



Android: HttpClient

Assane Seck
Ingénieur-Informaticien



Sommaire

HTTPCLIENT

Solutions réseau natives

HTTP avec HttpClient

Poster des formulaires

XML et JSON



LA RÈGLE DU THREAD UI

- ▶ La plupart des téléphones Android ont des solutions réseau intégrées
 - WiFi, Réseau 3G, 3G+, etc.
- ▶ Il y a différents moyens de se connecter:
 - WebKit
 - HTTP, XMPP, SMTP,... en utilisant des librairies de tiers ou incluses dans des JAR sur l'appareil
 - Sockets brutes
 - ...
- ▶ Android n'a pas de support natif pour SOAP et XML-RPC, mais la librairie Apache HttpComponents est incluse.
 - Construisez une surcouche ci-dessus pour le SOAP/XML-RPC
 - Utilisez le pour faire du REST: simples requêtes HTTP pour URL ordinaires sur toute une série de verbes HTTP (POST, GET, PUT, DELETE) avec des types de données formatés (XML, JSON, etc.)



Sommaire

HTTPCLIENT

Solutions réseau natives

HTTP avec HttpClient

Poster des formulaires

XML et JSON



ANDROID



SIMPLE HTTP AVEC HTTPCLIENT

- ▶ La première étape lors de l'utilisation de HttpClient : créer un HttpClient
 - HttpClient est une interface, vous pouvez utiliser DefaultHttpClient
- ▶ Ensuite : créez une HttpRequest
 - Il y a différentes implémentations par verbe:
 - HttpGet(String url)
 - HttpPost(String url)
 - Vous pouvez remplir l'URL et d'autres données de configuration
- ▶ Finalement, faites appel à execute() sur votre HttpClient en utilisant la HttpRequest comme paramètre
- ▶ Attention, Il faudra inscrire la permission Internet pour utiliser la technologie HttpClient!
- ▶ Dans build.gradle de app

```
android {  
    useLibrary 'org.apache.http.legacy'  
    compileSdkVersion 28  
}  
defaultConfig {
```

```
<uses-permission android:name="android.permission.INTERNET" />
```

SIMPLE HTTP AVEC HTTPCLIENT

- ▶ `execute()` sur `HttpRequest` renvoie un objet `HttpResponse`
 - Contient:
 - code de réponse
 - Headers HTTP
 - ...
- ▶ Il est possible d'utiliser différentes formes de `execute()` qui prend une instance de `ResponseHandler<String>` comme paramètre
 - C'est à vous de traiter la réponse !



SIMPLE HTTP AVEC HTTPCLIENT

```
protected class Downloader extends AsyncTask<String, Void, String> {  
    @Override  
    protected String doInBackground(String... params) {  
        return null;  
    }  
}
```

Le premier paramètre est le tableau d'élément reçu par la tâche (**doInBackground**)

Le deuxième paramètre définit le type de données transmis durant la progression du Traitement

Le troisième paramètre définit le type de retour de la tâche (**doInBackground**)



SIMPLE HTTP AVEC HTTPCLIENT

```
public class MainActivity extends Activity {
ProgressDialog progress;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
Button bt = (Button) findViewById(R.id.btValider);
progress = new ProgressDialog(MainActivity.this);
progress.setMessage("Connexion en cours .....");
bt.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
Downloader d = new Downloader();
d.execute("http://10.0.2.2/android/connexion.php?login=seck&pwd=seck");
}
});
}

protected class Downloader extends AsyncTask<String, Void, String> {
@Override
protected String doInBackground(String... params) {
HttpClient client = new DefaultHttpClient();
HttpGet getMethod = new HttpGet(params[0]);
try {
ResponseHandler<String> responseHandler = new BasicResponseHandler();
String responseBody = client.execute(getMethod, responseHandler);
return responseBody;
} catch (Throwable t) {
}
return null;
}
@Override
protected void onPreExecute() {
progress.show();
}
@Override
protected void onPostExecute(String result) {
Toast.makeText(getApplicationContext(), result, Toast.LENGTH_LONG)
.show();
progress.dismiss();
}}}
```


