GameUpdate Library Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

1 Library Overview 3
Scope of This Document 3
Purpose and Features 3
Main Feature 3
Embedding into a Program 3
Sample Programs 3
Reference Materials 4
2 Using the Library 5
Loading Modules 5
Initializing the Network Library 5
Initializing the GameUpdate Library 5
Patch Confirmation Processing 5
Interruption Processing 6
Termination Processing 6
3 Precautions 7
libnet Initialization When Using the GameUpdate Library 7
Limitation When Using the Debugger of Visual Studio 7

1 Library Overview

Scope of This Document

This document provides an explanation of the GameUpdate library, which checks for patches for an applicable application. The library accesses the patch server to obtain and analyze the patch version file.

Purpose and Features

The GameUpdate library checks for patches for an applicable application. An application can use the GameUpdate library to check whether an updatable patch exists and obtain its application version (`APP_VER`).

The patch is downloaded and installed by the system software.

Main Feature

The main feature offered by the GameUpdate library is as follows.

- Accessing the patch server to obtain and analyze the patch version file

Embedding into a Program

The files required for using the GameUpdate library are as follows.

Filename	Description
libgameupdate.h	Header file
libSceGameUpdate_stub.a	Stub library file
libSceGameUpdate_stub_weak.a	Weak import stub library file

Include libgameupdate.h in the source program.

The GameUpdate library can be linked only using the PRX format. To use the GameUpdate library, statically link libSceGameUpdate_stub.a or libSceGameUpdate_stub_weak.a. The PRX module is stored in the storage managed by the system software, and it is loaded/unloaded by the libsysmodule API.

For details regarding the PRX format, refer to the "libsysmodule Overview" document.

libnet, libssl, and libhttp must be initialized to use the GameUpdate library. For information on how to use libnet, libssl, and libhttp, refer to "libnet Overview", "libssl Overview", and "libhttp Overview" documents respectively.

Sample Programs

The following program is provided as a GameUpdate library sample program for reference purposes.

sample_code/network/api_libgameupdate/

This sample shows the basic usage of the GameUpdate library.

The program can be launched from the Visual Studio debugger, but in that case, no inquiry will be sent to the update server. To check actual operation, use the package configuration file provided with the GameUpdate library sample to create an application package from the Package Generator and use it. Regarding the Package Generator, refer to the "Package Generator User's Guide" document.

For sample information, refer to the readme.

Reference Materials

Regarding the patch system of PlayStation®Vita, refer to the following document.

- Patch Overview

2 Using the Library

Loading Modules

Loading the PRX module and dependent modules are executed by the libsysmodule API. Specify `SCE_SYSMODULE_GAME_UPDATE` as the module ID.

Initializing the Network Library

Before using the GameUpdate library, libnet, libssl, and libhttp must be initialized. For instructions, refer to the "libnet Overview", "libssl Overview", and "libhttp Overview" documents respectively.

In addition, for libnet initialization, refer to the "libnet Initialization When Using the GameUpdate Library" section of the "3 Precautions" chapter.

Initializing the GameUpdate Library

After loading related modules of the GameUpdate library with libsysmodule and initializing libnet, libssl, and libhttp, call `sceGameUpdateInit()`. Specify NULL as the argument.

```
ret = sceGameUpdateInit(NULL);
if (ret < 0){
        // Error handling
}
```

Patch Confirmation Processing

Call `sceGameUpdateRun()` to execute patch confirmation processing. In the *`result`* argument, specify the `SceGameUpdateResult` structure for obtaining the result.

```
int                 ret;
SceGameUpdateResult result;

memset(&result, 0, sizeof(result));
result.size = sizeof(result);

ret = sceGameUpdateRun(&result);
if (ret < 0){
        // Error handling
}
```

`sceGameUpdateRun()` is a blocking function that performs the following processing in the calling thread.

- Connect to the patch server
- Obtain and analyze the patch version file

When the call of `sceGameUpdateRun()` succeeds (returns 0), whether or not an applicable patch exists for an application is returned in the *`patchExist`* member of the `SceGameUpdateResult` structure in the *`result`* argument. An applicable patch exists when `SCE_TRUE` is returned in *`patchExist`*. There is no applicable patch when `SCE_FALSE` is returned.

The check processing will also return that an applicable patch exists when the patch for an application requires new system software, and also when the patch is being downloaded or already downloaded but not applied.

The check processing will return that there is no applicable patch even if a patch exists on the patch server if an application is of the latest application version (`APP_VER`).

When a patch exists, the application version of the latest patch (`APP_VER`) is passed to the *`appVer`* member in the "*`xx.yy`*" format. For example, the application version 01.01 will be "`01.01`".

To prompt the user to perform an update to apply an available patch, display a system defined message of Message Dialog. Regarding the method for displaying a system defined message, refer to the "Message Dialog Overview" document.

Interruption Processing

The processing of `sceGameUpdateRun()` can be force-aborted with `sceGameUpdateAbort()`.

Force-aborted `sceGameUpdateRun()` will return `SCE_GAME_UPDATE_ERROR_ABORTED`.

```
ret = sceGameUpdateAbort();
if (ret < 0){
        // Error handling
}
```

Termination Processing

Use `sceGameUpdateTerm()` to terminate the GameUpdate library.

If this function is called while `sceGameUpdateRun()` is executing, interruption processing will be performed. At this time, `sceGameUpdateTerm()` will block until interruption processing finishes.

```
ret = sceGameUpdateTerm();
if (ret < 0){
        // Error handling
}
```

3 Precautions

libnet Initialization When Using the GameUpdate Library

From its initialization until termination, the GameUpdate library shares a section of the user space memory of libnet.

The size of memory that is consumed is directly related to the number of hybrid patch packages that is being distributed (approximately 1 KiB per hybrid patch package), specify a large size for memory upon initializing libnet when distributing many hybrid patch packages.

Limitation When Using the Debugger of Visual Studio

The GameUpdate library does not correctly run when using the debugger of Visual Studio. `sceGameUpdateRun()` will always return that there is no patch.