OIC REMOCE ACCESS SPECIFICATION V1.0.0

Open Interconnect Consortium (OIC)
admin@openinterconnect.org

Legal Disclaimer

NOTHING CONTAINED IN THIS DOCUMENT SHALL BE DEEMED AS GRANTING YOU ANY KIND OF LICENSE IN ITS CONTENT, EITHER EXPRESSLY OR IMPLIEDLY, OR TO ANY INTELLECTUAL PROPERTY OWNED OR CONTROLLED BY ANY OF THE AUTHORS OR DEVELOPERS OF THIS DOCUMENT. THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "AS IS" BASIS, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE AUTHORS AND DEVELOPERS OF THIS SPECIFICATION HEREBY DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. OPEN INTERCONNECT CONSORTIUM, INC. FURTHER DISCLAIMS ANY AND ALL WARRANTIES OF NON-INFRINGEMENT, ACCURACY OR LACK OF VIRUSES.

The OIC logo is a trademark of Open Interconnect Consortium, Inc. in the United States or other countries. *Other names and brands may be claimed as the property of others.

CONTENTS

FIGURES:

TABLES:

1 Scope

1.1 Rationale for limitations/phasing

Many of the specific details for a final commercially-viable implementation of a general Remote-Access solution are dependent on concepts presently being defined in other parts of the OIC Standards Working Group:

- Device on-boarding/ownership-transfer/local provisioning – Both the *state* an OIC device will be in once it has been successfully provisioned to an owner in the local domain (such as the user's 'home'), as well as the *process* and *tools* (the On-Boarding Tool, or OBT) used to get the device into that state are being defined in the Security TG and Core Framework. The Remote Access approach will be an extension of the above, and will rely on the approved Security and Core Framework standards.
  HOWEVER: While the specific Remote Access final specification must depend on the specific approved Specifications above, the core concepts for Remote Access functionality are described and can be implemented to verify the assumptions and vet fundamental implementation details/assumptions. Near-term modification of the Remote-Access Specification following this initial version will include the specifics as the other upstream-dependencies are formalized/approved. Implementation of basic Remote-Access functionality (XMPP client implementation, XMPP Server deployment, etc.) can proceed, and the security provisions will be added later.

- Inter-server federation requirements – The initial phase is intended to support the simplest single-vendor Remote-Access use case(s), and interoperable, multi-vendor use-cases will be specified in a later (soon) phase. This initial phase is intended to vet the basic design and implementation parameters proposed, and the multi-vendor, multi-server requirements will build on the foundation vetted here.

- ICE/STUN/TURN implementation – Initial Remote-access requirements are being driven by the need to facilitate secure remote (outside of the local domain) communication of the basic OIC CoAP/JSON/CBOR CRUDN messages. Adding media streaming, bulk-file, and other similar requirements that potentially prefer peer-to-peer communication paths will build on the infrastructure provided here via XMPP (via Jingle),

2 Normative references

Normative references follow RFC 2119 conventions. OIC Resource definition tables with a 'Mandatory' column identify OIC Resource properties that MUST be implemented by all OIC devices that instantiates the resource if Mandatory is YES. All OIC devices MAY implement oic resource properties unless otherwise specified in the table.

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IETF RFC 6120, (XMPP CORE) *Extensible Messaging and Presence Protocol (XMPP): Core*
http://xmpp.org/rfcs/rfc6120.html

IETF RFC 6121, (XMPP IM) *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*
http://xmpp.org/rfcs/rfc6121.html

IETF RFC 6122, (XMPP ADDR) *Extensible Messaging and Presence Protocol (XMPP): Address Format*
http://xmpp.org/rfcs/rfc6122.html

IETF RFC 3923, (XMPP E2E) *End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)*
http://xmpp.org/rfcs/rfc3923.html

IETF RFC 4854, (XMPP URN) *A Uniform Resource Name (URN) Namespace for Extensions to the Extensible Messaging and Presence Protocol (XMPP)*
http://xmpp.org/rfcs/rfc4854.html

IETF RFC 4979, (XMPP ENUM) *IANA Registration for Enumservice 'XMPP'*
http://tools.ietf.org/html/rfc4979

IETF RFC 5122, (XMPP URI) *Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)*
http://xmpp.org/rfcs/rfc5122.html

IETF RFC 7590, *Use of Transport Layer Security (TLS) in the Extensible Messaging and Presence Protocol (XMPP)*
https://tools.ietf.org/html/rfc7590

IETF RFC 4648, *The Base16. Base32, and Base64 Data Encodings*
https://tools.ietf.org/html/rfc4648

XEP-0047, *In-Band Bytestreams*
http://xmpp.org/extensions/xep-0047.html

XEP-0199, *XMPP Ping*
http://xmpp.org/extensions/xep-0199.html

OIC Security, *Open Interconnect Consortium Security Capabilities*, Version 1.0

OIC Core, *Open Interconnect Consortium Core Specification*, Version 1.0

3 Terms, definitions, symbols and abbreviations

Terms, definitions, symbols and abbreviations used in this specification are defined by the OIC Core specification. Additional terms specific to normative Remote Access mechanisms are defined in this document in context.

This section restates terminology that is defined elsewhere, in this document or in other OIC specifications as a convenience for the reader. It is considered non-normative.

