REST Web Services Interface

Product guide for prerelease

Document version 1841, produced on Friday, September 06, 2024.

main 90adc8bf40040649230176bbdd465f6261a2d8e0

Contents

List of Tables

1 Document History

Publication Date	Product Release	Notes
May 29, 2024	3.5.1	No updates.
April 15, 2024	3.5.0	No updates.
September 20, 2023	3.4.0	No updates.
July 24, 2023	3.3.0	New release.
December 1, 2022	3.2.0	New release.
September 29, 2022	3.1.2	Updated for new models and services; added documentation about Swagger and audit logs.
March 15, 2022	3.1.0	Document created.

2 REST Web Services Interface

This document describes the REST-style web services offered by Instant Connect Enterprise.

Every function provided to ICE clients by the ICE Server is accessible through these web services. They represent the same set of functionality that the ICE Desktop and ICE Mobile client's use to provision the system, and to be provisioned by it.

Most commonly, these services will be used by a customer's IT infrastructure to trigger a configuration change in Instant Connect as a response to some IT-detected condition. For example,

- An IoT system detects a machine failure on the factory floor; an intercom channel should be automatically established with the foreman and employees near the machine so they can quickly collaborate on a solution.
- An inventory control system detects that a store has run out of supply for a popular item; an automated voice (text-to-speech) alert should be transmitted on a channel that is shared by customer associates so staff is aware of shortage and can advise customers accordingly.
- A new employee has been on-boarded by corporate HR; as part of the on-boarding procedure the employee should automatically be added to certain channel groups and the enterprise would like this process to occur automatically.

This document assumes the reader is familiar with REST-style web services and has experience administering an ICE system (configuring channels, user access, etc.)

3 ICE System Architecture

The Instant Connect Enterprise system consists of two *planes* of data:

- A ***media plane*** that is responsible for microphone and speaker control; audio encryption and decryption; audio mixing (when multiple streams are heard together); voice encoding using one of the system's supported vocoders; tactical-mode location and presence.
- A ***provisioning plane*** that is responsible for user management and client authentication; licensing; enterprise-mode location and presence; channel configuration; private calling and telephony signaling.

In the most common deployment configuration, clients establish two connections to the ICE Server: A secure web socket (port 443) for provisioning, and an SSL socket (port 7443) to transmit and receive audio to a RallyPoint. When a channel is configured to use multicast, clients communicate audio via RTP using an IGMP group address and port specified by the channel.

Components illustrated in the diagram are described below (this is not a comprehensive list of all software components present in the ICE Server system):

Component	Description
RallyPoint	RallyPoints distribute audio traffic transmitted on a channel to other clients joined to the channel that are also connect to the RallyPoint (or mesh of RallyPoints).
Client Bridge	Authenticates client access to the system and terminates their web socket connection. Asynchronously delivers relevant and authorized system state changes to clients.
REST Bridge	A facade that exposes internal message-oriented service interfaces as REST-style web services.
Server Bridge	A service proxy used by other server entities. At this time, Instant Connect does not support any external, or third-party server integration through this proxy.
ICE Telephony	Connects the ICE platform with external telephony services enabling dial-into channels and soft-phone functionality in ICE Clients.

Component	Description
ICE Bridging	Enables participants of one channel to communicate with participants of another channels (sometimes called channel “patching” or “cross-banding”).
Kakfa Message Bus	An open-source, cloud-scale stream processing platform used to convey data between system components.
Platform Services	Provides the ICE Server’s primary application logic.
Cassandra Database	An open source, cloud-scale, no-SQL database that is used to store and lookup data.

The REST APIs described in this document apply only to the provisioning plane. They cannot be used to transmit or receive buffers of audio.

At this time, Instant Connect does not provide a public API for our media engine and RallyPoint.

4 ICE Server Architecture

The ICE Server is a message-oriented system modeled on the concept of *services* and *models*.

While you will not need to construct messages yourself, this information is provided to offer a better understanding of how these REST services operate “under the hood”.

4.1 Messages

A message encapsulates information sent between entities in the system. Messages can be sent, point-to-point between a single sender and a single receiver or *broadcast* from a single sender to any entity interested in receiving the message.

A message is a JSON-formatted string consisting of:

- A message type string, representing the kind of message.
- A header object indicating the address of the entity that the message is directed to; whether this message represents a request being made of a service method; and a sender-generated correlation ID (used to associate a request message with its response message).
- A payload object consisting of JSON-formatted attribute/value pairs.

