NAME

ares_library_init_android − c-ares library Android initialization

SYNOPSIS

#include <ares.h>

int ares_library_init_android(jobject *connectivity_manager*)

int ares_library_android_initialized();

void ares_library_init_jvm(JavaVM **jvm*)

DESCRIPTION

The *ares_library_init_android(3)* function performs initializations internally required by the c-ares library when used on Android. This can take place anytime after *ares_library_init(3)*. It must take place after *ares_library_init_jvm*. ares_library_init_android must be called before DNS resolution will work on Android 8 (Oreo) or newer when targetSdkVersion is set to 26+.

As of Android 8 (API level 26) getting DNS server information has becomei more restrictive and can only be accessed using the Connectivity Manager. It is necessary to pass the connectivity manager to c-ares via JNI. Also, the ACCESS_NETWORK_STATE permission must be present in the Android application.

Android older than 8 do not need to to be initialized as they are less restrictive. However, this is a run time not compile time limitation. Proper Android initialization should take place regardless of the targeted Android version.

Deinitialization will take place though *ares_library_cleanup(3)*.

The **ares_library_init_jvm** function allows the caller to register the JVM with c-ares. It's meant to be called during JNI_OnLoad because you're guaranteed to have the JVM in that function. The JVM is required in order to use the Connectivity Manager registered using *ares_library_init_android(3)*. This must be call before *ares_library_init_android(3)*.

The **ares_library_android_initialized** function can be used to check whether c-ares has been initialized for use with Android.

RETURN VALUES

ARES_SUCCESS will be returned on success otherwise an error code will be returned.

THREAD SAFETY

These init functions are not thread safe. You have to call it once the program has started, but this call must be done before the program starts any other thread. This is required to avoid potential race conditions in library initialization, and also due to the fact these might call functions from other libraries that are thread unsafe, and could conflict with any other thread that is already using these other libraries.

JNI

Accessing the Connectivity Manager though Java:

Register the *ares_library_android_init*.

```
 static JNINativeMethod funcs[] = {
 { "initialize_native",     "(Landroid/net/ConnectivityManager;)I",
   (void *)&ares_library_init_android}
 };

 JNIEXPORT jint JNICALL JNI_OnLoad(JavaVM *vm, void *reserved)
 {
```

```
  JNIEnv *env = NULL;
  jclass  cls = NULL;
  jint    res;

  if ((*vm)->GetEnv(vm, (void **)&env, JNI_VERSION_1_6) != JNI_OK)
    return -1;

  cls = (*env)->FindClass(env, JNIT_CLASS);
  if (cls == NULL)
    return -1;

  res = (*env)->RegisterNatives(env, cls, funcs, sizeof(funcs)/sizeof(funcs[0]));
  if (res != 0)
    return -1;

  ares_library_init_jvm(vm);
  return JNI_VERSION_1_6;
 }
```

Calling the registered function from Java:

```
 public class MyObject {
  static {
   System.loadLibrary("cares");
  }

  private static native boolean initialize_native(ConnectivityManager
   connectivity_manager);

  public static boolean initialize(Context context) {
   initialize_native((ConnectivityManager)context.getSystemService(Context.CONNECTIVITY_SERVICE));
  }
 }
```

Initializing the Connectivity Manager in JNI directly using an Android Context. It is assumed the JVM has already been registered through *JNI_OnLoad*.

```
 void initialize(jobject android_context)
 {
  jclass obj_cls = jni_get_class(env, "android/content/Context");
  jmethodID obj_mid = jni_get_method_id(env, obj_cls, "getSystemService", "(Ljava/lang/String;)Ljava/lang/Object;");
  jfieldID fid = (*env)->GetStaticFieldID(env, obj_cls, "CONNECTIVITY_SERVICE", "Ljava/lang/String;");
  jstring str = (*env)->GetStaticObjectField(env, obj_cls, fid);
  connectivity_manager = (*env)->CallObjectMethod(env, android_context, obj_mid, str);
  if (connectivity_manager == NULL)
   return;
  ares_library_init_android(connectivity_manager);
 }
```

AVAILABILITY

This function was first introduced in c-ares version 1.15.0.

SEE ALSO

ares_library_init(3), **ares_library_cleanup**(3),