The Wayback Machine - https://web.archive.org/web/20130818191236/http://xmpp.org/extensions/xep-0327.html

XEP-0327: Rayo

Abstract: This specification defines an XMPP protocol extension for the third-party control of telephone calls and other similar media sessions. The protocol includes support for session management/signaling, as well as advanced media resources such as speech recognizers, speech synthesizers and audio/video recorders. The protocol serves a different purpose from that of first-party protocols such as Jingle or SIP, and is compatible with those protocols.

Authors: Ben Langfeld, Jose de Castro

Copyright: © 1999 - 2013 XMPP Standards Foundation. SEE LEGAL NOTICES.

Status: Experimental
Type: Standards Track
Version: 0.2
Last Updated: 2013-06-10

WARNING: This Standards-Track document is Experimental. Publication as an XMPP Extension Protocol does not imply approval of this proposal by the XMPP Standards Foundation. Implementation of the protocol described herein is encouraged in exploratory implementations, but production systems are advised to carefully consider whether it is appropriate to deploy implementations of this protocol before it advances to a status of Draft.

Table of Contents

1. Introduction
2. How it works
3. Requirements
4. Terminology
   4.1. Glossary
   4.2. Conventions
5. Concepts and Approach
   5.1. Actors
      5.1.1. Server
      5.1.2. Client(s)
      5.1.3. Calls
      5.1.4. Mixers
      5.1.5. Commands
      5.1.6. Components
   5.2. Addressing Scheme
   5.3. Delivery Mechanism
6. Session Flow
   6.1. Client Registration
   6.2. Session Establishment
      6.2.1. Outbound Call
         6.2.1.1. Errors
         6.2.1.2. Nested join
      6.2.2. Inbound Call
   6.3. Joining Calls
      6.3.1. Errors
      6.3.2. Unjoin Command
         6.3.2.1. Errors
      6.3.3. Unjoined Event
      6.3.4. Multiple Joins
   6.4. Mixers
   6.5. Component Execution
      6.5.1. Initial Errors
      6.5.2. Command Errors
      6.5.3. Output Component
         6.5.3.1. Join considerations
         6.5.3.2. Commands
         6.5.3.3. Events
         6.5.3.4. Completion
      6.5.4. Input Component
         6.5.4.1. Join considerations
         6.5.4.2. Commands
         6.5.4.3. Events
         6.5.4.4. Completion
      6.5.5. Prompt Component
         6.5.5.1. Join considerations
         6.5.5.2. Commands
         6.5.5.3. Events
         6.5.5.4. Completion
      6.5.6. Record Component
         6.5.6.1. Join considerations
         6.5.6.2. Commands
         6.5.6.3. Events
         6.5.6.4. Completion
   6.6. Session Termination
      6.6.1. Call Redirection
      6.6.2. Call Rejection
      6.6.3. Call Hangup
      6.6.4. Call End Notification
   6.7. Headers
7. Formal Definition

7.1. Header Element
7.2. Offer Element
7.3. Ringing Element
7.4. Answered Element
7.5. End Element
7.5.1. End Reason Element
7.6. Accept Element
7.7. Answer Element
7.8. Redirect Element
7.9. Reject Element
7.9.1. Reject Reason Element
7.10. Hangup Element
7.11. Dial Element
7.12. Join Element
7.13. Unjoin Element
7.14. Joined Element
7.15. Unjoined Element
7.16. Started Speaking Element
7.17. Stopped Speaking Element
7.18. Ref Element
7.19. Components
7.19.1. Stop Element
7.19.2. Complete Element
7.19.2.1. Complete Reason Element
7.19.3. Media Output
7.19.3.1. Output Element
7.19.3.2. Pause Element
7.19.3.3. Resume Element
7.19.3.4. SpeedUp Element
7.19.3.5. SpeedDown Element
7.19.3.6. VolumeUp Element
7.19.3.7. VolumeDown Element
7.19.3.8. Seek Element
7.19.3.9. Finish Element
7.19.3.10. MaxTime Element
7.19.4. Media Input
7.19.4.1. Input Element
7.19.4.2. Match
7.19.4.3. Noinput
7.19.4.4. Nomatch
7.19.5. Prompt
7.19.5.1. Prompt Element
7.19.6. Media Recording
7.19.6.1. Record Element
7.19.6.2. Pause Element
7.19.6.3. Resume Element
7.19.6.4. Recording Element
7.19.6.5. Max Duration Element
7.19.6.6. Initial Timeout Element
7.19.6.7. Final Timeout Element
8. Use Cases
9. Determining Support
10. Extending Rayo
11. Implementation Notes
12. Security Considerations
12.1. Denial of Service
12.2. Communication Through Gateways
12.3. Information Exposure
13. IANA Considerations
14. XMPP Registrar Considerations
14.1. Protocol Namespaces
14.2. Namespace Versioning
14.3. Rayo Components Registry
15. XML Schema
15.1. Rayo
15.2. Rayo Ext
15.3. Rayo Ext Complete
15.4. Rayo Output
15.5. Rayo Output Complete
15.6. Rayo Input
15.7. Rayo Input Complete
15.8. Rayo Prompt
15.9. Rayo Record
15.10. Rayo Record Complete
16. History
17. Acknowledgements

Appendices
A: Document Information
B: Author Information
C: Legal Notices
D: Relation to XMPP
E: Discussion Venue
F: Requirements Conformance
G: Notes
H: Revision History

1. Introduction

Rayo is a protocol to allow third-party remote control over media sessions, audio/video mixers and a variety of advanced media resources such as speech recognizers, speech synthesizers and audio/video recorders. These capabilities can be combined to create a wide variety of applications such as menu-based phone systems, in-game conferencing and anonymous dating services. Unlike Jingle or even SIP, a Rayo client is not concerned with being a party to either the session negotiation or the media stream itself.

- A Rayo server takes on the role of negotiating a media session between itself and some other endpoint, or between two external endpoints, by way of an implementation-specific means, be that Jingle, SIP, the public-switched telephone network, or anything else. The server may even bridge multiple networks.
- The server then presents the Rayo protocol as an interface to a Rayo client, allowing it to monitor and/or exercise third-party control over the established media sessions.
- The client has the option to accept/reject/answer inbound session requests, request the creation of outbound sessions and monitor their progress, execute media operations such as speech synthesis, speech recognition & recording, and to end sessions.

The relationship between the calling parties, the Rayo server and the Rayo client looks something like this:

```
    [caller] ----SIP---- [rayo server] ( -----Jingle---- [callee] ) optional
                              |
                              |
                          rayo client
```

This document defines the core Rayo protocol, and contains provisions for its extension by further specifications.

2. How it works

In order to understand the nature of a Rayo interaction, here we show a simple example of a control session.

Example 1. New call announces itself to a potential controlling party

```
<presence from='9f00061@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <c xmlns='http://jabber.org/protocol/caps'
     hash='sha-1'
     node='urn:xmpp:rayo:call:1'
     ver='QgayPKawpkPSDYmwT/WM94uAlu0='/>
  <offer xmlns='urn:xmpp:rayo:1'
         to='tel:+18003211212'
         from='tel:+13058881212'/>
</presence>
```

In this example, a call from 'tel:+13058881212' has reached the Rayo server 'shakespeare.lit' by calling 'tel:+18003211212', and been assigned an ID '9f00061'. The server has determined that 'juliet@capulet.lit' is a valid candidate to be the client to whom the server delegates control of the call, and so has directed an offer event to her 'balcony' resource.

The client, 'juliet@capulet.lit', then decides that it is able to handle the incoming call, and so accepts it from the server, thus gaining exclusive control and indicating to the calling party that the call will be processed and that it should ring.

Example 2. Potential controlling party attempts to become definitive controlling party by sending the call an accept command

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='hd721'>
  <accept xmlns='urn:xmpp:rayo:1'/>
</iq>
```

Example 3. Call acknowledges accept command to the (now) definitive controlling party

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='hd721'/>
```

Following confirmation from the server that the attempt to gain control of the call was successful, the client proceeds to answer the call, opening up the media stream between the caller and the server.

Example 4. Controlling party answers the call

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='43jo3'>
  <answer xmlns='urn:xmpp:rayo:1'/>
</iq>
```

Example 5. Call acknowledges answer command to controlling party

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='43jo3'/>
```

Once the client has confirmation that the call has been answered, it triggers the start of a media output component in order to play a message to the caller using a Text-to-speech (TTS) engine.

Example 6. Controlling party requests a new output component

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='j9d3j'>
  <output xmlns='urn:xmpp:rayo:output:1'
          voice='allison'>
    <document content-type="text/plain">
      <![CDATA[
        You have no new messages. Goodbye!
      ]]>
    </document>
  </output>
</iq>
```

Example 7. Call acknowledges request for new output component and provides its ID

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='j9d3j'>
  <ref xmlns='urn:xmpp:rayo:1' uri='xmpp:9f00061@call.shakespeare.lit/fgh4590'/>
</iq>
```

After confirmation that the output component was successfully created, the client then awaits notification of its completion.

Example 8. Output component announces its completion, giving the reason

```
<presence from='9f00061@call.shakespeare.lit/fgh4590'
          to='juliet@capulet.lit/balcony'
          type='unavailable'>
 <complete xmlns='urn:xmpp:rayo:ext:1'>
   <finish xmlns='urn:xmpp:rayo:output:complete:1' />
 </complete>
</presence>
```

The client then decides it has no further operations to perform on the call, and that the call should end. It instructs the server to hang up the call gracefully.

Example 9. Controlling party hangs up the call

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='f3wh8'>
  <hangup xmlns='urn:xmpp:rayo:1'/>
</iq>
```

Example 10. Call acknowledges hangup command to controlling party

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='f3wh8'/>
```

Example 11. Controlling party receives notification of the call being terminated

```
<presence from='9f00061@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'
          type='unavailable'>
  <end xmlns='urn:xmpp:rayo:1'>
    <hangup-command/>
  </end>
</presence>
```

3. Requirements

The protocol defined herein is designed to provide the following features:

1. **Call Control:** Incoming calls are "offered" to clients at which point they can be answered, rejected, redirected to another destination, etc. Outbound calls may also be made and monitored. Every attempt is made to be shield the Rayo client from the low level telephony protocol (e.g. SIP, Jingle, PSTN, etc).
2. **Audio File Playback:** A compatible Rayo server will fetch a file from a a specified URL and play the containing audio to the caller.
3. **Speech Synthesis / TTS:** In cases where dynamic data must be spoken, a Speech Synthesis engine may be used to play computer generated speech to the caller.
4. **Touch-tone Events / DTMF:** Rayo surfaces real-time event when the caller presses keys on their touch-tone keypad.
5. **Speech Recognition:** Enables the phone application to take spoken queues allowing for sophisticated voice-driven menus and directory services.
6. **Call Recording:** Can be used to capture the caller's voice (e.g. Voicemail) or both sides of the call for auditing and compliance purposes.
7. **Mixing:** Typically referred to as an audio "conference"; calls can be joined together so that the participants can hear each other in real-time.

Many third-party call control protocols have preceeded Rayo (see Asterisk's AGI/AMI, FreeSWITCH's eventsocket, Microsoft's TAPI, Java's JTAPI, Novell/AT&T's TSAPI, CSTA, etc). None of these protocols is ideal, and all have one or more of the following drawbacks:

- **Totally ground-up wire protocol** requiring implementation all the way down to the socket.
- **Platform/vendor/hardware specific** - each system implements its own proprietary protocol (in many cases, without a formal published specification) which does not allow easily porting an application from one back-end to another.
- **Synchronous interface** - Operations on calls or other entities are often blocking, and one must serialise all control messages.

- **Inconsistent** - evolved, rather than designed.
- **Lacking in scalability** - client/server sometimes tied one-to-one, servers rarely clustered, advanced message routing not possible.
- **Poor security** - lack of wire-level encryption, lack of or sub-standard authentication mechanisms, lack of or limited authorization mechanisms, lack of or poor sandboxing between multiple tenants on one system.
- **Inextensible** - The specification of extensions to the core protocol is either impossible or very difficult.

Rayo has been designed with these failings in mind, and intends to address many concerns not addressed by these earlier attempts. The following considerations were made:

- **Simple client library implementation** - XMPP client libraries exist in all modern languages, and many are of a high standard of quality and maturity.
- **Cross-platform standard** - The protocol must not expose any platform specifics and all elements should be candidates for implementation on any suitable platform. Addiditionally, the protocol must be ratified as a standard following a community discussion.
- **Asynchronous interface** - The protocol should present an asynchronous interface for the purposes of performance and flexibility in performing parallel operations.
- **Consistent** - The protocol must provide a considered, unobtrusive, logically and philisophically consistent interface.
- **Federated** - The protocol must support communication between client and server entities on separately owned, operated and addressed networks.
- **Flexible routing** - The protocol must lend itself to routing across wide networks such as the internet, and to potential complex routing such as proxying or redirection. Additionally, the client and server should each be aware of the presence of the other and be able to use such information to make routing decisions.
- **Extensible** - The protocol must provide for the possibility of extra functionality being added by future specifications or an individual implementation.
- **Secure** - The protocol should include appropriate measures for authentication and authorization of participants, as well as preventing third-parties from intercepting control messages.

Many of the features in the above list are available to Rayo at no specification or implementation cost, since they are core to XMPP itself and thus Rayo inherits these 'for free'.

Additionally, the protocol is required to abstract away the complexity of the back-end negotiation, especially the details of the transport protocols such as SIP or Jingle, but to map conceptually to such protocols.

4. Terminology

4.1 Glossary

Third-party call control (3PCC)
: The observation and/or control of a live media session by an entity which is not a direct party to the session.

Command
: Commands instruct the receiving entity to perform some atomic action. Commands may be executed against a given call, component or mixer and can be considered completed as soon as they receive a response. Some commands create components, and return a reference to the component in their response.

Component
: Components extend the Rayo protocol by providing additional media and call control functionality. Components are created by an appropriate command, which returns a reference to the component. Components are executed asynchronously, and have a lifecycle attached to them, with the ability to trigger events or have commands issued to it. Once a component is stopped or comes to an end naturally, it will issue a special <complete/> event, indicating that it has ceased executing and deliver any required data.

Potential controlling party (PCP)
: An XMPP entity to which an offer to control an incoming call may be sent.

Definitive controlling party (DCP)
: The XMPP entity which gains a lock on control of a session, either by requesting the session's creation, or being the first respondent to an offer.

Security Zone
: A security zone is the conceptual border around a call which defines which parties may interact with the call's media or signaling. A security zone MUST contain the rayo server's internal implementation, the media server to which the call is joined, the DCP, and any JID whose bare form is the same as the DCP. A server MAY relax this definition further, for example to consider all JIDs at the same domain to be in the same security zone.

4.2 Conventions

In examples, the following JIDs are used:

- **juliet@capulet.lit/balcony, romeo@montague.lit/orchard** - Potential controlling parties
- **shakespeare.lit** - The root domain of the Rayo service

5. Concepts and Approach

A complete Rayo deployment has several elements and interacting entities which must be understood.

5.1 Actors

5.1.1 Server

A Rayo server is an entity which is capable of receiving and initiating calls and being party to their media stream, while exposing a Rayo interface to a client in order to permit control over its calls. The Rayo server may handle calls in any way supported by the implementation, such as SIP, Jingle, etc, and should expose a full XMPP domain at the root level of the service deployment (eg shakespeare.lit).

The Rayo server is responsible for keeping track of valid clients, routing calls to the correct potential controlling parties, performing authorization measures on received stanzas, etc.

For the purposes of this specification, complex server-side deployments such as clusters, proxies, gateways, protocol translators, etc are not considered. Further details of such concepts may be found in their (present or future) relevant specifications.

