Android Services & Local IPC: Programming Bound Services with Messengers (Part 2)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

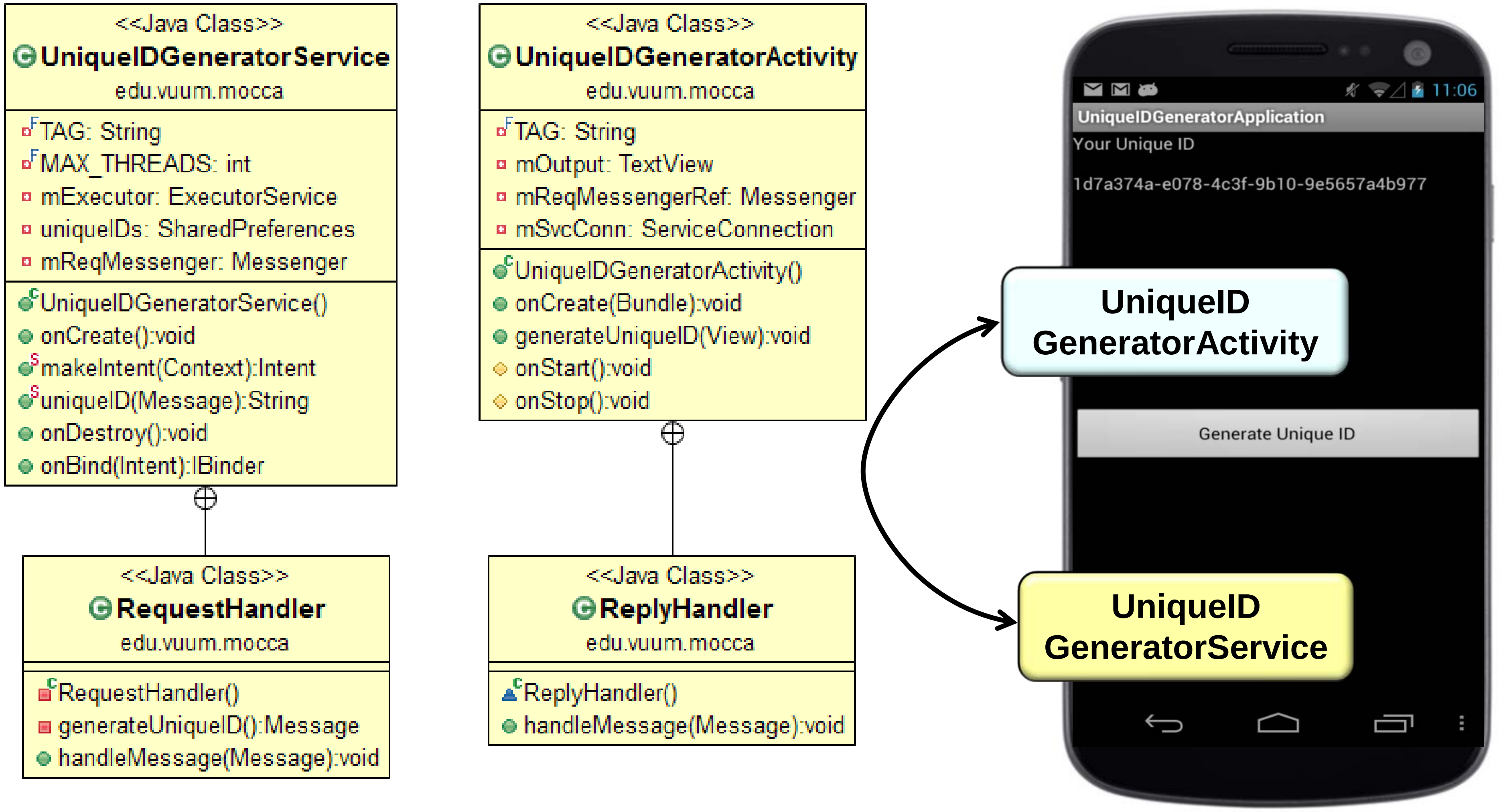
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Module

- Understand the implementation of the Unique ID Generator Application

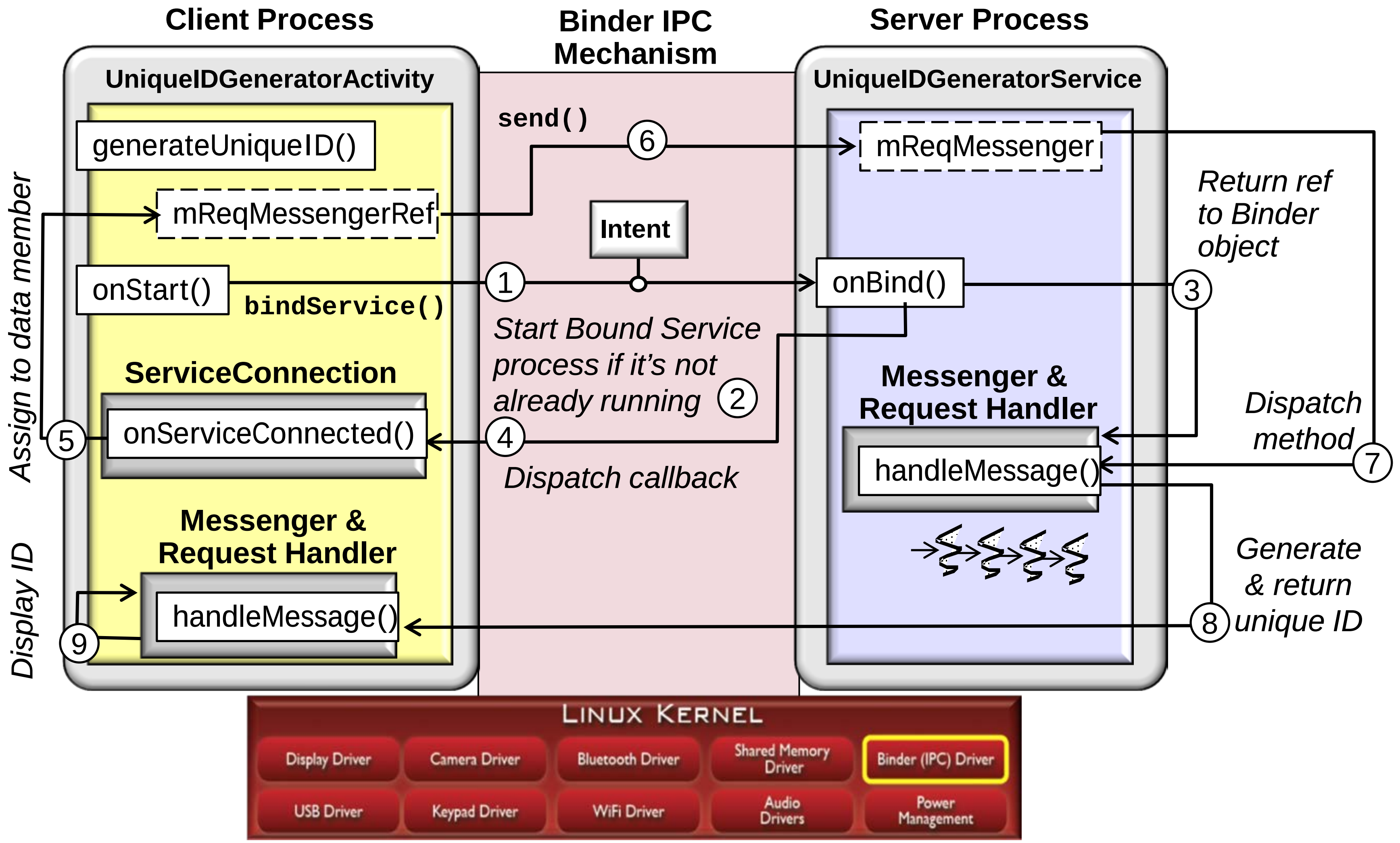


<<Java Class>>
UniqueIDGeneratorService
edu.vuum.mocca
TAG: String
MAX_THREADS: int
mExecutor: ExecutorService
uniqueIDs: SharedPreferences
mReqMessenger: Messenger
UniqueIDGeneratorService()
onCreate():void
makeIntent(Context):Intent
uniqueID(Message):String
onDestroy():void
onBind(Intent):IBinder
<<Java Class>>
UniqueIDGeneratorActivity
edu.vuum.mocca
TAG: String
mOutput: TextView
mReqMessengerRef: Messenger
mSvcConn: ServiceConnection
UniqueIDGeneratorActivity()
onCreate(Bundle):void
generateUniqueID(View):void
onStart():void
onStop():void
<<Java Class>>
RequestHandler
edu.vuum.mocca
RequestHandler()
generateUniqueID():Message
handleMessage(Message):void
<<Java Class>>
ReplyHandler
edu.vuum.mocca
ReplyHandler()
handleMessage(Message):void
11:06
UniqueIDGeneratorApplication
Your Unique ID
1d7a374a-e078-4c3f-9b10-9e5657a4b977
UniqueID GeneratorActivity
Generate Unique ID
UniqueID GeneratorService

See previous part on "Programming Bound Services with Messengers, Part 1"

Protocol for Bound Service Interactions

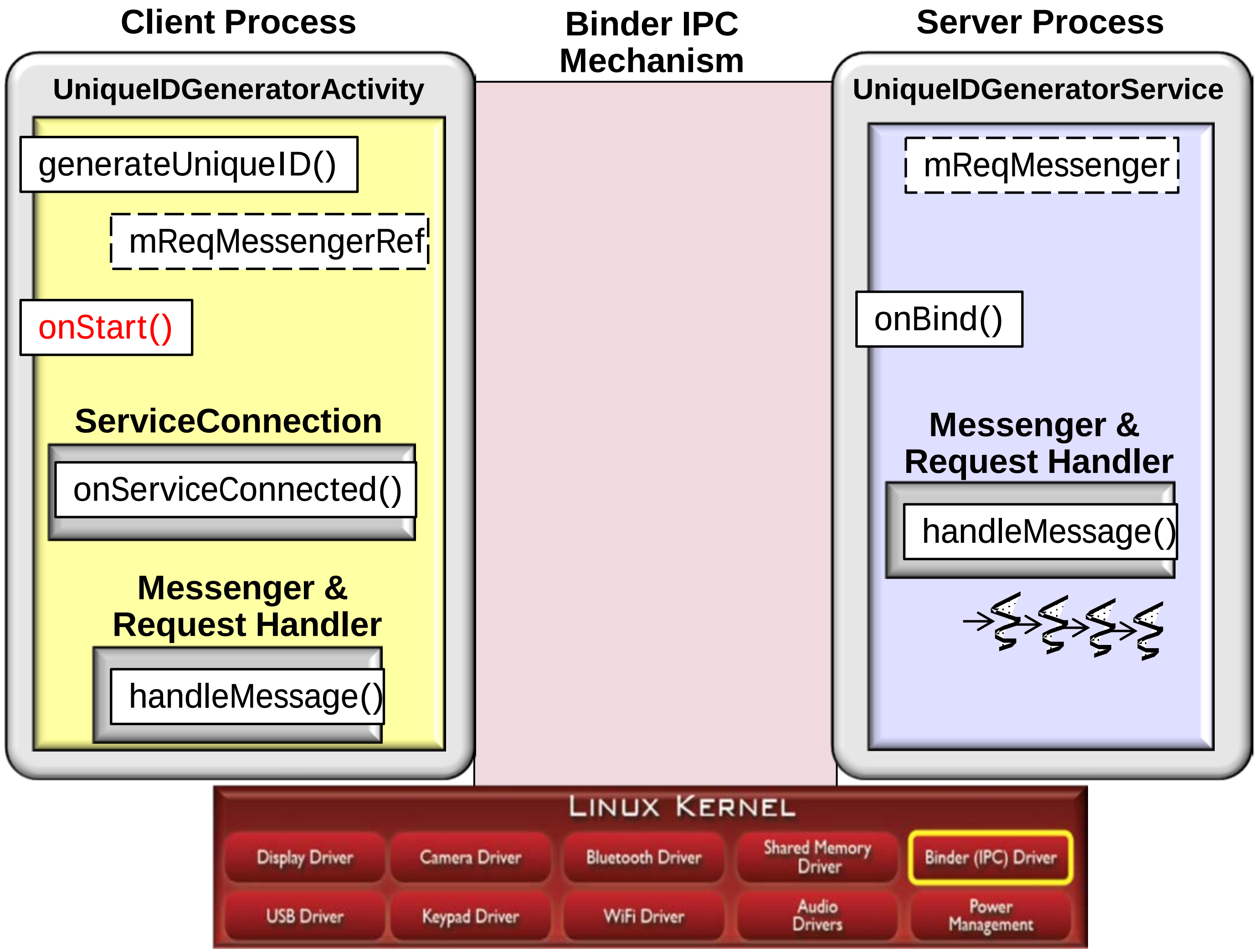
- A protocol is used to interact between Activities & Bound Services



We first examine the protocol for launching, connecting, & communicating with a Bound Service

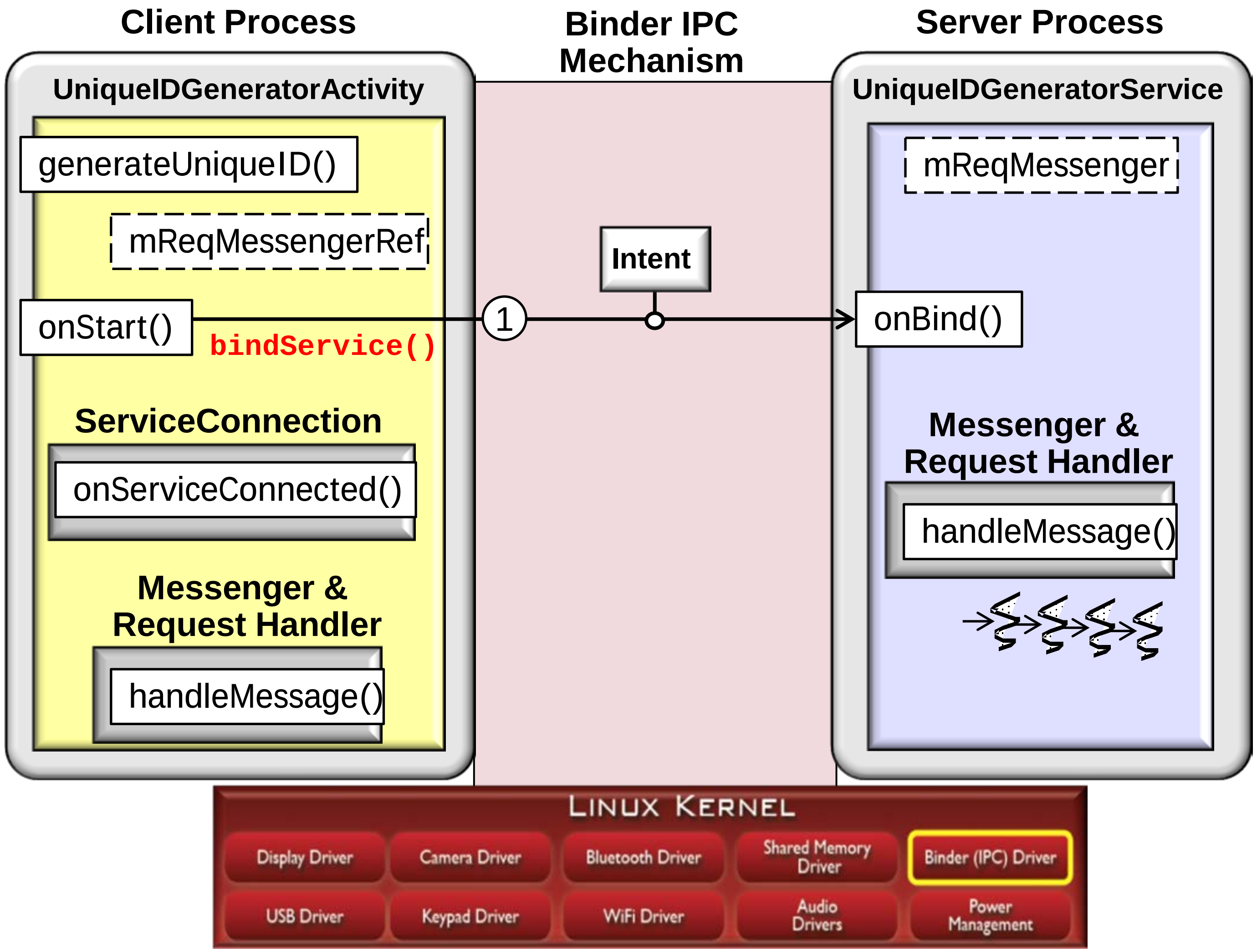
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



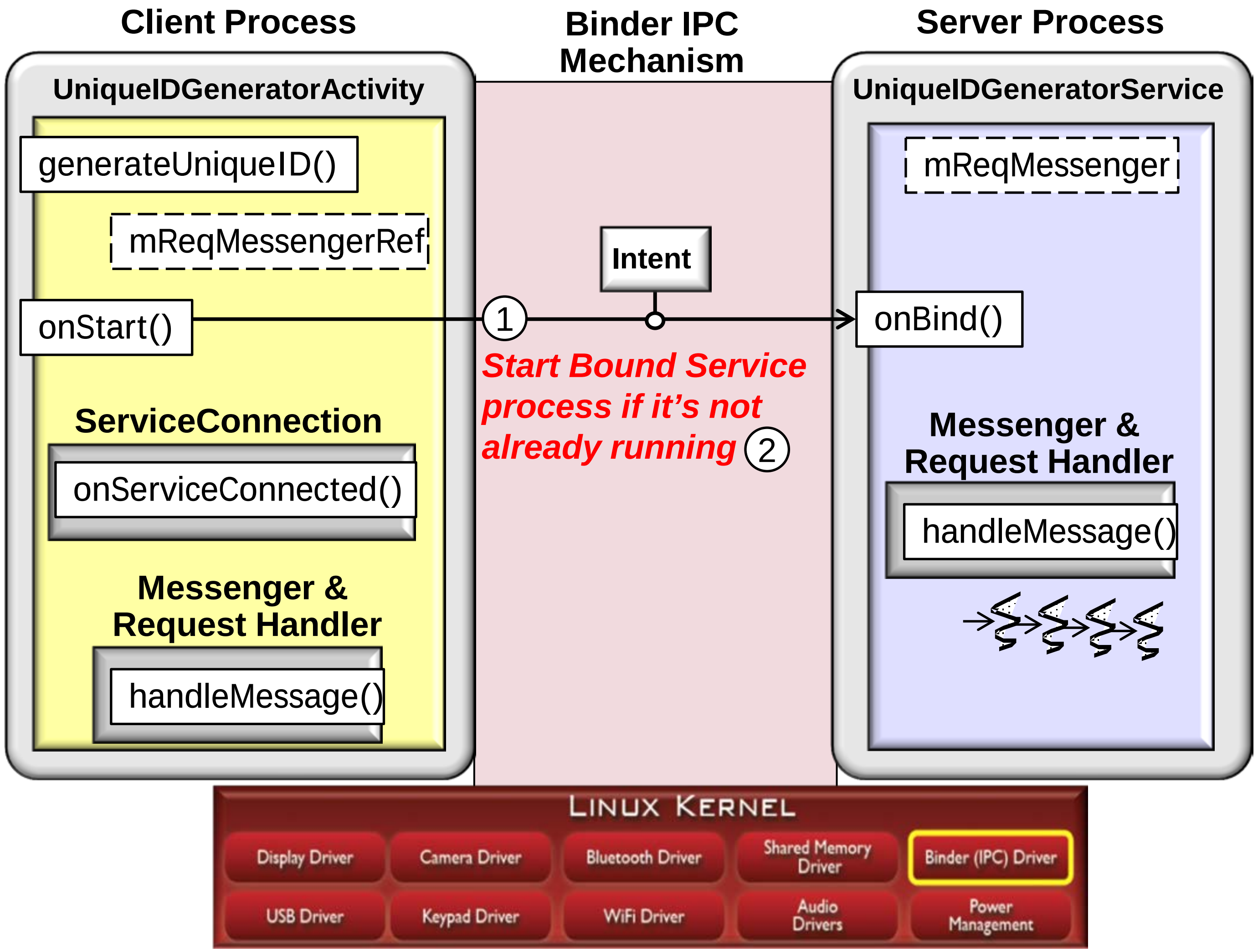
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



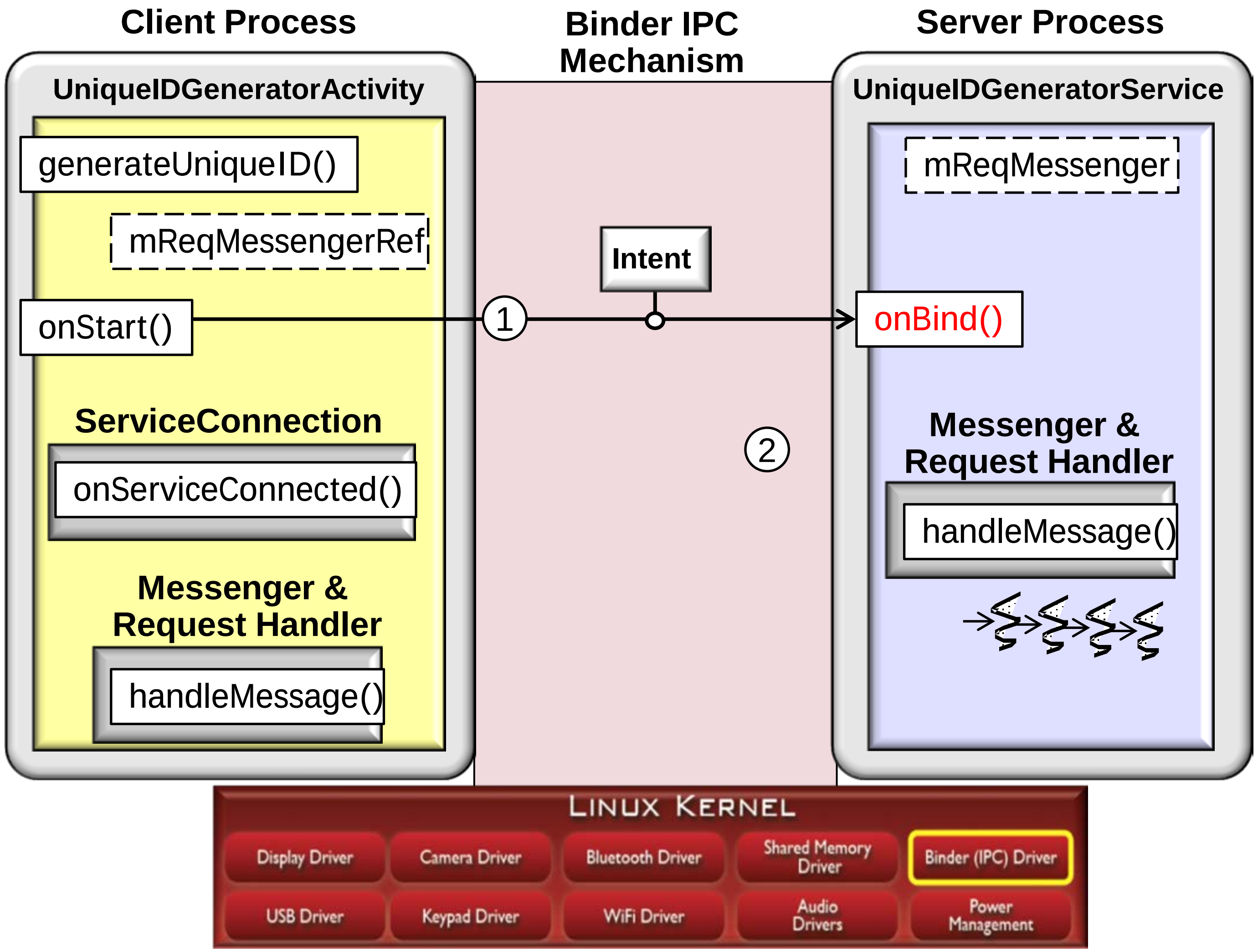
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



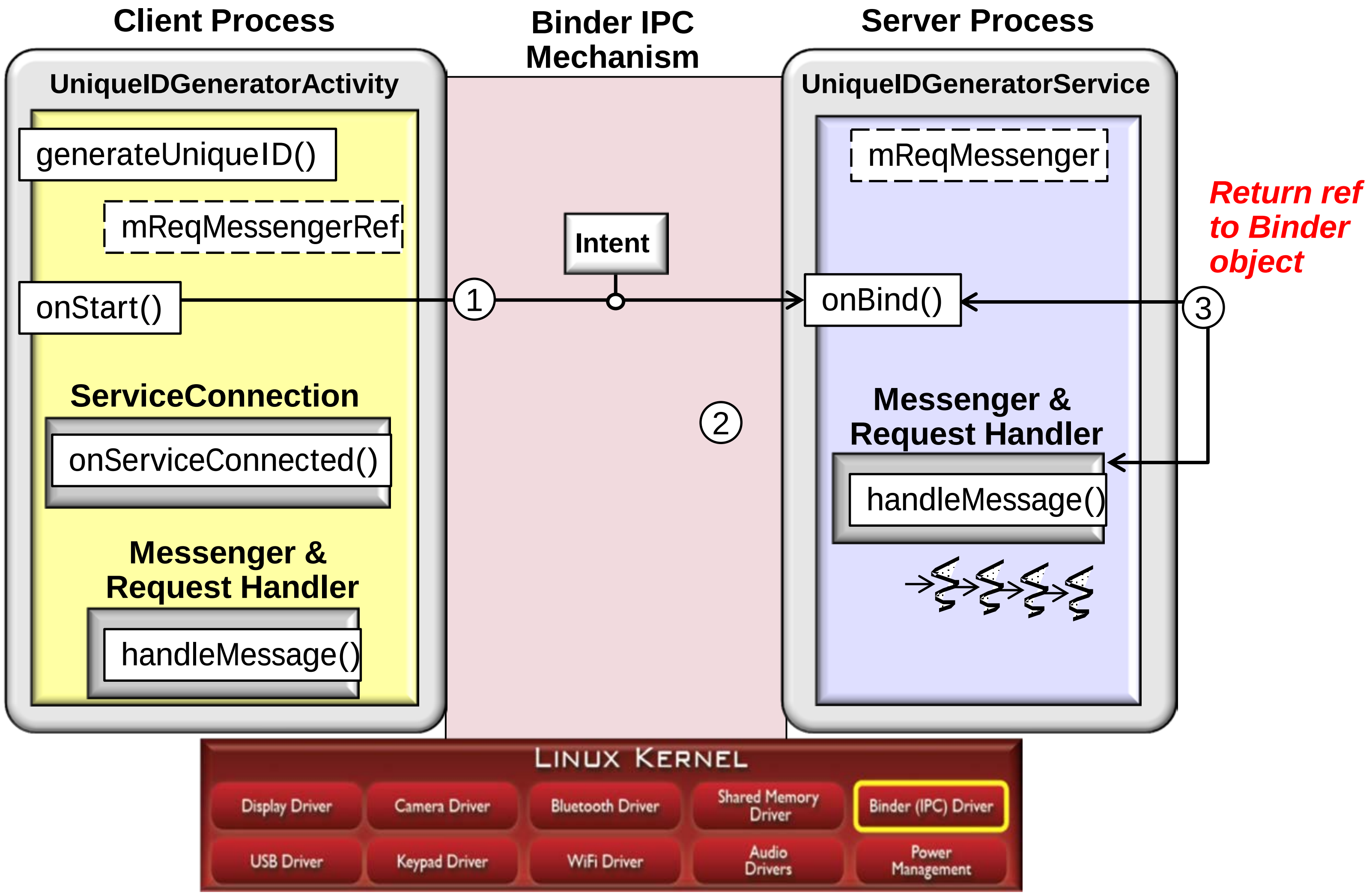
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