3.1 Terms and definitions

The definitions from the Core Specification apply. In addition, the following terminologies are used in this specification:

Remote access
Interaction between an OIC Client and OIC Server where each OIC Devices is on a different network

Remote Access Endpoint (RAE) Server
An OIC Server which supports an XMPP client and it can publish its (oic) resource(s) to the XMPP server, thus becoming remotely addressable and accessible
It also supports ICE/STUN/TURN if the application on the OIC server requires it

RAE Client
An OIC Client which supports an XMPP client functionality.

XC-Proxy
Acts as a (OIC) Resource Directory for RA-Constrained OIC Devices and performs bidirectional protocol mapping between XMPP and OIC Devices.

RA-Constrained OIC Device:
An OIC Device without any XMPP client functionality.

OIC Resource
an Resource described by OIC that has CRUDN actions and represent functionality.

XMPP Resource
the extension part of the full JID that makes an full JID of an bare JID.

3.2 Symbols and abbreviations

Symbol	Description
RA	Remote access
RAE	Remote Access Endpoint
RA-Constrained Device	An OIC Device which is not capable (by itself) of supporting RA capabilities
RA-Capable Device	Any OIC Device which is capable of providing RA-services. This includes RAE and XC-Proxy Devices

Table 1 - Symbols, terminology and abbreviations

4 Document conventions and organization

4.1 Notation

In this document, features are described as required, recommended, allowed or DEPRECATED as follows:

Required (or shall or mandatory).

> These basic features shall be implemented to comply with the Remote Access Architecture. The phrases “shall not”, and “PROHIBITED” indicate behavior that is prohibited, i.e. that if performed means the implementation is not in compliance.

Recommended (or should).

> These features add functionality supported by Remote Access Architecture and should be implemented. Recommended features take advantage of the capabilities Remote Access Architecture, usually without imposing major increase of complexity. Notice that for compliance testing, if a recommended feature is implemented, it shall meet the specified requirements to be in compliance with these guidelines. Some recommended features could become requirements in the future. The phrase “should not” indicates behavior that is permitted but not recommended.

Allowed (or allowed).

> These features are neither required nor recommended by the Remote Access Architecture, but if the feature is implemented, it shall meet the specified requirements to be in compliance with these guidelines. These features are not likely to become requirements in the future.

DEPRECATED.

> Although these features are still described in this specification, they should not be implemented except for backward compatibility. The occurrence of a deprecated feature during operation of an implementation compliant with the current specification has no effect on the implementation's operation and does not produce any error conditions. Backward compatibility may require that a feature is implemented and functions as specified but it shall never be used by implementations compliant with this specification.

Strings that are to be taken literally are enclosed in "double quotes".

Words that are emphasized are printed in *italic*.

5 High Level Overview

5.1 Rationale (Informative)

Most IoT initiatives describe methods/protocols for devices to interact with one another. These IoT technologies are often by themselves incapable of supporting general, bidirectional Internet connectivity, either owing to limitations in connectivity and/or incompatibility between the specified protocols and those used on the Internet. Often these limitations are a result of the constraints imposed on IoT devices: Cost, power, etc., or additionally the presence of NATs (Network Address Translation devices) or other network topologies that inhibit general connectivity.

The Remote Access specification describes the use of XMPP and ICE (with STUN & TURN) to securely and scalably add Internet connectivity both to so-called constrained device networks and additionally for network topologies that obfuscate or otherwise inhibit general connectivity.

There are two operational models to accomplish Remote Access:

1. Some devices will possess adequate resources (CPU power, memory…) to be able to employ the techniques and protocols described here to successfully accomplish generalized Remote Access 'by themselves' (without the assistance of additional devices within their local network /subnet). Owing to the impact of Moore's Law, it is expected there will be an increasing number of devices of this type over time.

2. For so-called Remote-Access-constrained devices (devices not capable of directly supporting/hosting general Internet connectivity and the protocols described here): The infrastructure and mechanisms by which adequately-capable devices may provide services to (to proxy on behalf of) networks of these constrained devices will be described in a next version of this specification.

5.2 Philosophy/Approach (Informative)

Remote access is accomplished by leveraging the XMPP and ICE(/STUN/TURN) standards. The Remote Access feature is optional to implement and can be included when the OIC Device has the resources (CPU, Memory, etc.) to implement this feature. Many external references are available for XMPP and ICE standards/protocols for those who are unfamiliar with these standards/protocols.

In general:

- Each Remote Access capable device must have first been 'on-boarded' and provisioned such that it is uniquely and securely associated with a single owner.

- Each OIC Remote-Access capable device will connect through a XMPP account on a XMPP server, and this XMPP server must be accessible via the public internet.

- All devices on the same XMPP account can talk to each other. The devices on the same account are automatically placed in the account Roster. The Roster determines to whom the account can talk too. One of the implicit mechanism of the Roster is that all connections made by the same user account will establish an instance of that connection in the Roster. The identification mechanism of the different connections is established by the XMPP resource part of the full JID.

- By default in the XMPP world, XMPP stanza are exchanged between XMPP clients (end points). In OIC specifications, the messaging between the OIC Devices is achieved by the Restful paradigm by defining CRUDN payloads. This means that the CRUDN message is placed in the payload of an XMPP stanza, transmitted via XMPP, and decoded on the receiving end.

5.3 Architecture

The Remote Access (RA) architecture of OIC is based on the support of the OIC defined CRUDN message protocol [OIC CORE], XMPP and ICE/STUN/TURN (when the application on the OIC Device requires it). Figure 1 shows the high level RA Architecture of OIC for Remote Access with one XMPP Server.

