naming, it is RECOMMENDED that the PubSub service of a SpimeGate's XMPP is located at the address *pubsub.mydomain.com* and the addressable PubSub node is named *spimeseek*. The PubSub service SHOULD be configured to include **payloads**.

When an entity publishes a SpimeSeek request to a *spimeseek* PubSub node, the request is then distributed to the PubSub subscribers list, which SHOULD normally consist of:

- primarily **ScopeNodes**, which are the entities holding data and therefore the only entities enabled to provide a response;
- **relaying services**, i.e. services that receive a SpimeSeek request and forward it to another *spimeseek* PubSub, a mechanism which allows for fast and wide propagation of the SpimeSeek requests across the OpenSpime network.

When a ScopeNode receives a SpimeSeek request, it SHOULD verify that:

- the request has **not already been treated**;
- the **expiration date** of the request is a date in the future;
- the **digital signature** of the request is valid (i.e. is of the sought spime or of an authorized claimer).

Only if the above conditions are met, the ScopeNode SHOULD, if available data is found, provide a **response** to the request originator.

2.2.0. SpimeSeek Request – Complete Reference Schema

The **Complete Reference Schema** of an OpenSpime SpimeSeek request post to a XMPP PubSub service is as follows.

```
<iq from='{sender-osid}'
  to='{pubsub}'
  xml:lang='en'
  type='set'
  id='{iq-id}'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='{node-id}'>
      <item id='{item-id}'>
        <openspime xmlns='openspime:protocol:core' version='0.9'>
          <originator cert='{cert-servid}' osid='{originator-osid}'>
            <sign>...</sign>
            <claimkey cert='{cert-servid}' claims='{claimed-osid}'>...
          </claimkey>
          <.../>
        </originator>
        <transport to='{recipient-osid}' content-type='{content-type}'>
          <spimeseek xmlns='http://openspime.org/protocol/spimeseek'
            version='0.9'>
            <request id='{seek-id}'>
              <spimeid>...</spimeid>
              <expdate>...</expdate>
              <encrequired>...</encrequired>
            </request>
          </spimeseek>
        </transport>
      </openspime>
    </item>
  </publish>
</pubsub>
</iq>
```

Please note that the content of the SpimeSeek request SHOULD be transmitted unencrypted, i.e. the 'content-type' attribute of the <transport/> node SHOULD NOT be set or SHOULD have 'text-plain' as value, so that every recipient is enabled to read the request.

The **Complete Reference Schema** of an OpenSpime SpimeSeek request message which is then distributed by the PubSub service to the node subscribers is as follows.

```
<message from='{sender-osid}'
  to='{pubsub}'      xml:lang='en'
  id='{iq-id}'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='{spimeseek-node}'>
      <item id='{item-id}'>
        <openspime xmlns='openspime:protocol:core' version='0.9'>
          <originator cert='{cert-servid}' osid='{originator-osid}'>
            <sign>...</sign>
            <claimkey cert='{cert-servid}' claims='{claimed-osid}'>...
          </claimkey>
          <.../>
        </originator>
        <transport to='{recipient-osid}' content-type='{content-type}'>
          <spimeseek xmlns='http://openspime.org/protocol/spimeseek'
            version='0.9'>
            <request id='{seek-id}'>
              <spimeid>...</spimeid>
              <expdate>...</expdate>
              <encrequired>...</encrequired>
            </request>
          </spimeseek>
        </transport>
      </openspime>
    </item>
  </items>
</event>
</message>
```

Where:

- the <iq/> and <message/> stanzas are defined by the XMPP protocol;
- the <openspime/> namespace is defined by the OpenSpime Core Reference Schema (§1.2);
- the content of the <transport/> element is defined by the OpenSpime SpimeSeek Request Namespace Reference Schema.

2.2.1. SpimeSeek Logic

Two possible case scenarios of SpimeSeek usage follow.

- A SpimeSeek request is originated by a **spime** which is requesting localization information regarding itself, in which case it MUST:
 - sign its SpimeSeek request with its digital signature (i.e. there MUST be a <sign/> element under the <originator/> node);
 - set its own SpimeID as value of the 'osid' attribute of the <originator/> element (so that all responses are sent to itself).
- A SpimeSeek request is originated by an **authorized claimer** which is requesting localization information regarding a claimed spime, in which case it MUST:
 - sign its SpimeSeek request with its own digital signature (i.e. there MUST be a <sign/> element under the <originator/> node);
 - include a Claim Key (i.e. there MUST be a <claimkey/> element under the <originator/> parent node) which certifies that the originator is authorized by the sought spime;
 - set its own OSID as value of the 'osid' attribute of the <originator/> element (so that all responses are sent to itself).

