Luna Headers Reference Manual

Generated by Doxygen 1.5.5

Wed Sep 16 12:26:46 2009

Contents

Chapter 1

Main Page

LunaService

Example client usage:

```
bool retVal;
LSError lserror;
LSErrorInit(&lserror);

LSHandle *serviceHandle;
retVal = LSRegister(NULL, &serviceHandle, &lserror);
if (!retVal) goto error;

retVal = LSCall(serviceHandle, "luna://com.palm.contacts/category/listContacts",
        "{ \"json payload\" }", listContactsHandler, user_data, &token, &lserror);
if (!retVal) goto error;

LSGmainAttach(serviceHandle, gmainLoop, &lserror);
g_main_loop_run(gmainLoop);
```

Example service usage.

```
// callback
static bool
listContacts(LSHandle *sh, LSMessage *message)
{
     LSMessage *reply = NULL;

     bool retVal;
     LSError lserror;
     LSErrorInit(&lserror);

     retVal = LSMessageReply(sh, message, "{ JSON REPLY PAYLOAD }", &lserror);
     if (!retVal)
     {
         LSErrorPrint(&lserror, stderr);
         LSErrorFree(&lserror);
     }

     return retVal;
}

static LSMethod ipcMethods[] = {
   { "listContacts", listContacts },
   { },
```

```
};
...

// Service registration thread
bool retVal;
LSError lserror;
LSErrorInit(&lserror);

LSHandle *serviceHandle;
retVal = LSRegister("com.palm.contacts", &serviceHandle, &lserror);
if (!retVal) goto error;

retVal = LSRegisterCategory(serviceHandle, "/category",  ipcMethods, NULL, NULL, &lserror);
if (!retVal) goto error;

retVal = LSGmainAttach(serviceHandle, gmainLoop, &lserror);
if (!retVal) goto error;

g_main_loop_run(gmainLoop);
```

Storing a message for replying in another thread.

```
Queue messageQueue;
...

static bool
listContacts(LSHandle *sh, LSMessage *message)
{
     bool retVal;

     LSError lserror;
     LSErrorInit(&lserror);

     LSMessageRef(message);

     queue(messageQueue, message);
}

...

void
SomeOtherThread()
{
    LSError lserror;
    LSErrorInit(&lserror);

    LSMessage *message = dequeue(messageQueue);
    ...
    if (!LSMessageReply(sh, message, "{PAYLOAD IN JSON}", lserror))
    {
        LSErrorPrint(&lserror);
        LSErrorFree(&lserror);
    }

    ....
}
```

Example run loop via select. See LSCustomSelectExample() for latest example.

```
bool retVal;

do
{
    int nfd = -1;
    fd_set rdfds, wrfds, exfds;
```

```
    FD_ZERO(&rdfds);
    FD_ZERO(&wrfds);
    FD_ZERO(&exfds);

    retVal = LSCustomGetFds(sh, &nfd, &rdfds, &wrfds, &exfds, lserror);
    if (!retVal) return -1;

    int ret = select(nfd, &rdfds, &wrfds, &exfds, NULL);
    if (ret < 0)
    {
        perror("select");
        break;
    }

    // Pull incoming bytes off socket and push outgoing bytes onto it.
    retVal = LSCustomSendRecvBytes(sh, &rdfds, &wrfds, &exfds, lserror);
    if (!retVal)
    {
        break;
    }

    // Transmit byte and Dispatch incomming at most 1 message
    retVal = LSCustomDispatchMessage(sh, NULL, lserror);
    if (!retVal)
    {
        break;
    }

} while (true);
```

Example run loop via select if you want to handle messages directly.

```
while (serviceRunning)
{
    fd_set rdfds, wrfds, exfds;
    FD_ZERO(&rdfds);
    FD_ZERO(&wrfds);
    FD_ZERO(&exfds);

    LSGetFd(serviceHandle, &maxfd, &rdfds, &wrfds, &exfds, &lserror);

    ret = select(maxfd, &rdfds, &wrfds, &exfds, NULL);
    if (ret > 0)
    {
        LSMessage *message;
        char *reply = NULL;

        LSMessageFetch(serviceHandle, &message, &lserror);

        if (strcmp(LSMessageGetName(message), "listContacts"))
        {
            char *payload;
            payload = LSMessageGetPayload(message);
            ...
            reply = "{ JSON PAYLOAD }";
            LSMessageReply(serviceHandle, message, reply, &lserror);
        }
    }
}
```

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Module Documentation

4.1 LunaService

Classes

- struct LSError

  Error object which contains information about first error since it was initialized via LSErrorInit.

