Avaya Client SDK External Application API

Notice

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer

"Documentation" means information published in varying mediums which may include product information, operating instructions and performance specifications that are generally made available to users of products. Documentation does not include marketing materials. Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of Documentation unless such modifications, additions, or deletions were performed by or on the express behalf of Avaya. End User agrees to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer

Avaya is not responsible for the contents or reliability of any linked websites referenced within this site or Documentation provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty

Avaya provides a limited warranty on Avaya hardware and software. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product while under warranty is available to Avaya customers and other parties through the Avaya Support website: https://support.avaya.com/helpcenter/ getGenericDetails?detailId=C20091120112456651010 under the link "Warranty & Product Lifecycle" or such successor site as designated by Avaya. Please note that if You acquired the product(s) from an authorized Avaya Channel Partner

outside of the United States and Canada, the warranty is provided to You by said Avaya Channel Partner and not by Avaya.

"Hosted Service" means an Avaya hosted service subscription that You acquire from either Avaya or an authorized Avaya Channel Partner (as applicable) and which is described further in Hosted SAS or other service description documentation regarding the applicable hosted service. If You purchase a Hosted Service subscription, the foregoing limited warranty may not apply but You may be entitled to support services in connection with the Hosted Service as described further in your service description documents for the applicable Hosted Service. Contact Avaya or Avaya Channel Partner (as applicable) for more information.

Hosted Service

THE FOLLOWING APPLIES ONLY IF YOU PURCHASE AN AVAYA HOSTED SERVICE SUBSCRIPTION FROM AVAYA OR AN AVAYA CHANNEL PARTNER (AS APPLICABLE), THE TERMS OF USE FOR HOSTED SERVICES ARE AVAILABLE ON THE AVAYA WEBSITE, HTTPS://SUPPORT.AVAYA.COM/LICENSEINFO UNDER THE LINK "Avaya Terms of Use for Hosted Services" OR SUCH SUCCESSOR SITE AS DESIGNATED BY AVAYA, AND ARE APPLICABLE TO ANYONE WHO ACCESSES OR USES THE HOSTED SERVICE. BY ACCESSING OR USING THE HOSTED SERVICE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE DOING SO (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER"), AGREE TO THE TERMS OF USE. IF YOU ARE ACCEPTING THE TERMS OF USE ON BEHALF A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT THAT YOU HAVE THE AUTHORITY TO BIND SUCH ENTITY TO THESE TERMS OF USE. IF YOU DO NOT HAVE SUCH AUTHORITY, OR

IF YOU DO NOT WISH TO ACCEPT THESE TERMS OF USE, YOU MUST NOT ACCESS OR USE THE HOSTED SERVICE OR AUTHORIZE ANYONE TO ACCESS OR USE THE HOSTED SERVICE. Licenses THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, HTTPS://SUPPORT.AVAYA.COM/LICENSEINFO, UNDER THE LINK "AVAYA SOFTWARE LICENSE TERMS (Avaya Products)" OR SUCH SUCCESSOR SITE AS DESIGNATED BY AVAYA, ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AVAYA CHANNEL PARTNER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AVAYA CHANNEL PARTNER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA CHANNEL PARTNER; AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER"), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING

CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE ("AVAYA").

Avaya grants You a license within the scope of the license types described below, with the exception of Heritage Nortel Software, for which the scope of the license is detailed below. Where the order documentation does not expressly identify a license type, the applicable license will be a Designated System License. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the documentation or other materials available to You. "Software" means computer programs in object code, provided by Avaya or an Avaya Channel Partner, whether as stand-alone products, pre-installed on hardware products, and any upgrades, updates, patches, bug fixes, or modified versions thereto. "Designated Processor" means a single stand-alone computing device. "Server" means a Designated Processor that hosts a software application to be accessed by multiple users. "Instance" means a single copy of the Software executing at a particular time: (i) on one physical machine; or (ii) on one deployed software virtual machine ("VM") or similar deployment.

License types

Designated System(s) License (DS). End User may install and use each copy or an Instance of the Software only on a number of Designated Processors up to the number indicated in the order. Avaya may require the Designated Processor(s) to be identified in the order by type, serial number, feature key, Instance, location or other specific designation, or to be provided by End User to Avaya through electronic means established by Avaya specifically for this purpose.

Concurrent User License (CU). End User may install and use the Software on multiple Designated Processors or one or more Servers, so long as only the licensed number of Units are accessing and using the Software at any given time. A "Unit" means the unit on which Avaya, at its sole discretion, bases the pricing of its licenses and can be, without limitation, an agent, port or user, an e-mail or voice mail account in the name of a person or corporate function (e.g., webmaster or helpdesk), or a directory entry in the administrative database utilized by the Software that permits one user to interface with the Software. Units may be linked to a specific, identified Server or an Instance of the Software.

Database License (DL). End User may install and use each copy or an Instance of the Software on one Server or on multiple Servers provided that each of the Servers on which the Software is installed communicates with no more than one Instance of the same database.

CPU License (CP). End User may install and use each copy or Instance of the Software on a number of Servers up to the number

---

---

indicated in the order provided that the performance capacity of the Server(s) does not exceed the performance capacity specified for the Software. End User may not re-install or operate the

Software on Server(s) with a larger performance capacity without Avaya's prior consent and payment of an upgrade fee.

Named User License (NU). You may: (i) install and use each copy or Instance of the Software on a single Designated Processor or Server per authorized Named User (defined below); or (ii) install and use each copy or Instance of the Software on a Server so long as only authorized Named Users access and use the Software. "Named User", means a user or device that has been expressly authorized by Avaya to access and use the Software. At Avaya's sole discretion, a "Named User" may be, without limitation, designated by name, corporate function (e.g., webmaster or helpdesk), an e-mail or voice mail account in the name of a person or corporate function, or a directory entry in the administrative database utilized by the Software that permits one user to interface with the Software.

Shrinkwrap License (SR). You may install and use the Software in accordance with the terms and conditions of the applicable license agreements, such as "shrinkwrap" or "clickthrough" license accompanying or applicable to the Software ("Shrinkwrap License").

Heritage Nortel Software

"Heritage Nortel Software" means the software that was acquired by Avaya as part of its purchase of the Nortel Enterprise Solutions Business in December 2009. The Heritage Nortel Software is the software contained within the list of Heritage Nortel Products located at https://support.avaya.com/LicenseInfo under the link "Heritage Nortel Products" or such successor site as designated by Avaya. For Heritage Nortel Software, Avaya grants Customer a license to use Heritage Nortel Software provided hereunder solely to the extent of the authorized activation or authorized usage level, solely for the purpose specified in the Documentation, and solely as embedded in, for execution on, or for communication with Avaya equipment. Charges for Heritage Nortel Software may be based on extent of activation or use authorized as specified in an order or invoice.

Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation, Software, Hosted Service, or hardware provided by Avaya. All content on this site, the documentation, Hosted Service, and the product provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software unless expressly authorized by Avaya. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil offense under the applicable law.

Virtualization

The following applies if the product is deployed on a virtual machine. Each product has its own ordering code and license types. Note that each Instance of a product must be separately licensed and ordered. For example, if the end user customer or Avaya Channel Partner would like to install two Instances of the same type of products, then two products of that type must be ordered.

Third Party Components

"Third Party Components" mean certain software programs or portions thereof included in the Software or Hosted Service may contain software (including open source software) distributed under third party agreements ("Third Party Components"), which contain terms regarding the rights to use certain portions of the Software ("Third Party Terms"). As required, information regarding distributed Linux OS source code (for those products that have distributed Linux OS source code) and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply is available in the products, Documentation or on Avaya's website at: https:// support.avaya.com/Copyright or such successor site as designated by Avaya. The open source software license terms provided as Third Party Terms are consistent with the license rights granted in these Software License Terms, and may contain additional rights benefiting You, such as modification and distribution of the open source

software. The Third Party Terms shall take precedence over these Software License Terms, solely with respect to the applicable Third Party Components to the extent that these Software License Terms impose greater restrictions on You than the applicable Third Party Terms.

The following applies only if the H.264 (AVC) codec is distributed with the product. THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP://WWW.MPEGLA.COM.

Service Provider

THE FOLLOWING APPLIES TO AVAYA CHANNEL PARTNER'S HOSTING OF AVAYA PRODUCTS OR SERVICES. THE PRODUCT OR HOSTED SERVICE MAY USE THIRD PARTY COMPONENTS SUBJECT TO THIRD PARTY TERMS AND REQUIRE A SERVICE PROVIDER TO BE INDEPENDENTLY LICENSED DIRECTLY FROM THE THIRD PARTY SUPPLIER. AN AVAYA CHANNEL PARTNER'S HOSTING OF AVAYA PRODUCTS MUST BE AUTHORIZED IN WRITING BY AVAYA AND IF THOSE HOSTED PRODUCTS USE OR EMBED CERTAIN THIRD PARTY SOFTWARE, INCLUDING BUT NOT LIMITED TO MICROSOFT SOFTWARE OR CODECS, THE AVAYA CHANNEL PARTNER IS REQUIRED TO INDEPENDENTLY OBTAIN ANY APPLICABLE LICENSE AGREEMENTS, AT THE

AVAYA CHANNEL PARTNER'S EXPENSE, DIRECTLY FROM THE APPLICABLE THIRD PARTY SUPPLIER.

WITH RESPECT TO CODECS, IF THE AVAYA CHANNEL PARTNER IS HOSTING ANY PRODUCTS THAT USE OR EMBED THE G.729 CODEC, H.264 CODEC, OR H.265 CODEC, THE AVAYA CHANNEL PARTNER ACKNOWLEDGES AND AGREES THE AVAYA CHANNEL PARTNER IS RESPONSIBLE FOR ANY AND ALL RELATED FEES AND/OR ROYALTIES. THE G.729 CODEC IS LICENSED BY SIPRO LAB TELECOM INC. SEE WWW.SIPRO.COM/CONTACT.HTML. THE H.264 (AVC) CODEC IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO: (I) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (II) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION FOR H.264 (AVC) AND H.265 (HEVC) CODECS MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP:// WWW.MPEGLA.COM.

Compliance with Laws

You acknowledge and agree that it is Your responsibility for complying with any applicable laws and regulations, including, but not limited to laws and regulations related to call recording, data privacy, intellectual property, trade secret, fraud, and music performance rights, in the country or territory where the Avaya product is used.

Preventing Toll Fraud

"Toll Fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of Toll Fraud associated with your system and that, if Toll Fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Toll Fraud intervention

If You suspect that You are being victimized by Toll Fraud and You need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see

---

---

the Avaya Support website: https://support.avaya.com or such successor site as designated by Avaya.

Security Vulnerabilities

Information about Avaya's security support policies can be found in the Security Policies and Support section of https:// support.avaya.com/security.

Suspected Avaya product security vulnerabilities are handled per the Avaya Product Security Support Flow (https:// support.avaya.com/css/P8/documents/100161515).

Downloading Documentation

For the most current versions of Documentation, see the Avaya Support website: https://support.avaya.com, or such successor site as designated by Avaya.

Contact Avaya Support

See the Avaya Support website: https://support.avaya.com for product or Hosted Service notices and articles, or to report a problem with your Avaya product or Hosted Service. For a list of support telephone numbers and contact addresses, go to the Avaya Support website: https://support.avaya.com (or such successor site as designated by Avaya), scroll to the bottom of the page, and select Contact Avaya Support.

Trademarks

The trademarks, logos and service marks ("Marks") displayed in this site, the Documentation, Hosted Service(s), and product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, its licensors, its suppliers, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the Documentation, Hosted Service(s) and product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party.

Avaya is a registered trademark of Avaya Inc.

All non-Avaya trademarks are the property of their respective owners. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Table of Contents

- Document Purpose
  - Document Version
- What's New
- Overview

    - Application Integration
    - Call Control Capability Table
    - Why JSON?
    - Named Pipe API Model
        - Pipe Discovery
        - Application Sandboxing for MacOS
        - Basic Interworking Model
        - Enhanced Interworking Model
    - Security Model
        - Authentication
        - Networking
        - Named Pipe Connectivity Model
        - Protocol Framing
        - Denial of Service
        - Rate Limiting
        - Application Identification
        - Disabling the External Application Interface
        - Internationalization
    - Backward and Forward Compatibility
        - Versions
        - External Application API Backward Compatibility
        - External Application API Forward Compatibility
        - Current Version
- API Definition
    - Media Types
    - Pipe Management
    - Pipe Management Events
    - Calls
    - Call Events
    - Media Devices
- Call Flows
        - Legend
    - API Control Call Flows
        - Client SDK Initialization
        - Register
        - Unregister
        - DisconnectRequest
    - Call Handling
        - Make Call - Named Pipe
        - Make Call - Named Pipe
        - Answer Call
        - Mute and Unmute call
        - Hold and Retrieve call
        - Terminate or End Call
    - Media Device Listing
        - GetActiveAudioDevices

    - Application Interworking
        - Call Created by UC Application
        - Call Created by Peer External Application
        - Call Hold by UC Application
        - Call Hold by Peer External Application
- References
- Appendix
    - Windows C# Named Pipe Sample Code

Document Purpose

The purpose of this document is to define the Unified Communications Application (UCA) portfolio requirements for external applications to interact with the Avaya Client SDK communication package and with applications that utilize this package, such as Avaya Workplace.

This document describes the resources that make up the official *External Application Interface for the Client SDK Communication Services Package* v1.2 (*API*).

Document Version

What's New

External Application APIs were up versioned to v1.2, with the primary difference being that calls with media routed remotely, such as shared control and telecommuter calls, are clearly identifiable within the API as calls with remote media. Applications written with to the v1.0 of the External Application API will continue to work, and be supported.

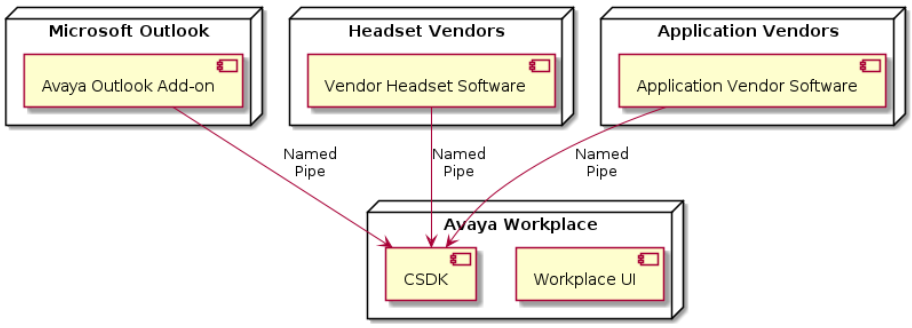
Overview

The purpose of the APIs is to allow general applications executing locally on the workstation to send primitive call control requests to Avaya Client SDK. The External API was defined as a consolidated API to enable application vendors authoring applications for multiple platforms to leverage a consistent API. This API is supported on Windows and Mac OS X platforms, and is not applicable to Android or iOS platforms.

The API enables applications to create and control calls, and to discover calls through JSON messaging over a named pipe. The API is versioned, and is versioned independently of the Avaya Client SDK version to provide external applications with a stable set of functionality.

Avaya Workplace is the lead Avaya application that is built on the Avaya Client SDK, and is often referred or used in examples throughout the document. The External Application API may be enabled by any application that is built on the Avaya Client SDK. It is the choice of the application to determine if the External Application API is enabled.

Within this document, the Avaya Client SDK will be referenced as either the Client SDK or CSDK.



Application Integration

When developing an application that uses the External API, Avaya recommends that partners integrate their applications with the Avaya Workplace (previously known as Avaya Equinox) application first, treating Avaya Workplace as the reference implementation. The External API will be available in other Client SDK applications written by Avaya or third party organizations. Once you have successfully integrated with Workplace, you can begin integrating with other Client SDK applications.

Call Control Capability Table

Avaya Workplace is the latest Avaya Unified Communications client, and it extends the capabilities of Avaya Communicator and Avaya one-X® Communicator available for External Applications. The following table shows the capabilities available to External Applications, and if the capability is newly introduced with the External Application Interface.

Call Control Capability	**Avaya Communicator 2.0 Interface**	**Avaya Equinox (3.0 - 3.6)**	**Avaya Workplace 3.7+**	**Notes**
----------	----------	----------	----------	----------
Accept Incoming Call	Y	Y	Y	Existing capability
Add/Remove video	N	Y	Y	capability introduced in Equinox
Block Camera	N	Y	Y	capability introduced in Equinox
Calling line ID	N	Y	Y	capability introduced in Equinox
Create call	Y	Y	Y	Existing capability • Not supported for HTTPUA calls.
Hold/Retrieve	Y	Y	Y	Existing capability
Ignore Incoming Call	N	Y	Y	Existing capability
Terminate Call	Y	Y	Y	Existing capability
Insert DTMF	**N**	**Y**	**Y**	**capability introduced in Equinox**
Mute/Unmute Call	Y	Y	Y	Existing capability
Media Device Listings	N	Y	Y	capability introduced in Equinox

Why JSON?

The External API uses JSON to exchange data between the External Application and the External Application Interface API. JSON offers the following advantages:

- JSON is simple, open, and interoperable.
- Data is defined to allow generic tools to manipulate data.
- JSON data is (almost) human readable.

A key aspect of JSON leveraged through the API is that applications consuming the JSON can easily disregard fields that it is unfamiliar, and the External API can do the same. This allows a large degree of flexibility between versions of External Applications and the External API, as long as JSON fields are never removed from the External API. This is critical to External API versioning, which is described below.

Named Pipe API Model

The External API is an Avaya proprietary API that provides external applications with a basic interface to influence call handling of the Client SDK. The interface uses JSON encoded messages over a platform-provided named pipe, with the named pipe connecting the external application to the Client SDK. The External API provided by the Client SDK is intended to support two different types of applications, simple click to call and basic call control. Basic application interoperability allows External Applications to invoke Make Call requests on the Client SDK.

Pipe Discovery

When the API is enabled, the Client SDK establishes a single public named pipe. The named pipe is strongly named, to allow applications to find the appropriate pipe. This allows multiple users to use the same workstation simultaneously, and allows multiple external applications to direct requests to the intended Client SDK instance. In the case where multiple Client SDK Applications are executing as the same user, the first Client SDK Application to acquire the named pipe "wins". If the pipe name is already in use when the Client SDK attempts to create the named pipe, the operation will fail, and the External Application API will not be available for the Client SDK instance. An alternative pipe name is not available.

Windows

```
PipeName = \\.\pipe\AvayaCSDK-%username%

i.e. \\.\\pipe\AvayaCSDK-bob
```

The name of the Windows pipe can be confirmed using pipelist.exe. Look for a pipe name starting with "AvayaCSDK".

MacOS X

```
Unix Domain Socket Name = <User Home Dir>/Application/ Support.com.avaya-%app
name%/AvayaCSDK-%platform user name%
i.e. /Users/joeuser/Library/Application Support/com.avaya/Avaya-
Workplace/AvayaCSDK-bob
```

The name of the MacOS Unix Domain Socket can be confirmed using "netstat -a". Look for a socket name starting with "AvayaCSDK".

Application Sandboxing for MacOS

Not supported in this release.

Basic Interworking Model

Basic interworking occurs on the public pipe and allows external applications to invoke two operations:

- **create** - Enables external applications to request the Client SDK to create a call on behalf of the Client SDK user.
- **register**- Enables external applications to request a dedicated pipe in order to register for call UpdateEvents. It also provides a dedicated pipe for the external application to send or make advanced requests.

Each request is acknowledged with a response. The "create" response will contain the call object containing the current state of the call attributes. If the External Application has not registered for events, no subsequent updates will be provided.

Enhanced Interworking Model

When the External Application registers for call UpdateEvents, mid-call operations are permitted and UpdateEvents are sent when a call attribute changes. Each External API call operation request has an associated response, and an operation response will be returned for each request received. The operation response will always contain the current state of the call object, with subsequent call attribute changes being provided to the External Application through the UpdateEvent.

As the External API is not the only mechanism available to control a call, External Applications must be able to handle UpdateEvents for call operations invoked through another mechanism, such as the UC client user interface. The External Application can track changes in the call that are made externally (through the Avaya Client SDK application) because call attribute changes are sent through an UpdateEvent without a call operation response.

Similarly, the External API can have multiple External Applications connected simultaneously. External Applications only receive operation responses for requests invoked by the External Application that sent the request, and call UpdateEvents will be received for all calls, irrespective of how the call was created.

Security Model

Authentication

All named pipes are created by the Avaya Client SDK. This allows the Client SDK to control security permissions for the named pipe.

All pipes should be created so that only applications running as the platform user are able to connect to the pipe.

The platform (OS) is responsible for enforcing authentication, as defined by the application.

Networking

The external Application and the Avaya Client SDK processes must exist on the same workstation. The External API interface is not available to remote network applications.

Named Pipe Connectivity Model

To allow multiple external applications to send requests on the *public* pipe, the external application shall disconnect from the pipe 200ms after the request is made. If a request is not received within 200ms of connecting, the pipe will be closed by the server.

The pipe name will be consistent for each platform user across Avaya Client SDK restarts.

The pipe is created with visibility local to the workstation. Network access is not permitted.

The named pipe can only be connected by a process with the same login identity as the Client SDK process owner.

Applications that register for call events can remain connected to the named pipe indefinitely.

Protocol Framing

External API requests and events are JSON-encoded over the pipe. Each message (request or event) is terminated with a NULL byte to act as a message delimiter. As JSON messages are syntactically strong, it is possible for either side of the pipe to be aware when a complete request is received. When the Avaya Client SDK receives a complete request, it will act on the request. If the JSON request is not properly terminated within 200ms, the Client SDK will disconnect the external application from the pipe, and purge the pipe buffer. This allows the pipe to be reset for both parties.

If the external application detects a malformed response or event, it will disconnect from the pipe and reconnect.

NullByte = \0

Multiple requests can be sent over the main pipe or the private pipe without waiting for the associated response, as long as the following is true;

- Each request is properly encoded, and terminated with a NullByte.
- The transaction ID for each request is unique.

The External API parser expects complete External API requests to be written at once, not byte by byte. The parser expects the NullByte to be encoded immediately following the closing brace of the JSON message.

Transaction Identifiers (TransactionIds)

TransactionIds are used to correlate responses with the intended request. Each request shall have its own transactionId, and the transactionId should be unique across time and space for all messages sent by the external application. The transaction IDs need not to be numeric, not monotonically increasing.

I.E.

a1-47.135.10.14, a2-47.135.10.14,a23-47.135.10.14

41.25.135.158-9bca, 41.25.135.158-9bcb,41.25.135.158-9bcc

Denial of Service

The Avaya Client SDK limits the number of private named pipes to 3.

Rate Limiting

Rate limiting is not implemented.

Application Identification

An Application-ID is used for correlation between external applications and the Avaya Client SDK. Register requests directed to the Client SDK must have the Application-ID populated. Register requests without an ApplicationId will be silently discarded. Two external applications cannot register with the same ApplicationId, and the later registration will supercede the original application registration.

Disabling the External Application Interface

The External Application Interface may be disabled by Avaya Client SDK. When the External Application interface is disabled, the public pipe will not be created.

Internationalization

Unicode support

UTF-8 encoding shall be used for data exchange.