5.1.2 Client(s)

A Rayo client is an entity which implements the Rayo protocol for the purpose of asserting control over calls made available by a Rayo server. The method by which such control measures are determined is outside the scope of this document, but may be the result of human interaction or some automated decision-making process.

A Rayo client is responsible for indicating its availability to a Rayo server and responding to offer messages appropriately.

5.1.3 Calls

A Rayo call is a short-lived XMPP entity within the scope of the deployment's root domain, perhaps at a sub-domain, with the purpose of representing a single session. It is usually a simple alias for the main server process.

A Rayo call is the entity with which most client interactions are made, and is responsible for sending its events to and receiving commands from a client. Calls may host components.

Calls have separate presence from the root domain of the service and thus appear to be separate entities.

5.1.4 Mixers

A Rayo mixer is an XMPP entity within the scope of the deployment's root domain, perhaps at a sub-domain, with the purpose of representing a service for the linking of media streams from several calls. It is usually a simple alias for the main server process.

A Rayo mixer is responsible for sending its events to and receiving commands from one or more clients, and can host components.

Mixers have separate presence from the root domain of the service and its calls and thus appear to be separate entities.

5.1.5 Commands

A Rayo command is a simple combination of request and response and may be issued directly to the service domain, a call, a mixer or a component attached to any of the former. Commands are executed serially and are generally very short-lived.

5.1.6 Components

Components extend the Rayo protocol by providing additional media and call control functionality.

Components have a lifecycle and are started by sending a specialized command to a call or mixer. Thus, a request for creation of a component will return a reference to the component's ID, and the component will continue to execute until it completes, potentially sending events and processing commands along the way (such as an instruction to pause or terminate), before finally issuing an event indicating its completion and thus unavailability. Multiple components may be active on a call or mixer at any one time, and commands may be executed on any entity during the execution of a component.

5.2 Addressing Scheme

All of the actors described in the previous section (with the exception of commands) are represented by XMPP entities with a JID of their own. Thus, a scheme for determining the JIDs of each of these entities is required. The following is the required naming scheme for Rayo deployments, where elements in square brackets are optional.

Table 1:

Actor	JID format	Example JID
Server	[service domain]	shakespeare.lit
Client	any JID	juliet@capulet.lit/balcony
Call	<call ID>@[<call sub-domain>.]<service domain>	f88eh2@call.shakespeare.lit
Mixer	<mixer name>@[<mixer sub-domain>.]<service domain>	conf1@mixer.shakespeare.lit
Call Component	<call ID>@[<call sub-domain>.]<service domain>/<component ID>	f88eh2@call.shakespeare.lit/8f83jf
Mixer Component	<mixer name>@[<mixer sub-domain>.]<service domain>/<component ID>	conf1@mixer.shakespeare.lit/932eu
Server Component	<service domain>/<component ID>	shakespeare.lit/f3fg4

Commands should be addressed to the entity on which they should be enacted. Individual commands only apply to certain object (for example instructing a component to hangup will return an error). In general, commands may be sent from a client to the service, a call, a mixer or a component. Events may be sent from a call, a mixer or a component to a client.

5.3 Delivery Mechanism

Rayo defines several events and commands which may be executed on one of the above actors. These payloads must be sent within an XMPP primitive element, and the rules are as such:

- **Events:** Sent as directed presence from the entity producing the event to a client.
- **Commands:** Sent as an <iq/> with the 'type' attribute 'set' from the client to the entity to be acted on. Responses returned as an <iq/> with the 'type' attribute either 'result' or 'error'.

6. Session Flow

This section describes the form, function and order of Rayo stanzas sent across the wire, and the circumstances in which they apply and/or may arise.

6.1 Client Registration

In order for a Rayo client to be considered a potential controlling party for incoming sessions, it MUST first notify the Rayo server that it is available for the receipt of calls. This is done by sending directed presence to the Rayo server with a <show/> element containing 'chat' as in the example:

Example 12. Client presents itself as available to the Rayo server

```
<presence from='juliet@capulet.lit/balcony'
          to='shakespeare.lit'>
  <c xmlns='http://jabber.org/protocol/caps'
     hash='sha-1'
     node='urn:xmpp:rayo:client:1'
     ver='QgayPKawpkPSDYmwT/WM94uAlu0='/>
 <show>chat</show>
</presence>
```

Conversely, when a Rayo client wishes not to be considered a potential controlling party, it SHOULD send directed presence to the Rayo server with a <show/> element containing 'dnd' as in the example:

Example 13. Client presents itself as unavailable to the Rayo server

```
<presence from='juliet@capulet.lit/balcony'
          to='shakespeare.lit'>
  <c xmlns='http://jabber.org/protocol/caps'
     hash='sha-1'
     node='urn:xmpp:rayo:client:1'
     ver='QgayPKawpkPSDYmwT/WM94uAlu0='/>
 <show>dnd</show>
</presence>
```

6.2 Session Establishment

Sessions may be established either at the request of the Rayo client (an outbound call) or as a result of a 3rd party request (an inbound call). Each scenario differs in the Rayo protocol only up to the point at which the session is established and media begins to flow. First we shall examine the sequence of stanzas passed between server and client in each of these scenarios.

6.2.1 Outbound Call

In order for a client to establish a new outbound call, it MUST first send a dial command to the server, indicating the proposed target for the call, its apparent source, and any meta-data to send to the target as headers.

Example 14. Client requests establishment of a new outbound session

```
<iq from='juliet@capulet.lit/balcony'
    to='shakespeare.lit'
    type='set'
    id='h7ed2'>
  <dial xmlns='urn:xmpp:rayo:1'
        to='tel:+13055195825'
        from='tel:+14152226789'>
    <header name="x-skill" value="agent" />
    <header name="x-customer-id" value="8877" />
  </dial>
</iq>
```

On successfully receiving and parsing the dial command, the server SHOULD perform its own proprietary authorization measures to ensure that only desirable outbound sessions are created. If it is established that the command should not be allowed, the server MUST return an error giving an authorization reason.

6.2.1.1 Errors

There are several reasons why the server might immediately return an error instead of acknowledging the creation of a new call:

- The client is unknown to the server and the server does not permit session creation by unknown clients.
- The client is not authorized to create this new session.
- The server does not support outbound calls.
- The server does not have sufficient resources to create a new session.
- The dial command was malformed.

If the client is unknown to the server and the server does not permit session creation by unknown clients, the server MUST return a <registration-required/> error with a type of 'auth'.

Example 15. Server indicates client is unknown and the server does not permit session creation by unknown clients

```
<iq from='shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <dial xmlns='urn:xmpp:rayo:1'
        to='tel:+13055195825'
        from='tel:+14152226789'>
    <header name="x-skill" value="agent" />
    <header name="x-customer-id" value="8877" />
  </dial>
  <error type='auth'>
    <registration-required xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the client is not authorized (as determined by an implementation/deployment-specific algorithm) to create a new outbound session given the parameters provided, the server MUST return a <not-

authorized/> error with a type of 'auth'.

Example 16. Server indicates client is not authorized to create a new outbound session given the parameters provided

```
<iq from='shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <dial xmlns='urn:xmpp:rayo:1'
        to='tel:+13055195825'
        from='tel:+14152226789'>
    <header name="x-skill" value="agent" />
    <header name="x-customer-id" value="8877" />
  </dial>
  <error type='auth'>
    <not-authorized xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the server does not support outbound calls, the server MUST return a <feature-not-implemented/> error with a type of 'cancel'.

Example 17. Server indicates that it does not support outbound calls

```
<iq from='shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <dial xmlns='urn:xmpp:rayo:1'
        to='tel:+13055195825'
        from='tel:+14152226789'>
    <header name="x-skill" value="agent" />
    <header name="x-customer-id" value="8877" />
  </dial>
  <error type='cancel'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the server does not have sufficient resources to create a new session, the server MUST return a <resource-constraint/> error with a type of 'wait'.

Example 18. Server indicates that it does not have sufficient resources to create a new session

```
<iq from='shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <dial xmlns='urn:xmpp:rayo:1'
        to='tel:+13055195825'
        from='tel:+14152226789'>
    <header name="x-skill" value="agent" />
    <header name="x-customer-id" value="8877" />
  </dial>
  <error type='wait'>
    <resource-constraint xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the dial command was malformed, the server MUST return a <bad-request/> error with a type of 'modify'.

Example 19. Server indicates the dial command was malformed

```
<iq from='shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <dial xmlns='urn:xmpp:rayo:1'
        to='foo:bar'
        from='tel:+14152226789'>
    <header name="x-skill" value="agent" />
    <header name="x-customer-id" value="8877" />
  </dial>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the command is successful and the call is queued, however, confirmation of such should be sent to the client, including a reference to the unique ID of the call. This call ID may be used to execute commands and filter events for the duration of the session.

Example 20. Confirmation of successful dial request and call ID

```
<iq from='shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'>
  <ref xmlns='urn:xmpp:rayo:1' uri='xmpp:9f00061@call.shakespeare.lit'/>
</iq>
```

Once the server receives notification that the session has been accepted by the third party, it should send a ringing event to the client to indicate such:

Example 21. Call announces its ringing state (accepted by 3rd party but not yet answered).

```
<presence from='9f00061@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <ringing xmlns='urn:xmpp:rayo:1'/>
</presence>
```

Similarly, once the server receives notification that the session has been answered, it should negotiate media between the dialed party and its local media server. Once media negotiation is complete, it should send an answered event to the client to indicate such:

Example 22. Call announces its answered state (media connected).

```
<presence from='9f00061@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <answered xmlns='urn:xmpp:rayo:1'/>
</presence>
```

6.2.1.2 Nested join

When sending a dial request, a client MAY specify a join target within the dial element:

Example 23. Client requests establishment of a new outbound session, with a nested join

```
<iq from='juliet@capulet.lit/balcony'
    to='shakespeare.lit'
    type='set'
    id='h7ed2'>
  <dial xmlns='urn:xmpp:rayo:1'
        to='tel:+13055195825'
        from='tel:+14152226789'>
    <join call-uri='xmpp:e8u398d902i90@call.shakespeare.lit' />
  </dial>
</iq>
```

In this case, the server MUST treat the session creation in the same way as without the join element, until the point of media negotiation. Here, the server should negotiate media as specified by the join element, in accordance with the rules defined in joining calls. Media MUST NOT be negotiated with the local media server, unless the join specifies so. The join operation MUST behave as described in joining calls.

6.2.2 Inbound Call

When the system receives a call from one of its connected networks, it MUST then expose that requested session to Rayo clients. It SHOULD use an implementation-specific routing mechanism to map incoming calls to some set of registered JIDs which are considered appropriate controlling parties. From this set, it SHOULD then remove any parties whom it can identify as being temporarily inappropriate for control (either unavailable based on presence, under too much load, or any other metric which the server has available). If, as a result, the set of Potentially Controlling Parties is empty, the server MUST reject the call with a 'decline' reason.

If the server can identify active Potential Controlling Parties, it MUST offer them control of the call simultaneously. The server must broadcast an offer on behalf of the call to all Potential Controlling Parties, using applicable to/from/header data from the incoming session. The server MUST also include entity capabilities information in the presence stanza containing the offer, in order to advertise the fact that the entity is a call, qualified by the node name "urn:xmpp:rayo:call:1".

Example 24. New call announces itself to a potential controlling party

```
<presence from='9f00061@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <c xmlns='http://jabber.org/protocol/caps'
     hash='sha-1'
     node='urn:xmpp:rayo:call:1'
     ver='QgayPKawpkPSDYmwT/WM94uAlu0='/>
  <offer xmlns='urn:xmpp:rayo:1'
         to='tel:+18003211212'
         from='tel:+13058881212'>
    <header name="x-skill" value="agent" />
    <header name="x-customer-id" value="8877" />
  </offer>
</presence>
```

Once the server has offered control, it MUST wait indefinitely for a response from a PCP. The server SHOULD monitor the availability of PCPs to whom offers have been sent. If they all cease to be PCPs (eg by going offline) then the call should be rejected in the same way as if there had not been any available PCPs to begin with.

If an offered PCP executes a command against the call, by sending a command node to the call's JID inside an IQ 'set', the server should execute the following routine:

1. If the call does not have a DCP, set it to the PCP from which the command originated.
2. If the call has a DCP, and the command did not originate from the DCP, return a conflict (cancel) error in response to the command of the following format:

   Example 25. Server indicates that the call already has another DCP and that control of the call is no longer available.

   ```
   <iq from='9f00061@call.shakespeare.lit'
       to='juliet@capulet.lit/balcony'
       type='error'
       id='h7ed2'>
     <accept xmlns='urn:xmpp:rayo:1'/>
     <error type='cancel'>
       <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </iq>
   ```

   otherwise;
3. If the command is an accept command, notify the remote calling party that the call has been accepted. Return an empty IQ result to the command issuing party to confirm successful execution.
4. If the command is an answer command, notify the remote calling party that the call has been answered and negotiate media between the calling party and the server's local media server. Return an empty IQ result to the command issuing party to confirm successful execution.
5. If the command is any other, handle it in the manner detailed in the following sections.

6.3 Joining Calls

Calls in a Rayo system are capable of having their media streams moved/manipulated. Once such manipulation is to join the media streams of two calls. In a scenario where callA and callB should be joined, the client MUST send a join command to either call (not both) specifying the call ID of the other call, like so:

Example 26. Client instructs callA to join to callB and the server acknowledges the join was completed

```
<iq from='juliet@capulet.lit/balcony'
    to='callA@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit'/>
</iq>

<iq from='callA@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

If the calls to be joined to each other are in the same security zone, the server MUST join the media streams of the two calls and return an empty IQ result to confirm that the operation has been successful. If the parties to be joined are not within the same security zone, an error should be returned as detailed below.

When calls are joined to each other by any mechanism, each call MUST dispatch a joined event specifying who they have been joined to:

Example 27. Call A and B were joined, both calls emit joined events

```
<presence from='callA@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <joined xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit'/>
</presence>

<presence from='callB@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <joined xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callA@call.shakespeare.lit'/>
</presence>
```

By default, the server MUST join the calls by bridging their audio through its local media server, with bidirectional media. In order to specify alternative behaviour, the client MAY specify a media option (either 'bridge' or 'direct') and/or a direction option (either 'duplex', 'send' or 'recv'), and the server MUST bridge accordingly.

6.3.1 Errors

There are several reasons why the call might return an error instead of acknowledging a join:

- The specified join party does not exist or cannot be found.
- The specified join party is inaccessible for the purposes of being joined due to security restrictions.
- The server does not have sufficient resources to complete the join.
- The join command was malformed.
- The specified media/direction options or their combination are not possible/supported.

If the specified join party does not exist or cannot be found, the server MUST return a <service-unavailable/> error with a type of 'cancel'.

Example 28. Call indicates that the specified join party does not exist or cannot be found

```
<iq from='callA@shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callC@call.shakespeare.lit'/>
  <error type='cancel'>
    <service-unavailable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the specified join party is inaccessible for the purposes of being joined due to security restrictions, the server MUST return a <not-allowed/> error with a type of 'cancel'.

Example 29. Call indicates that the specified join party is inaccessible for the purposes of being joined due to security restrictionss

```
<iq from='callA@shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callC@call.shakespeare.lit'/>
  <error type='cancel'>
    <not-allowed xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the server does not have sufficient resources to complete the join, the server MUST return a <resource-constraint/> error with a type of 'wait'.

Example 30. Call indicates that there are not sufficient resources to complete the join

```
<iq from='callA@shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit'/>
  <error type='wait'>
    <resource-constraint xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the join command was malformed (eg no call URI specified), the server MUST return a <bad-request/> error with a type of 'modify'.

Example 31. Call indicates that the join command was malformed

```
<iq from='callA@shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:'/>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the specified media/direction options or their combination are not possible/supported, the server MUST return a <feature-not-implemented/> error with a type of 'modify'.

