LibSledLuaPlugin API Reference

This document provides information about the API elements that are used in the successful implementation of LibSledLuaPlugin.

Note:
The specifications contained in this document are subject to change without prior notice.

Version	Revision Date	Author(s)	Comments
1.0	30-July-08	Sage Knowles, Patrick O'Leary	1st release
2.0	August-2010	Risa Galant, Patrick O'Leary	Major updates for SLED 3.5
5.0.0	Apr-14	Gary Staas	Updates for 5.0.0. Update format.
5.1.0	Oct-14	Gary Staas	Updates for 5.1.0.
5.1.2	Feb-15	Gary Staas	Open source version.

Table of Contents

About the LibSledLuaPlugin API Reference **4**

Introduction **6**

Library Summary 7

Defines **8**

Define Summary 9

sce namespace **10**

sce 11

sce::Sled namespace **12**

sce::Sled 13

ChopCharsCallback 14

EditAndContinueCallback 15

EditAndContinueFinishCallback 16

UserDataCallback 17

UserDataFinishCallback 18

debuggerAddLuaPlugin 19

luaPluginCreate 20

luaPluginDebuggerBreak 21

luaPluginDebuggerBreak 22

luaPluginGetId 23

luaPluginGetName 24

luaPluginGetVarExcludeFlags 25

luaPluginGetVersion 26

luaPluginIsMemoryTracerRunning 27

luaPluginIsProfilerRunning 28

luaPluginMemoryTraceNotify 29

luaPluginRegisterLuaState 30

luaPluginRequiredMemory 31

luaPluginResetMemoryTrace 32

luaPluginResetProfileInfo 33

luaPluginSetVarExcludeFlags 34

luaPluginShutdown 35

luaPluginUnregisterLuaState 36

sce::Sled::VarExcludeFlags namespace **37**

sce::Sled::VarExcludeFlags 38

Enum 39

sce::Sled::LuaPlugin class **40**

sce::Sled::LuaPlugin 41

sce::Sled::LuaPluginConfig struct **42**

sce::Sled::LuaPluginConfig 43

LuaPluginConfig 44

About the LibSledLuaPlugin API Reference

This document provides information about the API elements that are used in implementing LibSledLuaPlugin. Use this document as a reference to the specifics of the API—such as its functions, data structures, type definitions, and defined symbols. This is a C API.
All of the information in this document is taken from comments in the header (`*.h`) files that are located in the `components\sce_sled\src\sledluaplugin` directory.

LuaPlugin Class

`LuaPlugin` is the class for the Lua plugin object that allows debugging Lua scripts during run time. To debug Lua scripts with SLED in an application, the application must create a `LuaPlugin` instance when it runs. The SLED GUI communicates with `LuaPlugin` instances.

What Is and Is Not Included in this Reference

This reference document contains information that is most useful for using components of the LibSledLuaPlugin API. Header files that contain information that is mostly for internal use, or platform-specific information that is defined elsewhere, are not included in this reference. If you are digging into the code in more depth, you can look in the header files for the additional information.
The following LibSledLuaPlugin header files in `components\sce_sled\src\sledluaplugin` are included in this reference document and contain the public API:

- `params.h`
- `sledluaplugin.h`

Most of the information from items tagged with the following identifiers is not included in this reference document:

- `private`
- `protected`

Look in the header files if you want more information about the `private` and `protected` class members, or any of the files that are not included in this reference.

Related Documentation

The following documentation, available on SHIP, contains useful information about using the SLED and Lua Toolset.

- *Getting Started with SLED*
- *SLED User's Guide*
- *SLED Plugin Guide*

Introduction

Library Summary

Library Contents

Item	Description
sce	sce namespace.
sce::Sled	SLED namespace.
sce::Sled::VarExcludeFlags	Namespace to scope variable exclude flags.
sce::Sled::LuaPlugin	Class describing a Lua plugin instance.
sce::Sled::LuaPluginConfig	LuaPlugin configuration parameters.

Defines

Define Summary

Define	Value	Description
SCE_LIBSLEDLUAPLUGIN_VER_MAJOR	5	Major version.
SCE_LIBSLEDLUAPLUGIN_VER_MINOR	1	Minor version.
SCE_LIBSLEDLUAPLUGIN_VER_REVISION	1	Revision version.
SCE_LIBSLEDLUAPLUGIN_VER_OTHER	0	Extra version number.
SCE_LIBSLEDLUAPLUGIN_ID	1	Id of LibSledLuaPlugin. The ID must be the same between this library and the `Sled.Lua.dll`.
SCE_LIBSLEDLUAPLUGIN_NAME	"SLED Lua Plugin"	Plugin name.

sce namespace

Summary

sce

sce namespace.