Backward and Forward Compatibility

Versions

The External Application API is versioned independently of the Avaya Client SDK version to provide external applications with a stable set of functionality. The External Application API is versioned with a major version and a minor version. A critical application requirement is to be able to safely parse expected and unexpected fields. This allows the External Application API to add incremental and supporting data to the responses and events without fear of breaking the external application. The External Application API will behave similarly. The major version is updated for the following reasons:

- When a change to the API is introduced that is not backward compatible with the previous version.
    - Example: A new parameter is introduced, or a mandatory parameter becomes optional.
- When a portion of the API is deprecated.

The minor version of the API can change for the following reasons:

- A new capability is added to the API. This is a minor version update because new functionality will not impact existing applications.
- A new optional parameter is added to an API request, response, or event.

The External Application API implements the following version strategy, which is based on JSON's extensibility concepts.

If a new optional parameter is added to a media type, the minor version uses a dot increment.

Example: v1 becomes .v1.1, and v1.1 would become 1.2. v1.0 would become 1.1.

If a new mandatory parameter is added to a media type, the major version is incremented by 1 for *all media types.*.

Example: v1 becomes v2. v9 becomes v10.

If a new capability is added to the External Application API, the new media types associated with the capability would be versioned as v1.

If the external application registers for events, the External Application API will provide Event media types based on the version of the RegisterRequest. The versions of the following media types will be versioned at the same level. If the media type for one of the following changes, the versioning for all the media types must be updated.

Application Versioning table for Register request, Register Response, and Events

Application Sends	API Responds
vnd.avaya.clientresources.call.RegisterRequest.v1	**vnd.avaya.clientresources.call.RegisterResponse.v1** **vnd.avaya.clientresources.call.RegisterResponse.v1.2**
vnd.avaya.clientresources.call.RegisterRequest.v3	**vnd.avaya.clientresources.call.RegisterResponse.v3** **vnd.avaya.clientresources.call.RegisterResponse.v3.3**

The External API will use the latest available minor version of the API based on the version number of the register request provided by the External Application.

External Application API Backward Compatibility

External Application API backward compatibility is the ability for the API to handle requests from an older version of the API used by the external application. The External Application API will support the current major version of the media types, and the previous version of the External Application API. If the current version of the External Application API is v2, the External Application API implemention will be capable of returning v1 responses. If the current version of the External Application API is v3, the External API will be capable of returning v2 responses, but will not be capable of supporting v1 responses.

If a External Application API change forces the major version of the API to be updated, all supported media types versions will be updated to the next major version number.

This will occur when the Avaya Client SDK External Application API version is newer than the version of the External Application API used by the external application.

Example: Avaya Workplace is newer than the headset application.

External Application API Forward Compatibility

External Application API forward capability is the ability for the external application to send External Application API versions newer than the External Application API supports. In this case, the External Application API will reply to the CreateCall or Register request with an Error, Unsupported Media Type. When this occurs, the external application will reduce the major version of the API by 1, and attempt the request again. This process may be repeated until the version becomes v1, or until the version of the External Application API is not supported by the

external application. It is up the external application vendor to decide how many legacy versions of the External Application API to support.

This will occur when the external application version is newer than the version of the External Application API used by the Avaya External Application API

Example: Headset Application is newer than the Avaya Workplace.

Current Version

By default, all requests receive the v1 version of the External Application API. The version of the API is explicitly captured in the messages.

```
vnd.avaya.clientservices.call.v1.2
```

Versioning will be supported at an External Application API level, and all media types will use the same version. The version of an event sent to the application will be based on the major version of register request received. The External Application API will always provide the most recent minor version of the External Application API.

API Failures will return the following Accept header in the failure response.

```
vnd.avaya.clientservices.Error.v1.2
```

API Definition

Media Types

Applications shall always be prepared to receive a JSON Error response.

Errors

vnd.avaya.clientresources.Error.v1.2

Attribute	Type	Optional	Description
displayMessage	string	N	A message that contains information that can be displayed to an end user.
errorMessage	string	N	A message that contains information necessary for a developer to correct the problem.
errorCode	Enum	N	A code associated with a unique error condition on the server.

Attribute	Type	Optional	Description
transactionId	string	Y	The request associated with the error.

Error (JSON)

{
"vnd.avaya.clientresources.Error.v1.2" : {
"displayMessage" : " displayMessage text ",
"errorMessage" : " error Message text ",
"errorCode" : " error Code text ",
"transactionId": "1"
}
} \0

Failure and Error Codes

Error codes are similar to HTTP, to facilitate ease of use and understanding.

Code		Description
400	Bad Request	Could not parse request.
404	Not Found	Call/Resource not found.
406	Not Acceptable	Missing mandatory field.
408	Timeout	Timeout processing the request.
409	Conflict	applicationid has already registered on this pipe.
410	Gone	Desired entity (call) does not exist

415	UnsupportedMediaType	Improper media type.
500	Client SDK Error	Error processing request.
503	Service Unavailable	Client SDK not ready to accept requests.
497	Media Preserved	Call is in the media preserved state. The only supported operation is Terminate.

Pipe Management

Register

vnd.avaya.clientresources.RegisterRequest.v1.2

vnd.avaya.clientresources.RegisterResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Register Request (JSON)

```
{
"vnd.avaya.clientresources.RegisterRequest.v1.2" : {
  "applicationId": "app",
  "transactionId: "1"
  }
}\0
```

Success Response (JSON)

```
{
"vnd.avaya.clientresources.RegisterResponse.v1.2" : {
  "transactionId": "1"
  }
}\0
```

Unregister

vnd.avaya.clientresources.UnregisterRequest.v1.2

vnd.avaya.clientresources.Error.v1.2

Unregister Request (JSON)

```
{
"vnd.avaya.clientresources.UnregisterRequest.v1.2" : {
  "transactionId": "1"
  }

}\0
```

An unregister response is not required. The Client SDK will immediately close the connection as implicit acknowledgement to the request.

Pipe Management Events

DisconnectRequest

vnd.avaya.clientresources.DisconnectRequest.v1

DisconnectRequest (JSON)

<pre>{
"vnd.avaya.clientresources.DisconnectRequest.v1.2" {
"transactionId": "1"
}
}\0</pre>

A Disconnect response is not required. When a DisconnectRequest is received, the recipient can immediately close the pipe as acceptance. If the recipient does not close the pipe, the DisconnectRequest sender will close the pipe after 200ms.

Calls

Call Resource

The following Call attributes are included in all call related responses and call events. Fields can be left empty or blank intentionally by the External Application API.

vnd.avaya.clientresources.Call.v1.2

Attribute	Values	Relationship	Description
CallId	string	mandatory	Unique call identifier.
remote	true,false	mandatory	Call is on another device (MDA, Bridged lines, EC 500). The call is not being handled by the CSDK Application. Media is not running locally. Headset call control applications should not act on remote calls.
remotePartyName	string	mandatory	Name of the remote participant. The value can be empty or blank.
remotePartyNumber	string	mandatory	Remote Party CLID. The value can be empty or blank.

Attribute	Values	Relationship	Description
privacy	true,false	mandatory	Privacy set for Remote Party. The value can be empty or blank.
subject	string	mandatory	Call subject. The value can be empty or blank.

callState	alerting transferred ended ending established failed far-end renegotiating Held Holding idle ignored initiating remote alerting renegotiating transferring unholding video updating	mandatory	The call state, which represents: `public enum CallState values.` alerting- Call is alerting locally (incoming call). transferred - The call is being transferred by a remote party on the call. ended - Call ended (by far-end, or end request has been responded to by the remote party). ending - Request to end the call has been sent. established - Call has been established and is active (not held). failed- Failed. far-end renegotiating- Call renegotiating (requested by far-end). held- Held. holding - Holding. idle - Uninitialized. ignored - Call is ignored. initiating - Call initiated (outgoing only). remote alerting- Call is alerting remotely (outgoing call). renegotiating - Call is renegotiating (requested by us). transferring -Transfer initiated by the local user (outgoing).

Attribute	Values	Relationship	Description
			unholding - Retrieving. video updating - Video is being added or removed from the call.
muted	true,false	mandatory	Audio on or off.
videoPossible	true,false	mandatory	Client informs the API if video escalation is possible. Video license is acquired. Network applicable for video. If a video device is not available, this value will not be altered.
videoDirection	Inactive, Send_Receive, Send_Only, Receive_Only	mandatory	The direction of the video.
audioDirection	Remote, Inactive, Send_Receive, Send_Only, Receive_Only	mandatory	The direction of the audio. If audio is directed to a remote device, the direction will be remote.
transactionId	TransactionId of the request.	optional	transactionID is only provided in response messages, and it not provided in events.

Call Operations

Call Operations Summary table

Operation	MediaType	Description
query	vnd.avaya.clientresources.call.GetCallsRequest.v1.2	Discover calls on the client.
create	vnd.avaya.clientresources.call.CreateRequest.v1.2	Initiate call.
terminate	vnd.avaya.clientresources.call.TerminateRequest.v1.2	End call. Terminates the existing call, irrespective of call state.
accept	vnd.avaya.clientresources.call.AcceptRequest.v1.2	Answer the call. Possible state for incoming alerting state.
ignore	vnd.avaya.clientresources.call.IgnoreRequest.v1.2	Ignore the call. Possible state for incoming alerting state. Invoking the ignore operation, the client's ringer will be muted, and the incoming call notification will be suppressed. The call state does not change, and the call can still be answered.
hold	vnd.avaya.clientresources.call.HoldRequest.v1.2	Hold call.
mute	vnd.avaya.clientresources.call.MuteRequest.v1.2	Mute audio. Local mute operation supported in computer mode, and shared control mode. Network mute, for conferencing applications, is not supported. Mute in other phone mode (telecommuter mode) not supported.
video	vnd.avaya.clientresources.call.VideoRequest.v1.2	Add video to the call.
dtmf	vnd.avaya.clientresources.call.DTMFRequest.v1.2	Insert DTMF digits during the call.

Call Operations and Call State Validity Table

Call State		Query	Terminate	Accept	Ignore	Hold	Mute	Video	DTMF	
1	alerting	√	√	√	√	X	√	X	X	Call is alerting locally (incoming call).
2	initiating	√	√	X	X	X	√	X	√	Call initiated (outgoing only).
3	established	√	√	X	X	√	√	√	√	Call has been established and is active (not held).
4	held	√	√	X	X	√	√	X	X	Call has been established but is not active (held).
5	holding	√	√	X	X	X	√	X	X	Call is in process of being held by us.
6	unholding	√	√	X	X	X	√	X	X	Unholding or retrieving.
7	failed	√	√	X	X	X	√	X	X	call Failed.
8	idle	√	√	X	X	X	√	√	X	Call Uninitialized.
9	remote alerting	√	√	X	X	X	√	X	√	Call is alerting remotely (outgoing call).
10	ignored	√	X	√	√	X	√	X	X	Incoming session is ignored.

11	renegotiating	√	√	X	X	X	√	X	√	Call renegotiating (requested by us).
12	far-end renegotiating	√	√	X	X	X	√	X	√	Call renegotiating (requested by them).
13	transferring	√	√	X	X	X	√	X	X	Transfer initiated by us (outgoing).
14	transferred	√	√	X	X	X	√	X	X	Being transferred by them (incoming).
15	ending	√	X	X	X	X	√	X	X	Request to end the session has been sent.
16	Ended	√	X	X	X	X	√	X	X	Call ended (by far-end, or end request has been responded to by the remote party).
17	video updating	√	√	X	X	X	√	√	√	

States reflected in the English present participle (state names that end with "ing") are ephemeral states, and the application should expect a subsequent state transition to follow. While a call is in an ephemeral state, it is not possible for the external application to invoke signalling operations.

The operation response JSON message will always contain the state of the session, which may be unrelated to the operation requested.

Call State Transitions

Outgoing Call State Transition Table

Pre-Op Call State	Operation	Post Op Call State
idle	<any non-create operation>	idle
idle	Create	initiating
initiating	Terminate	ending/ended
initiating	<any other operation>	enitiating
remote alerting	Terminate	ending/ended
remote alerting	DTMF	remote alerting
remote alerting	<any other operation>	remote alerting
----------	----------	----------

Incoming Call State Transition Table

Pre-Op Call State	Operation	Post Op Call State
alerting	No Operation Timeout	terminated, null
alerting	Ignore	ignored
alerting	Accepted	established
alerting	Terminate	alerting
ignored	Ignore	ignored
ignored	Accepted	established
ignored	Terminate	ignored
----------	----------	----------

Established Call State Transition Table

Pre-Op Call State	Operation	Post Op Call State
established	Terminate	ending/ended
established	Join (MDA, from UC Client)	established
established	DTMF	established

Pre-Op Call State	Operation	Post Op Call State
established	Hold	holding/held
transferring	Terminate	ending/ended
transferring	any operation except Terminate	transferring
transferred	Terminate	ending/ended
transferred	any operation except Terminate	transferred
renegotiating	Terminate	ending/ended
renegotiating	any operation except Terminate	renegotiating
far-end renegotiating	Terminate	ending/ended
far-end renegotiating	any operation except Terminate	far-end renegotiating
----------	----------	----------

Held Call State Transition Table

Pre-Op Call State	Operation	Post Op Call State
established	Hold	holding/held
held	Unhold	unholding/established
held	Hold	held
held/holding/unholding	Terminate	ending/ended
holding/unholding	<any other operation>	holding/unholding/held/established
----------	----------	----------

Failed/Ending/Ended Call State Transition Table

Pre-Op Call State	Operation	Post Op Call State
ending	<any operation>	ending/ended
ended	<any operation>	ended

failed	<any operation>	failed
----------	----------	----------

 Applications registering for call events when calls are in progress may not receive events to support the existing state. Call Events are only guaranteed to be sent for call state transitions.

Call Messages

 External Application API documentation below highlights the important elements of the API, given the context of the request. All call attributes will be returned for call-related requests.

Get Calls

vnd.avaya.clientresources.call.GetCallsRequest.v1.2

vnd.avaya.clientresources.call.GetCallsResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

get Calls Request (JSON)

{
 	
"**vnd.avaya.clientresources.call.GetCallsRequest.v1.2": {**
 	
"transactionId": "1"
 	
}
} \0

Success Response -with single call (JSON)

get Calls Request (JSON)

```
{
"vnd.avaya.clientresources.call.GetCallsResponse.v1.2" : {
"vnd.avaya.clientresources.Call.v1.2" : {
"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",

"callId": "xyz123",

"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",

"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"

},

"transactionId": "1"

}

}\0
```

Success Response -with calls (JSON)

get Calls Request (JSON)

{
"vnd.avaya.clientresources.call.GetCallsResponse.v1.2" : {
"vnd.avaya.clientresources.Call.v1.2" : [{

"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "xyz123",

"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"

},

{

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",

"callID": "abc124",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}] ,

"transactionId": "1"

}

}\0

get Calls Request (JSON)
Success Response - without calls (JSON)

{
"vnd.avaya.clientresources.call.GetCallsResponse.v1.2" : {
 	
"transactionId": "1"
 	
}
 	
}\0

Errors Responses (JSON)

{
 	
"vnd.avaya.clientresources.Error.v1.2" : {
 	
"displayMessage" : " displayMessage text ",
 	
"errorMessage" : " error Message text ",
 	
"errorCode" : " error Code text ",
 	
"transactionId": "1"
 	
}
 	
}\0

GetCalls will return, at most, all active calls. The External API shall not constrain the number of active calls.

Mute Call

vnd.avaya.clientresources.call.MuteRequest.v1.2

vnd.avaya.clientresources.call.MuteResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Mute Request Payload (JSON)

```
{
 "vnd.avaya.clientresources.call.MuteRequest.v1.2": {
"callId": "xxx",
"muted": "true",
"transactionId": "1"
}
}\0
```

Success Response (JSON)

```
{
 "vnd.avaya.clientresources.call.MuteResponse.v1.2": {

"vnd.avaya.clientresources.Call.v1.2" : {

"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

 "callId": "abc124",

"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}

"transactionId": "1"
}
} \0
```

Failure Response

Unmute Call

vnd.avaya.clientresources.call.MuteRequest.v1.2

vnd.avaya.clientresources.call.MuteResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Unmute Request Payload (JSON)

{
"vnd.avaya.clientresources.call.MuteRequest.v1.2": {
"callId": "xxx",
"muted": "false",
"transactionID": "1"
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.MuteResponse.v1.2": {
 	
"vnd.avaya.clientresources.Call.v1.2" : {
 	
"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
"callId": "abc124",
 	
"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",
 	
"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},
 	
"transactionId": "1",
 	
}
} \0

Failure Response

Hold Call

vnd.avaya.clientresources.call.HoldRequest.v1.2

vnd.avaya.clientresources.call.HoldResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Hold Call Request Payload (JSON)

{
" **vnd.avaya.clientresources.call.HoldRequest.v1.2**": {
"callId": "xxx",
"held": "true",
"transactionId": "1"
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.HoldResponse.v1.2": {
 	
"vnd.avaya.clientresources.Call.v1.2" : {
 	
"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
"callId": "abc124",
 	
"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",
 	
"audioDirection" : "Remote,Inactive",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},
 	
"transactionId": "1",
},
} \0

Failure Response

Retrieve Call (unHold Call)

vnd.avaya.clientresources.call.HoldRequest.v1.2

vnd.avaya.clientresources.call.HoldResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Retrieve Call Request Payload (JSON)

{
"vnd.avaya.clientresources.call.HoldRequest.v1.2": {
"callId": "xxx",
"held": "false",
"transactionId": "1"
}
} \0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.HoldResponse.v1.2": {
 	
"vnd.avaya.clientresources.Call.v1.2" : {
 	
"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
"callId": "abc124",
 	
"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",
 	
"audioDirection" : "Remote, Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},
 	
"transactionId": "1"
}
} \0

Failure Response

Terminate Call

vnd.avaya.clientresources.call.TerminateRequest.v1.2

vnd.avaya.clientresources.call.TerminateResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Terminate Call Request Payload (JSON)

{
"vnd.avaya.clientresources.call.TerminateRequest.v1.2": {
"callId": "xxx",
"transactionId": "1"
}
}\0

Success Response (JSON)

```
{
 "vnd.avaya.clientresources.call. TerminateResponse.v1.2": {

"vnd.avaya.clientresources.Call.v1.2" : {

"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

 "callId": "abc124",

"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} ,

"transactionId": "1"

}
} \0
```

Failure Response

Answer Call (Accept Call)

vnd.avaya.clientresources.call.AcceptRequest.v1.2

vnd.avaya.clientresources.call.AcceptResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Accept Call Request Payload (JSON)

{

"vnd.avaya.clientresources.call.AcceptRequest.v1.2": {

"callId": "xxx",

"transactionId": "1"

}

}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.A cceptResponse.v1.2": {

"vnd.avaya.clientresources.Call.v1.2" : {

"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}

,
"transactionId": "1"
}
} \0

Failure Response

Ignore Call

vnd.avaya.clientresources.call.IgnoreRequest.v1.2

vnd.avaya.clientresources.call.IgnoreResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Ignore Call Request Payload (JSON)

{
"vnd.avaya.clientresources.call.IgnoreCall.v1.2": {
"callId": "xxx",
"transactionId": "1"
}
}\0

Success Response (JSON)

```
{
 "vnd.avaya.clientresources.call.IgnoreResponse.v1.2": {

"vnd.avaya.clientresources.Call.v1.2" : {

"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

 "callId": "abc124",

"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",

"audioDirection" : "remote,inactive",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},

"transactionId": "1"
}
} \0
```

Failure Response

CreateCall (MakeCall)

vnd.avaya.clientresources.call.CreateRequest.v1.2

vnd.avaya.clientresources.call.CreateResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Request Payload (JSON)

{
"vnd.avaya.clientresources.call. CreateRequest.v1.2": {
"remotePartyNumber" : "Remote Party Number Value" ,
"video" : "true" ,
"subject" : "string " ,
"conferencePasscode" : "0-9,#,*" ,
"conferenceId" : "0-9",
"lineAppearanceOwner" : "a-z, 0-9",
"lineAppearanceId" : "int",
"transactionId": "1"
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.CreateResponse.v1.2": {

"vnd.avaya.clientresources.Call.v1.2" : {

"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},

"transactionId": "1"

}
} \0

Failure Response

DTMF

vnd.avaya.clientresources.call.DTMFRequest.v1.2

vnd.avaya.clientresources.call.DTMFResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

DTMF Request Payload (JSON)

{
"vnd.avaya.clientresources.call.DTMFRequest.v1.2": {
"dtmfstring": "0-9,#,*",
"callId" : "xxx",
"transactionId": "1"
}
}\0

Response (JSON)

{
"vnd.avaya.clientresources.call.DTMFResponse.v1.2": {
 	
"vnd.avaya.clientresources.Call.v1.2" : {
 	
"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
"callId": "abc124",
 	
"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",
 	
"audioDirection" : "send_receive",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},
 	
"transactionId": "1"
}
} \0

Failure Response

Add videoCall (Escalate)

vnd.avaya.clientresources.call.videoRequest.v1.2

vnd.avaya.clientresources.call.videoResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Add video Request Payload (JSON)

{
"vnd.avaya.clientresources.call.VideoRequest.v1.2": {
"video": "true" ,
"callID" : "xxx",
"transactionId": "1"
}
} \0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.VideoResponse.v1.2": {

"vnd.avaya.clientresources.Call.v1.2" : {

"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",

"audioDirection" : "remote,send_receive",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} ,

"transactionId": "1"
}
} \0

Failure Response

Remove video (Deescalate)

vnd.avaya.clientresources.call.videoRequest.v1.2

vnd.avaya.clientresources.call.videoResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Remove Video Request Payload (JSON)

{
"vnd.avaya.clientresources.call.VideoRequest.v1.2": {
"video": "false" ,
"callID" : "xxx",
"transactionId": "1"
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.VideoResponse.v1.2": {
 	
"vnd.avaya.clientresources.Call.v1.2" : {
 	
"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
"callId": "abc124",
 	
"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",
 	
"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}
 	
,
"transactionId": "1"
}
} \0

Failure Response

Block Camera

vnd.avaya.clientresources.call.BlockCameraRequest.v1.2

vnd.avaya.clientresources.call.BlockCameraResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Block Camera Request Payload (JSON)

{
"**vnd.avaya.clientresources.call.BlockCameraRequest.v1.2**": {
"blockcamera": "true",
"callId": "xxx",
 	
"transactionId": "1"
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.BlockCameraResponse.v1.2": {
 	
"vnd.avaya.clientresources.Call.v1.2" : {
 	
"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
"callId": "abc124",
 	
"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",
 	
"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}
 	
"transactionId": "1"
}
} \0

Failure Response

UnBlock Camera

vnd.avaya.clientresources.call.BlockCameraRequest.v1.2

vnd.avaya.clientresources.call.BlockCameraResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Unblock Camera Request Payload (JSON)

{
"vnd.avaya.clientresources.call.BlockCameraRequest.v1.2": {
"blockcamera": "false",
"callId": "xxx",
 	
"transactionId": "1"
 	
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.BlockCameraResponse.v1.2": {

"vnd.avaya.clientresources.Call.v1.2" : {

"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}

,
"transactionId": "1"
},
}\0

Failure Response

Call Events

vnd.avaya.clientresources.call.UpdatedEvent.v1.2

UpdatedEvent (JSON)

```
{
"vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
 "transactionId": "1",

"vnd.avaya.clientresources.Call.v1.2" : {

"ASAIUserData" : "ASAI User Data",
"UCID" : "UCID value",
"VDNName" : "VDN Name",

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

 "callId": "abc124",

"collectedDigits" : "Collected Digits",
"inVDNTime" : "VDN Time",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}

}
}\0
```

Call Events will be reported when the call is local to the device (Computer mode), the call media is routed to the desk phone (Shared Control), and when call media is routed to the PSTN (Telecommuter or Other Phone mode). Call Events are also reported when the call is acted on by a remote device, in a MDA (Multiple Device Access, in Avaya Aura®) or Twinning (IP Office), or in a bridged line appearance call (BLA). Calls being managed by a remote device are identified with the remote attribute.