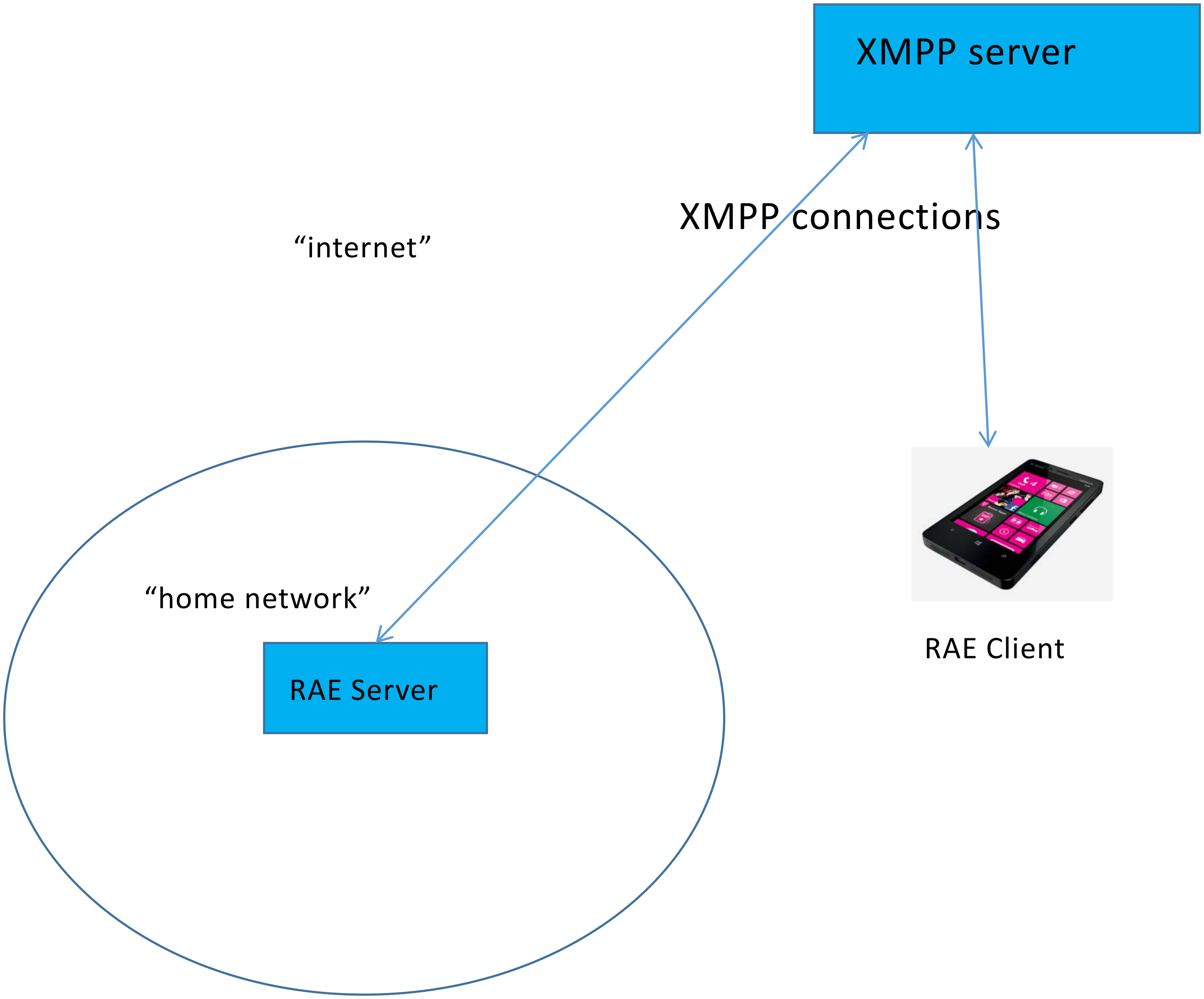


Figure 1 Remote Access High-Level Architecture

The RAE Server is an OIC Server with XMPP client functionality. This configuration is depicted in Figure 2. The RAE Server is configured with an address and account of the (known) XMPP server in the cloud. The RAE Client also contains an XMPP Client and connects to the same XMPP server using the same account information.

The RAE shall contact the XMPP server and establish a secure XMPP connection after power up.

When the OIC devices are connected to the same XMPP server and are using the same account information XMPP allows communication between those devices. The connection can be used to send XMPP stanzas from an RAE to another RAE.

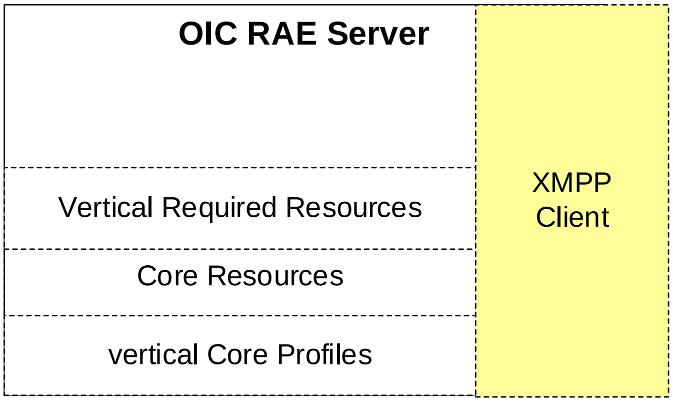
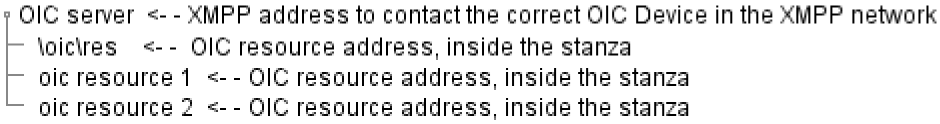


Figure 2 RAE Server depicted as an OIC Server with the XMPP Client.