Sommaire

HTTPCLIENT

Solutions réseau natives
HTTP avec HttpClient
Poster des formulaires
XML et JSON



POSTER DES FORMULAIRES

- ▶ Etapes pour poster un formulaire:
 - Créez une instance de `HttpPost`
 - Créez une `List<NameValuePair>`
 - Ajoutez des `BasicNameValuePair` à cette liste
 - Ajoutez cette liste à votre instance de `HttpPost` en utilisant la fonction `setEntity()`
 - Vous pouvez choisir de rendre votre formulaire url-encodé
 - Utilisez `UrlEncodedEntity`



SIMPLE HTTP AVEC HTTPCLIENT

```
public class MainActivity extends Activity {
ProgressDialog progress;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
Button bt = (Button) findViewById(R.id.btValider);
progress = new ProgressDialog(MainActivity.this);
progress.setMessage("Connexion en cours .....");
bt.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
Downloader d = new Downloader();
d.execute("http://10.0.2.2/supdeco/connexionpost.php");
}
});
}
protected class Downloader extends AsyncTask<String, Void, String> {
@Override
protected String doInBackground(String... params) {
try {
HttpClient client = new DefaultHttpClient();
HttpPost post = new HttpPost(
params[0]);
List<NameValuePair> form = new ArrayList<NameValuePair>();
form.add(new BasicNameValuePair("login", "seck"));
form.add(new BasicNameValuePair("pwd", "seck"));
post.setEntity(new UrlEncodedFormEntity(form, HTTP.UTF_8));
ResponseHandler<String> responseHandler = new BasicResponseHandler();
String responseBody = client.execute(post, responseHandler);
return responseBody;
} catch (Exception e) {
return null;
}
}
@Override
protected void onPreExecute() {
progress.show();
}
@Override
protected void onPostExecute(String result) {
Toast.makeText(getApplicationContext(), result, Toast.LENGTH_LONG)
.show();
progress.dismiss();
}}}
```

Sommaire

HTTPCLIENT

Solutions réseau natives

HTTP avec HttpClient

Poster des formulaires

XML et JSON



XML et JSON

- ▶ Nous pouvons utiliser deux manières de parser les données:
 - XML
 - W3C DOM traditionnel : `package org.w3c.dom`
 - Parser SAX : `org.xml.sax`
 - API XMLPullParser : `org.xmlpull`
 - JSON
 - `org.json`
 - `new JSONObject(String jsonString)`



XmlPullParser

- ▶ parseur XML poussé par des event (obtenu via la fonction `getEventType()` sur l'instance du XML pull parser) :

- Type d'event :

`XmlPullParser.START_TAG`, `.END_TAG`, `.END_DOCUMENT`,
`.TEXT`,...

- `getName()` afin d'obtenir le nom du tag
- `getText()` afin d'obtenir le texte
- `next()` afin d'aller vers le prochain élément

- ▶ Plus d'infos :

<http://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html>



XmlPullParser: exemple

```
public class SimpleXmlPullParser {
    public static void main(String args[]) throws XmlPullParserException,
        IOException {
        XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
        factory.setNamespaceAware(true);
        XmlPullParser xpp = factory.newPullParser();
        xpp.setInput(new StringReader("<foo>Hello World!</foo>"));
        int eventType = xpp.getEventType();
        while (eventType != XmlPullParser.END_DOCUMENT) {
            if (eventType == XmlPullParser.START_DOCUMENT) {
                Log.i("XML", "Start document");
            } else if (eventType == XmlPullParser.START_TAG) {
                Log.i("XML", "Start tag " + xpp.getName());
            } else if (eventType == XmlPullParser.END_TAG) {
                Log.i("XML", "End tag " + xpp.getName());
            } else if (eventType == XmlPullParser.TEXT) {
                Log.i("XML", "Text " + xpp.getText());
            }
            eventType = xpp.next();
        }
        Log.i("End document");
    }
}
```

