Designing an Intent-based API

Darryn Campbell
SW Architect, Zebra Technologies
@darryncampbell

October 24th 2019

Designing an Intent-based API

Who am I?

- Developer advocate / Software architect for Zebra Technologies
  - Android OEM developing task specific devices
- Responsible for our Android developer kits & APIs

@darryncampbell | darryncampbell.co.uk

Designing an Intent-based API

Problem statement

ACME inc. wants to release their next, great device-side API

And so…

Want application developers to be able to use the API

It therefore follows that…

The more developers can use the API, the better

Therefore…

The more languages, frameworks, paradigms we can support, the better

Achieved by a prioritized backlog and…

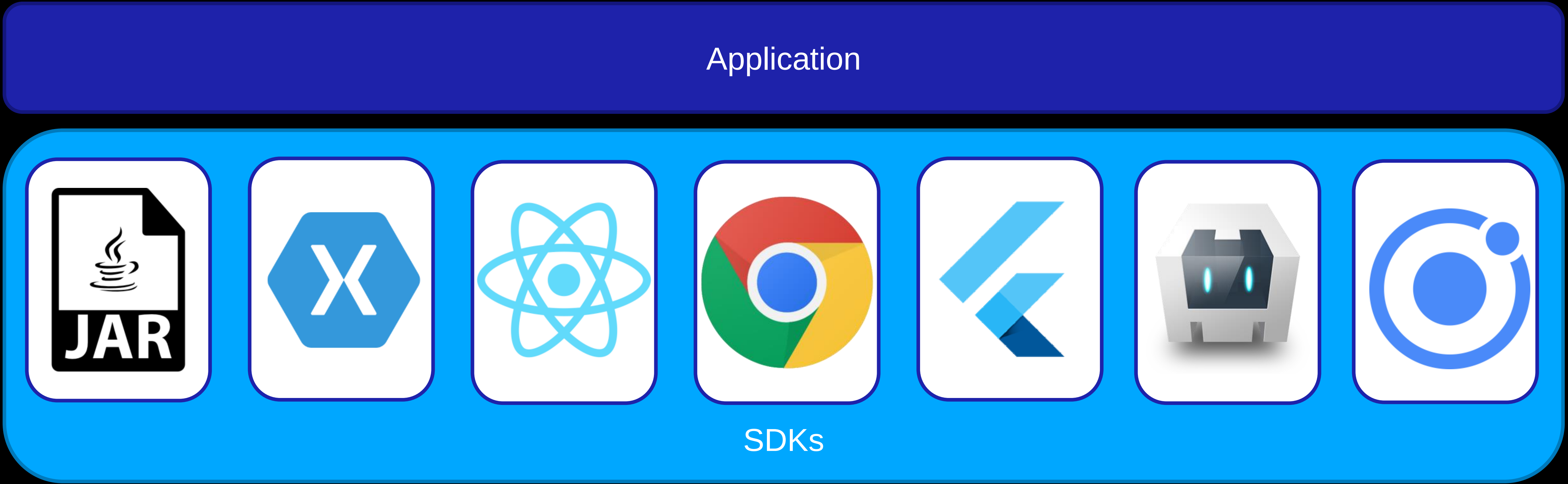
A fully resourced team adding new features each release

with minimal distraction from bugs

Designing an Intent-based API

Problem statement

- There are a few of options:
  - – Support languages one by one and provide SDKs for each