The full JID of the connection address of the RAE will be used to as the XMPP address for sending the XMPP stanzas (the “to” address in the XMPP messaging scheme). The OIC CRUDN messaging is directed from and OIC Client to an OIC Resource in an OIC server. To have equivalent mechanism available over XMPP, the stanza will contain the CRUDN message including the addressing of the OIC Resource implemented in the OIC server.



```
OIC server  <- - XMPP address to contact the correct OIC Device in the XMPP network
├ \oic\res     <- -  OIC resource address, inside the stanza
├ oic resource 1  <- - OIC resource address, inside the stanza
└ oic resource 2  <- - OIC resource address, inside the stanza
```

Figure 3 XMPP and OIC Resource addressing levels.

Hence this means that 2 levels of addressing are needed:

- Addressing the XMPP stanza towards the OIC Device
    - This is achieved by XMPP addressing, using the full JID
- Addressing of the OIC Resource in the OIC Device
    - This is achieved in the XMPP stanza payload mimicking CRUDN actions including the addressing

How to use the different XMPP and OIC addresses is depicted in

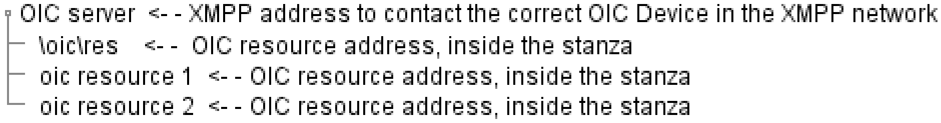

OIC server <- - XMPP address to contact the correct OIC Device in the XMPP network
\oic\res <- - OIC resource address, inside the stanza
oic resource 1 <- - OIC resource address, inside the stanza
oic resource 2 <- - OIC resource address, inside the stanza

Figure 3.

6 Remote Access Components and Accounts

6.1 XMPP Server

An OIC XMPP server is deployed on the public internet and is used for following purposes:

a) Announcing the presence of OIC devices from outside the proximal network.

b) Exchanging low-bandwidth OIC messages (data packets) for accessing/managing remote communication between OIC Clients and OIC Servers connected through XMPP

The OIC XMPP Server operational model does not mandate the specific location (domain or URL) for an XMPP Server infrastructure, and it is expected that a manufacturer will either operate their own XMPP servers or will contract with a service-provider for XMPP Server services for the RA-capable devices they sell. Account creation on XMPP Servers

Before an XMPP Server can be used, at minimum the end-user has to have an account on the XMPP server. This procedure is expected to be done out-of-band. The user's bare-JID (XMPP user-account/server) and credentials will be communicated to the user separately (out-of-band).

6.2 XMPP login

The XC Proxy will have an OIC resource identifier that will allow it to be identified as an RAE. It will log into the relevant XMPP Server(s) on behalf of the RA-Constrained Devices which have published themselves to the bridge. Included in the account credentials, etc. for a device will be (some implicit):

- Its bare-JID (XMPP username/account and server)
- The account credentials
- The relevant XMPP server address and port

6.2.1 Remote Access Call Flow for RAE

An OIC Client shall have an out of bound mechanism (a.k.a. a user interface) to enter the account information and XMPP connection information to establish a connection to the XMPP server.

The OIC server (without the same mechanisms of an OIC Client) shall have a Remote Access OIC resource to set the account and XMPP server information. An OIC Client (with the already supplied account and XMPP server information will provide the information to the OIC Server. When an OIC server is not properly initialized, an OIC Client will have to provide the correct information to the OIC Server. When these are set, the OIC Server will try to (re-)establish the connection. It will be possible to detect the result by looking at the connection status property returned via XMPP.