- struct **LSMethod**
- struct **LSSignal**
- struct **LSProperty**

Defines

- #define LSMESSAGE_TOKEN_INVALID 0

  Invalid token number.

Typedefs

- typedef unsigned long **LSMessageToken**
- typedef struct LSError **LSError**
- typedef struct LSHandle LSHandle

  Handle to service.

- typedef struct LSPalmService LSPalmService

  Handle to public service.

- typedef struct LSMessage LSMessage

  Message object.

- typedef bool(∗ LSMethodFunction)(LSHandle ∗sh, LSMessage ∗msg, void ∗category_context)

  Type for method callbacks.

- typedef bool(∗ LSPropertyGetFunction)(LSHandle ∗sh, LSMessage ∗msg, void ∗category_-context)

  Type for property get callback.

- typedef bool(∗ LSPropertySetFunction)(LSHandle ∗sh, LSMessage ∗msg, void ∗category_-context)

  Type for property set callback.

- typedef void(∗ **LSDisconnectHandler**)(LSHandle ∗sh, void ∗user_data)

Enumerations

- enum LSMethodFlags { **LUNA_METHOD_FLAG_DEPRECATED** = (1 << 0) }

  Method flags.

- enum LSSignalFlags { **LUNA_SIGNAL_FLAG_DEPRECATED** = (1 << 0) }

  Signal flags.

- enum LSPropertyFlags { **LUNA_PROPERTY_FLAG_DEPRECATED** = (1 << 0) }

  Property flags.

Functions

- bool **LSErrorInit** (LSError ∗error)
- void **LSErrorFree** (LSError ∗error)
- bool **LSErrorIsSet** (LSError ∗lserror)
- void **LSErrorPrint** (LSError ∗lserror, FILE ∗out)
- bool **LSRegister** (const char ∗name, LSHandle ∗∗sh, LSError ∗lserror)
- bool **LSRegisterPubPriv** (const char ∗name, LSHandle ∗∗sh, bool public_bus, LSError ∗lserror)
- bool **LSSetDisconnectHandler** (LSHandle ∗sh, LSDisconnectHandler disconnect_handler, void ∗user_data, LSError ∗lserror)
- bool **LSRegisterCategory** (LSHandle ∗sh, const char ∗category, LSMethod ∗methods, LSSignal ∗signals, LSProperty ∗properties, LSError ∗lserror)
- bool **LSRegisterCategoryAppend** (LSHandle ∗sh, const char ∗category, LSMethod ∗methods, LSSignal ∗signals, LSError ∗lserror)
- bool **LSCategorySetData** (LSHandle ∗sh, const char ∗category, void ∗user_data, LSError ∗lserror)
- bool **LSUnregister** (LSHandle ∗service, LSError ∗lserror)
- const char ∗ **LSHandleGetName** (LSHandle ∗sh)
- bool **LSRegisterPalmService** (const char ∗name, LSPalmService ∗∗ret_palm_service, LSError ∗lserror)
- bool **LSUnregisterPalmService** (LSPalmService ∗psh, LSError ∗lserror)
- bool **LSPalmServiceRegisterCategory** (LSPalmService ∗psh, const char ∗category, LSMethod ∗methods_public, LSMethod ∗methods_private, LSSignal ∗signals, void ∗category_user_data, LSError ∗lserror)
- LSHandle ∗ **LSPalmServiceGetPrivateConnection** (LSPalmService ∗psh)
- LSHandle ∗ **LSPalmServiceGetPublicConnection** (LSPalmService ∗psh)
- LSHandle ∗ **LSMessageGetConnection** (LSMessage ∗message)
- bool **LSMessageIsPublic** (LSPalmService ∗psh, LSMessage ∗message)
- void **LSMessageRef** (LSMessage ∗message)