Designing an Intent-based API

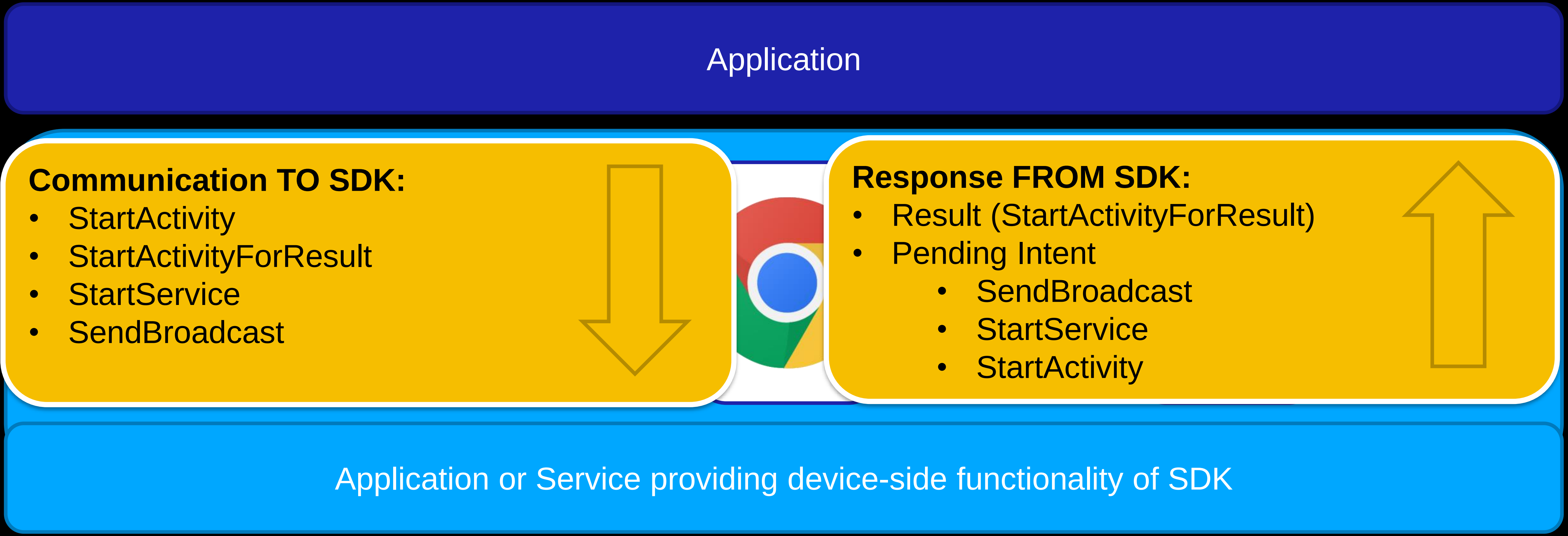
Problem statement

- There are a few of options:
  - – Support languages one by one and provide SDKs for each
- Move everything to the cloud
  - – Did somebody say SDKaaS?
  - – Expose a REST API
- Introducing… An Intent-based approach

Designing an Intent-based API

Proposed Solution

Introducing an Intent-based approach



Application

Communication TO SDK:

- StartActivity
- StartActivityForResult
- StartService
- SendBroadcast

Response FROM SDK:

- Result (StartActivityForResult)
- Pending Intent
    - SendBroadcast
    - StartService
    - StartActivity

Application or Service providing device-side functionality of SDK

Designing an Intent-based API

Proposed Solution

- Advantages of this approach
  - Your API is instantly accessible by any application development paradigm that supports Intents
    - Android (Java), Android (Kotlin), Xamarin
  - Many frameworks will support Android through community generated plugins
    - Flutter, Cordova, Ionic, React Native
  - Chrome supports Intents, in a limited capacity
    - Both sending and receiving, with the appropriate schemes
  - Intents are Industry standard
    - Communication is bi-directional and secure
  - Frameworks come and go every day
    - How do you know what the next big framework will be?
    - This way early framework adopters can usually experiment with your APIs with frameworks you haven’t even heard of yet.

Designing an Intent-based API

Proposed Solution – Simple examples – Application providing the functionality

- Simple example supports StartActivity and StartActivityForResult
- Identical to how an application would expose functionality to other applications

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    if (intent.hasExtra("JokeRequest"))
    {
        showJoke(giveJoke())
        finish()
    }