Media Devices

The media device API provides the external application with the current audio and video device that is selected for calls. The external application cannot change the active devices. Active devices can only be managed with the Client SDK Application.

Media Device

Resource	Description
/Resources/MediaDevices/audio	Specifies active audio devices.

Media Device Resource

Attribute	Description
recordingDevice	Specifies the recording device.
playbackDevice	Specifies the playback device. This does not apply to video.
deviceName	Specifies the name of the device.

Media Device Operations

Operation	URI	Description
query audio devices	/Resources/MediaDevices/audio	Returns the active audio input or output device.

Audio Device

vnd.avaya.clientresources.device.ActiveAudioDeviceRequest.v1.2

vnd.avaya.clientresources.device.ActiveAudioDeviceResponse.v1.2

vnd.avaya.clientresources.Error.v1.2

Audio Request Payload (JSON)

{
"vnd.avaya.clientresources.call.ActiveAudioDeviceRequest.v1.2": {
 	
"transactionId": "14"
 	
}
 	
}\0

Successful Response (JSON)

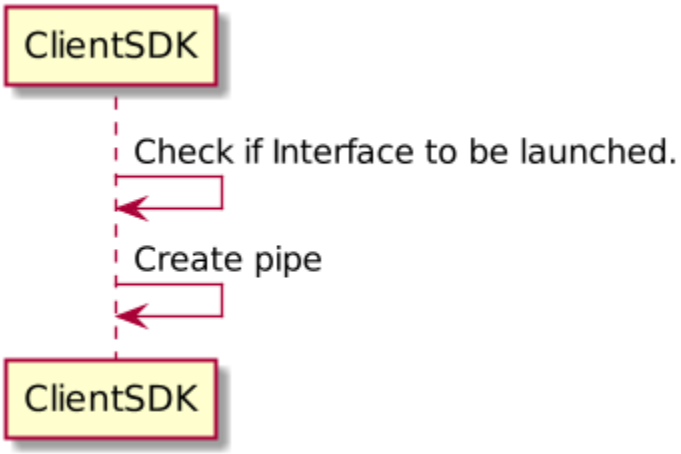
```
{
"vnd.avaya.clientresources.device.ActiveAudioDeviceResponse.v1.2": {
    "RecordingDevice.v1.2" : {
       "deviceName" :  "string"
    },
    "PlaybackDevice.v1.2" : {
       "deviceName" :  "string"
    }
"transactionId": "14"
}
}\0
```

Call Flows

Legend

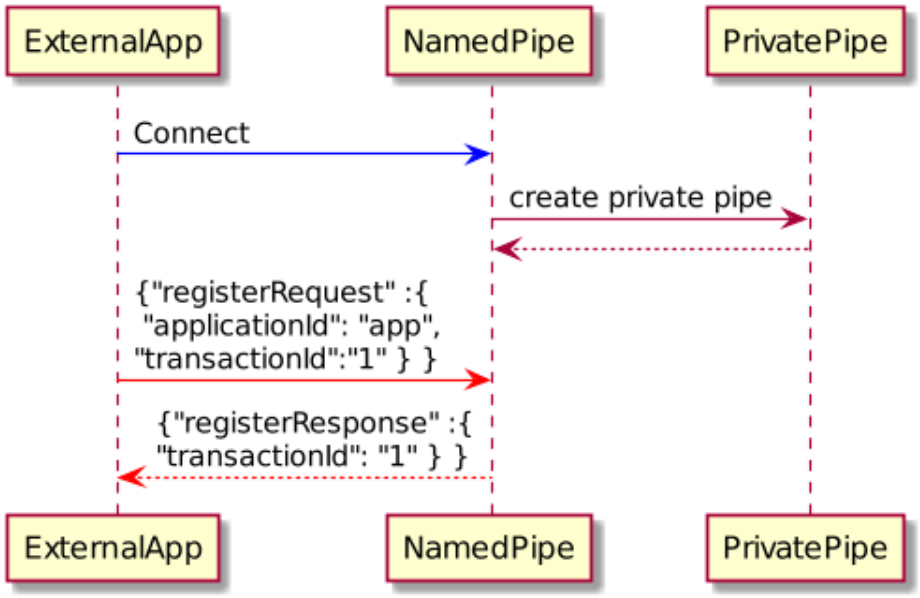
API Control Call Flows

Client SDK Initialization

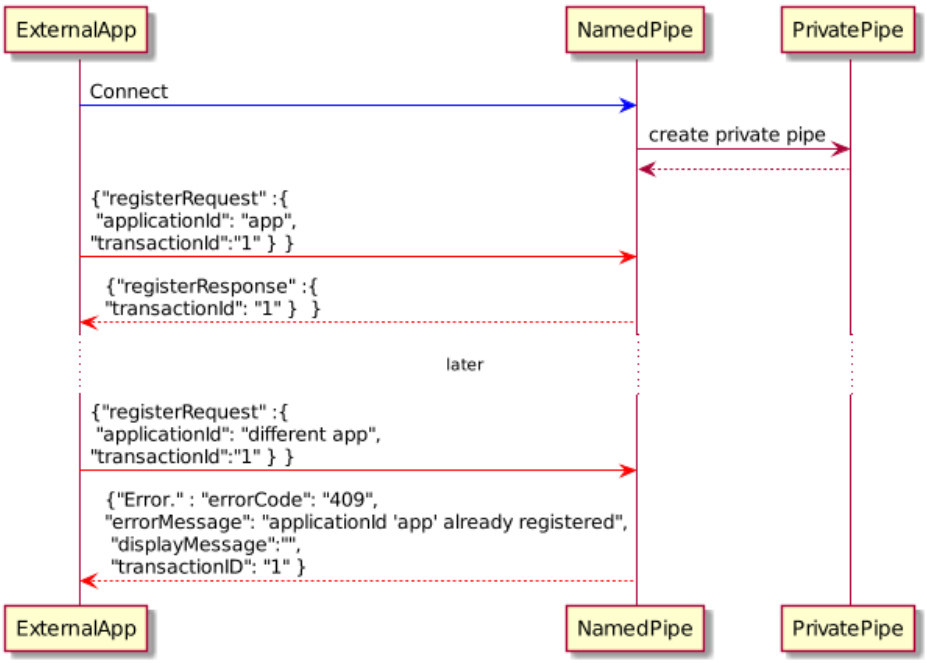


Register

This is sent by the external application when it wants to provide a rich call control experience.

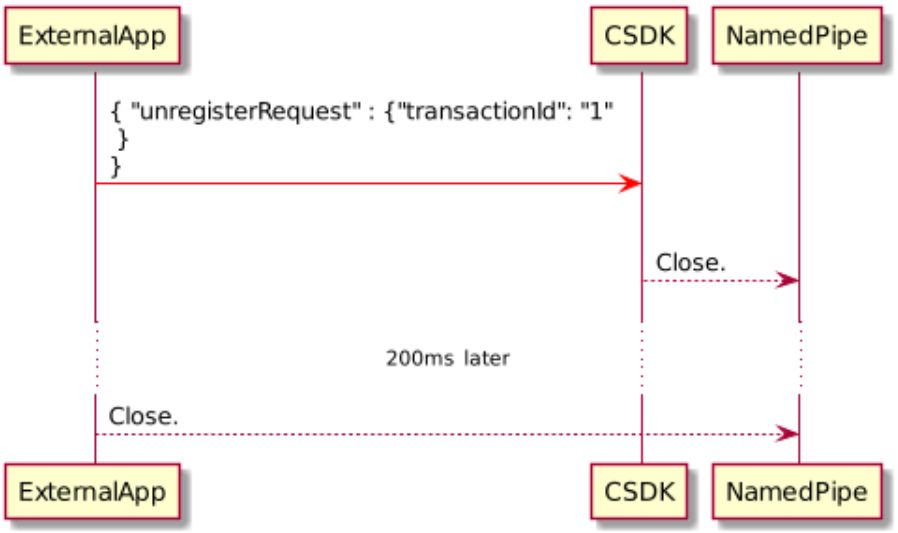


When the application registers again with a different applicationId, it is considered an application error. The registration is rejected by the Client SDK if the applicationId is different than the applicationId used in the initial Register request. The pipe is not closed and the original applicationId remains valid.



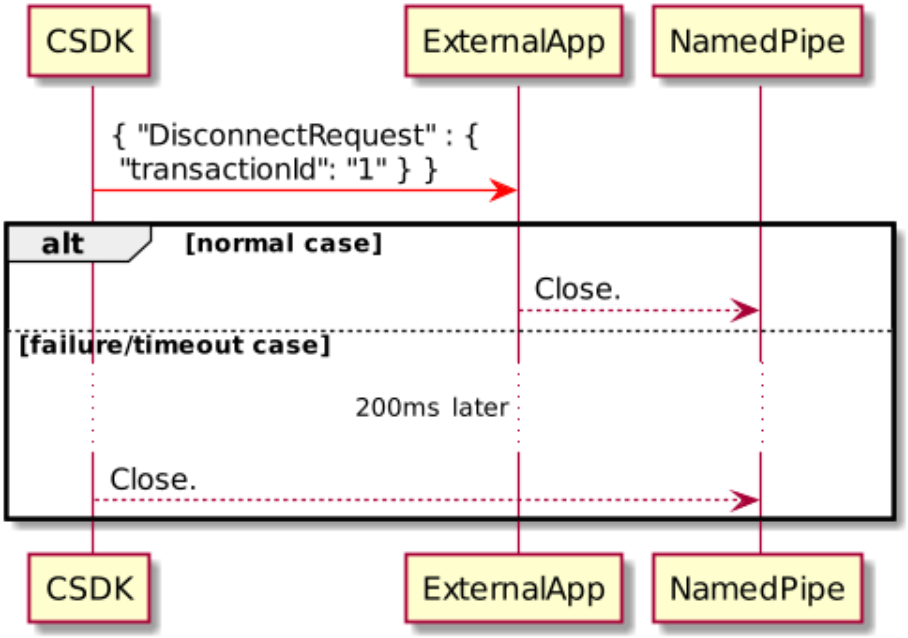
Unregister

This is sent by external application when it no longer wants the External Pipe. For example, this can occur during an application shut down or when the work station is in Sleep mode.



DisconnectRequest

This is sent by the Client SDK when it no longer wants the External Pipe. For example, this can occur during an application shut down or when the work station is in Sleep mode.



Call Handling

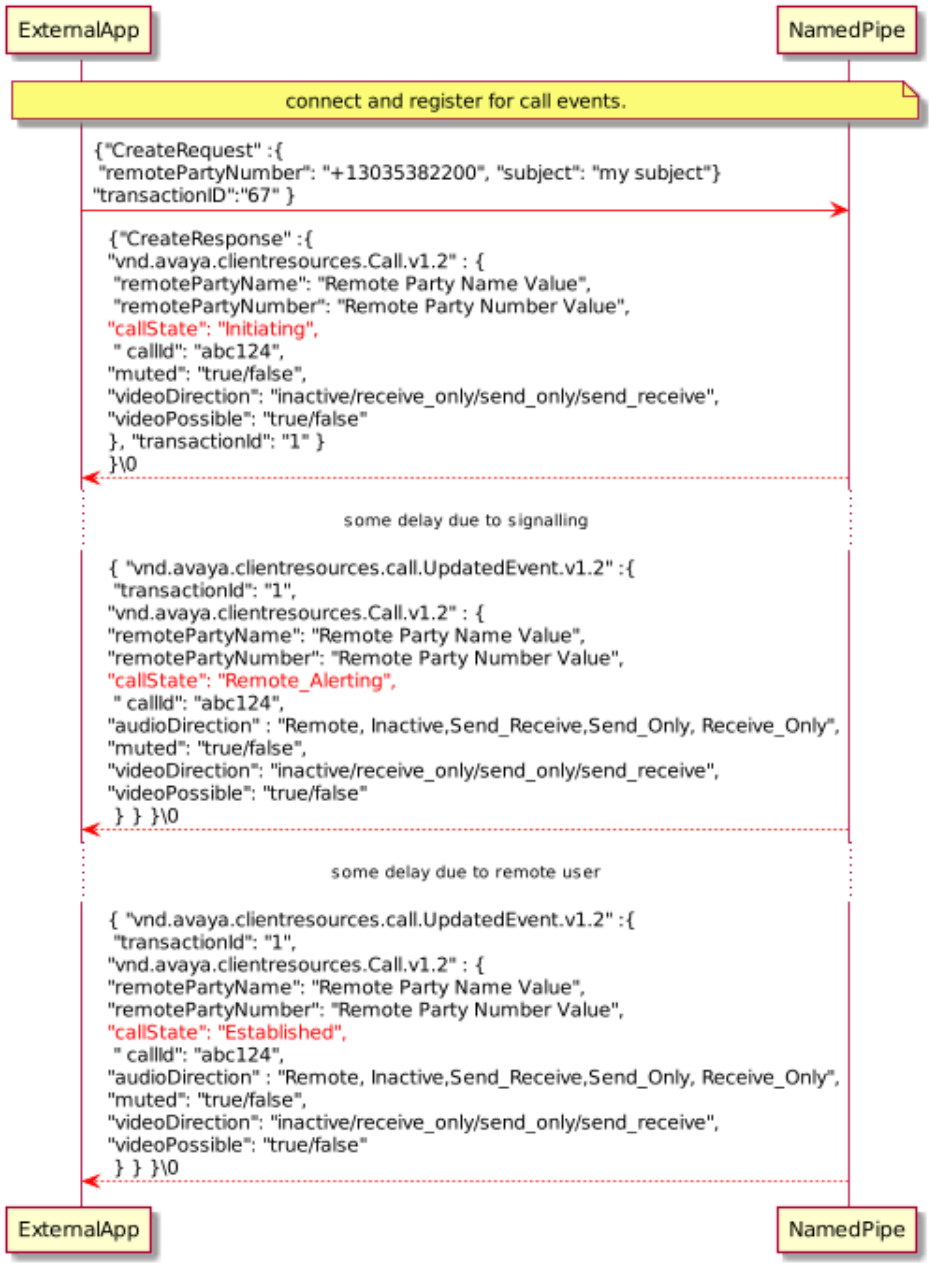
The following sections show the possible call flows for each operation, but the actual combination of responses and CallUpdatedEvents depend on the state of the call object at the time the response is fired. Call state transitions vary depending on the remote endpoint, network latency, and External Application API internal implementation. Examples include the following:

When creating a call to a conference server that answers the call immediately, the Remote_Alerting call state transition might be skipped and only Established is reported.

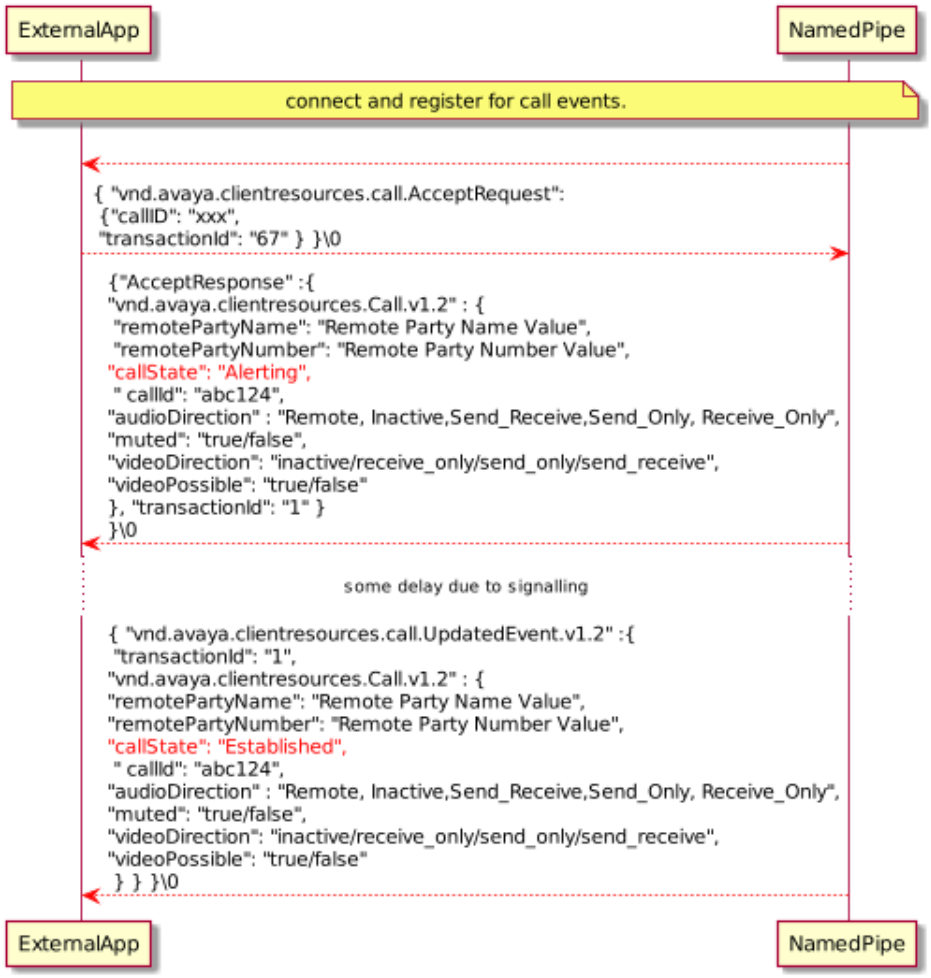
Holding and Unholding might not be reported and the more stable Held or Established call state is returned.

Make Call - Named Pipe

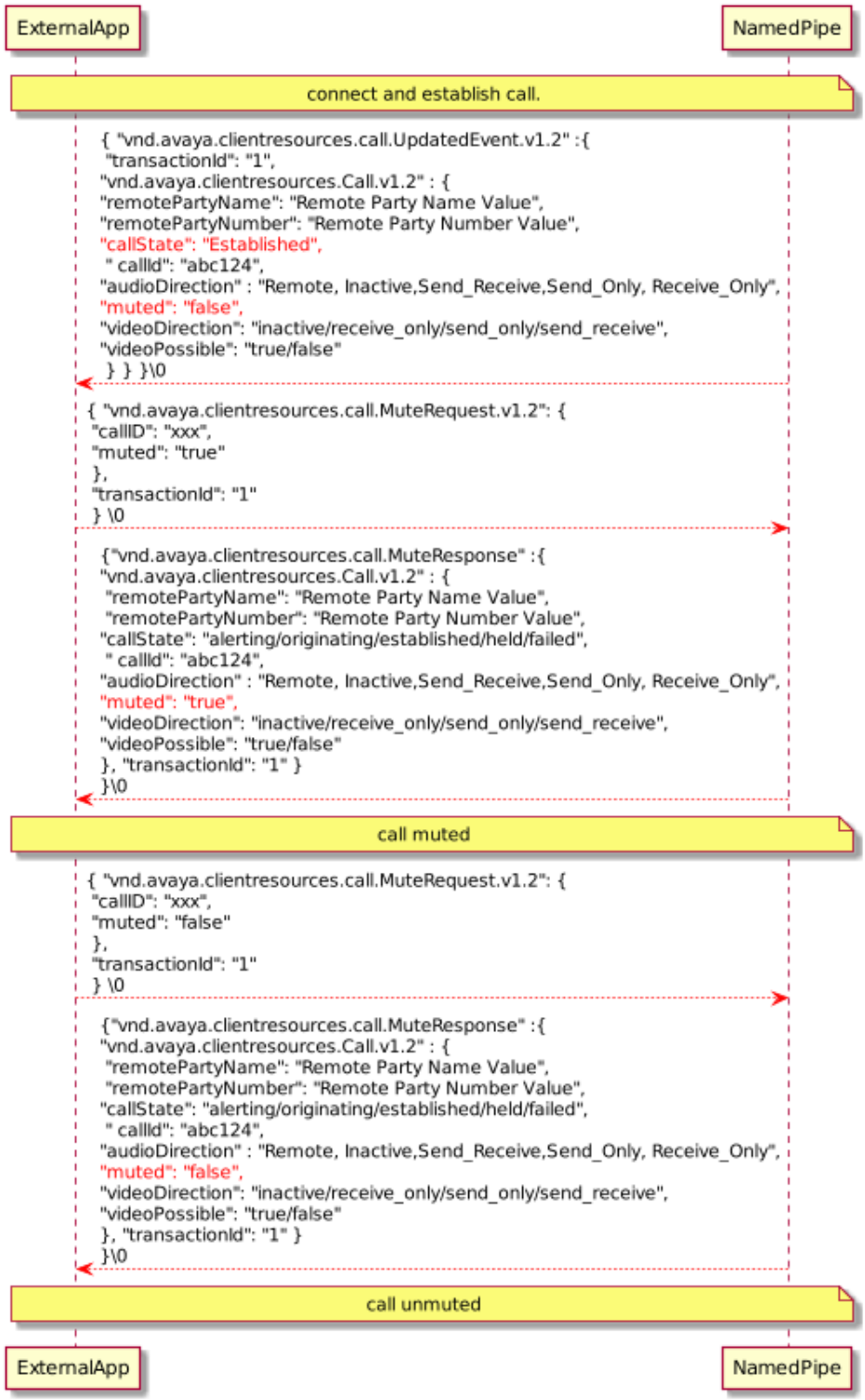
Make Call - Named Pipe


ExternalApp
NamedPipe
connect and register for call events.
{"CreateRequest" :{
"remotePartyNumber": "+13035382200", "subject": "my subject"}
"transactionID":"67" }
{"CreateResponse" :{
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Initiating",
" callId": "abc124",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}, "transactionId": "1" }
}\0
some delay due to signalling
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Remote_Alerting",
" callId": "abc124",
"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
some delay due to remote user
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Established",
" callId": "abc124",
"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
ExternalApp
NamedPipe

