v4.29.0

This version enhances support for Multi-Conversations and enables Multi-Party Conversations. Users of the Web SDK can now be joined into multiple conversations and allows multiple users to message each other. Read about Multi-Party Conversation on our guide.

What's New

Initializing the SDK

The new version of the SDK should now be initialized with an `integrationId` rather than an `appId` . You can find your `integrationId` through the Sunshine Conversations dashboard when connecting a Web SDK integration or through the API.

```
Smooch.init({integrationId: '<integration-id>'}).then(function() {
    // Your code after init is complete
});
```

Note that to support backwards compatibility an `appId` can still be used to initialize the Web SDK, but the SDK will be initialized with the web SDK integration with the oldest creation date.

New APIs

The new version of the SDK brings some new APIs to help manage conversations.

getConversationById(conversationId)

An asynchronous method to fetch a specific conversation.

```
Smooch.getConversationById('<conversation-id>').then(function(conversat
    // Your code after conversation is returned
});
```

getConversations(offset)

An asynchronous method that returns a list of conversations sorted by the `lastUpdatedAt` property. An optional offset argument can be specified for paging purposes.

```
Smooch.getConversations().then(function(conversations) {
    // Your code after conversation list is returned
});
```

New Events

The new version of the SDK brings new conversation events.

`participant:added`, `participant:removed`

These events are triggered when a participant is added or removed from a conversation.

`conversation:added`, `conversation:removed`

These events are triggered when the user is added or removed from a conversation.

What's Changed

APIs

Many SDK APIs have been updated to allow for an optional `conversationId` argument. This includes `sendMessage(message, conversationId)`, `startTyping(conversationId)`, `stopTyping(conversationId)`, and `markAllAsRead(conversationId)`.

For all of these APIs if the `conversationId` argument is not provided, the user's active conversation will be used.

Events

A `data` object is now being exposed with the `message`, `message:received`, `message:sent`, `participant:added`, `participant:removed`, `conversation:added`, `conversation:removed`, `unreadCount`, `connected`, `disconnected`, `typing:start`, and `typing:stop` events. This object contains a truncated version of the conversation associated with the event.

```
Smooch.on('connected', function(data) {
    console.log('Connected with conversation ', data.conversation._id);
    console.log(data);
});

// data object
data = {
    conversation: {
        _id: '<conversation-id>',
        unreadCount: 0,
        lastUpdatedAt: 1581010017.596,
        type: "multiUser",
        participants: [
            {
                _id: '<participant-id>',
                appUserId: '<appUser-id>',
                unreadCount: 0,
                lastRead: 1581010017.596
            }
        ],
        metadata: {}
    }
}
```

```
Smooch.on('unreadCount', function(unreadCount, data) {
    console.log(`The number of unread messages was updated for conversa
});
```

Delegates

A `data` object is now being exposed by the `beforeSend`, `beforeDisplay`, and `beforePostbackSend` delegates. This object contains a truncated version of the conversation the delegate is impacting.

```
Smooch.init({
    integrationId: '<integration-id>',
    delegate: {
        beforeSend(message, data) {
```

```
            if (data.conversation._id === '<conversation-id>') {
                message.metaga = {
                    any: 'info'
                };
            }

            return message;
        }
    }
});
```