}
```

Joke Service

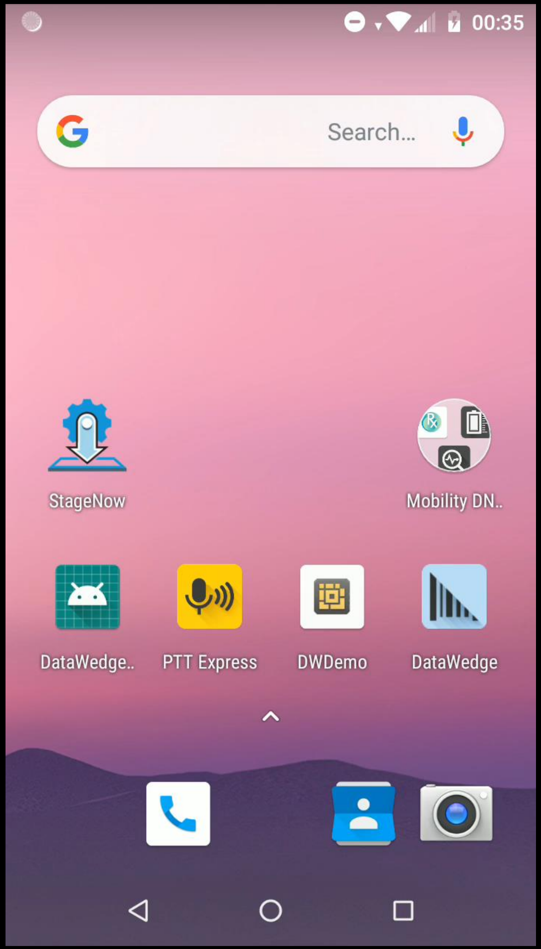
```
fun showJoke(joke: String)
{
    Toast.makeText(this, joke, Toast.LENGTH_LONG).show()

    val jokeIntent = Intent()
    jokeIntent.putExtra("theJoke", joke)

    setResult(RESULT_OK, jokeIntent)}
```

Designing an Intent-based API

Proposed Solution – Simple example with Kotlin

- Native applications in Kotlin / Java can invoke the API
- Note: Hardcoded strings are potential for error
- Note: Use of a custom extra to differentiate the Intent
- Note: Simple implementation for calling apps

```
val requestJokeIntent = Intent()
requestJokeIntent.setClassName(
    "com.darryncampbell.jokegenerator",
    "com.darryncampbell.jokegenerator.MainActivity")
requestJokeIntent.putExtra("JokeRequest", "")

startActivityForResult(requestJokeIntent, 0)
```

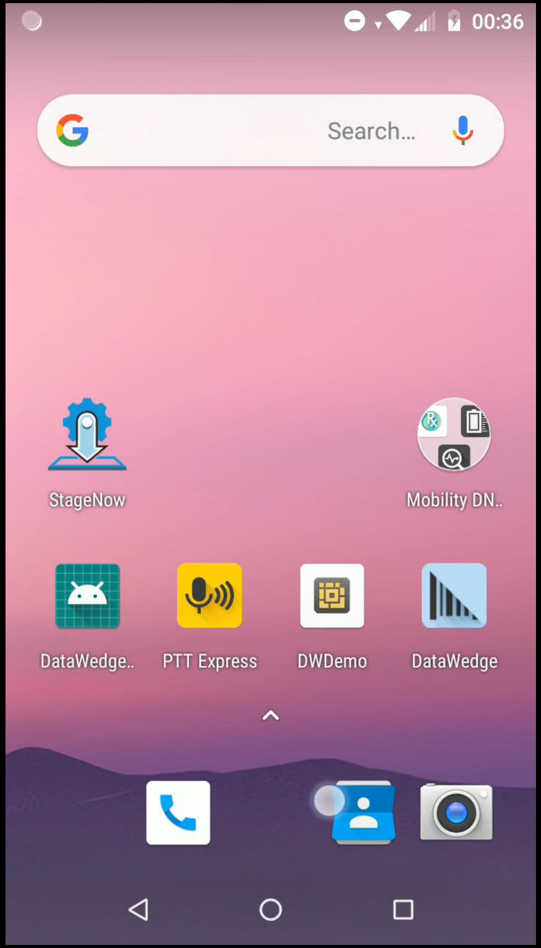


Designing an Intent-based API

Proposed Solution – Simple example with Xamarin

- Native applications in Xamarin can invoke the API
- Note: Hardcoded strings are potential for error
- Note: Use of a custom extra to differentiate the Intent
- Note: Simple implementation for calling apps