Example 32. Call indicates that the specified media/direction options or their combination are not possible/supported.

```
<iq from='callA@shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit' media='direct' direction='recv'/>
  <error type='modify'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

6.3.2 Unjoin Command

When the client wishes to terminate an existing join, it MUST send an unjoin command specifying the join to break (call-id).

Example 33. Client instructs callA to unjoin from callB

```
<iq from='juliet@capulet.lit/balcony'
    to='callA@call.shakespeare.lit'
    type='set'
```

```
    id='h7ed2'>
  <unjoin xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit'/>
</iq>
```

The server MUST unjoin the media streams of the two calls, rejoin both to the media server and return an empty IQ result to confirm that the operation has been successful:

Example 34. CallA acknowledges unjoin from callB

```
<iq from='callA@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

Optionally, if no join is specified on the unjoin command, all existing joins must be broken:

Example 35. Client instructs callA to unjoin from every existing join

```
<iq from='juliet@capulet.lit/balcony'
    to='callA@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <unjoin xmlns='urn:xmpp:rayo:1'/>
</iq>

<iq from='callA@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

6.3.2.1 Errors

There are several reasons why the call might return an error instead of acknowledging an unjoin command:

- The specified join does not exist.
- The unjoin command was malformed.

If the specified join does not exist, the server MUST return a <service-unavailable/> error with a type of 'cancel'.

Example 36. Call indicates that the specified join does not exist

```
<iq from='callA@shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callC@call.shakespeare.lit'/>
  <error type='cancel'>
    <service-unavailable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the unjoin command was malformed (eg an empty call URI specified), the server MUST return a <bad-request/> error with a type of 'modify'.

Example 37. Call indicates that the join command was malformed

```
<iq from='callA@shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <unjoin xmlns='urn:xmpp:rayo:1' call-uri='xmpp:'/>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

6.3.3 Unjoined Event

Calls may be unjoined from other calls either in response to an unjoin command, as the result of one of the calls disconnecting, or as the result of an error. The server MUST monitor calls for being unjoined from another call, and emit an unjoined event when this is detected.

Example 38. CallA announces that it has been unjoined from callB, and vice versa

```
<presence from='callA@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <unjoined xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit'/>
</presence>

<presence from='callB@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <unjoined xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callA@call.shakespeare.lit'/>
```

```
</presence>
```

6.3.4 Multiple Joins

If a client wishes to modify the parameters of a join, it MUST send a new join command to the target call with the new parameters. The server MUST renegotiate media using the new parameters, and acknowledge the command's completion. The server MUST NOT re-send joined events.

Example 39. Client joins callA to callB in receive-only mode, and then 'upgrades' the join to full-duplex

```
<iq from='juliet@capulet.lit/balcony'
    to='callA@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit' direction='recv'/>
</iq>

<iq from='callA@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>

<presence from='callA@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <joined xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit'/>
</presence>

<presence from='callB@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <joined xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callA@call.shakespeare.lit'/>
</presence>

<iq from='juliet@capulet.lit/balcony'
    to='callA@call.shakespeare.lit'
    type='set'
    id='h7ed3'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit' direction='duplex'/>
</iq>

<iq from='callA@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed3'/>
```

Rayo calls SHOULD support being joined to more than one other call at a time, each join having different parameters. Creating a new join MUST NOT destroy existing joins. If a join is requested but cannot be created without destroying existing joins, the call MUST return a conflict (cancel) error.

Example 40. Call indicates that the requested joins cannot be created in parallel

```
<iq from='juliet@capulet.lit/balcony'
    to='callA@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit'/>
</iq>

<iq from='callA@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>

<presence from='callA@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <joined xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit'/>
</presence>

<presence from='callB@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <joined xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callA@call.shakespeare.lit'/>
</presence>

<iq from='juliet@capulet.lit/balcony'
    to='callA@call.shakespeare.lit'
    type='set'
    id='h7ed3'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callC@call.shakespeare.lit'/>
</iq>

<iq from='callA@shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed3'>
  <join xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callC@call.shakespeare.lit'/>
  <error type='cancel'>
```

```
    <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

6.4 Mixers

While calls may generally be joined peer-to-peer in any desirable combination, such an implementation is not necessarily scalable or practical to manage. Rayo, therefore, includes the concept of mixers, which are entities like calls, to which calls or other mixers may be joined in the same way as joining multiple calls directly. A mixer MUST be implicitly created the first time a call attempts to join it, MUST immediately broadcast presence to all controlling parties who have calls joined to it, and must respond to the join command with a reference to the mixer. The server MUST include entity capabilities information in the first presence stanza it sends, in order to advertise the fact that the entity is a mixer, qualified by the node name "urn:xmpp:rayo:mixer:1". A mixer MUST emit events (joined, unjoined) to all controlling parties who have calls joined to it, using the same semantics as joining calls.

In order to support friendly-named mixers without causing naming collisions between security zones, a server SHOULD represent a mixer internally using some alternative name scoped to the client's security zone and mapped to the friendly name/URI presented to the client for the emission of events and processing of commands. A server MUST NOT allow clients to interact with mixers allocated within other security zones either by observing their status or media.

Example 41. Client instructs call to join a mixer

```
<iq from='juliet@capulet.lit/balcony'
    to='callA@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <join xmlns='urn:xmpp:rayo:1' mixer-name='myMixer'/>
</iq>

<presence from='myMixer@mixer.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <c xmlns='http://jabber.org/protocol/caps'
     hash='sha-1'
     node='urn:xmpp:rayo:mixer:1'
     ver='QgayPKawpkPSDYmwT/WM94uAlu0='/>
</presence>

<iq from='callA@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'>
  <ref xmlns='urn:xmpp:rayo:1' uri='xmpp:myMixer@mixer.shakespeare.lit'/>
</iq>

<presence from='callA@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <joined xmlns='urn:xmpp:rayo:1' mixer-name='myMixer'/>
</presence>

<presence from='myMixer@mixer.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <joined xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callA@call.shakespeare.lit'/>
</presence>
```

Mixers MUST respect the normal rules of XMPP presence subscriptions. If a client sends directed presence to a mixer, the mixer MUST implicitly create a presence subscription for the client. On receiving unavailable presence, the mixer MUST stop sending events to the client.

The error conditions on joining a mixer are the same as for calls, as are the unjoin and join modification semantics. Additionally, mixers SHOULD be able to host components just like calls, following the rules defined for each component.

Example 42. Client renders output to a mixer

```
<iq from='juliet@capulet.lit/balcony'
    to='myMixer@mixer.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <output xmlns='urn:xmpp:rayo:output:1'/>
</iq>

<iq from='myMixer@mixer.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'>
  <ref xmlns='urn:xmpp:rayo:1' uri='xmpp:myMixer@mixer.shakespeare.lit/d38d3'/>
</iq>
```

If the media server providing the mixer supports active speaker detection, it MUST emit active speaker events to all clients with a presence subscription. Such events MUST indicate the start and end of speaking for a particular call ID joined to the mixer.

Example 43. Mixer indicates overlapping speaking status of two joined calls

```
<presence from='myMixer@mixer.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <started-speaking xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callA@call.shakespeare.lit'/>
```

```
</presence>

<presence from='myMixer@mixer.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <started-speaking xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit'/>
</presence>

<presence from='myMixer@mixer.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <stopped-speaking xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callB@call.shakespeare.lit'/>
</presence>

<presence from='myMixer@mixer.shakespeare.lit'
          to='juliet@capulet.lit/balcony'>
  <stopped-speaking xmlns='urn:xmpp:rayo:1' call-uri='xmpp:callA@call.shakespeare.lit'/>
</presence>
```

Once the last participant unjoins from the mixer, the mixer SHOULD be destroyed. When a mixer is destroyed, it MUST send unavailable presence to all entities which have a presence subscription.

6.5 Component Execution

Components are long-lived elements of a call or mixer which may execute in parallel, have a lifecycle (may send events and/or process commands during their execution, indicate their completion asynchronously) and typically implement media operations. A server SHOULD implement components in such a way that it is acceptable to execute multiple components of the same type or of differing types simultaneously. A server SHOULD implement all core components.

In the event that a call or mixer receives a command which triggers the execution of a component, it MUST use the normal command handling routine, schedule the component for immediate execution and return a reference to the requesting client as confirmation of the component's creation:

Example 44. Client requests execution of an output component

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <output xmlns='urn:xmpp:rayo:output:1'/>
</iq>
```

Example 45. Server provides reference to output component

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'>
  <ref xmlns='urn:xmpp:rayo:1' uri='xmpp:9f00061@call.shakespeare.lit/eh3u82'/>
</iq>
```

If a component execution command is received prior to the call being answered, the server MUST NOT answer the call, and SHOULD attempt to use early-media techniques to perform the relevant operation without answering the call. If such early-media is not possible, it MUST return an error indicating that the call state is incorrect (unexpected-request).

The whole command MUST be parsed up-front, and any applicable validation performed before acknowledgement of the command.

6.5.1 Initial Errors

There are several reasons why the server might immediately return an error instead of acknowledging the creation of a new component:

- The server does not implement the component.
- The server does not implement a particular option value for the component.
- Some aspect of the component does not comply with this specification.
- The server does not have sufficient resources to create a new component on this call/mixer.
- The component would cause a resource conflict with another component on this call/mixer.
- The call/mixer is not in the correct state to begin executing the component.

If the server does not implement the command/component, it should return a feature-not-implemented (cancel) error:

Example 46. Server indicates a lack of support for the requested component/command

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <output xmlns='urn:xmpp:rayo:output:1'/>
  <error type='cancel'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the server does not implement a particular option value for the command/component, it should return a feature-not-implemented (modify) error:

Example 47. Server indicates a lack of support for some aspect of the requested component/command

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <output xmlns='urn:xmpp:rayo:output:1'
          repeat-times='4'/>
  <error type='modify'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the command does not meet the specification, the server should return a bad-request (modify) error:

Example 48. Server indicates the requested component/command does not meet the specification

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <output xmlns='urn:xmpp:rayo:output:1'
          repeat-times='foo'/>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the server does not have sufficient resources to create the component, it should return a resource-constraint (wait) error:

Example 49. Server indicates a lack of resources to create the requested component

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <output xmlns='urn:xmpp:rayo:output:1'/>
  <error type='wait'>
    <resource-constraint xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the server is not able to create the component due to a resource conflict with another component, it should return a resource-constraint (wait) error:

Example 50. Server indicates a resource conflict with another component

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <output xmlns='urn:xmpp:rayo:output:1'/>
  <error type='wait'>
    <resource-constraint xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the server is not able to create the component due to the call being in an incorrect state, it should return an unexpected-request (wait) error:

Example 51. Server indicates incorrect call state

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <output xmlns='urn:xmpp:rayo:output:1'/>
  <error type='wait'>
    <unexpected-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

Once acknowleged, the component MUST begin execution according to its particular specification. During its execution, it MAY emit events relevant to its progress, and an implementation MUST be capable of emitting events specified for each component. Any events should be sent inside a directed presence element to the executing party.

During execution, the component MUST respond to commands addressed to it. Each component has its own set of commands, but all components have the 'stop' command in common. On receipt of the stop command, the component MUST acknowledge that it has been instructed to stop and gracefully cease its execution in whatever way is appropriate to the particular component.

Example 52. Client requests component stop, server confirms

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <stop xmlns='urn:xmpp:rayo:ext:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

6.5.2 Command Errors

There are several reasons why a component might return an error instead of acknowledging a command:

- The component does not implement the command.
- The component does not implement a particular option value for the command.
- Some aspect of the command does not comply with the components specification.
- The command is not appropriate for the component at its current stage of execution.
- The command is issued by a party other than that which created the component.

If the component does not implement the command, it should return a feature-not-implemented (cancel) error:

Example 53. Component indicates a lack of support for the requested command

```
<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <stop xmlns='urn:xmpp:rayo:ext:1'/>
  <error type='cancel'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the component does not implement a particular option/value for the command, it should return a feature-not-implemented (modify) error:

Example 54. Component indicates a lack of support for some option/value on the command

```
<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <stop xmlns='urn:xmpp:rayo:ext:1'/>
  <error type='modify'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If some aspect of the command does not comply with the component's spec, it should return a bad-request (modify) error:

Example 55. Component indicates some aspect of the command is not spec compliant

```
<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <stop xmlns='urn:xmpp:rayo:ext:1'/>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the command is not appropriate for the component's current stage of execution, it should return a unexpected-request (wait) error:

Example 56. Component indicates that the command is not appropriate for the component's current stage of execution

```
<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <stop xmlns='urn:xmpp:rayo:ext:1'/>
  <error type='wait'>
    <unexpected-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the command is issued by a party other than the component creator, it should return a conflict (cancel) error:

Example 57. Component indicates command cannot be executed by anyone other than the component owner

```
<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/courtyard'
    type='error'
    id='h7ed2'>
  <stop xmlns='urn:xmpp:rayo:ext:1'/>
  <error type='cancel'>
    <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

When the component ceases to execute, it MUST send a complete event with a valid reason to the requesting party as directed presence with a type of 'unavailable'.

Example 58. Component indicates it has completed due to being stopped

```
<presence from='9f00061@call.shakespeare.lit/eh3u28'
          to='juliet@capulet.lit/courtyard'
          type='unavailable'>
  <complete xmlns='urn:xmpp:rayo:ext:1'>
    <stop xmlns='urn:xmpp:rayo:ext:complete:1'/>
  </complete>
</presence>
```

Once a component is completed, or if it did not exist, the server should return an item-not-found (cancel) error as response to any commands:

Example 59. Non-existant component receives a command

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <stop xmlns='urn:xmpp:rayo:ext:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <stop xmlns='urn:xmpp:rayo:ext:1'/>
  <error type='cancel'>
    <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

6.5.3 Output Component

Media output is a core concept in Rayo, and is provided by the output component. The component allows media to be rendered to a call or a mixer, using the server's local media server. A server MUST support audio file playback and MUST support the text/uri-list document format. A server MAY support speech synthesis and MAY support SSML. The component is created using an <output/> command, containing one or more documents to render, along with a set of options to determine the nature of the rendering.

Example 60. Client renders a simple SSML document to a call

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <output xmlns='urn:xmpp:rayo:output:1'>
    <document content-type='application/ssml+xml'>
      <![CDATA[
        <?xml version="1.0"?>
        <!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN"
                           "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
        <speak version="1.0"
               xmlns="http://www.w3.org/2001/10/synthesis"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
                           http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
               xml:lang="en-US">
          <p>
            <s>You have 4 new messages.</s>
            <s>The first is from Stephanie Williams and arrived at <break/> 3:45pm.</s>
            <s>
              The subject is <prosody rate="-20%">ski trip</prosody>
            </s>
          </p>
        </speak>
      ]]>
    </document>
  </output>
```

```
</iq>
```