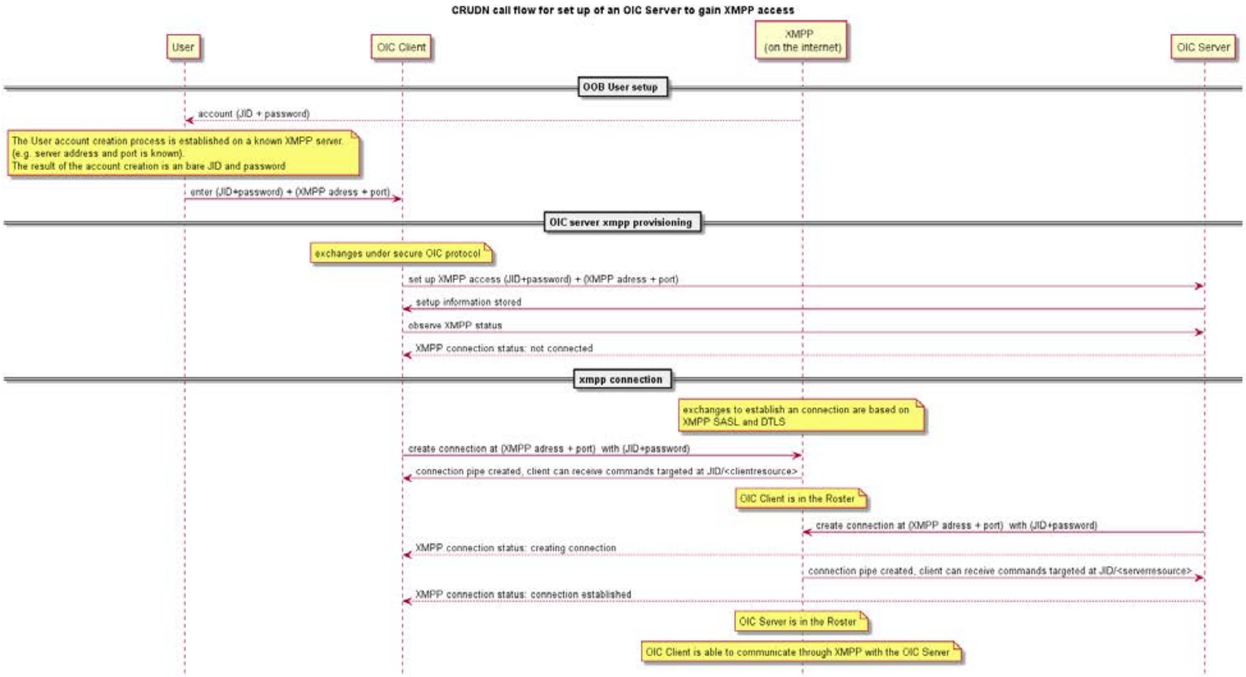

CRUDN call flow for set up of an OIC Server to gain XMPP access
User
OIC Client
XMPP
(on the internet)
OIC Server
OOB User setup
account (JID + password)
The User account creation process is established on a known XMPP server.
(e.g. server address and port is known).
The result of the account creation is an bare JID and password
enter (JID+password) + (XMPP adress + port)
OIC server xmpp provisioning
exchanges under secure OIC protocol
set up XMPP access (JID+password) + (XMPP adress + port)
setup information stored
observe XMPP status
XMPP connection status: not connected
xmpp connection
exchanges to establish an connection are based on
XMPP SASL and DTLS
create connection at (XMPP adress + port) with (JID+password)
connection pipe created, client can receive commands targeted at JID/<clientresource>
OIC Client is in the Roster
create connection at (XMPP adress + port) with (JID+password)
XMPP connection status: creating connection
connection pipe created, client can receive commands targeted at JID/<serverresource>
XMPP connection status: connection established
OIC Server is in the Roster
OIC Client is able to communicate through XMPP with the OIC Server

Figure 4 CRUDN call flow for RAE setup.

Figure 4 depicts the steps to enable the RAE so that it can contact the XMPP server. The communications to create a JID (userid@domain) on an XMPP server are out of bounds. The data to connect to a server is supplied out of bounds. This is that any XMPP server can be used to create an OIC remote access connection. The communication between the OIC Client to pass the JID and password together with the XMPP connection data is done by OIC commands. This means that the communication of all the XMPP credentials are either out of bounds or are exchanged under the established security mechanisms defined by OIC.

6.2.2 OIC defined Resources for Remote access

The OIC server that supports Remote Access shall implement 2 resources, namely:

The oic.ra.xmpp resource indicates the XMPP server address and connection status.

The oic.ra.user resource indicates the user credential on the XMPP server.

The resources shall comply with the core specification and shall implement all mandatory properties. Note that only the additional (remote access relevant) properties are listed in this document.

6.2.2.1 OIC define Resource for XMPP connection (oic.ra.xmpp)

The resource to set the xmpp connection data is identified with rt = “oic.ra.xmpp”.

The resource properties for this resource are listed in **Table 2**.

Table 2. oic.ra.xmpp resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
XMPP Server Address	address	s			R, W	Yes	XMPP server address
XMPP Server Port	port	number			R, W	Yes	XMPP server port
Status	status	enum			R	Yes	Status of the Connection to the XMPP server
Error reason	error	string					Vendor defined appropriate error message when status is “Error”

Status will have the enum values: “Connected”, “Error”, “NotInitialized”.

6.2.2.2 OIC defined Resource for XMPP user data (oic.ra.user)

The resource to set the XMPP connection data is identified with rt = “oic.ra.user”.

The resource properties for this resource are listed in **Table 3**

It is highly recommended that this resource will be access restricted for reading during normal operation (e.g. when being used by a normal end user), hence only the user that is allowed to do onboarding should be allowed to read/write this resource.

Table 3. oic.ra.user resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
UserID	jid	string			R, W	Yes	Bare JID
credential	port	string			R, W	Yes	Base64 encoded credential

7 Discovery & Presence

7.1 Registration

Before an OIC Device can connect to its XMPP server it needs to be provisioned with a username (JID – Jabber ID) and a passphrase or other security model (such as SAML) – the specific requirements for user- and device-account credentials/security can be found in [OIC Security].

The XMPP account is created based on the identity of the user. Each device will be logged in under a (XMPP) resource for the specific end user; e.g.:

<user>@<domain.com>/<resource>, where

*“**user**” (a.k.a.: ‘username’, ‘local’ or ‘node’ in XMPP parlance) is the Jabber ID (or JID) unique to that user for the specific IdP (example: john@facebook.com)*
***domain.com** is the “domain” (a.k.a: ‘server’ or ‘host’ in XMPP parlance) for the XMPP “user”, above*
*‘**resource**’ is the device name/id the user is logging into*

Note: In XMPP parlance, '`user@domain.com`' is referred to as a "bare-JID" while 'user@domain.com/resource' is referred to as a "full-JID".