The ScopeNodes receiving a SpimeSeek request MUST verify that it is **valid** before providing a response. Therefore:

- **If the request is originated by the sought spime**
 - It is the sought spime which signs the SpimeSeek request. Therefore, the <claimkey/> element of the <openspime/> parent node has not been specified.
 - The ScopeNode MUST verify that the OSID of the originator spime (i.e. the value of the 'osid' attribute of the <originator/> element) corresponds to the SpimeID specified in the <spimeid/> element of the <request/> node. If these do not correspond, the ScopeNode MUST NOT treat the SpimeSeek request.
 - The ScopeNode MUST then obtain the **public RSA key** of the **originator** spime. To do so, it SHOULD contact the ServID specified in the 'cert' attribute of the <originator/> element using the Public Key Publishing XMPP extension (XEP-0189). This ServID SHOULD be published on openspime.org to be accepted as a valid certification service.
 - The obtained public RSA key of the originator spime MUST be used by the ScopeNode to verify the **digital signature**. If the signature cannot be verified, the ScopeNode MUST NOT treat the SpimeSeek request.

➤ **If the request is originated by an authorized claimer of the sought spime**

- It is a claimer, which still needs to be verified, that signs the SpimeSeek request. Therefore, the `<claimkey/>` element of the `<openspime/>` node has been specified.
- The ScopeNode MUST verify that the **claimed entity** corresponds to the **sought spime**, i.e. that the OSID specified in the 'claims' attribute of the `<claimkey/>` element of the `<originator/>` parent node is equal to the OSID specified in the `<spimeid/>` element of the `<request/>` parent node. If these do not correspond, the ScopeNode MUST NOT treat the SpimeSeek request.
- The ScopeNode MUST then obtain the **public RSA key** of the **claimed entity** specified in the 'claims' attribute of the `<claimkey/>` element. To do so, it SHOULD contact the ServID specified in the 'cert' attribute of the `<claimkey/>` element using the Public Key Publishing XMPP extension (XEP-0189). This ServID SHOULD be published on openspime.org to be accepted as a valid certification service.
- The obtained public RSA key of the claimed entity MUST be used by the ScopeNode to decode and decrypt the **Claim Key**, which allows to verify that the originator of the SpimeSeek request is an **authorized claimer** of the sought spime (i.e. that the OSID specified in the value of the 'osid' attribute of the `<originator/>` element corresponds to the value of the `<osid/>` contained in the decrypted Claim Key, and that the `<expdate/>` element of the Claim Key specifies a date in the future). If the validity of the Claim Key cannot be verified, the ScopeNode MUST NOT treat the SpimeSeek request.
- The ScopeNode MUST then obtain the **public RSA key** of the **originator** entity. To do so, it SHOULD contact the ServID specified in the 'cert' attribute of the `<originator/>` element using the Public Key Publishing XMPP extension (XEP-0189). This ServID SHOULD be published on openspime.org to be accepted as a valid certification service.
- The obtained public RSA key of the originator entity MUST then be used by the ScopeNode to verify the **digital signature** of the SpimeSeek request. If the validity of the signature cannot be verified, the ScopeNode MUST NOT treat the SpimeSeek request.

2.2.2. SpimeSeek Request – Namespace Reference Schema

The `<spimeseek/>` element qualified by the `http://openspime.org/protocol/spimeseek` namespace MUST be an **immediate child** of a `<transport/>` element.

The **SpimeSeek** protocol extension **Reference Schema** for a SpimeSeek request is as follows (a formal description can be found in the XML Schema section in §2.2.7).

```
<spimeseek xmlns='http://openspime.org/protocol/spimeseek' version='0.9'>
  <request id='{seek-id}'>
    <spimeid>...</spimeid>
    <expdate>...</expdate>
    <encrequired>...</encrequired>
  </request>
</spimeseek>
```

Node Elements are explained here below.

2.2.2.0. The `<spimeseek/>` element

The `<spimeseek/>` element is the node which qualifies the SpimeSeek extension 'wrapper' namespace. This element SHOULD specify the namespace 'http://openspime.org/protocol/spimeseek', and SHOULD have the 'version' attribute set.