Example 61. Client renders a plain text document to a call

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <output xmlns='urn:xmpp:rayo:output:1'>
    <document content-type='text/plain'>
      <![CDATA[
        Thanks for calling, goodbye!
      ]]>
    </document>
  </output>
</iq>
```

The server MUST validate that it has apropriate resources/mechanisms to render the requested document before acknowledging the component creation.

6.5.3.1 Join considerations

In the case that an output component is executed on a call joined to other calls or mixers, the output SHOULD be rendered only to the call and not the joined parties (also known as 'whisper'). In the case that an output component is executed on a mixer, the output should be rendered into the mixer, such that all participants receive the output (also known as 'announce').

6.5.3.2 Commands

The output component implements several commands for manipulating the output during its execution.

A client may instruct an output component to pause by sending a pause command. The server MUST cause the media server to pause rendering, maintaining position within the document and allowing for later resumption.

Example 62. Client requests output component pause, server confirms

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <pause xmlns='urn:xmpp:rayo:output:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

A client may instruct an output component to resume rendering if it has previously been paused. The server MUST cause the media server to resume rendering at the last pause marker.

Example 63. Client requests output component resume, server confirms

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <resume xmlns='urn:xmpp:rayo:output:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

A client may instruct an output component to increase the rendering rate by a unit amount, defined by the media server. The server MUST cause the media server to perform the rate increase and acknowledge the command.

Example 64. Client requests output component speed up, server confirms

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <speed-up xmlns='urn:xmpp:rayo:output:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

A client may instruct an output component to decrease the rendering rate by a unit amount, defined by the media server. The server MUST cause the media server to perform the rate decrease and acknowledge the command.

Example 65. Client requests output component speed down, server confirms

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <speed-down xmlns='urn:xmpp:rayo:output:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

A client may instruct an output component to increase the rendering volume by a unit amount, defined by the media server. The server MUST cause the media server to perform the volume increase and acknowledge the command.

Example 66. Client requests output component volume up, server confirms

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <volume-up xmlns='urn:xmpp:rayo:output:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

A client may instruct an output component to decrease the rendering volume by a unit amount, defined by the media server. The server MUST cause the media server to perform the volume decrease and acknowledge the command.

Example 67. Client requests output component volume down, server confirms

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <volume-down xmlns='urn:xmpp:rayo:output:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

A client may instruct an output component to move the play marker forward or back in time by a specified amount before resuming output. The server MUST cause the media to seek as instructed and acknowledge the command.

The attributes of the <seek/> element are as follows.

Example 68. Client requests output seek forward by 20s, server confirms

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <seek xmlns='urn:xmpp:rayo:output:1' direction='forward' amount='20000'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

6.5.3.3 Events

The output component does not provide any intermediate events.

6.5.3.4 Completion

The output completion reason MUST be one of the core Rayo reasons, finish (indicating that the document finished rendering naturally) or max-time (indicating that the maximum time was exceeded). Output component completion does not provide any metadata.

Example 69. Component indicates it has completed due to reaching the end of the document

```
<presence from='9f00061@call.shakespeare.lit/eh3u28'
          to='juliet@capulet.lit/courtyard'
          type='unavailable'>
  <complete xmlns='urn:xmpp:rayo:ext:1'>
    <finish xmlns='urn:xmpp:rayo:output:complete:1'/>
  </complete>
</presence>
```

6.5.4 Input Component

Media input is a core concept in Rayo, and is provided by the input component. The component allows input to be collected from a call by way of either DTMF (dual-tone multi-frequency) or ASR (automatic speech recognition), using the server's local media server. A Rayo server MUST support DTMF input and MUST support SRGS XML grammars (application/srgs+xml). A server MAY suport speech input, and MAY support other grammar formats. The component is created using an <input/> command, containing one or more grammar documents by which to control input, along with a set of options to determine the nature of the collection.

Example 70. Client requests DTMF input collection from a call

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <input xmlns='urn:xmpp:rayo:input:1' mode='dtmf'>
    <grammar content-type='application/srgs+xml'>
      <![CDATA[
        <?xml version="1.0"?>
        <grammar mode="dtmf" version="1.0"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                                     http://www.w3.org/TR/speech-grammar/grammar.xsd"
                 xmlns="http://www.w3.org/2001/06/grammar">

          <rule id="digit">
            <one-of>
              <item> 0 </item>
              <item> 1 </item>
              <item> 2 </item>
              <item> 3 </item>
              <item> 4 </item>
              <item> 5 </item>
              <item> 6 </item>
              <item> 7 </item>
              <item> 8 </item>
              <item> 9 </item>
            </one-of>
          </rule>

          <rule id="pin" scope="public">
            <one-of>
              <item>
                <item repeat="4"><ruleref uri="#digit"/></item>
                #
              </item>
              <item>
                * 9
              </item>
            </one-of>
          </rule>
        </grammar>
      ]]>
    </grammar>
  </input>
</iq>
```

The server MUST validate that it has appropriate resources/mechanisms to collect the requested input before acknowledging the component creation.

6.5.4.1 Join considerations

In the case that an input component is executed on a call joined to other calls or mixers, the input SHOULD be collected only from the call and not the joined parties. Input components executed on a mixer SHOULD collect and combine input from all participants joined to the mixer.

6.5.4.2 Commands

The input component does not implement any intermediate commands, other than those specified for all components.

6.5.4.3 Events

The input component does not provide any intermediate events.

6.5.4.4 Completion

The input completion reason MUST be one of the core Rayo reasons, or one of the following reasons. Input component completion provides match metadata for the <finish/> reason only.

- match (indicating that one of the grammars matched the received input).
- noinput (indicating that no input was received before a timeout was encountered).
- nomatch (indicating that input was received which did not match any of the specified grammars).

If the media server reports a match to one of the provided grammars, the server MUST present the results of the match to the client by way of a match document in the format requested by the client. A server MUST be capable of supporting NLSML, and may support other formats.

Example 71. Component indicates it has completed due to one of the grammars returning a match

```
<presence from='9f00061@call.shakespeare.lit/eh3u28'
          to='juliet@capulet.lit/courtyard'
          type='unavailable'>
  <complete xmlns='urn:xmpp:rayo:ext:1'>
    <match xmlns='urn:xmpp:rayo:input:complete:1' content-type="application/nlsml+xml">
      <![CDATA[
        <result xmlns="http://www.ietf.org/xml/ns/mrcpv2" grammar="http://foodorder">
          <interpretation>
            <input mode="dtmf" confidence="100">1 2 3 4</input>
          </interpretation>
        </result>
      ]]>
    </match>
  </complete>
</presence>
```

6.5.5 Prompt Component

Prompt is a convenience component to wrap input and output components, combine their lifecycles, and allow input to barge-in on an output component in the standard sense.

Example 72. Client requests DTMF input collection from a call

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <prompt xmlns='urn:xmpp:rayo:prompt:1'>
    <output xmlns='urn:xmpp:rayo:output:1'>
      <document content-type='application/ssml+xml'>
        <![CDATA[
          <?xml version="1.0"?>
          <!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN"
                                 "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
          <speak version="1.0"
                 xmlns="http://www.w3.org/2001/10/synthesis"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
                              http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
                 xml:lang="en-US">
            <p>
              <s>Please enter your pin number now.</s>
            </p>
          </speak>
        ]]>
      </document>
    </output>

    <input xmlns='urn:xmpp:rayo:input:1' mode='dtmf'>
      <grammar content-type='application/srgs+xml'>
        <![CDATA[
          <?xml version="1.0"?>
          <grammar mode="dtmf" version="1.0"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                                       http://www.w3.org/TR/speech-grammar/grammar.xsd"
                   xmlns="http://www.w3.org/2001/06/grammar">

            <rule id="digit">
              <one-of>
                <item> 0 </item>
                <item> 1 </item>
                <item> 2 </item>
                <item> 3 </item>
                <item> 4 </item>
                <item> 5 </item>
                <item> 6 </item>
                <item> 7 </item>
                <item> 8 </item>
                <item> 9 </item>
              </one-of>
            </rule>

            <rule id="pin" scope="public">
              <one-of>
                <item>
```

```
                <item repeat="4"><ruleref uri="#digit"/></item>
                #
              </item>
              <item>
                * 9
              </item>
            </one-of>
          </rule>
        </grammar>
      ]]>
    </grammar>
  </input>
</prompt>
</iq>
```

The server MUST validate that it has appropriate resources/mechanisms to render the requested output and collect the requested input before acknowledging the component creation.

6.5.5.1 Join considerations

The prompt component follows the same combined join considerations as output and input components.

6.5.5.2 Commands

The prompt component implements implements all intermediate commands from output and input and behaves the same. If output component commands are executed after the output component has ceased executing, a <unexpected-request> error MUST be returned.

6.5.5.3 Events

The prompt component emits intermediate events from the nested output and input components.

6.5.5.4 Completion

The input completion reason MUST be one of the core Rayo reasons, or one of the Input component reasons. Events signalling completion of the output components are suppressed.

6.5.6 Record Component

Call recording is a core concept in Rayo, and is provided by the record component. The component allows media to be captured from a call or a mixer, using the server's local media server, stored, and made available to clients. The component is created using a <record/> command, with a set of options to determine the nature of the recording.

Example 73. Client requests a simple recording

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <record xmlns='urn:xmpp:rayo:record:1'/>
</iq>
```

The server MUST validate that it has apropriate resources/mechanisms to make the recording before acknowledging the component creation. The component MUST ignore any hints that it does not understand.

6.5.6.1 Join considerations

In the case that a record component is executed on a call joined to other calls or mixers, the direction attibute will specify if the sent audio, received audio, or both will be present in the recording.

In send mode, only the audio sent by the caller is recorded.

In recv mode, when just joined to the media server, should record TTS, audio playback, etc; when joined to another call, should record that other call's sending audio (probably a human talking) also. When joined to a mixer, should record the audio send from the mixer (other people talking) also.

Duplex mode is a combination of send and recv. The platform may mix these or record them as separate channels.

When executing a record against a mixer, send mode is not supported. Recv mode records audio from all mixer participants. Duplex is a clone of recv.

6.5.6.2 Commands

The record component implements several commands for manipulating the recording during its execution.

A client may instruct a record component to pause by sending a pause command. The server MUST cause the media server to pause recording, allowing for later appending.

Example 74. Client requests record component pause, server confirms

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <pause xmlns='urn:xmpp:rayo:record:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
```

```
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

A client may instruct a record component to resume recording if it has previously been paused. The server MUST cause the media server to resume recording, appending to the original file.

Example 75. Client requests record component resume, server confirms

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit/eh3u28'
    type='set'
    id='h7ed2'>
  <resume xmlns='urn:xmpp:rayo:record:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit/eh3u28'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

6.5.6.3 Events

The record component does not provide any intermediate events.

6.5.6.4 Completion

The record completion reason MUST be one of the core Rayo reasons, or one of the following reasons. Record component completion provides recording metadata in all cases.

- max-duration (indicating that the max duration was reached).
- initial-timeout (indicating that the initial timeout was triggered).
- final-timeout (indicating that the final timeout was triggered).

The server MUST present the recording for consumption by the client by way of recording meta-data on the complete reason, including a URI at which the recording may be fetched. It MUST provide uri, duration & size data as specified on the recording element.

Example 76. Component indicates it has completed due to being stopped, providing the recording

```
<presence from='9f00061@call.shakespeare.lit/eh3u28'
          to='juliet@capulet.lit/courtyard'
          type='unavailable'>
  <complete xmlns='urn:xmpp:rayo:ext:1'>
    <stop xmlns='urn:xmpp:rayo:ext:complete:1'/>
    <recording xmlns='urn:xmpp:rayo:record:complete:1' uri='xmpp:http://rayo.io/recordings/foo.mp3' duration='2
  </complete>
</presence>
```

6.6 Session Termination

Session termination may occur by one of several methods:

- An inbound call may be redirected by a PCP to some other target.
- An inbound call may be rejected by a PCP.
- An active call, whether inbound or outbound, may be hung up (gracefully ended) by a DCP.
- An active call may be hung up (gracefully ended) by the remote party.
- An outbound call may be rejected by the remote party.

A call end notification will be dispatched to the PCP if one of the following conditions is met:

- The call has been accepted and has a PCP.
- The call is outbound and implicitly has a PCP (the requesting party).

6.6.1 Call Redirection

If a client can determine a more appropriate target for an incoming call, it may wish to relay this information to the caller in the form of a URI (eg SIP). The client MUST do this before accepting a call. The target URI must be specified in the 'to' attribute of the redirect element.

Example 77. Client instructs a call to redirect, with some headers

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <redirect xmlns='urn:xmpp:rayo:1'
            to='sip:other@there.com'>
    <header name="x-skill" value="agent" />
    <header name="x-customer-id" value="8877" />
  </redirect>
</iq>
```

The server should send an appropriate redirection instruction to the underlying session.

If the server is able to successfully relay the redirection to the calling party, it should send an empty IQ result to confirm the command has completed execution:

Example 78. Server acknowledges successful redirect

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

If the server is unable to perform the redirect because the call has already been accepted, it should return a not-allowed (cancel) error indicating such:

Example 79. Server indicates that the call already has another DCP and that control of the call is no longer available.

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <redirect xmlns='urn:xmpp:rayo:1'
            to='sip:other@there.com'>
    <header name="x-skill" value="agent" />
    <header name="x-customer-id" value="8877" />
  </redirect>
  <error type='cancel'>
    <not-allowed xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

6.6.2 Call Rejection

If a client cannot handle an incoming call, it MAY reject it. The client MUST do this before accepting the call. The target URI must be specified in the 'to' attribute of the redirect element.

Example 80. Client instructs a call to reject, with some headers

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <reject xmlns='urn:xmpp:rayo:1'>
    <header name="x-reject-description" value="Sorry, she cannae take it!" />
  </reject>
</iq>
```

The server should reject the underlying session. If the server is able to do so successfully, it should send an empty IQ result to confirm the command has completed execution:

Example 81. Server acknowledges successful rejection

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

If the server is unable to perform the rejection because the call has already been accepted, it should return a not-allowed (cancel) error indicating such:

Example 82. Server indicates that the call already has another DCP and that control of the call is no longer available.

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <reject xmlns='urn:xmpp:rayo:1'>
    <header name="x-reject-description" value="Sorry, she cannae take it!" />
  </reject>
  <error type='cancel'>
    <not-allowed xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

6.6.3 Call Hangup

If a client wishes to end a call it should send a hangup command to the call instructing it to do so:

Example 83. Client instructs a call to hangup, with some headers

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
```

```
  <hangup xmlns='urn:xmpp:rayo:1'>
    <header name="x-call-result" value="4" />
  </hangup>
</iq>
```

The server should queue the call for immediate hangup and return a response indicating success of the command:

Example 84. Server acknowledges hangup queueing

```
<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='result'
    id='h7ed2'/>
```

The server MUST follow this sequence to hang up a call:

- Terminate all components of the call, and components MUST send a complete event indicating hangup as the cause.
- Terminate all joins on the call, sending unjoined events.
- Terminate the underlying session.

6.6.4 Call End Notification

The server MUST monitor a call's underlying session and react appropriately in the case that it comes to an end:

- The server MUST determine the reason for the call ending to be one of the appropriate end reasons.
- If the call ending was not a result of a hangup command from a client, the server MUST terminate all components on the call, which MUST send a complete event indicating hangup as the cause. The server MUST additionally terminate all joins on the call, sending unjoined events.
- The server MUST send an end event (of type unavailable) on behalf of the call, specifying the above determined reason as a child element, to all JIDs to which an offer was sent or from which a command was received.
- The server MUST NOT send any more events from a call which has ended and declared itself unavailable.
- The server MUST respond to any commands sent to an ended call (or one which never existed) with an item-not-found (cancel) error.

Example 85. Controlling party receives notification of the call being terminated due to a remote party hangup

```
<presence from='9f00061@call.shakespeare.lit'
          to='juliet@capulet.lit/balcony'
          type='unavailable'>
  <end xmlns='urn:xmpp:rayo:1'>
    <hangup/>
  </end>
</presence>
```

Example 86. Non-existant call receives a command

```
<iq from='juliet@capulet.lit/balcony'
    to='9f00061@call.shakespeare.lit'
    type='set'
    id='h7ed2'>
  <answer xmlns='urn:xmpp:rayo:1'/>
</iq>

<iq from='9f00061@call.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    type='error'
    id='h7ed2'>
  <answer xmlns='urn:xmpp:rayo:1'/>
  <error type='cancel'>
    <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

6.7 Headers

In elements which may carry child <header/> elements, a server or client MAY specify several header elements with the same name. In such cases, these MUST be considered to form a collection of ordered values for the key provided.