Note: As defined by the XMPP RFCs, the username, domain and resource-parts of a JID can contain nearly any Unicode character, and the case-sensitivity model (actually referred to as 'case-folding' in XMPP, whose rules are defined by a technology called stringprep, specified in RFC 3454) which applies to the Resource portion of a full-JID are described in RFCs 5122, 6122).The bare-JID is case-IN-sensitive.

7.1.1 Connection identification

The connection of an OIC Device to the XMPP server is identified by the (XMPP) resource. Hence OIC mandates that an XMPP client supplies the full-JID when establishing the connection. The full JID can be used to distinguish:

- OIC devices from other connections
- Whether an OIC Device is an OIC Client or OIC Server
- Which device type (rt) the device is.

The following scheme full-JID scheme shall be supplied by an OIC Client:

Client RAE: **`{user}@{domain.com}/OIC/1.0/Client/{UUID}`**

The UUID shall be maintained over the lifecycle of the OIC Client. That is, when an OIC Client re-establish a connection after a reboot it shall use the same UUID.

The following scheme full-JID scheme shall be supplied by an OIC Server:

RAE Server: **`{user}@{domain.com}/OIC/1.0/{OIC-device type}/{UUID}`**

The UUID shall be maintained over the lifecycle of the OIC Server and is the same UUID as defined in property "di" of resource /oic/d. The OIC-device-type shall be the same value as the property "rt" in /oic/d.

When an RAE Device implements an OIC Client and an OIC Server then the full-JID of the RAE Server shall be used. Note that an XMPP Client allows to send and receive commands, hence the established XMPP connection can be used by both the OIC Client and the OIC Server.

These full-JID formats (above) allow for:

- Discovery of the device-type (resource-type) directly from the full-JID on the Roster supplied by the XMPP server — without having to query the device(s)
- Elimination of full-JID-collision via use of the UUIDs
- A ***non***-multi-cast-type mechanism to do device discovery.
- Upgradability of the protocol mechanism by the changing version number (1.0).

Example of an OIC Server full-JID, denoting a light device:

```
me@mydomain.com/OIC/1.0/oic.d.light/FFFFB960-BABE-46F7-BEC0-9E6234671ADC0
```

Example of an OIC Client full-JID:

```
me@mydomain.com/OIC/1.0/Client/FXFFB960-FFFF-46F7-BABE-9E6234671ADC1
```

7.2 Connection Authentication

The RAE will establish a connection to the XMPP server using the bare JID. The connection is regarded established when the initial login occurs and it completes the preconditions described in [RFC-6120] (also known as XMPP-CORE). The stream establishment shall include security negotiation (TLS, SASL) as described in section 5 and 6 of [RFC-6120].

SASL authentication in XMPP allows for multiple mechanism to be used. OIC RAE shall use as minimum mechanism “SCRAM-SHA-1”.

In the binding step (as described in section 7.4 (Advertising Support)) the OIC RAE shall offer the XMPP resource with the format as described in 7.1.1. When the XMPP server changes the offered full JID in the binding process the RAE shall disconnect the stream. Upon a successful bind the RAE is reachable over XMPP by its own globally unique full JID.

7.3 Roster and Presence

When the client has connected to the XMPP server, it shall retrieve the Roster and signal its presence status. The retrieval of the Roster on login is described in section 2.2 of [RFC-6121]. The Roster is the list JIDs of other XMPP users (referred to as Roster ‘members’) it can communicate with and get presence indications from other entries in the Roster.
The presence is announced as described in section 4.2 of [RFC-6121].
The presence mapping for OIC devices is as described in **Table 4**.

Table 4. XMPP presence (status type) mapping

XMPP status type	OIC interpretation
available (no @type attribute)	OIC is reachable and working
unavailable	OIC device is not reachable

The XMPP messages can have priority. When priorities are used, the priority mappings to XMPP for OIC devices are:

OIC Servers with no additional XMPP features:	priority range of [-100 to -33].
OIC Servers with additional XMPP features:	priority range of [1 to 66].
OIC Clients with no additional XMPP features:	priority range of [-66 to -1].
OIC Clients with additional XMPP feature:	priority range of [33 to 100].

The Roster is not the decision point when it comes to authorization. It merely gives the connecting user/device the ability to:

- Discover other the online status of users (read: OIC Devices) in their Roster (a.k.a: ‘presence’).
- Send and receive data to JIDs in their Roster.

This can serve as the first enforcement point of access control to avoid unnecessary or malicious traffic to the smart device or gateway in the home the represents the in-home devices. After a client has connected and discovered all of the online entities it can communicate with it can now start communicating with the end device.

7.3.1 CRUDN messaging over XMPP

RAE connected over the XMPP server can directly exchange data between each other by using the In-band Bytestreams [XEP-0047]. In-band Bytestreams establishes a session to exchange binary data. This session shall be set up in a bi-directional way. The used stanza type for the connection shall be “message”. The block size of the stanza size shall be maximum 65535 bytes. To set up the byte stream the full JID of the RAE shall be used.

Each individual stanza over the connection will correspond with either a CRUDN request or respond message.