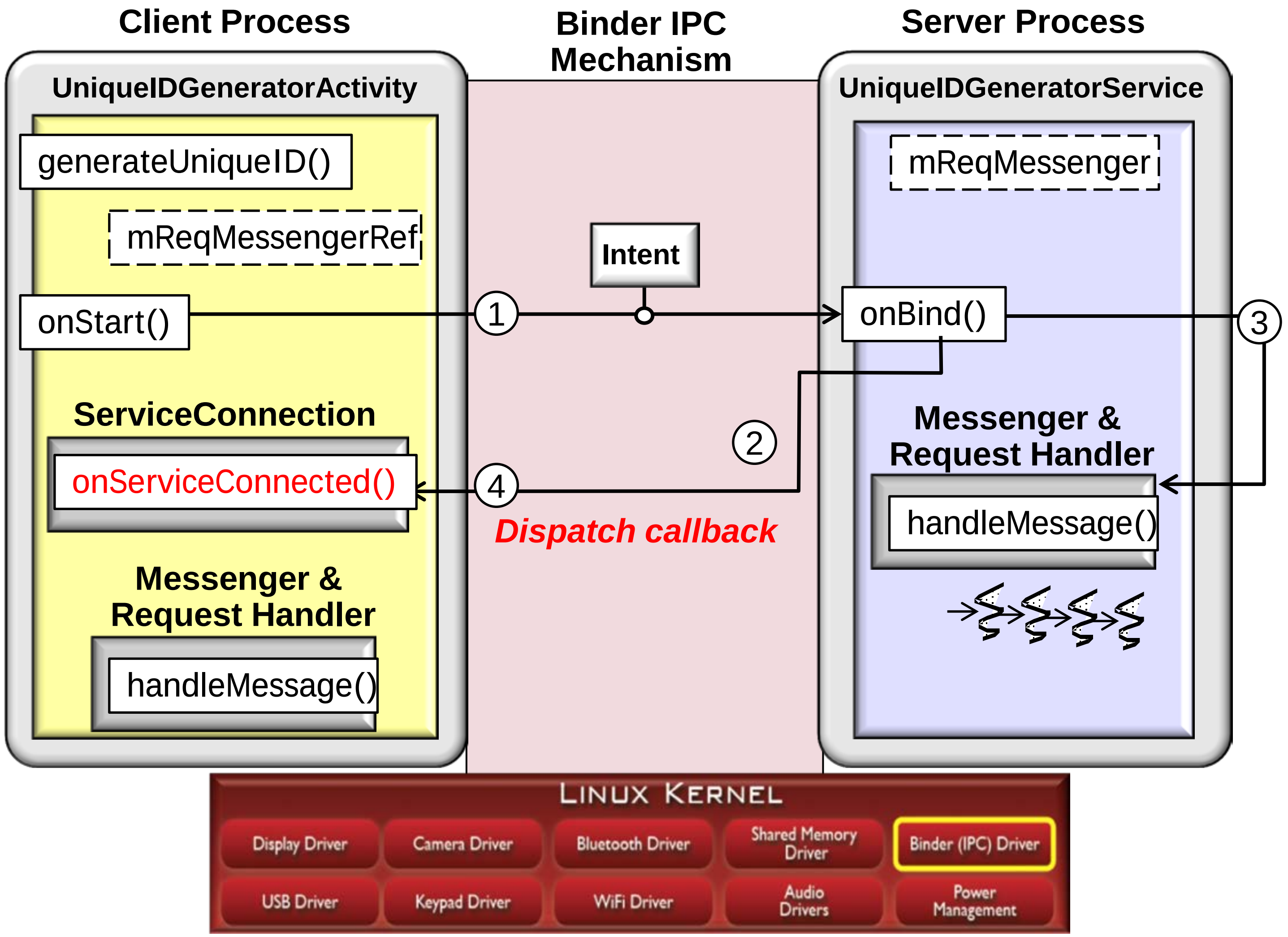
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



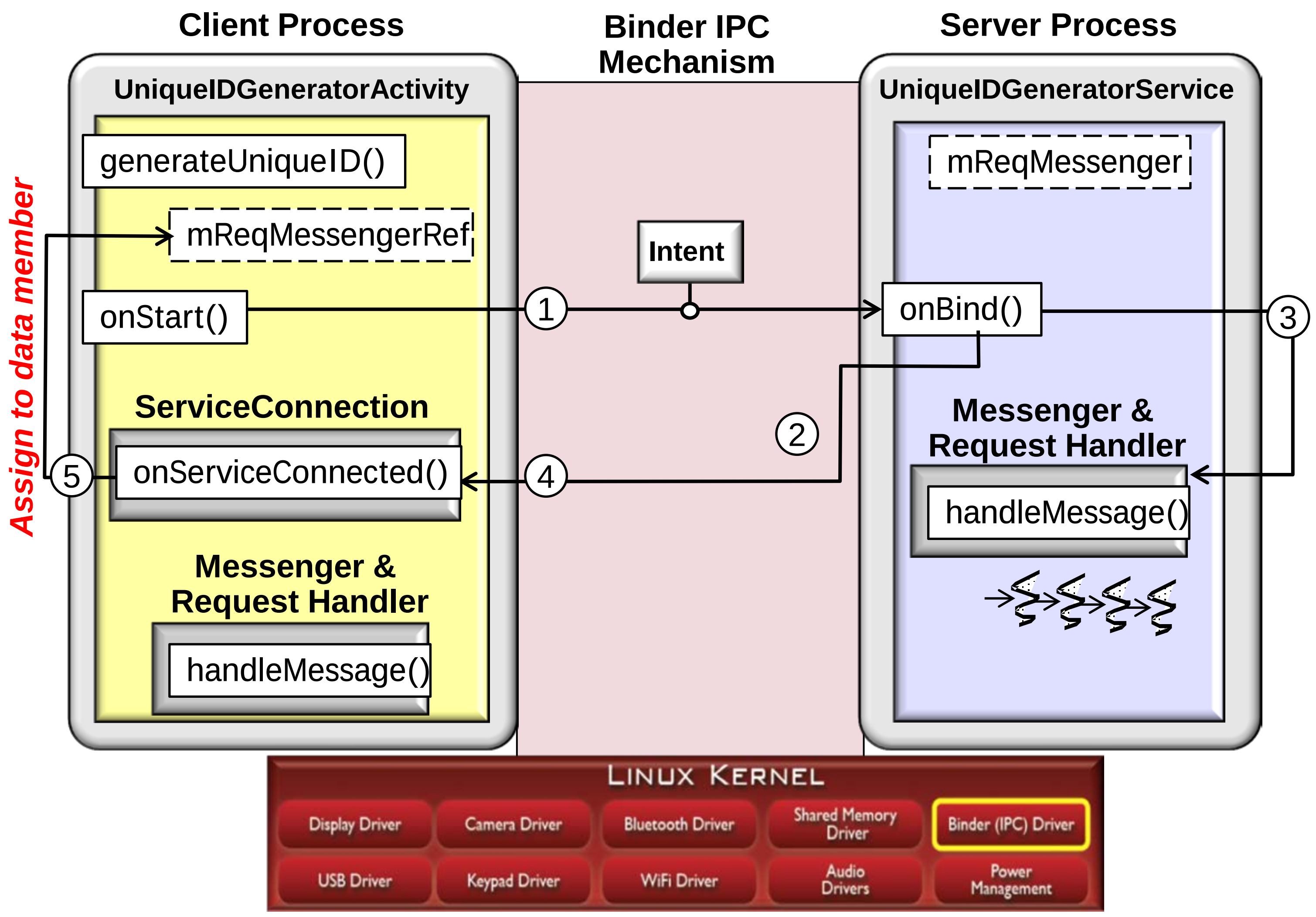
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



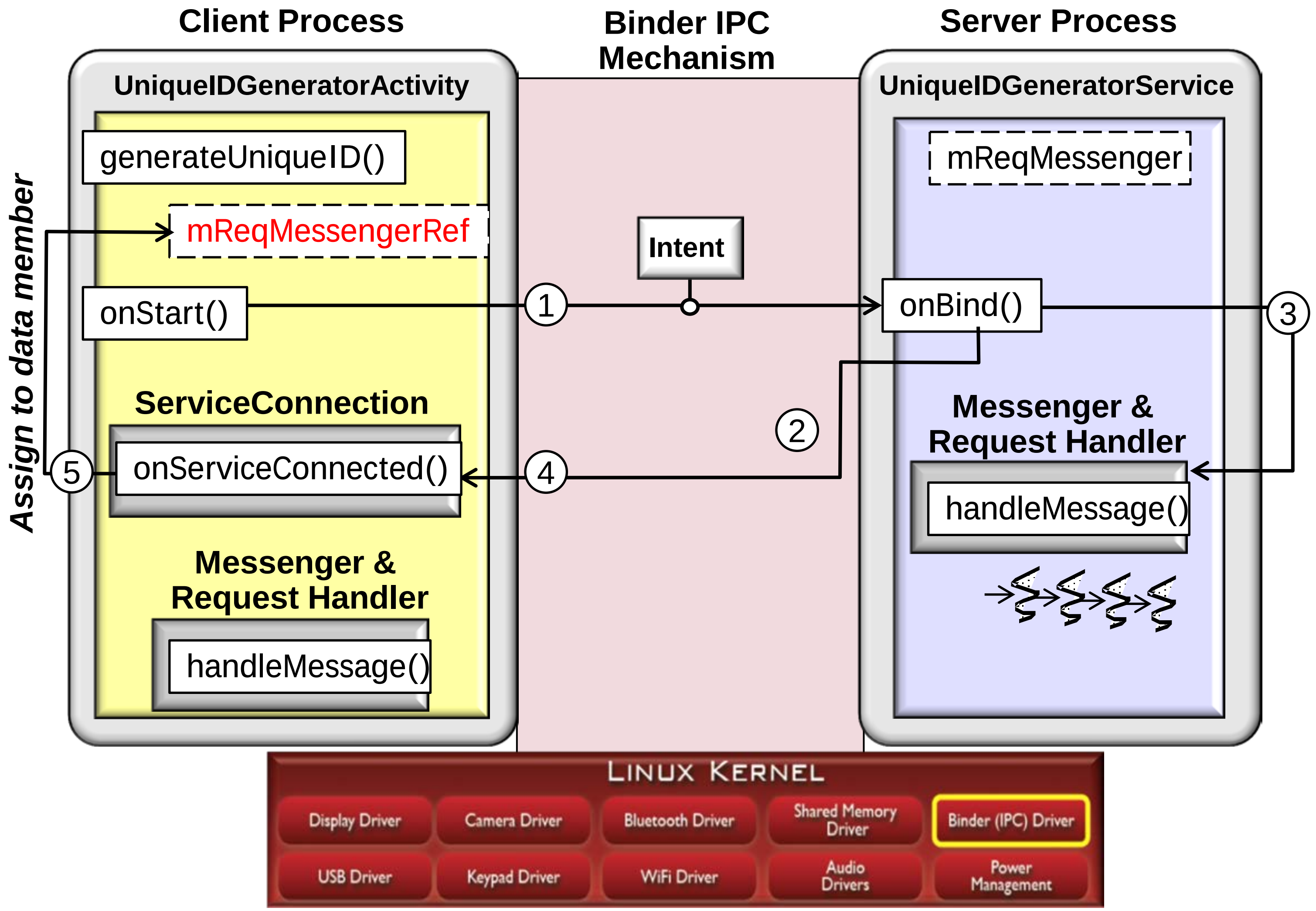
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



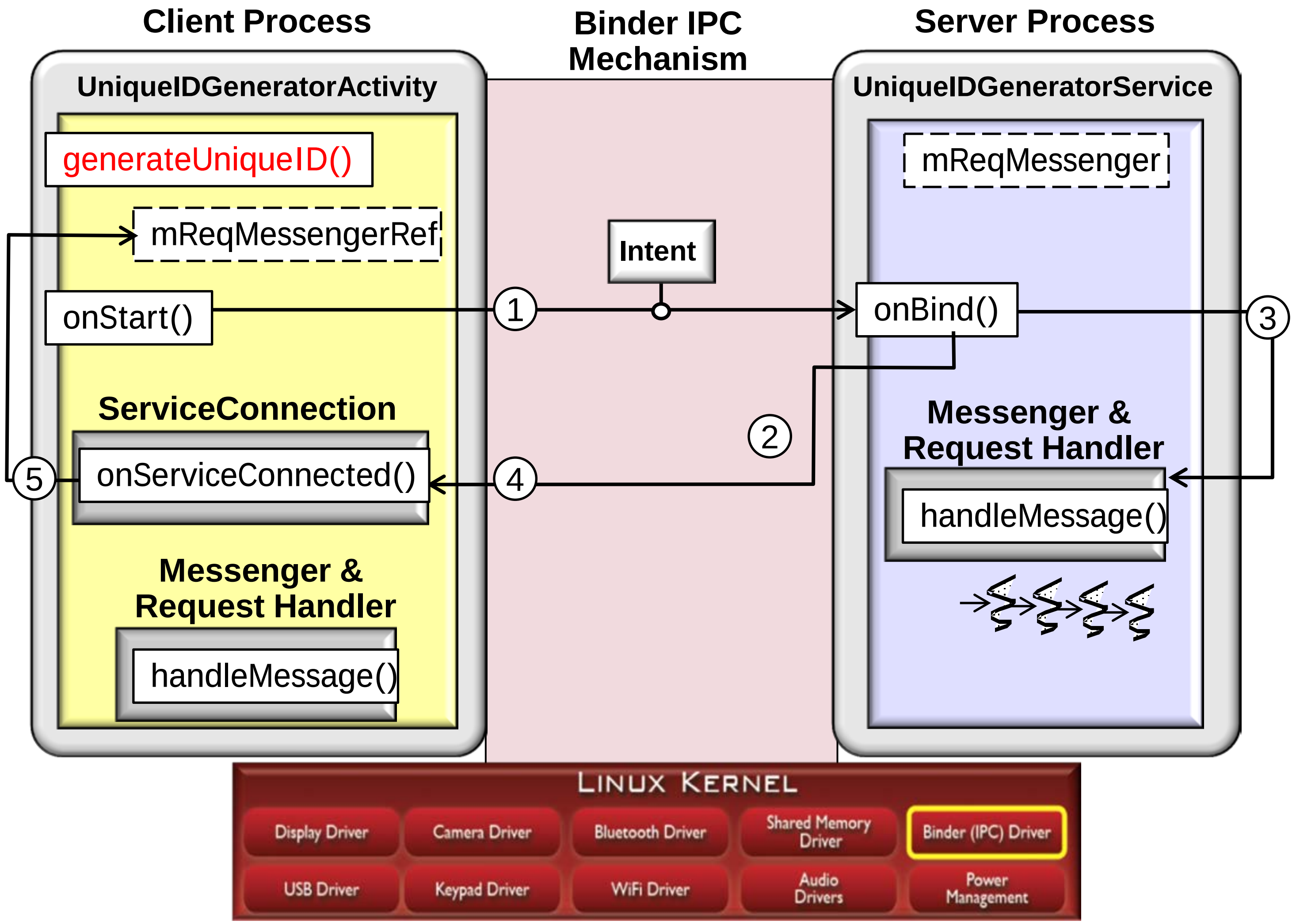
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



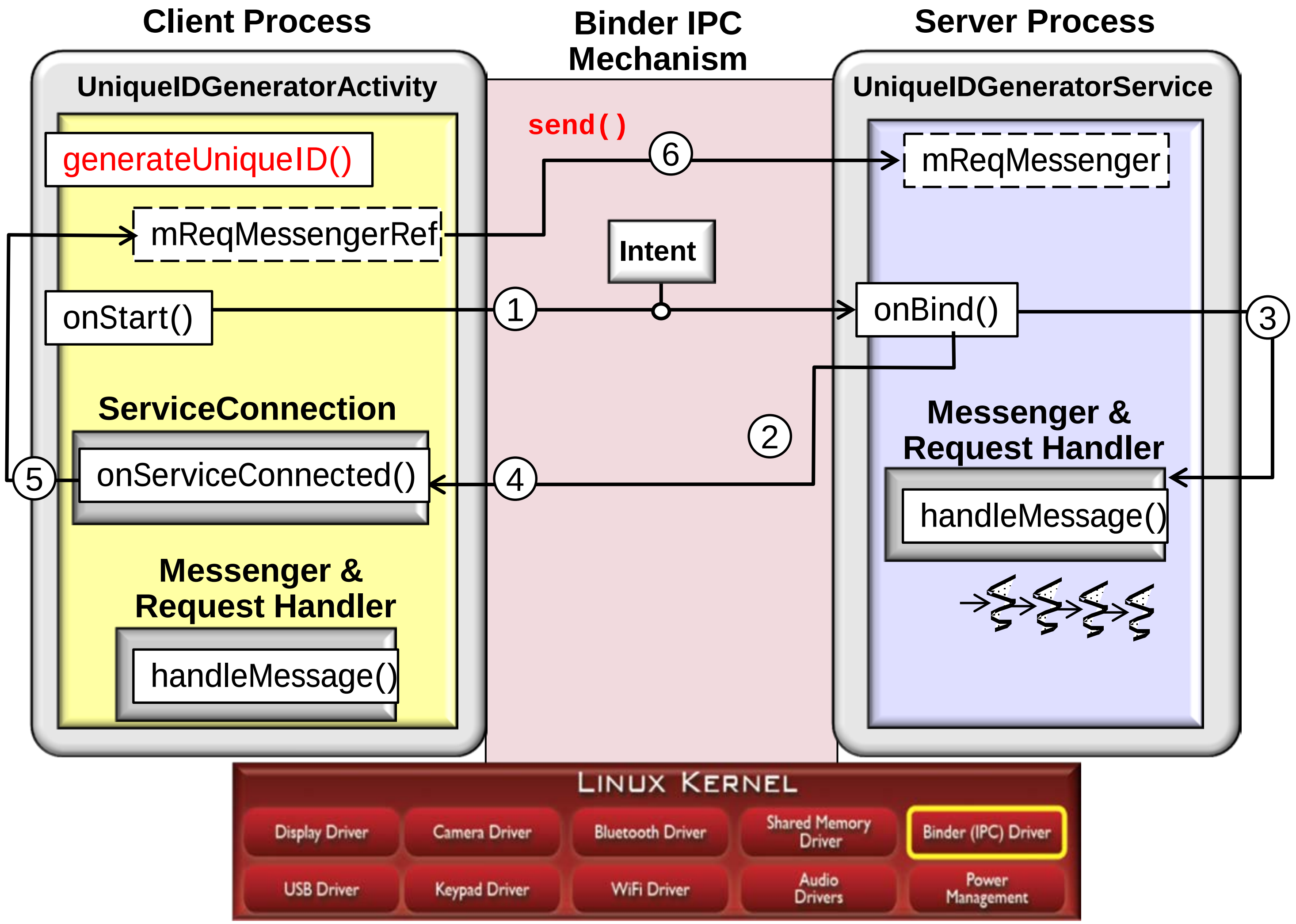
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



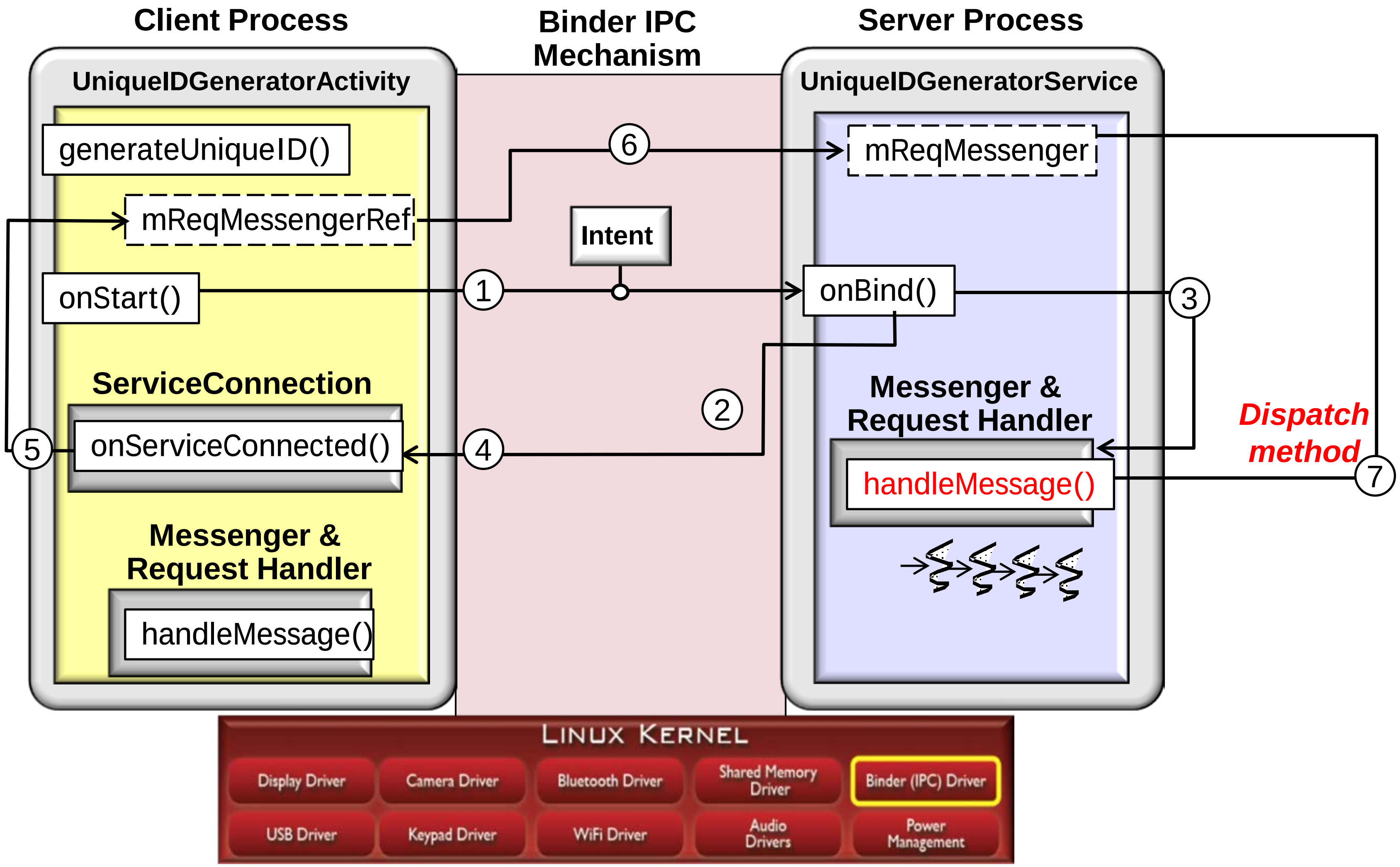
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



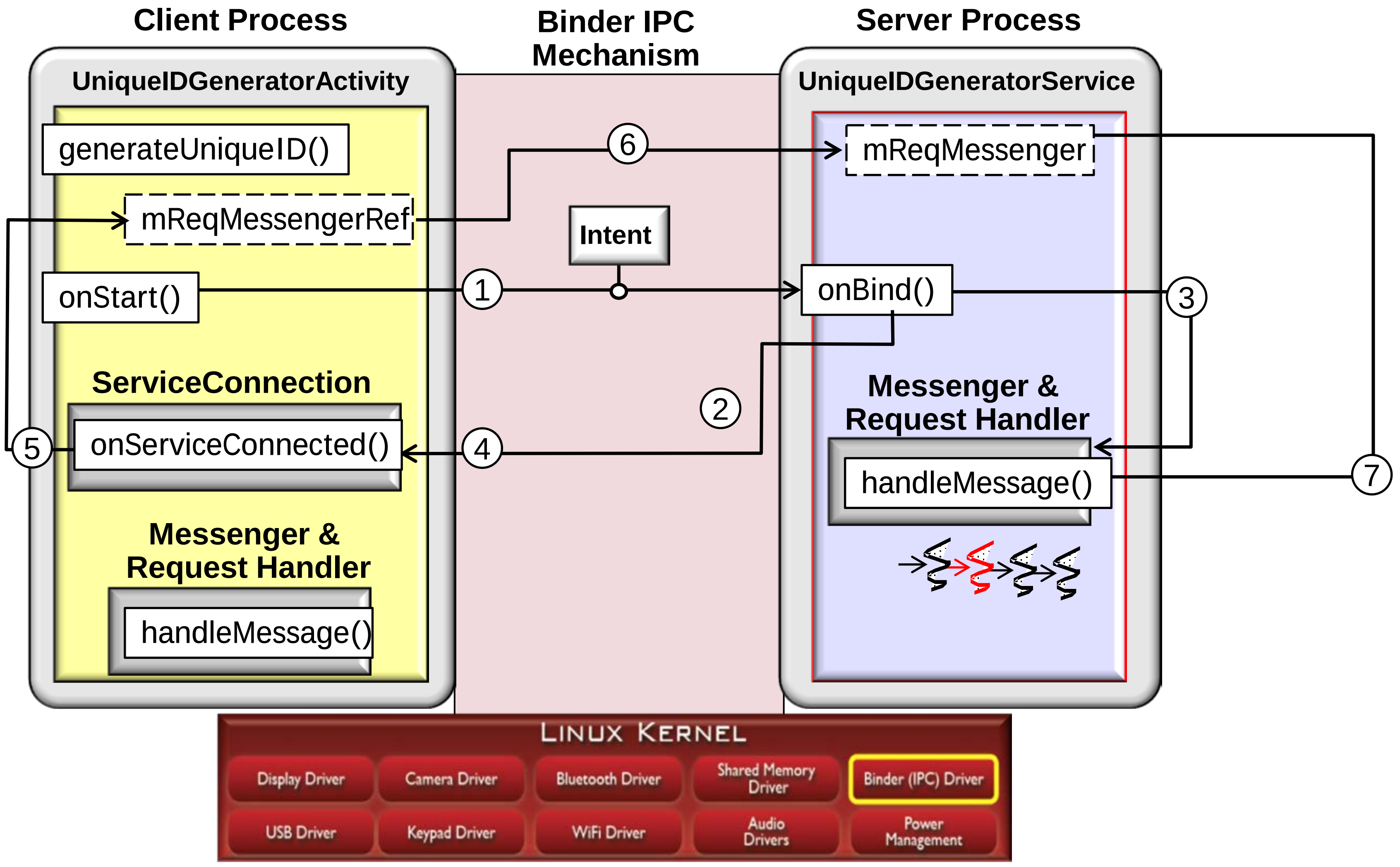
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