Example 87. Client requests establishment of a new outbound session with multiple values for the SIP Route header

```
<iq from='juliet@capulet.lit/balcony'
    to='shakespeare.lit'
    type='set'
    id='h7ed2'>
  <dial xmlns='urn:xmpp:rayo:1'
        to='tel:+13055195825'
        from='tel:+14152226789'>
    <header name="Route" value="foo" />
    <header name="Route" value="bar" />
  </dial>
</iq>
```

7. Formal Definition

7.1 Header Element

The <header/> element MUST be empty.

The attributes of the <header/> element are as follows.

Table 2: Attributes of Header Element

Attribute	Definition	Inclusion
name	A token giving the name by which the header may be known.	REQUIRED
value	The string value of the named header.	REQUIRED

7.2 Offer Element

Informs the recipient that a new call is available for control and invites it to take control using progress commands below.

The <offer/> element MAY contain one or more <header/> elements.

The attributes of the <offer/> element are as follows.

Table 3: Attributes of Offer Element

Attribute	Definition	Inclusion
to	The target URI for the call. May me a tel URI, SIP URI, a JID (for Jingle) or some other platform-specific addressing mechanism.	REQUIRED
from	The caller ID URI for the call. May be a tel URI, SIP URI, a JID (for Jingle) or some other platform-specific addressing mechanism.	OPTIONAL

7.3 Ringing Element

Indication that an outbound call has begun ringing, or accepted by the remote party.

The <ringing/> element MAY contain one or more <header/> elements.

The <ringing/> element has no attributes.

7.4 Answered Element

Indication that an outbound call has been answered and that the 3rd party negotiation has completed. At this point, the media stream should be open.

The <answered/> element MAY contain one or more <header/> elements.

The <answered/> element has no attributes.

7.5 End Element

Indication that the call has come to an end, giving the reason.

The <end/> element MUST contain a single end reason element. It MAY also contain one or more <header/> elements.

The <end/> element has no attributes.

7.5.1 End Reason Element

The following are valid end reason elements. Unless otherwise stated, they all MUST be empty, and they do not have any attributes.

<hungup/>
Indication that the call ended due to a normal hangup by the remote party.

<hangup-command/>
Indication that the call ended due to a normal hangup triggered by a hangup command.

<timeout/>
Indication that the call ended due to a timeout in contacting the remote party.

<busy/>
Indication that the call ended due to being rejected by the remote party subsequent to being accepted.

<rejected/>
Indication that the call ended due to being rejected by the remote party before being accepted.

<error/>
Indication that the call ended due to a system error.

7.6 Accept Element

Instructs the server to send notification to the calling party that the call will be dealt with and that ringing may begin.

The <accept/> element MAY contain one or more <header/> elements.

The <accept/> element has no attributes.

7.7 Answer Element

Instructs the server to pick up an incoming call and connect the media stream.

The <answer/> element MAY contain one or more <header/> elements.

The <answer/> element has no attributes.

7.8 Redirect Element

Instructs the calling party that the call will not be accepted and that instead it should try to call the URI indicated in the command.

The <redirect/> element MAY contain one or more <header/> elements.

The attributes of the <redirect/> element are as follows.

Table 4: Attributes of Offer Element

Attribute	Definition	Inclusion
to	The new target URI for the call to be redirected to.	REQUIRED

7.9 Reject Element

Instructs the server to reject the call with a given reason.

The <reject/> element MUST contain a single reject reason element. It MAY also contain one or more <header/> elements.

The <reject/> element has no attributes.

7.9.1 Reject Reason Element

The following are valid reject reason elements. Unless otherwise stated, they all MUST be empty, and they do not have any attributes.

<decline/>
Indicates that the controlling party refused the call for an unspecified reason, such as access control.

<busy/>
Indicates that the controlling party refused the call due to excess load.

<error/>
Indicates that the controlling party refused the call because some error occurred.

7.10 Hangup Element

Instructs the server to bring the call to an end naturally.

The <hangup/> element MAY contain one or more <header/> elements.

The <hangup/> element has no attributes.

7.11 Dial Element

Instructs the server to create a new call and surrender control of it to the requesting party.

The <dial/> element MAY contain one or more <header/> elements. It MAY contain one or more <join/> elements, instructing the server to join the new call in the indicated manner rather than the default (join to the local media server).

The attributes of the <dial/> element are as follows.

Table 5: Attributes of Dial Element

Attribute	Definition	Inclusion	Default
to	Indicates the party to whom the call should be directed.	REQUIRED	
from	Indicates the caller ID with which the call should appear to originate.	OPTIONAL	
timeout	Indicates the maximum time allowed for a response to be provided by the third party before the call should be considered to have come to an end.	OPTIONAL	-1

7.12 Join Element

Instructs the server to join the media streams of the call and the specified party, given direction and media negotiation parameters.

The <join/> element MUST be empty.

The attributes of the <join/> element are as follows.

Table 6: Attributes of Join Element

Attribute	Definition	Inclusion	Default
direction	Indicates the direction in which the media should flow between the call and the 3rd party. Must be one of the following values: • **"duplex"** - Indicates that media should flow in both directions between the parties. • **"send"** - Indicates that media should only flow from the target call to the third party. • **"recv"** - Indicates that media should only flow from the third party to the target call.	OPTIONAL	duplex
media	Indicates the manner in which the server should negotiate media between the two parties. Must be one of the following values: • **"bridge"** - Instructs the server to bridge the parties media streams via its local media server. • **"direct"** - Instructs the server to have the parties negotiate media directly with one another.	OPTIONAL	bridge

call-uri	Indicates the 3rd party call URI to which the target call should be joined.	REQUIRED unless mixer-name is set. MUST NOT be set if mixer-name is set.	
mixer-name	Indicates the mixer name to which the target call should be joined.	REQUIRED unless call-uri is set. MUST NOT be set if call-uri is set.	

7.13 Unjoin Element

Instructs the server to unjoin the media streams of the call and the specified party.

The <unjoin/> element MUST be empty.

The attributes of the <unjoin/> element are as follows.

Table 7: Attributes of Unjoin Element

Attribute	Definition	Inclusion
call-uri	Indicates the 3rd party call URI from which the target call should be unjoined.	REQUIRED unless mixer-name is set. MUST NOT be set if mixer-name is set.
mixer-name	Indicates the mixer name from which the target call should be unjoined.	REQUIRED unless call-uri is set. MUST NOT be set if call-uri is set.

7.14 Joined Element

Indicates that the call was successfully joined to the specified party.

The <joined/> element MUST be empty.

The attributes of the <joined/> element are as follows.

Table 8: Attributes of Joined Element

Attribute	Definition	Inclusion
call-uri	Indicates the 3rd party call URI to which the target call was joined.	REQUIRED unless mixer-name is set. MUST NOT be set if mixer-name is set.
mixer-name	Indicates the mixer name to which the target call was joined.	REQUIRED unless call-uri is set. MUST NOT be set if call-uri is set.

7.15 Unjoined Element

Indicates that the call ceased to be joined to the specified party.

The <unjoined/> element MUST be empty.

The attributes of the <unjoined/> element are as follows.

Table 9: Attributes of Unjoined Element

Attribute	Definition	Inclusion
call-uri	Indicates the 3rd party call URI from which the target call was unjoined.	REQUIRED unless mixer-name is set. MUST NOT be set if mixer-name is set.
mixer-name	Indicates the mixer name from which the target call was unjoined.	REQUIRED unless call-uri is set. MUST NOT be set if call-uri is set.

7.16 Started Speaking Element

Indicates that a call joined to a mixer with which the controlling party has an events subscription has activated a speech detector, providing its URI.

The <started-speaking/> element MUST be empty.

The attributes of the <started-speaking/> element are as follows.

Table 10: Attributes of Started Speaking Element

Attribute	Definition	Inclusion
call-uri	Indicates the URI of the call which has triggered the speech detector.	REQUIRED

7.17 Stopped Speaking Element

Indicates that a call joined to a mixer with which the controlling party has an events subscription has ceased activation of a speech detector, providing its URI.

The <stopped-speaking/> element MUST be empty.

The attributes of the <stopped-speaking/> element are as follows.

Table 11: Attributes of Stopped Speaking Element

Attribute	Definition	Inclusion
call-uri	Indicates the URI of the call which has triggered the speech detector.	REQUIRED

7.18 Ref Element

Used to give the address of a newly created resource, either a call or a component.

The <ref/> element MUST be empty.

The attributes of the <ref/> element are as follows.

Table 12: Attributes of Ref Element

Attribute	Definition	Inclusion
uri	Gives the URI of the new resource.	REQUIRED

7.19 Components

7.19.1 Stop Element

Instructs a component to come to an end before it completes naturally.

The <stop/> element MUST be empty.

The <stop/> element has no attributes.

7.19.2 Complete Element

Indicates that the component has come to an end and no further processing will occurr. Gives the reason for the termination.

The <complete/> element MUST contain exactly one child element, indicating the reason for the complete event being raised. The reason may be a core complete reason or a reason specific to a particular component.

The <complete/> element has no attributes.

7.19.2.1 Complete Reason Element

The following are valid complete reason elements. They all MAY contain further component-specific metadata elements, but they do not have any attributes.

<stop/>
: Indicates that the component came to an end because it was issued a stop command by the controlling party.

<hangup/>
: Indicates that the component came to an end because the call ended.

<error/>
: Indicates that the component came to an end because it encountered an error.

7.19.3 Media Output

An output component is used to instruct the server to generate audible output to a call or mixer.

7.19.3.1 Output Element

Instructs the server to begin an output component executing on the target call or mixer with the specified document and parameters.

The <output/> element MUST contain one or more <document/> elements. A server MUST support the application/ssml+xml content type, but MAY additionally support others.

The attributes of the <output/> element are as follows.

Table 13: Attributes of Output Element

Attribute	Definition	Possible Values	Default	Inclusion
start-offset	Indicates some offset through which the output should be skipped before rendering begins.	A positive integer in ms.	0	OPTIONAL
start-paused	Indicates wether or not the component should be started in a paused state to be resumed at a later time.	true|false	false	OPTIONAL
repeat-interval	Indicates the duration of silence that should space repeats of the rendered document.	A positive integer in ms.	0	OPTIONAL
repeat-times	Indicates the number of times the output should be played.	A positive integer or 0 to repeat forever.	1	OPTIONAL
max-time	Indicates the maximum amount of time for which the output should be allowed to run before being terminated. Includes repeats.	A positive integer in ms or -1 to disable.	-1	OPTIONAL
renderer	Indicates which media engine the server should use to render the Output. The server defines the possible values and falls back to the platform default if not specified.	An arbitrary string		OPTIONAL
voice	The voice with which to speak the requested document	Any voice supported by the TTS engine.		OPTIONAL

Presents a document for rendering by the output engine.

The <document/> element MUST have either a url attribute set OR a content type and a body, containing a document for output rendering enclosed within CDATA.

The attributes of the <document/> element are as follows.

Table 14: Attributes of Document Element

Attribute	Definition	Possible Values	Default	Inclusion
url	Provides a URI at which the document is available.	Any valid URI scheme supported by the server (eg HTTP).	none	REQUIRED unless content-type and content are set
content-type	Indicates the content type of the document provided as CDATA.	A document content type token		REQUIRED unless url is set

7.19.3.2 Pause Element

Instructs the server to pause the media output, but not terminate the component.

The <pause/> element MUST be empty.

The <pause/> element has no attributes.

7.19.3.3 Resume Element

Instructs the server to continue rendering the output from the last pause marker.

The <resume/> element MUST be empty.

The <resume/> element has no attributes.

7.19.3.4 SpeedUp Element

Instructs the server to increase the rate of output by a unit amount.

The <speed-up/> element MUST be empty.

The <speed-up/> element has no attributes.

7.19.3.5 SpeedDown Element

Instructs the server to decrease the rate of output by a unit amount.

The <speed-down/> element MUST be empty.

The <speed-down/> element has no attributes.

7.19.3.6 VolumeUp Element

Instructs the server to increase the volume of output by a unit amount.

The <volume-up/> element MUST be empty.

The <volume-up/> element has no attributes.

7.19.3.7 VolumeDown Element

Instructs the server to decrease the volume of output by a unit amount.

The <volume-down/> element MUST be empty.

The <volume-down/> element has no attributes.

7.19.3.8 Seek Element

Instructs the server to move the play marker of the output forward or back in time before resuming output.

The <seek/> element MUST be empty.

The attributes of the <seek/> element are as follows.

Table 15: Attributes of Seek Element

Attribute	Definition	Possible Values	Inclusion
direction	Indicates the direction in time in which to move the play marker.	forward|back	REQUIRED
amount	Indicates the duration by which to move the play marker.	A positive integer, in ms.	REQUIRED

7.19.3.9 Finish Element

Indicates that the output component came to an end as a result of reaching the end of the document to be rendered.

The <finish/> element MUST be empty.

The <finish/> element has no attributes.

7.19.3.10 MaxTime Element

Indicates that the output component came to an end due to the maximum time limit being reached.

The <max-time/> element MUST be empty.

The <max-time/> element has no attributes.

7.19.4 Media Input

An input component is used to instruct the server to gather media input from a call or mixer, using either DTMF or ASR.

7.19.4.1 Input Element

Instructs the server to begin an input detector of the specified mode, with certain attributes, governed by the rules provided in one or more grammar documents.

The <input/> element MUST contain one or more <grammar/> elements.

The attributes of the <input/> element are as follows.

Table 16: Attributes of Input Element

Attribute	Definition	Possible Values	Default	Inclusion
mode	The method by which to collect input.	any|dtmf|voice	any	OPTIONAL
terminator	Indicates a terminator token which, when encountered, should cause the input detection to cease.	A token string	none	OPTIONAL
recognizer	Indicates the name of the particular input processor to be engaged, used only for routing purposes (eg to choose which MRCP profile to invoke).	A token string	none	OPTIONAL
language	Specifies the recognition language to the recognizer.	Any valid ISO 639-3 language code	en-US	OPTIONAL
initial-timeout	Indicates the amount of time preceding input which may expire before a timeout is triggered.	Any positive integer in miliseconds, or -1 to disable.	-1	OPTIONAL
inter-digit-timeout	Indicates (in the case of DTMF input) the amount of time between input digits which may expire before a timeout is triggered.	Any positive integer in miliseconds, or -1 to disable.	-1	OPTIONAL
sensitivity	Indicates how sensitive the interpreter should be to loud versus quiet input. Higher values represent greater sensitivity.	A decimal value between 0 and 1.	0.5	OPTIONAL
min-confidence	Indicates the confidence threshold, below which a match is to be considered unreliable.	A decimal value between 0 and 1.	0	OPTIONAL
max-silence	Indicates the maximum period of silence which may be encountered during input gathering before a timeout is triggered.	Any positive integer in miliseconds, or -1 to disable.	-1	OPTIONAL
match-content-type	Indicates the required response document format.	Must support at least application/nlsml+xml, but may support others such as application/emma+xml.	application/nlsml+xml	OPTIONAL

Provides the grammar document by which the input detection should be governed.

The <grammar/> element MUST have either a url attribute set OR a content type and a body.

The attributes of the <grammar/> element are as follows.

Table 17: Attributes of Grammar Element

Attribute	Definition	Possible Values	Default	Inclusion
url	Provides a URI at which the grammar document is available.	Any valid URI scheme supported by the server (eg HTTP).	none	REQUIRED unless content-type and content are set
content-type	Indicates the content type of the grammar document provided as CDATA.	A grammar content type token	none	REQUIRED unless url is set

7.19.4.2 Match

Indicates that the component came to an end due to one of its grammars matching the received input.

