Getting started with NaCl

Published 2014-10-27 | (Compatible with SDK 4.5,5.0,5.1 and 2013,2014 models)

> Introduction to the Native Client technology, implemented in Samsung Smart TV.

Contents

Native Client - Introduction

Native Client is a technology introduced by Google. Its main concept is to run applications written in any programming language from JavaScript/HTML page in your browser. Currently supported languages are C and C++, and the compiled Native Client modules can be executed under Chrome browser.

The Native Client module is a binary with .nexe extension. It is written in C/C++ language, but requires the use of a set of interfaces called Pepper API (PPAPI). The code is compiled into an executable form (.nexe file), so it can be executed by the Native Client Plug-in provided with the browser. NaCl application consists of two major components: the web part - HTML page and JavaScript source, and the NaCl module. The module is embedded in HTML page that is displayed by the browser.

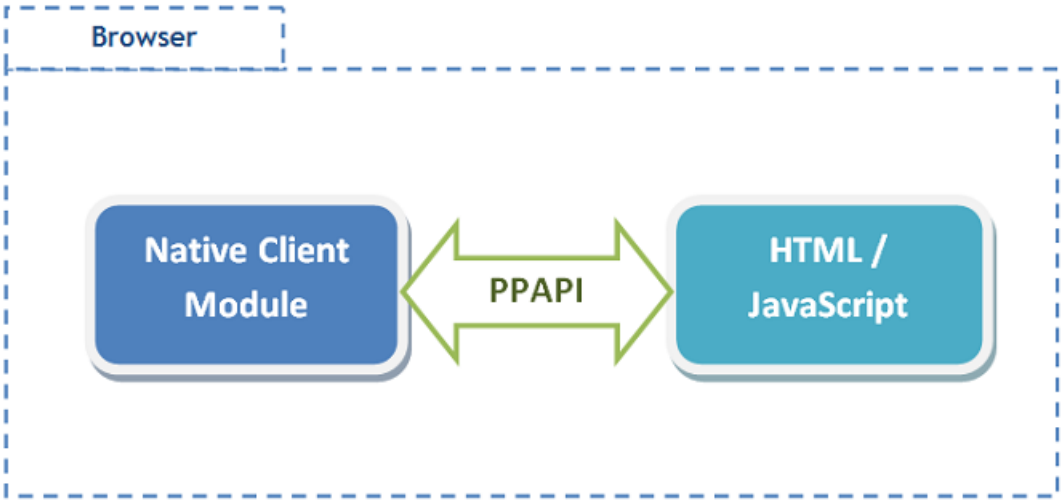

Figure 1 Native Client embedded within a web page.

Using Native Client opens up new possibilities for the web applications. The **Native Client** site names these as the main advantages of using NaCl technology in web applications:

More features

Because Native Client enables running modules written in C and C++, there is a possibility to render 2D and 3D graphics, play audio, access the file system, provide threading management and others.

Portability

Native Client is a sandboxing technology, so any environment supporting this technology will be able to execute provided module. There is only one catch - the modules compiled for native client are architecture specific.

Security

The Native Client module is executed in a sandbox, and the functionalities are provided by the Pepper API, so the system resources are safe. Sandboxes are briefly described in Sandboxing section of this document.

Easy migration

It will be easy to port already existing applications to be executed as a NaCl modules, as the Native Client supports C and C++ (in the future, the list of supported languages will be expanding).

Performance

The native code execution is faster than JavaScript, so this gives a chance to perform operations involving heavy computing or graphics rendering.

Native Client technology was implemented in Samsung Smart TV. This enables running NaCl applications with the browser in your Smart TV or prepare a widget with NaCl module. For easy development and testing, you can check your application using Samsung Smart TV Emulator.

> Note
>
> Native Client was implemented in Samsung Smart TVs. For easier application development and testing check out also Samsung Smart TV Emulator.

"P" is for portable

PNaCl - Portable Native Client, comes as the extension of Native Client technology. In PNaCl's case, the native code is first compiled into transitional form (.pexe file), which is distributed with the application. This intermediate form is architecture independent, and using a translation tool, it is translated to the Native Client module, dependent on the target's architecture (.nexe file).

The main advantage of using PNaCl is that the application developer does not need to worry about the architecture of the system that is going to be executed on, because the translation to the executable file is performed on the target system.

> Note
>
> **Native Client modules are architecture specific.** This means that in order to run the native module on a given platform, it has to be compiled according to the platform's architecture (ex. x86-32, x86-64).

Samsung Smart TV is performing the translation to "arm" executable file, just after the widget installation. The translation step is executed only once, during the installation process.