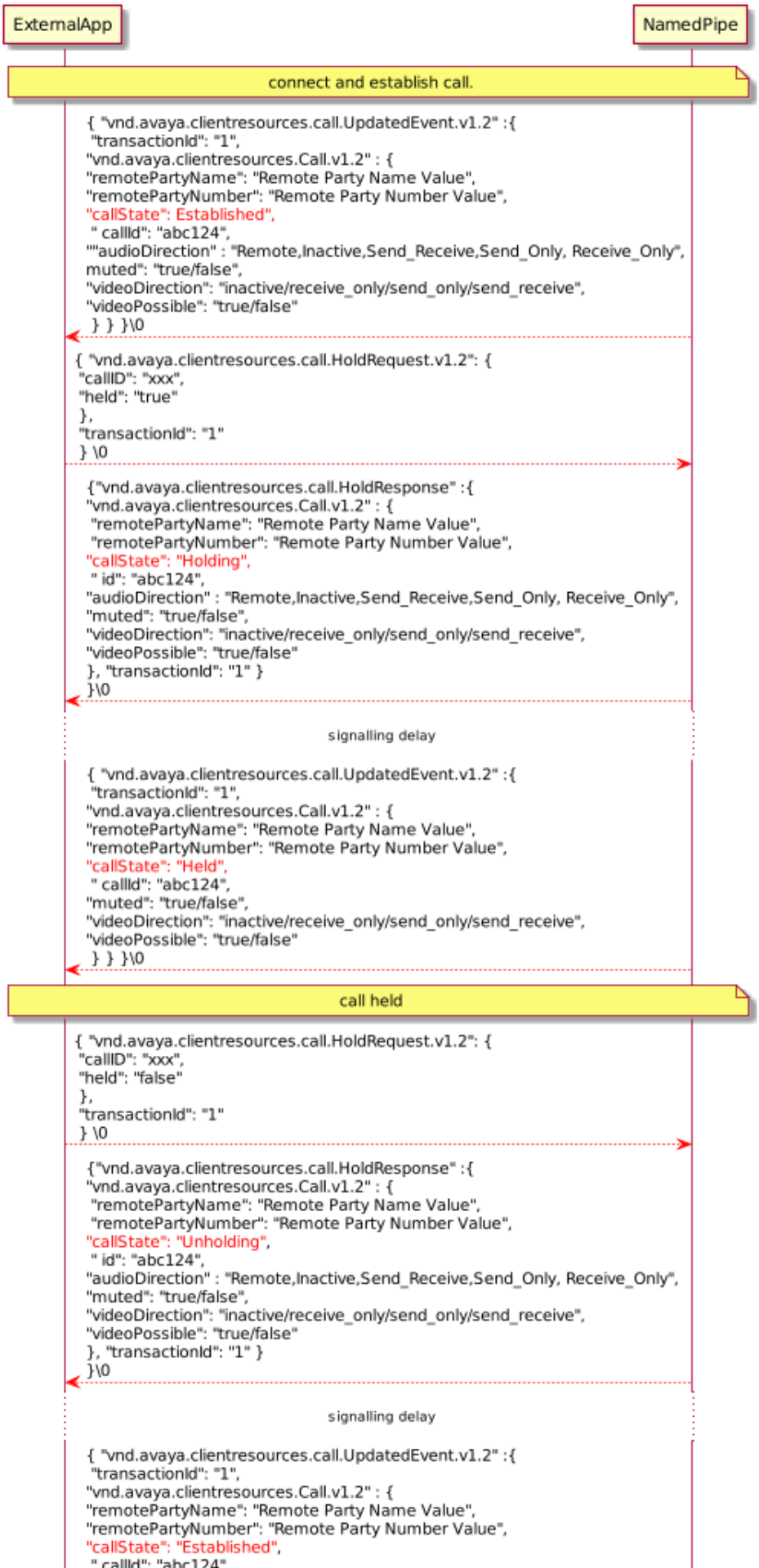
Answer Call



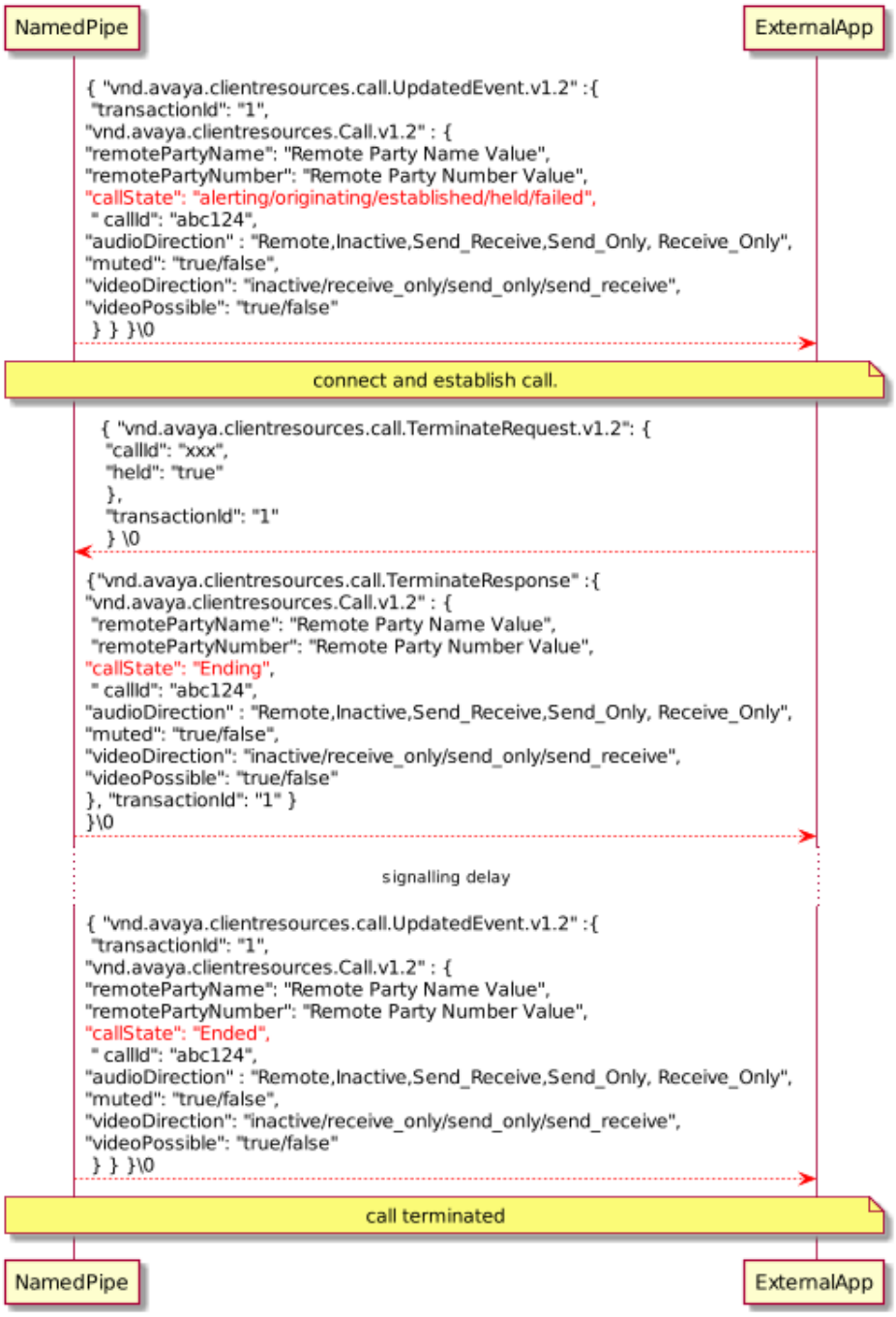
Mute and Unmute call



Hold and Retrieve call

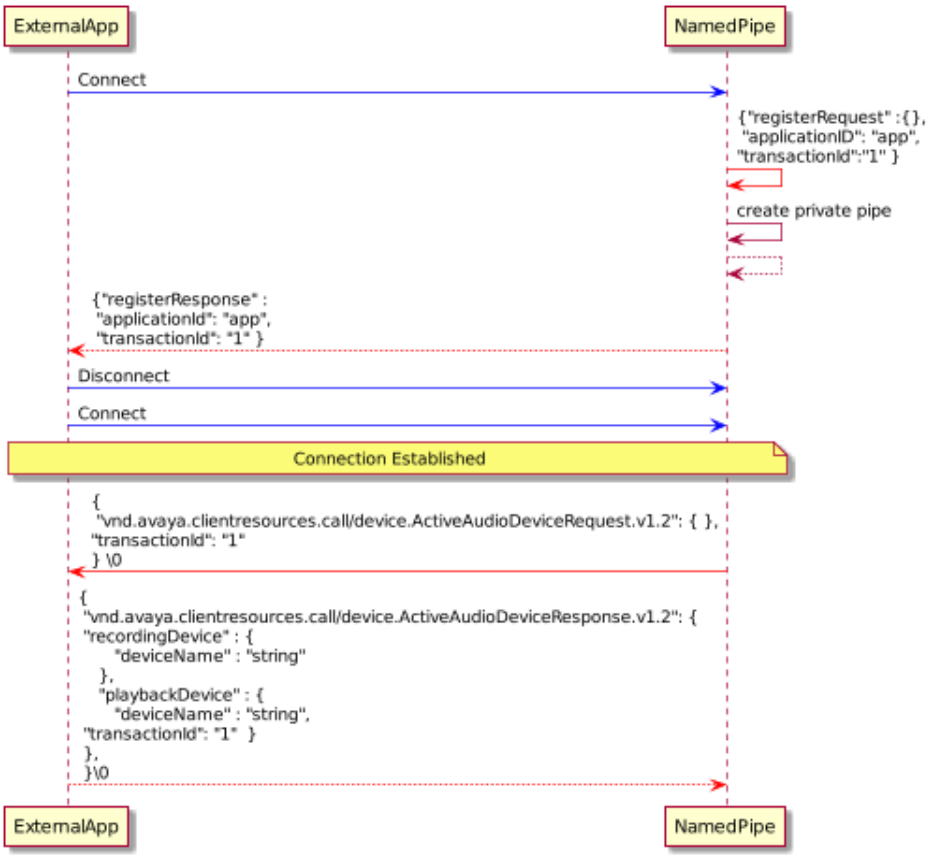

ExternalApp
NamedPipe
connect and establish call.
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": Established",
" callId": "abc124",
""audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
{ "vnd.avaya.clientresources.call.HoldRequest.v1.2": {
"callID": "xxx",
"held": "true"
},
"transactionId": "1"
} \0
{"vnd.avaya.clientresources.call.HoldResponse" :{
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Holding",
" id": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}, "transactionId": "1" }
}\0
signalling delay
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Held",
" callId": "abc124",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
call held
{ "vnd.avaya.clientresources.call.HoldRequest.v1.2": {
"callID": "xxx",
"held": "false"
},
"transactionId": "1"
} \0
{"vnd.avaya.clientresources.call.HoldResponse" :{
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Unholding",
" id": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}, "transactionId": "1" }
}\0
signalling delay
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Established",

Terminate or End Call



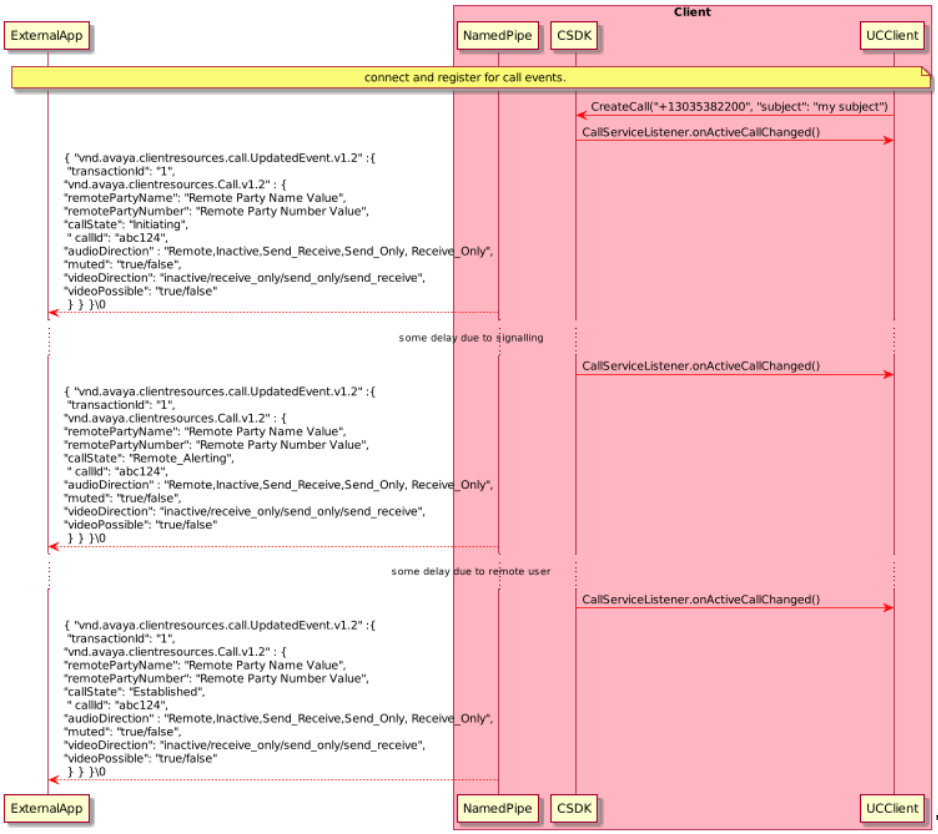
Media Device Listing

GetActiveAudioDevices



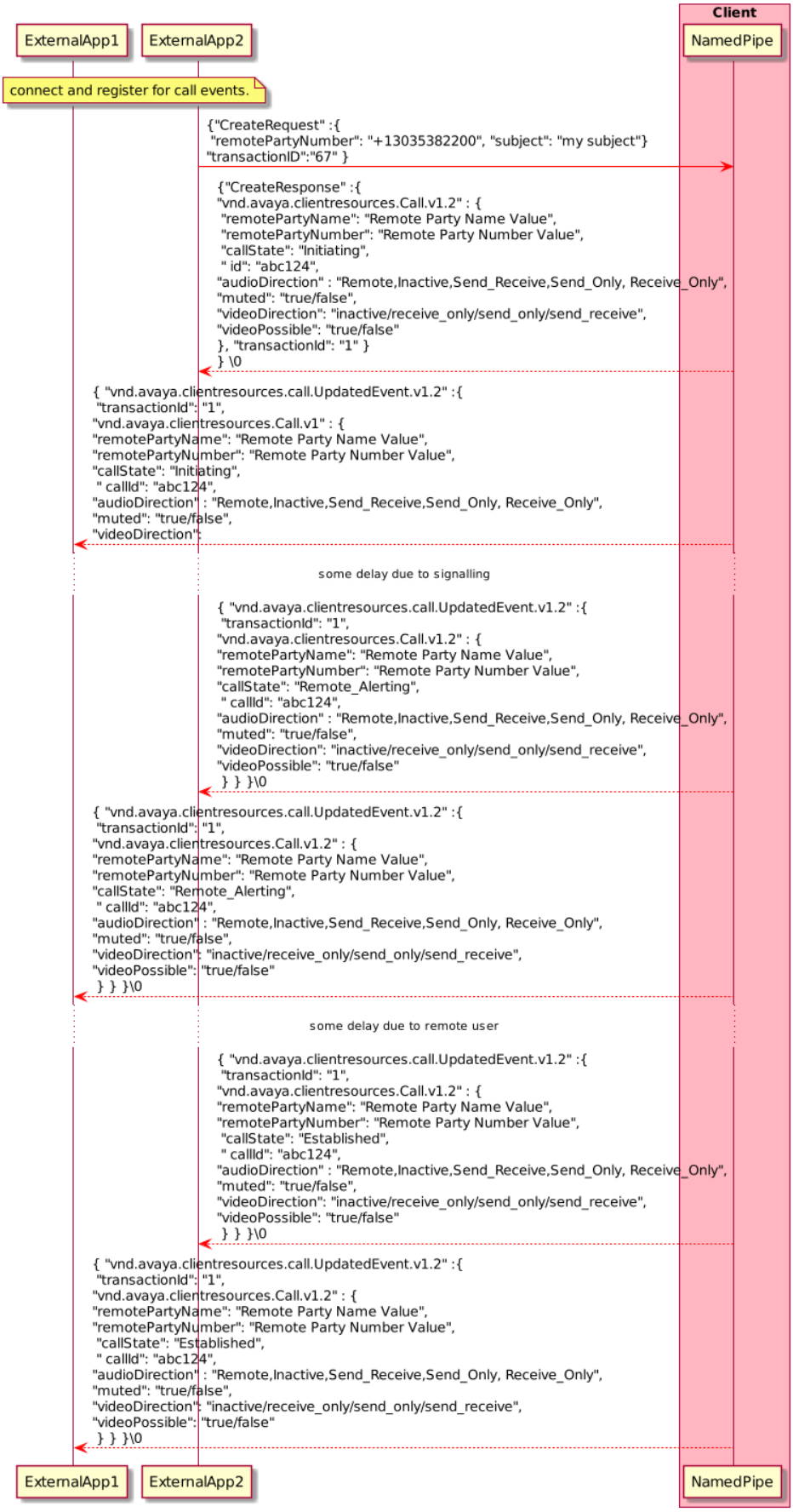
Application Interworking

Call Created by UC Application



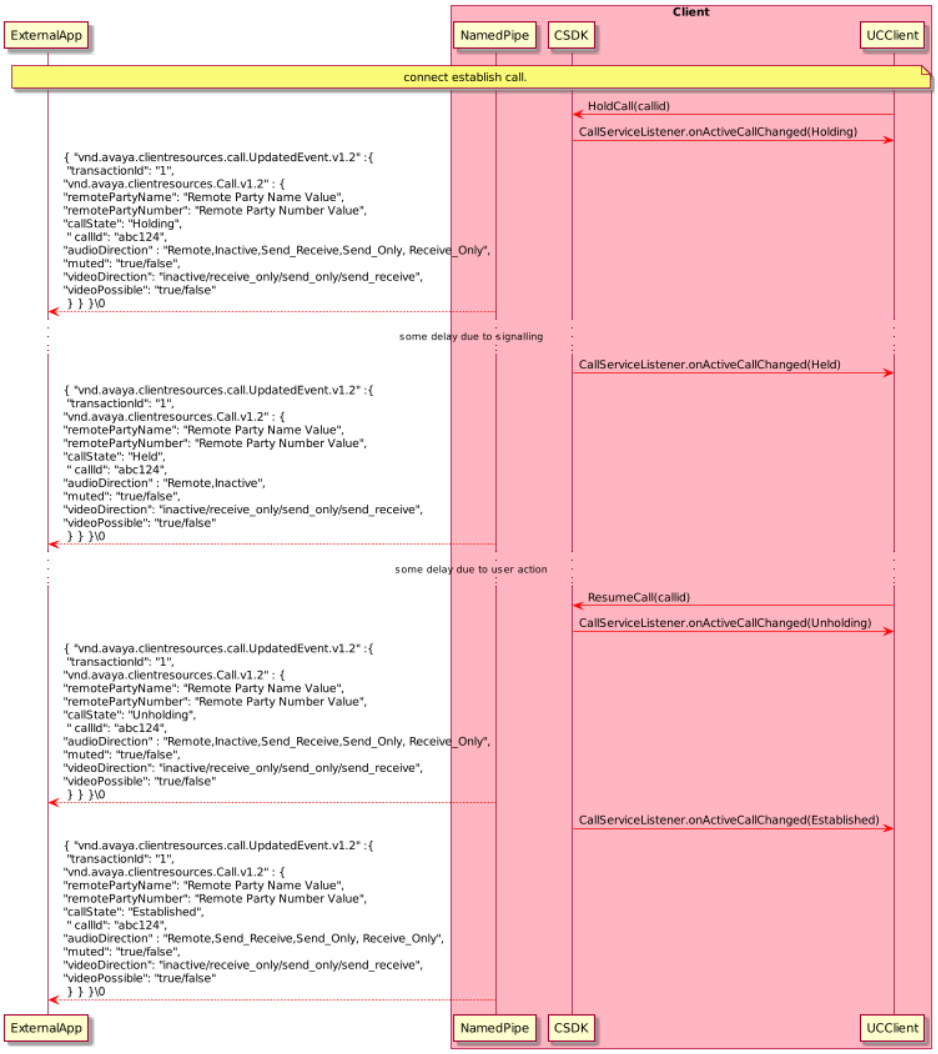
The above example shows the external application receiving call updates ahead of the External Application API, but this is for illustrative purposes and this behaviour is not guaranteed. The order of events or notifications is an implementation detail.

Call Created by Peer External Application


Client
ExternalApp1
ExternalApp2
NamedPipe
connect and register for call events.
{"CreateRequest" :{
"remotePartyNumber": "+13035382200", "subject": "my subject"}
"transactionID":"67" }
{"CreateResponse" :{
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Initiating",
" id": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}, "transactionId": "1" }
} \0
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Initiating",
" callId": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection":
some delay due to signalling
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Remote_Alerting",
" callId": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Remote_Alerting",
" callId": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
some delay due to remote user
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Established",
" callId": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Established",
" callId": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
ExternalApp1
ExternalApp2
NamedPipe

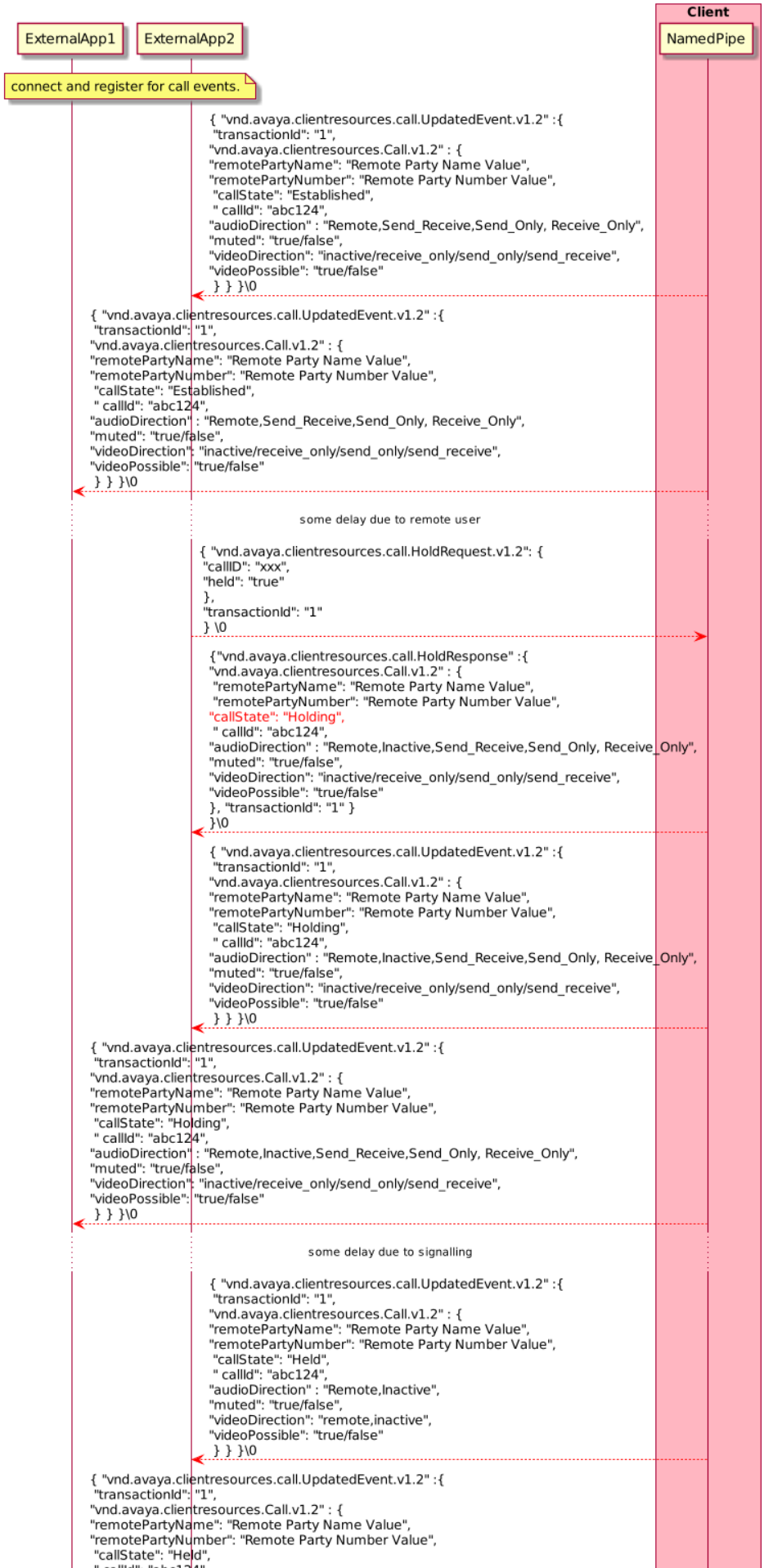
The above example shows the ExternalApp2 receiving call updates ahead of ExternalApp2, but this is for illustrative purposes and this behaviour is not guaranteed. The order of events or notifications is an implementation detail.