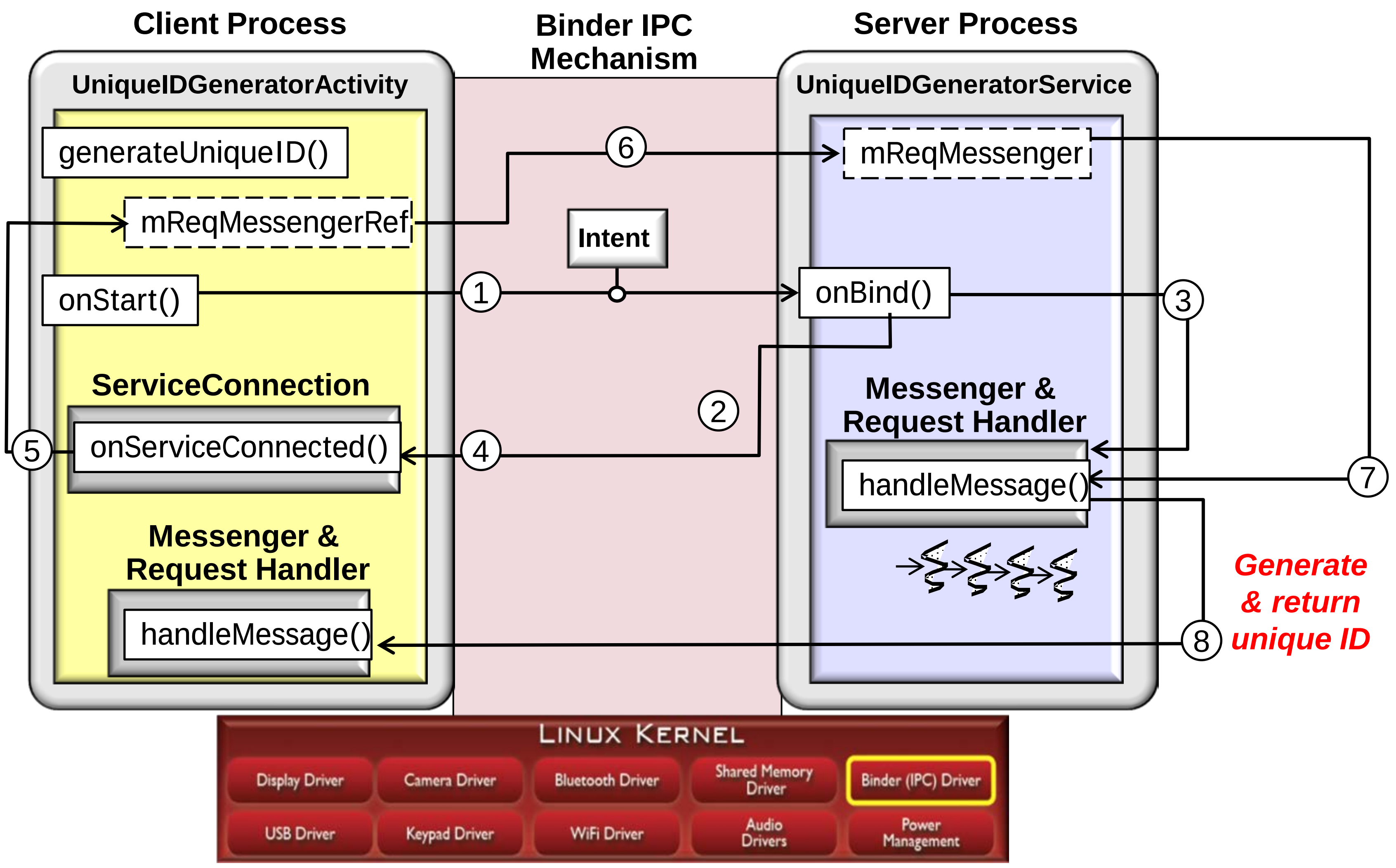
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



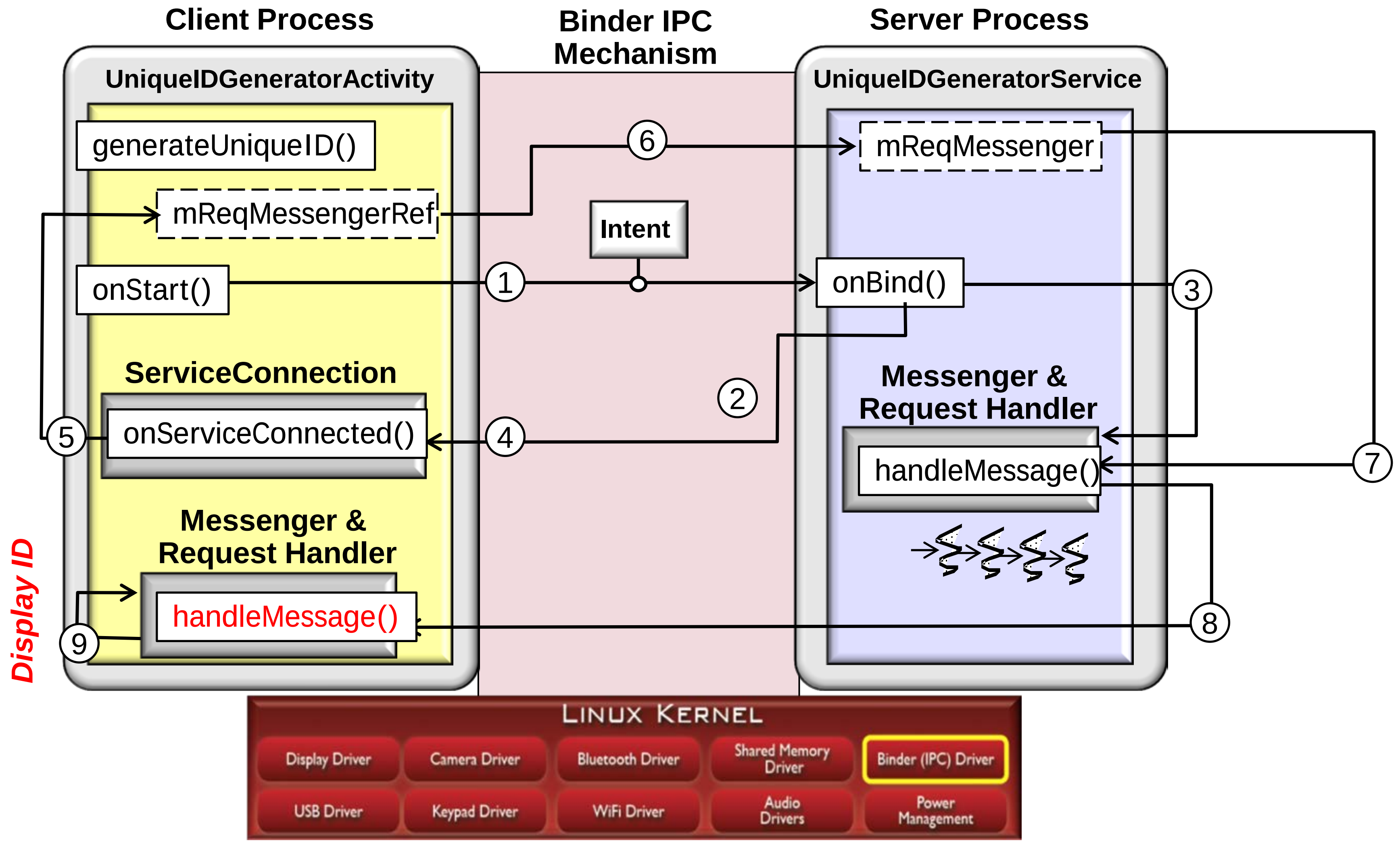
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



Protocol for Bound Service Interactions

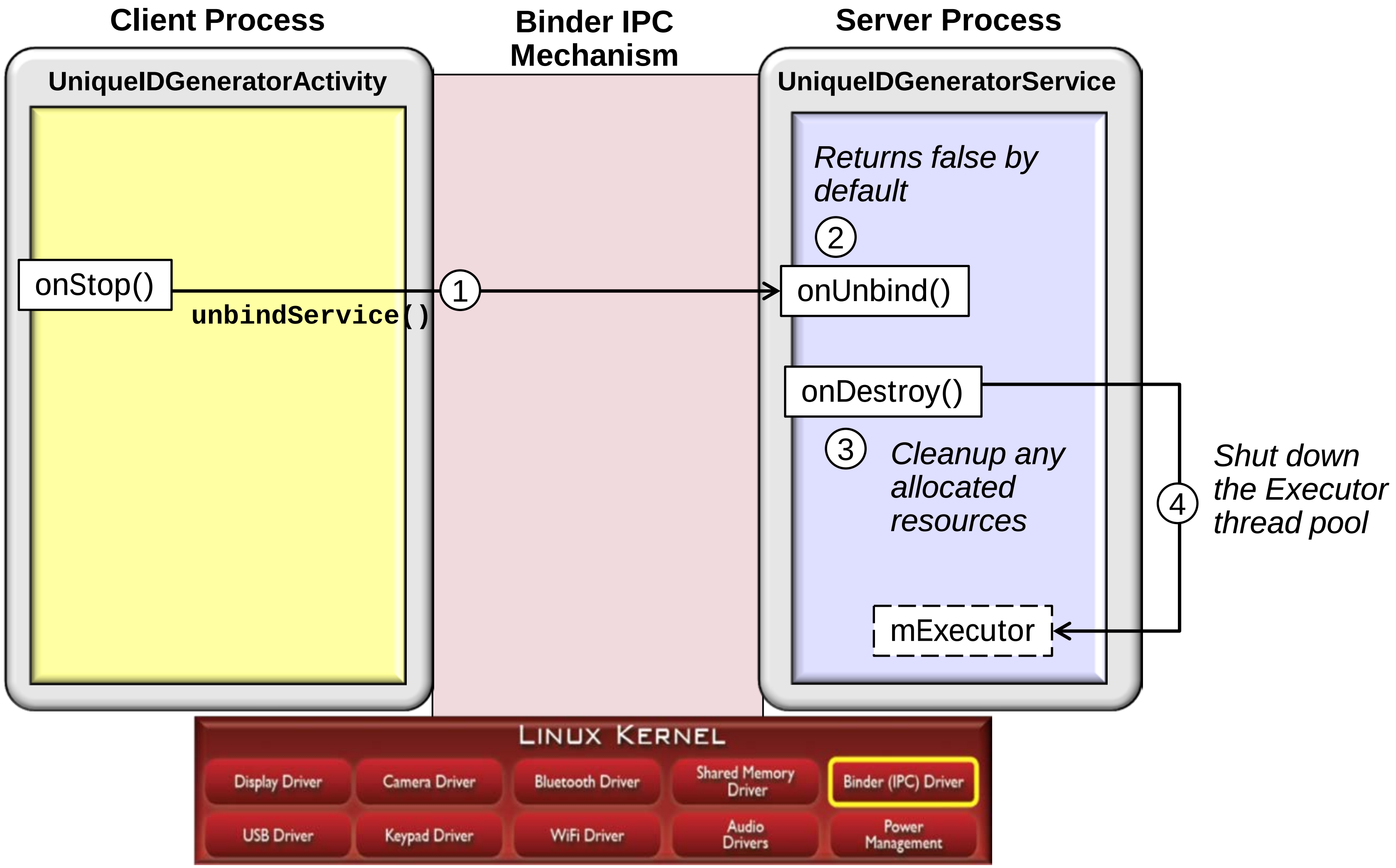
- A protocol is used to interact between Activities & Bound Services



The Protocol for Shutting Down Bound Services

Protocol for Bound Service Interactions

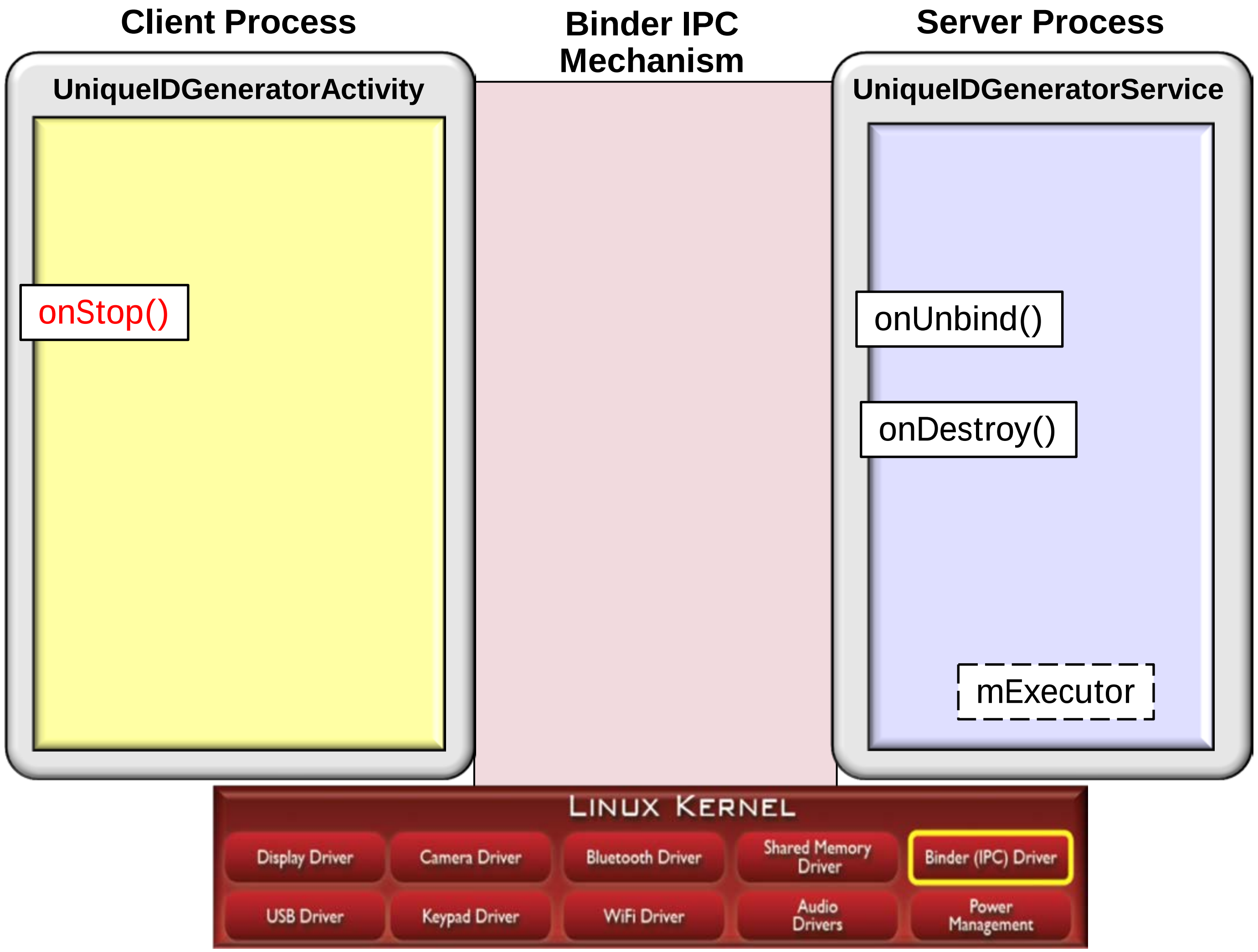
- A protocol is used to interact between Activities & Bound Services



We next examine the protocol for unbinding from & shutting down a Bound Service

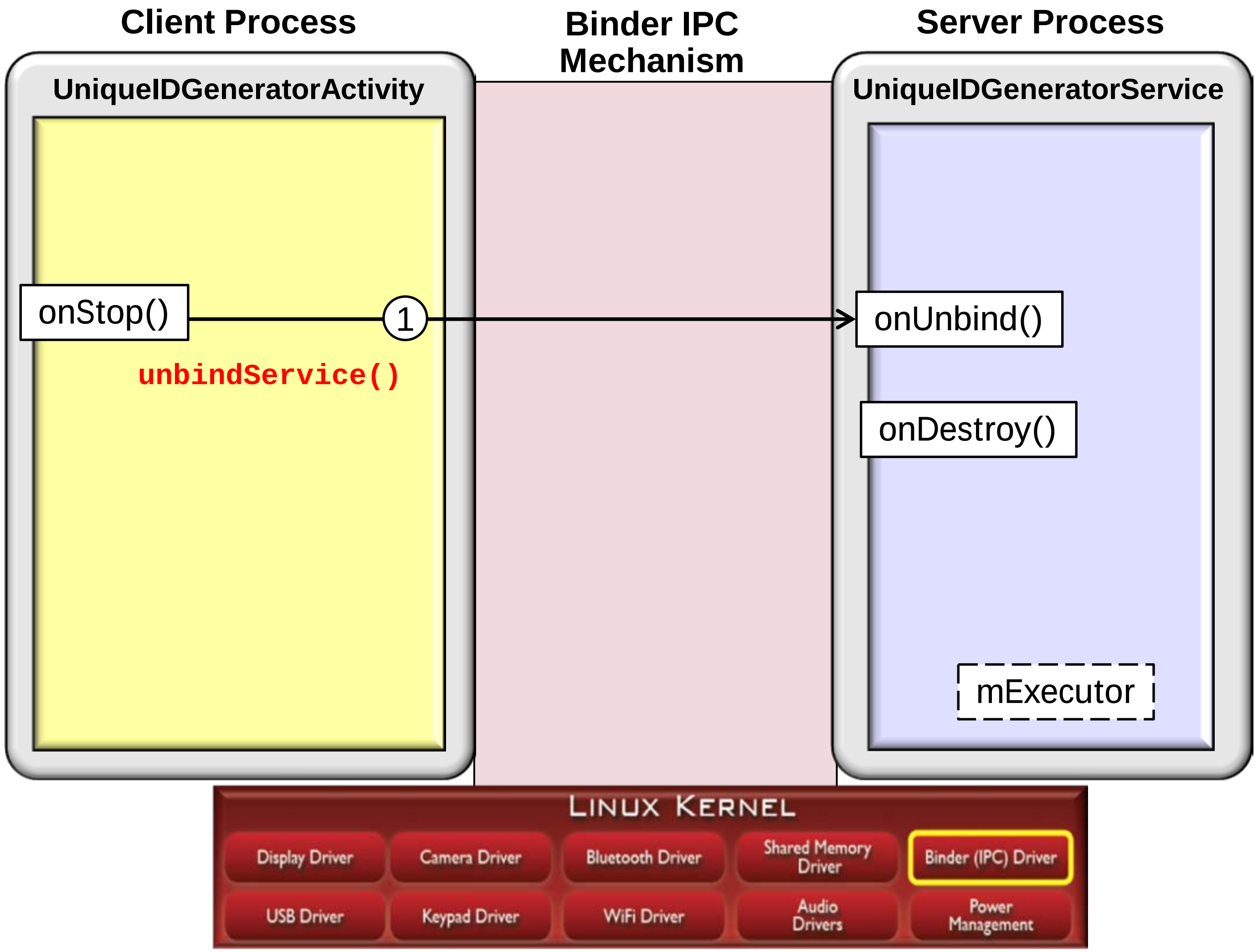
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



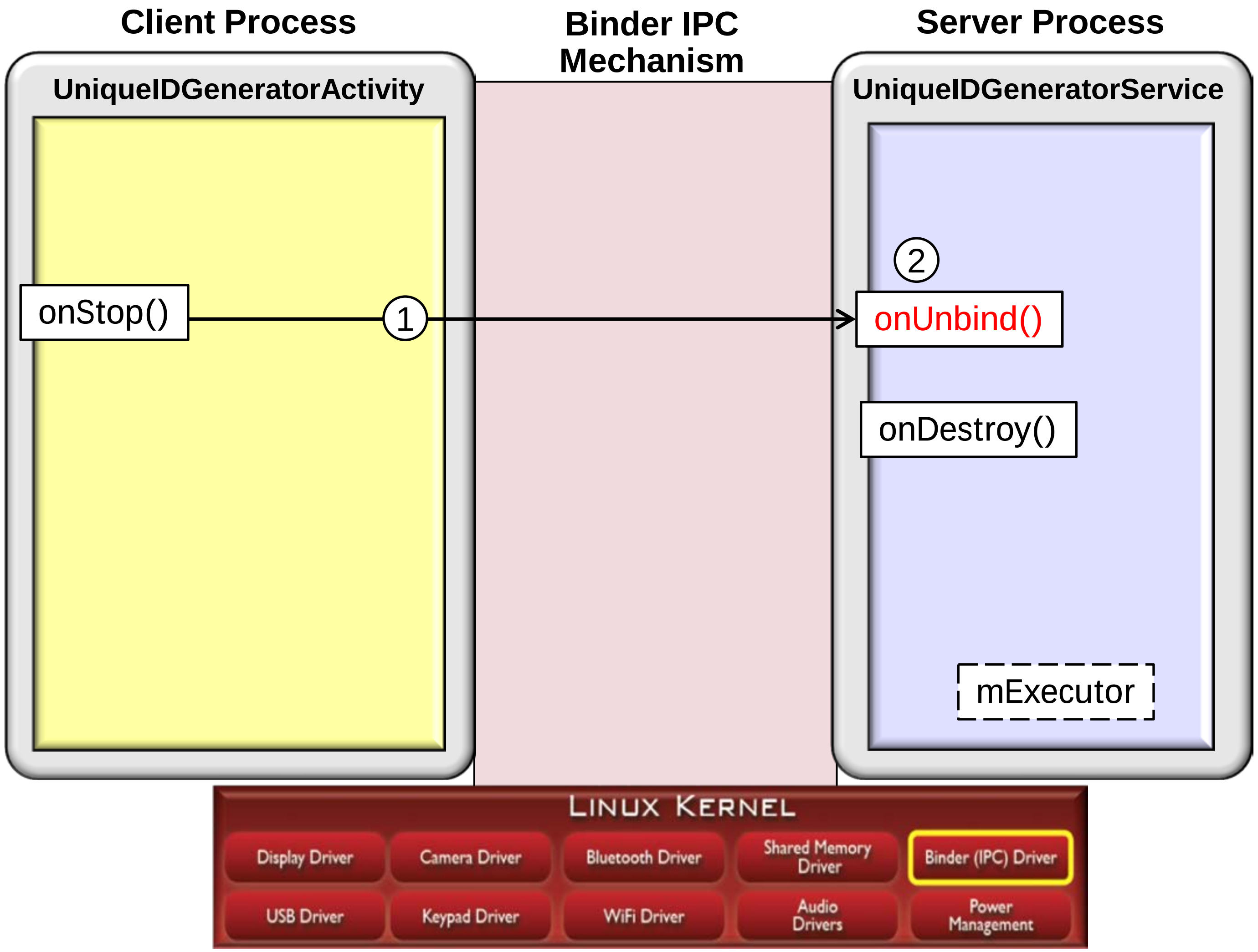
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



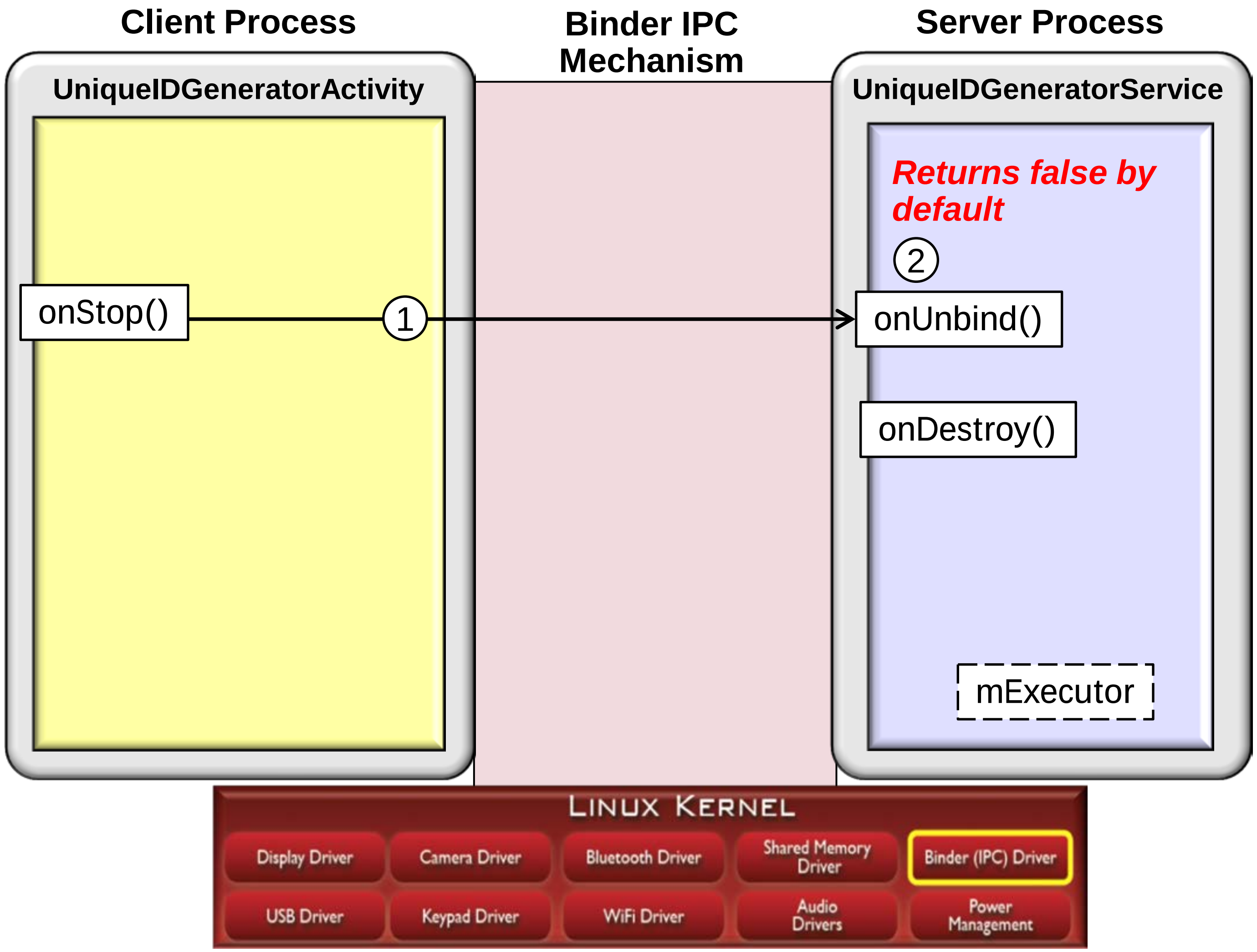
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