The <match/> element MUST contain a valid response document within CDATA.

The attributes of the <matchr/> element are as follows.

Table 18: Attributes of Grammar Element

Attribute	Definition	Possible Values	Default	Inclusion
content-type	Indicates the content type of the result document provided as CDATA.	A result document content type token	application/nlsml+xml	REQUIRED

7.19.4.3 Noinput

Indicates that the component came to an end because a timeout was triggered before input was received.

The <noinput/> element MUST be empty.

The <noinput/> element has no attributes.

7.19.4.4 Nomatch

Indicates that the component came to an end because input was received which did not match any of the specified grammars.

The <nomatch/> element MUST be empty.

The <nomatch/> element has no attributes.

7.19.5 Prompt

An prompt component is a mixture of audio output and input, and is used to link the lifecycle of both such input may interrupt output via an arbitrary grammar.

7.19.5.1 Prompt Element

Instructs the server to begin an input detector of the specified mode, with certain attributes, governed by the rules provided in one or more grammar documents, while simultaneously rendering output.

The <prompt/> element MUST contain an <input/> element and an <output/> element.

The attributes of the <prompt/> element are as follows.

Table 19: Attributes of Prompt Element

Attribute	Definition	Possible Values	Default	Inclusion
barge-in	Whether or not the input detector is permitted to interrupt the output.	true|false	true	OPTIONAL

7.19.6 Media Recording

A record component is used to instruct the server to record audible or visual media for temporary or permanent storage.

7.19.6.1 Record Element

Instructs the server to begin recording input to the call to a file.

The <record/> element MAY contain one or more <hint/> elements.

The attributes of the <record/> element are as follows.

Table 20: Attributes of Record Element

Attribute	Definition	Possible Values	Default	Inclusion
format	File format used during recording.	A valid format token, such as 'mp3', 'wav', 'h264'. Implementation specific.	mp3	OPTIONAL

start-beep	Indicates whether subsequent record will be preceded with a beep.	true|false	false	OPTIONAL
stop-beep	Indicates whether subsequent record stop will be preceded with a beep.	true|false	false	OPTIONAL
start-paused	Whether subsequent record will start in PAUSE mode.	true|false	false	OPTIONAL
max-duration	Indicates the maximum duration for the recording.	Any positive integer in miliseconds, or -1 to disable.	-1	OPTIONAL
initial-timeout	Controls how long the recognizer should wait after the end of the prompt for the caller to speak before sending a Recorder event.	Any positive integer in miliseconds, or -1 to disable.	-1	OPTIONAL
final-timeout	Controls the length of a period of silence after callers have spoken to conclude they finished.	Any positive integer in miliseconds, or -1 to disable.	-1	OPTIONAL
direction	Indicates the direction of the call to record, meaning which call legs(s) are included in the resulting file, in case the call is joined to another or a mixer.	• **"duplex"** - Records both sent and received audio. • **"send"** - Indicates that only the audio sent from the caller is to be recorded. Not supported when Record is executed against a mixer. • **"recv"** - Indicates that only and all audio received by the caller is recorded.	duplex	OPTIONAL
mix	Whether all channels (call legs) should be mixed into a single recording channel.	true|false	false	OPTIONAL

Optional format-specific encoding hint

The <hint/> element MUST be empty.

The attributes of the <hint/> element are as follows.

Table 21: Attributes of Hint Element

Attribute	Definition	Inclusion
name	The name of the hint value as expected by the recorder.	REQUIRED
value	The value of the hint provided.	REQUIRED

7.19.6.2 Pause Element

Instructs the server to cease recording input but to leave the destination open for appending to permit resumption from the same point.

The <pause/> element MUST be empty.

The <pause/> element has no attributes.

7.19.6.3 Resume Element

Instructs the server to continue recording input, appending to the same destination.

The <resume/> element MUST be empty.

The <resume/> element has no attributes.

7.19.6.4 Recording Element

Provides the result of a recording, as a reference to its location.

The <recording/> element MUST be empty.

The attributes of the <recording/> element are as follows.

Table 22: Attributes of Recording Element

Attribute	Definition	Possible Values	Inclusion
uri	Indicates the URI at which the recording is made available.	A valid URI	REQUIRED
duration	Indicates the duration of the completed recording.	A positive integer in milliseconds.	REQUIRED
size	Indicates the filesize of the completed	A positive integer in	REQUIRED

	recording.	bytes.	

7.19.6.5 Max Duration Element

Indicates that the component came to an end due to the max duration being reached.

The <max-duration/> element MUST be empty.

The <max-duration/> element has no attributes.

7.19.6.6 Initial Timeout Element

Indicates that the component came to an end due to no input being detected before the initial-timeout.

The <initial-timeout/> element MUST be empty.

The <initial-timeout/> element has no attributes.

7.19.6.7 Final Timeout Element

Indicates that the component came to an end because no input had been detected for the final timeout duration.

The <final-timeout/> element MUST be empty.

The <final-timeout/> element has no attributes.

8. Use Cases

STRONGLY RECOMMENDED.

9. Determining Support

If an entity supports Rayo, it MUST advertise that fact by returning a feature of "urn:xmpp:rayo:0" (see Namespace Versioning regarding the possibility of incrementing the version number) in response to a **Service Discovery** [1] information request. The response MUST also include features for the application formats and transport methods supported by the responding entity, as described in the relevant specifications.

Example 88. Service Discovery Information Request - Client to Server

```
<iq from='kingclaudius@shakespeare.lit/castle'
    id='disco1'
    to='call.rayo.org'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Example 89. Service Discovery Information Response - Client to Server

```
<iq from='call.rayo.org'
    id='disco1'
    to='kingclaudius@shakespeare.lit/castle'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:rayo:0'/>
  </query>
</iq>
```

Example 90. Service Discovery Information Request - Server to Client

```
<iq from='call.rayo.org'
    id='disco1'
    to='laertes@shakespeare.lit/castle'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Example 91. Service Discovery Information Response - Server to Client

```
<iq from='laertes@shakespeare.lit/castle'
    id='disco1'
    to='call.rayo.org'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:rayo:client:0'/>
  </query>
</iq>
```

In order for an application to determine whether an entity supports this protocol, where possible it SHOULD use the dynamic, presence-based profile of service discovery defined in **Entity Capabilities** [2]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

10. Extending Rayo

Rayo is a protocol designed for extensibility. Rayo implementations and deployments have great flexibility in the way they map the Rayo protocol to their underlying transport and media layers, and the functionality they provide around the Rayo interface to the system.

Further commands and components may also be added to the Rayo protocol in order to extend its capabilities. Such extensions should be submitted to the XSF as ProtoXEPs and use namespaces aligning with the core component namespaces.

11. Implementation Notes

A server MUST document any cases where its behaviour differs from that in this specification (such as lack of support for particular options/components/etc) and return an error whenever a command is not understood. A server MUST NOT silently ignore any instructions.

12. Security Considerations

12.1 Denial of Service

Rayo sessions can be resource-intensive. Therefore, it is possible to launch a denial-of-service attack against an entity by burdening it with too many Rayo sessions. Care must be taken to accept sessions only from known entities and only if the entity's device is able to process such sessions.

12.2 Communication Through Gateways

Rayo communications can be enabled through gateways to non-XMPP networks, whose security characteristics can be quite different from those of XMPP networks. For example, on some SIP networks authentication is optional and "from" addresses can be easily forged. Care must be taken in communicating through such gateways.

12.3 Information Exposure

Mere negotiation of a Rayo session can expose sensitive information about the parties (e.g. IP addresses). Care must be taken in communicating such information, and end-to-end encryption should be used if the parties do not trust the intermediate servers or gateways.

13. IANA Considerations

This document requires no interaction with the **Internet Assigned Numbers Authority (IANA)** [3].

14. XMPP Registrar Considerations

14.1 Protocol Namespaces

This specification defines the following XML namespaces:

- urn:xmpp:rayo:1
- urn:xmpp:rayo:client:1
- urn:xmpp:rayo:call:1
- urn:xmpp:rayo:mixer:1
- urn:xmpp:rayo:ext:1
- urn:xmpp:rayo:ext:complete:1
- urn:xmpp:rayo:output:1
- urn:xmpp:rayo:output:complete:1
- urn:xmpp:rayo:input:1
- urn:xmpp:rayo:input:complete:1
- urn:xmpp:rayo:prompt:1
- urn:xmpp:rayo:record:1
- urn:xmpp:rayo:record:complete:1

The **XMPP Registrar** [4] includes the foregoing namespaces in its registry at <http://xmpp.org/registrar/namespaces.html>, as governed by **XMPP Registrar Function** [5].

14.2 Namespace Versioning

If the protocol defined in this specification undergoes a major revision that is not fully backward-compatible with an older version, or that contains significant new features, the XMPP Registrar shall increment the protocol version number found at the end of the XML namespaces defined herein, as described in Section 4 of **XEP-0053**.

14.3 Rayo Components Registry

The XMPP Registrar maintains a registry of Rayo components. All component registrations with the exception of those defined above shall be defined in separate specifications (not in this document). Components defined within the XEP series MUST be registered with the XMPP Registrar, resulting in protocol URNs of the form "urn:xmpp:rayo:component_name:X" (where "component_name" is the registered name of the component and "X" is a non-negative integer).

In order to submit new values to this registry, the registrant shall define an XML fragment of the following form and either include it in the relevant XMPP Extension Protocol or send it to the email address <registrar@xmpp.org>:

```
<component>
  <name>The name of the component.</name>
  <desc>A natural-language summary of the component.</desc>
  <doc>The document in which the component is specified.</doc>
</component>
```

15. XML Schema

15.1 Rayo

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xmpp:rayo:1"
  xmlns:tns="urn:xmpp:rayo:1"
  elementFormDefault="qualified">

  <annotation>
    <documentation>
      The protocol documented by this schema is defined at http://rayo.org/xep
```

```
  </documentation>
</annotation>

<!-- Header elements -->
<complexType name="headerType">
  <attribute name="name" type="token" use="required">
    <annotation>
      <documentation>
        A token giving the name by which the header may be known.
      </documentation>
    </annotation>
  </attribute>
  <attribute name="value" type="string" use="required">
    <annotation>
      <documentation>
        The string value of the named header.
      </documentation>
    </annotation>
  </attribute>
</complexType>

<!-- Offer Event -->
<element name="offer">
  <annotation>
    <documentation>
      Informs the recipient that a new call is available for control and invites it to take control using progress comm
    </documentation>
  </annotation>
  <complexType>
    <attribute name="to" type="anyURI" use="required">
      <annotation>
        <documentation>
          The target URI for the call. May be a tel URI, SIP URI, a JID (for Jingle) or some other platform-specific ac
        </documentation>
      </annotation>
    </attribute>
    <attribute name="from" type="anyURI" use="optional">
      <annotation>
        <documentation>
          The caller ID URI for the call. May be a tel URI, SIP URI, a JID (for Jingle) or some other platform-specific
        </documentation>
      </annotation>
    </attribute>
    <sequence>
      <element name="header" type="tns:headerType" minOccurs="0" maxOccurs="unbounded">
        <annotation>
          <documentation>
            Set of header variables sent by the originating party (eg SIP INVITE headers).
          </documentation>
        </annotation>
      </element>
    </sequence>
  </complexType>
</element>

<complexType name="callProgressType">
  <sequence>
    <element name="header" type="tns:headerType" minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>

<!-- Ringing Event -->
<element name="ringing" type="tns:callProgressType">
  <annotation>
    <documentation>
      Indication that an outbound call has begun ringing, or accepted by the remote party.
    </documentation>
  </annotation>
</element>

<!-- Answered Event -->
<element name="answered" type="tns:callProgressType">
  <annotation>
    <documentation>
      Indication that an outbound call has been answered and that the 3rd party negotiation has completed. At this poi
    </documentation>
  </annotation>
</element>

<!-- End Event -->
<element name="end">
  <annotation>
    <documentation>
      Indication that the call has come to an end, giving the reason.
    </documentation>
  </annotation>
  <complexType>
```

```
    <sequence>
      <choice>
        <element name="hungup" type="tns:empty">
          <annotation>
            <documentation>
              Indication that the call ended due to a normal hangup by the remote party.
            </documentation>
          </annotation>
        </element>
        <element name="hangup-command" type="tns:empty">
          <annotation>
            <documentation>
              Indication that the call ended due to a normal hangup triggered by a hangup command.
            </documentation>
          </annotation>
        </element>
        <element name="timeout" type="tns:empty">
          <annotation>
            <documentation>
              Indication that the call ended due to a timeout in contacting the remote party.
            </documentation>
          </annotation>
        </element>
        <element name="busy" type="tns:empty">
          <annotation>
            <documentation>
              Indication that the call ended due to being rejected by the remote party subsequent to being accepted.
            </documentation>
          </annotation>
        </element>
        <element name="rejected" type="tns:empty">
          <annotation>
            <documentation>
              Indication that the call ended due to being rejected by the remote party before being accepted.
            </documentation>
          </annotation>
        </element>
        <element name="error" type="tns:empty">
          <annotation>
            <documentation>
              Indication that the call ended due to a system error.
            </documentation>
          </annotation>
        </element>
      </choice>
      <element name="header" type="tns:headerType" minOccurs="0" maxOccurs="unbounded">
        <annotation>
          <documentation>
           Set of header variables sent by the remote party along with the indication of the call ending.
          </documentation>
        </annotation>
      </element>
    </sequence>
  </complexType>
</element>

<!-- Accept Command -->
<element name="accept">
  <annotation>
    <documentation>
      Instructs the server to send notification to the calling party that the call will be dealt with and that ringing
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="header" type="tns:headerType" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<!-- Answer Command -->
<element name="answer">
  <annotation>
    <documentation>
      Instructs the server to pick up an incoming call and connect the media stream.
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="header" type="tns:headerType" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<!-- Redirect Command -->
<element name="redirect">
  <annotation>
```

```
      <documentation>
        Instructs the calling party that the call will not be accepted and that instead it should try to call the URI inc
      </documentation>
    </annotation>
    <complexType>
      <attribute name="to" type="anyURI" use="required">
        <annotation>
          <documentation>
            The new target URI for the call to be redirected to.
          </documentation>
        </annotation>
      </attribute>
      <sequence>
        <element name="header" type="tns:headerType" minOccurs="0" maxOccurs="unbounded" />
      </sequence>
    </complexType>
  </element>

  <!-- Reject Command -->
  <element name="reject">
    <annotation>
      <documentation>
        Instructs the server to reject the call with a given reason.
      </documentation>
    </annotation>
    <complexType mixed="true">
      <sequence>
        <choice>
          <element name="decline" type="tns:empty">
            <annotation>
              <documentation>
                Indicates that the controlling party refused the call for an unspecified reason, such as access control.
              </documentation>
            </annotation>
          </element>
          <element name="busy" type="tns:empty">
            <annotation>
              <documentation>
                Indicates that the controlling party refused the call due to excess load.
              </documentation>
            </annotation>
          </element>
          <element name="error" type="tns:empty">
            <annotation>
              <documentation>
                Indicates that the controlling party refused the call because some error occurred.
              </documentation>
            </annotation>
          </element>
        </choice>
        <element name="header" type="tns:headerType" minOccurs="0" maxOccurs="unbounded" />
      </sequence>
    </complexType>
  </element>

  <!-- Hangup Command -->
  <element name="hangup">
    <annotation>
      <documentation>
        Instructs the server to bring the call to an end naturally.
      </documentation>
    </annotation>
    <complexType>
      <sequence>
        <element name="header" type="tns:headerType" minOccurs="0" maxOccurs="unbounded" />
      </sequence>
    </complexType>
  </element>