Definition

```
namespace sce {}
```

Description

Namespace for sce classes and functions.

Inner Classes, Structures, and Namespaces

Item	Description
sce::Sled	SLED namespace.

sce::Sled namespace

Summary

sce::Sled

SLED namespace.

Definition

```
namespace Sled {}
```

Description

Namespace for `Sled` classes and functions.

Function Summary

Function	Description
`debuggerAddLuaPlugin`	Add `LuaPlugin` to `SledDebugger`.
`luaPluginCreate`	Create `LuaPlugin` instance.
`luaPluginDebuggerBreak`	Force breakpoint on specific `lua_State` and send data via TTY.
`luaPluginDebuggerBreak`	Force breakpoint on next `lua_State` that runs an instruction and send data via TTY.
`luaPluginGetId`	Get ID of plugin.
`luaPluginGetName`	Get name of plugin.
`luaPluginGetVarExcludeFlags`	Get current variable exclude flags.
`luaPluginGetVersion`	Get plugin version information.
`luaPluginIsMemoryTracerRunning`	Check whether memory tracer is running.
`luaPluginIsProfilerRunning`	Check whether profiler is running.
`luaPluginMemoryTraceNotify`	Provide way to report Lua allocations to library for tracking with memory tracer.
`luaPluginRegisterLuaState`	Register `lua_State*` with library.
`luaPluginRequiredMemory`	Calculate size in bytes required for `LuaPlugin` instance based on configuration structure.
`luaPluginResetMemoryTrace`	Reset internal memory trace list.
`luaPluginResetProfileInfo`	Reset internal profile information list.
`luaPluginSetVarExcludeFlags`	Set variable exclude flags.
`luaPluginShutdown`	Shut down `LuaPlugin` instance.
`luaPluginUnregisterLuaState`	Unregister `lua_State*` from library.

Inner Classes, Structures, and Namespaces

Item	Description
sce::Sled::LuaPlugin	Class describing a Lua plugin instance.
sce::Sled::LuaPluginConfig	`LuaPlugin` configuration parameters.
sce::Sled::VarExcludeFlags	Namespace to scope variable exclude flags.

Type Definitions

ChopCharsCallback

Typedef for chop characters callback function.

Definition

```
#include <params.h>
namespace sce {
    namespace Sled {
        typedef const char * (*ChopCharsCallback)(
            const char *pszFilePath
        );
    }
}
```

Arguments

pszFilePath	Path to file

Return Values

None

Description

Typedef for a chop characters callback function.

EditAndContinueCallback

Typedef for an edit and continue callback function.

Definition

```
#include <params.h>
namespace sce {
    namespace Sled {
        typedef const char * (*EditAndContinueCallback)(
            const char *pszFilePath,
            void *pUserData
        );
    }
}
```

Arguments

pszFilePath	The relative path of the file that needs to be reloaded
pUserData	Optional user-controlled userdata

Return Values

A `const char*` to the contents of the reloaded file

Description

Typedef for an edit and continue callback function. This function alerts the user that they need to load a specific file synchronously, because LibSledLuaPlugin needs to operate on the contents. `EditAndContinueCallback()` returns a `const char*` to the contents of that file to LibSledLuaPlugin for processing.

See Also

`EditAndContinueFinishCallback`

EditAndContinueFinishCallback

Typedef for edit and continue callback function.

Definition

```
#include <params.h>
namespace sce {
    namespace Sled {
        typedef void (*EditAndContinueFinishCallback)(
            const char *pszFilePath,
            void *pUserData
        );
    }
}
```

Arguments

pszFilePath	The relative path of the file that needs to be reloaded
pUserData	Optional user-controlled userdata

Return Values

None

Description

Typedef for an edit and continue callback function. This function synchronously loads the file. Call this function after regular edit and continue callback processing has finished so that memory can be freed, if necessary.

See Also

EditAndContinueCallback

UserDataCallback

Typedef for user data callback function.

Definition

```
#include <params.h>
namespace sce {
    namespace Sled {
        typedef const char * (*UserDataCallback)(
            void *pLuaUserData,
            void *pUserData
        );
    }
}
```

Arguments

pLuaUserData	Lua userdata
pUserData	Optional user-controlled userdata

Return Values

None

Description

String describing the Lua user data that is sent back to SLED. This string shows up in the "Value" column of the respective [Globals/Locals/Upvalues/etc.] window.

UserDataFinishCallback

Typedef for user data callback function.

Definition

```
#include <params.h>
namespace sce {
    namespace Sled {
        typedef void (*UserDataFinishCallback)(
            void *pLuaUserData,
            void *pUserData
        );
    }
}
```

Arguments

pLuaUserData	Lua user data
pUserData	Optional user-controlled userdata

Return Values

None

Description

Typedef for a user data callback function. Call this function after regular user data callback processing has finished so that memory can be freed, if necessary.