There MUST be zero or one <spimeseek/> elements under the <transport/> parent node. Should a <spimeseek/> element exist under the <transport/> parent node, there SHOULD NOT be other <transport/> child nodes; if others were to exist, only the first child node of the <transport/> element SHOULD be considered by the receiving entity.

2.2.2.1. The <request/> element

This node specifies the request nature of the message.

This element MUST have the 'id' attribute, which is an identifier which should be used together with the originator entity's OSID as **primary SpimeSeek identifier**.

There MUST zero or one <request/> elements under the <spimeseek/> parent node. Should a <request/> element exist under the <spimeseek/> parent node, there SHOULD NOT be other <spimeseek/> child nodes; if others were to exist, only the first child node of the <spimeseek/> element SHOULD be considered by the receiving entity.

2.2.2.2. The <spimeid/> element

This element MUST have a valid SpimeID as value. This is the spime that will be searched for. There MUST be one <spimeid/> element child of the <request/> node.

2.2.2.3. The <expdate/> element

This is the element node which specifies the expiration date of the search. The date specified in this element is indicative only, since it is up to the ScopeNode owners to decide when a search should not be treated. It is RECOMMENDED that this value should not be set more than 24 hours from the date when the search is first initiated.

The value of the <expdate/> element MUST be in the format defined in the international standard ISO 8601, more specifically:

[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]±[hh]:[mm]

There MUST be one <expdate/> element under the <request/> node.

2.2.2.4. The <encrequired/> element

This element specifies if the response MUST, MUST NOT, or SHOULD be encrypted. The value of this element MUST be one of the following:

- **true**: if the response MUST be encrypted;
- **false**: if the response MUST be not encrypted;
- **preferred**: if the response SHOULD be encrypted but, if encryption is not available, an unencrypted response is accepted.

There MUST be one <encrequired/> element child of the <request/> node.

2.2.3. SpimeSeek Request Successful Acknowledgement

A PubSub service notifies the **successful** reception of a SpimeSeek request via an empty <iq/> stanza, as defined by the XMPP Publish-Subscribe Extension Specifications (XEP-0060).

2.2.4. SpimeSeek Response – Complete Reference Schema

This is the format of the response sent by a ScopeNode which has received a SpimeSeek request, has validated it, and has found localization data for the sought spime.

The `<openspime/>` element qualified by the `openspime:protocol:core` namespace MUST be an **immediate child** of a `<message/>` stanza.

The **Complete Reference Schema** of an OpenSpime successful SpimeSeek response message is as follows.

```
<message from='{sender-osid}'
  to='{recipient-osid}'
  xml:lang='en'
  id='{message-id}'>
  <openspime xmlns='openspime:protocol:core' version='0.9'>
    <originator cert='{cert-servid}' osid='{originator-osid}'>
      <sign>...</sign>
      <claimkey cert='{cert-servid}' claims='{claimed-osid}'>...
    </claimkey>
    <.../>
  </originator>
  <transport to='{recipient-osid}' content-type='{content-type}'
    transport-key='{transport-key}'>
    <spimeseek xmlns='http://openspime.org/protocol/spimeseek'
      version='0.9'
      <response id='{seek-id}'>
        <entry>
          <date>...</date>
          <exposure>...</exposure>
          <lat>...</lat>
          <lon>...</lon>
          <ele unit='{unit-type}'>...</ele>
        </entry>
      </response>
    </spimeseek>
  </transport>
</openspime>
</message>
```

Where:

- the `<message/>` stanza is defined by the XMPP protocol;
- the `<openspime/>` namespace is defined by the OpenSpime Core Reference Schema (§1.2);
- the content of the `<transport/>` element is defined by the OpenSpime SpimeSeek Response Namespace Reference Schema.

2.2.5. SpimeSeek Response – Namespace Reference Schema

The <spimeseek/> element qualified by the *http://openspime.org/protocol/spimeseek* namespace MUST be an **immediate child** of a <transport/> element.

The **SpimeSeek** protocol extension **Reference Schema** for a successful SpimeSeek response is as follows (a formal description can be found in the XML Schema section in §2.2.7).

```
<spimeseek xmlns='http://openspime.org/protocol/spimeseek' version='0.9'>
  <response id='{seek-id}'>
    <entry>
      <date>...</date>
      <exposure>...</exposure>
      <lat>...</lat>
      <lon>...</lon>
      <ele unit='{unit-type}'>...</ele>
    </entry>
  </response>
</spimeseek>
```

Node Elements are explained here below.