XML DOM PARSER

```
public class MainActivity extends Activity {

    public void buildForecasts(String raw) throws Exception {
        DocumentBuilder builder=DocumentBuilderFactory
            .newInstance()
            .newDocumentBuilder();
        Document doc=builder.parse(new InputSource(new StringReader(raw)));
        NodeList times=doc.getElementsByTagName("start-valid-time");

        for (int i=0;i<times.getLength();i++) {
            Element time=(Element)times.item(i);
            Forecast forecast=new Forecast();
            forecasts.add(forecast);
            forecast.setTime(time.getFirstChild().getNodeValue());
        }
        NodeList temps=doc.getElementsByTagName("value");

        for (int i=0;i<temps.getLength();i++) {
            Element temp=(Element)temps.item(i);
            Forecast forecast=forecasts.get(i);
            forecast.setTemp(new Integer(temp.getFirstChild().getNodeValue()));
        }
        NodeList icons=doc.getElementsByTagName("icon-link");
        for (int i=0;i<icons.getLength();i++) {
            Element icon=(Element)icons.item(i);
            Forecast forecast=forecasts.get(i);
            forecast.setIcon(icon.getFirstChild().getNodeValue());
        }
    }
}
```


Retrofit, dependencies

```
compile 'com.google.code.gson:gson:2.6.2'  
compile 'com.squareup.retrofit2:converter-gson:2.1.0'  
compile 'com.squareup.okhttp3:logging-interceptor:3.4.1'  
compile 'com.squareup.okhttp3:okhttp:3.4.1'
```

Retrofit, APIClient

```
private static Retrofit retrofit = null;
```

```
HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();  
interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
```

```
OkHttpClient client = new OkHttpClient  
    .Builder()  
    .addInterceptor(interceptor)  
    .build();
```

```
retrofit = new Retrofit.Builder()  
    .baseUrl("http://ip:8080/")  
    .addConverterFactory(GsonConverterFactory.create())  
    .client(client)  
    .build();
```

Retrofit, APIInterface

```
interface APIInterface {  
  
    @GET("user/connexion/{login}/{password}")  
    Call<User> connexion(@Path("login") String login, @Path("password") String password);  
  
    @POST("user/connexionp")  
    Call<User> connexionp(@Body User user);  
  
}
```

Retrofit, APIInterface

```
interface APIInterface {  
  
    @GET("user/connexion/{login}/{password}")  
    Call<User> connexion(@Path("login") String login, @Path("password") String password);  
  
    @POST("user/connexionp")  
    Call<User> connexionp(@Body User user);  
  
}
```

Retrofit, GET

```
APIInterface apiInterface = APIClient
    .getClient()
    .create(APIInterface.class);
```

```
Call<User> call = apiInterface.connexion (login, password);
call.enqueue(new Callback<User>() {
    @Override
    public void onResponse(Call<User> call, Response<User> response) {

        User user = response.body();
        Log.v("User:", user.status+"");

    }

    @Override
    public void onFailure(Call<User> call, Throwable t) {

        call.cancel();

    }

});
```

Retrofit, POST

```
Call<User> call = apiInterface.connexionp (user);
call.enqueue(new Callback<User>() {
    @Override
    public void onResponse(Call<User> test, Response<User> response) {

        User user = response.body();
        Log.v("User:", user.status+ "");

    }

    @Override
    public void onFailure(Call<User> call, Throwable t) {

        call.cancel();

    }

});
```

Retrofit, annotation

```
public class User {  
  
    @SerializedName("id")  
    public int id;  
    @SerializedName("firstname")  
    public String firstname;  
    @SerializedName("lastname")  
    public String lastname;  
    @SerializedName("login")  
    public String login;  
    @SerializedName("password")  
    public String password;  
    @SerializedName("status")  
    public String status;  
  
}
```

OkHttpClient

```
1. OkHttpClient client = new OkHttpClient();

2. Request.Builder builder = new Request.Builder();
   builder.url(params[0]);

3. Request request = builder.build();

   Response response = client.newCall(request).execute();
   return response.body().string();
```