Protocol for Bound Service Interactions

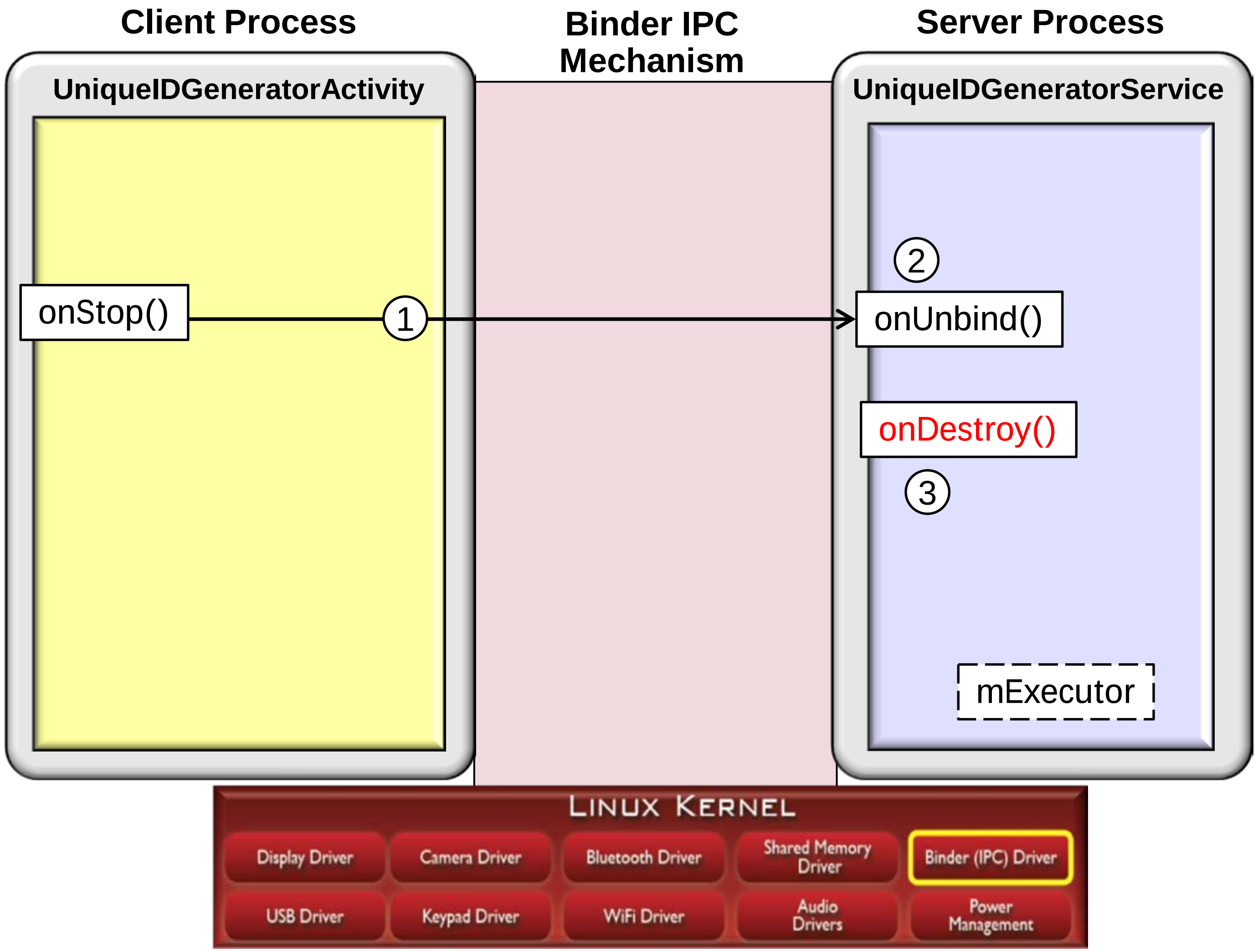
- A protocol is used to interact between Activities & Bound Services



developer.android.com/reference/android/app/Service.html#onUnbind(android.content.Intent)

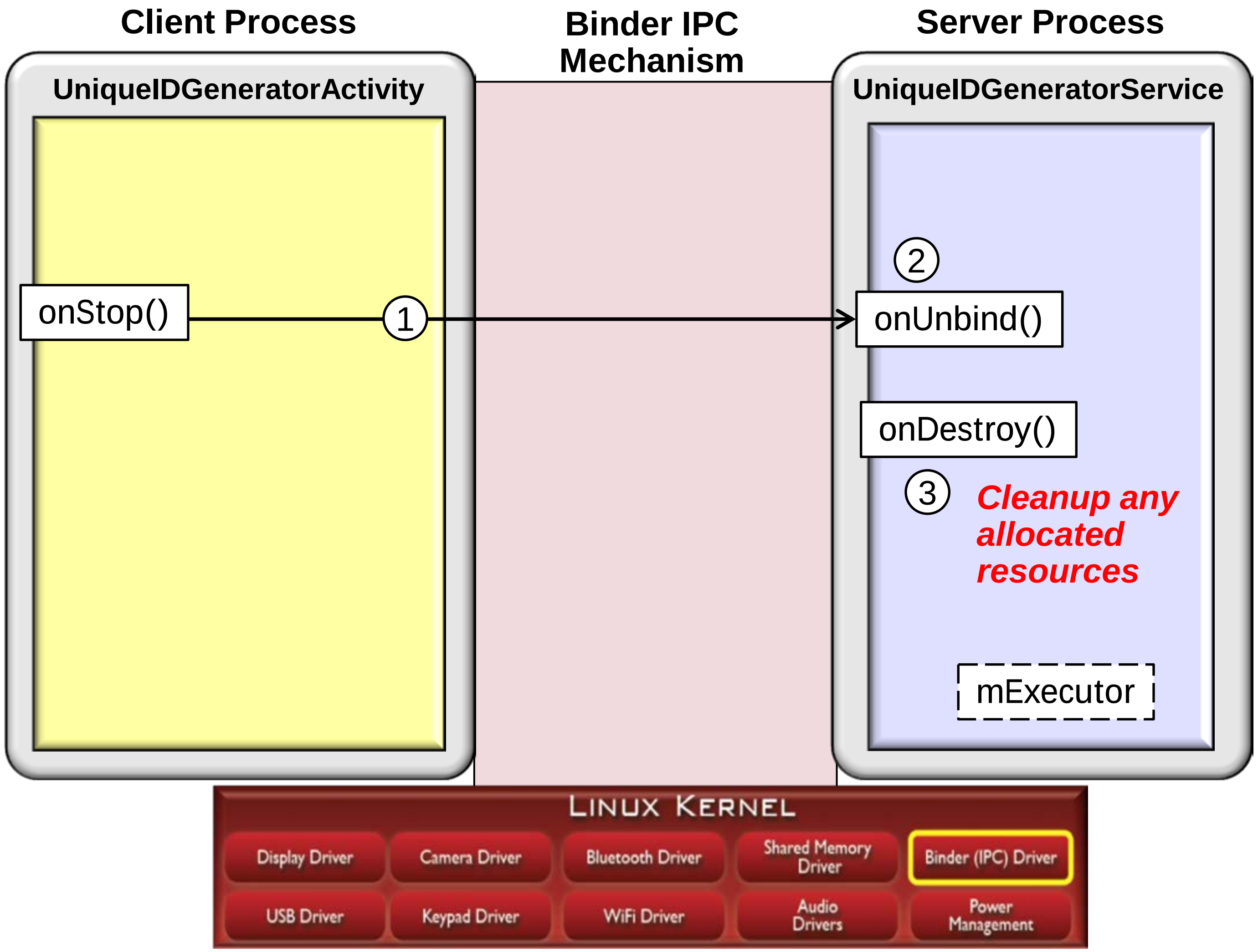
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



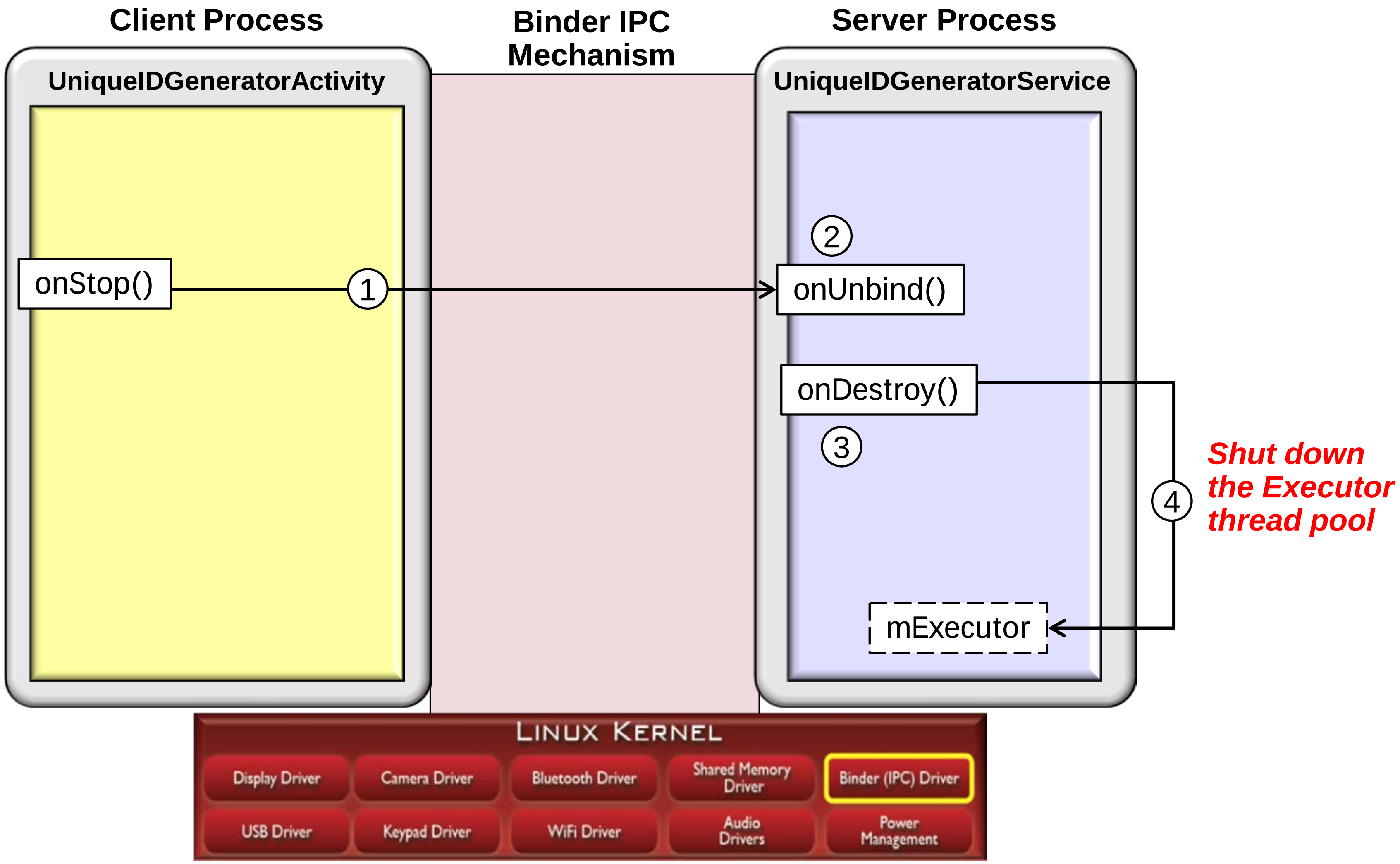
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



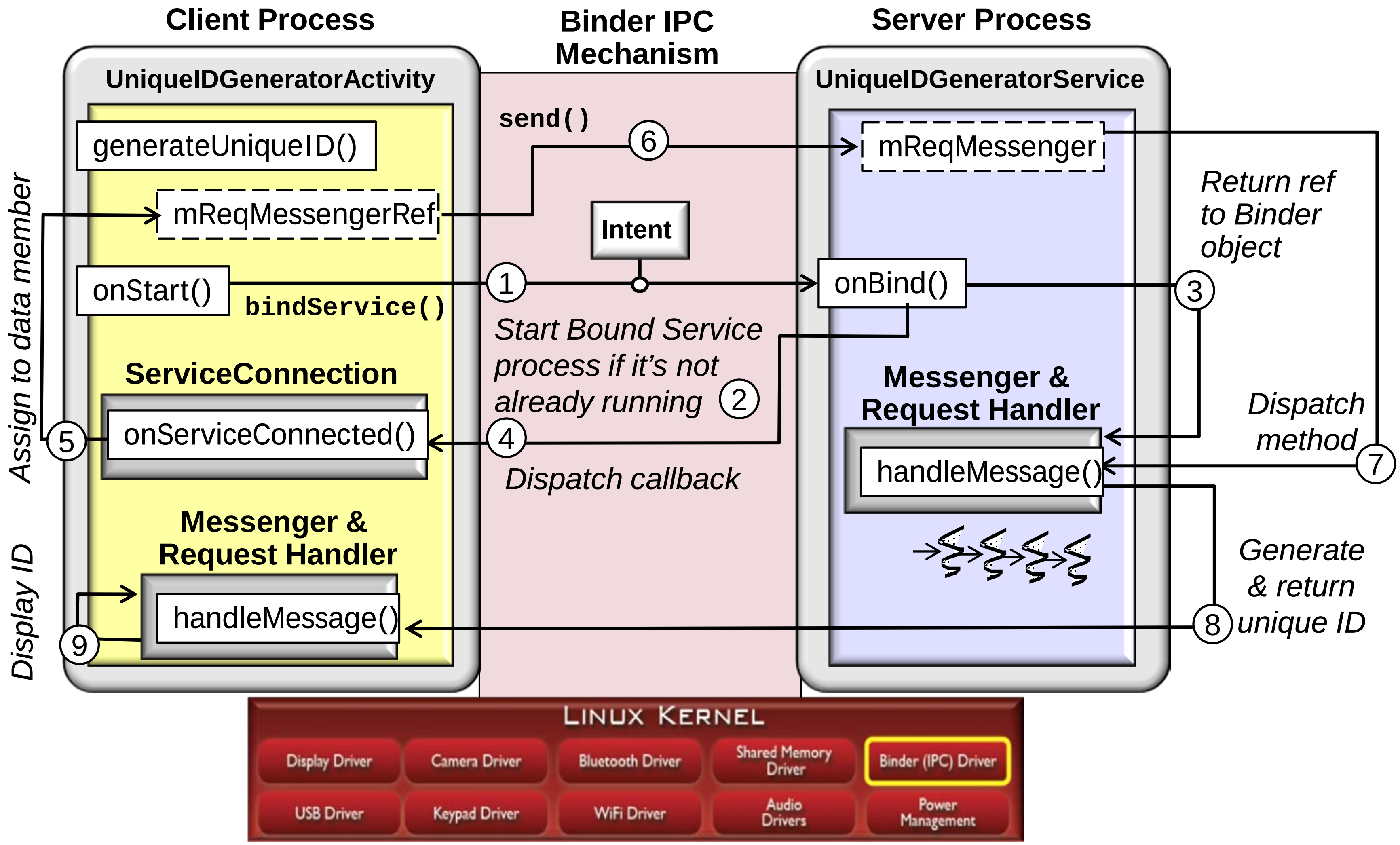
Protocol for Bound Service Interactions

- A protocol is used to interact between Activities & Bound Services



Protocol for Bound Service Interactions

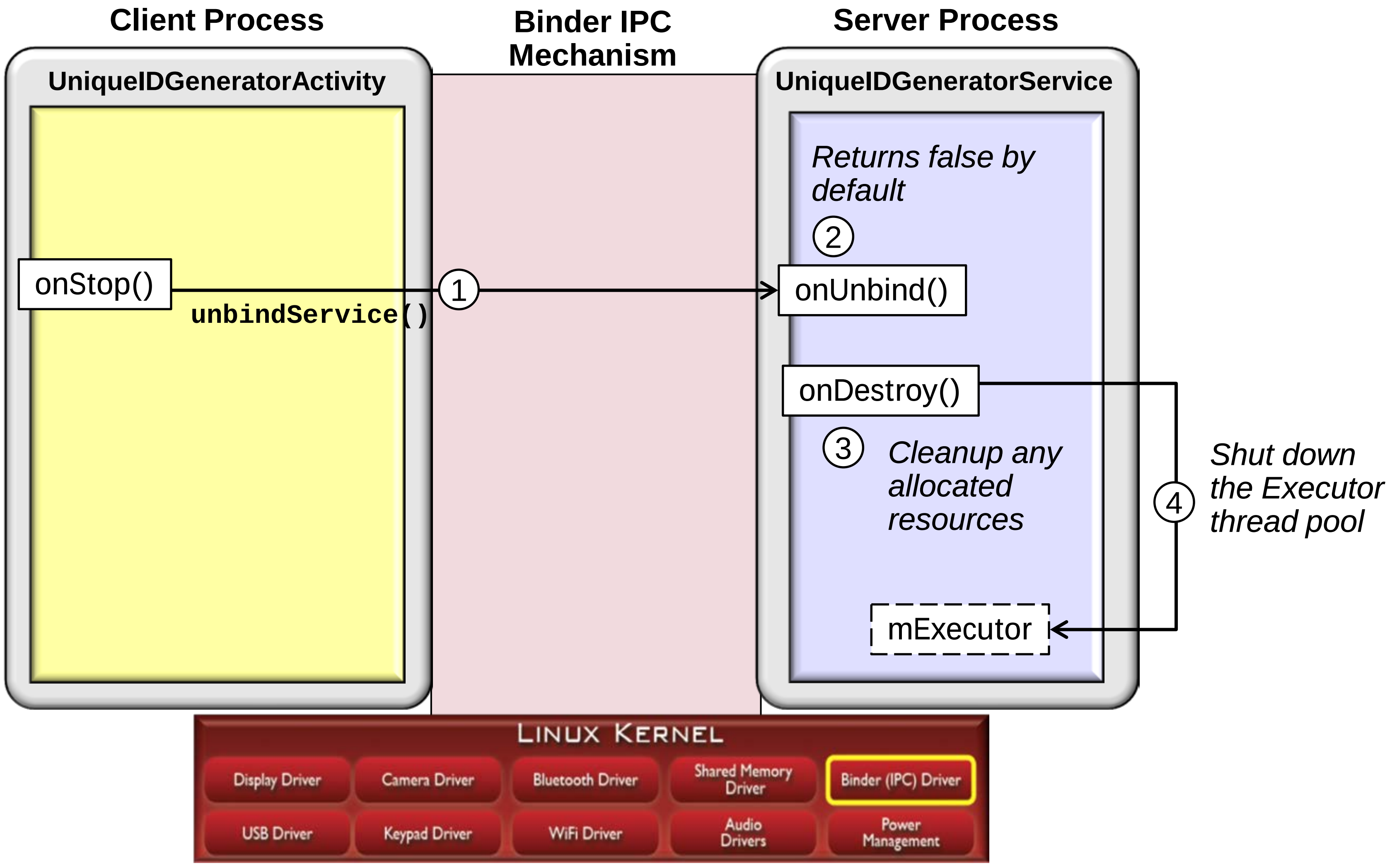
- A protocol is used to interact between Activities & Bound Services



This interaction protocol is used by essentially all Bound Services

Protocol for Bound Service Interactions

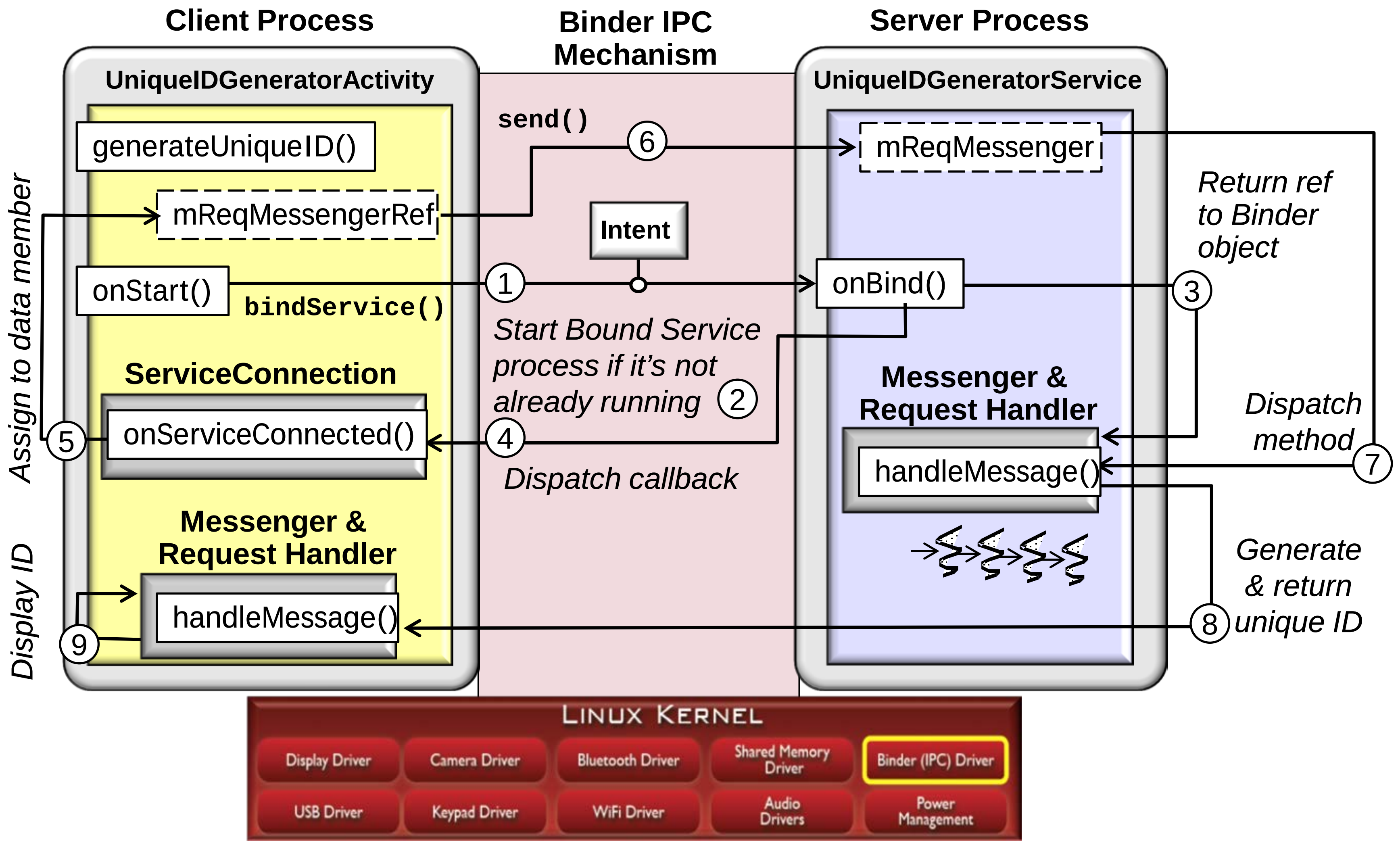
- A protocol is used to interact between Activities & Bound Services



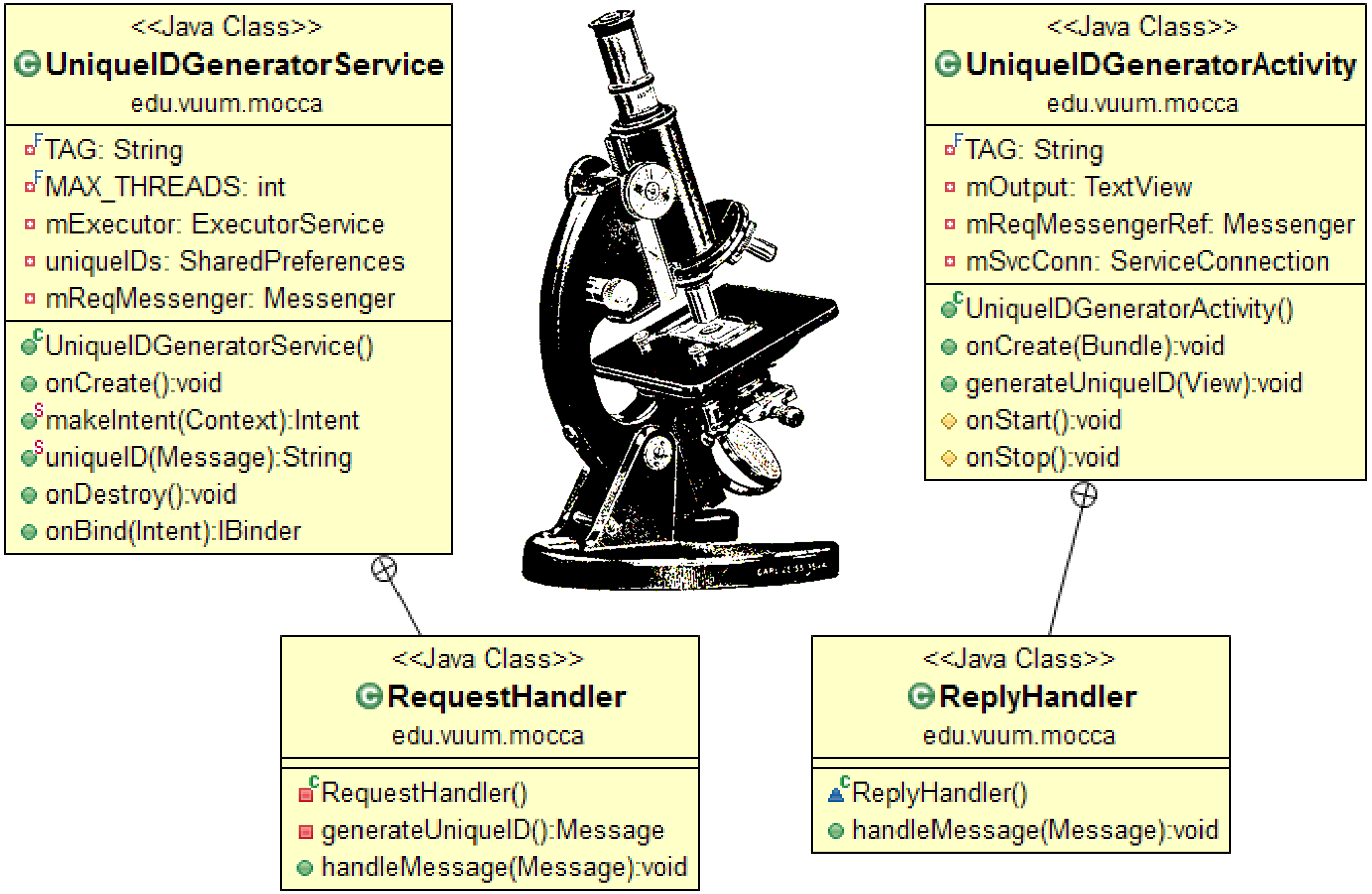
This interaction protocol is used by essentially all Bound Services

