libLiveArea Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

1 Library Overview

Scope of This Document

libLiveArea is a library used for updating LiveArea™. This document describes the update feature of XML defined in LiveArea™ provided by libLiveArea for updating content information zone.

Purpose and Features

libLiveArea is a library used for updating LiveArea™. By using libLiveArea, an application can update content information zone of LiveArea™. For the overview and specifications of LiveArea™ and content information zone, refer to the "System Software Overview" and "LiveArea™ Specifications" documents.

Main Features

The following are the main features provided by libLiveArea.

- Feature for updating all the content information zone
- Feature for updating the content information zone partially in "**frame**" units
- Feature for obtaining revision and data for the user contents

Used Resources

libLiveArea uses the following system resources.

Resource	Description
Footprint	16 KiB when PRX is loaded
Work memory	Unused
Thread	Unused
Processor time	Can be ignored
IO (Storage)	IO occurs on the system side. It depends on the number of files in the specified directory and the file size.

Embedding into a Program

Include livearea.h in the source program. Various header files will be automatically included as well.

Load the PRX module in the program, as follows.

```
if ( sceSysmoduleLoadModule(SCE_SYSMODULE_LIVEAREA) != SCE_OK ) {
    // Error handling
}
```

Upon building the program, link libSceLiveArea_stub_*.a statically.

Sample Programs

The following file is provided as a sample program that use libLiveArea for reference purposes.

sample_code/system/api_livearea/basic/

This sample shows the basic usage of libLiveArea.

Reference Materials

For specifications of LiveArea™ and XML defined in LiveArea™, refer to the following documents.

- System Software Overview
- LiveArea™ Specifications

2 Usage Procedure

Main APIs Used in Basic Processing

API	Description
`sceLiveAreaReplaceAllSync()`	Requests for update of all the content information zone (blocking version)
`sceLiveAreaReplaceAllAsync()`	Requests for update of all the content information zone (non-blocking version)
`sceLiveAreaUpdateFrameSync()`	Requests update the content information zone partially in "**frame**" units (blocking version)
`sceLiveAreaUpdateFrameAsync()`	Requests update the content information zone partially in "**frame**" units (non-blocking version)
`sceLiveAreaGetStatus()`	Gets the state and the result of update processing of the content information zone
`sceLiveAreaGetRevision()`	Gets the "**content-rev**" attribute specified by "**livearea**" tag in the XML defined in LiveArea™
`sceLiveAreaGetFrameRevision()`	Gets the "**rev**" attribute specified by "**frame**" tag in the XML defined in LiveArea™
`sceLiveAreaGetFrameUserData()`	Gets the "**userdata**" attribute specified by "**frame**" tag in the XML defined in LiveArea™

Update All the Content Information Zone

Locate a directory that contains the XML defined in LiveArea™ under a directory such as app0: and savedata0: that can be accessed in advance.

Locating Example

app0:my_livearea/template.xml

app0:my_livearea/xxxx.png

app0:my_livearea/bg0.png

> **Note**
> For specifications of the XML defined in LiveArea™ and the XML description method for update, refer to the "LiveArea™ Specifications" document.

> **Note**
> For details on directories that can be accessed, refer to the "Directories That Can Be Specified with libLiveArea" section in the Chapter 3: Notes.

(1) Request for an update processing

Call `sceLiveAreaReplaceAllSync()` or `sceLiveAreaReplaceAllAsync()` with passing the directory path where the XML defined in LiveArea™ is placed.

Blocking version:

```
ret = sceLiveAreaReplaceAllSync("app0:my_livearea");
if (ret) {
        /* Error handling */
}
```

Non-blocking version:

```
ret = sceLiveAreaReplaceAllAsync("app0:my_livearea");
if (ret) {
        /* Error handling */
}
```

`sceLiveAreaReplaceAllSync()` will return after the update processing is completed.

`sceLiveAreaReplaceAllAsync()` will return immediately, and then the update processing is performed asynchronously on the system side. Subsequently, perform (2) "Obtain update processing state".

Note
`sceLiveAreaReplaceAllAsync()` also reserves shared resources required for an asynchronous processing performed on the system side. To release the resources, wait for the processing completion with `sceLiveAreaGetStatus()` described below or unload the PRX. Other libraries' operation will be affected if the shared resources remain unreleased. This release processing is not necessary for `sceLiveAreaReplaceAllSync()`.