```
Intent requestJokeIntent = new Intent();
requestJokeIntent.SetClassName(
      "com.darryncampbell.jokegenerator",
      "com.darryncampbell.jokegenerator.MainActivity");
requestJokeIntent.PutExtra("JokeRequest", "");
StartActivityForResult(requestJokeIntent, 100);
```



Designing an Intent-based API

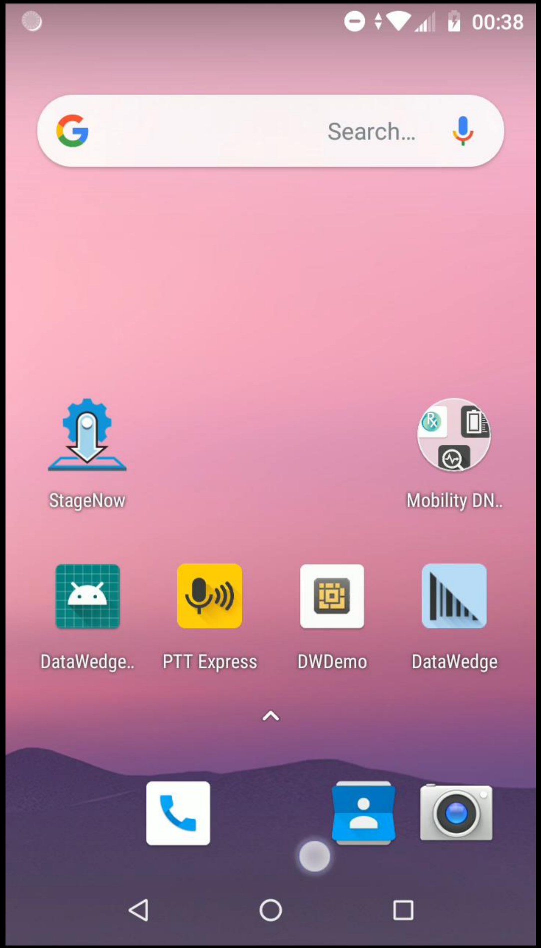
Proposed Solution – Simple example with Flutter

- Cross-platform applications in Flutter can invoke the API
- Note: Requires the use of a plugin
- Note: Plugin implementation of StartActivityForResult is not generic

```
dependencies:
  flutter:
    sdk: flutter
  intent: ^1.1.0
```

```
import 'package:intent/intent.dart';
```

```
Intent()
  ..setAction("android.intent.action.VIEW")
  ..setData(Uri(scheme: "flutterjoke", path: "joke"))
  ..putExtra("JokeRequest", "")
  ..startActivity().catchError((e) => print(e));
```

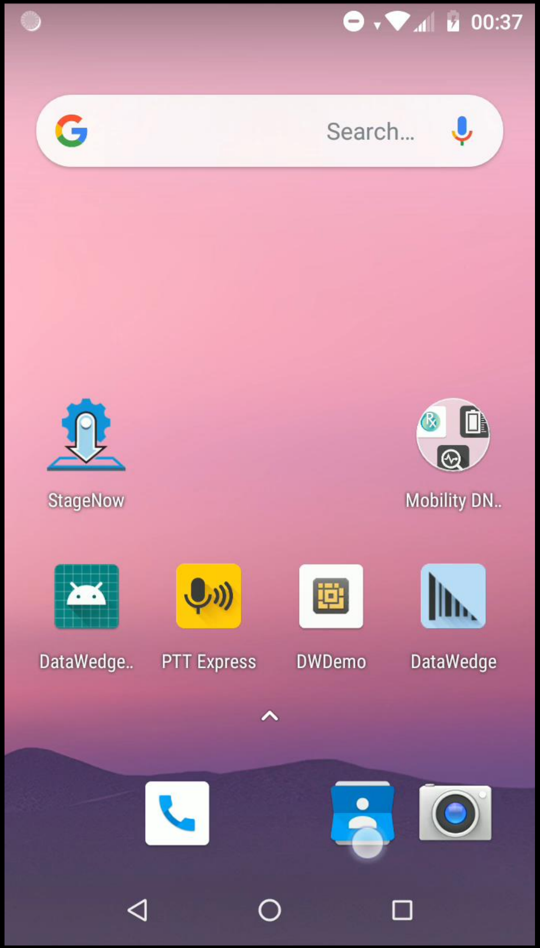


Designing an Intent-based API

Proposed Solution – Simple example with Chrome (JavaScript)

- Web based applications can invoke the API
- Note: Demo uses Chrome Android Intents
- Note: Receiving application requires custom scheme

