ContactsGatewayService(CGS):: GetCommsFeatureMetadataForOwner API

Problem Statement

In the recent iOS update with version number 15.1, Alexa app is getting crashed when users try to invite their contact using Native SMS app. This document focuses to write a new API to mitigate this kind of issues.

Ticket : https://t.corp.amazon.com/D31656316

Background

Users can invite their contacts by choosing Invite option in contact card/new call page/favourites popup. When multiple channels are applicable for inviting a contact, user is present with a bottom sheet displaying this list of applicable channels among which they can choose to send the invite.

Native SMS option is displayed by default in the bottom sheet. In iOS versions >= 15.1, Alexa app is getting crashed when user selects the native SMS option in the bottom sheet.

Mitigation

We made change in createInvitation API to throw a popup for cases when user tries to invite via SMS in iOS devices with version >= 15.1 instead of crashing the app entirely.

Impact

As the adoption rate of new iOS version increases, we are seeing more number of instances where we are throwing the above error message.(16% as of Nov 30, 2021)

Hubble query : https://hubble.amazon.com/?queryID=5041dcbf-7c27-4580-a889-184314b1d6e1

Necessity for moving invitation channels to backend

Right now, the applicable invitation channels are computed in react native code base.

Hence we have limited control over end user experience in the backend code. So, we need to either dial down Invites completely to mitigate this issue completely till we fix the root cause.

We need to have individual controls over each invitation channel which would help us to not show only the impacted channels in Invites flow and other channels can operate as usual.

Since app code changes follows a more strict schedule for getting deployed, the control of applicable channels for customer is moved to backend via newly introduced GetCommsFeatureDyanamicProperties API in ContactsGatewayService.

We cannot use config service since the data returned is not static and it involves computation based on the request params.

This control of having invitation channels in backend would give us quick recoverability in case of issues , especially when we expand Invites across multiple channels such as Email and other apps.

Possible options

Option 1 : Specific API in CGS

We can introduce GetInvitationChannels API in CGS which would return the applicable channels for the given customer.

Request and response definition

Request API path :

`GET /users/{commsId}/invitationChannels`

Response definition :

Response definition

```
public static class Response {
        // Contains the invitation channels(SMS, WhatsApp)
    List<String> invitationChannels;
}
```

Considerations with this approach

- This will serve only for returning invitation channels and will not be generic enough to accommodate computation of any other feature related information.
- This is suitable only for the mitigation and in future, if the issue is fixed with app side code changes, it would be difficult to deprecate this API since old app versions would still be invoking this API.

Option 2 : Generic API in CGS

We can introduce a new API called GetCommsFeatureMetadataForOwner in CGS which has a query param determining the feature name.

CGS maintains internal mapping between each feature and the processor for computing information related to that feature.

This way, the dynamic properties corresponding to particular feature present in query will be populated in the response.

Request and response definition

Request API path :

`GET /users/{commsId}/commsFeatureMetadataForOwner?featureIdentifier=<FeatureName>`

Response definition :

Response definition

```
public static class Response {
        String featureName;
    List<FeatureProps> featureProperties;
}


public class FeatureProps {
        String propKey;
        List<String> propValues;
}
```

Pros :

- App side code will be light weight.

- Branching will be present in backend and there will be a single API which can be called to get feature related properties varying based on query param.
- Extendable for other use-cases in future too.

Considerations :

When this API is extended for other use cases, this API might incur different latencies for different features, which needs to be into analysed since there would be impact on UPL if latency is more.

