

# **OPENSPIME CORE PROTOCOL V0.9 APPLICATION EXAMPLES**



- 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. APPLICATION EXAMPLES..... 5**
- 1.0. DATA REPORTING ..... 5**
  - 1.0.0. UNENCRYPTED AND UNSIGNED DATA REPORTING ..... 5
  - 1.0.1. ENCRYPTED AND SIGNED DATA REPORTING ..... 6
- 1.1. CLAIMING..... 7**
  - 1.1.0. STANDARD CLAIM ..... 7
- 1.2. SPIMESEEK ..... 8**
  - 1.2.0. A SPIKE SEEKS NON ENCRYPTED INFORMATION ABOUT ITSELF..... 8
  - 1.2.1. AN AUTHORIZED CLAIMER SEEKS ENCRYPTED INFORMATION ABOUT A SPIKE..... 11

## 0. Introduction

This document illustrates some OpenSpime protocol v0.9 and its core extensions application examples.

### 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 application examples of 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>), the OpenSpime architecture, and the OpenSpime protocol v0.9 with its core extensions.

### 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. Application Examples

The examples here below clarify the usage of the OpenSpime XMPP protocol extension by providing real life examples.

### 1.0. Data Reporting

#### 1.0.0. Unencrypted and Unsigned Data Reporting

Data is transmitted from a spime to a ScopeNode using a <iq/> stanza.

```
<iq from='4A6B-5638-0312111D@spime.openspime.com/spime'  
  to='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  xml:lang='en'  
  type='set'  
  id='200804021658230'>  
  <openspime xmlns='openspime:protocol:core' version='0.9'>  
    <transport>  
      <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>  
      </data>  
    </transport>  
  </openspime>  
</iq>
```

XMPP core protocol or XEP-0060 (PubSub XMPP extension) errors or custom errors MAY be reported here to the originating spime. Otherwise, if the data is successfully transmitted, the ScopeNode provides all checking functionalities, and if everything is successful, it replies with an empty <iq/> stanza directly to the originating spime.

```
<iq from='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  to='4A6B-5638-0312111D@spime.openspime.com/spime'  
  xml:lang='en'  
  type='result'  
  id='200804021658230'>  
</iq>
```

In case that errors are encountered (such as, for instance, inconsistent data with the scope), the ScopeNode reports these errors directly to the originating spime with an <iq/> stanza.

```
<iq from='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  to='4A6B-5638-0312111D@spime.openspime.com/spime'  
  xml:lang='en'  
  type='error'  
  id='200804021658230'>  
  <error type='modify'>  
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' />  
    <inconsistent-data-with-scope xmlns='openspime:protocol:extension:data:error' />  
  </error>  
</iq>
```

### 1.0.1. Encrypted and Signed Data Reporting

Data is transmitted from a spime to a ScopeNode using a `<iq/>` stanza.

```
<iq from='4A6B-5638-0312111D@spime.openspime.com/spime'  
  to='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  xml:lang='en'  
  type='set'  
  id='200804021658230'>  
  <openspime xmlns='openspime:protocol:core' version='0.9'>  
    <originator cert='services@spime.openspime.com/cert'>  
      <sign> ** Base64 encoded data ** </sign>  
    </originator>  
    <transport content-type='x-openspime/aes-base64'  
      transport-key='** Base64 encoded data **}'>  
      ** Base64 encoded data **  
    </transport>  
  </openspime>  
</iq>
```

XMPP core protocol errors MAY be reported here to the originating spime.

The recipient ScopeNode checks that the SpimeGate certification services specified in as 'cert' attribute of the `<sign/>` element is included in the list published on openspime.org, and if so, it requests the public RSA key of the originating spime with an `<iq/>` stanza, using the XMPP extension Public Key Publishing (XEP-0189).

```
<iq from='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  to='services@spime.openspime.com/cert'  
  type='get'  
  xml:lang='en'  
  id='200804021658232'>  
  <pubkeys xmlns='urn:xmpp:tmp:pubkey'  
    jid='4A6B-5638-0312111D@spime.openspime.com/spime' />  
</iq>
```