```
<a href="
intent://joke/#Intent;
scheme=dccjoke;
package=com.darryncampbell.jokegenerator;
S.JokeRequest=1;end">
<button>Tell a Joke</button></a>
```



Designing an Intent-based API

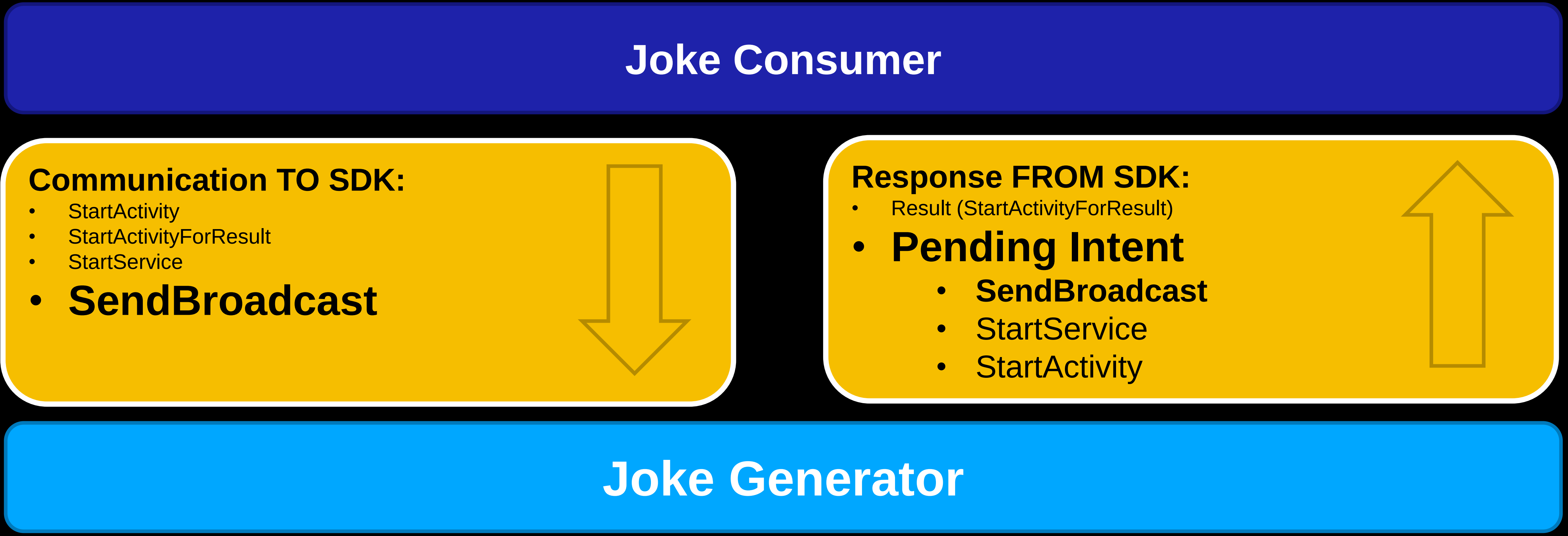
Proposed Solution - Simple example - Summary

- "Most flexible" approach
  - Most approaches developers take will support this either natively or through existing Intent plugins
  - Not demoed here: React Native, Cordova, Ionic, others but they would work equally well
- "Least flexible" approach
  - As in the case of Chrome, you saw the joke generator application launch briefly in the background
  - Does not support callbacks to the originator app beyond StartActivityForResult

Designing an Intent-based API

More complex Example

- Using Pending Intents and receiving requests via a Broadcast Receiver



Joke Consumer

Communication TO SDK:
- StartActivity
- StartActivityForResult
- StartService
- **SendBroadcast**

Response FROM SDK:
- Result (StartActivityForResult)
- **Pending Intent**
  - **SendBroadcast**
  - StartService
  - StartActivity

Joke Generator

Designing an Intent-based API

Proposed Solution – Feature-rich examples – App providing the functionality

- Functionality providing application receives explicit broadcast intents containing a Pending Intent (from the caller), this PI is provided with a joke and invoked

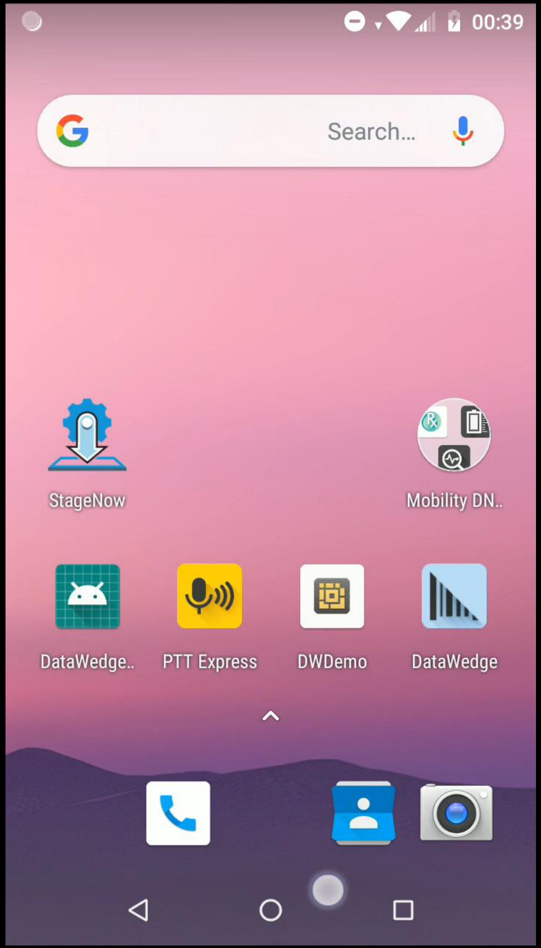
```
class JokeReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
    if (intent.hasExtra("callbackPI"))
        {
            val piResponse = intent.getParcelableExtra<PendingIntent>("callbackPI")
            val randomNumber = (Math.random () * Jokes.jokes.size).toInt()
            val jokeString = Jokes.jokes.get(randomNumber)
            val jokeIntent = Intent()
            jokeIntent.putExtra("joke", jokeString)
            piResponse.send(context, 0, jokeIntent, null, null)
        }
    }
}
```

Designing an Intent-based API

Proposed Solution – Feature-rich example with Kotlin

- Note: Harder for the caller to construct
- Note: Hardcoded strings
- Note: PI could send broadcast or start activity / service