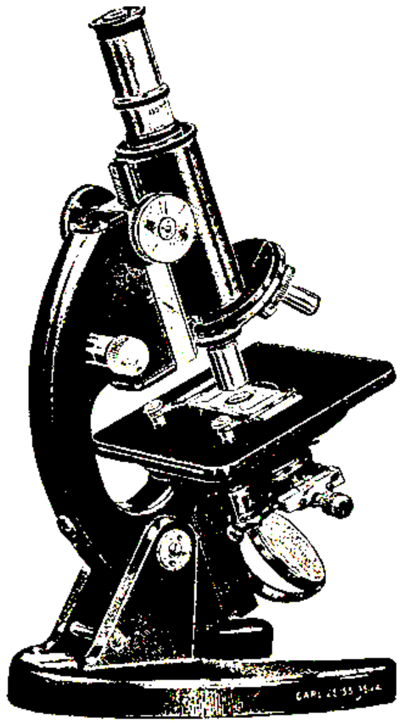
Overview of the UniqueIDGenerator Application Implementation

UniqueIDGeneratorActivity Implementation

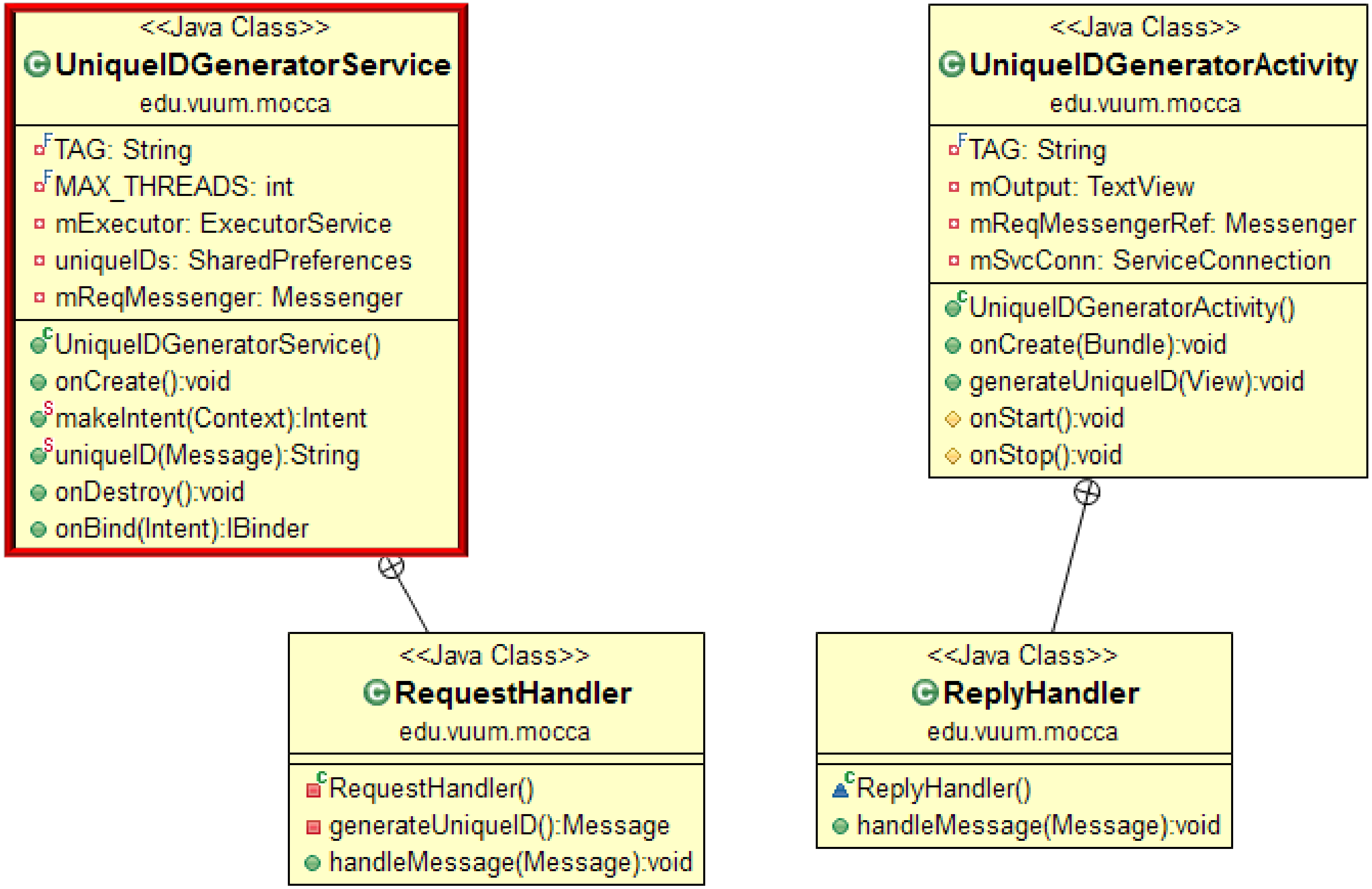


UniqueIDGeneratorActivity Implementation

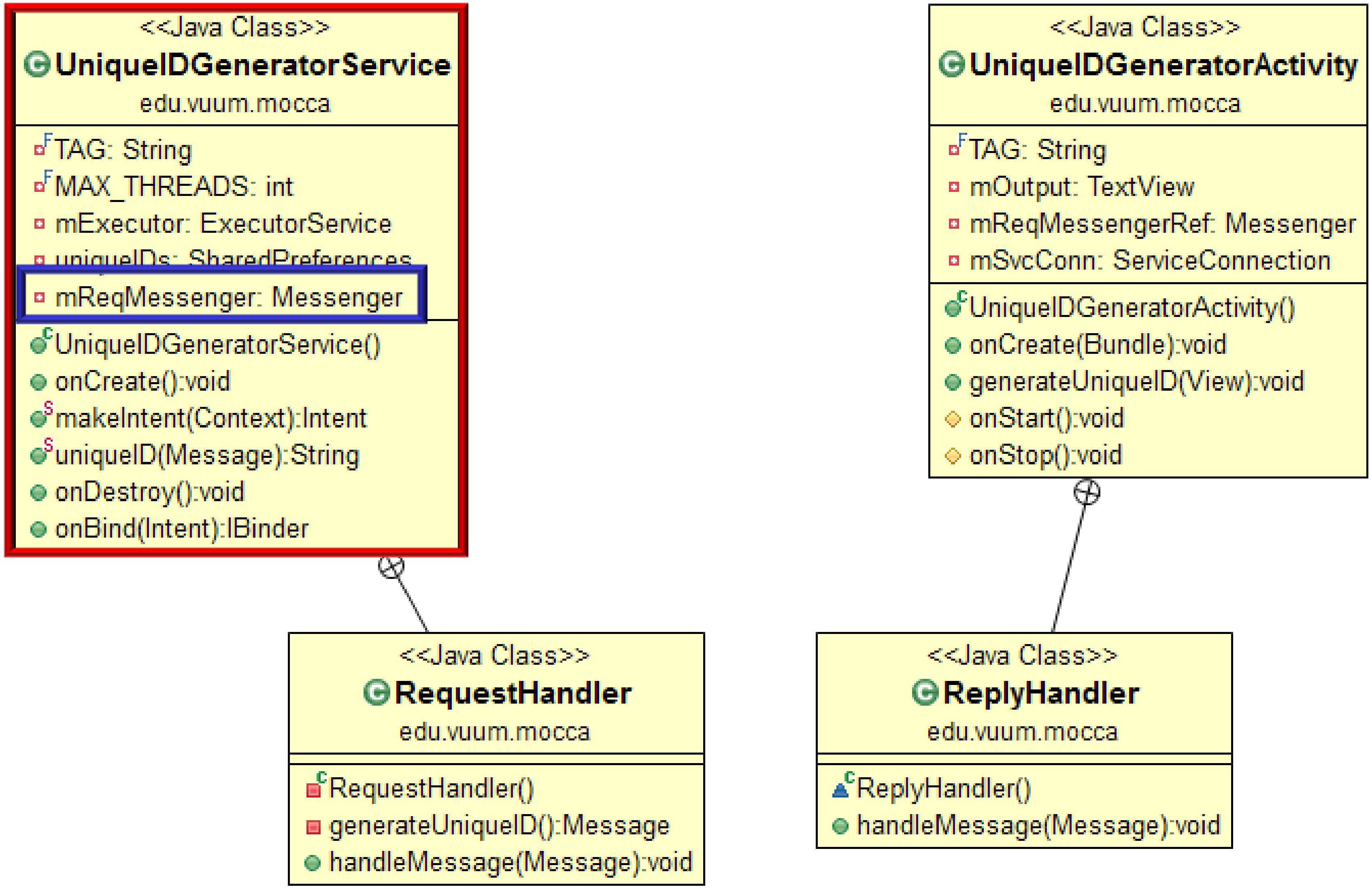




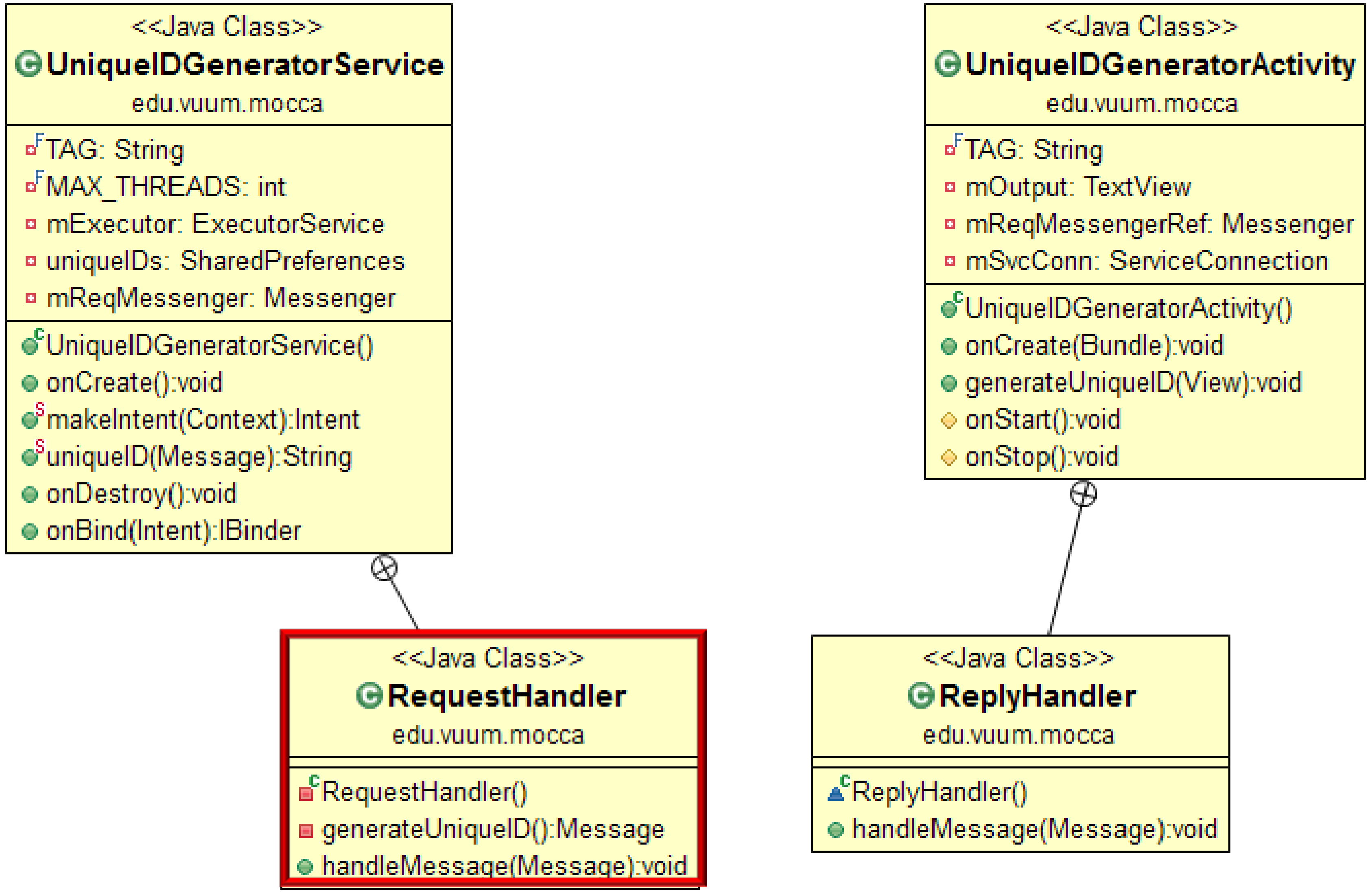
UniqueIDGeneratorActivity Implementation



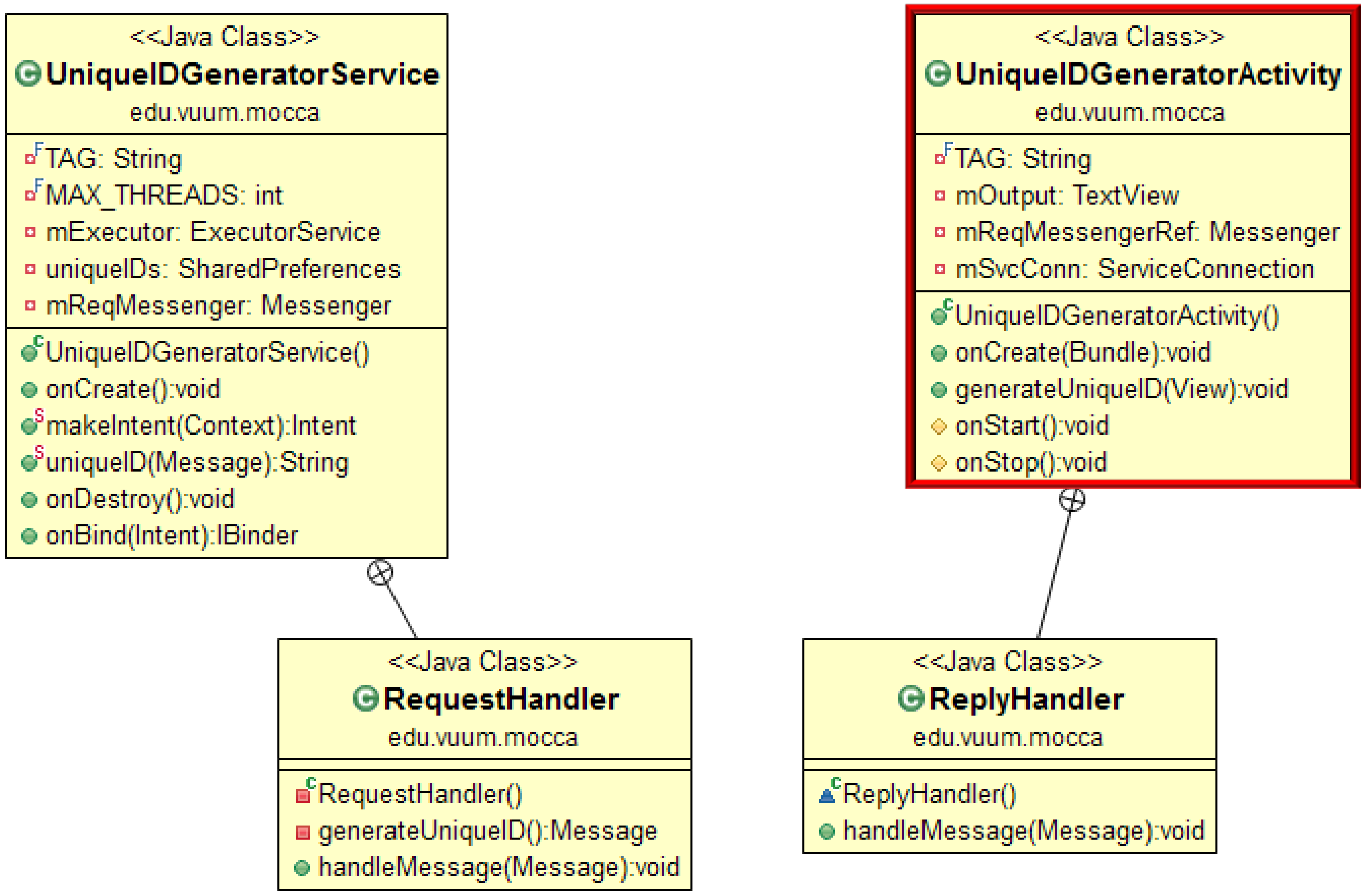
UniqueIDGeneratorActivity Implementation



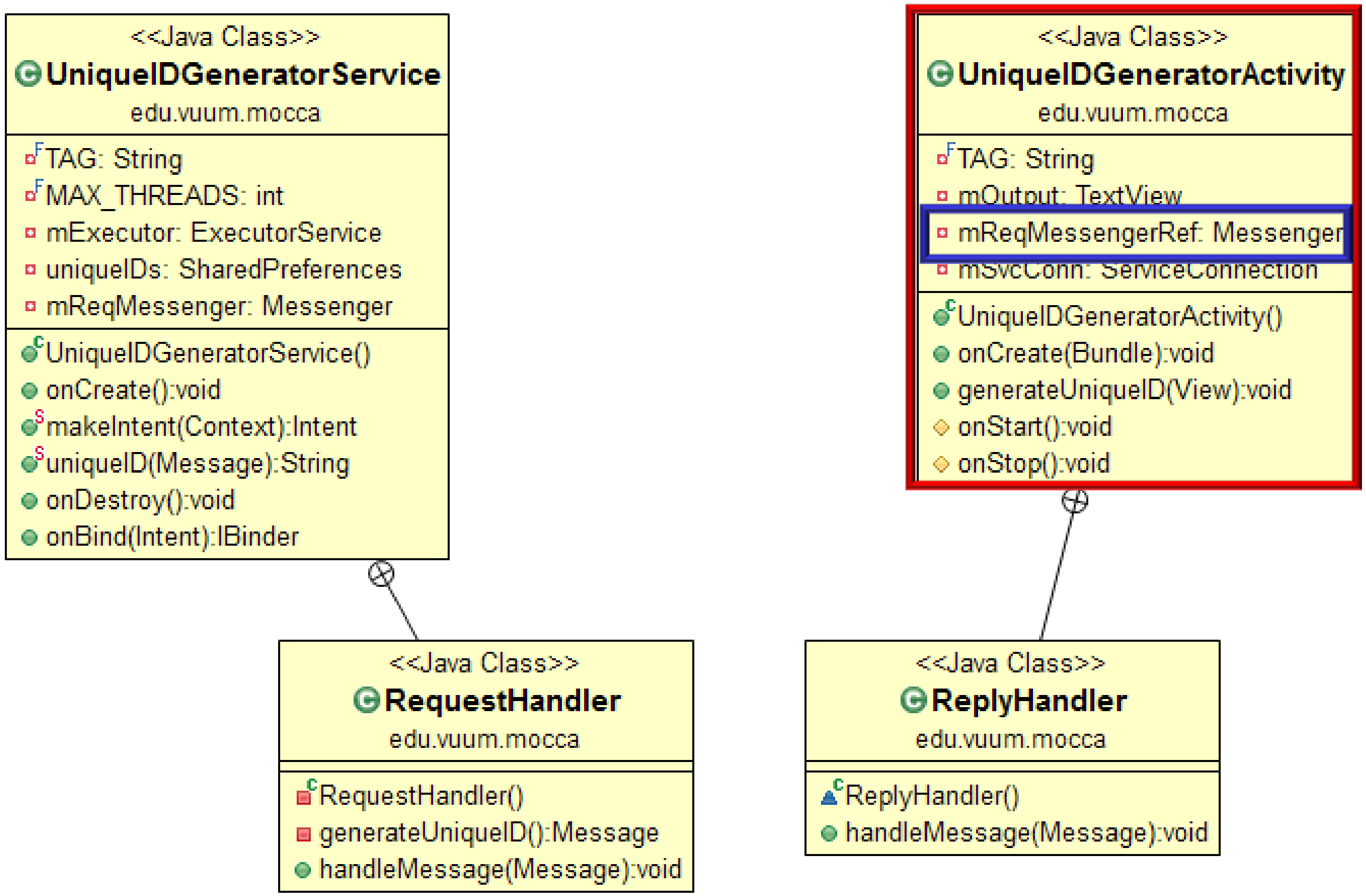
UniqueIDGeneratorActivity Implementation



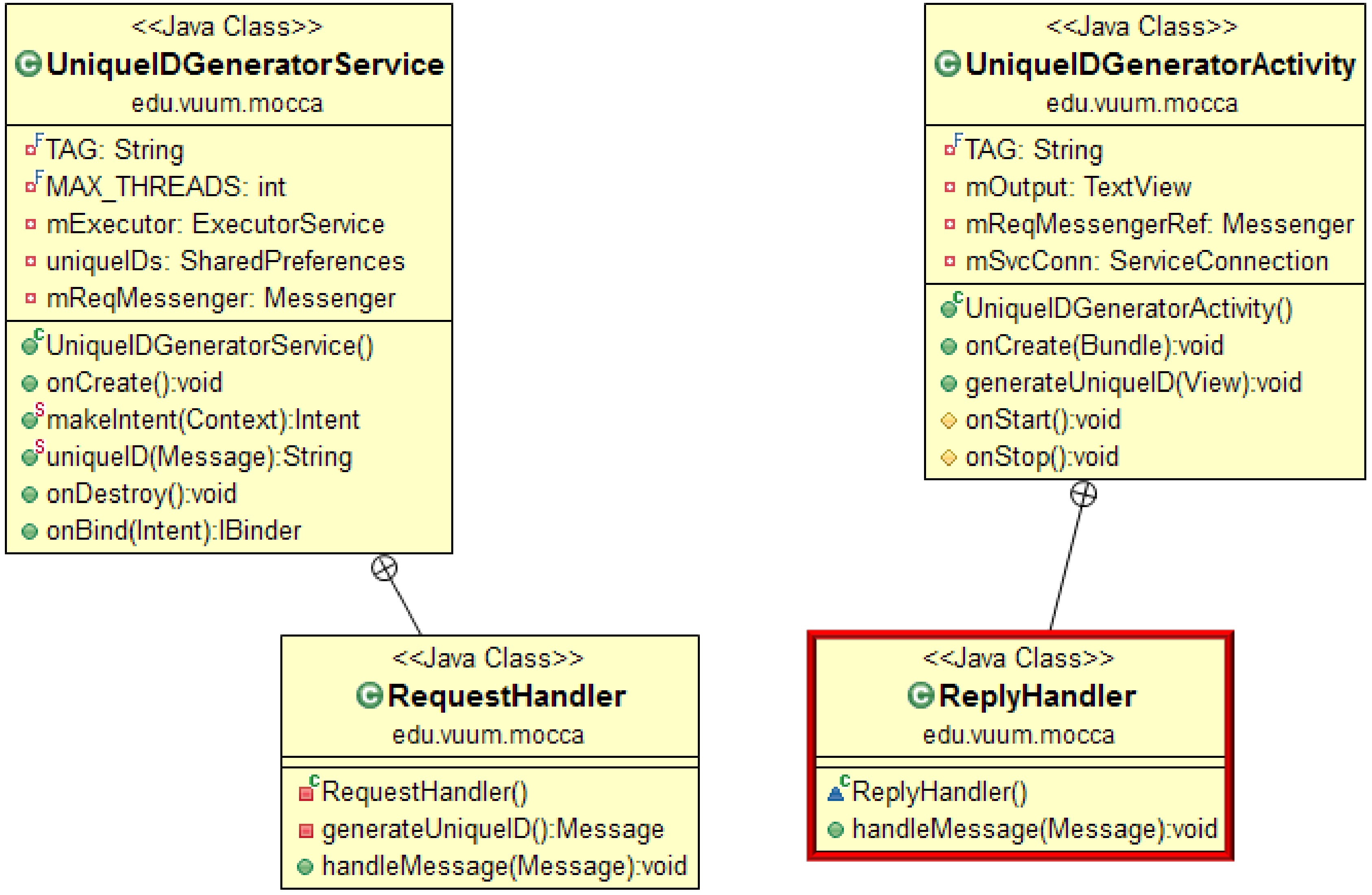
UniqueIDGeneratorActivity Implementation