XMPP protocol errors MAY be reported here to the requesting ScopeNode, as specified in §9.3.3 of the RFC 3920. Otherwise, if everything is successful, the SpimeGate certification services replies with an `<iq/>` stanza.

```
<iq from='services@spime.openspime.com/cert'  
  to='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  type='result'  
  xml:lang='en'  
  id='200804021658232'>  
  <pubkeys xmlns='urn:xmpp:tmp:pubkey'  
    jid='4A6B-5638-0312111D@spime.openspime.com/spime'>  
    <KeyInfo xmlns='http://www.w3.org/2000/09/xmldsig#'>  
      <KeyValue>  
        <RSAKeyValue>  
          <Modulus> ** Base64 encoded data ** </Modulus>  
          <Exponent> ** Base64 encoded data ** </Exponent>  
        </RSAKeyValue>  
      </KeyValue>  
    </KeyInfo>  
  </pubkeys>  
</iq>
```

The public RSA key of the originating spime being now known, the ScopeNode can now verify the signature of the received data, by checking that the content of the `<transport/>` element corresponds to the digital signature included in the `<sign/>` element.

Finally, the SpimeGate decrypts the data and provides all checking functionalities, and if everything is successful, it replies with an empty `<iq/>` stanza directly to the originating spime.

```
<iq from='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  to='4A6B-5638-0312111D@spime.openspime.com/spime'  
  xml:lang='en'  
  type='result'  
  id='200804021658230'>  
</iq>
```

In case that errors are encountered (such as, for instance, inconsistent data with the scope), the ScopeNode reports these errors directly to the originating spime with an `<iq/>` stanza.

```
<iq from='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  to='4A6B-5638-0312111D@spime.openspime.com/spime'  
  xml:lang='en'  
  type='error'  
  id='200804021658230'>  
  <error type='modify'>  
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' />  
    <inconsistent-data-with-scope xmlns='openspime:protocol:extension:data:error' />  
  </error>  
</iq>
```

## 1.1. Claiming

### 1.1.0. Standard Claim

A spime owner claims a spime, using a `<iq/>` stanza.

```
<iq from='owner@mydomain.com/web'  
  to='4A6B-5638-0312111D@spime.openspime.com/spime'  
  xml:lang='en'  
  type='set'  
  id='200804021658230'>  
  <openspime xmlns='openspime:protocol:core' version='0.9'>  
    <transport>  
      <claim xmlns='openspime:protocol:extension:claim' version='0.9'>  
        <request claims='4A6B-5638-0312111D@spime.openspime.com/spime' />  
      </claim>  
    </transport>  
  </openspime>  
</iq>
```

XMPP core protocol or other application-specific errors MAY be reported here to the requestor service, as specified in §9.3.3 and §9.3.4 of the RFC 3920. Otherwise, if the claim is successful, the spime replies with an `<iq/>` stanza.

```
<iq from='4A6B-5638-0312111D@spime.openspime.com/spime'  
  to='owner@mydomain.com'  
  xml:lang='en'  
  type='result'  
  id='200804021658230'>  
  <openspime xmlns='openspime:protocol:core' version='0.9'>  
    <transport>  
      <claim>  
        <response authorizes='owner@mydomain.com'>  
          <claimkey> ** Base64 encoded data ** </claimkey>  
        </response>  
      </claim>  
    </transport>  
  </openspime>  
</iq>
```

## 1.2. SpimeSeek

### 1.2.0. A Spime seeks non encrypted information about itself

A spime publishes a SpimeSeek request to a *spimeseek* PubSub node with a <iq/> stanza.

```
<iq from='4A6B-5638-0312111D@spime.openspime.com/spime'  
  to='pubsub.openspime.com'  
  xml:lang='en'  
  type='set'  
  id='200804021658230'>  
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>  
    <publish node='spimeseek'>  
      <item id='98acbd983ab55f1298bfca'>  
        <openspime xmlns='openspime:protocol:core' version='0.9'>  
          <originator cert='services@spime.openspime.com/cert'  
            osid='4A6B-5638-0312111D@spime.openspime.com/spime'>  
            <sign> ** Base64 encoded data ** </sign>  
          </originator>  
          <transport>  
            <spimeseek xmlns='http://openspime.org/protocol/spimeseek'  
              version='0.9'>  
              <request id='7abc78af21'>  
                <spimeid>  
                  4A6B-5638-0312111D@spime.openspime.com/spime  
                </spimeid>  
                <expdate>2008-04-01T12:23:54+01:00</expdate>  
                <encrequired>>false</encrequired>  
              </request>  
            </spimeseek>  
          </transport>  
        </openspime>  
      </item>  
    </publish>  
  </pubsub>  
</iq>
```