Call Hold by UC Application



The above example shows the user holding and resuming the call from the external application. It is possible that the user can use *HoldCall* using the external application and resume through the External Application API, or vice versa.

Call Hold by Peer External Application


Client
ExternalApp1
ExternalApp2
NamedPipe
connect and register for call events.
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Established",
" callId": "abc124",
"audioDirection" : "Remote,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Established",
" callId": "abc124",
"audioDirection" : "Remote,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
some delay due to remote user
{ "vnd.avaya.clientresources.call.HoldRequest.v1.2": {
"callID": "xxx",
"held": "true"
},
"transactionId": "1"
} \0
{"vnd.avaya.clientresources.call.HoldResponse" :{
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Holding",
" callId": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}, "transactionId": "1" }
}\0
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Holding",
" callId": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Holding",
" callId": "abc124",
"audioDirection" : "Remote,Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} } }\0
some delay due to signalling
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Held",
" callId": "abc124",
"audioDirection" : "Remote,Inactive",
"muted": "true/false",
"videoDirection": "remote,inactive",
"videoPossible": "true/false"
} } }\0
{ "vnd.avaya.clientresources.call.UpdatedEvent.v1.2" :{
"transactionId": "1",
"vnd.avaya.clientresources.Call.v1.2" : {
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "Held",

In the above example, the sequence shows ExternalApp2 holding the call, and ExternalApp1 retrieving (unholding) the call. In addition, there is a UpdatedEvent containing the same call state information as the HoldResponse. Since the data is identical, there is minimal impact, but it is not ideal. The order of events is an implementation detail.

References

Windows NamedPipe Reference: https://msdn.microsoft.com/en-us/library/windows/desktop/aa365590%28v=vs.85%29.aspx?f=255&MSPPError=-2147217396

Windows PipeList: https://technet.microsoft.com/en-us/sysinternals/dd581625.aspx

Appendix

Windows C# Named Pipe Sample Code

```
using System;
using System.IO;
using System.IO.Pipes;

namespace ExternalApplicationConnect
{
 class Program
 {
 static void Main(string[] args)
 {
 NamedPipeClientStream pipeClient = new NamedPipeClientStream(".","AvayaCSDK-
Administrator",PipeDirection.InOut,PipeOptions.None);
 if (pipeClient.IsConnected != true)
 {
 pipeClient.Connect();
 }

 StreamReader sr = new StreamReader(pipeClient);
 StreamWriter sw = new StreamWriter(pipeClient);
 string registerString =
"{ \"vnd.avaya.clientresources.RegisterRequest.v1.2\" : { \"applicationId\" :
\"TestApp\", \"transactionId\" : \"23763992\" } } \0";
 string createCallString =
"{ \"vnd.avaya.clientresources.call.CreateRequest.v1.2\":
{ \"remotePartyNumber\":\"+13035382200,,683042\", \"transactionId\":
\"1765675\" } } \0 ";
 try
 {
 sw.Write(createCallString);
 sw.Flush();
 }
 catch (Exception ex) { throw ex; }
 }
 }
```

}

Word Doc....

Avaya Client SDK External Application API

Notice

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer

"Documentation" means information published in varying mediums which may include product information, operating instructions and performance specifications that are generally made available to users of products. Documentation does not include marketing materials. Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of Documentation unless such modifications, additions, or deletions were performed by or on the express behalf of Avaya. End User agrees to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer

Avaya is not responsible for the contents or reliability of any linked websites referenced within this site or Documentation provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty

Avaya provides a limited warranty on Avaya hardware and software. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty

language, as well as information regarding support for this product while under warranty is available to Avaya customers and other parties through the Avaya Support website: https://support.avaya.com/helpcenter/ getGenericDetails?detailId=C20091120112456651010 under the link "Warranty & Product Lifecycle" or such successor site as designated by Avaya. Please note that if You acquired the product(s) from an authorized Avaya Channel Partner outside of the United States and Canada, the warranty is provided to You by said Avaya Channel Partner and not by Avaya.

"Hosted Service" means an Avaya hosted service subscription that You acquire from either Avaya or an authorized Avaya Channel Partner (as applicable) and which is described further in Hosted SAS or other service description documentation regarding the applicable hosted service. If You purchase a Hosted Service subscription, the foregoing limited warranty may not apply but You may be entitled to support services in connection with the Hosted Service as described further in your service description documents for the applicable Hosted Service. Contact Avaya or Avaya Channel Partner (as applicable) for more information.

Hosted Service

THE FOLLOWING APPLIES ONLY IF YOU PURCHASE AN AVAYA HOSTED SERVICE SUBSCRIPTION FROM AVAYA OR AN AVAYA CHANNEL PARTNER (AS APPLICABLE), THE TERMS OF USE FOR HOSTED SERVICES ARE AVAILABLE ON THE AVAYA WEBSITE, HTTPS://SUPPORT.AVAYA.COM/LICENSEINFO UNDER THE LINK "Avaya Terms of Use for Hosted Services" OR SUCH SUCCESSOR SITE AS DESIGNATED BY AVAYA, AND ARE APPLICABLE TO ANYONE WHO ACCESSES OR USES THE HOSTED SERVICE. BY ACCESSING OR USING THE HOSTED SERVICE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE DOING SO (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER"), AGREE TO THE TERMS OF USE. IF YOU ARE ACCEPTING THE TERMS OF USE ON BEHALF A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT THAT YOU HAVE THE AUTHORITY TO BIND SUCH ENTITY TO THESE TERMS OF USE. IF YOU DO NOT HAVE SUCH AUTHORITY, OR

IF YOU DO NOT WISH TO ACCEPT THESE TERMS OF USE, YOU MUST NOT ACCESS OR USE THE HOSTED SERVICE OR AUTHORIZE ANYONE TO ACCESS OR USE THE HOSTED SERVICE. Licenses THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, HTTPS://SUPPORT.AVAYA.COM/LICENSEINFO , UNDER THE LINK "AVAYA SOFTWARE LICENSE TERMS (Avaya Products)" OR SUCH SUCCESSOR SITE AS DESIGNATED BY AVAYA, ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AVAYA CHANNEL PARTNER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AVAYA CHANNEL PARTNER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA CHANNEL PARTNER; AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF

AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER"), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE ("AVAYA").

Avaya grants You a license within the scope of the license types described below, with the exception of Heritage Nortel Software, for which the scope of the license is detailed below. Where the order documentation does not expressly identify a license type, the applicable license will be a Designated System License. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the documentation or other materials available to You. "Software" means computer programs in object code, provided by Avaya or an Avaya Channel Partner, whether as stand-alone products, pre-installed on hardware products, and any upgrades, updates, patches, bug fixes, or modified versions thereto. "Designated Processor" means a single stand-alone computing device. "Server" means a Designated Processor that hosts a software application to be accessed by multiple users. "Instance" means a single copy of the Software executing at a particular time: (i) on one physical machine; or (ii) on one deployed software virtual machine ("VM") or similar deployment.

License types

Designated System(s) License (DS). End User may install and use each copy or an Instance of the Software only on a number of Designated Processors up to the number indicated in the order. Avaya may require the Designated Processor(s) to be identified in the order by type, serial number, feature key, Instance, location or other specific designation, or to be provided by End User to Avaya through electronic means established by Avaya specifically for this purpose.

Concurrent User License (CU). End User may install and use the Software on multiple Designated Processors or one or more Servers, so long as only the licensed number of Units are accessing and using the Software at any given time. A "Unit" means the unit on which Avaya, at its sole discretion, bases the pricing of its licenses and can be, without limitation, an agent, port or user, an e-mail or voice mail account in the name of a person or corporate function (e.g., webmaster or helpdesk), or a directory entry in the administrative database utilized by the Software that permits one user to interface with the Software. Units may be linked to a specific, identified Server or an Instance of the Software.

Database License (DL). End User may install and use each copy or an Instance of the Software on one Server or on multiple Servers provided that each of the Servers on which the Software is installed communicates with no more than one Instance of the same database.

CPU License (CP). End User may install and use each copy or Instance of the Software on a number of Servers up to the number indicated in the order provided that the performance capacity of the Server(s) does not exceed the performance capacity specified for the Software. End User may not re-install or operate the Software on Server(s) with a larger performance capacity without Avaya's prior consent and payment of an upgrade fee.

Named User License (NU). You may: (i) install and use each copy or Instance of the Software on a single Designated Processor or Server per authorized Named User (defined below); or (ii) install and use each copy or Instance of the Software on a Server so long as only authorized Named Users access and use the Software. "Named User", means a user or device that has been expressly authorized by Avaya to access and use the Software. At Avaya's sole discretion, a "Named User" may be, without limitation, designated by name, corporate function (e.g., webmaster or helpdesk), an e-mail or voice mail account in the name of a person or corporate function, or a directory entry in the administrative database utilized by the Software that permits one user to interface with the Software.

Shrinkwrap License (SR). You may install and use the Software in accordance with the terms and conditions of the applicable license agreements, such as "shrinkwrap" or "clickthrough" license accompanying or applicable to the Software ("Shrinkwrap License").

Heritage Nortel Software

"Heritage Nortel Software" means the software that was acquired by Avaya as part of its purchase of the Nortel Enterprise Solutions Business in December 2009. The Heritage Nortel Software is the software contained within the list of Heritage Nortel Products located at https://support.avaya.com/LicenseInfo under the link "Heritage Nortel Products" or such successor site as designated by Avaya. For Heritage Nortel Software, Avaya grants Customer a license to use Heritage Nortel Software provided hereunder solely to the extent of the authorized activation or authorized usage level, solely for the purpose specified in the Documentation, and solely as embedded in, for execution on, or for communication with Avaya equipment. Charges for Heritage Nortel Software may be based on extent of activation or use authorized as specified in an order or invoice.

Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation, Software, Hosted Service, or hardware provided by Avaya. All content on this site, the documentation, Hosted Service, and the product provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software unless expressly authorized by Avaya. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil offense under the applicable law.

Virtualization

The following applies if the product is deployed on a virtual machine. Each product has its own ordering code and license types. Note that each Instance of a product must be separately licensed and ordered. For example, if the end user customer or Avaya Channel Partner would like to install two Instances of the same type of products, then two products of that type must be ordered.

Third Party Components

"Third Party Components" mean certain software programs or portions thereof included in the Software or Hosted Service may contain software (including open source software) distributed under third party agreements ("Third Party Components"), which contain terms regarding the rights to use certain portions of the Software ("Third Party Terms"). As required, information regarding distributed Linux OS source code (for those products that have distributed Linux OS source code) and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply is available in the products, Documentation or on Avaya's website at: https:// support.avaya.com/Copyright or such successor site as designated by Avaya. The open source software license terms provided as Third Party Terms are consistent with the license rights granted in these Software License Terms, and may contain additional rights benefiting You, such as modification and distribution of the open source software. The Third Party Terms shall take precedence over these Software License Terms, solely with respect to the applicable Third Party Components to the extent that these Software License Terms impose greater restrictions on You than the applicable Third Party Terms.

The following applies only if the H.264 (AVC) codec is distributed with the product. THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP://WWW.MPEGLA.COM .

Service Provider

THE FOLLOWING APPLIES TO AVAYA CHANNEL PARTNER'S HOSTING OF AVAYA PRODUCTS OR SERVICES. THE PRODUCT OR HOSTED SERVICE MAY USE THIRD PARTY COMPONENTS SUBJECT TO THIRD PARTY TERMS AND REQUIRE A SERVICE PROVIDER TO BE INDEPENDENTLY LICENSED DIRECTLY FROM THE THIRD PARTY SUPPLIER. AN AVAYA CHANNEL PARTNER'S HOSTING OF AVAYA PRODUCTS MUST BE AUTHORIZED IN WRITING BY AVAYA AND IF THOSE HOSTED PRODUCTS USE OR EMBED CERTAIN THIRD PARTY SOFTWARE, INCLUDING BUT NOT LIMITED TO MICROSOFT SOFTWARE OR CODECS, THE AVAYA CHANNEL PARTNER IS REQUIRED TO INDEPENDENTLY OBTAIN ANY APPLICABLE LICENSE AGREEMENTS, AT THE AVAYA CHANNEL PARTNER'S EXPENSE, DIRECTLY FROM THE APPLICABLE THIRD PARTY SUPPLIER.

WITH RESPECT TO CODECS, IF THE AVAYA CHANNEL PARTNER IS HOSTING ANY PRODUCTS THAT USE OR EMBED THE G.729 CODEC, H.264 CODEC, OR H.265 CODEC, THE AVAYA CHANNEL PARTNER ACKNOWLEDGES AND AGREES THE AVAYA CHANNEL PARTNER IS RESPONSIBLE FOR ANY AND ALL RELATED FEES AND/OR ROYALTIES. THE G.729 CODEC IS LICENSED BY SIPRO LAB TELECOM INC. SEE WWW.SIPRO.COM/CONTACT.HTML. THE H.264 (AVC) CODEC IS LICENSED UNDER THE

AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO: (I) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (II) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION FOR H.264 (AVC) AND H.265 (HEVC) CODECS MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP:// WWW.MPEGLA.COM.

Compliance with Laws

You acknowledge and agree that it is Your responsibility for complying with any applicable laws and regulations, including, but not limited to laws and regulations related to call recording, data privacy, intellectual property, trade secret, fraud, and music performance rights, in the country or territory where the Avaya product is used.

Preventing Toll Fraud

"Toll Fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of Toll Fraud associated with your system and that, if Toll Fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Toll Fraud intervention

If You suspect that You are being victimized by Toll Fraud and You need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support website: https://support.avaya.com or such successor site as designated by Avaya.

Security Vulnerabilities

Information about Avaya's security support policies can be found in the Security Policies and Support section of https:// support.avaya.com/security .

Suspected Avaya product security vulnerabilities are handled per the Avaya Product Security Support Flow (https:// support.avaya.com/css/P8/documents/100161515).

Downloading Documentation

For the most current versions of Documentation, see the Avaya Support website: https://support.avaya.com , or such successor site as designated by Avaya.

Contact Avaya Support

See the Avaya Support website: https://support.avaya.com for product or Hosted Service notices and articles, or to report a problem with your Avaya product or Hosted Service. For a list of support telephone numbers and contact addresses, go to the Avaya Support website: https://support.avaya.com (or such successor site as designated by Avaya), scroll to the bottom of the page, and select Contact Avaya Support.

Trademarks

The trademarks, logos and service marks ("Marks") displayed in this site, the Documentation, Hosted Service(s), and product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, its licensors, its suppliers, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the Documentation, Hosted Service(s) and product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party.

Avaya is a registered trademark of Avaya Inc.

All non-Avaya trademarks are the property of their respective owners. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Table of Contents

DOCUMENT PURPOSE

DOCUMENT VERSION

WHAT'S NEW

OVERVIEW

- APPLICATION INTEGRATION
- CALL CONTROL CAPABILITY TABLE
- WHY JSON?
- NAMED PIPE API MODEL
  - *Pipe Discovery*
  - *Application Sandboxing for MacOS*
  - *Basic Interworking Model*
  - *Enhanced Interworking Model*
- SECURITY MODEL
  - *Authentication*
  - *Networking*
  - *Named Pipe Connectivity Model*
  - *Protocol Framing*
  - *Denial of Service*
  - *Rate Limiting*
  - *Application Identification*
  - *Disabling the External Application Interface*
  - *Internationalization*
- BACKWARD AND FORWARD COMPATIBILITY
  - *Versions*
  - *External Application API Backward Compatibility*
  - *External Application API Forward Compatibility*
  - *Current Version*

API DEFINITION

- *Media Types*
- *Pipe Management*
- *Pipe Management Events*
- *Calls*
- *Call Events*
- *Media Devices*

CALL FLOWS

- *Legend*
- API CONTROL CALL FLOWS
  - *Client SDK Initialization*
  - *Register*
  - *Unregister*
  - *DisconnectRequest*
- CALL HANDLING
  - *Make Call - Named Pipe*
  - *Make Call - Named Pipe*
  - *Answer Call*
  - *Mute and Unmute call*
  - *Hold and Retrieve call*
  - *Terminate or End Call*
- MEDIA DEVICE LISTING
  - *GetActiveAudioDevices*
- APPLICATION INTERWORKING
  - *Call Created by UC Application*
  - *Call Created by Peer External Application*
  - *Call Hold by UC Application*
  - *Call Hold by Peer External Application*

REFERENCES

APPENDIX

WINDOWS C# NAMED PIPE SAMPLE CODE

Document Purpose

The purpose of this document is to define the Unified Communications Application (UCA) portfolio requirements for external applications to interact with the Avaya Client SDK communication package and with applications that utilize this package, such as Avaya Workplace.

This document describes the resources that make up the official *External Application Interface for the Client SDK Communication Services Package* v1.1 (*API*).

Document Version

Version	Date	Author	Comment
187	Oct 02, 2020 11:58	McIntyre, Stephen (Stephen)	Clarified platform support.

What's New

External Application APIs were up versioned to v1.1, with the primary difference being that calls with media routed remotely, such as shared control and telecommuter calls, are clearly

identifiable within the API as calls with remote media. Applications written with to the v1.0 of the External Application API will continue to work, and be supported.

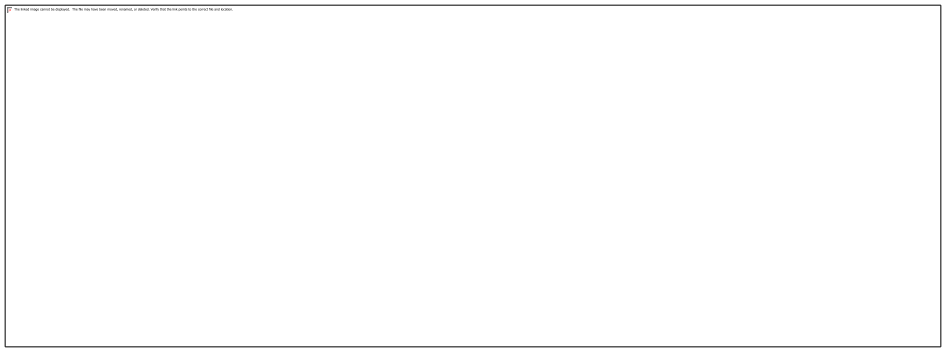
Overview

The purpose of the APIs is to allow general applications executing locally on the workstation to send primitive call control requests to Avaya Client SDK. The External API was defined as a consolidated API to enable application vendors authoring applications for multiple platforms to leverage a consistent API. This API is supported on Windows and Mac OS X platforms, and is not applicable to Android or iOS platforms.

The API enables applications to create and control calls, and to discover calls through JSON messaging over a named pipe. The API is versioned, and is versioned independently of the Avaya Client SDK version to provide external applications with a stable set of functionality.

Avaya Workplace is the lead Avaya application that is built on the Avaya Client SDK, and is often referred or used in examples throughout the document. The External Application API may be enabled by any application that is built on the Avaya Client SDK. It is the choice of the application to determine if the External Application API is enabled.

Within this document, the Avaya Client SDK will be referenced as either the Client SDK or CSDK.



Application Integration

When developing an application that uses the External API, Avaya recommends that partners integrate their applications with the Avaya Workplace (previously known as Avaya Equinox) application first, treating Avaya Workplace as the reference implementation. The External API will be available in other Client SDK applications written by Avaya or third party organizations. Once you have successfully integrated with Workplace, you can begin integrating with other Client SDK applications.

Call Control Capability Table

Avaya Workplace is the latest Avaya Unified Communications client, and it extends the capabilities of Avaya Communicator and Avaya one-X® Communicator available for External Applications. The following table shows the capabilities available to External Applications, and if the capability is newly introduced with the External Application Interface.

Call Control Capability	**Avaya Communicator 2.0 Interface**	**Avaya Equinox (3.0 - 3.6)**	**Avaya Workplace 3.7+**	**Notes**
Accept Incoming Call	Y	Y	Y	Existing capability
Add/Remove video	N	Y	Y	capability introduced in Equinox
Block Camera	N	Y	Y	capability introduced in Equinox
Calling line ID	N	Y	Y	capability introduced in Equinox
Create call	Y	Y	Y	Existing capability • Not supported for HTTPUA calls.
Hold/Retrieve	Y	Y	Y	Existing capability
Ignore Incoming Call	N	Y	Y	Existing capability
Terminate Call	Y	Y	Y	Existing capability
Insert DTMF	**N**	**Y**	**Y**	**capability introduced in Equinox**
Mute/Unmute Call	Y	Y	Y	Existing capability
Media Device Listings	N	Y	Y	capability introduced in Equinox

Why JSON?

The External API uses JSON to exchange data between the External Application and the External Application Interface API. JSON offers the following advantages:

- JSON is simple, open, and interoperable.
- Data is defined to allow generic tools to manipulate data.
- JSON data is (almost) human readable.

A key aspect of JSON leveraged through the API is that applications consuming the JSON can easily disregard fields that it is unfamiliar, and the External API can do the same. This allows a large degree of flexibility between versions of External Applications and the External API, as long as JSON fields are never removed from the External API. This is critical to External API versioning, which is described below.

Named Pipe API Model

The External API is an Avaya proprietary API that provides external applications with a basic interface to influence call handling of the Client SDK. The interface uses JSON encoded messages over a platform-provided named pipe, with the named pipe connecting the external application to the Client SDK. The External API provided by the Client SDK is intended to support two different types of applications, simple click to call and basic call control. Basic application interoperability allows External Applications to invoke Make Call requests on the Client SDK.

Pipe Discovery

When the API is enabled, the Client SDK establishes a single public named pipe. The named pipe is strongly named, to allow applications to find the appropriate pipe. This allows multiple users to use the same workstation simultaneously, and allows multiple external applications to direct requests to the intended Client SDK instance. In the case where multiple Client SDK Applications are executing as the same user, the first Client SDK Application to acquire the named pipe "wins". If the pipe name is already in use when the Client SDK attempts to create the named pipe, the operation will fail, and the External Application API will not be available for the Client SDK instance. An alternative pipe name is not available.

Windows

```
PipeName = \\.\pipe\AvayaCSDK-%username%

i.e. \\.\\pipe\AvayaCSDK-bob
```

The name of the Windows pipe can be confirmed using pipelist.exe . Look for a pipe name starting with "AvayaCSDK".

MacOS X

```
Unix Domain Socket Name = <User Home Dir>/Application/ Support.com.avaya-%app
name%/AvayaCSDK-%platform user name%
```

```
i.e. /Users/joeuser/Library/Application Support/com.avaya/Avaya-
Workplace/AvayaCSDK-bob
```

The name of the MacOS Unix Domain Socket can be confirmed using "netstat -a". Look for a socket name starting with "AvayaCSDK".

Application Sandboxing for MacOS

Not supported in this release.