OkHttpClient, plugin

implementation '**com.squareup.okhttp3:okhttp:3.4.1**'

Permission

▶ **<uses-permission**
android:name="android.permission.INTERNET"
/>

OkHttpClient, Security

<manifest

xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.okhttpproject">

<uses-permission android:name="android.permission.INTERNET"/>

<application

android:usesCleartextTraffic="true"

android:allowBackup="true"

android:icon="@mipmap/ic_launcher"

android:label="@string/app_name"

android:roundIcon="@mipmap/ic_launcher_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

OkHttpClient-GET

► OkHttpClient client = **new** OkHttpClient();

```
Request request = new Request.Builder()
    .url(url)
    .build();
```

```
client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {
        call.cancel();
    }
})
```

```
    @Override
    public void onResponse(Call call, Response response) throws IOException {

        final String myResponse = response.body().string();
    }
}
```

OkHttpClient-POST

```
OkHttpClient client = new OkHttpClient();
RequestBody formBody = new FormBody.Builder()
    .add("first_name", "test")
    .add("last_name", "ddd")
    .add("degrees", "BAC")
    .add("formation", "GL")
    .build();
Request request = new Request.Builder()
    .url(url)
    .post(formBody)
    .build();
client.newCall(request).enqueue(new Callback
```

OkHttpClient, Ecran

```
MainActivity.this.runOnUiThread(new Runnable() {  
    @Override  
    public void run() {  
        try {  
  
            JSONObject json = new JSONObject(myResponse);
```


Update GUI

► Handler **mHandler** = **new** Handler(Looper.getMainLooper());

```
mHandler.post(new Runnable() {  
    @Override  
    public void run() {  
        mTextView.setText(« my text »);  
    }  
});
```


OkHttpClient, Security





► Config Windows

 Pare-feu Windows Defender avec fonctions avancées de sécurité

Fichier Action Affichage ?



 Pare-feu Windows Defender av

-  Règles de trafic entrant
-  Règles de trafic sortant
-  Règles de sécurité de conne
- >  Analyse

Pare-feu Windows Defender avec fonctions avancées de sécurité sur Ordinateur local



Le Pare-feu Windows Defender avec fonctions avancées de sécurité offre une sécurité réseau pour les ordinateurs Windows.

Vue d'ensemble



Pour votre sécurité, certains paramètres sont contrôlés par la stratégie de groupe.

Profil de domaine



Le Pare-feu Windows Defender est désactivé.

Profil privé



Le Pare-feu Windows Defender est activé.



Les connexions entrantes qui ne correspondent pas à une règle sont autorisées.



Les connexions sortantes qui ne correspondent pas à une règle sont autorisées.

Le profil public est actif



Le Pare-feu Windows Defender est activé.



Les connexions entrantes qui ne correspondent pas à une règle sont autorisées.



Les connexions sortantes qui ne correspondent pas à une règle sont autorisées.



[Propriétés du Pare-feu Windows Defender](#)

Démarrer

Authentifier les communications entre les ordinateurs

Créez des règles de sécurité de connexion afin de spécifier comment et quand les connexions entre les ordinateurs sont authentifiées et protégées à l'aide de la sécurité du protocole Internet (IPsec).



[Règles de sécurité de connexion](#)