  <!-- Dial Command -->
  <element name="dial">
    <annotation>
      <documentation>
        Instructs the server to create a new call and surrender control of it to the requesting party.
      </documentation>
    </annotation>
    <complexType>
      <attribute name="to" type="anyURI" use="required">
        <annotation>
          <documentation>
            Indicates the party to whom the call should be directed.
          </documentation>
        </annotation>
      </attribute>
      <attribute name="from" type="anyURI" use="optional">
        <annotation>
          <documentation>
```

```
          Indicates the caller ID with which the call should appear to originate.
        </documentation>
      </annotation>
    </attribute>
    <attribute name="timeout" type="tns:timeoutType" use="optional" default="-1">
      <annotation>
        <documentation>
          Indicates the maximum time allowed for a response to be provided by the third party before the call should be
        </documentation>
      </annotation>
    </attribute>
    <sequence>
      <element name="header" type="tns:headerType" minOccurs="0" maxOccurs="unbounded" />
      <element name="join" type="tns:joinCommandType" minOccurs="0" maxOccurs="unbounded">
        <annotation>
          <documentation>
            Instructs the server to join the new call in the indicated manner rather than the default (join to the loca
          </documentation>
        </annotation>
      </element>
    </sequence>
  </complexType>
</element>

<!-- Join Command -->
<element name="join">
  <annotation>
    <documentation>
      Instructs the server to join the media streams of the call and the specified party, given direction and media neg
    </documentation>
  </annotation>
  <complexType>
    <complexContent>
      <extension base="tns:joinType">
        <attribute name="direction" use="optional" default="duplex">
          <annotation>
            <documentation>
              Indicates the direction in which the media should flow between the call and the 3rd party.
            </documentation>
          </annotation>
          <simpleType>
            <restriction base="token">
              <enumeration value="duplex">
                <annotation>
                  <documentation>
                    Indicates that media should flow in both directions between the parties.
                  </documentation>
                </annotation>
              </enumeration>
              <enumeration value="send">
                <annotation>
                  <documentation>
                    Indicates that media should only flow from the target call to the third party.
                  </documentation>
                </annotation>
              </enumeration>
              <enumeration value="recv">
                <annotation>
                  <documentation>
                    Indicates that media should only flow from the third party to the target call.
                  </documentation>
                </annotation>
              </enumeration>
            </restriction>
          </simpleType>
        </attribute>
        <attribute name="media" use="optional" default="bridge">
          <annotation>
            <documentation>
              Indicates the manner in which the server should negotiate media between the two parties.
            </documentation>
          </annotation>
          <simpleType>
            <restriction base="token">
              <enumeration value="bridge">
                <annotation>
                  <documentation>
                    Instructs the server to bridge the parties media streams via its local media server.
                  </documentation>
                </annotation>
              </enumeration>
              <enumeration value="direct">
                <annotation>
                  <documentation>
                    Instructs the server to have the parties negotiate media directly with one another.
                  </documentation>
                </annotation>
```

```
            </enumeration>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
</element>

<!-- Unjoin Command -->
<element name="unjoin" type="tns:joinType">
  <annotation>
    <documentation>
      Instructs the server to unjoin the media streams of the call and the specified party.
    </documentation>
  </annotation>
</element>

<!-- Joined Event -->
<element name="joined" type="tns:joinType">
  <annotation>
    <documentation>
      Indicates that the call was successfully joined to the specified party.
    </documentation>
  </annotation>
</element>

<!-- Unjoined Event -->
<element name="unjoined" type="tns:joinType">
  <annotation>
    <documentation>
      Indicates that the call ceased to be joined to the specified party.
    </documentation>
  </annotation>
</element>

<complexType name="joinType">
  <attribute name="call-uri" type="anyURI" use="optional">
    <annotation>
      <documentation>
        Indicates the 3rd party call URI to which the target call should be joined. May not be set if the mixer-name at
      </documentation>
    </annotation>
  </attribute>
  <attribute name="mixer-name" type="token" use="optional">
    <annotation>
      <documentation>
        Indicates the mixer name to which the target call should be joined. May not be set if the call-id attribute is
      </documentation>
    </annotation>
  </attribute>
</complexType>

<!-- Started Speaking Event -->
<element name="started-speaking" type="tns:activeSpeakerType">
  <annotation>
    <documentation>
      Indicates that a call joined to a mixer with which the controlling party has an events subscription has activated
    </documentation>
  </annotation>
</element>

<!-- Stopped Speaking Event -->
<element name="stopped-speaking" type="tns:activeSpeakerType">
  <annotation>
    <documentation>
      Indicates that a call joined to a mixer with which the controlling party has an events subscription has ceased ac
    </documentation>
  </annotation>
</element>

<complexType name="activeSpeakerType">
  <attribute name="call-uri" type="anyURI" use="required">
    <annotation>
      <documentation>
        Indicates the URI of the call which has triggered the speech detector.
      </documentation>
    </annotation>
  </attribute>
</complexType>

<!-- Resource Reference -->
<element name="ref">
  <annotation>
    <documentation>
      Used to give an indication of the identity of a newly created resource, either a call or a component.
    </documentation>
```

```
    </annotation>
    <complexType>
      <attribute name="uri" type="anyURI" use="required">
        <annotation>
          <documentation>
            Gives the URI of the new resource.
          </documentation>
        </annotation>
      </attribute>
    </complexType>
  </element>

  <!-- Utility: Empty Type -->
  <simpleType name="empty">
    <restriction base="string">
      <enumeration value='' />
    </restriction>
  </simpleType>

  <!-- Utility: Duration Type -->
  <simpleType name="durationType">
    <restriction base="long">
      <annotation>
        <documentation>
          Value is a duration in milleseconds
        </documentation>
      </annotation>
    </restriction>
  </simpleType>

  <!-- Utility: Timeout Type -->
  <simpleType name="timeoutType">
    <annotation>
      <documentation>
        A value of -1 indicates no timeout
      </documentation>
    </annotation>

    <restriction base="tns:durationType">
      <minInclusive value="-1"/>
    </restriction>
  </simpleType>

  <!-- Utility: Fraction Decimal Type -->
  <simpleType name="fractionDecimalType">
    <restriction base="decimal">
      <minInclusive value="0"/>
      <maxInclusive value="1"/>
    </restriction>
  </simpleType>

</schema>
```

15.2 Rayo Ext

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xmpp:rayo:ext:1"
  xmlns:tns="urn:xmpp:rayo:ext:1"
  elementFormDefault="qualified"
  xmlns:core="urn:xmpp:rayo:1">

  <annotation>
    <documentation>
      The protocol documented by this schema is defined at http://rayo.org/xep
    </documentation>
  </annotation>

  <!-- Stop Command -->
  <element name="stop" type="core:empty">
    <annotation>
      <documentation>
        Instructs a component to come to an end before it completes naturally.
      </documentation>
    </annotation>
  </element>

  <!-- Complete Event -->
  <element name="complete">
    <annotation>
      <documentation>
        Indicates that the component has come to an end and no further processing will occurr. Gives the reason for the t
      </documentation>
    </annotation>
    <complexType mixed="true">
      <choice minOccurs="1" maxOccurs="1">
```

```
        <any>
          <annotation>
            <documentation>
              The reason for component termination. May be either one of the core termination reasons (stop, hangup, erro
            </documentation>
          </annotation>
        </any>
      </choice>
      <sequence>
        <any minOccurs="0" maxOccurs="unbounded">
          <annotation>
            <documentation>
              May be any component specific meta-data elements, such as <recording>.
            </documentation>
          </annotation>
        </any>
      </sequence>
    </complexType>
  </element>

</schema>
```

15.3 Rayo Ext Complete

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xmpp:rayo:ext:complete:1"
  xmlns:tns="urn:xmpp:rayo:ext:complete:1"
  elementFormDefault="qualified"
  xmlns:core="urn:xmpp:rayo:1">

  <annotation>
    <documentation>
      The protocol documented by this schema is defined at http://rayo.org/xep
    </documentation>
  </annotation>

  <!-- Complete due to a <stop/> command -->
  <element name="stop" type="core:empty">
    <annotation>
      <documentation>
        Indicates that the component came to an end because it was issued a stop command by the controlling party.
      </documentation>
    </annotation>
  </element>

  <!-- Complete due to a hangup -->
  <element name="hangup" type="core:empty">
    <annotation>
      <documentation>
        Indicates that the component came to an end because the call ended.
      </documentation>
    </annotation>
  </element>

  <!-- Complete due to a system error -->
  <element name="error" type="string">
    <annotation>
      <documentation>
        Indicates that the component came to an end because it encountered an error.
      </documentation>
    </annotation>
  </element>

</schema>
```

15.4 Rayo Output

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xmpp:rayo:output:1"
  xmlns:tns="urn:xmpp:rayo:output:1"
  elementFormDefault="qualified"
  xmlns:core="urn:xmpp:rayo:1">

  <annotation>
    <documentation>
      The protocol documented by this schema is defined at http://rayo.org/xep
    </documentation>
  </annotation>

  <complexType name="documentType">
    <simpleContent>
      <attribute name="url" type="anyURI" use="optional">
        <annotation>
```

```
        <documentation>
          Provides a URI at which the document is available. Must not be provided if the content-type attribute is set
        </documentation>
      </annotation>
    </attribute>
    <attribute name="content-type" type="string" use="optional">
      <annotation>
        <documentation>
          Indicates the content type of the document provided as CDATA. Must not be set if the url attribute is set.
        </documentation>
      </annotation>
    </attribute>

    <restriction base="CDATA" />
  </simpleContent>
</complexType>

<!-- Main output command -->
<element name="output">
  <annotation>
    <documentation>
      Instructs the server to begin an output component executing on the target call or mixer with the specified docume
    </documentation>
  </annotation>
  <complexType>
    <attribute name="start-offset" type="core:durationType" use="optional" default="0">
      <annotation>
        <documentation>
          Indicates some offset through which the output should be skipped before rendering begins.
        </documentation>
      </annotation>
    </attribute>
    <attribute name="start-paused" type="boolean" use="optional" default="false">
      <annotation>
        <documentation>
          Indicates wether or not the component should be started in a paused state to be resumed at a later time.
        </documentation>
      </annotation>
    </attribute>
    <attribute name="repeat-interval" type="core:durationType" use="optional" default="0">
      <annotation>
        <documentation>
          Indicates the duration of silence that should space repeats of the rendered document.
        </documentation>
      </annotation>
    </attribute>
    <attribute name="repeat-times" type="nonNegativeInteger" use="optional" default="1">
      <annotation>
        <documentation>
          Indicates the number of times the output should be played.
        </documentation>
      </annotation>
    </attribute>
    <attribute name="max-time" type="core:timeoutType" use="optional" default="-1">
      <annotation>
        <documentation>
          Indicates the maximum amount of time for which the output should be allowed to run before being terminated. I
        </documentation>
      </annotation>
    </attribute>
    <attribute name="renderer" type="string" use="optional">
      <annotation>
        <documentation>
          Indicates which media engine the server should use to render the Output.
        </documentation>
      </annotation>
    </attribute>
    <attribute name="voice" type="string" use="optional">
      <annotation>
        <documentation>
          The voice with which to speak the requested document.
        </documentation>
      </annotation>
    </attribute>

    <sequence>
      <element name="document" type="tns:documentType" minOccurs="1" maxOccurs="unbounded">
        <annotation>
          <documentation>
            Provides the document for rendering.
          </documentation>
        </annotation>
      </element>
    </sequence>
  </complexType>
</element>
```

```
<!-- Pause command -->
<element name="pause" type="core:empty">
  <annotation>
    <documentation>
      Instructs the server to cease rendering output at the current marker and permit resumption from the same point.
    </documentation>
  </annotation>
</element>

<!-- Resume command -->
<element name="resume" type="core:empty">
  <annotation>
    <documentation>
      Instructs the server to continue rendering the output from the last pause marker.
    </documentation>
  </annotation>
</element>

<!-- Speed up command -->
<element name="speed-up" type="core:empty">
  <annotation>
    <documentation>
      Instructs the server to increase the rate of output by a unit amount.
    </documentation>
  </annotation>
</element>

<!-- Speed down command -->
<element name="speed-down" type="core:empty">
  <annotation>
    <documentation>
      Instructs the server to decrease the rate of output by a unit amount.
    </documentation>
  </annotation>
</element>

<!-- Volume up command -->
<element name="volume-up" type="core:empty">
  <annotation>
    <documentation>
      Instructs the server to increase the volume of output by a unit amount.
    </documentation>
  </annotation>
</element>

<!-- Volume down command -->
<element name="volume-down" type="core:empty">
  <annotation>
    <documentation>
      Instructs the server to decrease the volume of output by a unit amount.
    </documentation>
  </annotation>
</element>

<!-- Seek command -->
<element name="seek">
  <annotation>
    <documentation>
      Instructs the server to move the play marker of the output forward or back in time before resuming output.
    </documentation>
  </annotation>
  <complexType>
    <attribute name="direction" type="token" use="required">
      <annotation>
        <documentation>
          Indicates the direction in time in which to move the play marker.
        </documentation>
      </annotation>
      <simpleType>
        <restriction base="token">
          <enumeration value="forward"/>
          <enumeration value="back"/>
        </restriction>
      </simpleType>
    </attribute>
    <attribute name="amount" use="required">
      <annotation>
        <documentation>
          Indicates the duration by which to move the play marker.
        </documentation>
      </annotation>
      <simpleType>
        <restriction base="core:durationType">
          <minInclusive value="0"/>
        </restriction>
      </simpleType>
    </attribute>
```

```
    </complexType>
  </element>

</schema>
```

15.5 Rayo Output Complete

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xmpp:rayo:output:complete:1"
  xmlns:tns="urn:xmpp:rayo:output:complete:1"
  elementFormDefault="qualified"
  xmlns:core="urn:xmpp:rayo:1">

  <annotation>
    <documentation>
      The protocol documented by this schema is defined at http://rayo.org/xep
    </documentation>
  </annotation>

  <!-- Finish reason -->
  <element name="finish" type="core:empty">
    <annotation>
      <documentation>
        Indicates that the output component came to an end as a result of reaching the end of the document to be rendere
      </documentation>
    </annotation>
  </element>

  <!-- MaxTime reason -->
  <element name="max-time" type="core:empty">
    <annotation>
      <documentation>
        Indicates that the output component came to an end due to the maximum time limit being reached.
      </documentation>
    </annotation>
  </element>