```kotlin
val requestJokeIntent = Intent()
requestJokeIntent.setClassName(
    "com.darryncampbell.jokegenerator",
    "com.darryncampbell.jokegenerator.JokeReceiver")
val responseIntent = Intent(this,
    JokeConsumerBroadcastReceiver::class.java)
val piResponse = PendingIntent.getBroadcast(
    applicationContext, 0, responseIntent, 0)
requestJokeIntent.putExtra("callbackPI", piResponse)
sendBroadcast(requestJokeIntent)
```

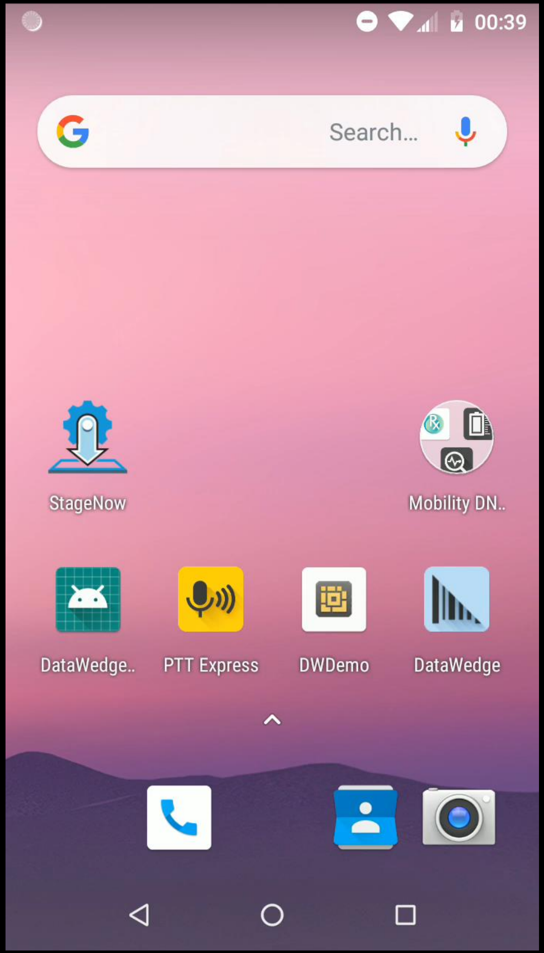


Designing an Intent-based API

Proposed Solution – Feature-rich example with Xamarin

- Note: Harder for the caller to construct
- Note: Hardcoded strings
- Note: PI could send broadcast or start activity / service