4.2 Services

A service is a logical component that provides one or more *service methods* that can be invoked by another entity in the system by sending a message addressed to the service. A service method listens for *request* messages addressed to it, performs some process, and emits a *response* message in acknowledgement. Each service method defined by each service can be invoked through a REST service API.

Use unsupported APIs at your own risk.

Unsupported services may not be fully implemented or supported by ICE Desktop or ICE Mobile clients. APIs are subject to breaking changes or removal in future releases, and could put the system in an undesired state.

4.2.1 Service types

Service	Description
Alert Service	Methods for managing and sending text alerts.
Assembly Point Service	Methods for managing RallyPoints.
Audit Log Service	Methods for enumerating system audit logs and generating CSV-formatted reports from them.
Certificate Entity Service	Methods for managing cryptographic certificate entities. **Not supported for third-party use.**
Channel Group Service	Methods for managing groups of channels. **Not supported for third-party use.**
Channel Service	Methods for creating, enumerating, and modifying channels.
Cisco IP Phone Service	Methods for enabling or disabling ICE access to specific Cisco IP Desk Phones connected to a Cisco Call Manager.
External Server Service	Methods for creating and enumerating external servers. An external server is a non-person entity that can “attach” (log into) to the ICE server and interact with it via web socket.
File Data Service	Methods for interacting with files (like text message attachments or audio alerts) stored in the ICE Server. **Not supported for third-party use.**

Service	Description
Kenwood NEXEDGE Radio Service	Methods for interacting and managing Kenwood NEXEDGE radio systems. **Not supported for third-party use.**
License Feature Block Service	Methods for creating and managing tactical license blocks.
License Service	Methods for applying a license to the ICE Server and querying consumers of licensed features.
Metric Service	Methods enabling clients to report operational metrics and call data records to the ICE Server.
Multicast Service	Methods for managing ranges of multicast addresses. This service is used internally for “checking out” multicast group addresses and ports before assigning them to channels.
Organization Service	Methods for creating and enumerating organizations. **Not supported for third-party use.**
P25 Crypto Service	Methods for managing P25 encryption keys used in DFSI and ISSI radio system integration. **Not supported for third-party use.**
P25 DFSIG Radio Gateway Service	Methods for managing P25 radio system integration. **Not supported for third-party use.**
Person Group Service	Methods for managing groups of people. **Not supported for third-party use.**
Patch Service	Methods for creating and managing channel patches.
Person Service	Methods for managing users that can log into and access the system.
Reflector Service	Methods for creating and managing static reflections.
Radio Descriptor Service	Methods for creating and enumerating radio system descriptors. **Not supported for third-party use.**
Role Service	Methods for creating enumerating roles (permission groups) in the system. **Not supported for third-party use.**
Session Service	Methods for managing event subscription. **Not supported for third-party use.**
Telephony Service	Methods for managing dial numbers and placing telephone calls.

Service	Description
Text Message Session Service	Methods for creating and enumerating text message threads (sessions).
Webhook Service	Methods for creating and sending outbound webhooks.
Workflow Service	Methods for creating and enumerating workflow automations.

4.2.2 Example of a message sent to a service

An example of a message sent to the Person Service requesting a query be performed and the first ten persons matching the search string `def` be returned

```
{
  "type": "person:FindPersons",
  "headers": {
    "isRequest": true,
    "destination": "SERV:person:",
    "correlationId": "c8fb3787-32e4-4296-b8cd-ab8f13c58133"
  },
  "payload": {
    "messageType": "person:FindPersons",
    "attributes": {
      "searchString": "def",
      "limit": 10
    }
  }
}
```

4.3 Models

A model represents an addressable object in the system that is analogous to an object in an object-oriented programming language. A model encapsulates data (in form of attribute/value pairs) with a set of operations that can be performed on the data.

4.3.1 Capabilities

ICE models support a multiple inherence model through *capabilities*. Models are composed of two or more *capabilities*; a *base capability* common to all models, and one or more type-specific capabilities.

Each capability defines its own set of attributes and methods, the name of each is prefixed with the type (or namespace) of the capability. This prevents two capabilities from inadvertently aliasing the attributes or functions of another capability applied to a single model.