The payload of the IQ stanza is comprised of:

- URL to the OIC Resource
    - Method as attribute
- Headers (as being used to convey extra information for negotiation purposes)
- Body (optional)
    - Payload of the body in JSON

The payload must be `base64-encoding` before added as a payload.
Methods are defined as the CRUDN messages as described in the Core specification.
Note that the Notification mechanism Observe is an extended Retrieve message based on CoAP Get. The header names and payloads are defined as HTTP headers (they are ASCII instead of binary).

The payload of a binary message is defined as (before base64-encoding):

```
<rest xmlns="rest.oic.org">
 <url method="methodname">fully qualified url</url>
 <headers>
      <!—optional headers if needed →
      <header name="header name">header value</header>
      <!—additional headers →
 </headers>
  <!—optional body if needed →
  <body>
     <json xmlns="urn:xmpp:json:0">
        json payload as described in the core and/or vertical
   </json>
  </body>
</rest>
```

Method defined as HTTP (see core mappings): GET, POST, PUT, DELETE, RESPONSE
Note that the response in HTTP is formatted as a number and status. The full response line will be placed in the payload of url tag.

Example of a Get and response message (before base64-encoding):
Request:

```
<rest xmlns="rest.oic.org">
 <url method="Get">coap://mydevice/mybinaryswitch</url>
 <headers>
      <header name="Accept">application/json</header>
      <header name="Accept-Charset">UTF-8</header>
      <header name="Date">Fri, 14 Aug 2015 08:49:37 GMT</header>

 </headers>
</rest>
```

Response:

```
<rest xmlns="rest.oic.org">
 <url method="Response">200 OK</url>
 <headers>
      <header name="Content-Encoding">Application/JSON</header>
      <header name="Accept-Charset">UTF-8</header>
      <header name="Date"> Fri, 14 Aug 2015 08:49:38 GMT</header>
 </headers>
 <body>
     <json xmlns="urn:xmpp:json:0">
             {
                "rt":     "oic.r.switch.binary",
                "id":     "unique_example_id",
                "value":  false
              }
     </json>
  </body>
</rest>
```

7.4 Ungraceful Disconnect

The XMPP server may enforce client-side heartbeats to ‘quickly’ detect when a client goes offline ungracefully instead of relying solely on the TCP retransmission timeout (which is OS/platform dependent and could be large – on the order of 15 minutes). This can be accomplished with, XMPP Ping [XEP-0199]. This XEP describes how an XMPP client can send an XMPP ping periodically. The ping can be used by the XMPP server to disconnect clients that did not send a ping within a certain interval. Selecting the interval for disconnecting the client should be chosen carefully, since the interval will impose resource requirements (CPU, memory, etc.) of the XMPP Server infrastructure. The ping interval is vendor specific.

Annex A
Resource Types definitions used in Remote Access

A.1 Remote Access XMPP

A.1.1 Introduction

This resource specifies the XMPP server access.

A.1.2 Wellknown URI

/XMPPResURI

A.1.3 Resource Type

The resource type (rt) is defined as: oic.ra.xmpp.

A.1.4 RAML Definition

```
#%RAML 0.8
title: OICRemoteAccessXMPP
version: v1.0-20150819
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.s"]

/XMPPResURI:
  description: |
    This resource specifies the xmpp server access.

  is : ['interface']
  get:
    description: |
      Retrieves the xmpp access.

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "id": "http://openinterconnect.org/schemas/oic.ra.xmpp#",
                "$schema": "http://json-schema.org/draft-04/schema#",
                "title": "XMPP server connection information",
                "definitions": {
                  "oic.ra.xmpp": {
                    "type": "object",
                    "properties": {
                      "address":  {
                        "type": "string",
                        "description": "address of the XMPP server"
                      },
                      "port":   {
                        "type": "number",
                        "description": "port number of the XMPP server"
                      },
                      "status": {
                        "enum": ["Connected", "Error", "NotInitialized"],
                        "description": "ReadOnly, connection status"
                      },
                       "ErrorReason": {
                        "type": "string",
                        "description": "ReadOnly, The error reason if the status is in error"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  {"$ref":
"http://openinterconnect.org/schemas/oic.core.json#/definitions/oic.core"},
                  {"$ref": "#/definitions/oic.ra.xmpp"}
                ],
                "required": ["address","port","status","ErrorReason"]
              }

            example: |
              {
                "rt":          "oic.ra.xmpp",
                "address":            "www.cisco.oic.xmpp.com",
                "port":        8080,
                "status":      "Connected",
                "ErrorReason":  ""
              }

  post:
    description: |
      Sets the new jid and credential

    body:
      application/json :
        schema: |
          {
            "id": "http://openinterconnect.org/schemas/oic.ra.xmpp-Update#",
            "$schema": "http://json-schema.org/draft-04/schema#",
            "title": "XMPP server connection information for updating",
            "definitions": {
              "oic.ra.xmpp-Update": {
                "type": "object",
                "properties": {
                  "address":  {
                    "type": "string",
                    "description": "address of the XMPP server"
                  },
                  "port":   {
                    "type": "number",
                    "description": "port number of the XMPP server"
                  },
                  "status": {
                    "enum": ["Connected, Error, NotInitialized"],
                    "description": "ReadOnly, connection status"
                  },
                       "ErrorReason": {
                    "type": "string",
                    "description": "ReadOnly, The error reason if the status is in error"
                  }
                }
              }
            },
            "type": "object",
            "allOf": [
              {"$ref": "http://openinterconnect.org/schemas/oic.core.json#/definitions/oic.core"},
              {"$ref": "#/definitions/oic.ra.xmpp-Update"}
            ],
            "required": ["address","port"]
          }

        example: |
          {
                "rt":         "oic.ra.xmpp",
                "address":     "www.new.cisco.oic.xmpp.com",
                "port":       8081
          }

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "id": "http://openinterconnect.org/schemas/oic.ra.xmpp-Update#",
                "$schema": "http://json-schema.org/draft-04/schema#",
                "title": "XMPP server connection information for updating",
                "definitions": {
                  "oic.ra.xmpp-Update": {
                    "type": "object",
                    "properties": {
                      "address":  {
                        "type": "string",
                        "description": "address of the XMPP server"
                      },
                      "port":   {
                        "type": "number",
                        "description": "port number of the xmpp server"
                      },
                      "status": {
                        "enum": ["Connected, Error, NotInitialized"],
                        "description": "ReadOnly, connection status"
                      },
                       "ErrorReason": {
                        "type": "string",
                        "description": "ReadOnly, The error reason if the status is in error"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  {"$ref":
"http://openinterconnect.org/schemas/oic.core.json#/definitions/oic.core"},
                  {"$ref": "#/definitions/oic.ra.xmpp-Update"}
                ],
                "required": ["address","port"]
              }

            example: |
              {
                "rt":         "oic.ra.xmpp",
                "address":     "www.new.cisco.oic.xmpp.com",
                "port":       8081
              }