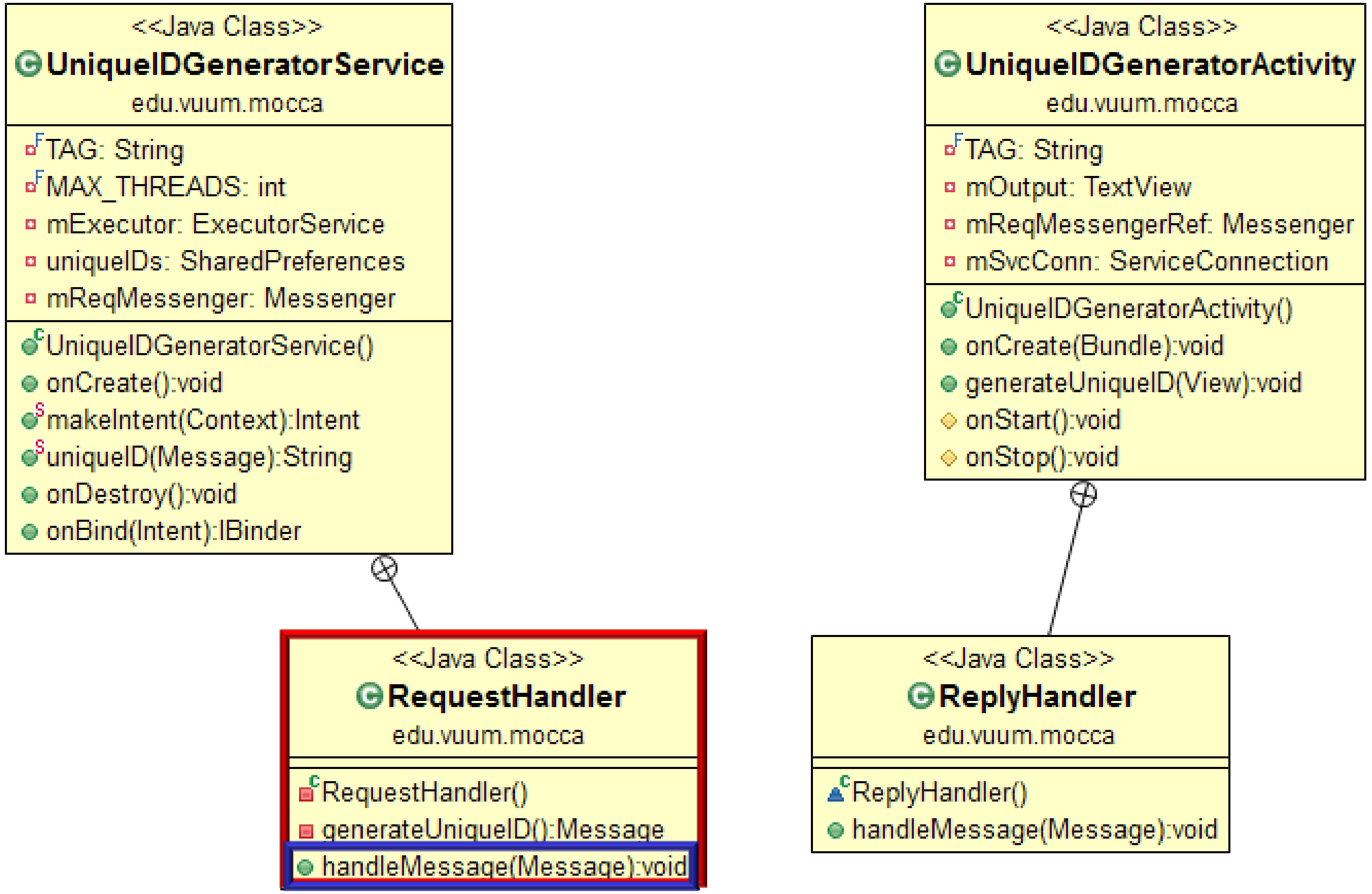
UniqueIDGeneratorActivity Implementation



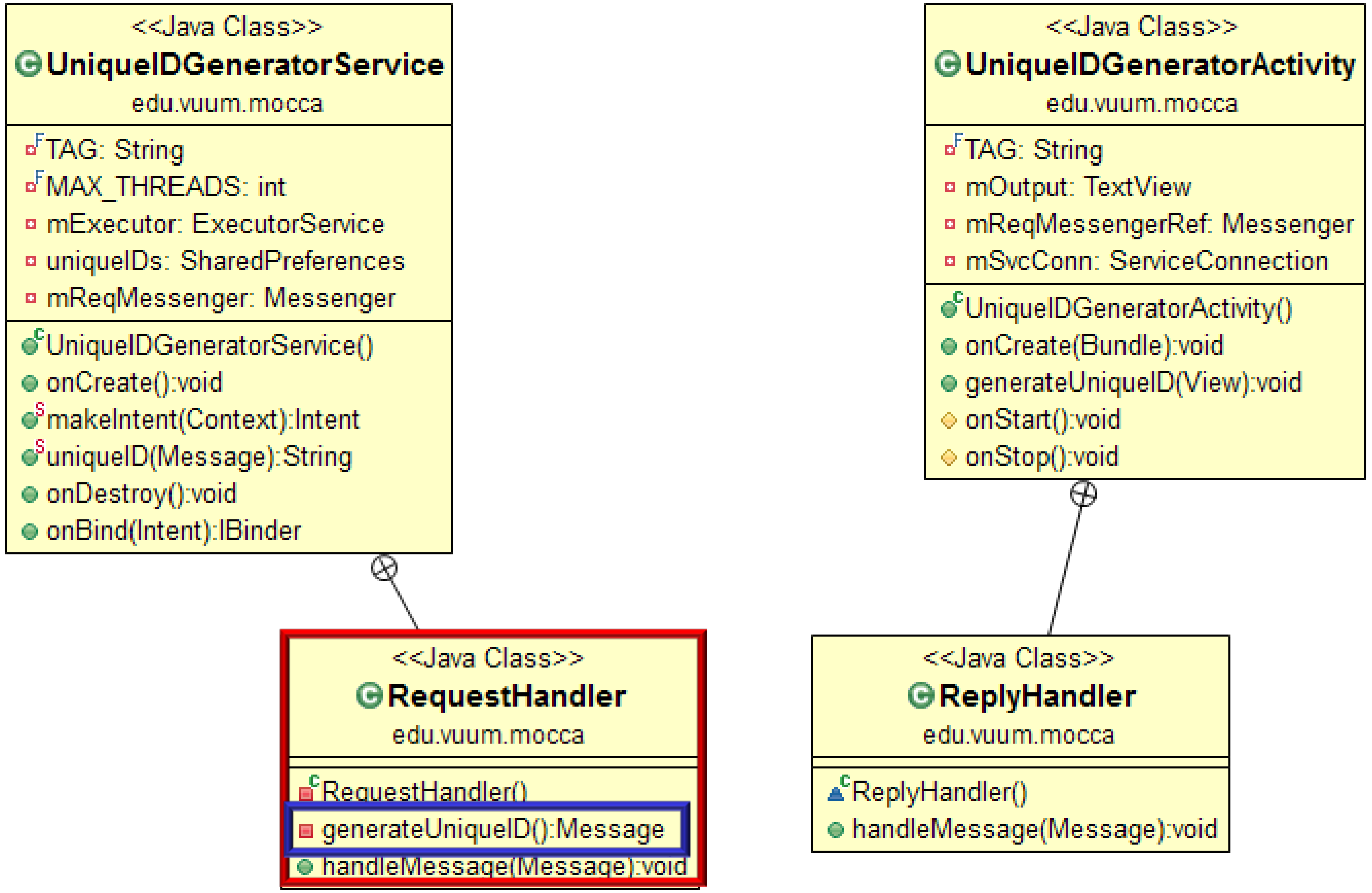
UniqueIDGeneratorActivity Implementation



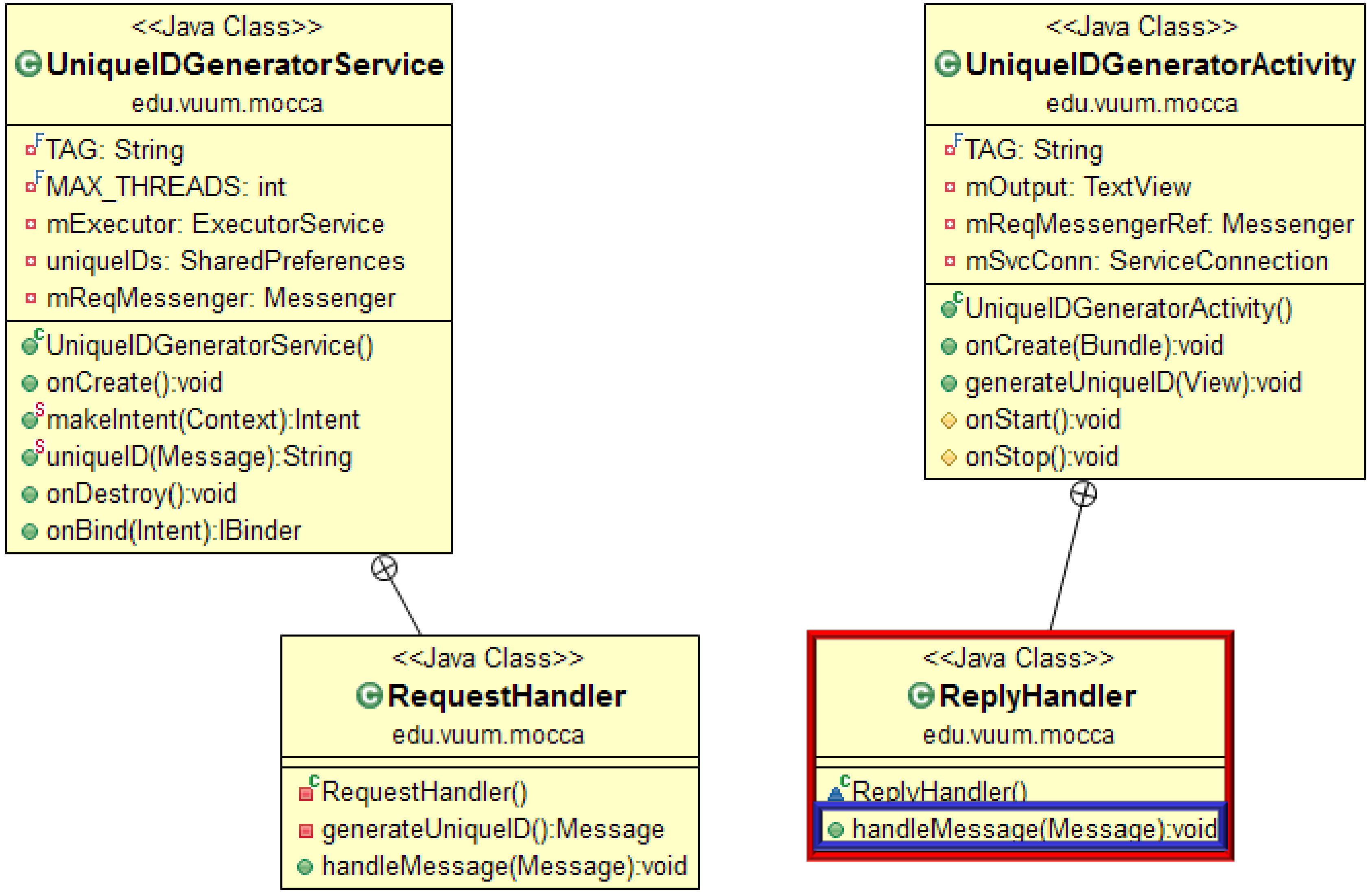
UniqueIDGeneratorActivity Implementation



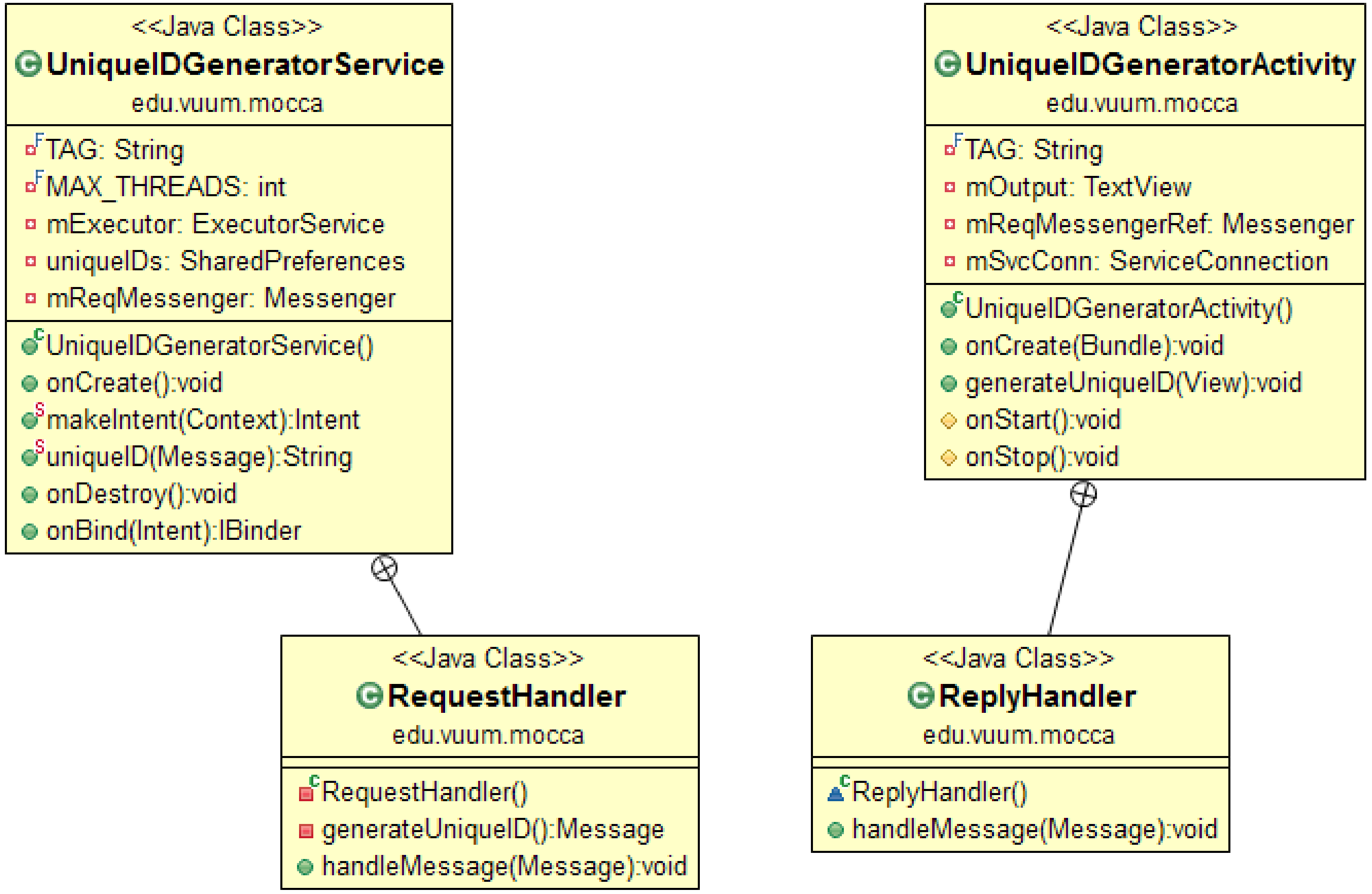
UniqueIDGeneratorActivity Implementation



UniqueIDGeneratorActivity Implementation

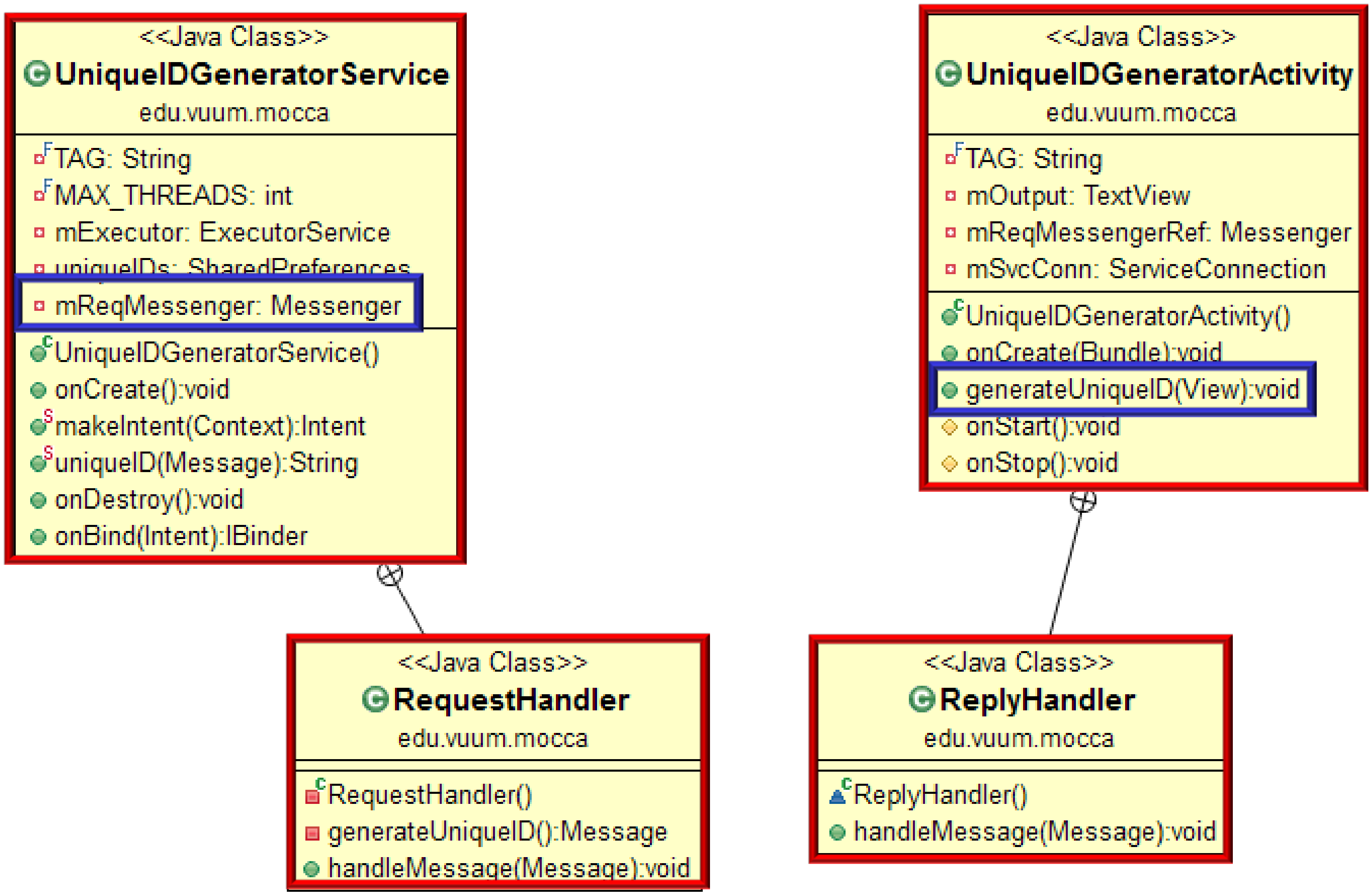


UniqueIDGeneratorActivity Implementation



Note the symmetry in the design of the UniqueIDGenerator Application

UniqueIDGeneratorActivity Implementation



Note the symmetry in the design of the UniqueIDGenerator Application

UniqueID GeneratorActivity Implementation (Part 1)

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...

  private TextView mOutput;


  private Messenger mReqMessengerRef = null;
  ...
```

Interact with user to call a Bound Service that generates a system-wide Unique ID

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...

  private TextView mOutput;


  private Messenger mReqMessengerRef = null;
  ...
```

Interact with user to call a Bound Service that generates a system-wide Unique ID

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...

  private TextView mOutput;


  private Messenger mReqMessengerRef = null;
  ...
```

Location where the unique ID is displayed

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...


  private TextView mOutput;


  private Messenger mReqMessengerRef = null;
  ...
```

Reference to the Messenger implemented in the UniqueIDGeneratorService

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...
```

Used to receive a Messenger reference after binding to the UniqueIDGeneratorService using bindService()

```
  private ServiceConnection mSvcConn = new ServiceConnection() {


    public void onServiceConnected(ComponentName className,
                                   Ibinder binder) {


      mReqMessengerRef = new Message(binder);
    }


    public void onServiceDisconnected(ComponentName className) {
      mReqMessengerRef = null;
    }
  };
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...

  private ServiceConnection mSvcConn = new ServiceConnection() {
                                    Called after UniqueIDGeneratorService is connected
    public void onServiceConnected(ComponentName className,
                                   Ibinder binder) {

      mReqMessengerRef = new Message(binder);
    }

    public void onServiceDisconnected(ComponentName className) {
      mReqMessengerRef = null;
    }
  };
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...

  private ServiceConnection mSvcConn = new ServiceConnection() {

    public void onServiceConnected(ComponentName className,
                                   Ibinder binder) {
```

Create a new Messenger that encapsulates the returned IBinder

```
      mReqMessengerRef = new Message(binder);
    }

    public void onServiceDisconnected(ComponentName className) {
      mReqMessengerRef = null;
    }
  };
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...


  private ServiceConnection mSvcConn = new ServiceConnection() {


    public void onServiceConnected(ComponentName className,
                                   Ibinder binder) {


      mReqMessengerRef = new Message(binder);
    }
```

Called if the Service crashes

```
    public void onServiceDisconnected(ComponentName className) {
      mReqMessengerRef = null;
    }
  };
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...

  private ServiceConnection mSvcConn = new ServiceConnection() {

    public void onServiceConnected(ComponentName className,
                                   Ibinder binder) {

      mReqMessengerRef = new Message(binder);
    }
```

Called if the Service crashes

```
    public void onServiceDisconnected(ComponentName className) {
      mReqMessengerRef = null;
    }
  };
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...
  protected void onStart() {

    ...

    bindService(UniqueIDGeneratorService.makeIntent(this),
                mSvcConn, Context.BIND_AUTO_CREATE);
  }
```

Hook method called by Android when this Activity becomes visible

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...
  protected void onStart() {

    ...

    bindService(UniqueIDGeneratorService.makeIntent(this),
                mSvcConn, Context.BIND_AUTO_CREATE);
  }
```

Bind to UniqueIDGeneratorService associated with this Intent

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...
  protected void onStart() {

    ...

    bindService(UniqueIDGeneratorService.makeIntent(this),
                mSvcConn, Context.BIND_AUTO_CREATE);
  }
```

Bind to UniqueIDGeneratorService associated with this Intent

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...
  protected void onStart() {

    ...

    bindService(UniqueIDGeneratorService.makeIntent(this),
                mSvcConn, Context.BIND_AUTO_CREATE);
  }
```

Bind to UniqueIDGeneratorService associated with this Intent

UniqueID GeneratorActivity Implementation (Part 2)

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...
```

Called when user presses "Generate Unique ID" button

```
  public void generateUniqueID(View view) {



    Message request = Message.obtain();
    request.replyTo = new Messenger(new ReplyHandler());
    ...
    mReqMessengerRef.send(request);
    ...
  }
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...



  public void generateUniqueID(View view) {
```

Create a request Message that indicates Service should send reply back to ReplyHandler encapsulated by Messenger

```
    Message request = Message.obtain();
    request.replyTo = new Messenger(new ReplyHandler());
    ...
    mReqMessengerRef.send(request);
    ...
  }
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...



  public void generateUniqueID(View view) {
```

Create a request Message that indicates Service should send reply back to ReplyHandler encapsulated by Messenger

```
    Message request = Message.obtain();
    request.replyTo = new Messenger(new ReplyHandler());
    ...
    mReqMessengerRef.send(request);
    ...
  }
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...



  public void generateUniqueID(View view) {
```

Create a request Message that indicates Service should send reply back to ReplyHandler encapsulated by Messenger

```
    Message request = Message.obtain();
    request.replyTo = new Messenger(new ReplyHandler());
    ...
    mReqMessengerRef.send(request);
    ...
  }
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...


  public void generateUniqueID(View view) {


    Message request = Message.obtain();
    request.replyTo = new Messenger(new ReplyHandler());
    ...
    mReqMessengerRef.send(request);
    ...
  }
```

Send the request Message to the UniqueIDGeneratorService

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...



  public void generateUniqueID(View view) {



    Message request = Message.obtain();
    request.replyTo = new Messenger(new ReplyHandler());
    ...
    mReqMessengerRef.send(request);
    ...
  }
```

Send the request Message to the UniqueIDGeneratorService

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...
```

Receives the reply containing the unique ID sent by the UniqueIDGeneratorService

```
  class ReplyHandler extends Handler {



    public void handleMessage(Message reply) {



      String uniqueID = UniqueIDGeneratorService.uniqueID(reply);

      mOutput.setText(uniqueID);
    }
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...
```

Receives the reply containing the unique ID sent by the UniqueIDGeneratorService

```
  class ReplyHandler extends Handler {



    public void handleMessage(Message reply) {



      String uniqueID = UniqueIDGeneratorService.uniqueID(reply);

      mOutput.setText(uniqueID);
    }
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...



  class ReplyHandler extends Handler {


    public void handleMessage(Message reply) {



      String uniqueID = UniqueIDGeneratorService.uniqueID(reply);

      mOutput.setText(uniqueID);
    }
```

Callback to handle the reply from the UniqueIDGeneratorService

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...



  class ReplyHandler extends Handler {



    public void handleMessage(Message reply) {
```

Get the unique ID encapsulated in the reply Message

```
      String uniqueID = UniqueIDGeneratorService.uniqueID(reply);

      mOutput.setText(uniqueID);
    }
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...

  class ReplyHandler extends Handler {

    public void handleMessage(Message reply) {

      String uniqueID = UniqueIDGeneratorService.uniqueID(reply);

      mOutput.setText(uniqueID);
    }
```

Display the unique ID

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...


  protected void onStop() {
    unbindService(mSvcConn);




    super.onStop();
  }
```

Hook method called by Android when this Activity becomes invisible

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...


  protected void onStop() {
    unbindService(mSvcConn);
```

Unbind from the UniqueIDGeneratorService

```
    super.onStop();
  }
```

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...
  protected void onStart() {

    ...

    bindService(UniqueIDGeneratorService.makeIntent(this),
                mSvcConn, Context.BIND_AUTO_CREATE);
  }
```

Hook method called by Android when this Activity becomes visible

UniqueIDGeneratorActivity Implementation

```
public class UniqueIDGeneratorActivity extends Activity {
  ...
  protected void onStart() {

    ...

    bindService(UniqueIDGeneratorService.makeIntent(this),
                mSvcConn, Context.BIND_AUTO_CREATE);
  }
```

Bind to UniqueIDGeneratorService associated with this Intent

UniqueID GeneratorService Implementation (Part 1)

UniqueIDGeneratorService Implementation

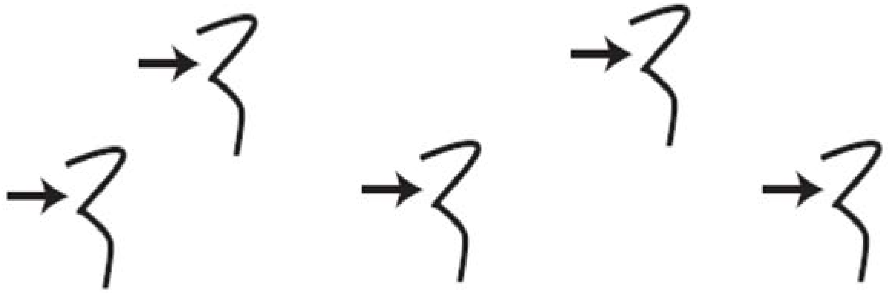
```
public class UniqueIDGeneratorService extends Service {
  ...
```

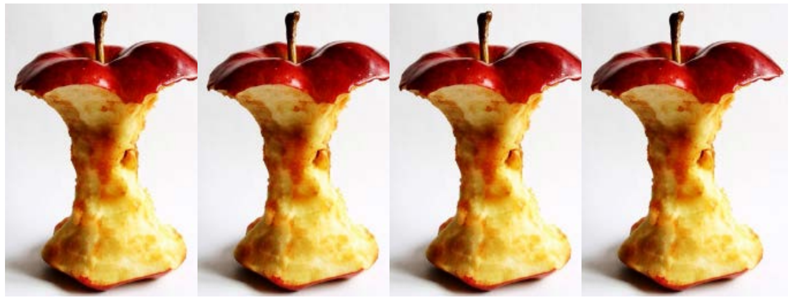
This Service generates unique IDs within a pool of Threads

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
   ...
```