Through the base capability, every model contains the following:

- A permanent, universally unique ID (`base:id`) that differentiates the object from every other object in the system. For example, `e071175e-2bf8-4c06-96cd-88115ed3e295`.
- An address (`base:address`), consisting of the object's namespace and ID concatenated together with a colon and prefixed with `SERV:`. For example, `SERV:person:e071175e-2bf8-4c06-96cd-88115ed3e295`.
- A method (`base:GetAttributes`) for retrieving the attributes of the model and a method (`base:SetAttributes`) for setting attributes.

4.3.2 Example of a message sent to a model

An example of a message addressed to a Channel model requesting the list of persons participating in the channel:

```
{
  "type": "chan:ListPersons",
  "headers": {
    "isRequest": true,
    "destination": "SERV:chan:5fc7653f-5e57-4569-b74f-1fcd24ff22d6",
    "correlationId": "7dffa31c-9aef-4a2d-b9da-713784035962"
  },
  "payload": {
    "messageType": "chan:ListPersons",
    "attributes": {}
  }
}
```

4.3.3 Model types

Model	Description
Alert	A text alert that can be sent to users on a channel.
Assembly Point	An Assembly Point refers to one or more 'meshed' RallyPoints functioning as a logical point for bridging communications across multicast domains.

Model	Description
Certificate Entity	A certificate entity refers to an element (like a key or certificate) in a public key infrastructure (PKI) that can be shared with other Instant Connect entities.
Channel	A Channel represents the configuration of a half or full-duplex voice pathway and includes attributes like codec, encryption, and network connectivity. Channels are assigned to users (Person entities) who then have access to read the channel configuration and join the talk group.
Channel Group	Channel groups allow an administrator to perform certain 'bulk' operations on Channel entities. That is, a group of Channels can be treated as a single entity.
External Server	An external server is an ICE Server software component that can communicates with the platform's services using the same APIs that a user agent would (that is, an external server does not access the database or message bus directly). However, unlike an end user, an external server has organization-level access to all services and entities in the system.
File Data	Metadata about a file created and managed via the File Data Service.
Kenwood NEXEDGE Radio Group	Represents a Kenwood NEXEDGE radio group as managed through the Kenwood NEDEGE Radio Service.
License	An Instant Connect Enterprise software license.
License Feature Block	Represents a license feature block; a logical collection of tactical licenses used for a specific purpose.
Multicast	Represents a multicast group address and port that can be used by a client for communication.
Organization	An organization is loosely equivalent to a tenant in a multi-tenant system. All resources and entities are associated with an Organization.
P25 Crypto Key	Represents a P25 radio system encryption key.
Patch	Represents a patch (sometimes called a "bridge" or "cross-band") of two or more channels.

Model	Description
Person	A Person is a user of the Instant Connect Enterprise system who can log into ICE Server and access resources. Note that 'tactical'-mode users do not necessarily have or require a Person entity in ICE Server as these users do not have to log in for access to tactical channels.
Person Group	Person groups allow an administrator to perform certain 'bulk' operations on Person entities. That is, a group of Persons can be treated as a single entity.
Platform Client	A platform client represents an 'abstract' ICE Server client. Every entity that accesses server resources has a platform client associated with it and a single PlatformClient may have multiple authorized entities sharing it. The platform client represents the 'message domain' that the accessor is operating in.
Radio Descriptor	A model of a radio descriptor.
Reflector	A model of a static reflector.
Role	When a role is assigned to a user (Person), that user inherits all the permissions of the role in addition to whatever permissions that user may already have.
Text Message Session	A text messaging thread ("session") between users or a channel.
Webhook	Represents an outbound webhook used as an action in a workflow automation.
Workflow	A model of a workflow automation.

5 Getting Started

Use of the REST web services is a licensed feature of Instant Connect Enterprise.

Verify that you have purchased this feature:

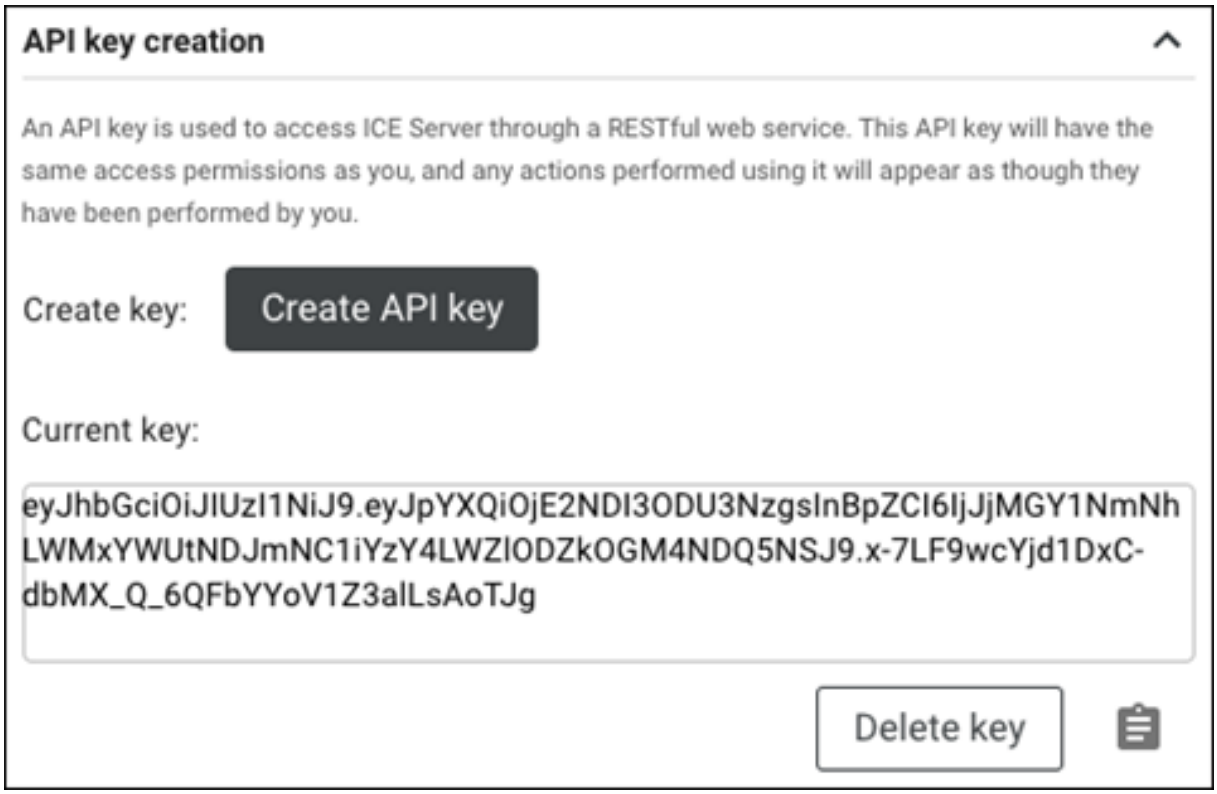
1. Open the ICE Desktop application
2. Log into your ICE Server using a credential with administrator privliges
3. Click the settings (gear) icon in the top-right of the dashboard window.

4. Click the License item in the Settings slide-out menu.

If you see “RESTFUL_SERVICES” listed in “Licensed Features” panel, your system is authorized for access through REST web service. Contact your Instant Connect sales representative for assistance if you don’t see a license for this feature and believe you’re entitled to it.

5.1 Authorizing Access

Every call to a REST service endpoint must be authorized using an API key. API keys are created and retrieved using the ICE Desktop client (Click the gear icon on the dashboard, then navigate to the “License” panel).



Each API key is associated with the authorized ICE user that created it, and every invocation authorized with the key inherits the same permissions as the user. Similarly, for audit purposes, any actions performed via the REST interface will be logged as having been performed by the user linked to the API key.

The API key must be sent to ICE Server as an HTTP header, `Authorization: Bearer`(replace API_KEY with the your system API Key). For example:

```
curl -X POST "https://my.ice.com/instant-connect/chan/ListAllChannels" \
  -H "accept: application/json" \
```

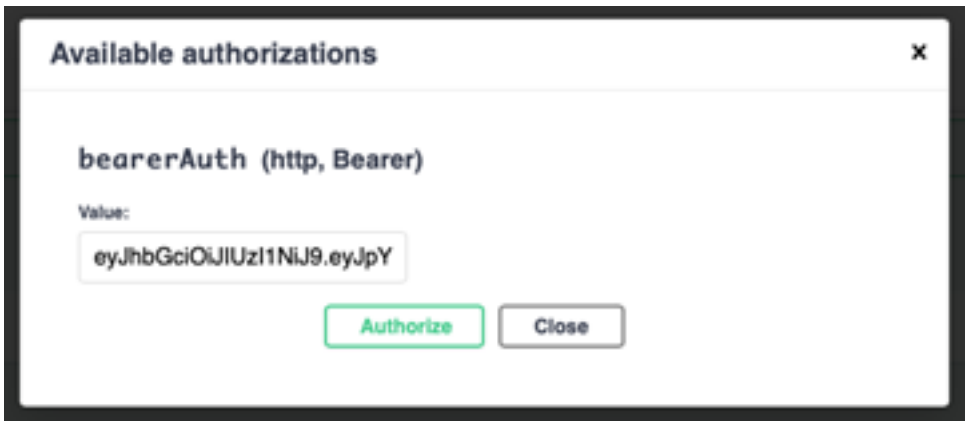
```
  -H "Authorization: Bearer API_KEY" \
  -H "Content-Type: application/json" \
  -d "{}"
```