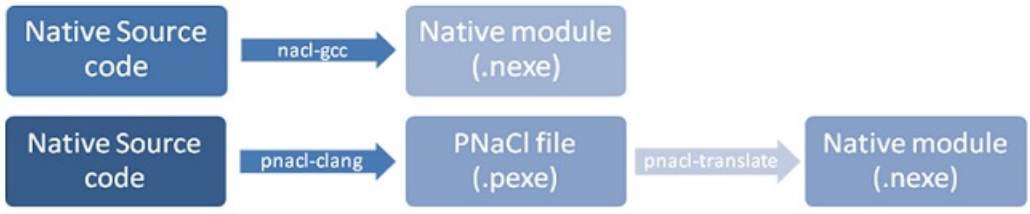

Figure 2 Building process with and without the PNaCl intermediate form.

More info

For more information, check the following sites:

Native Client Technical Overview

Native Client Project

NaCl application

PNaCl application for Samsung Smart TV is a standard web application extended with a NaCl module. The web application

can be either a widget, or HTML page loaded by the browser. At the stage of application development, the PNaCl module consists of files written in the native code (currently C or C++ is supported). The PNaCl application that can be distributed contains the source code built into .pexe binary. Later, when the application is installed, the PNaCl binary is translated into Native Client module by the target system. The result of this translation is a single .nexe file which can be executed on the target machine.

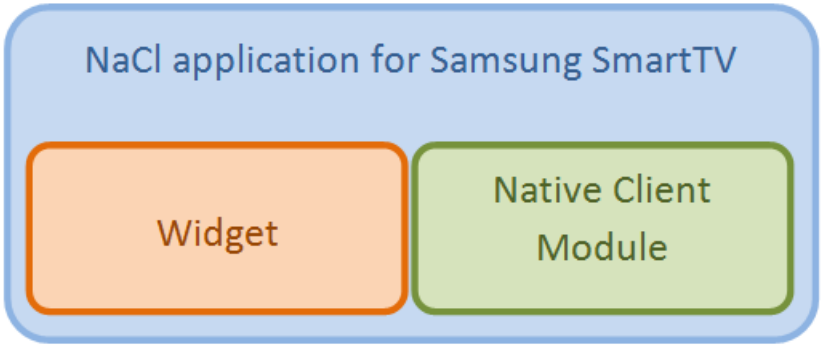

Figure 3 Structure of the application using NaCl for Samsung Smart TV.
The communication between the module, and the web application layer (JavaScript/HTML) is provided by Pepper Plug-in API (called Pepper API). Pepper API also allows the module to use such resources as graphics, audio and others. All together, these components create opportunities to develop web applications that are more advanced than the ones created without the Native Client technology.

Pepper API

Pepper Plug-in API (called Pepper API or PPAPI) is a set of interfaces, which allow to perform some key application operations like graphics 2D or 3D operations, file system access and other.

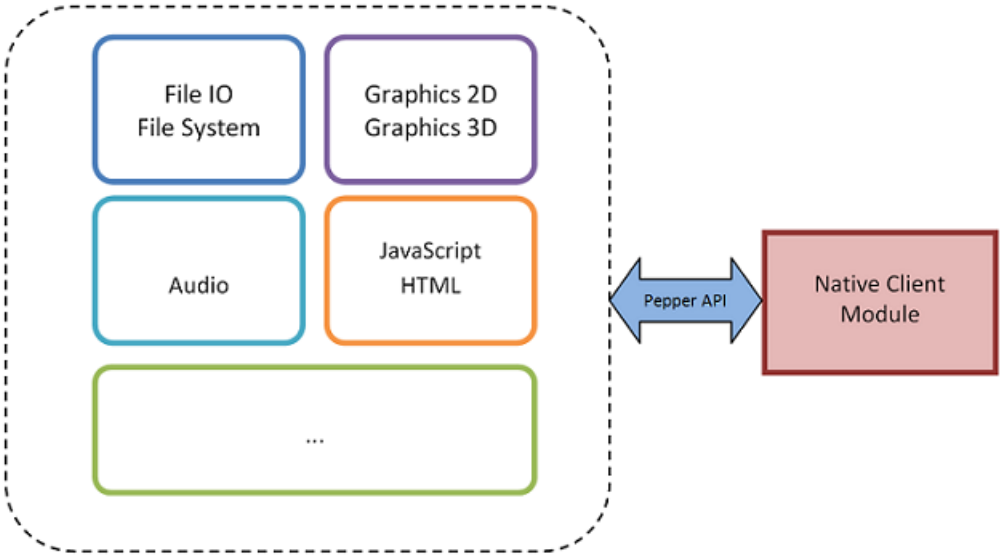

Figure 4 Pepper API.
Pepper API allows to call browser functions from the Native Client module, and provides interfaces for Native Client functions that can be called by the browser.