This Service generates unique IDs within a pool of Threads





A Thread pool can process requests concurrently & improve performance on a multi-core device

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  private ExecutorService mExecutor;


  private final int MAX_THREADS = 4;


  private SharedPreferences uniqueIDs = null;


  private Messenger mReqMessenger = null;




  public Ibinder onBind(Intent intent) {
    return mReqMessenger.getBinder();
  }
```

UniqueIDGeneratorService Implementation

The Thread pool implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  private ExecutorService mExecutor;


  private final int MAX_THREADS = 4;


  private SharedPreferences uniqueIDs = null;


  private Messenger mReqMessenger = null;




  public Ibinder onBind(Intent intent) {
    return mReqMessenger.getBinder();
  }
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  private ExecutorService mExecutor;


  private final int MAX_THREADS = 4;


  private SharedPreferences uniqueIDs = null;


  private Messenger mReqMessenger = null;



  public Ibinder onBind(Intent intent) {
    return mReqMessenger.getBinder();
  }
```

The maximum number of Threads used to service download requests

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  private ExecutorService mExecutor;


  private final int MAX_THREADS = 4;


  private SharedPreferences uniqueIDs = null;


  private Messenger mReqMessenger = null;



  public Ibinder onBind(Intent intent) {
    return mReqMessenger.getBinder();
  }
```

Stores unique IDs persistently

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  private ExecutorService mExecutor;


  private final int MAX_THREADS = 4;


  private SharedPreferences uniqueIDs = null;


  private Messenger mReqMessenger = null;
```

A Messenger encapsulating a RequestHandler that processes request Messages sent from the UniqueIDGeneratorActivity

```
  public Ibinder onBind(Intent intent) {
    return mReqMessenger.getBinder();
  }
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  private ExecutorService mExecutor;


  private final int MAX_THREADS = 4;


  private SharedPreferences uniqueIDs = null;


  private Messenger mReqMessenger = null;
```

Factory method that returns the IBinder associated with the Request Messenger

```
  public Ibinder onBind(Intent intent) {
    return mReqMessenger.getBinder();
  }
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  private ExecutorService mExecutor;


  private final int MAX_THREADS = 4;


  private SharedPreferences uniqueIDs = null;


  private Messenger mReqMessenger = null;


  public Ibinder onBind(Intent intent) {
    return mReqMessenger.getBinder();
  }
```

Factory method that returns the IBinder associated with the Request Messenger

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  private ExecutorService mExecutor;


  private final int MAX_THREADS = 4;


  private SharedPreferences uniqueIDs = null;


  private Messenger mReqMessenger = null;
```

Factory method that returns the IBinder associated with the Request Messenger

```
  public Ibinder onBind(Intent intent) {
    return mReqMessenger.getBinder();
  }
```

UniqueIDGeneratorService Implementation

Hook method called when Service is created

```
public class UniqueIDGeneratorService extends Service {
  ...

  public void onCreate() {



    mReqMessenger =
      new Messenger(new RequestHandler());




    uniqueIDs =
      PreferenceManager.getDefaultSharedPreferences(this);




    mExecutor = Executors.newFixedThreadPool(MAX_THREADS);
  }
  ...
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  public void onCreate() {
```

Messenger encapsulating RequestHandler used to process request Messages sent from UniqueIDGeneratorActivity

```
    mReqMessenger =
      new Messenger(new RequestHandler());



    uniqueIDs =
      PreferenceManager.getDefaultSharedPreferences(this);



    mExecutor = Executors.newFixedThreadPool(MAX_THREADS);
  }
  ...
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  public void onCreate() {



    mReqMessenger =
      new Messenger(new RequestHandler());
```

Reference to the default file used by the SharedPreferences framework for this Service

```
    uniqueIDs =
      PreferenceManager.getDefaultSharedPreferences(this);



    mExecutor = Executors.newFixedThreadPool(MAX_THREADS);
  }
  ...
```

developer.android.com/reference/android/content/SharedPreferences.html

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  public void onCreate() {



    mReqMessenger =
      new Messenger(new RequestHandler());



    uniqueIDs =
      PreferenceManager.getDefaultSharedPreferences(this);
```

Create a Thread pool configured to use MAX_THREADS

```
    mExecutor = Executors.newFixedThreadPool(MAX_THREADS);
  }
  ...
```

developer.android.com/reference/java/util/concurrent/Executors.html

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...


  public static Intent makeIntent(Context context) {
    return new Intent(context,
                      UniqueIDGeneratorService.class);
  }


  public static String uniqueID(Message replyMessage) {
    return replyMessage.getData().getString("ID");
  }

  ...
```

Helper methods shield the Activity from implementation details of the Service

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  public static Intent makeIntent(Context context) {
    return new Intent(context,
                      UniqueIDGeneratorService.class);
  }


  public static String uniqueID(Message replyMessage) {
    return replyMessage.getData().getString("ID");
  }

  ...
```

Factory method that creates an Intent that's associated with the Service

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  public static Intent makeIntent(Context context) {
    return new Intent(context,
                                    UniqueIDGeneratorService.class);
  }


  public static String uniqueID(Message replyMessage) {
    return replyMessage.getData().getString("ID");
  }

  ...
```

Factory method that creates an Intent that's associated with the Service

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  public static Intent makeIntent(Context context) {
    return new Intent(context,
                      UniqueIDGeneratorService.class);
  }


  public static String uniqueID(Message replyMessage) {
    return replyMessage.getData().getString("ID");
  }

  ...
```

Factory method that creates an Intent that's associated with the Service

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...


  public static Intent makeIntent(Context context) {
    return new Intent(context,
                      UniqueIDGeneratorService.class);
  }


  public static String uniqueID(Message replyMessage) {
    return replyMessage.getData().getString("ID");
  }

  ...
```

Helper method that extracts the encapsulated unique ID from Message

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...


  public static Intent makeIntent(Context context) {
    return new Intent(context,
                      UniqueIDGeneratorService.class);
  }

  public static String uniqueID(Message replyMessage) {
    return replyMessage.getData().getString("ID");
  }

  ...
```

Helper method that extracts the encapsulated unique ID from Message

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...


  public static Intent makeIntent(Context context) {
    return new Intent(context,
                      UniqueIDGeneratorService.class);
  }
```

Helper method that extracts the encapsulated unique ID from Message

```
  public static String uniqueID(Message replyMessage) {
    return replyMessage.getData().getString("ID");
  }

  ...
```

UniqueID GeneratorService Implementation (Part 2)

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  private class RequestHandler extends Handler {



    public void handleMessage(Message request) {

      final Messenger replyMessenger = request.replyTo;



      mExecutor.execute(new Runnable() {
        public void run() {
           Message reply = generateUniqueID();
        ...
          replyMessenger.send(reply);
        ...
```

This class processes request Messages sent by the UniqueIDGeneratorActivity

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...

  private class RequestHandler extends Handler {



    public void handleMessage(Message request) {

      final Messenger replyMessenger = request.replyTo;



      mExecutor.execute(new Runnable() {
        public void run() {
           Message reply = generateUniqueID();
        ...
          replyMessenger.send(reply);
        ...
```

This class processes request Messages sent by the UniqueIDGeneratorActivity

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...


  private class RequestHandler extends Handler {
```

Hook method called when a request Message arrives from UniqueIDGeneratorActivity

```
    public void handleMessage(Message request) {

      final Messenger replyMessenger = request.replyTo;



      mExecutor.execute(new Runnable() {
        public void run() {
           Message reply = generateUniqueID();
        ...
          replyMessenger.send(reply);
        ...
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...


  private class RequestHandler extends Handler {



    public void handleMessage(Message request)

      final Messenger replyMessenger = request.replyTo;
```

Store the replyMessenger

```
      mExecutor.execute(new Runnable() {
        public void run() {
           Message reply = generateUniqueID();
        ...
          replyMessenger.send(reply);
        ...
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...


  private class RequestHandler extends Handler {



    public void handleMessage(Message request) {

      final Messenger replyMessenger = request.replyTo;

```

Put a runnable that generates a unique ID into the thread pool for subsequent concurrent processing

```
      mExecutor.execute(new Runnable() {
        public void run() {
           Message reply = generateUniqueID();
        ...
          replyMessenger.send(reply);
        ...
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...


  private class RequestHandler extends Handler {



    public void handleMessage(Message request) {

      final Messenger replyMessenger = request.replyTo;
```

Put a runnable that generates a unique ID into the thread pool for subsequent concurrent processing

```
      mExecutor.execute(new Runnable() {
        public void run() {
           Message reply = generateUniqueID();
        ...
          replyMessenger.send(reply);
        ...
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...


  private class RequestHandler extends Handler {



    public void handleMessage(Message request) {

      final Messenger replyMessenger = request.replyTo;
```

Put a runnable that generates a unique ID into the thread pool for subsequent concurrent processing

```
      mExecutor.execute(new Runnable() {
        public void run() {
           Message reply = generateUniqueID();
        ...
          replyMessenger.send(reply);
        ...
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...


  private class RequestHandler extends Handler {



    public void handleMessage(Message request) {

      final Messenger replyMessenger = request.replyTo;


      mExecutor.execute(new Runnable() {
        public void run() {
           Message reply = generateUniqueID();
        ...
          replyMessenger.send(reply);
        ...
```

Put a runnable that generates a unique ID into the thread pool for subsequent concurrent processing

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...
  private class RequestHandler extends Handler {
    private Message generateUniqueID() {
      String uniqueID;

      synchronized(this) {
        do { uniqueID = UUID.randomUUID().toString(); }
        while(uniqueIDs.getInt(uniqueID, 0) == 1);

        SharedPreferences.Editor editor = uniqueIDs.edit();
        editor.putInt(uniqueID, 1);
        editor.commit();
      }
      Message reply = Message.obtain();
      Bundle data = new Bundle();
      data.putString("ID", uniqueID);
      reply.setData(data);
      return reply;
    }
```

Return Message containing unique system-wide ID

UniqueIDGeneratorService Implementation

Protect the critical section

```
public class UniqueIDGeneratorService extends Service {
  ...
  private class RequestHandler extends Handler {
    private Message generateUniqueID() {
      String uniqueID;

      synchronized(this) {
        do { uniqueID = UUID.randomUUID().toString(); }
        while(uniqueIDs.getInt(uniqueID, 0) == 1);

        SharedPreferences.Editor editor = uniqueIDs.edit();
        editor.putInt(uniqueID, 1);
        editor.commit();
      }
      Message reply = Message.obtain();
      Bundle data = new Bundle();
      data.putString("ID", uniqueID);
      reply.setData(data);
      return reply;
    }
```

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...
  private class RequestHandler extends Handler {
    private Message generateUniqueID() {
      String uniqueID;

      synchronized(this) {
        do { uniqueID = UUID.randomUUID().toString(); }
        while(uniqueIDs.getInt(uniqueID, 0) == 1);

        SharedPreferences.Editor editor = uniqueIDs.edit();
        editor.putInt(uniqueID, 1);
        editor.commit();
      }
      Message reply = Message.obtain();
      Bundle data = new Bundle();
      data.putString("ID", uniqueID);
      reply.setData(data);
      return reply;
    }
```

Keep generating random UUID until one is found that's unique

developer.android.com/reference/java/util/UUID.html

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...
  private class RequestHandler extends Handler {
    private Message generateUniqueID() {
      String uniqueID;

      synchronized(this) {
        do { uniqueID = UUID.randomUUID().toString(); }
        while(uniqueIDs.getInt(uniqueID, 0) == 1);

        SharedPreferences.Editor editor = uniqueIDs.edit();
        editor.putInt(uniqueID, 1);
        editor.commit();
      }
      Message reply = Message.obtain();
      Bundle data = new Bundle();
      data.putString("ID", uniqueID);
      reply.setData(data);
      return reply;
    }
```

Keep generating random UUID until one is found that's unique

en.wikipedia.org/wiki/Universally_unique_identifier #Random_UUID_probability_of_duplicates

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...
  private class RequestHandler extends Handler {
    private Message generateUniqueID() {
      String uniqueID;

      synchronized(this) {
        do { uniqueID = UUID.randomUUID().toString(); }
        while(uniqueIDs.getInt(uniqueID, 0) == 1);

        SharedPreferences.Editor editor = uniqueIDs.edit();
        editor.putInt(uniqueID, 1);
        editor.commit();
      }
      Message reply = Message.obtain();
      Bundle data = new Bundle();
      data.putString("ID", uniqueID);
      reply.setData(data);
      return reply;
    }
```

Keep generating random UUID until one is found that's unique

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...
  private class RequestHandler extends Handler {
    private Message generateUniqueID() {
      String uniqueID;

      synchronized(this) {
        do { uniqueID = UUID.randomUUID().toString(); }
        while(uniqueIDs.getInt(uniqueID, 0) == 1);

        SharedPreferences.Editor editor = uniqueIDs.edit();
        editor.putInt(uniqueID, 1);
        editor.commit();
      }
      Message reply = Message.obtain();
      Bundle data = new Bundle();
      data.putString("ID", uniqueID);
      reply.setData(data);
      return reply;
    }
```

Keep generating random UUID until one is found that's unique

frameworks/base/core/java/android/app/SharedPreferencesImpl.java

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...
  private class RequestHandler extends Handler {
    private Message generateUniqueID() {
      String uniqueID;

      synchronized(this) {
        do { uniqueID = UUID.randomUUID().toString(); }
        while(uniqueIDs.getInt(uniqueID, 0) == 1);

        SharedPreferences.Editor editor = uniqueIDs.edit();
        editor.putInt(uniqueID, 1);
        editor.commit();
      }
      Message reply = Message.obtain();
      Bundle data = new Bundle();
      data.putString("ID", uniqueID);
      reply.setData(data);
      return reply;
    }
```

Add unique ID as a "key" to the persistent collection

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...
  private class RequestHandler extends Handler {
    private Message generateUniqueID() {
      String uniqueID;

      synchronized(this) {
        do { uniqueID = UUID.randomUUID().toString(); }
        while(uniqueIDs.getInt(uniqueID, 0) == 1);

        SharedPreferences.Editor editor = uniqueIDs.edit();
        editor.putInt(uniqueID, 1);
        editor.commit();
      }
      Message reply = Message.obtain();
      Bundle data = new Bundle();
      data.putString("ID", uniqueID);
      reply.setData(data);
      return reply;
    }
```

Add unique ID to Bundle & set as data in reply Message to Activity

UniqueIDGeneratorService Implementation

```
public class UniqueIDGeneratorService extends Service {
  ...
  private class RequestHandler extends Handler {
    private Message generateUniqueID() {
      String uniqueID;

      synchronized(this) {
        do { uniqueID = UUID.randomUUID().toString(); }
        while(uniqueIDs.getInt(uniqueID, 0) == 1);

        SharedPreferences.Editor editor = uniqueIDs.edit();
        editor.putInt(uniqueID, 1);
        editor.commit();
      }
      Message reply = Message.obtain();
      Bundle data = new Bundle();
      data.putString("ID", uniqueID);
      reply.setData(data);
      return reply;
    }
```

Add unique ID to Bundle & set as data in reply Message to Activity

UniqueIDGeneratorService Implementation

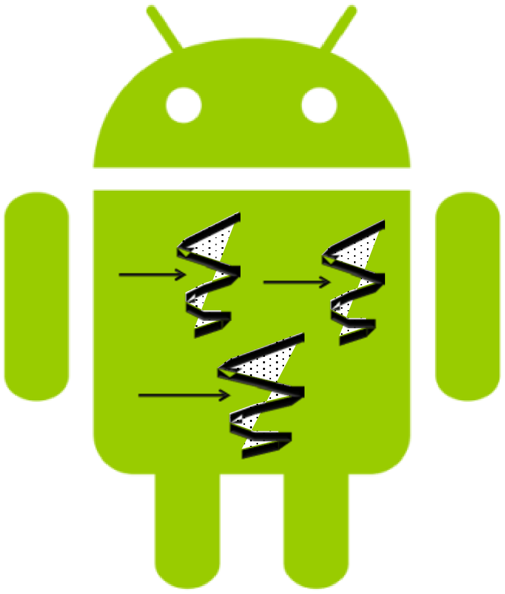
```
public class UniqueIDGeneratorService extends Service {
  ...
  private class RequestHandler extends Handler {
    private Message generateUniqueID() {
      String uniqueID;

      synchronized(this) {
        do { uniqueID = UUID.randomUUID().toString(); }
        while(uniqueIDs.getInt(uniqueID, 0) == 1);