XMPP core protocol, XEP-0060 (PubSub XMPP extension) or custom errors MAY be reported here to the originating spime. Otherwise, if the data is successfully transmitted, the PubSub service replies with an empty <iq/> stanza directly to the originating spime.

```
<iq from='pubsub.openspime.com'  
  to='4A6B-5638-0312111D@spime.openspime.com/spime'  
  xml:lang='en'  
  type='result'  
  id='200804021658230'>  
</iq>
```

The pubsub service, which SHOULD be configured to include payloads, distributes the SpimeSeek request to its subscribers with a <message/> stanza.

```
<message from='pubsub.openspime.com'  
  to='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  xml:lang='en'  
  id='200804021658232'>  
  <event xmlns='http://jabber.org/protocol/pubsub#event'  
    <items node='spimeseek'>  
      <item id='98acbd983ab55f1298bfca'>  
        <openspime xmlns='openspime:protocol:core' version='0.9'>  
          <originator cert='services@spime.openspime.com/cert'  
            osid='4A6B-5638-0312111D@spime.openspime.com/spime'>  
            <sign> ** Base64 encoded data ** </sign>  
          </originator>  
          <transport>  
            <spimeseek xmlns='http://openspime.org/protocol/spimeseek'  
              version='0.9'>  
              <request id='7abc78af21'>  
                <spimeid>  
                  4A6B-5638-0312111D@spime.openspime.com/spime  
                </spimeid>  
                <expdate>2008-04-01T12:23:54+01:00</expdate>  
                <encrequired>>false</encrequired>  
              </request>  
            </spimeseek>  
          </transport>  
        </openspime>  
      </item>  
    </items>  
  </event>  
</message>
```

Another SpimeSeek service MAY be subscriber of this list, and it therefore MAY publish it to another PubSub service, propagating the request. However, the subscribers of a *spimeseek* PubSub SHOULD mainly be ScopeNodes, therefore let's continue this example with a ScopeNode which receives this message.

The recipient ScopeNode verifies that:

- the request has **not already been treated**, based on the primary SpimeSeek identifier composed of the 'osid' attribute of the <originator/> element and the 'id' attribute of the <request/> stanza;
- the **expiration date** of the request is a date in the future.

If one of the two above conditions is not met, the ScopeNode MUST ignore the request.

Otherwise, the recipient ScopeNode checks that the SpimeGate certification services specified in the 'cert' attribute of the <originator/> element is included in the list published on openspime.org, and if so, it requests the public RSA key of the originator spime with an <iq/> stanza, using the XMPP extension Public Key Publishing (XEP-0189).

```
<iq from='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  to='services@spime.openspime.com/cert'  
  type='get'  
  xml:lang='en'  
  id='200804021658232'>  
  <pubkeys xmlns='urn:xmpp:tmp:pubkey'  
    jid='4A6B-5638-0312111D@spime.openspime.com/spime' />  
</iq>
```

XMPP core protocol or XEP-0189 (Public Key Publishing XMPP extension) errors MAY be reported here to the originating spime. Otherwise, if everything is successful, the SpimeGate certification services replies with a <iq/> stanza.

```
<iq from='services@spime.openspime.com/cert'  
  to='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  type='result'  
  xml:lang='en'  
  id='200804021658232'>  
  <pubkeys xmlns='urn:xmpp:tmp:pubkey'  
    jid='4A6B-5638-0312111D@spime.openspime.com/spime'>  
    <KeyInfo xmlns='http://www.w3.org/2000/09/xmldsig#'>  
      <KeyValue>  
        <RSAKeyValue>  
          <Modulus> ** Base64 encoded data ** </Modulus>  
          <Exponent> ** Base64 encoded data ** </Exponent>  
        </RSAKeyValue>  
      </KeyValue>  
    </KeyInfo>  
  </pubkeys>  
</iq>
```

The public RSA key of the originating spime being now known, the ScopeNode can now verify the signature of the received data.