Basic Interworking Model

Basic interworking occurs on the public pipe and allows external applications to invoke two operations:

- **create** - Enables external applications to request the Client SDK to create a call on behalf of the Client SDK user.
- **register** - Enables external applications to request a dedicated pipe in order to register for call UpdateEvents. It also provides a dedicated pipe for the external application to send or make advanced requests.

Each request is acknowledged with a response. The "create" response will contain the call object containing the current state of the call attributes. If the External Application has not registered for events, no subsequent updates will be provided.

Enhanced Interworking Model

When the External Application registers for call UpdateEvents, mid-call operations are permitted and UpdateEvents are sent when a call attribute changes. Each External API call operation request has an associated response, and an operation response will be returned for each request received. The operation response will always contain the current state of the call object, with subsequent call attribute changes being provided to the External Application through the UpdateEvent.

As the External API is not the only mechanism available to control a call, External Applications must be able to handle UpdateEvents for call operations invoked through another mechanism, such as the UC client user interface. The External Application can track changes in the call that are made externally (through the Avaya Client SDK application) because call attribute changes are sent through an UpdateEvent without a call operation response.

Similarly, the External API can have multiple External Applications connected simultaneously. External Applications only receive operation responses for requests invoked by the External Application that sent the request, and call UpdateEvents will be received for all calls, irrespective of how the call was created.

Security Model

Authentication

All named pipes are created by the Avaya Client SDK. This allows the Client SDK to control security permissions for the named pipe.

All pipes should be created so that only applications running as the platform user are able to connect to the pipe.

The platform (OS) is responsible for enforcing authentication, as defined by the application.

Networking

The external Application and the Avaya Client SDK processes must exist on the same workstation. The External API interface is not available to remote network applications.

Named Pipe Connectivity Model

To allow multiple external applications to send requests on the *public* pipe, the external application shall disconnect from the pipe 200ms after the request is made. If a request is not received within 200ms of connecting, the pipe will be closed by the server.

The pipe name will be consistent for each platform user across Avaya Client SDK restarts.

The pipe is created with visibility local to the workstation. Network access is not permitted.

The named pipe can only be connected by a process with the same login identity as the Client SDK process owner.

Applications that register for call events can remain connected to the named pipe indefinitely.

Protocol Framing

External API requests and events are JSON-encoded over the pipe. Each message (request or event) is terminated with a NULL byte to act as a message delimiter. As JSON messages are syntactically strong, it is possible for either side of the pipe to be aware when a complete request is received. When the Avaya Client SDK receives a complete request, it will act on the request. If the JSON request is not properly terminated within 200ms, the Client SDK will disconnect the external application from the pipe, and purge the pipe buffer. This allows the pipe to be reset for both parties.

If the external application detects a malformed response or event, it will disconnect from the pipe and reconnect.

NullByte = \0

Multiple requests can be sent over the main pipe or the private pipe without waiting for the associated response, as long as the following is true;

- Each request is properly encoded, and terminated with a NullByte.
- The transaction ID for each request is unique.

The External API parser expects complete External API requests to be written at once, not byte by byte. The parser expects the NullByte to be encoded immediately following the closing brace of the JSON message.

Transaction Identifiers (TransactionIds)

TransactionIds are used to correlate responses with the intended request. Each request shall have its own transactionId, and the transactionId should be unique across time and space for all messages sent by the external application. The transaction IDs need not to be numeric, not monotonically increasing.

I.E.

a1-47.135.10.14, a2-47.135.10.14,a23-47.135.10.14

41.25.135.158-9bca, 41.25.135.158-9bcb,41.25.135.158-9bcc

Denial of Service

The Avaya Client SDK limits the number of private named pipes to 3.

Rate Limiting

Rate limiting is not implemented.

Application Identification

An Application-ID is used for correlation between external applications and the Avaya Client SDK. Register requests directed to the Client SDK must have the Application-ID populated. Register requests without an ApplicationId will be silently discarded. Two external applications cannot register with the same ApplicationId, and the later registration will supercede the original application registration.

Disabling the External Application Interface

The External Application Interface may be disabled by Avaya Client SDK. When the External Application interface is disabled, the public pipe will not be created.

Internationalization

Unicode support

UTF-8 encoding shall be used for data exchange.

Backward and Forward Compatibility

Versions

The External Application API is versioned independently of the Avaya Client SDK version to provide external applications with a stable set of functionality. The External Application API is versioned with a major version and a minor version. A critical application requirement is to be able to safely parse expected and unexpected fields. This allows the External Application API to add incremental and supporting data to the responses and events without fear of breaking the external application. The External Application API will behave similarly. The major version is updated for the following reasons:

- When a change to the API is introduced that is not backward compatible with the previous version.
    - Example: A new parameter is introduced, or a mandatory parameter becomes optional.
- When a portion of the API is deprecated.

The minor version of the API can change for the following reasons:

- A new capability is added to the API. This is a minor version update because new functionality will not impact existing applications.
- A new optional parameter is added to an API request, response, or event.

The External Application API implements the following version strategy, which is based on JSON's extensibility concepts.

If a new optional parameter is added to a media type, the minor version uses a dot increment.

Example: v1 becomes .v1.1, and v1.1 would become 1.2. v1.0 would become 1.1.

If a new mandatory parameter is added to a media type, the major version is incremented by 1 for *all media types.* .

Example: v1 becomes v2. v9 becomes v10.

If a new capability is added to the External Application API, the new media types associated with the capability would be versioned as v1.

If the external application registers for events, the External Application API will provide Event media types based on the version of the RegisterRequest. The versions of the following media types will be versioned at the same level. If the media type for one of the following changes, the versioning for all the media types must be updated.

Application Versioning table for Register request, Register Response, and Events

Application Sends	API Responds
vnd.avaya.clientresources.call.RegisterRequest.v1	**vnd.avaya.clientresources.call.RegisterResponse.v1** **vnd.avaya.clientresources.call.RegisterResponse.v1.1**
vnd.avaya.clientresources.call.RegisterRequest.v3	**vnd.avaya.clientresources.call.RegisterResponse.v3** **vnd.avaya.clientresources.call.RegisterResponse.v3.3**

The External API will use the latest available minor version of the API based on the version number of the register request provided by the External Application.

External Application API Backward Compatibility

External Application API backward compatibility is the ability for the API to handle requests from an older version of the API used by the external application. The External Application API will support the current major version of the media types, and the previous version of the External Application API. If the current version of the External Application API is v2, the External Application API implemention will be capable of returning v1 responses. If the current version of the External Application API is v3, the External API will be capable of returning v2 responses, but will not be capable of supporting v1 responses.

If a External Application API change forces the major version of the API to be updated, all supported media types versions will be updated to the next major version number.

This will occur when the Avaya Client SDK External Application API version is newer than the version of the External Application API used by the external application.

Example: Avaya Workplace is newer than the headset application.

External Application API Forward Compatibility

External Application API forward capability is the ability for the external application to send External Application API versions newer than the External Application API supports. In this case, the External Application API will reply to the CreateCall or Register request with an Error, Unsupported Media Type. When this occurs, the external application will reduce the major version of the API by 1, and attempt the request again. This process may be repeated until the version becomes v1, or until the version of the External Application API is not supported by the external application. It is up the external application vendor to decide how many legacy versions of the External Application API to support.

This will occur when the external application version is newer than the version of the External Application API used by the Avaya External Application API

Example: Headset Application is newer than the Avaya Workplace.

Current Version

By default, all requests receive the v1 version of the External Application API. The version of the API is explicitly captured in the messages.

```
vnd.avaya.clientservices.call.v1.1
```

Versioning will be supported at an External Application API level, and all media types will use the same version. The version of an event sent to the application will be based on the major version of register request received. The External Application API will always provide the most recent minor version of the External Application API.

API Failures will return the following Accept header in the failure response.

```
vnd.avaya.clientservices.Error.v1.1
```

API Definition

Media Types

Applications shall always be prepared to receive a JSON Error response.

Errors

vnd.avaya.clientresources.Error.v1.1

Attribute	Type	Optional	Description
displayMessage	string	N	A message that contains information that can be displayed to an end user.
errorMessage	string	N	A message that contains information necessary for a developer to correct the problem.
errorCode	Enum	N	A code associated with a unique error condition on the server.
transactionId	string	Y	The request associated with the error.

Error (JSON)
{ "vnd.avaya.clientresources.Error.v1.1" : { "displayMessage" : " displayMessage text ", "errorMessage" : " error Message text ", "errorCode" : " error Code text ", "transactionId": "1" } } \0

Failure and Error Codes

Error codes are similar to HTTP, to facilitate ease of use and understanding.

Code		Description
400	Bad Request	Could not parse request.
404	Not Found	Call/Resource not found.
406	Not Acceptable	Missing mandatory field.
408	Timeout	Timeout processing the request.
409	Conflict	applicationid has already registered on this pipe.
410	Gone	Desired entity (call) does not exist
415	UnsupportedMediaType	Improper media type.
500	Client SDK Error	Error processing request.
503	Service Unavailable	Client SDK not ready to accept requests.
497	Media Preserved	Call is in the media preserved state. The only supported operation is Terminate.

Pipe Management

Register

vnd.avaya.clientresources.RegisterRequest.v1.1

vnd.avaya.clientresources.RegisterResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Register Request (JSON)

{
"vnd.avaya.clientresources.RegisterRequest.v1.1" : {
"applicationId": "app",
"transactionId: "1"
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.RegisterResponse.v1.1" : {
"transactionId": "1"
}
}\0

Unregister

vnd.avaya.clientresources.UnregisterRequest.v1.1

vnd.avaya.clientresources.Error.v1.1

Unregister Request (JSON)

{
"vnd.avaya.clientresources.UnregisterRequest.v1.1" : {
"transactionId": "1"
}
 	
}\0

An unregister response is not required. The Client SDK will immediately close the connection as implicit acknowledgement to the request.

Pipe Management Events

DisconnectRequest

vnd.avaya.clientresources.DisconnectRequest.v1

DisconnectRequest (JSON)

<pre>{
  "vnd.avaya.clientresources.DisconnectRequest.v1.1" {
  "transactionId": "1"
  }
}\0</pre>

A Disconnect response is not required. When a DisconnectRequest is received, the recipient can immediately close the pipe as acceptance. If the recipient does not close the pipe, the DisconnectRequest sender will close the pipe after 200ms.

Calls

Call Resource

The following Call attributes are included in all call related responses and call events. Fields can be left empty or blank intentionally by the External Application API.

vnd.avaya.clientresources.Call.v1.1

Attribute	Values	Relationship	Description
CallId	string	mandatory	Unique call identifier.
remote	true,false	mandatory	Call is on another device (MDA, Bridged lines, EC 500). The call is not being handled by the CSDK Application. Media is not running locally. Headset call control applications should not act on remote calls.
remotePartyName	string	mandatory	Name of the remote participant. The value can be empty or blank.
remotePartyNumber	string	mandatory	Remote Party CLID. The value can be empty or blank.
privacy	true,false	mandatory	Privacy set for Remote Party. The value can be empty or blank.

Attribute	Values	Relationship	Description
Subject	string	mandatory	Call subject. The value can be empty or blank.

callState	alerting transferred ended ending established failed far-end renegotiating Held Holding idle ignored initiating remote alerting renegotiating transferring unholding video updating	mandatory	The call state, which represents: `public enum CallState values.` alerting- Call is alerting locally (incoming call). transferred - The call is being transferred by a remote party on the call. ended - Call ended (by far-end, or end request has been responded to by the remote party). ending - Request to end the call has been sent. established - Call has been established and is active (not held). failed- Failed. far-end renegotiating- Call renegotiating (requested by far-end). held- Held. holding - Holding. idle - Uninitialized. ignored - Call is ignored. initiating - Call initiated (outgoing only). remote alerting- Call is alerting remotely (outgoing call). renegotiating - Call is renegotiating (requested by us). transferring -Transfer initiated by the local user (outgoing). unholding - Retrieving .

Attribute	Values	Relationship	Description
			video updating - Video is being added or removed from the call.
muted	true,false	mandatory	Audio on or off.
videoPossible	true,false	mandatory	Client informs the API if video escalation is possible. Video license is acquired. Network applicable for video. If a video device is not available, this value will not be altered.
videoDirection	Inactive, Send_Receive, Send_Only, Receive_Only	mandatory	The direction of the video.
audioDirection	Remote, Inactive, Send_Receive, Send_Only, Receive_Only	mandatory	The direction of the audio. If audio is directed to a remote device, the direction will be remote.
transactionId	TransactionId of the request.	optional	transactionID is only provided in response messages, and it not provided in events.

Call Operations

Call Operations Summary table

Operation	MediaType	Description
query	vnd.avaya.clientresources.call.GetCallsRequest.v1.1	Discover calls on the client.
create	vnd.avaya.clientresources.call.CreateRequest.v1.1	Initiate call.
terminate	vnd.avaya.clientresources.call. TerminateRequest .v1.1	End call. Terminates the existing call, irrespective of call state.
accept	vnd.avaya.clientresources.call.AcceptRequest.v1.1	Answer the call. Possible state for incoming alerting state.
ignore	vnd.avaya.clientresources.call.IgnoreRequest.v1.1	Ignore the call. Possible state for incoming alerting state. Invoking the ignore operation, the client's ringer will be muted, and the incoming call notification will be suppressed. The call state does not change, and the call can still be answered.
hold	vnd.avaya.clientresources.call.HoldRequest.v1.1	Hold call.
mute	vnd.avaya.clientresources.call.MuteRequest.v1.1	Mute audio. Local mute operation supported in computer mode, and shared control mode. Network mute, for conferencing applications, is not supported. Mute in other phone mode (telecommuter mode) not supported.
video	vnd.avaya.clientresources.call.VideoRequest.v1.1	Add video to the call.
dtmf	vnd.avaya.clientresources.call.DTMFRequest.v1.1	Insert DTMF digits during the call.

Call Operations and Call State Validity Table

Call State		Query	Terminate	Accept	Ignore	Hold	Mute	Video	DTMF	
1	alerting	√	√	√	√	X	√	X	X	Call is alerting locally (incoming call).
2	initiating	√	√	X	X	X	√	X	√	Call initiated (outgoing only).
3	established	√	√	X	X	√	√	√	√	Call has been established and is active (not held).
4	held	√	√	X	X	√	√	X	X	Call has been established but is not active (held).
5	holding	√	√	X	X	X	√	X	X	Call is in process of being held by us.
6	unholding	√	√	X	X	X	√	X	X	Unholding or retrieving.
7	failed	√	√	X	X	X	√	X	X	call Failed.
8	idle	√	√	X	X	X	√	√	X	Call Uninitialized.
9	remote alerting	√	√	X	X	X	√	X	√	Call is alerting remotely (outgoing call).
10	ignored	√	X	√	√	X	√	X	X	Incoming s ession is ignored.

11	renegotiating	√	√	X	X	X	√	X	√	Call renegotiating (requested by us).
12	far-end renegotiating	√	√	X	X	X	√	X	√	Call renegotiating (requested by them).
13	transferring	√	√	X	X	X	√	X	X	Transfer initiated by us (outgoing).
14	transferred	√	√	X	X	X	√	X	X	Being transferred by them (incoming).
15	ending	√	X	X	X	X	√	X	X	Request to end the session has been sent.
16	Ended	√	X	X	X	X	√	X	X	Call ended (by far-end, or end request has been responded to by the remote party).
17	video updating	√	√	X	X	X	√	√	√	

ing") are ephemeral states, and the application should expect a subsequent state transition to follow. While a call is in an ephemeral state, it is not possible for the external application to invoke signalling operations.

The operation response JSON message will always contain the state of the session, which may be unrelated to the operation requested.

Call State Transitions

Outgoing Call State Transition Table

Pre-Op Call State	Operation	Post Op Call State
idle	<any non-create operation>	idle
idle	Create	initiating
initiating	Terminate	ending/ended
initiating	<any other operation>	enitiating
remote alerting	Terminate	ending/ended
remote alerting	DTMF	remote alerting
remote alerting	<any other operation>	remote alerting

Incoming Call State Transition Table

Pre-Op Call State	Operation	Post Op Call State
alerting	No Operation Timeout	terminated, null
alerting	Ignore	ignored
alerting	Accepted	established
alerting	Terminate	alerting
ignored	Ignore	ignored
ignored	Accepted	established
ignored	Terminate	ignored

Established Call State Transition Table

Pre-Op Call State	Operation	Post Op Call State
established	Terminate	ending/ended
established	Join (MDA, from UC Client)	established
established	DTMF	established
established	Hold	holding/held
transferring	Terminate	ending/ended
transferring	any operation except Terminate	transferring

Pre-Op Call State	Operation	Post Op Call State
transferred	Terminate	ending/ended
transferred	any operation except Terminate	transferred
renegotiating	Terminate	ending/ended
renegotiating	any operation except Terminate	renegotiating
far-end renegotiating	Terminate	ending/ended
far-end renegotiating	any operation except Terminate	far-end renegotiating

Held Call State Transition Table

Pre-Op Call State	Operation	Post Op Call State
established	Hold	holding/held
held	Unhold	unholding/established
held	Hold	held
held/holding/unholding	Terminate	ending/ended
holding/unholding	<any other operation>	holding/unholding/held/established

Failed/Ending/Ended Call State Transition Table

Pre-Op Call State	Operation	Post Op Call State
ending	<any operation>	ending/ended
ended	<any operation>	ended
failed	<any operation>	failed

 Applications registering for call events when calls are in progress may not receive events to support the existing state. Call Events are only guaranteed to be sent for call state transitions.

Call Messages

 External Application API documentation below highlights the important elements of the API, given the context of the request. All call attributes will be returned for call-related requests.

Get Calls

vnd.avaya.clientresources.call.GetCallsRequest.v1.1

vnd.avaya.clientresources.call.GetCallsResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

get Calls Request (JSON)

{
 	
" **vnd.avaya.clientresources.call.GetCallsRequest** .v1.1": {
 	
"transactionId": "1"
 	
}
} \0

Success Response -with single call (JSON)

{
"vnd.avaya.clientresources.call.GetCallsResponse.v1.1" : {
"vnd.avaya.clientresources.Call.v1.1" : {
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
 	
"callId": "xyz123",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
 	
},
 	
"transactionId": "1"
 	
}
 	
}\0

Success Response -with calls (JSON)

get Calls Request (JSON)

```
{
"vnd.avaya.clientresources.call.GetCallsResponse.v1.1" : {
"vnd.avaya.clientresources.Call.v1.1" : [ {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "xyz123",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"

},

{

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",

  "callID": "abc124",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}] ,

"transactionId": "1"

}

}\0
```

Success Response - without calls (JSON)

```
{
"vnd.avaya.clientresources.call.GetCallsResponse.v1.1" : {

"transactionId": "1"

}

}\0
```

get Calls Request (JSON)
Errors Responses (JSON)

{
"vnd.avaya.clientresources.Error.v1.1" : {
"displayMessage" : " displayMessage text ",
"errorMessage" : " error Message text ",
"errorCode" : " error Code text ",
"transactionId": "1"
}
}\0

GetCalls will return, at most, all active calls. The External API shall not constrain the number of active calls.

Mute Call

vnd.avaya.clientresources.call.MuteRequest.v1.1

vnd.avaya.clientresources.call.MuteResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Mute Request Payload (JSON)

{
  "vnd.avaya.clientresources.call.MuteRequest.v1.1": {
"callId": "xxx",
"muted": "true",
"transactionId": "1"
}
}\0

Success Response (JSON)

{
  "vnd.avaya.clientresources.call.MuteResponse.v1.1": {
 	
"vnd.avaya.clientresources.Call.v1.1" : {
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
  "callId": "abc124",
 	
"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}
 	
"transactionId": "1"
}
} \0

Failure Response

Unmute Call

vnd.avaya.clientresources.call.MuteRequest.v1.1

vnd.avaya.clientresources.call.MuteResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Unmute Request Payload (JSON)

```
{
  "vnd.avaya.clientresources.call.MuteRequest.v1.1": {
"callId": "xxx",
"muted": "false",
"transactionID": "1"
}
}\0
```

Success Response (JSON)

```
{
  "vnd.avaya.clientresources.call.MuteResponse.v1.1": {

"vnd.avaya.clientresources.Call.v1.1" : {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

  "callId": "abc124",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},

"transactionId": "1",

}
} \0
```

Failure Response

Hold Call

vnd.avaya.clientresources.call.HoldRequest.v1.1

vnd.avaya.clientresources.call.HoldResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Hold Call Request Payload (JSON)

{
" **vnd.avaya.clientresources.call.HoldRequest.v1.1** ": {
"callId": "xxx",
"held": "true",
"transactionId": "1"
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.HoldResponse.v1.1": {

"vnd.avaya.clientresources.Call.v1.1" : {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"audioDirection" : "Remote,Inactive",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},

"transactionId": "1",
},
} \0

Failure Response

Retrieve Call (unHold Call)

vnd.avaya.clientresources.call.HoldRequest.v1.1

vnd.avaya.clientresources.call.HoldResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Retrieve Call Request Payload (JSON)

{
"vnd.avaya.clientresources.call.HoldRequest.v1.1": {
"callId": "xxx",
"held": "false",
"transactionId": "1"
}
} \0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.HoldResponse.v1.1": {
 	
"vnd.avaya.clientresources.Call.v1.1" : {
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
"callId": "abc124",
 	
"audioDirection" : "Remote, Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},
 	
"transactionId": "1"
}
} \0

Failure Response

Terminate Call

vnd.avaya.clientresources.call. TerminateRequest .v1.1

vnd.avaya.clientresources.call. TerminateResponse .v1.1

vnd.avaya.clientresources.Error.v1.1

Terminate Call Request Payload (JSON)

```
{

"vnd.avaya.clientresources.call. TerminateRequest .v1.1": {

"callId": "xxx",

"transactionId": "1"

}

}\0
```

Success Response (JSON)

```
{
  "vnd.avaya.clientresources.call. Terminate Response.v1.1": {

"vnd.avaya.clientresources.Call.v1.1" : {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

  "callId": "abc124",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} ,

"transactionId": "1"

}
} \0
```

Failure Response

Answer Call (Accept Call)

vnd.avaya.clientresources.call.AcceptRequest.v1.1

vnd.avaya.clientresources.call.AcceptResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Accept Call Request Payload (JSON)

{
 	
"vnd.avaya.clientresources.call.AcceptRequest.v1.1": {
 	
"callId": "xxx",
 	
"transactionId": "1"
 	
}
 	
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.A cceptResponse.v1.1": {
 	
"vnd.avaya.clientresources.Call.v1.1" : {
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
"callId": "abc124",
 	
"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}
 	
,
"transactionId": "1"
}
} \0

Failure Response

Ignore Call

vnd.avaya.clientresources.call.IgnoreRequest.v1.1

vnd.avaya.clientresources.call.IgnoreResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Ignore Call Request Payload (JSON)

{
"vnd.avaya.clientresources.call.IgnoreCall.v1.1": {
"callId": "xxx",
"transactionId": "1"
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.IgnoreResponse.v1.1": {

"vnd.avaya.clientresources.Call.v1.1" : {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"audioDirection" : "remote,inactive",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},

"transactionId": "1"
}
} \0

Failure Response

CreateCall (MakeCall)

vnd.avaya.clientresources.call.CreateRequest.v1.1

vnd.avaya.clientresources.call.CreateResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Request Payload (JSON)

