


## description: Android'de ön planda çalışan servisler

### Foreground Service

#### Temel Hususlar

Kullanıcının bildirim veya arayüz ile haberi olan arkaplan görevleridir

- Önceli servislerdir ve öncelik seviyesi bildirilmelidir
- Kullanıcıya **kaldırılmayan bir bildirim** gösterilmesi zorunludur
- Kullanıcının arkaplan işlemlerinden haberdar olması amaçlanır
- Servisin çalıştırılması için **FOREGROUND\_SERVICE** iznine ihtiyaç duyulur
  - Android'in **izin isteme hiyerarşisine** uygun ilerler
  - İzin alınmadığı takdirde **SecurityException** hatası verir

{% hint style="info" %}  Android dokümanında **Running a service in the foreground** alanında işlenmektedir {% endhint %}

#### Gerekli İzinleri Alma

- Android 8.0 ve sonrası için **FOREGROUND\_SERVICE** iznine ihtiyaç duyulur
- Örnek izin sistemi için alttaki kodu kullanabilirsiniz

```
static final int PERMISSION_FOREGORUND = 1;

void startTelemetryService() {
    if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.FOREGROUND_SERVICE) != PackageManager.PERMISSION_GRANTED) {
        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.FOREGROUND_SERVICE)) {
            Toast.makeText(this, "Haberleşme hizmeti için izne ihtiyaç vardır",
                Toast.LENGTH_SHORT).show();
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
                ActivityCompat.requestPermissions(this, new String[]
                    {Manifest.permission.FOREGROUND_SERVICE}, PERMISSION_FOREGORUND);
            }
        } else {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
                ActivityCompat.requestPermissions(this, new String[]
                    {Manifest.permission.FOREGROUND_SERVICE}, PERMISSION_FOREGORUND);
            }
        }
        startService(new Intent(this, TelemetryService.class));
    }
}
```

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
    if (requestCode == PERMISSION_FOREGORUND) {
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            startTelemetryService();
        }
    }
}
```

{% page-ref page="../temel/izinlerin-yonetimi.md" %}

## Temel Yapıyı Oluşturma

```
public class TelemetryService extends Service {
    private static final String TAG = "TelemetryService";

    private Looper telemetryLooper;
    private TelemetryHandler telemetryHandler;

    public TelemetryService() {
    }

    private final class TelemetryHandler extends Handler {
        public TelemetryHandler(Looper looper) {
            super(looper);
        }

        @Override
        public void handleMessage(@NonNull Message msg) {
            try {
                Log.i(TAG, "Mesaj alındı");
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // Kesme isteği geldiğinde isteği uygulama
                Thread.currentThread().interrupt();
            }
            stopSelf(msg.arg1);
        }
    }

    @Override
    public void onCreate() {
        Log.i(TAG, "Servis oluşturuldu");

        // Arkaplanda çalışacak thread'in tanımlanması ve başlatılması (UI thread'i
        bloklamaması lazım)
        HandlerThread thread = new HandlerThread("TelemetryStartArguments",
        Process.THREAD_PRIORITY_BACKGROUND);
        thread.start();
    }
}
```

```
telemetryLooper = thread.getLooper();
telemetryHandler = new TelemetryHandler(telemetryLooper);
}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.i(TAG, "Servis başlatıldı");

    Message msg = telemetryHandler.obtainMessage();
    msg.arg1 = startId; // İsteklerin yönetimi için kimlikleri saklamalıyız
    telemetryHandler.sendMessage(msg);

    // Eğer servis öldüyse, bu dönüşten sonra Intent'siz tekrar başlat
    return START_STICKY;
}

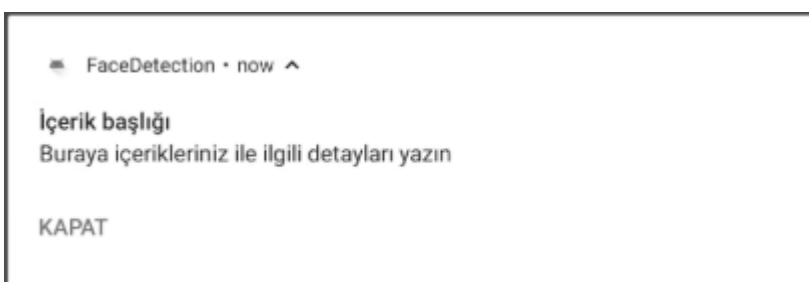
@Override
public IBinder onBind(Intent intent) {
    return null;
    // TODO: Return the communication channel to the service.
    // throw new UnsupportedOperationException("Not yet implemented");
}