        SharedPreferences.Editor editor = uniqueIDs.edit();
        editor.putInt(uniqueID, 1);
        editor.commit();
      }
      Message reply = Message.obtain();
      Bundle data = new Bundle();
      data.putString("ID", uniqueID);
      reply.setData(data);
      return reply;
    }
```

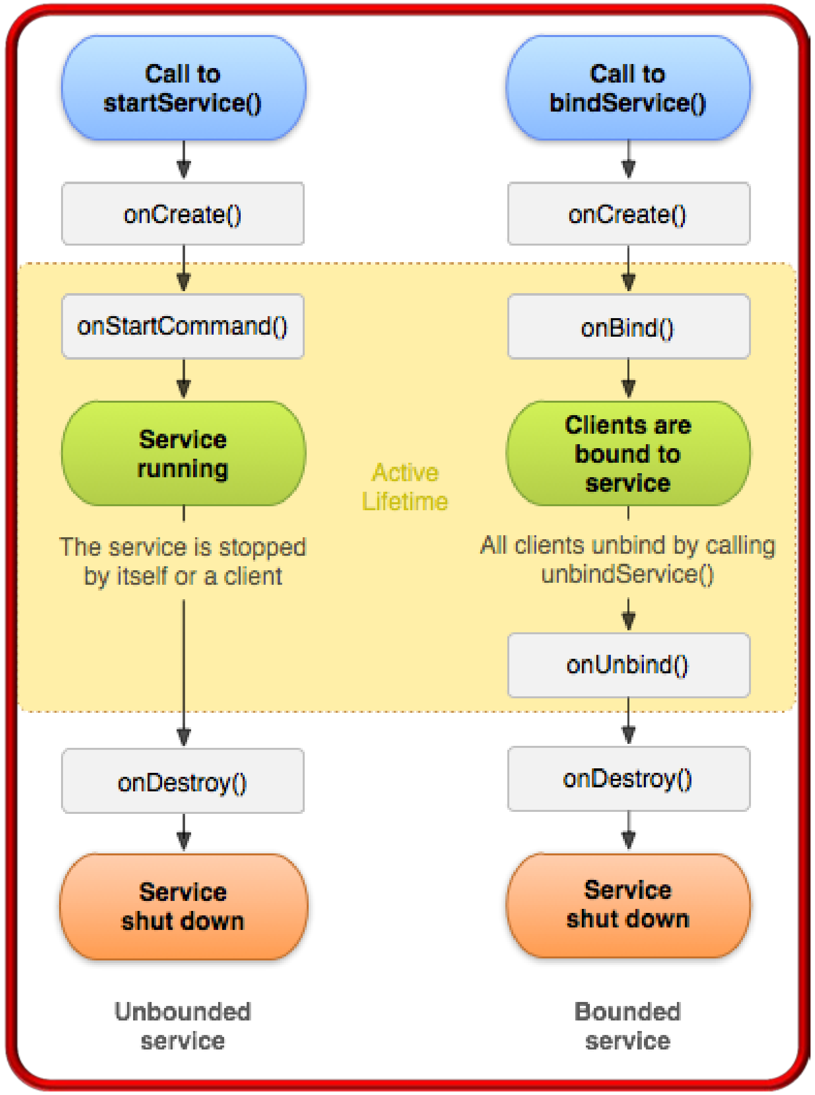
Add unique ID to Bundle & set as data in reply Message to Activity

Summary

Summary

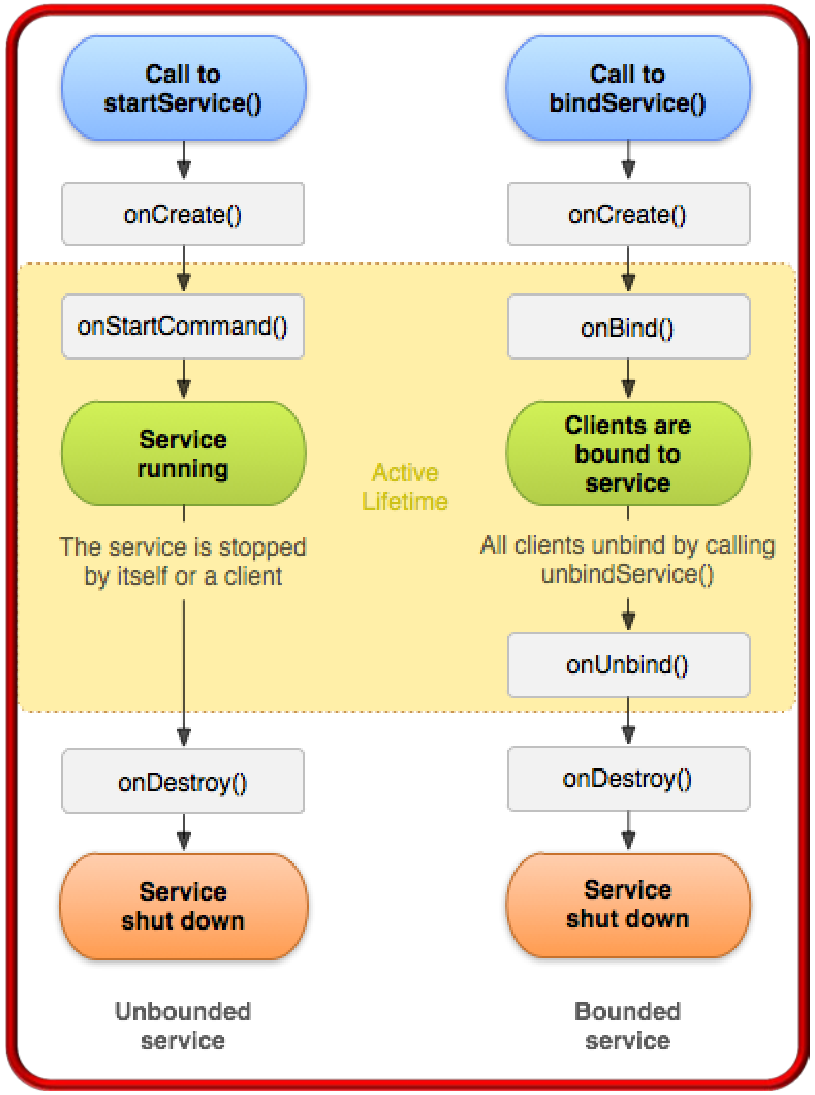
- Apps can use Services to implement long-duration operations in the background





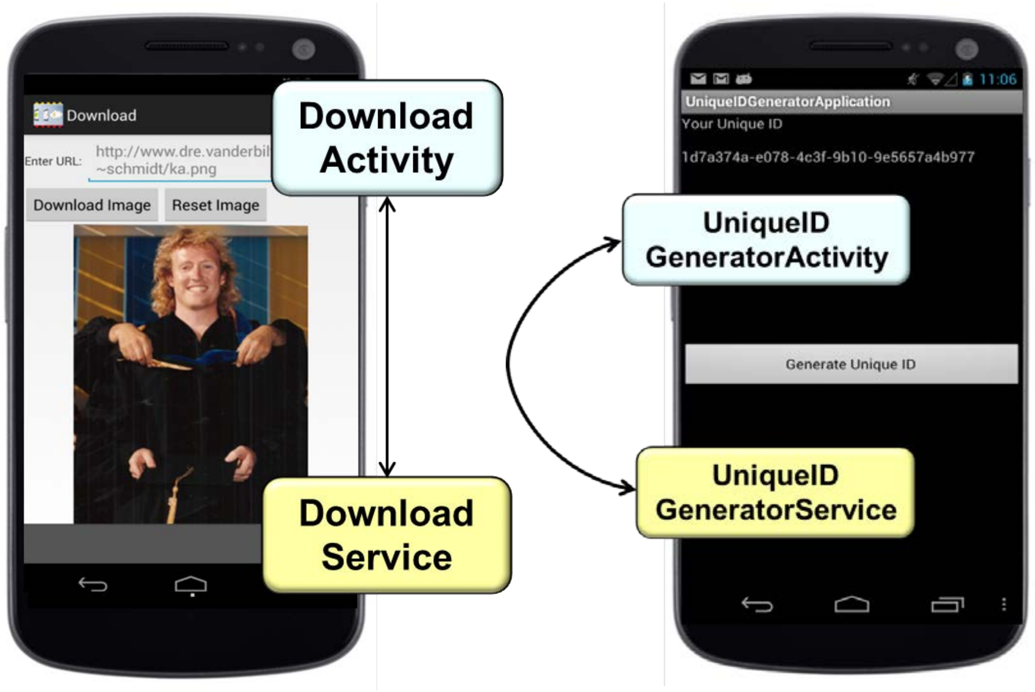
Summary

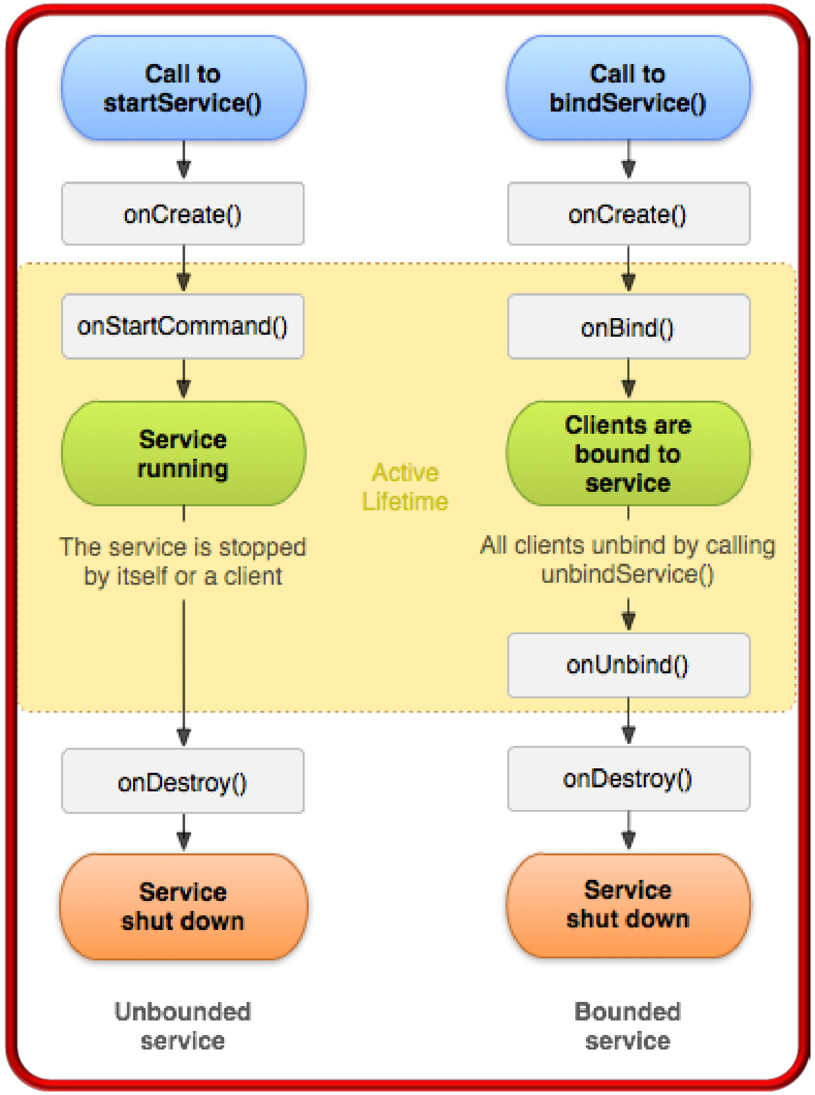
- Apps can use Services to implement long-duration operations in the background
  - They're useful for packaging a cohesive set of functionality into a form that's independent of the component that initiates it



Summary

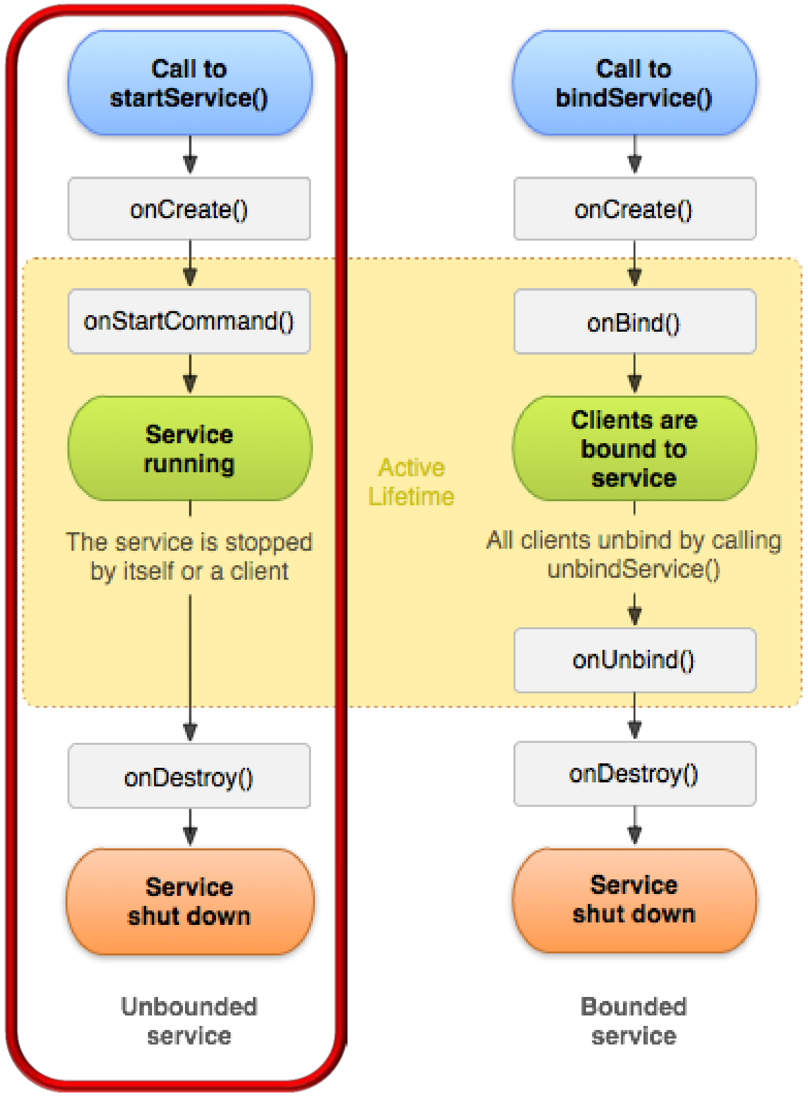
- Apps can use Services to implement long-duration operations in the background
  - They're useful for packaging a cohesive set of functionality into a form that's independent of the component that initiates it


Download
Enter URL:
http://www.dre.vanderbil ~schmidt/ka.png
Download Image
Reset Image
Download Activity
Download Service
UniqueIDGeneratorApplication
Your Unique ID
1d7a374a-e078-4c3f-9b10-9e5657a4b977
Generate Unique ID
UniqueID GeneratorActivity
UniqueID GeneratorService



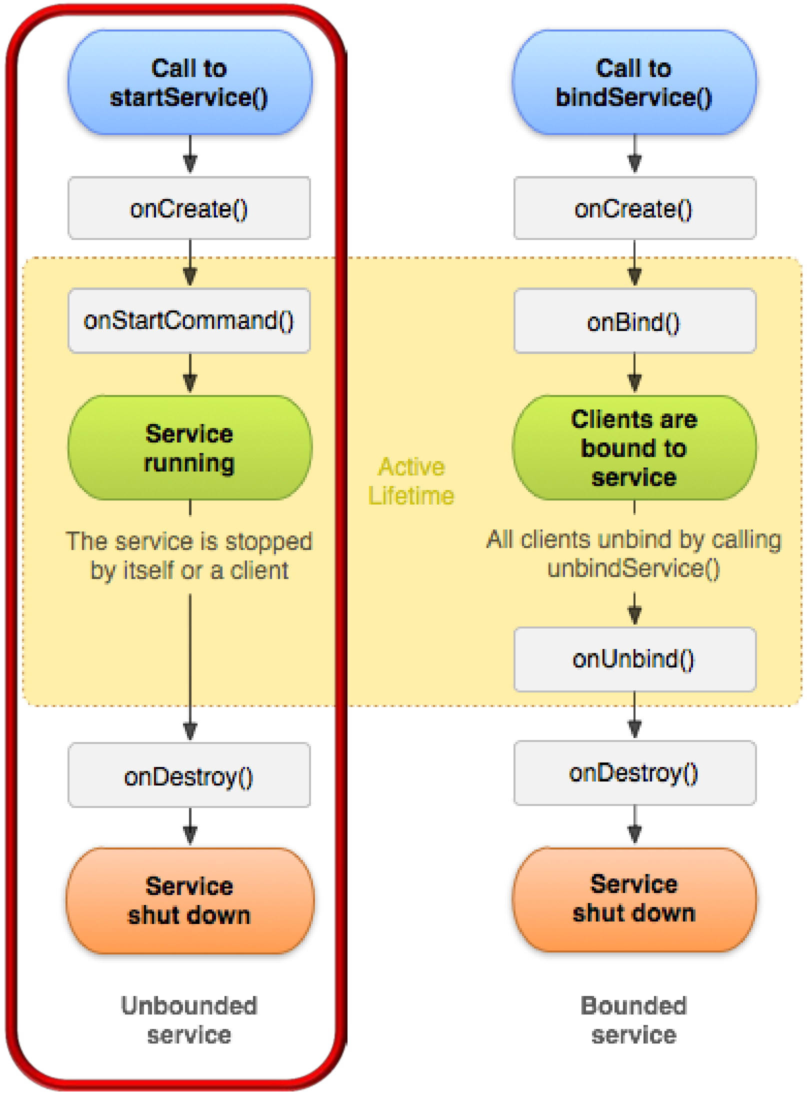
Summary

- Apps can use Services to implement long-duration operations in the background
- Started Services are easy to program for simple Activity-to-Service interactions



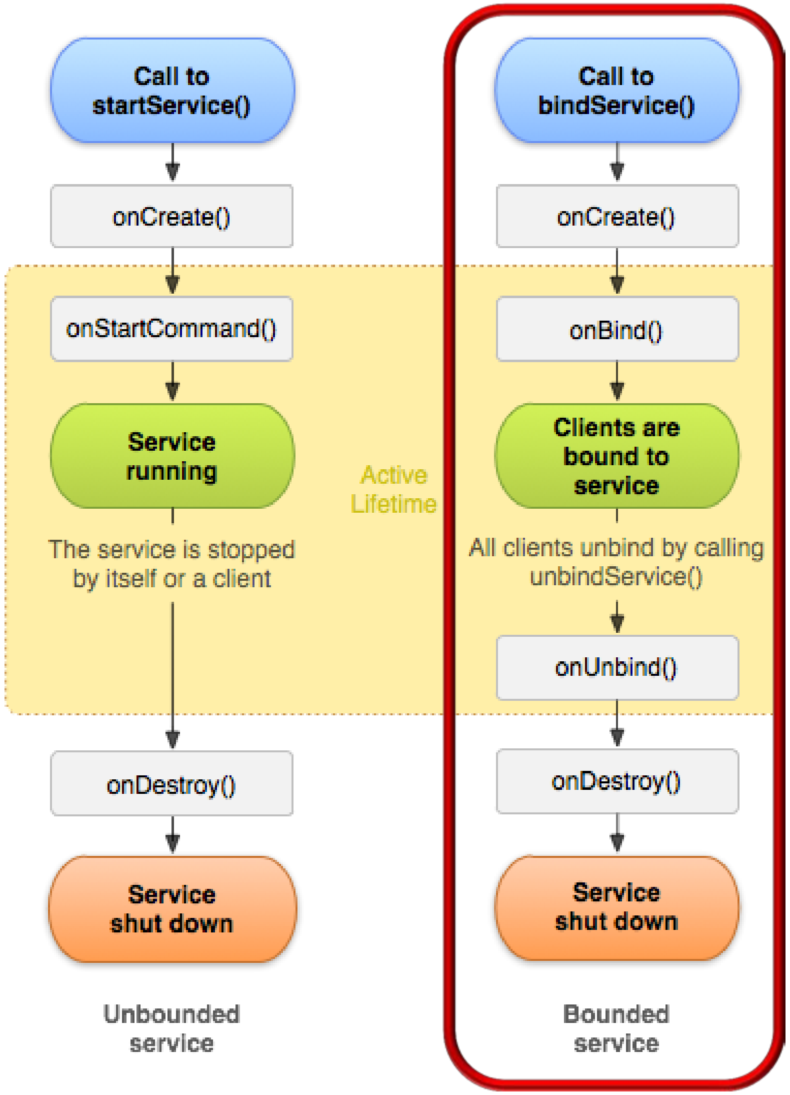
Summary

- Apps can use Services to implement long-duration operations in the background
- Started Services are easy to program for simple Activity-to-Service interactions
  - However, they require more complex & ad hoc programming for extended two-way conversations with clients



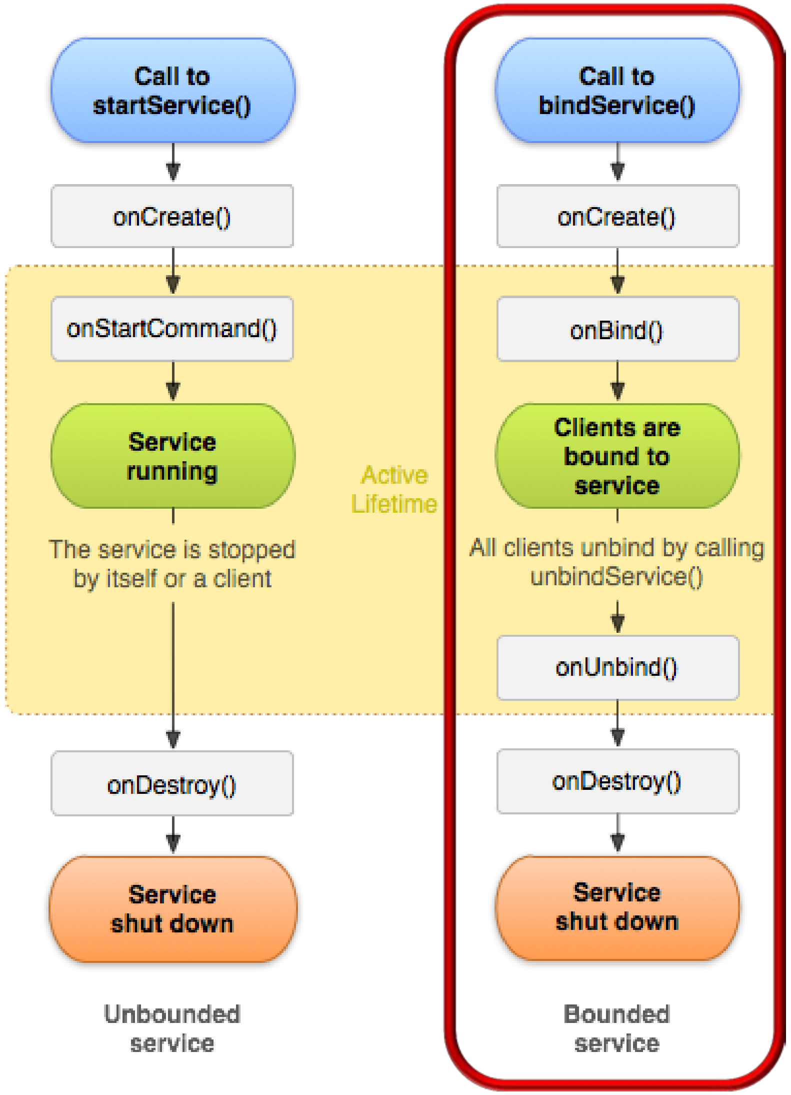
Summary

- Apps can use Services to implement long-duration operations in the background
- Started Services are easy to program for simple Activity-to-Service interactions
- Bound Services may be a better choice for more complex two-way interactions between Activities & Services



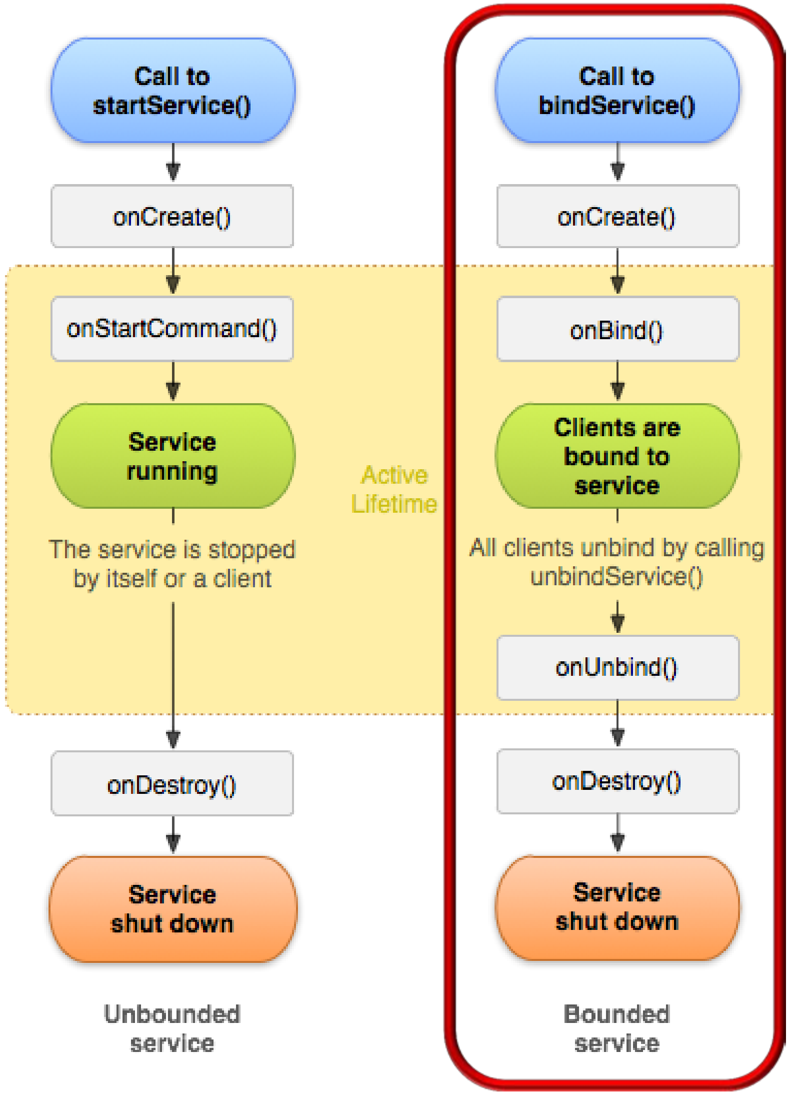
Summary

- Apps can use Services to implement long-duration operations in the background
- Started Services are easy to program for simple Activity-to-Service interactions
- Bound Services may be a better choice for more complex two-way interactions between Activities & Services
  - Supports two-way conversations



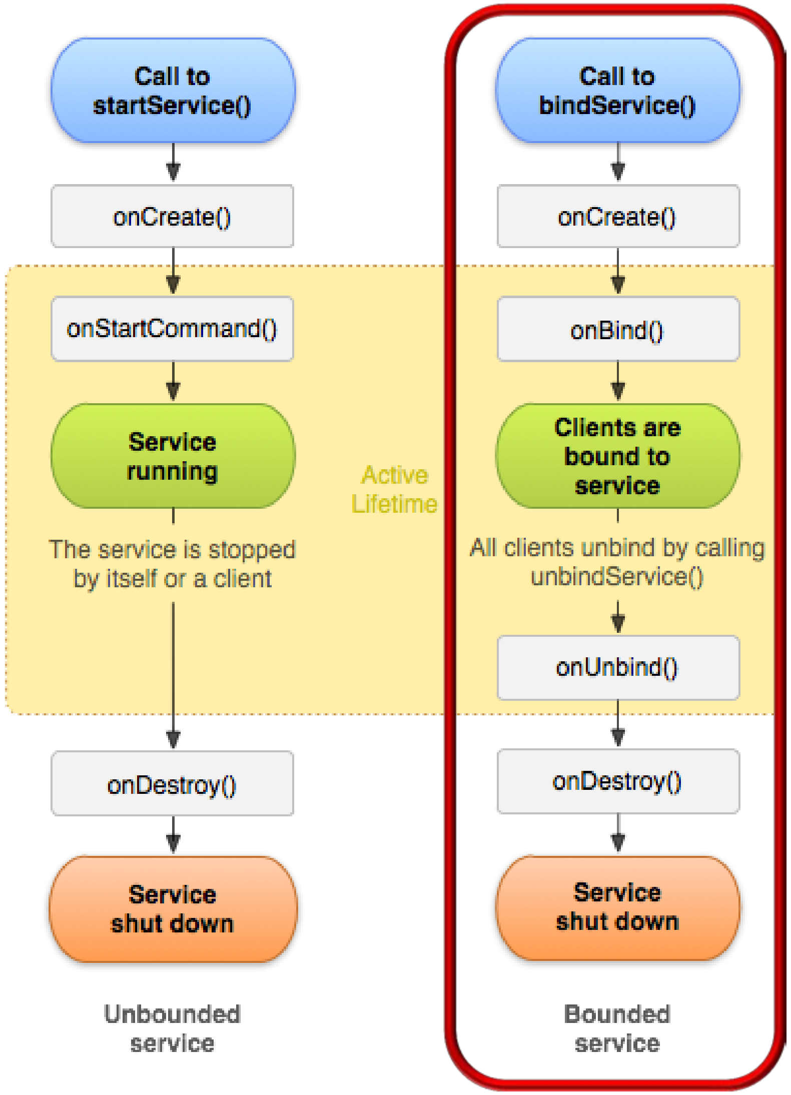
Summary

- Apps can use Services to implement long-duration operations in the background
- Started Services are easy to program for simple Activity-to-Service interactions
- Bound Services may be a better choice for more complex two-way interactions between Activities & Services
  - Supports two-way conversations
  - Many initialization & communication details are handled by Android



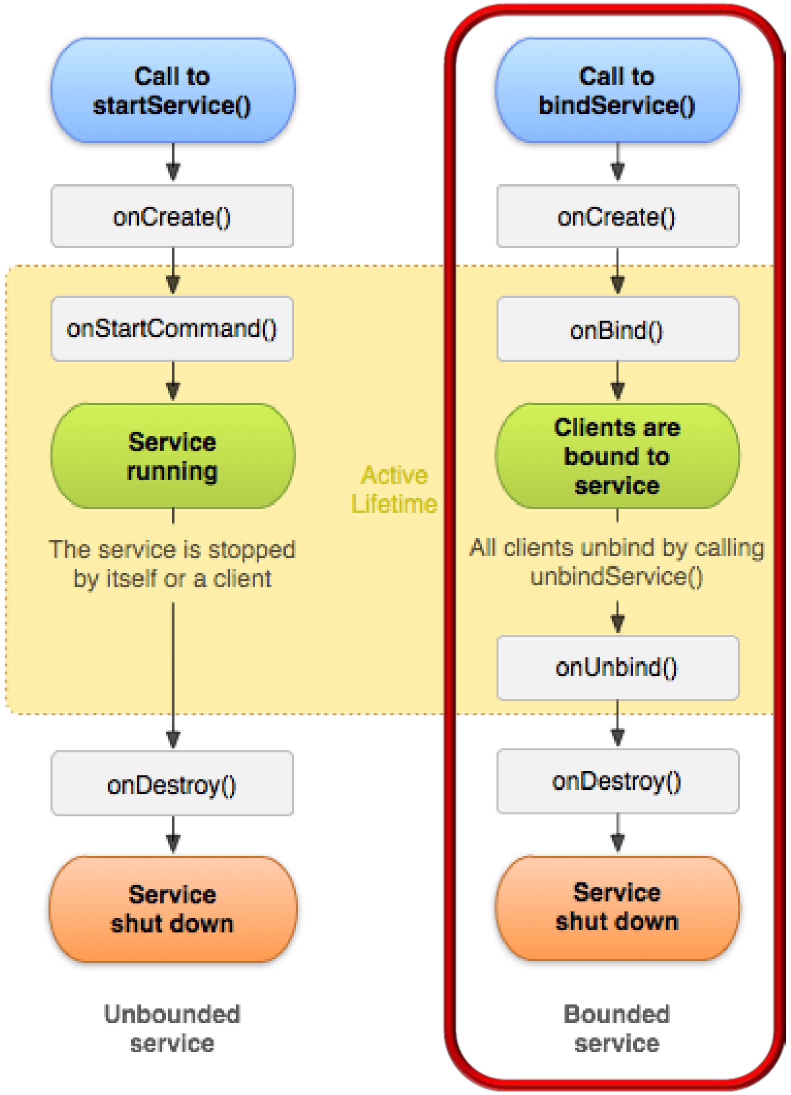
Summary

- Apps can use Services to implement long-duration operations in the background
- Started Services are easy to program for simple Activity-to-Service interactions
- Bound Services may be a better choice for more complex two-way interactions between Activities & Services
  - Supports two-way conversations
  - Many initialization & communication details are handled by Android
  - Their lifecycle is managed automatically by Android



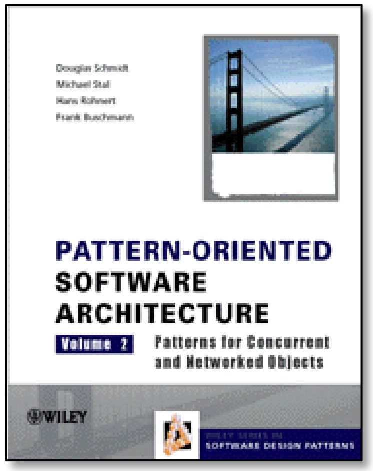
Summary

- Apps can use Services to implement long-duration operations in the background
- Started Services are easy to program for simple Activity-to-Service interactions
- Bound Services may be a better choice for more complex two-way interactions between Activities & Services
  - Supports two-way conversations
  - Many initialization & communication details are handled by Android
  - Their lifecycle is managed automatically by Android
  - However, programmers must understand details of the connection & interaction protocol



Summary

- Apps can use Services to implement long-duration operations in the background
- Started Services are easy to program for simple Activity-to-Service interactions
- Bound Services may be a better choice for more complex two-way interactions between Activities & Services
- Knowledge of *Broker* & Proxy patterns help clarify key roles & relationships in Bound Services



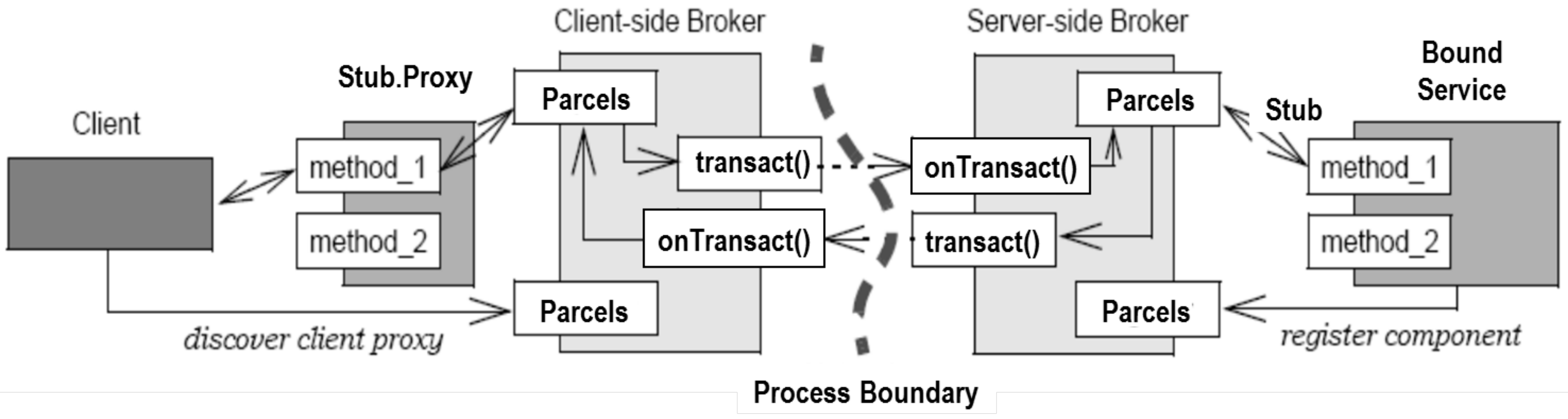
Douglas Schmidt
Michael Stal
Hans Rohnert
Frank Buschmann

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

Volume 2 Patterns for Concurrent and Networked Objects

WILEY

WILEY SERIES IN SOFTWARE DESIGN PATTERNS



See upcoming videos on "the *Broker* pattern"