2.2.5.0. The <spimeseek/> element

The <spimeseek/> element is the node which qualifies the SpimeSeek extension 'wrapper' namespace. This element SHOULD specify the namespace 'http://openspime.org/protocol/spimeseek', and SHOULD have the 'version' attribute set.

There MUST be zero or one <spimeseek/> elements under the <transport/> parent node. Should a <spimeseek/> element exist under the <transport/> parent node, there SHOULD NOT be other <transport/> child nodes; if others were to exist, only the first child node of the <transport/> element SHOULD be considered by the receiving entity.

2.2.5.1. The <response/> element

This node specifies the response nature of the message. This element MUST have the attributes 'id', , which is an identifier that should be used together with the recipient entity's OSID as **primary SpimeSeek identifier**.

There MUST zero or one <response/> elements under the <spimeseek/> parent node. Should a <response/> element exist under the <spimeseek/> parent node, there SHOULD NOT be other <spimeseek/> child nodes; if others were to exist, only the first child node of the <spimeseek/> element SHOULD be considered by the receiving entity.

2.2.5.2. The <entry/> element

This is the container node for all individual localization elements of a single entry.

There MAY be one or more <entry/> elements under the <entries/> parent node. A maximum number of 5 <entry/> elements is RECOMMENDED.

2.2.5.3. The <date/> element

This is the entry's data element node which specifies the date. The value of the <date/> element MUST be in the format defined in the international standard ISO 8601, more specifically:

[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]±[hh]:[mm]

There MAY be one <date/> element under a single <entry/> node.

2.2.5.4. The <exposure/> element

This is the entry's data element node which specifies the type of spime's exposure while collecting the data. This value provides qualitative data only. The value of this element MUST be one of the following:

- indoor
- outdoor
- liquid
- solid
- gas
- plasma
- vacuum

There MAY be one <exposure/> element under a single <entry/> node.

2.2.5.5. The <lat/> element

This is the entry's data element node which specifies the latitude GPS coordinates of the data collection. The value MUST be expressed in decimal form.

There MAY be one <lat/> element under a single <entry/> node.

2.2.5.6. The <lon/> element

This is the entry's data element node which specifies the longitude GPS coordinates of the data collection. The value MUST be expressed in decimal form.

There MAY be one <lon/> element under a single <entry/> node.

2.2.5.7. The <ele/> element

This is the entry's data element node which specifies the elevation coordinate of the data collection. This element MUST have a 'unit' attribute which MUST have one of the following values:

- **m**: for values expressed in meters;
- **f**: for values expressed in feet.

There MAY be one <ele/> element under a single <entry/> node.

2.2.6. Error Response

Any errors occurred during a SpimeSeek request are reported according to the XMPP Publish-Subscribe Extension Specifications (XEP-0060).

Eventually, additional errors MAY be reported via an application-specific <iq/> stanza error by including a properly-namespaced child in the error element of the <iq/> stanza, as defined in the XMPP core protocol (RFC 3920, §9.3.4).

The verbosity level of the response <iq/> stanza notifications depends on recipient entity implementations. However, the following condition is defined for use in stanza errors:

- `<encryption-not-allowed xmlns='openspime:protocol:extension:spimeseek:error'>`
the SpimeSeek request was sent encrypted, but multiple recipients cannot receive a privately encrypted message.
- `<no-signature xmlns='openspime:protocol:extension:spimeseek:error'>`
the SpimeSeek request was sent without signature.