```
Intent requestJokeIntent = new Intent();
requestJokeIntent.SetClassName(
  "com.darryncampbell.jokegenerator",
  "com.darryncampbell.jokegenerator.JokeReceiver");
Intent responseIntent = new Intent(this,
  Java.Lang.Class.FromType(typeof(
    JokeConsumerXamarinBroadcastReceiver)));
PendingIntent piResponse =
  PendingIntent.GetBroadcast(Application.Context, 0,
    responseIntent, 0);
requestJokeIntent.PutExtra("callbackPI", piResponse);
SendBroadcast(requestJokeIntent);
```



Designing an Intent-based API

Proposed Solution – Feature-rich example - Summary

- “Most flexible” approach
  - Developers can receive callbacks through Broadcast, Start Service or Start Activity
  - No concerns with having to handle Start Activity and finish() in the application providing the functionality.
- “Least flexible” approach
  - No cross-platform frameworks seem to support this level of complexity in their Intent Plugins (you could of course write or enhance an existing open source plugin)
  - More hardcoded strings and a more complex setup could potentially lead to developer frustration

Designing an Intent-based API

Common points about an Intent approach in general

- Maximum size of Intent payload is fixed (differs by OS version)
- In rare cases, performance may be an issue compared to a native call
- Still need to install another app – Other examples of this exist e.g. AR Core
- Possible to mis-copy hard coded strings - Places a lot of responsibility on the caller of the API
- Cross platform frameworks don't natively support Intents
  - Support for Intents in cross platform framework plugins can vary from platform to platform
  - Action View is most common but as previously stated, provides the least functionality
  - StartActivity does not support returning data
- Complex data structures can be returned through the Intent

Designing an Intent-based API

What we have done at Zebra

- At Zebra we have adopted the Pending Intent model as the most flexible approach.
  - – Starting with our IrDA APIs (interfacing with legacy hardware)
  - – Link to APIs available on the Resources slide
- APIs consist of:
  - – An Action, consistently named
  - – A series of properties, on which you can call SET, GET and REGISTER
    - Changes REGISTERED properties are notified via Pending Intent
  - – A series of DO commands, to instruct the API to perform actions
    - Data is returned via Pending Intents

Designing an Intent-based API

Resources

- Applications shown in this presentation along with code samples:
  - https://github.com/darryncampbell/droidcon_london_2019
- Zebra Intent-based APIs:
  - https://techdocs.zebra.com/emdk-for-android/7-3/intents/
- Videos shown in this presentation:
  - https://www.youtube.com/watch?v=j_91hkgryS8&list=PLj8D9Diz5FBpJ2hZdI9NS_C_3enVRlbSY

Designing an Intent-based API

Questions?

http://developer.zebra.com

Designing an Intent-based API

This presentation will soon be available on the droidcon London website at the following link:

https://skillsmatter.com/conferences/11785-droidcon-london-2019#skillscasts

http://developer.zebra.com

Thank You

ZEBRA and the stylized Zebra head are trademarks of Zebra Technologies Corp., registered in many jurisdictions worldwide. All other trademarks are the property of their respective owners. ©2019 Zebra Technologies Corp. and/or its affiliates. All rights reserved.