- void **LSMessageUnref** (LSMessage *message)
- bool **LSMessagePrint** (LSMessage *lmsg, FILE *out)
- const char * **LSMessageGetUniqueToken** (LSMessage *message)
- const char * **LSMessageGetKind** (LSMessage *message)
- const char * **LSMessageGetApplicationID** (LSMessage *message)
- const char * **LSMessageGetSender** (LSMessage *message)
- const char * **LSMessageGetCategory** (LSMessage *message)
- const char * **LSMessageGetMethod** (LSMessage *message)
- const char * **LSMessageGetPayload** (LSMessage *message)
- json_t * **LSMessageGetPayloadJSON** (LSMessage *message)
- bool **LSMessageIsSubscription** (LSMessage *lsmgs)
- LSMessageToken **LSMessageGetToken** (LSMessage *call)
- LSMessageToken **LSMessageGetResponseToken** (LSMessage *reply)
- bool **LSMessageRespond** (LSMessage *message, const char *reply_payload, LSError *lserror)
- bool **LSMessageReply** (LSHandle *sh, LSMessage *lsmsg, const char *replyPayload, LSError *lserror)
- bool **LSMessageReturn** (LSHandle *sh, LSMessage *message, const char *replyPayload, LSError *error)
- bool **LSGmainAttach** (LSHandle *sh, GMainLoop *mainLoop, LSError *lserror)
- bool **LSGmainAttachPalmService** (LSPalmService *psh, GMainLoop *mainLoop, LSError *lserror)
- bool **LSGmainSetPriority** (LSHandle *sh, int priority, LSError *lserror)
- bool **LSGmainSetPriorityPalmService** (LSPalmService *psh, int priority, LSError *lserror)

4.1.1 Define Documentation

4.1.1.1 #define LSMESSAGE_TOKEN_INVALID 0

Invalid token number.

This is seen if you do LSMessageGetResponseToken() on a message that is not a reply. It is also a good neutral value to initialize an array of unitialized message tokens.

4.1.2 Typedef Documentation

4.1.2.1 typedef bool(* LSMethodFunction)(LSHandle *sh, LSMessage *msg, void *category_context)

Type for method callbacks.

Table registration of callbacks.

Parameters:

LSMethodFunction

sh

msg

Return values:

true if message successfully processed.

false if some error occurred and you would like the callback to be called again later.

4.1.2.2 typedef bool(* LSPropertyGetFunction)(LSHandle *sh, LSMessage *msg, void *category_context)

Type for property get callback.

Parameters:

LSPropertyGetFunction

sh

msg

Return values:

Same as LSMethodFunction()

4.1.2.3 typedef bool(* LSPropertySetFunction)(LSHandle *sh, LSMessage *msg, void *category_context)

Type for property set callback.

Parameters:

LSPropertySetFunction

sh

msg

Return values:

Same as LSMethodFunction()

4.2 LunaServiceClient

Typedefs

- typedef bool(∗ LSServerStatusFunc)(LSHandle ∗sh, const char ∗serviceName, bool connected, void ∗ctx)

  Function callback to be called when serviceName connects/disconnects.

- typedef bool(∗ LSFilterFunc)(LSHandle ∗sh, LSMessage ∗reply, void ∗ctx)

  Callback function called on incomming message.

Functions

- bool **LSCall** (LSHandle ∗sh, const char ∗uri, const char ∗payload, LSFilterFunc callback, void ∗user_data, LSMessageToken ∗ret_token, LSError ∗lserror)
- bool **LSCallOneReply** (LSHandle ∗sh, const char ∗uri, const char ∗payload, LSFilterFunc callback, void ∗ctx, LSMessageToken ∗ret_token, LSError ∗lserror)
- bool **LSCallFromApplication** (LSHandle ∗sh, const char ∗uri, const char ∗payload, const char ∗applicationID, LSFilterFunc callback, void ∗ctx, LSMessageToken ∗ret_token, LSError ∗lserror)
- bool **LSCallFromApplicationOneReply** (LSHandle ∗sh, const char ∗uri, const char ∗payload, const char ∗applicationID, LSFilterFunc callback, void ∗ctx, LSMessageToken ∗ret_token, LSError ∗lserror)
- bool **LSCallCancel** (LSHandle ∗sh, LSMessageToken token, LSError ∗lserror)