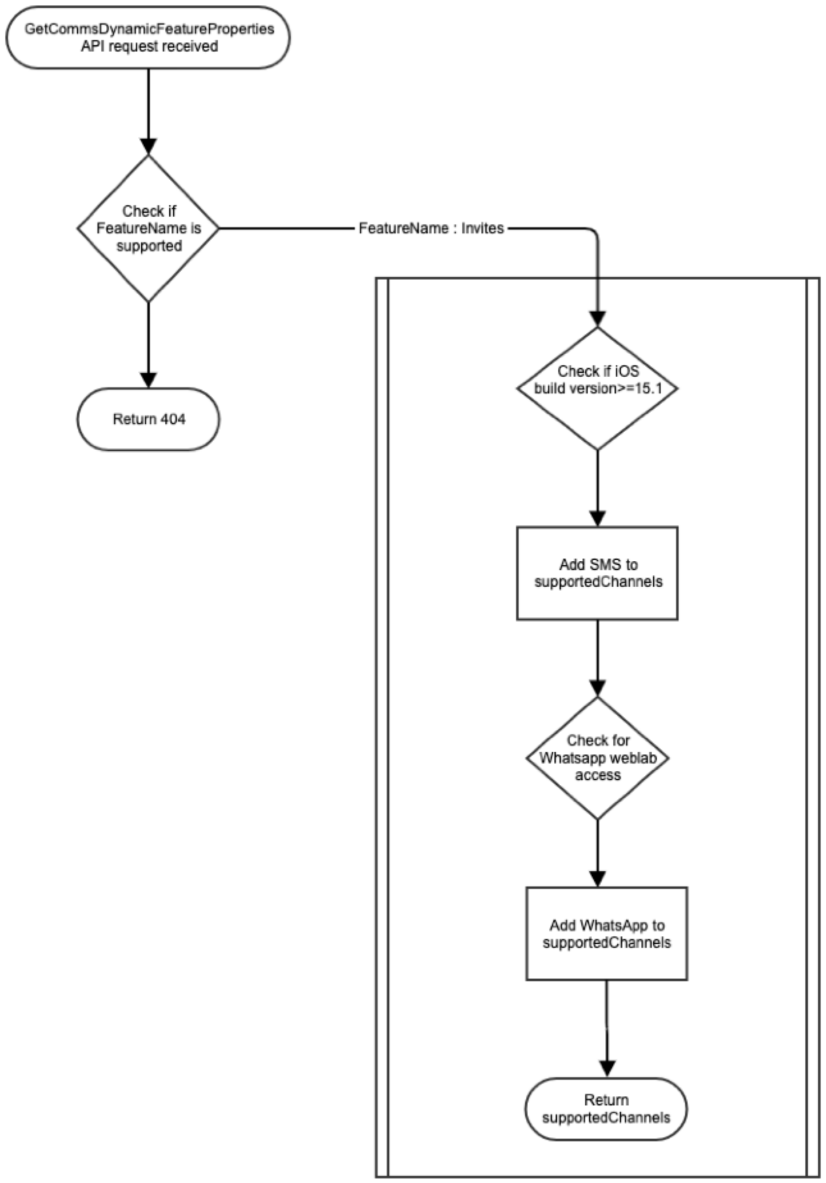
Comparison of analysed options

Property	GetInvitationChannels API	GetCommsFeatureDynamicProperties API
Invitation channels moved to cloud	✓	✓
No multiple branching logics for each feature	✓	✗
Comms-wide reusability	✗	✓
Reduced dependency on App side logic for dynamic properties channels	✗	✓
Extendable for other use cases	✗	✓

Block diagram

Service : ContactsGatewayService

API : GetCommsFeatureDynamicProperties


GetCommsDynamicFeatureProperties API request received
Check if FeatureName is supported
FeatureName : Invites
Return 404
Check if iOS build version>=15.1
Add SMS to supportedChannels
Check for Whatsapp weblab access
Add WhatsApp to supportedChannels
Return supportedChannels

Real time flow

From react native code base, AlexaMobileCommsReactNative library would call the above API in places when Invite option is clicked.

List of places where invite option is present :

- Contact card
- New call page

The list of channels returned by this API would be used to populate the bottom sheet being shown in app when user taps on Invite option.

User would be able to invite via the chosen channel.

Inference

The option 2 of GetCommsFeatureDynamicProperties API is more scalable than option 1. This control of having configurable options in backend instead of app code base can be used to mitigate app related issues in a quick time span, since the control logic now resides in backend where we can do an emergency deployment to mitigate issues instead of completely dialling down Invites and waiting till subsequent app release for the fix to be reached to customers.