5.2 Swagger

Documentation for each REST API is provided by Instant Connect in the form of an OpenAPI document. This is a standard for documenting REST-style APIs in a way that can be machine processed and rendered into a human-readable document.

Swagger is an industry standard tool for producing an interactive web-based set of documentation from an OpenAPI document. Swagger is installed by default in ICE systems and can be reached by opening a browser and navigating the to the `/swagger` path of your server's host. For example, `https://demo.icnow.app/swagger`.

In order to use the interactive "Try it out" functionality of the web application, you must first authorize yourself on the Swagger page. To do so, click the "Authorize" button near the top-right of the page, then enter your API key into the "bearer auth" field displayed. For example:



6 ICE Cookbook

6.0.1 Using the REST API to extract audit logs

In some environments it may be useful to programmatically export audit logs or migrate them to another system using ETL (extract, transfer and load) tools. The REST API can be used to accomplish this.

Start by retrieving an API key from the "License" page of the settings window in ICE Desktop. Recall that at API key is tied to the user who created it and any REST request made using that API key will

appear to have been executed by the user whose API key authorized the request.

We will use the `/audit/PageAuditLogs` service method to retrieve audit logs. This service method accepts four arguments:

1. `startTime` - A timestamp, in "UNIX time", representing the number milliseconds since the UNIX epoch (midnight on January 1, 1970 UTC). Use the value `0` to start with the oldest audit log messages available.
2. `endTime` - A UNIX timestamp representing the most recent time for which audit log messages should be retrieved. For example, `1664465568409`.
3. `pageSize` - The number of records to be returned in each page. Very large page sizes will put heavy load on the system and may fail altogether. A value between 10-100 is probably appropriate.
4. `pageNumber` - Indicates the page to be returned. In general, systems should initially request `pageNumber` of `0`, then increment this value until all pages have been returned. The number of available pages can be calculated by dividing the `totalCount` value returned in the initial response by the page size.

The response will look like:


```
{
  "messageType": "audit:PageAuditLogsResponse",
  "attributes": {
    "totalCount": 10000,
    "auditlogs": [
      {
        "actor": "SERV:person:d23b8eb5-60fb-4f6b-84f9-b41f3ceaf0c8",
        "entityName": "liza liza",
        "entityType": "Person",
        "eventTime": 1664465294640,
        "description": "liza liza connected to server from host
           109.241.246.186",
        "action": "Session Created",
        "actorName": "liza liza",
        "entity": "SERV:person:d23b8eb5-60fb-4f6b-84f9-b41f3ceaf0c8"
      }
    ]
  }
}
```


Utilize an ETL tool or script of your own creation to extract and transfer this data into the system or format of your choice.

7 FAQs

7.0.1 Can I use these APIs to transmit or receive audio?

No. Remember that these APIs provide an interface to the ICE provisioning system; they do not provide control of the underlying media engine. They can be used to configure server-managed entities like people, channels, and licenses but they do not enable access to realtime voice communications or instant replays.

At this time, Instant Connect does not provide a public API or library for accessing realtime voice communications.

7.0.2 Why don't these APIs use canonical REST HTTP verbs? Why is every access modeled as an HTTP POST?

ICE Server was not designed using a traditional, three-tier service-oriented architecture. Instead, the system is based on a reactive, message-oriented design. Native ICE clients communicate with the server by sending a receiving messages to it via a web socket. Clients are able to "react" to configuration changes with very low latency because they receive asynchronous notifications of state changes. Clients are not polling a service interface to be made aware of such changes.

These REST APIs are constructed as a facade fronting the system's native message-oriented architecture. To achieve this efficiently, the REST APIs were designed to produce and consume payloads that mimic the native message payloads. Restructuring these messages to mimic REST-canonical query-string or path-parameters was deemed cost prohibitive.