2.2.7. XML Schemas

2.2.7.0. SpimeSeek

Schema location: <http://openspime.org/protocol/extension/spimeseek.xsd>

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  targetNamespace='openspime:protocol:extension:spimeseek'
  version='0.9'
  elementFormDefault='qualified'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='spimeseek'>
    <xs:complexType>
      <xs:choice>
        <xs:element name='request'>
          <xs:complexType>
            <xs:sequence>
              <xs:element name='spimeid' type='xs:string' />
              <xs:element name='expdate' type='xs:dateTime' />
              <xs:element name='encrequired'>
                <xs:simpleType>
                  <xs:restriction base='xs:string'>
                    <xs:enumeration value='true' />
                    <xs:enumeration value='false' />
                    <xs:enumeration value='preferred' />
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name='id' type='xs:string' use='required' />
          </xs:complexType>
        </xs:element>
        <xs:element name='response'>
          <xs:complexType>
            <xs:sequence>
              <xs:element name='entry' minOccurs='0' maxOccurs='unbounded'>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name='date' type='xs:dateTime' minOccurs='0' />
                    <xs:element name='exposure' minOccurs='0'>
                      <xs:simpleType>
                        <xs:restriction base='xs:string'>
                          <xs:enumeration value='outdoor' />
                          <xs:enumeration value='liquid' />
                          <xs:enumeration value='solid' />
                          <xs:enumeration value='gas' />
                          <xs:enumeration value='plasma' />
                          <xs:enumeration value='vacuum' />
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name='lat' type='xs:decimal' minOccurs='0' />
                    <xs:element name='lon' type='xs:decimal' minOccurs='0' />
                    <xs:element name='ele' minOccurs='0'>
                      <xs:complexType>
                        <xs:attribute name='unit'>
                          <xs:simpleType>
                            <xs:restriction base='xs:string'>
                              <xs:enumeration value='m' />
                              <xs:enumeration value='f' />
                            </xs:restriction>
                          </xs:simpleType>
                        </xs:complexType>
                      </xs:element>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

```
        </xs:attribute>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name='id' type='xs:string' use='required' />
</xs:complexType>
</xs:element>
</xs:choice>
<xs:attribute name='version'>
  <xs:simpleType>
    <xs:restriction base='xs:string'>
      <xs:pattern value='([0-9])+\.([0-9])+' />
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:schema>
```

2.2.7.1. SpimeSeek Errors

Schema location: <http://openspime.org/protocol/extension/spimeseek/error.xsd>

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  targetNamespace='openspime:protocol:extension:spimeseek:error'
  version='0.9'
  elementFormDefault='qualified'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='encryption-not-allowed' type='empty' />
  <xs:element name='no-signature' type='empty' />
  <xs:group name='protocolExtensionSpimeSeekErrorGroup'>
    <xs:choice>
      <xs:element ref='encryption-not-allowed' />
      <xs:element ref='no-signature' />
    </xs:choice>
  </xs:group>
  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value='' />
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

3. Appendix

3.0. Appendix A: XEP-0189: Public Key Publishing

The Public Key Publishing XMPP extension (XEP-0189, <http://www.xmpp.org/extensions/xep-0189.html>), SHOULD be used to manage all **public RSA key publishing**.

3.0.0. Public RSA key exchange between entities

All possible key exchanges managed by the XEP-0189 XMPP extension can be used:

- Public Key Publication and Retrieval via PEP
- Requesting Public Keys Directly From Another Entity
- Requesting Public Keys Directly From a Third Party
- Sending Public Keys Directly To Another Entity

3.0.1. Public RSA key request to certification authorities

To request public RSA keys from a certification authority, the *Requesting Public Keys Directly From a Third Party* protocol MUST be used.

3.0.2. The <KeyInfo/> element format

The <KeyInfo/> element SHOULD have the following format:

```
<KeyInfo xmlns='http://www.w3.org/2000/09/xmldsig#'>
  <KeyName></KeyName>
  <KeyValue>
    <RSAKeyValue>
      <Modulus>...</Modulus>
      <Exponent>AQAB</Exponent>
    </RSAKeyValue>
  </KeyValue>
</KeyInfo>
```

Node Elements are explained here below.

3.0.2.0. The <KeyInfo/> element

The <KeyInfo/> element is the top-parent node which qualifies the 'key' namespace.

There MUST be one <KeyInfo/> element as top-parent node.

3.0.2.1. The <KeyName/> element

This <KeyName/> element contains the key name identifier for the key.

There MAY be one <KeyName/> element under the <keyInfo/> parent node.

3.0.2.2. The <RSAKeyValue/> element

This <RSAKeyValue/> element is the container node for the public RSA key data.

There MUST be one <RSAKeyValue/> element under the <keyInfo/> parent node.

3.0.2.3. The <Exponent/> element

The modulus of the public RSA Key, encoded in base64, MUST be the value of the <Modulus/> element. There MUST be one <Modulus/> element under the <RSAKeyValue/> parent node.

3.0.2.4. The <Modulus/> element

The exponent of the public RSA Key, encoded in base64, MUST be the value of the <Exponent/> element.

There MUST be one <Exponent/> element under the <RSAKeyValue/> parent node.

3.0.3. Error Responses

Any errors occurred during a data reporting operation MUST be notified via an XMPP defined condition (RFC 3920, §9.3.3) or via an application-specific <iq/> stanza error by including a properly-namespaced child in the error element of the <iq/> stanza, as defined in the XMPP core protocol (RFC 3920, §9.3.4).

For further information and examples of these responses, please refer to the OpenSpime Protocol Application Examples document.