If everything is successful, the ScopeNode seeks for data which may have been transmitted to it by the spime. If data is found, a <message/> stanza is sent to the originator of the request, in this case the **sought spime** itself.

```
<message from='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  to='4A6B-5638-0312111D@spime.openspime.com/spime'  
  xml:lang='en'  
  id='200804021658234'>  
  <ospime xmlns='ospime:protocol:core' version='0.9'>  
    <transport>  
      <spimeseek xmlns='http://ospime.org/protocol/spimeseek' version='0.9'>  
        <response id='7abc78af21'>  
          <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>  
          </entry>  
        </response>  
      </spimeseek>  
    </transport>  
  </ospime>  
</message>
```

### 1.2.1. An authorized claimer seeks encrypted information about a spime

An authorized claimer publishes a SpimeSeek request to a *spimeseek* PubSub node with a `<iq/>` stanza.

```
<iq from='spimeowner@mydomain.com/web'>
  to='pubsub.openspime.com'
  xml:lang='en'
  type='set'
  id='200804021658230'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='spimeseek'>
      <item id='98acbd983ab55f1298bfca'>
        <openspime xmlns='openspime:protocol:core' version='0.9'>
          <originator cert='cert.spimeowners.com'
            osid='spimeowner@mydomain.com/web'>
            <sign> ** Base64 encoded data ** </sign>
            <claimkey cert='services@spime.openspime.com/cert'
              claims='4A6B-5638-0312111D@spime.openspime.com/spime'>
              ** Base64 encoded data **
            </claimkey>
          </originator>
        </transport>
        <spimeseek xmlns='http://openspime.org/protocol/spimeseek'
          version='0.9'>
          <request id='7abc78af21'>
            <spimeid>
              4A6B-5638-0312111D@spime.openspime.com/spime
            </spimeid>
            <expdate>2008-04-01T12:23:54+01:00</expdate>
            <encrequired>>true</encrequired>
          </request>
        </spimeseek>
      </transport>
    </openspime>
  </item>
</publish>
</pubsub>
</iq>
```

XMPP core protocol, XEP-0060 (PubSub XMPP extension) or custom errors MAY be reported here to the originating spime. Otherwise, if the data is successfully transmitted, the PubSub service replies with an empty `<iq/>` stanza directly to the originating spime.

```
<iq from='pubsub.openspime.com'
  to='spimeowner@mydomain.com/web'
  xml:lang='en'
  type='result'
  id='200804021658230'>
</iq>
```

The PubSub service, which SHOULD be configured to include payloads, distributes the SpimeSeek request to its subscribers with a `<message/>` stanza.

```
<message from='pubsub.openspime.com'  
  to='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  xml:lang='en'  
  id='200804021658232'>  
  <event xmlns='http://jabber.org/protocol/pubsub#event'  
    <items node='spimeseek'>  
      <item id='98acbd983ab55f1298bfca'>  
        <openspime xmlns='openspime:protocol:core' version='0.9'>  
          <originator cert='cert.spimeowners.com'  
            osid='spimeowner@mydomain.com/web'  
            <sign> ** Base64 encoded data ** </sign>  
            <claimkey cert='services@spime.openspime.com/cert'  
              claims='4A6B-5638-0312111D@spime.openspime.com/spime'  
                ** Base64 encoded data **  
            </claimkey>  
          </originator>  
          <transport>  
            <spimeseek xmlns='http://openspime.org/protocol/spimeseek'  
              version='0.9'>  
              <request id='7abc78af21'>  
                <spimeid>  
                  4A6B-5638-0312111D@spime.openspime.com/spime  
                </spimeid>  
                <expdate>2008-04-01T12:23:54+01:00</expdate>  
                <encrequired>>true</encrequired>  
              </request>  
            </spimeseek>  
          </transport>  
        </openspime>  
      </item>  
    </items>  
  </event>  
</message>
```

Another SpimeSeek service MAY be subscriber of this list, and it therefore MAY publish it to another PubSub service, propagating the request. However, the subscribers of a *spimeseek* PubSub SHOULD mainly be ScopeNodes, therefore let's continue this example with a ScopeNode which receives this message.

The recipient ScopeNode verifies that:

- the request has **not already been treated**, based on the primary SpimeSeek identifier composed of the 'osid' attribute of the <originator/> element and the 'id' attribute of the <request/> stanza;
- the **expiration date** is a date in the future.

If one of the two above conditions is not met, the ScopeNode MUST ignore the request.

Otherwise, since the <claimkey/> element of the <originator/> parent node has been specified, the ScopeNode MUST verify it before proceeding.