{
"vnd.avaya.clientresources.call. CreateRequest.v1.1": {
"remotePartyNumber" : "Remote Party Number Value" ,
"video" : "true" ,
"subject" : "string " ,
"conferencePasscode" : "0-9,#,*" ,
"conferenceId" : "0-9",
"lineAppearanceOwner" : "a-z, 0-9",
"lineAppearanceId" : "int",
"transactionId": "1"
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.CreateResponse.v1.1": {

"vnd.avaya.clientresources.Call.v1.1" : {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},

"transactionId": "1"

}
} \0

Failure Response

DTMF

vnd.avaya.clientresources.call.DTMFRequest.v1.1

vnd.avaya.clientresources.call.DTMFResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

DTMF Request Payload (JSON)

{
"vnd.avaya.clientresources.call.DTMFRequest.v1.1": {
"dtmfstring": "0-9,#,*",
"callId" : "xxx",
"transactionId": "1"
}
}\0

Response (JSON)

{
"vnd.avaya.clientresources.call.DTMFResponse.v1.1": {

"vnd.avaya.clientresources.Call.v1.1" : {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"audioDirection" : "send_receive",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
},

"transactionId": "1"
}
} \0

Failure Response

Add videoCall (Escalate)

vnd.avaya.clientresources.call.videoRequest.v1.1

vnd.avaya.clientresources.call.videoResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Add video Request Payload (JSON)

{
"vnd.avaya.clientresources.call.VideoRequest.v1.1": {
"video": "true" ,
"callID" : "xxx",
"transactionId": "1"
}
} \0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.VideoResponse.v1.1": {

"vnd.avaya.clientresources.Call.v1.1" : {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"audioDirection" : "remote,send_receive",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
} ,

"transactionId": "1"
}
} \0

Failure Response

Remove video (Deescalate)

vnd.avaya.clientresources.call.videoRequest.v1.1

vnd.avaya.clientresources.call.videoResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Remove Video Request Payload (JSON)

{
"vnd.avaya.clientresources.call.VideoRequest.v1.1": {
"video": "false" ,
"callID" : "xxx",
"transactionId": "1"
}
}\0

Success Response (JSON)

Remove Video Request Payload (JSON)

{
"vnd.avaya.clientresources.call.VideoResponse.v1.1": {
 	
"vnd.avaya.clientresources.Call.v1.1" : {
 	
"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",
 	
"callId": "abc124",
 	
"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}
 	
,
"transactionId": "1"
}
} \0

Failure Response

Block Camera

vnd.avaya.clientresources.call.BlockCameraRequest.v1.1

vnd.avaya.clientresources.call.BlockCameraResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Block Camera Request Payload (JSON)

{
" **vnd.avaya.clientresources.call.BlockCameraRequest.v1.1** ": {
"blockcamera": "true",
"callId": "xxx",

"transactionId": "1"
}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.BlockCameraResponse.v1.1": {

"vnd.avaya.clientresources.Call.v1.1" : {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}

"transactionId": "1"
}
} \0

Failure Response

UnBlock Camera

vnd.avaya.clientresources.call.BlockCameraRequest.v1.1

vnd.avaya.clientresources.call.BlockCameraResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Unblock Camera Request Payload (JSON)

{
"vnd.avaya.clientresources.call.BlockCameraRequest.v1.1": {
"blockcamera": "false",
"callId": "xxx",

"transactionId": "1"

}
}\0

Success Response (JSON)

{
"vnd.avaya.clientresources.call.BlockCameraResponse.v1.1": {

"vnd.avaya.clientresources.Call.v1.1" : {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

"callId": "abc124",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}

,
"transactionId": "1"
},
}\0

Failure Response

Call Events

vnd.avaya.clientresources.call.UpdatedEvent.v1.1

UpdatedEvent (JSON)

```
{
"vnd.avaya.clientresources.call.UpdatedEvent.v1.1" :{
  "transactionId": "1",

"vnd.avaya.clientresources.Call.v1.1" : {

"remotePartyName": "Remote Party Name Value",
"remotePartyNumber": "Remote Party Number Value",
"callState": "alerting/originating/established/held/failed",

  "callId": "abc124",

"audioDirection" : "Remote, Inactive,Send_Receive,Send_Only, Receive_Only",
"muted": "true/false",
"videoDirection": "inactive/receive_only/send_only/send_receive",
"videoPossible": "true/false"
}

}
}\0
```

Call Events will be reported when the call is local to the device (Computer mode), the call media is routed to the desk phone (Shared Control), and when call media is routed to the PSTN (Telecommuter or Other Phone mode). Call Events are also reported when the call is acted on by a remote device, in a MDA (Multiple Device Access, in Avaya Aura®) or Twinning (IP Office), or in a bridged line appearance call (BLA). Calls being managed by a remote device are identified with the remote attribute.

Media Devices

The media device API provides the external application with the current audio and video device that is selected for calls. The external application cannot change the active devices. Active devices can only be managed with the Client SDK Application.

Media Device

Resource	Description
/Resources/MediaDevices/audio	Specifies active audio devices.

Media Device Resource

Attribute	Description
recordingDevice	Specifies the recording device.
playbackDevice	Specifies the playback device. This does not apply to video.
deviceName	Specifies the name of the device.

Media Device Operations

Operation	URI	Description
query audio devices	/Resources/MediaDevices/audio	Returns the active audio input or output device.

Audio Device

vnd.avaya.clientresources.device.ActiveAudioDeviceRequest.v1.1

vnd.avaya.clientresources.device.ActiveAudioDeviceResponse.v1.1

vnd.avaya.clientresources.Error.v1.1

Audio Request Payload (JSON)

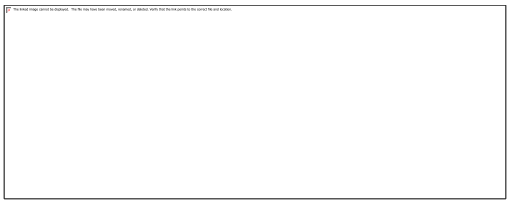
{
"vnd.avaya.clientresources.call. ActiveAudioDeviceRequest.v1.1": {
 	
"transactionId": "14"
 	
}
 	
}\0

Successful Response (JSON)

{
"vnd.avaya.clientresources.device.ActiveAudioDeviceResponse.v1.1": {
"RecordingDevice.v1.1" : {
"deviceName" : "string"
},
"PlaybackDevice.v1.1" : {
"deviceName" : "string"
}
"transactionId": "14"
}
}\0

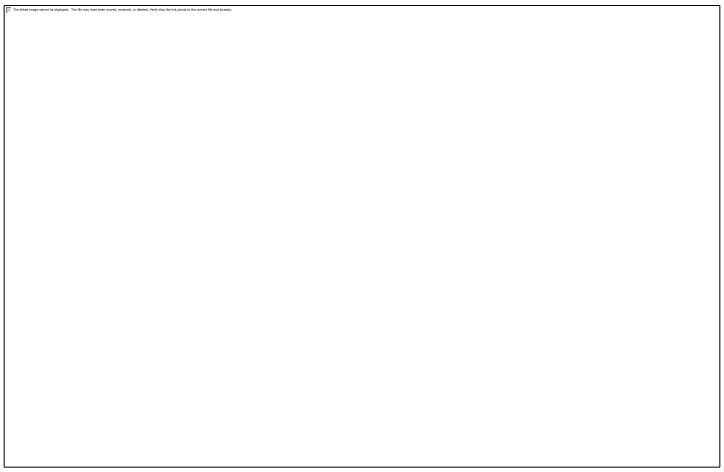
Call Flows

Legend



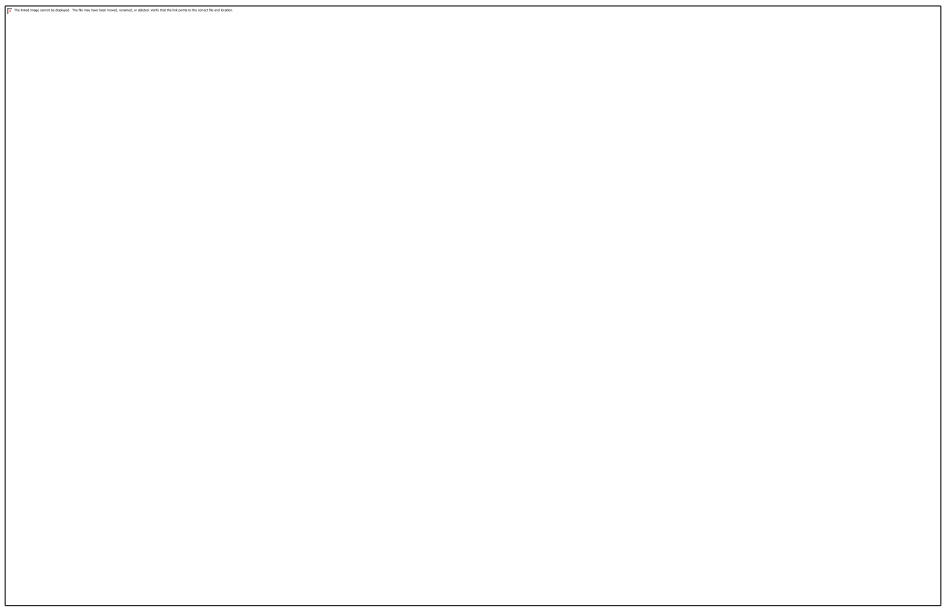
API Control Call Flows

Client SDK Initialization



Register

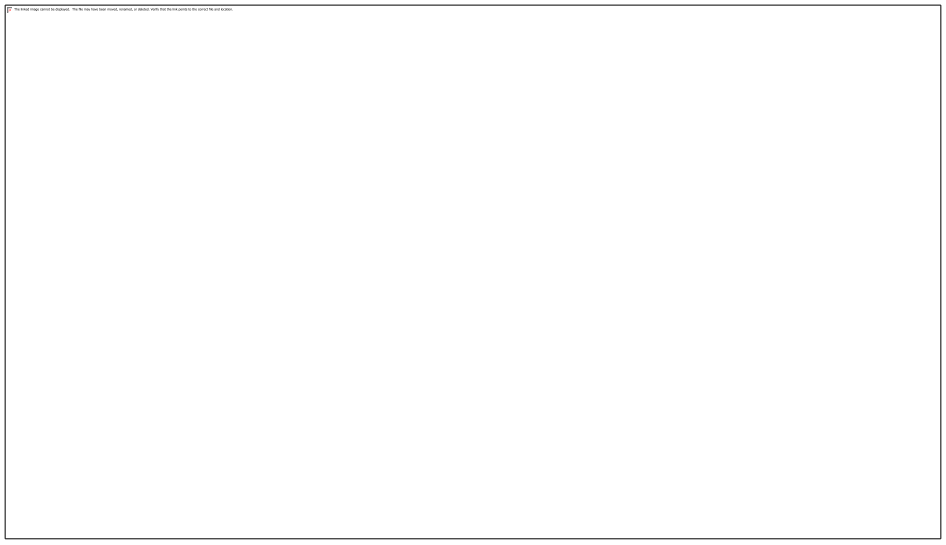
This is sent by the external application when it wants to provide a rich call control experience.



When the application registers again with a different applicationId, it is considered an application error. The registration is rejected by the Client SDK if the applicationId is different than the applicationId used in the initial Register request. The pipe is not closed and the original applicationId remains valid.

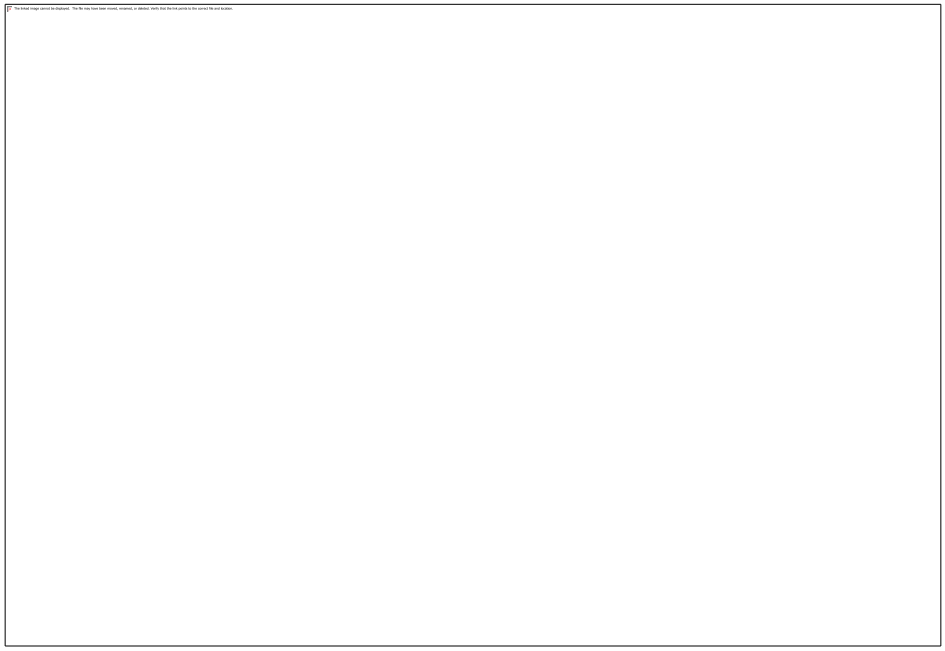
Unregister

This is sent by external application when it no longer wants the External Pipe. For example, this can occur during an application shut down or when the work station is in Sleep mode.



DisconnectRequest

This is sent by the Client SDK when it no longer wants the External Pipe. For example, this can occur during an application shut down or when the work station is in Sleep mode.



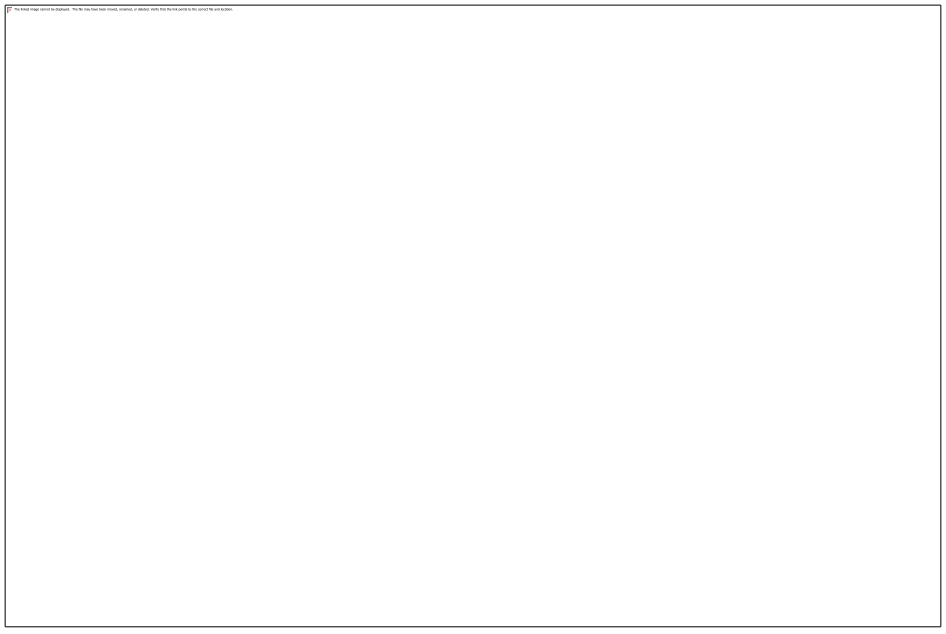
Call Handling

The following sections show the possible call flows for each operation, but the actual combination of responses and CallUpdatedEvents depend on the state of the call object at the time the response is fired. Call state transitions vary depending on the remote endpoint, network latency, and External Application API internal implementation. Examples include the following:

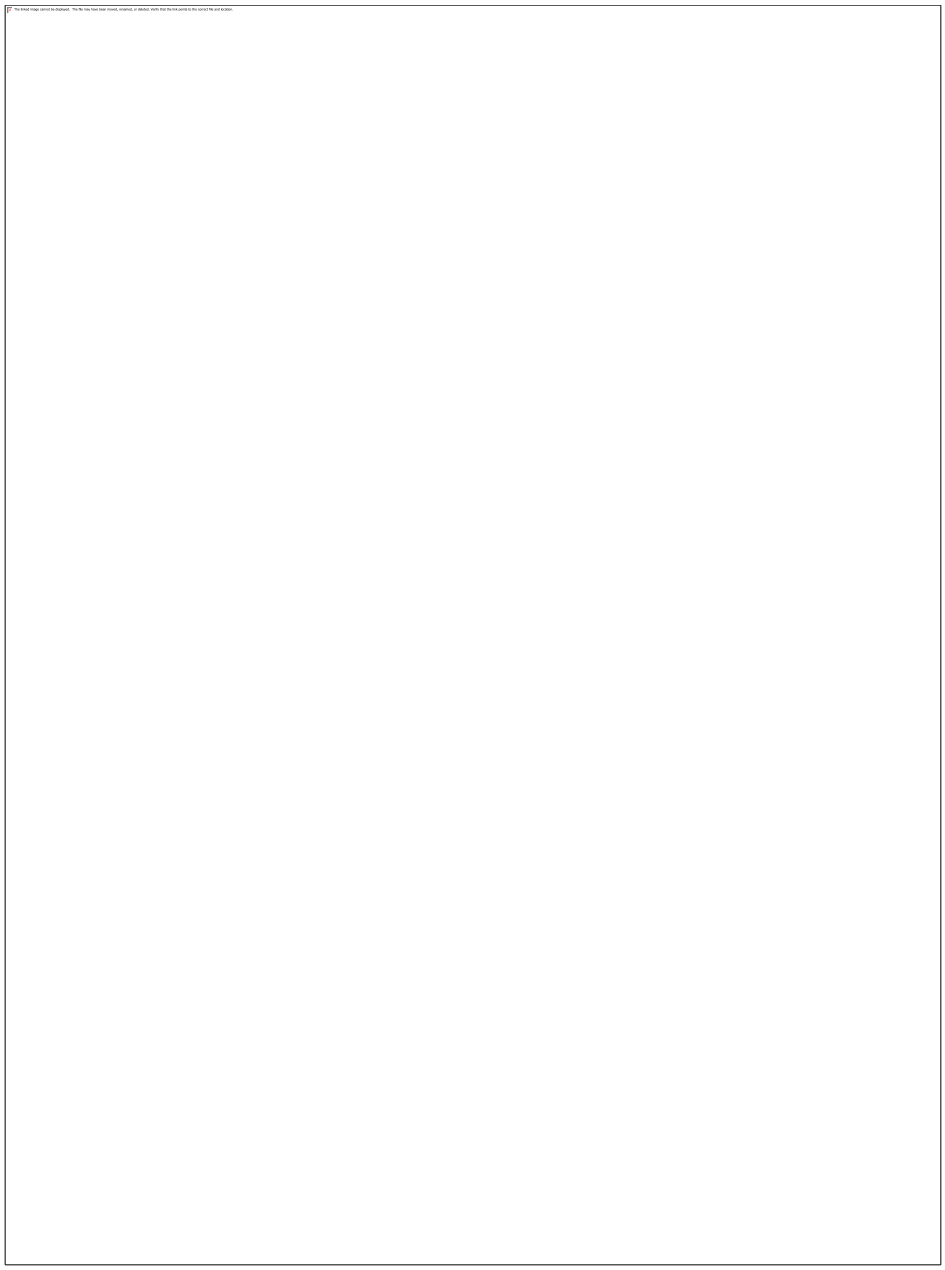
When creating a call to a conference server that answers the call immediately, the Remote_Alerting call state transition might be skipped and only Established is reported.

Holding and Unholding might not be reported and the more stable Held or Established call state is returned.