(2) Obtain update processing state (for non-blocking version only)

The processing status and a processing result of `sceLiveAreaReplaceAllAsync()` can be obtained with `sceLiveAreaGetStatus()`. When this function is called in the state of update completion, `SCE_LIVEAREA_OK(0)` will return. Perform a retry if `SCE_LIVEAREA_BUSY(1)`, which indicates the update is in progress, is returned. An error (negative value) is returned if an error occurs during an update processing. This function returns immediately after being called.

```
ret = sceLiveAreaGetStatus();
if (ret == SCE_LIVEAREA_BUSY) {
        /* Update is in progress. Perform a retry at a certain interval */
} else if (ret == SCE_LIVEAREA_OK) {
        /* Processing performed when an update is successfully completed */
} else {
        /* Processing performed when an update is ended in failure */
}
```

Note
If you want to update/delete a file located in the specified directory path, wait for update completion (`SCE_LIVEAREA_OK`) before performing the processing. The system accesses the file in the directory during an update processing.

Note
Make sure to perform a retry at a certain interval to obtain the status so as to avoid busy loop (a state where a thread performing a retry continues to occupy a CPU, and consequently other threads cannot use the CPU for processing).

Partial Update of the Content Information Zone in "Frame" Units

Locate a directory that contains an image file under a directory such as app0: and savedata0: that can be accessed in advance.

Locating Example

app0:my_livearea_update/xxxx.png

app0:my_livearea_update/yyyy.png

Also, prepare the character string with the "**frame**" tag in the XML defined in LiveArea™ on the memory.

```
const char* frameXmlStr =
              "<frame id='frame1' rev='1' >"
                 "<liveitem><image>xxxx.png</image></liveitem>"
              "</frame>";
```

Note
For details on directories that can be accessed, refer to the "Directories That Can Be Specified with libLiveArea" section in the Chapter 3: Notes.

Note
In order to prepare character strings of "**frame**" tag in advance, describe the XML special symbols ('', '"', '&', '<', '>') by replacing those with escape sequences (''', '"', '&', '<', '>').
(Bad example) `<text><str>Love & Peace</str></text>`
(Good example) `<text><str>Love &amp; Peace</str></text>`

(1) Request for an update processing

Call `sceLiveAreaUpdateFrameSync()` or `sceLiveAreaUpdateFrameAsync()` with passing the path of the directory where the image files and character string with the LiveArea XML's "**frame**" tag are located.

Blocking version:

```
ret = sceLiveAreaUpdateFrameSync(SCE_LIVEAREA_FORMAT_VER_CURRENT,
                                 frameXmlStr,
                                 strlen(frameXmlStr),
                                 "app0:my_livearea_update",
                                 SCE_LIVEAREA_FLAG_NONE);
if (ret) {
        /* Error handling */
}
```

Non-blocking version:

```
ret = sceLiveAreaUpdateFrameAsync(SCE_LIVEAREA_FORMAT_VER_CURRENT,
                                  frameXmlStr,
                                  strlen(frameXmlStr),
                                  "app0:my_livearea_update",
                                  SCE_LIVEAREA_FLAG_NONE);
if (ret) {
        /* Error handling */
}
```

`sceLiveAreaUpdateFrameSync()` will return after the update processing is completed.

`sceLiveAreaUpdateFrameAsync()` will return immediately, and then the update processing is performed asynchronously on the system side. Subsequently, perform (2) "Obtain update processing state".

Note
`sceLiveAreaUpdateFrameAsync()` also reserves shared resources required for an asynchronous processing performed on the system side. To release the resources, wait for the processing completion with `sceLiveAreaGetStatus()` described below or unload the PRX. Other libraries' operation will be affected if the shared resources remain unreleased. This release processing is not necessary for `sceLiveAreaReplaceAllSync()`.

Note
When creating character strings of "**frame**" tag in a program, add escape processing to replace the XML's special symbols ("'", "\"", "&", "<", ">") with escape sequences ("'", """, "&", "<", ">"). The sample program describes an example for implementing the escape processing.

(2) Obtain update processing state (for non-blocking version only)

The processing status and a processing result of `sceLiveAreaUpdateFrameAsync()` can be obtained with `sceLiveAreaGetStatus()`. When this function is called in the state of update completion, `SCE_LIVEAREA_OK(0)` will return. Perform a retry if `SCE_LIVEAREA_BUSY(1)`, which indicates the update is in progress, is returned. An error (negative value) is returned if an error occurs during an update processing. This function returns immediately after being called.

```
ret = sceLiveAreaGetStatus();
if (ret == SCE_LIVEAREA_BUSY) {
        /* Update is in progress. Perform a retry at a certain interval */
} else if (ret == SCE_LIVEAREA_OK) {
        /* Processing performed when an update is successfully completed */
} else {
        /* Processing performed when an update is ended in failure */
}
```

Note
If you want to update/delete a file located in the specified directory path, wait for update completion (`SCE_LIVEAREA_OK`) before performing the processing. The system accesses the file in the directory during an update processing.