Functions

debuggerAddLuaPlugin

Add `LuaPlugin` to `SledDebugger`.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t debuggerAddLuaPlugin(
            SledDebugger *debugger,
            LuaPlugin *plugin
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

debugger	`SledDebugger` to use
plugin	`LuaPlugin` to use

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null debugger or plugin

Description

Add a `LuaPlugin` to the `SledDebugger`. It's a helper method, because `LuaPlugin` is an incomplete type and the `SledDebugger` `debuggerAddPlugin()` method is expecting a `SledDebuggerPlugin` instance.

See Also

`luaPluginCreate`

luaPluginCreate

Create LuaPlugin instance.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginCreate(
            const LuaPluginConfig *config,
            void *location,
            LuaPlugin **outLuaPlugin
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

config	A configuration structure that details the settings to use
location	The location in memory in which to place the LuaPlugin instance. It must be as big as the value returned by luaPluginRequiredMemory().
outLuaPlugin	The LuaPlugin instance that is created

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null configuration structure
SCE_SLED_ERROR_INVALIDCONFIGURATION	Invalid value in the configuration structure

Description

Create a LuaPlugin instance.

See Also

luaPluginRequiredMemory, luaPluginShutdown, debuggerAddLuaPlugin

luaPluginDebuggerBreak

Force breakpoint on specific `lua_State` and send data via TTY.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginDebuggerBreak(
            LuaPlugin *plugin,
            lua_State *luaState,
            const char *pszText
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin	LuaPlugin to use
luaState	Pointer to a lua_State
pszText	Data to send via TTY

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin

Description

Force a breakpoint on a specific `lua_State` and send data via TTY.

luaPluginDebuggerBreak

Force breakpoint on next `lua_State` that runs an instruction and send data via TTY.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginDebuggerBreak(
            LuaPlugin *plugin,
            const char *pszText
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin	LuaPlugin to use
pszText	Data to send via TTY

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin

Description

Force a breakpoint on the next `lua_State` that runs an instruction and send data via TTY.

See Also

luaPluginGetVarExcludeFlags, luaPluginGetVarExcludeFlags

luaPluginGetId

Get ID of plugin.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginGetId(
            const LuaPlugin *plugin,
            uint16_t *outId
        );
    }
}
```

Calling Conditions

Multithread safe.

Arguments

plugin	LuaPlugin to use
outId	A number greater than zero

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin or outId

Description

Get ID of the plugin. The ID must be unique across all other language plugins. The ID 0 (zero) is reserved for the `SledDebugger` class.

See Also

luaPluginGetName, luaPluginGetVersion

luaPluginGetName

Get name of plugin.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginGetName(
            const LuaPlugin *plugin,
            const char **outName
        );
    }
}
```

Calling Conditions

Multithread safe.

Arguments

plugin	LuaPlugin to use
outName	The name of the plugin

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin or outName

Description

Get the name of the plugin.

See Also

luaPluginGetId, luaPluginGetVersion

luaPluginGetVarExcludeFlags

Get current variable exclude flags.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginGetVarExcludeFlags(
            const LuaPlugin *plugin,
            int32_t *outFlags
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin	LuaPlugin to use
outFlags	Current variable exclude flags

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin or outFlags

Description

Get the current variable exclude flags.