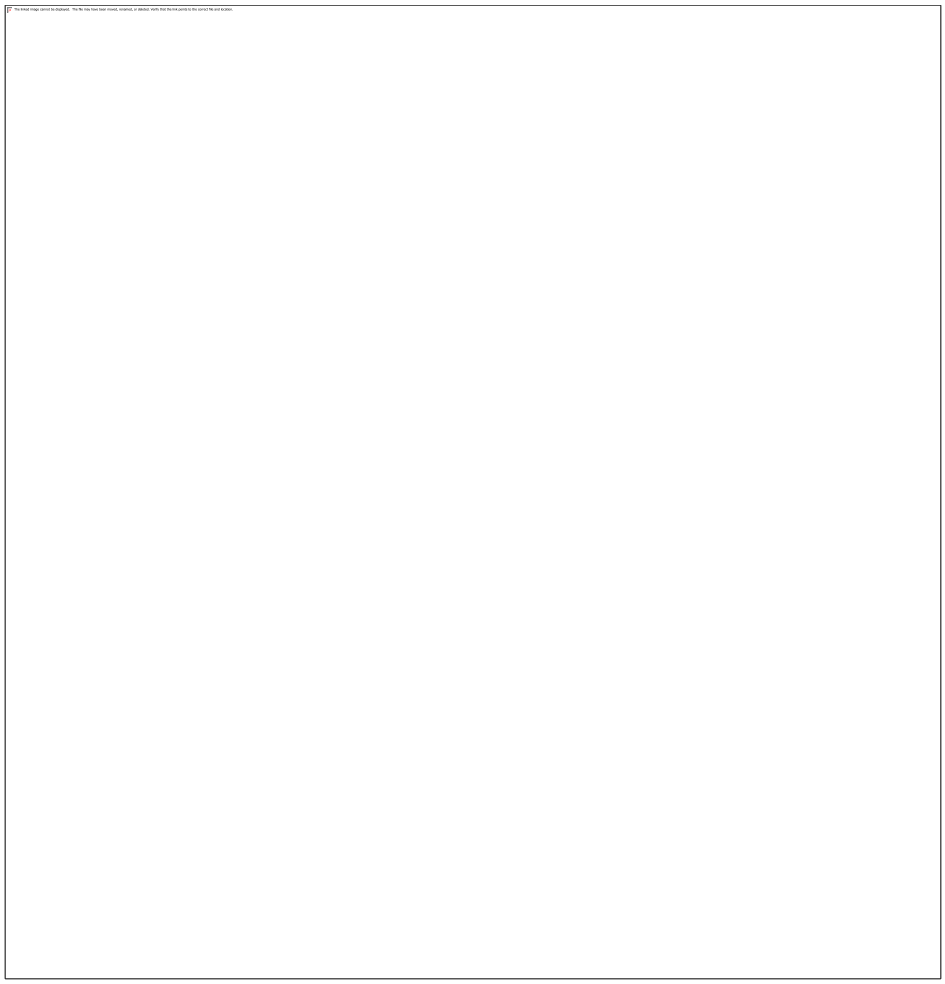
Make Call - Named Pipe



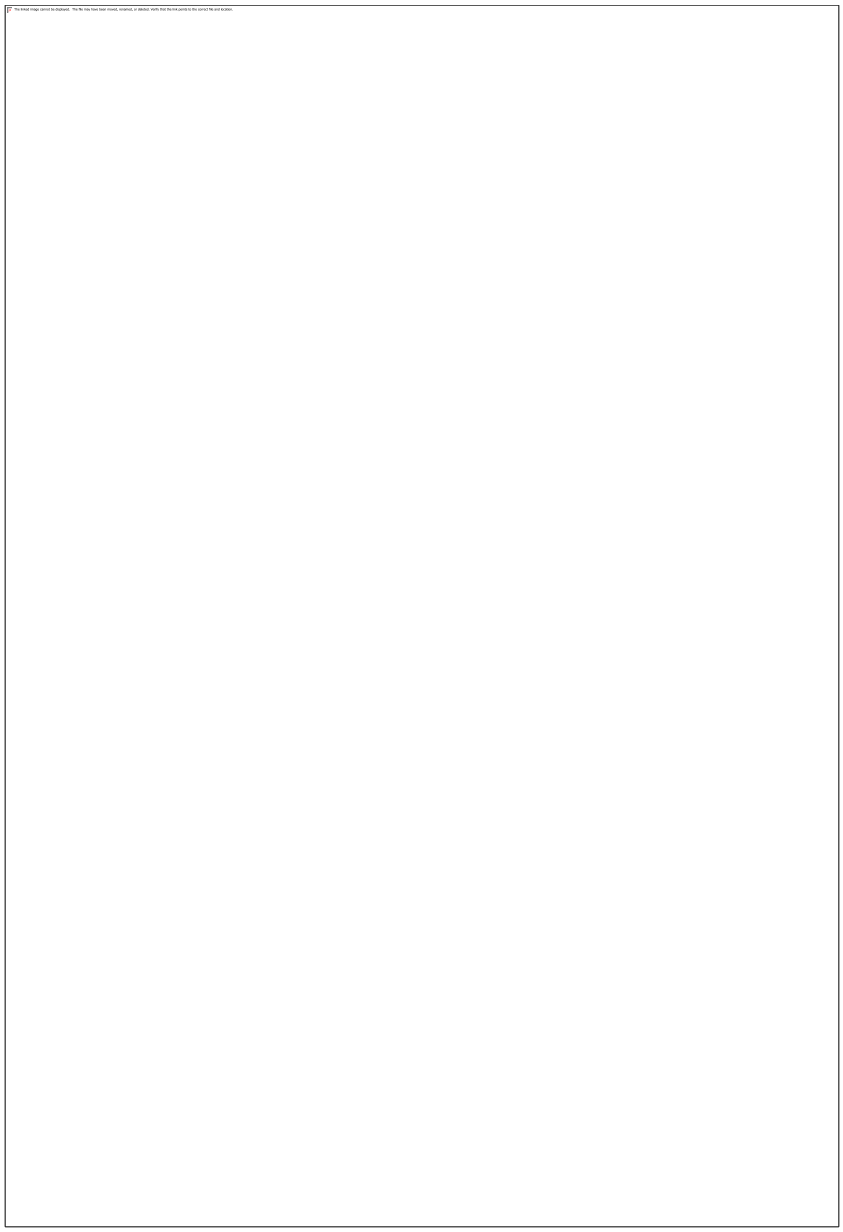
Make Call - Named Pipe


The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

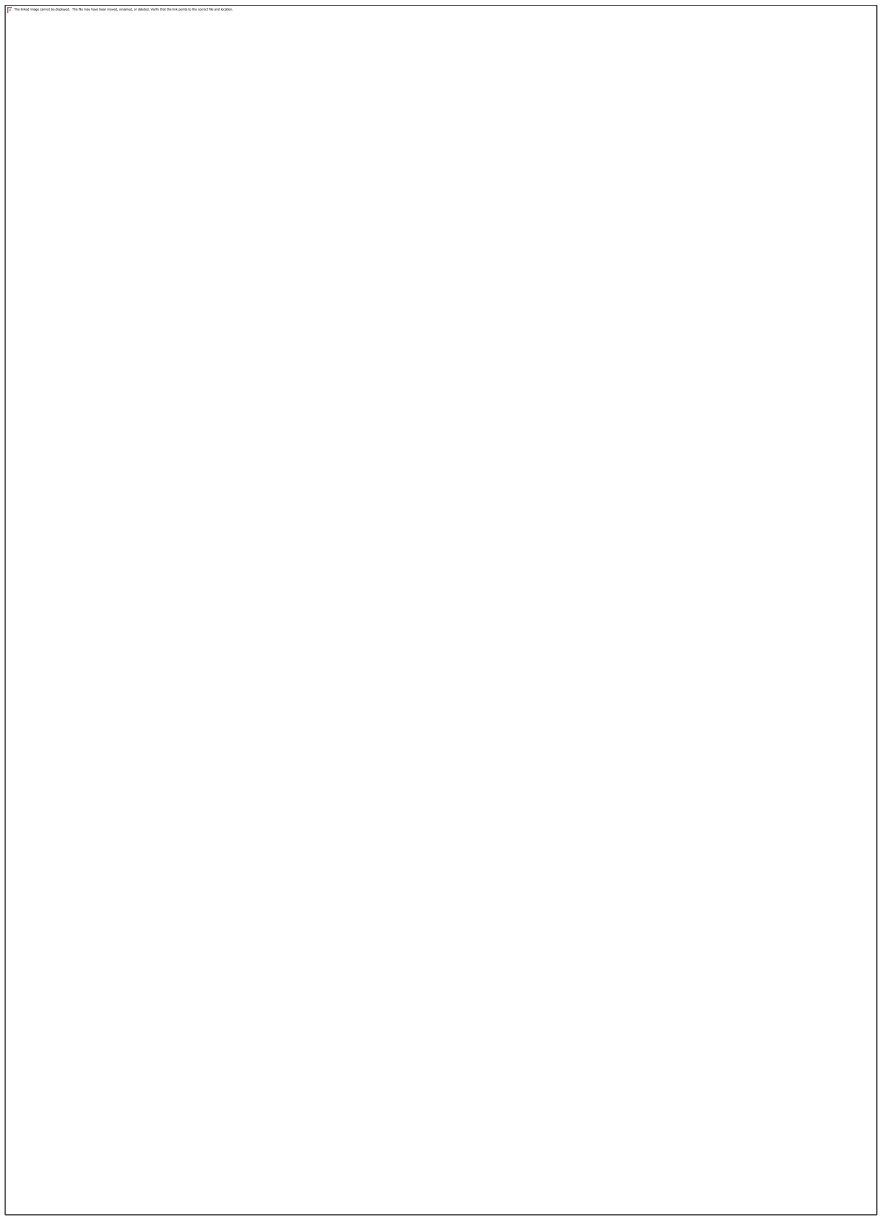
Answer Call



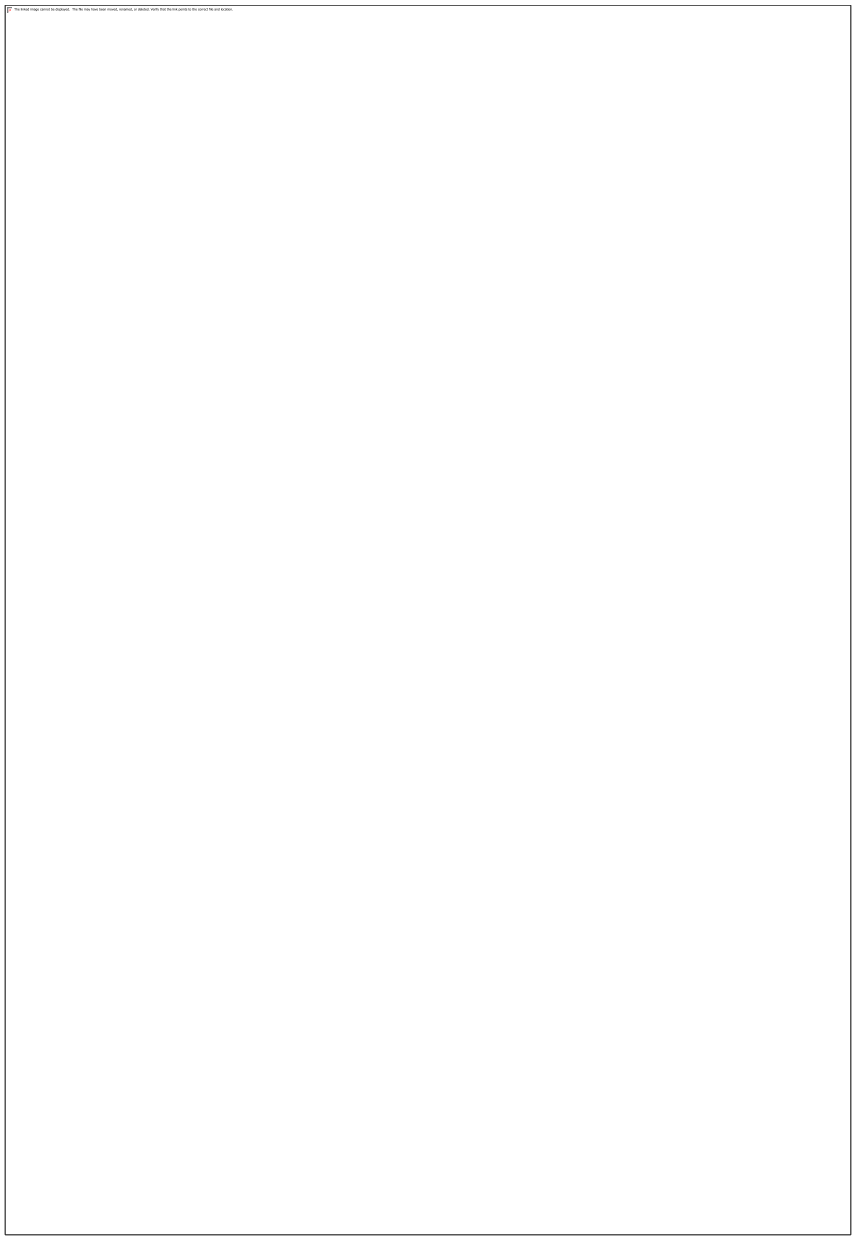
Mute and Unmute call



Hold and Retrieve call

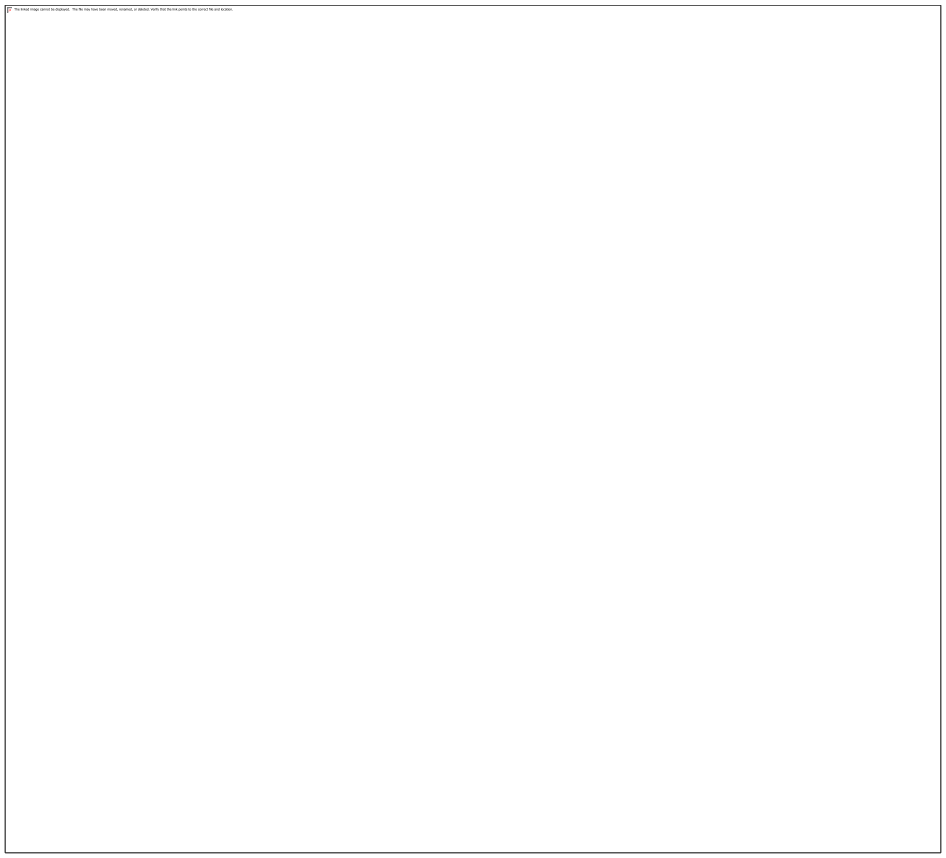


Terminate or End Call



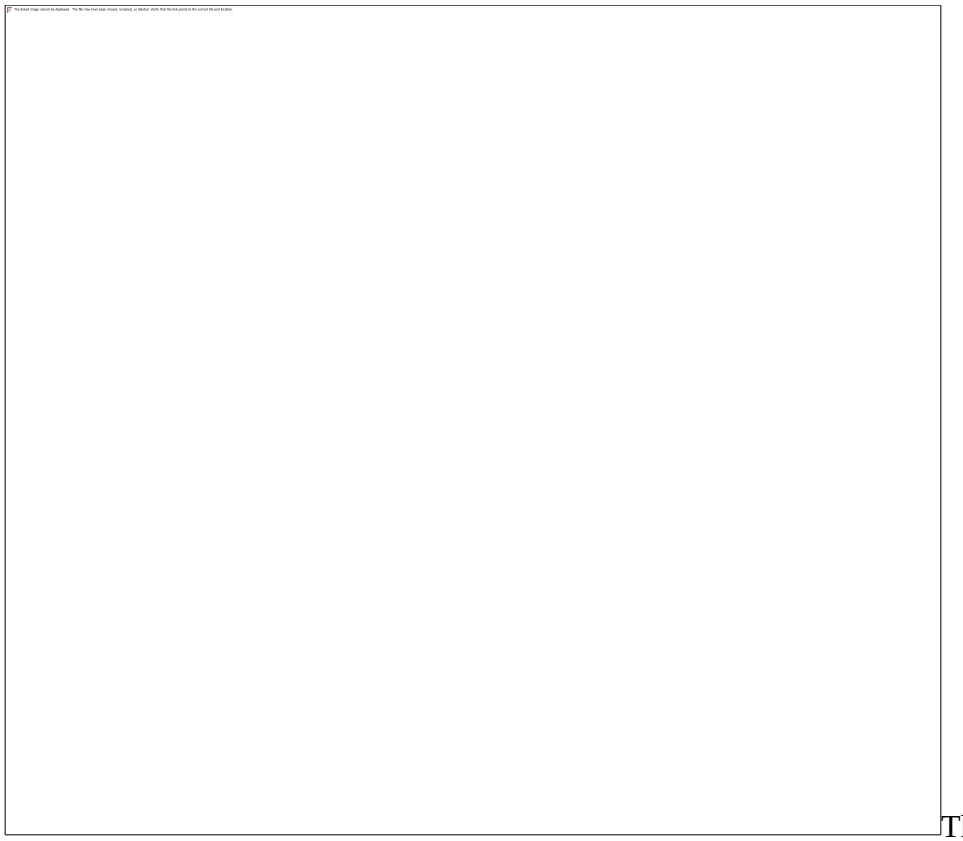
Media Device Listing

GetActiveAudioDevices



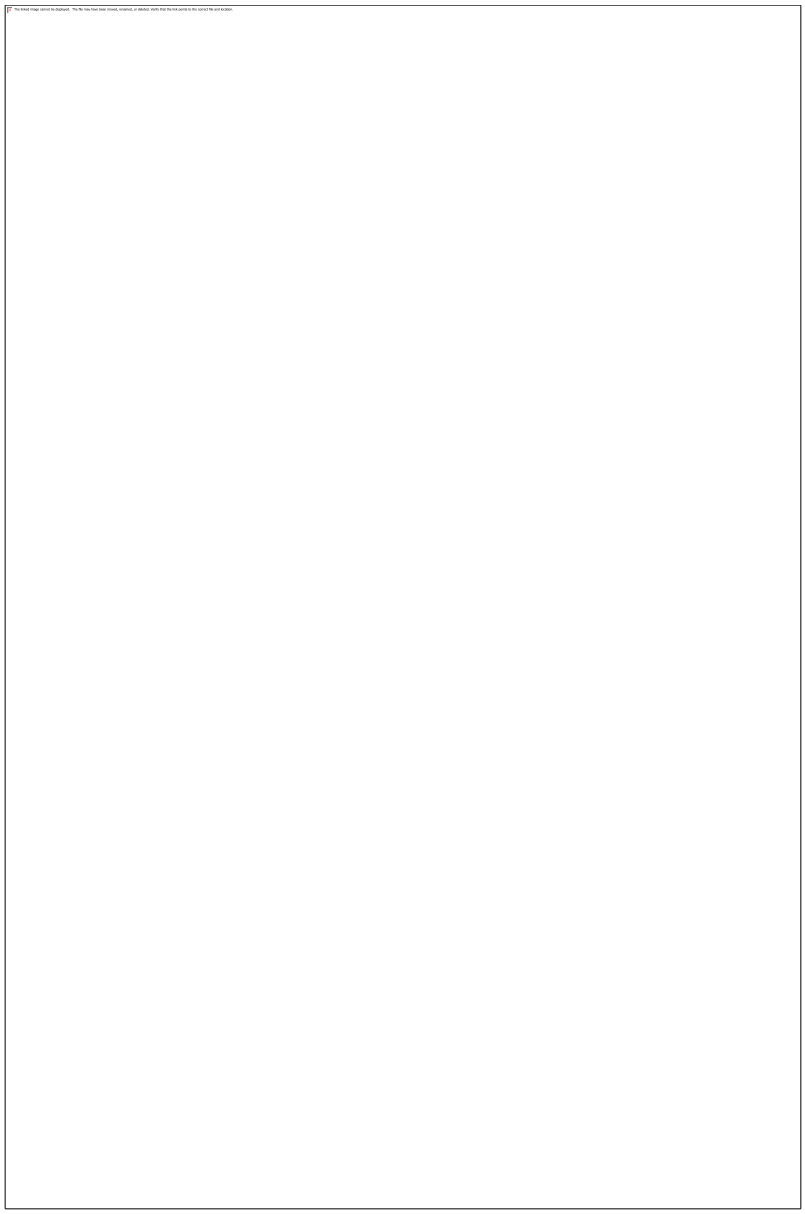
Application Interworking

Call Created by UC Application



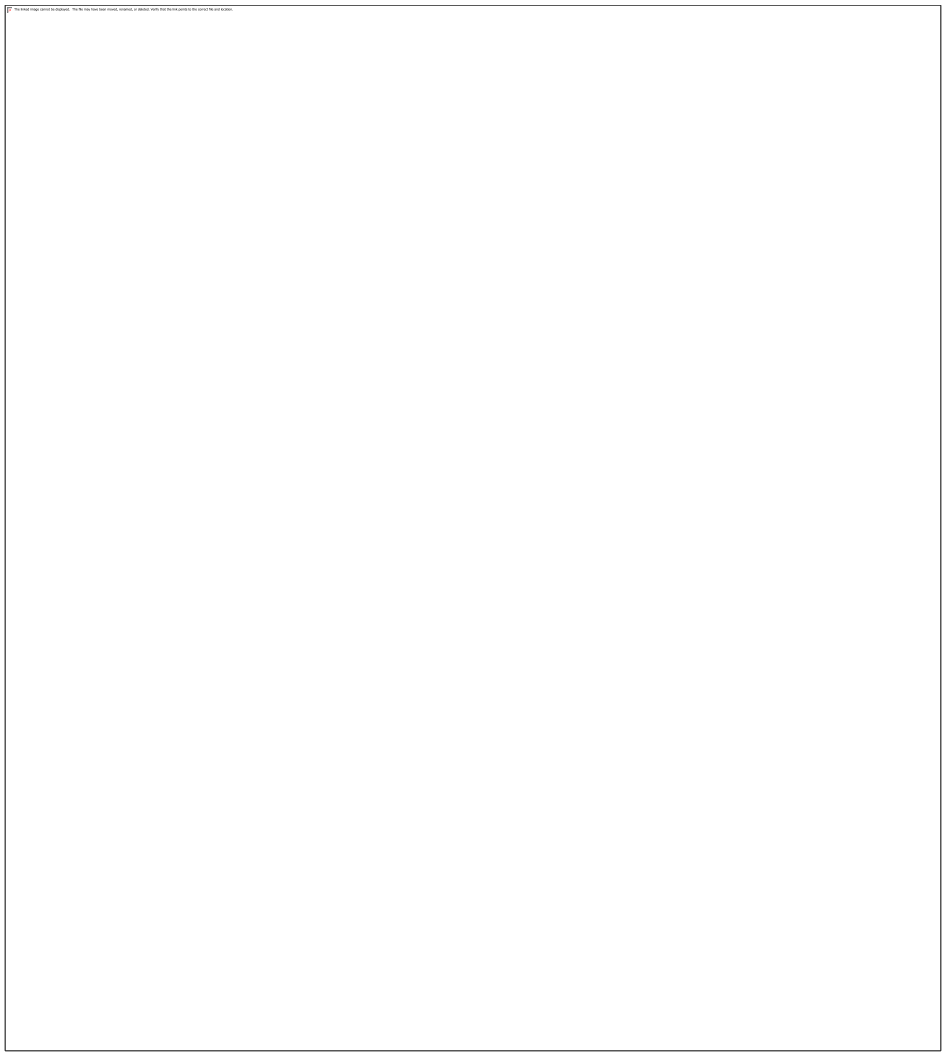
The above example shows the external application receiving call updates ahead of the External Application API, but this is for illustrative purposes and this behaviour is not guaranteed. The order of events or notifications is an implementation detail.

Call Created by Peer External Application



The above example shows the ExternalApp2 receiving call updates ahead of ExternalApp2, but this is for illustrative purposes and this behaviour is not guaranteed. The order of events or notifications is an implementation detail.

Call Hold by UC Application



The above example shows the user holding and resuming the call from the external application. It is possible that the user can use *HoldCall* using the external application and resume through the External Application API, or vice versa.

Call Hold by Peer External Application

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

In the above example, the sequence shows ExternalApp2 holding the call, and ExternalApp1 retrieving (unholding) the call. In addition, there is a UpdatedEvent containing the same call state information as the HoldResponse. Since the data is identical, there is minimal impact, but it is not ideal. The order of events is an implementation detail.

References

Windows NamedPipe Reference: https://msdn.microsoft.com/en-us/library/windows/desktop/aa365590%28v=vs.85%29.aspx?f=255&MSPPError=- 2147217396

Windows PipeList: https://technet.microsoft.com/en-us/sysinternals/dd581625.aspx

Appendix

Windows

C# Named Pipe Sample Code

```
using System;
using System.IO;
using System.IO.Pipes;

namespace ExternalApplicationConnect
{
  class Program
  {
  static void Main(string[] args)
```

```
  {
  NamedPipeClientStream pipeClient = new NamedPipeClientStream(".","AvayaCSDK-
Administrator",PipeDirection.InOut,PipeOptions.None);
  if (pipeClient.IsConnected != true)
  {
  pipeClient.Connect();
  }

  StreamReader sr = new StreamReader(pipeClient);
  StreamWriter sw = new StreamWriter(pipeClient);
  string registerString =
"{ \"vnd.avaya.clientresources.RegisterRequest.v1.1\" : { \"applicationId\" :
\"TestApp\", \"transactionId\" : \"23763992\" } } \0";
  string createCallString =
"{ \"vnd.avaya.clientresources.call.CreateRequest.v1.1\":
{ \"remotePartyNumber\":\"+13035382200,,683042\", \"transactionId\":
\"1765675\" } } \0 ";
  try
  {
  sw.Write(createCallString);
  sw.Flush();
  }
  catch (Exception ex) { throw ex; }
  }
  }
}
```