For the latest version of PPAPI see **Pepper C API** or **Pepper C++ API**. However, the Pepper API available on Samsung Smart TVs is a combination of the previous and latest versions of Pepper API. The full documentation and interfaces available for Samsung Smart TV development can be found at Pepper C API. There are Pepper API interfaces provided for C and C++ language. However the C++ interfaces are, in fact, an overlay for C functions.

Manifest file

The structure is determined by the JSON format, and the file extension is .nmf.

The structure of files acceptable by the Samsung Smart TV and Samsung Smart TV Emulator is a limited version of the manifest file described in NaCl Manifest Specification.

Native Client manifest file specifies which Native Client module is to be loaded for a given architecture. The manifest file has to be a valid JSON file, and has to contain a single “program” section, which specifies the executable Native Client modules that will be loaded for a given architecture.

The currently supported options by the Native Client are:
"x86-32" - for 32-bit architectures
"x86-64" - for 64-bit architectures
"arm" - for ARM
"portable" - for PNaCl

Each element corresponding to the target architecture ("x86-64", "x86-32", "arm") contains valid "url" element pointing to executable .nexe file.

Example of the contents of the manifest file for PNaCl application:

```
{
  "program": {
    "portable": {
      "pnacl-translate": {
        "url": "myapp.pexe"
      }
    }
  }
}
```

As Samsung Smart TV and Samsung Smart TV Emulator supports PNaCl (Portable Native Client) the section that should be used is “portable”.

The “portable” keyword is used for systems supporting PNaCl (Portable Native Client). The URL given in the “portable” caption is not an executable file. This is an intermediate form. The systems supporting PNaCl, before loading the Native Client module, perform translation to a .nexe file, based on the actual system architecture type.

Embedding NaCl module

To add a NaCl module to your application, just add the following code to the HTML body of the web page.

```
<object type="application/x-nacl" id="nacl_module" src="hello_world.nmf"></object>
```

The object **type** application/x-nacl signalizes that the following object is of a type that is executed by the NaCl plug-in.
The **src** attribute points to the manifest file. The manifest file contains the path to the file that contains the NaCl module.
Object **id** is a unique identifier of the HTML object.

Once the web page is loaded, an instance of the NaCl module is created.

NaCl module and NaCl instance

The NaCl instance is the object located on the web page. The module compiled to a .nexe file can have multiple instances - one for every object located on a web page. The module is loaded only once for an application, even if multiple instances occur on a web page.

Translation step

The native code can be recognized by the Native Client plug-in, once it has a .nexe form.

A PNaCl application comes with native code compiled to a portable form - a file with .pexe extension. The translation to the final, architecture-specific module, is performed during the installation process on Samsung Smart TV. In case of Samsung Smart TV Emulator, the translation is performed automatically during the launch of the application.

The translation is a process that consists of the following steps:

1. The manifest file is parsed, in order to find the valid path to the .pexe element

   The .pexe element path is located under the node “portable” in the JSON tree of the manifest file.
2. Once the file is found, the translation is performed, using pnacl-translate tool provided with the Native Client SDK.

The output of this translation is .nexe executable file. The output file is architecture specific, so it will be different for Samsung Smart TV, which is based on arm architecture, and for Samsung Smart TV Emulator, which can be executed under Linux or Windows platforms.

3. The manifest file is modified, and a node for the target architecture is placed under the “program” node in the manifest. The below example contains a modified manifest file for 64-bit system:

```
{
  "program": {
    "x86-64": {
      "url": "myapp.nexe"
    }
  }
}
```

The translation step in Samsung Smart TV Emulator, is performed whenever the application is launched. This way, all the modifications are included in the latest run of the application.

However, the translation on Samsung Smart TV is performed only once, during the widget’s installation process. The result of the translation is stored in the file system and is used whenever the widget is launched.

> Note
>
> **On Samsung Smart TV the translation step is performed only once**, during the widget’s installation process. The .nexe file, which is the result of the translation, is stored in the file system and is used whenever the widget is launched.

Sandboxing

Native Client technology provides a high level of security, due to the usage of sandboxing technology. Native Client technology uses **inner** and **outer** sandbox.

The **inner sandbox’s** task is to validate the Native Client module against security violations, before the module is executed. The check is performed during static analysis to ensure that the following conditions are satisfactory.

Calling (directly or indirectly) unsafe instructions is not permitted. The example instructions that are considered unsafe are syscall or lds.

Maintaining data integrity by disallowing loading and storing data outside the sandbox.

The **outer sandbox** mediates calls between the Native Client module and the Browser.

Find out more about Native Client’s sandboxes in **Native Client: A Sandbox for Portable, Untrusted x86 Native Code**.

References

Nacl Tutorial Read more **Google Developers Native Client Site - Tutorial.**

For further read please check **Google Developers Native Client Site**.