Note
Make sure to perform a retry at a certain interval to obtain the status so as to avoid busy loop (a state where a thread performing a retry continues to occupy a CPU, and consequently other threads cannot use the CPU for processing).

Obtain User Content Revision and Data

(1) Obtain the "content-rev" attribute specified by the "livearea" tag

It is possible to obtain the "content-rev" attribute specified with "livearea" tag in the XML defined in LiveArea™ with `sceLiveAreaGetRevision()`. If obtainment is successful, `SCE_LIVEAREA_OK(0)` will return. If obtainment fails, an error (negative value) will return.

```
SceInt64 contentRev;
ret = sceLiveAreaGetRevision(&contentRev);
if (ret) {
        /* Error handling */
}
```

(2) Obtain the "rev" attribute specified by the "frame" tag

The "rev" attribute specified with "frame" tag in the XML defined in LiveArea™ can be obtained by specifying the "id" attribute of "frame" in `sceLiveAreaGetFrameRevision()`. If obtainment is successful, `SCE_LIVEAREA_OK(0)` will return. If obtainment fails, an error (negative value) will return.

```
SceInt64 revision;
ret = sceLiveAreaGetFrameRevision("frame1", &revision);
if (ret) {
        /* Error handling */
}
```

(3) Obtain the "userdata" attribute specified by the "frame" tag

The "userdata" attribute specified with "frame" tag in the XML defined in LiveArea™ can be obtained by specifying the "id" attribute of "frame" in `sceLiveAreaGetFrameUserData()`. If obtainment is successful, `SCE_LIVEAREA_OK(0)` will return. If obtainment fails, an error (negative value) will return.

```
char userData[SCE_LIVEAREA_MAX_USER_DATA_LEN + 1];
ret = sceLiveAreaGetFrameUserData("frame1", userData, sizeof(userData));
if (ret) {
        /* Error handling */
}
```

3 Notes

Directories That Can Be Specified with libLiveArea

Directories under app0: and savedata0: can be specified, excluding those under directories that are reserved by the system and whose names begin with "sce_" (e.g. "sce_sys"). As an exception, it is possible to specify directories allowed by the system to place XML defined in LiveArea™ ("app0:sce_sys/livearea/contents" and "app0:sce_sys/retail/livearea/contents").

In addition, directories under host0: and ux0:data/ can also be specified during Development Mode.

Examples that can be specified

- app0:my_livearea1 (directory on the application side)
- app0:sce_sys/livearea/contents (directory that includes the initial XML defined in LiveArea™)
- app0:sce_sys/retail/livearea/contents
  (directory that includes XML defined in LiveArea™ of the Full state of upgradable contents)
- savedata0:my_livearea1 (save data directory on the application side)

Examples that can be specified only during Development Mode

- host0:xxx/yyy
- ux0:data/xxx

Examples that cannot be specified

- app0:sce_sys/xxx/yyy (directory for the system file)
- savedata0:sce_sys/xxx/yyy (directory for the system file)
- addcont0:xxx/yyy (directory on the mount point that is not allowed)

Precautions for Upgrade Processing to Full Version with Store Checkout Dialog Library

There are some points to note concerning the simultaneous use of LiveArea™ update processing with libLiveArea and upgrade processing to the full version with the Store Checkout Dialog library.

For details, refer to the precautions in the "LiveArea™ Specifications" document.

For the Store Checkout Dialog library, refer to the "Store Checkout Dialog Overview" document.

Notes

Using libLiveArea for changing LiveArea™ during application boot-up raises the system load and should be avoided. LiveArea™ during application boot-up can be changed by providing files in the app0:sce_sys/livearea/contents directory beforehand. For the app0: specifications, refer to the "Application Development Process Overview" document, and for the app0:sce_sys/livearea/contents file configuration, refer to the "LiveArea™ Specifications" document.

Limitations

When booting up from a debugger or APP_HOME, LiveArea™ update information is valid only during executing an application, and the information will be deleted after the application is terminated (returns to the initial XML defined in LiveArea™).