Therefore, the recipient ScopeNode checks that the SpimeGate certification services specified in the 'cert' attribute of the <claimkey/> element is included in the list published on openspime.org, and if so, it requests the public RSA key of the claimed spime with an <iq/>stanza, using the XMPP extension Public Key Publishing (XEP-0189).

```
<iq from='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  to='services@spime.openspime.com/cert'  
  type='get'  
  xml:lang='en'  
  id='200804021658232'>  
  <pubkeys xmlns='urn:xmpp:tmp:pubkey'  
    jid='4A6B-5638-0312111D@spime.openspime.com/spime' />  
</iq>
```

XMPP protocol errors MAY be reported here to the originating spime, as specified in §9.3.3 of the RFC 3920 and XEP-0189. Otherwise, if everything is successful, the SpimeGate certification services replies with a <iq/> stanza.

```
<iq from='services@spime.openspime.com/cert'  
  to='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  type='result'  
  xml:lang='en'  
  id='200804021658232'>  
  <pubkeys xmlns='urn:xmpp:tmp:pubkey'  
    jid='4A6B-5638-0312111D@spime.openspime.com/spime'>  
    <KeyInfo xmlns='http://www.w3.org/2000/09/xmldsig#'>  
      <KeyValue>  
        <RSAKeyValue>  
          <Modulus> ** Base64 encoded data ** </Modulus>  
          <Exponent> ** Base64 encoded data ** </Exponent>  
        </RSAKeyValue>  
      </KeyValue>  
    </KeyInfo>  
  </pubkeys>  
</iq>
```

The public RSA key of the claimed spime being now known, the ScopeNode can now decode and decrypt the Claim Key included as value of the <claimkey/> element, and verify the **validity** of the **Claim Key**, which:

- MUST contain spimeowner@mydomain.com/web as value of the key's <osid/> element;
- MUST specify an expiration date (in the key's <expdate/> element) that is a date in the future.

If both conditions are met, the ScopeNode can now **verify the signature** of the request. To do so, the recipient ScopeNode checks that the SpimeGate certification services specified in as 'cert' attribute of the <originator/> element is included in the list published on openspime.org, and if so, it requests the public RSA key of the originator service with an <iq/> stanza, using the XMPP extension Public Key Publishing (XEP-0189).

```
<iq from='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  to='cert.spimeowners.com'  
  type='get'  
  xml:lang='en'  
  id='200804021658232'>  
  <pubkeys xmlns='urn:xmpp:tmp:pubkey'  
    jid='spimeowner@mydomain.com/web' />  
</iq>
```

XMPP protocol errors MAY be reported here to the originating spime, as specified in §9.3.3 of the RFC 3920 and XEP-0189. Otherwise, if everything is successful, the SpimeGate certification services replies with an <iq/> stanza.

```
<iq from='cert.spimeowners.com'  
  to='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  type='result'  
  xml:lang='en'  
  id='200804021658232'>  
  <pubkeys xmlns='urn:xmpp:tmp:pubkey'  
    jid='spimeowner@mydomain.com/web'>  
    <KeyInfo xmlns='http://www.w3.org/2000/09/xmldsig#'>  
      <KeyValue>  
        <RSAKeyValue>  
          <Modulus> ** Base64 encoded data ** </Modulus>  
          <Exponent> ** Base64 encoded data ** </Exponent>  
        </RSAKeyValue>  
      </KeyValue>  
    </KeyInfo>  
  </pubkeys>  
</iq>
```

The public RSA key of the originator service being now known, the ScopeNode can now verify the signature of the request (i.e. the value of the <sign/> element).

If everything is successful, the ScopeNode seeks for data which may have been transmitted to it by the spime. If data is found, a <message/> stanza is sent to the SpimeSeek originator entity, in this case the originator **authorized claimer**, encoded and encrypted with AES (since encryption has been requested), with its key being encrypted with the originator public RSA key.

```
<message from='775B-F2C2-2B3765F4@scopenode.openspime.com/co2'  
  to='spimeowner@mydomain.com/web'  
  xml:lang='en'  
  id='200804021658234'>  
  <openspime xmlns='openspime:protocol:core' version='0.9'>  
    <transport content-type='x-openspime/aes-base64'  
      transport-key='** Base64 encoded data **}'>  
      ** Base64 encoded data **  
    </transport>  
  </openspime>  
</message>
```