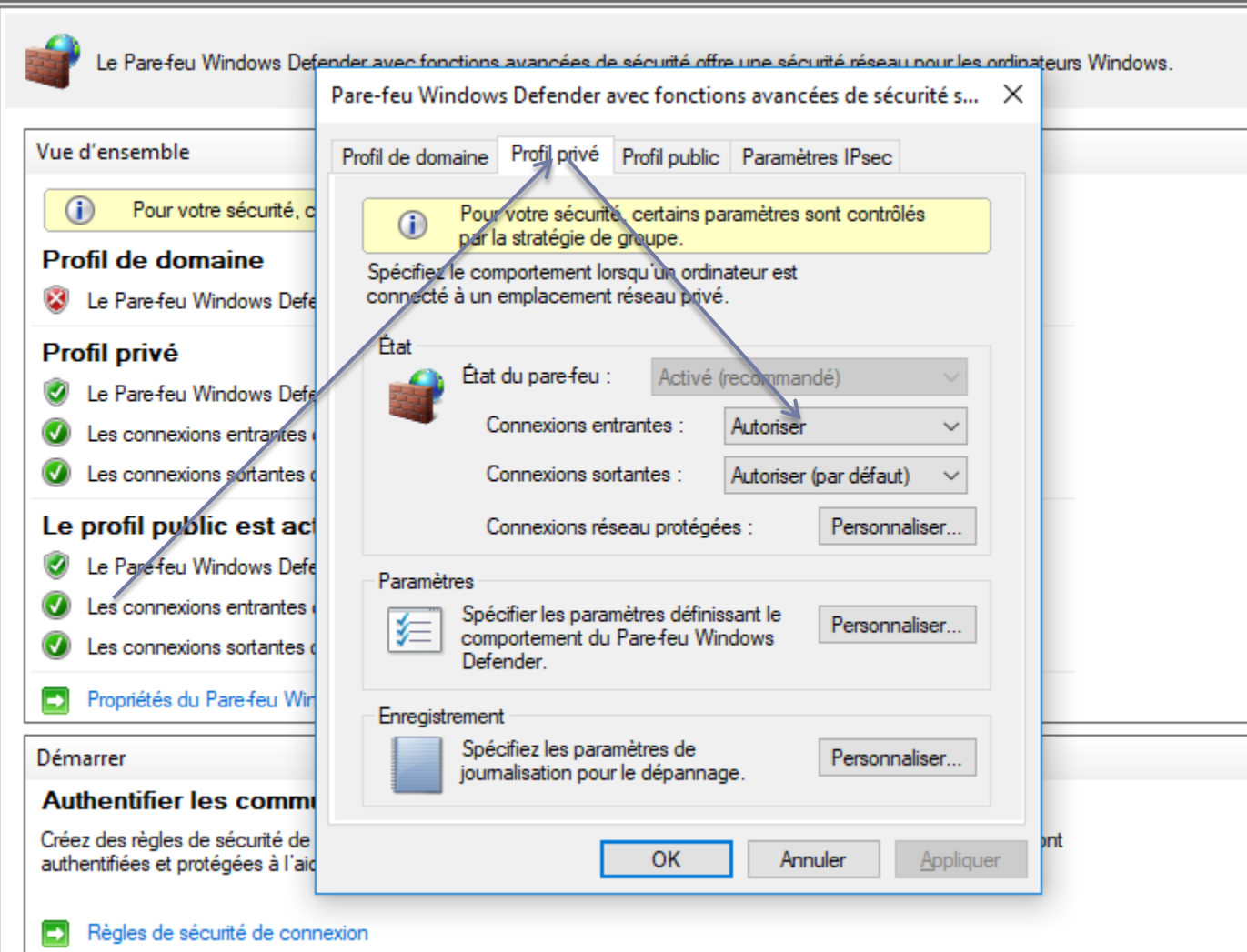
Afficher et créer des règles de pare-feu

Créez des règles de pare-feu pour autoriser ou bloquer les connexions vers des programmes ou ports spécifiques. Vous ne pouvez autoriser une connexion que si elle est authentifiée ou si elle provient d'un utilisateur, groupe ou ordinateur autorisé. Par défaut, les connexions entrantes sont bloquées si elles ne satisfont pas à une règle qui les autorise, et les connexions sortantes sont autorisées si elles ne satisfont pas à une règle qui les bloque.