@Override
public void onDestroy() {
    Log.i(TAG, "Servis kapatıldı");
}
}
```

{% hint style="info" %} ⓘ Ayrıntılı bilgi için [Create a Service](#) alanına bakabilirsiniz. {% endhint %}

## Bildirim Ekleme

- Android'in yeni gelen sürümleriyle beraber [NotificationChannel](#) yapısı gelmiştir
- Bu yapı ile her bildirim kategorilere ayrılmıştır
- Uygulama üzerindeki tüm bildirimleri susturmak yerine, belli başlı kategorileri susturma avantajı sağlar
- Kategorilere göre bildirim şekillerini düzenlemeye yardımcı olur



```
public class TelemetryService extends Service {
    private static final String TAG = "TelemetryService";

    private static final int REQUEST_SHOW_CONTENT = 0;
    private static final int REQUEST_STOP = 1;

    static final String ACTION_START_SERVICE = "Start telemetry service";
    static final String ACTION_STOP_SERVICE = "Stop telemetry service";

    @Override
    public void onCreate() {
        Log.d(TAG, "Servis oluşturuldu");
        startForeground();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        String intentAction = intent.getAction();
        if (intentAction != null) {
            switch (intentAction) {
                case TelemetryService.ACTION_START_SERVICE: {
                    Log.d(TAG, "Servis başlatıldı");
                    break;
                }
                case TelemetryService.ACTION_STOP_SERVICE: {
                    stopForegroundService();
                }
            }
        }

        // Eğer servis öldüyse, bu dönüşten sonra Intent'siz tekrar başlat
        return START_STICKY;
    }

    public void startForeground() {
        String channelId = "";
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            channelId = createNotificationChannel();
        }

        // Bildirime tıklandığında main uygulamayı açma isteği oluşturma
        Intent intent = new Intent(this, MainActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        PendingIntent contentIntent = PendingIntent.getActivity(this,
REQUEST_SHOW_CONTENT, intent, PendingIntent.FLAG_UPDATE_CURRENT);

        // Bildirim üzerinden servisi kapatma isteği oluşturma
        Intent stopSelf = new Intent(this, TelemetryService.class);
        stopSelf.setAction(TelemetryService.ACTION_STOP_SERVICE);
        PendingIntent pStopSelf;
        if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {
            pStopSelf = PendingIntent.getForegroundService(this, REQUEST_STOP,
```

```
stopSelf, PendingIntent.FLAG_CANCEL_CURRENT);
    } else {
        pStopSelf = PendingIntent.getService(this, REQUEST_STOP, stopSelf,
        PendingIntent.FLAG_CANCEL_CURRENT);
    }

    Notification notification = new NotificationCompat.Builder(this, channelId)
        .setSmallIcon(R.mipmap.face_deteciton)
        .setContentTitle("İçerik başlığı")
        .setContentText("Buraya içerikleriniz ile ilgili detyaları yazın")
        .setPriority(NotificationCompat.PRIORITY_MIN)
        .setCategory(NotificationCompat.CATEGORY_SERVICE)
        .setContentIntent(contentIntent)
        .setAutoCancel(true)
        .addAction(R.mipmap.ic_launcher, "Kapat", pStopSelf) // Kapatma
butonu ekleme
        .build();

    startForeground(101, notification);
}

@RequiresApi(Build.VERSION_CODES.O)
private String createNotificationChannel() {
    String channelId = "channel id";
    NotificationChannel channel = new NotificationChannel(channelId, "Bildirim
kategorisi ismi", NotificationManager.IMPORTANCE_DEFAULT);
    channel.setLightColor(Color.BLUE);
    channel.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);
    NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
    if (notificationManager != null) {
        notificationManager.createNotificationChannel(channel);
    }

    return channelId;
}

private void stopForegroundService() {
    Log.d(TelemetryService.TAG, "Servis sonlandırılıyor");

    stopForeground(true);
    stopSelf();
}

@Override
public void onDestroy() {
    Log.d(TAG, "Servis kapatıldı");
}
}
```

{% hint style="info" %}  Detaylı bilgiler için [Create Notification](#), [PendingIntent](#) alanlarına bakmanda fayda var {% endhint %}