Sendbird Calls SDK for Unity

Introduction

With Sendbird Calls SDK, users in a Sendbird application can communicate with one another in real-time audio using group calls in a room. Users can create and enter a room to start a group call and up to 100 participants are allowed in a room.

This document describes how to integrate the Calls SDK to your app and how to let users join and interact with one another on a group call. To learn more about the components of the Calls SDK, refer to the SDK API reference.

|**Note**: This documentation is for the private beta release available for select customers only.

Install the SDK

To start using the group call functionality in your app, you should first install the Calls SDK.

1. Go to **Window** > **Package Manager**.
2. Click **+** at the top left and select **Add package from git URL...**
3. Enter the following URL.
   https://github.com/sendbird/sendbird-calls-unity.git
4. Click **Add** to start installing the Calls SDK.

After installing the Calls SDK, you should initialize the `SendbirdCall` instance using your Application ID, which you can retrieve by signing into Sendbird Dashboard.

```
// Initialize the SendbirdCall instance to use APIs in your app.
SendbirdCall.Init(APP_ID);
```

You should also authenticate a user to connect to the Sendbird server by calling the `Authenticate()` method. Authenticate a user with their user ID, which is also found on Sendbird Dashboard.

```
 // The USER_ID below should be unique to your Sendbird application.
SbAuthenticateParams authenticateParams = new SbAuthenticateParams(USER_ID);
SendbirdCall.Authenticate(authenticateParams, (user, error) =>
{
    if( error == null ){/* The user has been authenticated successfully and is connected to the Sendbird server. */}
});
```

Create a room

You can create a room for users to enter and participate in a group call. When a room has been created, a unique room ID is given to the room and its status becomes `Open`. Each room can host up to 100 participants.

```
SbRoomParams roomParams = new SbRoomParams();
SendbirdCall.CreateRoom(roomParams, (room, error) =>
{
    if( error == null ){/* `room` is created with a unique identifier `room.RoomId`. */}
});
```

Enter a room

Users can enter a room by using a room ID. When a user enters a room, the user is assigned a unique Participant ID and a participant is created.

To enter a room, you should fetch the `room` instance from the Sendbird server using the room ID. You can fetch the most up-to-date `room` instance from the Sendbird server using the `SendbirdCall.FetchRoomById()` method. Alternatively, you can use the `SendbirdCall.GetCachedRoomById()` method to retrieve the most recently cached `room` instance from the Calls SDK.

```
SendbirdCall.FetchRoomById(ROOM_ID, (room, error) =>
{
    if( error == null ){/* `room` with the identifier `ROOM_ID` is fetched from the Sendbird Server. */}
});

SbRoom room = SendbirdCall.GetCachedRoomById(ROOM_ID);
// Returns the most recently cached room with the room ID from the SDK.
// If there is no such room with the given room ID, `null` is returned.
```

|**Note**: A user can enter a room using multiple devices or browser tabs. Entering from each device or browser tab creates a new participant for the user.

After retrieving the room, call the `SbRoom.Enter()` method to enter the room.

```
SbRoomEnterParams roomEnterParams = new SbRoomEnterParams();
room.Enter(roomEnterParams, (error) =>
{
    if( error == null ){/* User has successfully entered `room`. */}
});
```

Exit a room

Users can leave a room using the `room.Exit() ` method.

```
room.Exit();
```

Manage custom items

You can store additional information associated with a room as key-value pairs in a `room` object. Custom key-value items are saved as an object and can be updated or deleted as

long as the room status remains `Open`. You can add information related to customer service, refund, or inquiry as custom items to improve user experience.

Add custom items

After a room has been created, users can add custom items to `roomParams` as an object. By default, `customItems` is an empty object.

```
var customItems = new Dictionary<string, string> { { "key1", "value1" } };
var roomParams = new SbRoomParams { CustomItems = customItems };
SendbirdCall.CreateRoom(roomParams, (room, error) => { });
```

Update and delete custom items

You can update or delete custom items that are added to a room. To update a custom item, call the `room.UpdateCustomItems()` method. A new item with a new key will be added to the list of custom items. If a newly created custom item has the same key as the existing custom item, the value of the new item will replace the value of the existing item.

You can delete custom items by passing the list of keys to the existing custom items as a parameter to the `room.DeleteCustomItems()` method.

```
var items = new Dictionary<string, string> { { "key1", "value3" } };
room.UpdateCustomItems(items, (customItems, updatedKeys, error) =>
{
  // Custom items with matching keys in `updatedKeys` have been updated in the customItems`.
});

var keys = new List<string> { "key1" };
room.DeleteCustomItems(keys, (customItems, deletedKeys, error) =>
{
  // Custom items with matching keys in `deletedKeys` have been deleted from `customItems`.
```

```
});
```

Receive events

Participants in a room can receive events from the Sendbird server when other participants in the room have updated or deleted custom items. To receive events from other participants, use `OnCustomItemsUpdated()` and `OnCustomItemsDeleted()`. Each event contains the list of keys of the updated or deleted custom items in `updatedKeys` and `deletedItemKeys` respectively.