OkHttpClient, Security

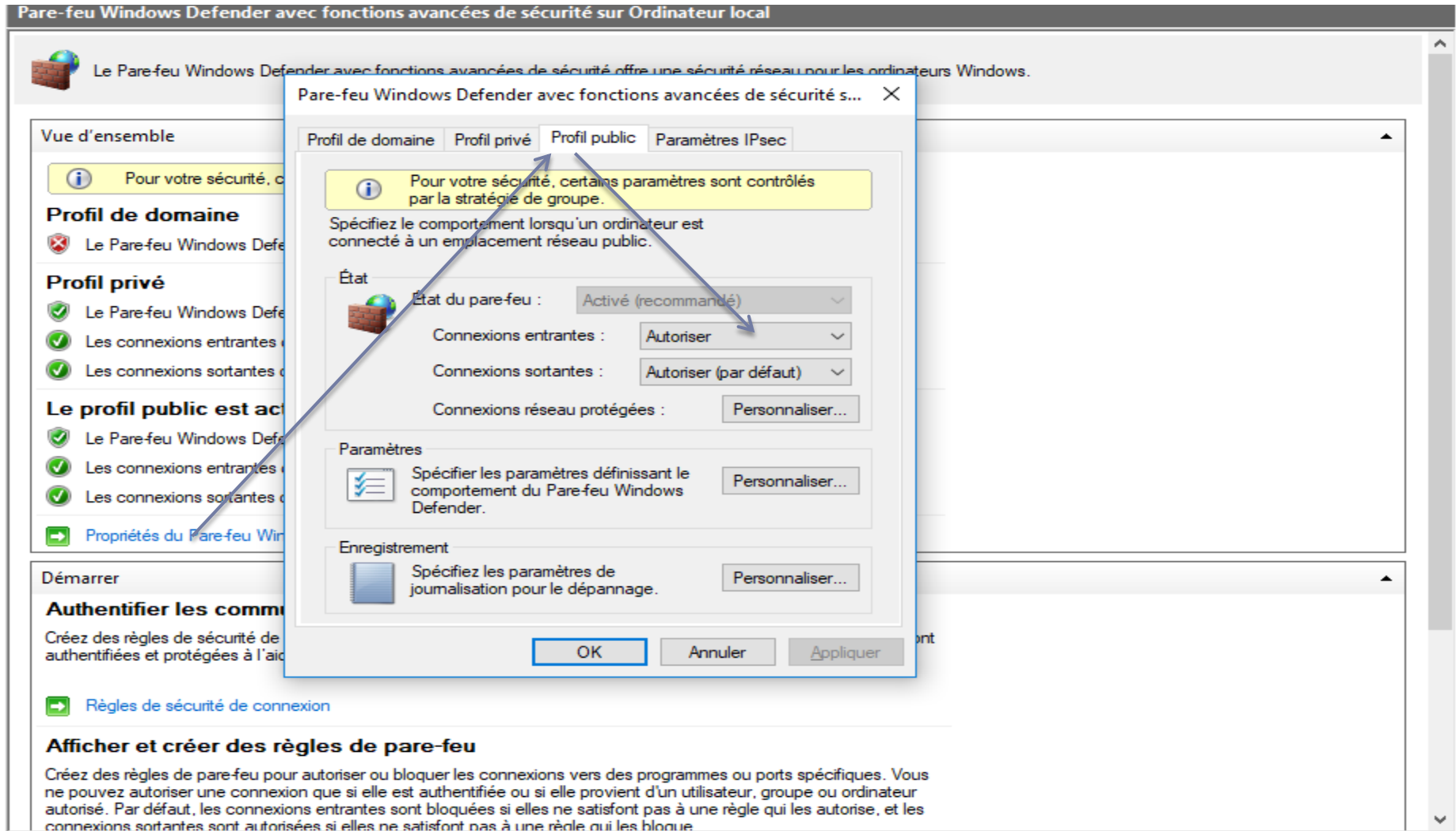
► Configuration

Pare-feu Windows Defender avec fonctions avancées de sécurité sur Ordinateur local



OkHttpClient, Security

► Config Windows



Test

- ▶ Pour ne pas avoir de blocage de permission
- ▶ Il faut desinstaller d'abord l'application sur l'émulateur ou sur le telephone avant de tester le OkHttpClient
- ▶ Proxy, il faut regarder si le parefeux ne sont pas activés (Connexion entrante et sortante)
- ▶ Demander au server Web d'accepter les connexion externes

FIN