</schema>
```

15.6 Rayo Input

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xmpp:rayo:input:1"
  xmlns:tns="urn:xmpp:rayo:input:1"
  elementFormDefault="qualified"
  xmlns:core="urn:xmpp:rayo:1">

  <annotation>
    <documentation>
      The protocol documented by this schema is defined at http://rayo.org/xep
    </documentation>
  </annotation>

  <complexType name="grammarType">
    <simpleContent>
      <attribute name="url" type="anyURI" use="optional">
        <annotation>
          <documentation>
            Provides a URI at which the grammar document is available. Must not be provided if the content-type attribute
          </documentation>
        </annotation>
      </attribute>
      <attribute name="content-type" type="string" use="optional">
        <annotation>
          <documentation>
            Indicates the content type of the grammar document provided as CDATA. Must not be set if the url attribute is
          </documentation>
        </annotation>
      </attribute>

      <restriction base="CDATA" />
    </simpleContent>
  </complexType>

  <!-- Main Input command -->
  <element name="input">
    <annotation>
      <documentation>
        Instructs the server to begin an input detector of the specified mode, with certain attributes, governed by the
      </documentation>
    </annotation>
    <complexType>
      <simpleContent>
```

```
<attribute name="mode" use="optional" default="dtmf">
  <annotation>
    <documentation>
      The method by which to collect input.
    </documentation>
  </annotation>
  <simpleType>
    <restriction base="token">
      <enumeration value="any" />
      <enumeration value="voice" />
      <enumeration value="dtmf" />
    </restriction>
  </simpleType>
</attribute>
<attribute name="terminator" type="token" use="optional" default="">
  <annotation>
    <documentation>
      Indicates a terminator token which, when encountered, should cause the input detection to cease.
    </documentation>
  </annotation>
</attribute>
<attribute name="recognizer" type="token" use="optional" default="">
  <annotation>
    <documentation>
      Indicates the name of the particular input processor to be engaged, used only for routing purposes (eg to c
    </documentation>
  </annotation>
</attribute>
<attribute name="language" type="token" use="optional" default="en-US">
  <annotation>
    <documentation>
      Specifies the recognition language to the recognizer.
    </documentation>
  </annotation>
</attribute>
<attribute name="initial-timeout" type="core:timeoutType" use="optional" default="-1">
  <annotation>
    <documentation>
      Indicates the amount of time preceding input which may expire before a timeout is triggered.
    </documentation>
  </annotation>
</attribute>
<attribute name="inter-digit-timeout" type="core:timeoutType" use="optional" default="-1">
  <annotation>
    <documentation>
      Indicates (in the case of DTMF input) the amount of time between input digits which may expire before a tir
    </documentation>
  </annotation>
</attribute>
<attribute name="sensitivity" type="core:fractionDecimalType" use="optional" default="0.5">
  <annotation>
    <documentation>
      Indicates how sensitive the interpreter should be to loud versus quiet input. Higher values represent great
    </documentation>
  </annotation>
</attribute>
<attribute name="min-confidence" type="core:fractionDecimalType" use="optional" default="0">
  <annotation>
    <documentation>
      Indicates the confidence threshold, below which a match is to be considered unreliable.
    </documentation>
  </annotation>
</attribute>
<attribute name="max-silence" type="core:timeoutType" use="optional" default="-1">
  <annotation>
    <documentation>
      Indicates the maximum period of silence which may be encountered during input gathering before a timeout is
    </documentation>
  </annotation>
</attribute>
<attribute name="match-content-type" type="token" use="optional" default="application/nlsml+xml">
  <annotation>
    <documentation>
      Indicates the required response document format.
    </documentation>
  </annotation>
</attribute>

<sequence>
  <element name="grammar" type="tns:grammarType" minOccurs="1" maxOccurs="unbounded">
    <annotation>
      <documentation>
        Provides the grammar document by which the input detection should be governed.
      </documentation>
    </annotation>
  </element>
</sequence>
```

```
      </simpleContent>
    </complexType>
  </element>

</schema>
```

15.7 Rayo Input Complete

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xmpp:rayo:input:complete:1"
  xmlns:tns="urn:xmpp:rayo:input:complete:1"
  elementFormDefault="qualified"
  xmlns:core="urn:xmpp:rayo:1">

  <annotation>
    <documentation>
      The protocol documented by this schema is defined at http://rayo.org/xep
    </documentation>
  </annotation>

  <!-- Finish reason -->
  <element name="match">
    <annotation>
      <documentation>
        Indicates that the component came to an end due to one of its grammars matching the received input. Provides the
      </documentation>
    </annotation>

    <complexType>
      <simpleContent>
        <attribute name="content-type" type="token" use="required" default="application/nlsml+xml">
          <annotation>
            <documentation>
              Indicates the content type of the result document provided as CDATA.
            </documentation>
          </annotation>
        </attribute>
      </simpleContent>
    </complexType>
  </element>

  <!-- Initial timeout reason -->
  <element name="noinput" type="core:empty">
    <annotation>
      <documentation>
        Indicates that the component came to an end because a timeout was triggered before input was received.
      </documentation>
    </annotation>
  </element>

  <!-- NoMatch reason -->
  <element name="nomatch" type="core:empty">
    <annotation>
      <documentation>
        Indicates that the component came to an end because input was received which did not match any of the specified g
      </documentation>
    </annotation>
  </element>

</schema>
```

15.8 Rayo Prompt

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xmpp:rayo:prompt:1"
  xmlns:tns="urn:xmpp:rayo:prompt:1"
  elementFormDefault="qualified"
  xmlns:core="urn:xmpp:rayo:1"
  xmlns:output="urn:xmpp:rayo:output:1"
  xmlns:input="urn:xmpp:rayo:input:1">

  <annotation>
    <documentation>
      The protocol documented by this schema is defined at http://rayo.org/xep
    </documentation>
  </annotation>

  <!-- Main Prompt command -->
  <element name="prompt">
    <annotation>
      <documentation>
        Instructs the server to begin an input detector of the specified mode, with certain attributes, governed by the r
      </documentation>
```

```
    </annotation>
    <complexType>
      <simpleContent>
        <attribute name="barge-in" type="boolean" use="optional" default="true">
          <annotation>
            <documentation>
              Whether or not the input detector is permitted to interrupt the output.
            </documentation>
          </annotation>
        </attribute>

        <sequence>
          <element name="output" type="output:outputType" minOccurs="1" maxOccurs="1">
            <annotation>
              <documentation>
                Provides the output component to be executed
              </documentation>
            </annotation>
          </element>
          <element name="input" type="input:inputType" minOccurs="1" maxOccurs="1">
            <annotation>
              <documentation>
                Provides the input component to be executed
              </documentation>
            </annotation>
          </element>
        </sequence>
      </simpleContent>
    </complexType>
  </element>

</schema>
```

15.9 Rayo Record

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xmpp:rayo:record:1"
  xmlns:tns="urn:xmpp:rayo:record:1"
  elementFormDefault="qualified"
  xmlns:core="urn:xmpp:rayo:1">

  <annotation>
    <documentation>
      The protocol documented by this schema is defined at http://rayo.org/xep
    </documentation>
  </annotation>

  <!-- Main Record command -->
  <element name="record">
    <annotation>
      <documentation>
        Instructs the server to begin recording input to the call to a file.
      </documentation>
    </annotation>
    <complexType>
      <attribute name="format" type="token" use="optional" default="mp3">
        <annotation>
          <documentation>
            File format used during recording.
          </documentation>
        </annotation>
      </attribute>
      <attribute name="start-beep" type="boolean" use="optional" default="false">
        <annotation>
          <documentation>
            Indicates whether subsequent record will be preceded with a beep.
          </documentation>
        </annotation>
      </attribute>
      <attribute name="stop-beep" type="boolean" use="optional" default="false">
        <annotation>
          <documentation>
            Indicates whether subsequent record stop will be preceded with a beep.
          </documentation>
        </annotation>
      </attribute>
      <attribute name="start-paused" type="boolean" use="optional" default="false">
        <annotation>
          <documentation>
            Whether subsequent record will start in PAUSE mode.
          </documentation>
        </annotation>
      </attribute>
      <attribute name="max-duration" type="core:timeoutType" use="optional" default="-1">
        <annotation>
```

```
        <documentation>
          Indicates the maximum duration for the recording.
        </documentation>
      </annotation>
    </attribute>
    <attribute name="initial-timeout" type="core:timeoutType" use="optional" default="-1">
      <annotation>
        <documentation>
          Controls how long the recognizer should wait after the end of the prompt for the caller to speak before send:
        </documentation>
      </annotation>
    </attribute>
    <attribute name="final-timeout" type="core:timeoutType" use="optional" default="-1">
      <annotation>
        <documentation>
          Controls the length of a period of silence after callers have spoken to conclude they finished.
        </documentation>
      </annotation>
    </attribute>
    <attribute name="direction" use="optional" default="duplex">
      <annotation>
        <documentation>
          Indicates the direction of the call to record, as in media produced or received by the calling party.
        </documentation>
      </annotation>
      <simpleType>
        <restriction base="token">
          <enumeration value="duplex">
            <annotation>
              <documentation>
                Records both sent and received audio.
              </documentation>
            </annotation>
          </enumeration>
          <enumeration value="send">
            <annotation>
              <documentation>
                Indicates that only the audio sent from the caller is to be recorded. Not supported when Record is exec
              </documentation>
            </annotation>
          </enumeration>
          <enumeration value="recv">
            <annotation>
              <documentation>
                Indicates that only and all audio received by the caller is recorded.
              </documentation>
            </annotation>
          </enumeration>
        </restriction>
      </simpleType>
    </attribute>
    <attribute name="mix" type="boolean" use="optional" default="false">
      <annotation>
        <documentation>
          Whether all channels (call legs) should be mixed into a single recording channel.
        </documentation>
      </annotation>
    </attribute>

    <sequence>
      <element name="hint" minOccurs="0" maxOccurs="unbounded">
        <annotation>
          <documentation>
            Optional format-specific encoding hints
          </documentation>
        </annotation>
        <complexType>
          <attribute name="name" type="string" use="required">
            <annotation>
              <documentation>
                The name of the hint value as expected by the recorder.
              </documentation>
            </annotation>
          </attribute>
          <attribute name="value" type="string" use="required">
            <annotation>
              <documentation>
                The value of the hint provided.
              </documentation>
            </annotation>
          </attribute>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
```

```
  <!-- Pause command -->
  <element name="pause" type="core:empty">
    <annotation>
      <documentation>
        Instructs the server to cease recording input but to leave the destination open for appending to permit resumptio
      </documentation>
    </annotation>
  </element>

  <!-- Resume command -->
  <element name="resume" type="core:empty">
    <annotation>
      <documentation>
        Instructs the server to continue recording input, appending to the same destination.
      </documentation>
    </annotation>
  </element>

</schema>
```

15.10 Rayo Record Complete

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xmpp:rayo:record:complete:1"
  xmlns:tns="urn:xmpp:rayo:record:complete:1"
  elementFormDefault="qualified"
  xmlns:core="urn:xmpp:rayo:1">

  <annotation>
    <documentation>
      The protocol documented by this schema is defined at http://rayo.org/xep
    </documentation>
  </annotation>

  <!-- Recording data -->
  <element name="recording" type="core:empty">
    <attribute name="uri" type="anyURI" use="required">
      <annotation>
        <documentation>
          Indicates the URI at which the recording is made available.
        </documentation>
      </annotation>
    </attribute>
    <attribute name="duration" type="core:durationType" use="required">
      <annotation>
        <documentation>
          Indicates the duration of the completed recording.
        </documentation>
      </annotation>
    </attribute>
    <attribute name="size" type="long" use="required">
      <annotation>
        <documentation>
          Indicates the filesize (in bytes) of the completed recording.
        </documentation>
      </annotation>
    </attribute>
  </complexType>

  <!-- Max Duration reason -->
  <element name="max-duration" type="core:empty">
    <annotation>
      <documentation>
        Indicates that the component came to an end due to the max duration being reached.
      </documentation>
    </annotation>
  </element>

  <!-- Initial Timeout reason -->
  <element name="initial-timeout" type="core:empty">
    <annotation>
      <documentation>
        Indicates that the component came to an end due to no input being detected before the initial-timeout.
      </documentation>
    </annotation>
  </element>

  <!-- Final Timeout reason -->
  <element name="final-timeout" type="core:empty">
    <annotation>
      <documentation>
        Indicates that the component came to an end because no input had been detected for the final timeout duration.
      </documentation>
    </annotation>
  </element>
```

```
</schema>
```

16. History

Prior to the development of the Rayo protocol, the most widely-used 3PCC protocols were Asterisk's AGI and AMI. Unfortunately, these protocols have many drawbacks, not least in their inconsistency and relatively poor documentation, but also in that they are poorly secured and lacking in attributes required for significant scalability. Much 3PCC activity is also done using process-internal APIs rather than a wire protocol like XMPP.

Rayo was developed to satisfy three main desires:

- To separate the application logic from the back-end call processing infrastructure for large-scale scripting-based hosted voice application platforms. This helps to ensure that the performance of the back-end infrastructure cannot be impacted by the applications controling it, and specifically to allow sandboxing the applications.
- To create a platform-agnostic protocol for the control of live media sessions that has been designed from the start for such use.
- To enable authenticated, federated, web-scale 3PCC on platforms with APIs only suitable for trusted internal use (Asterisk, FreeSWITCH, etc).

Initial development of the Rayo specification and its reference implementation was sponsored by Voxeo Labs and Telefónica, with Punchblock being the first client library implementation, as part of the Adhearsion project. Since the beginning of the development process, further implementation efforts have begun on top of Asterisk's AGI/AMI protocols and FreeSWITCH EventSocket, native to FreeSWITCH (mod_rayo), as well as client library implementations in Node/CoffeeScript and Python.

17. Acknowledgements

The authors would like to acknowledge the input of teams at Voxeo Labs, Mojo Lingo and Telefónica in the development of the initial specification, and Grasshopper in expanding the implementation landscape.

Specific individuals who have contributed to the specification or to software significant to its completion include:

- Ben Langfeld
- Ben Klang
- Chris Rienzo
- Jason Goecke
- John Dyer
- Jose de Castro
- Juan de Bravo
- Luca Pradovera
- Martín Pérez

Appendices

Appendix A: Document Information

Series: XEP
Number: 0327
Publisher: XMPP Standards Foundation
Status: Experimental
Type: Standards Track
Version: 0.2
Last Updated: 2013-06-10
Approving Body: XMPP Council
Dependencies: XMPP Core
Supersedes: None
Superseded By: None
Short Name: NOT_YET_ASSIGNED
Source Control: HTML
This document in other formats: XML PDF

Appendix B: Author Information

Ben Langfeld

Email: ben@langfeld.me
JabberID: ben@langfeld.me
URI: http://langfeld.me

Jose de Castro

Email: jdecastro@voxeo.com
JabberID: jdecastro@voxeo.com
URI: http://voxeolabs.com

Appendix C: Legal Notices

Copyright

This XMPP Extension Protocol is copyright © 1999 - 2013 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification 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 Specification.

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

Disclaimer of Warranty

NOTE WELL: This Specification 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.

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 the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (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 the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

IPR Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF's Intellectual Property Rights Policy (a copy of which can be found at <http://xmpp.org/about-xmpp/xsf/xsf-ipr-policy/> or obtained by writing to XMPP Standards Foundation, 1899 Wynkoop Street, Suite 600, Denver, CO 80202 USA).

Appendix D: Relation to XMPP

The Extensible Messaging and Presence Protocol (XMPP) is defined in the XMPP Core (RFC 6120) and XMPP IM (RFC 6121) 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.

Appendix E: Discussion Venue

The primary venue for discussion of XMPP Extension Protocols is the <standards@xmpp.org> discussion list.

Discussion on other xmpp.org discussion lists might also be appropriate; see <http://xmpp.org/about/discuss.shtml> for a complete list.

Errata can be sent to <editor@xmpp.org>.

Appendix F: Requirements Conformance

The following requirements keywords as used in this document are to be interpreted as described in RFC 2119: "MUST", "SHALL", "REQUIRED"; "MUST NOT", "SHALL NOT"; "SHOULD", "RECOMMENDED"; "SHOULD NOT", "NOT RECOMMENDED"; "MAY", "OPTIONAL".

Appendix G: Notes

1. XEP-0030: Service Discovery <http://xmpp.org/extensions/xep-0030.html>.

2. XEP-0115: Entity Capabilities <http://xmpp.org/extensions/xep-0115.html>.

3. The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

4. The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <http://xmpp.org/registrar/>.

5. XEP-0053: XMPP Registrar Function <http://xmpp.org/extensions/xep-0053.html>.

Appendix H: Revision History

Note: Older versions of this specification might be available at http://xmpp.org/extensions/attic/

Version 0.2 (2013-06-10)

Updates to address open issues.

(psa)

Version 0.1 (2013-05-06)

Initial published version approved by the XMPP Council.

(psa)

Version 0.0.1 (2013-04-05)

First draft.

(bl)

END