Modification of custom items and delivery of events in the event handlers can only be done when the room status is `Open` and there are participants in the room. To see which custom items have been changed for ongoing or ended group calls in a room, you can use the Calls API or the `room.FetchCustomItems()` method.

```
class MyRoomListener : SbRoomEventListener
{
  public void OnCustomItemsUpdated(ReadOnlyCollection<string> updatedKeys)
  {
    // Custom items with `updatedKeys` have been updated in the target room.
  }

  public void OnCustomItemsDeleted(ReadOnlyCollection<string> deletedItemKeys)
  {
    // Custom items with `deletedKeys` have been deleted in the target room.
  }
}
```

Retrieve a list of rooms

You can retrieve a list of rooms by using the parameters in `RoomListQuery` as filters. The following table shows all supported filters for `RoomListQuery` to search for specific rooms you want to retrieve.

List of filters

Filter name	Description
Limit	Specifies the number of rooms retrieved at a time.
RoomIds	Includes rooms with the specified room IDs in the result returned.
RoomState	Includes rooms with the specified room state in the result returned.
CurrentParticipantCountRange	Includes rooms with a participant count that falls within the specified range of numbers in the result returned.
CreatedByUserIds	Includes rooms created by the specified user IDs in the result returned.
CreatedAtRange	Includes rooms that were created between the specified range of time in the result returned.

You can specify the filters as the following.

```
TimeSpan aWeekAgo = DateTime.UtcNow.Subtract(new DateTime(1970, 1, 1, 0, 0, 0,
DateTimeKind.Utc)) + TimeSpan.FromDays(-7);
var roomListQueryParams = new SbRoomListQueryParams()
{
  Limit = 50,
  RoomState = SbRoomState.Open,
  CurrentParticipantCountRange = SbRange.GreaterThanOrEqualTo(1),
  CreatedByUserIds = { "USER_ID1", "USER_ID2" },
  CreatedAtRange = SbRange.GreaterThanOrEqualTo((long)aWeekAgo.TotalMilliseconds)
};
```

To retrieve the list of rooms you specified using the filters, call the `SbRoomListQuery.Next(SbRoomListQueryHandler)` method.

```
var roomListQuery = SendbirdCall.CreateRoomListQuery(roomListQueryParams);
roomListQuery.Next((rooms, error) => { });
```

After retrieving the list of rooms, the user can select a room they wish to enter using the following code.

```
rooms[index].Enter(roomEnterParams, (error) =>
{
  // User has entered the room.
});
```

|**Note**: The room data returned is updated only after a user has entered the room. To update the details about a room, call the `SendbirdCall.FetchRoomById()` method.

Interact within a room

Using the participant object, you can manage participant actions such as turning on or off the microphone.

To turn on or off a participant's microphone, you can use the following methods from the `room.LocalParticipant` object.

```
// Mutes the local participant's microphone.
room.LocalParticipant.MuteMicrophone();

// Unmutes the local participant's microphone.
room.LocalParticipant.UnmuteMicrophone();
```

Handle events in a room

You can let users know about the actions taken in the room they are in. Users can receive event alerts when other participants enter or leave the room, change their microphone settings or when the room has been deleted.

Add event listener

You can add an event listener for users to receive events that occur in the room.

```
room.AddEventListener(new MyRoomListener());

class MyRoomListener : SbRoomEventListener { }
```

Receive events on enter and exit

When a user joins or leaves a room, other participants in the room receive the following events.

```
class MyRoomListener : SbRoomEventListener
{
  // Called when a remote participant has entered a room.
  public void OnRemoteParticipantEntered(SbRemoteParticipant remoteParticipant) { }

  // Called when a remote participant has exited a room.
  public void OnRemoteParticipantExited(SbRemoteParticipant remoteParticipant) { }
}
```

Receive events on microphone setting

When a user mutes their microphone using the `MuteMicrophone()` method, other participants receive the following event.

```
class MyRoomListener : SbRoomEventListener
{
  // Called when a remote participant's audio settings have changed.
  public void OnRemoteAudioSettingsChanged(SbRemoteParticipant remoteParticipant) { }
}
```

Receive events on deleted room

To delete a room using the Calls API, see the delete a room page in the Calls API documentation. When a room has been deleted, participants in the room receive the following event.

```
class MyRoomListener : SbRoomEventListener
{
  // Called when the room has been deleted.
  public void OnDeleted() { }
}
```

Remove event listener

To stop receiving events about a room, remove the registered event listeners using the following code.

```
// Removes a specific listener.
room.RemoveEventListener(myRoomListener);

// Removes all listeners.
room.RemoveAllEventListeners();
```

Reconnect to media stream

When a participant experiences connectivity issues and loses the media stream in a room, the Sendbird Calls SDK automatically tries to reconnect the participant to the room. If the Calls SDK fails to reconnect for about 40 seconds, an error is returned.

```
class MyRoomListener : SbRoomEventListener
{
  // Called when an error has occurred.
  public void OnError(SbError error, SbParticipantAbstract participant)
  {
    if (error.ErrorCode == SbErrorCode.LocalParticipantLostConnection)
    {
      // Handle reconnection failure here.
      // Clear resources for group calls.
    }
  }
}
```

|**Note**: See the SDK API reference for more information.