```

A.1.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
address	string	yes	Read Write	address of the XMPP server

port	number	yes	Read Write	port number of the XMPP server
status	enum	yes	Read Only	Connection Status
ErrorReason	string	yes	Read Only	The Error Reason if the Status is in Error

A.1.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/XMPPResURI		get	post		

A.2 Remote Access User data

A.2.1 Introduction

This resource specifies the XMPP user id and credentials.

A.2.2 Wellknown URI

/XMPPUserResURI

A.2.3 Resource Type

The resource type (rt) is defined as: oic.ra.user.

A.2.4 RAML Definition

```
#%RAML 0.8
title: OICRemoteAccessUser
version: v1.0-20150819
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.s"]


/XMPPUserResURI:
  description: |
    This resource specifies the XMPP user id and credentials.

  is : ['interface']
  get:
    description: |
      Retrieves the XMPP user data.

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "id": "http://openinterconnect.org/schemas/oic.ra.user#",
                "$schema": "http://json-schema.org/draft-04/schema#",
                "title": "XMPP server user information",
                "definitions": {
                  "oic.ra.user": {
                    "type": "object",
                    "properties": {
                      "jid":  {
                        "type": "string",
                        "description": "the bare jid"
                      },
                      "credential":   {
                        "type": "string",
                        "description": "base64 encoded string, the credential"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  {"$ref":
"http://openinterconnect.org/schemas/oic.core.json#/definitions/oic.core"},
                  {"$ref": "#/definitions/oic.ra.user"}
                ],
                "required": ["jid","credential"]
              }

            example: |
              {
                "rt":         "oic.ra.user",
                "jid":            "user@mydomain.com",
                "credential":   "AADRRRDSDSSDFERVVDESDFSDFSFSFDSSDF"
              }

  post:
    description: |
      Sets the new user data

    body:
      application/json :
        schema: |
          {
            "id": "http://openinterconnect.org/schemas/oic.ra.user#",
            "$schema": "http://json-schema.org/draft-04/schema#",
            "title": "XMPP server user information",
            "definitions": {
              "oic.ra.user": {
                "type": "object",
                "properties": {
                  "jid":  {
                    "type": "string",
                    "description": "the bare jid"
                  },
                  "credential":   {
                    "type": "string",
                    "description": "base64 encoded string, the credential"
                  }
                }
              }
            },
            "type": "object",
            "allOf": [
              {"$ref": "http://openinterconnect.org/schemas/oic.core.json#/definitions/oic.core"},
              {"$ref": "#/definitions/oic.ra.user"}
            ],
            "required": ["jid","credential"]
          }

        example: |
          {
                "rt":         "oic.ra.user",
                "jid":            "newuser@mydomain.com",
                "credential":   "NNAADRRRDSDSSDFERVVDESDFSDFSFSFDSSDF"
          }

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "id": "http://openinterconnect.org/schemas/oic.ra.user#",
                "$schema": "http://json-schema.org/draft-04/schema#",
                "title": "XMPP server user information",
                "definitions": {
                  "oic.ra.user": {
                    "type": "object",
                    "properties": {
                      "jid":  {
                        "type": "string",
                        "description": "the bare jid"
                      },
                      "credential":   {
                        "type": "string",
                        "description": "base64 encoded string, the credential"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  {"$ref":
"http://openinterconnect.org/schemas/oic.core.json#/definitions/oic.core"},
                  {"$ref": "#/definitions/oic.ra.user"}
                ],
                "required": ["jid","credential"]
              }

            example: |
              {
                "rt":          "oic.ra.user",
                "jid":             "newuser@mydomain.com",
                "credential":   "NNAADRRRDSDSSDFERVVDESDFSDFSFSFDSSDF"
               }
```

A.2.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
jid	string	yes	Read Write	the bare-JID
credential	string	yes	Read Write	base64 encoded string, the credential

A.2.6 CRUDN behaviour

Resource	Create	Read	Update	Delete	Notify
/XMPPUserResURI		get	post		