4.2.1 Typedef Documentation

4.2.1.1 typedef bool(∗ LSFilterFunc)(LSHandle ∗sh, LSMessage ∗reply, void ∗ctx)

Callback function called on incomming message.

Parameters:

sh service handle

reply reply message

void ∗ context

Return values:

true if message is handled.

4.2.1.2 typedef bool(∗ LSServerStatusFunc)(LSHandle ∗sh, const char ∗serviceName, bool connected, void ∗ctx)

Function callback to be called when serviceName connects/disconnects.

Parameters:

sh service handle

serviceName name of service that was brought up/down.

connected service was brought up if true.

Return values:

4.3 LunaServiceSubscription

Typedefs

- typedef struct LSSubscriptionIter **LSSubscriptionIter**

Functions

- bool **LSSubscriptionProcess** (LSHandle *sh, LSMessage *message, bool *subscribed, LSError *lserror)
- bool **LSSubscriptionSetCancelFunction** (LSHandle *sh, LSFilterFunc cancelFunction, void *ctx, LSError *lserror)
- bool **LSSubscriptionAdd** (LSHandle *sh, const char *key, LSMessage *message, LSError *lserror)

- bool **LSSubscriptionAcquire** (LSHandle *sh, const char *key, LSSubscriptionIter **ret_iter, LSError *lserror)
- void **LSSubscriptionRelease** (LSSubscriptionIter *iter)
- bool **LSSubscriptionHasNext** (LSSubscriptionIter *iter)
- LSMessage * **LSSubscriptionNext** (LSSubscriptionIter *iter)
- void **LSSubscriptionRemove** (LSSubscriptionIter *iter)
- bool **LSSubscriptionReply** (LSHandle *sh, const char *key, const char *payload, LSError *lserror)

- bool **LSSubscriptionRespond** (LSPalmService *psh, const char *key, const char *payload, LSError *lserror)
- bool **LSSubscriptionPost** (LSHandle *sh, const char *category, const char *method, const char *payload, LSError *lserror)

4.4 LunaServiceSignals

Functions

- bool **LSSignalSend** (LSHandle *sh, const char *uri, const char *payload, LSError *lserror)
- bool **LSSignalSendNoTypecheck** (LSHandle *sh, const char *uri, const char *payload, LSError *lserror)
- bool **LSSignalCall** (LSHandle *sh, const char *category, const char *methodName, LSFilterFunc filterFunc, void *ctx, LSMessageToken *ret_token, LSError *lserror)
- bool **LSSignalCallCancel** (LSHandle *sh, LSMessageToken token, LSError *lserror)
- bool **LSRegisterServerStatus** (LSHandle *sh, const char *serviceName, LSServerStatusFunc func, void *ctx, LSError *lserror)

Chapter 5

Class Documentation

5.1 LSError Struct Reference

Error object which contains information about first error since it was initialized via LSErrorInit.

`#include <lunaservice.h>`

Public Attributes

- int error_code
- char * message
- const char * file
- int line
- const char * func
- void * padding
- unsigned long magic

5.1.1 Detailed Description

Error object which contains information about first error since it was initialized via LSErrorInit.

5.1.2 Member Data Documentation

5.1.2.1 int LSError::error_code

public error code

5.1.2.2 char* LSError::message

public error message

5.1.2.3 const char* LSError::file

file in which error happened.

5.1.2.4 int LSError::line

line on which error happened.

5.1.2.5 const char* LSError::func

function on which error happened.

5.1.2.6 void* LSError::padding

Reserved for future use

5.1.2.7 unsigned long LSError::magic

use as cookie to detect invalid LSErrors

The documentation for this struct was generated from the following file:

- lunaservice.h

Index