See Also

luaPluginSetVarExcludeFlags, luaPluginDebuggerBreak

luaPluginGetVersion

Get plugin version information.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginGetVersion(
            const LuaPlugin *plugin,
            Version *outVersion
        );
    }
}
```

Calling Conditions

Multithread safe.

Arguments

plugin	LuaPlugin to use
outVersion	The plugin's version information

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin or outVersion

Description

Get the plugin version information.

See Also

luaPluginGetId, luaPluginGetName

luaPluginIsMemoryTracerRunning

Check whether memory tracer is running.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginIsMemoryTracerRunning(
            const LuaPlugin *plugin,
            bool *outResult
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin	LuaPlugin to use
outResult	True if the memory tracer is running; false if it is not running

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin or outResult

Description

Check whether or not the memory tracer is running.

See Also

luaPluginIsProfilerRunning, luaPluginResetMemoryTrace, luaPluginMemoryTraceNotify

luaPluginIsProfilerRunning

Check whether profiler is running.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginIsProfilerRunning(
            const LuaPlugin *plugin,
            bool *outResult
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin	LuaPlugin to use
outResult	True if the profiler is running; false if it is not running

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin or outResult

Description

Determine whether or not the profiler is running.

See Also

luaPluginIsMemoryTracerRunning, luaPluginResetProfileInfo

luaPluginMemoryTraceNotify

Provide way to report Lua allocations to library for tracking with memory tracer.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginMemoryTraceNotify(
            LuaPlugin *plugin,
            void *userData,
            void *oldPtr,
            void *newPtr,
            std::size_t oldSize,
            std::size_t newSize,
            bool *outResult
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin	LuaPlugin to use
userData	A pointer to user data (not currently used for anything)
oldPtr	A pointer to old memory being deallocated
newPtr	A pointer to new memory being allocated
oldSize	Old memory size
newSize	New memory size
outResult	True if information is logged; false if memory tracer is not running

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin or outResult

Description

Provide a way to report Lua allocations to the library for tracking using the memory tracer. Call this function from the allocator that is making all the Lua allocations, deallocations, and reallocations.

See Also

luaPluginIsMemoryTracerRunning, luaPluginResetMemoryTrace

luaPluginRegisterLuaState

Register `lua_State*` with library.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginRegisterLuaState(
            LuaPlugin *plugin,
            lua_State *luaState,
            const char *luaStateName = 0
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin	LuaPlugin to use
luaState	Pointer to a lua_State
luaStateName	A name for the lua_State. This name shows up on the lua_State GUI in SLED to help easily identify the lua_State.

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin
SCE_SLED_LUA_ERROR_NODEBUGGERINSTANCE	No debugger instance (not added to a SledDebugger instance)
SCE_SLED_LUA_ERROR_INVALIDLUASTATE	lua_State is null
SCE_SLED_LUA_ERROR_DUPLICATELUASTATE	lua_State is already registered
SCE_SLED_LUA_ERROR_OVERLUASTATELIMIT	No space for lua_State

Description

Register a `lua_State*` with the library. Lua states must be registered with the library if they are to be debugged.

See Also

luaPluginUnregisterLuaState

luaPluginRequiredMemory

Calculate size in bytes required for LuaPlugin instance based on configuration structure.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginRequiredMemory(
            const LuaPluginConfig *config,
            std::size_t *outRequiredMemory
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

config	The configuration structure that details the settings to use
outRequiredMemory	The amount of memory that is needed for the LuaPlugin instance

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null configuration structure
SCE_SLED_ERROR_INVALIDCONFIGURATION	Invalid value in the configuration structure

Description

Calculate the size in bytes required for a LuaPlugin instance based on a configuration structure.

See Also

luaPluginCreate

luaPluginResetMemoryTrace

Reset internal memory trace list.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginResetMemoryTrace(
            LuaPlugin *plugin
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin LuaPlugin to use

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin

Description

Reset the internal memory trace list.

See Also

luaPluginIsMemoryTracerRunning, luaPluginResetProfileInfo, luaPluginMemoryTraceNotify

luaPluginResetProfileInfo

Reset internal profile information list.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginResetProfileInfo(
            LuaPlugin *plugin
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin LuaPlugin to use

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin

Description

Reset the internal profile information list.

See Also

luaPluginIsProfilerRunning, luaPluginResetMemoryTrace

luaPluginSetVarExcludeFlags

Set variable exclude flags.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginSetVarExcludeFlags(
            LuaPlugin *plugin,
            int32_t flags
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin	LuaPlugin to use
flags	An OR'd set of variable groups to exclude from processing and sending when hitting a breakpoint

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin

Description

Set the variable groups to exclude from processing and sending when hitting a breakpoint.

See Also

luaPluginGetVarExcludeFlags, luaPluginDebuggerBreak

luaPluginShutdown

Shut down LuaPlugin instance.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginShutdown(
            LuaPlugin *plugin
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin The LuaPlugin instance to shut down

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin

Description

Shut down a LuaPlugin instance.

See Also

luaPluginCreate

luaPluginUnregisterLuaState

Unregister `lua_State*` from library.

Definition

```
#include <sledluaplugin.h>
namespace sce {
    namespace Sled {
        int32_t luaPluginUnregisterLuaState(
            LuaPlugin *plugin,
            lua_State *luaState
        );
    }
}
```

Calling Conditions

Not multithread safe.

Arguments

plugin	LuaPlugin to use
luaState	Pointer to a lua_State

Return Values

Value	Description
SCE_SLED_ERROR_OK	Success
SCE_SLED_ERROR_NULLPARAMETER	Null plugin
SCE_SLED_LUA_ERROR_NODEBUGGERINSTANCE	No debugger instance (not added to a SledDebugger instance)
SCE_SLED_LUA_ERROR_INVALIDLUASTATE	lua_State is null
SCE_SLED_LUA_ERROR_LUASTATENOTFOUND	lua_State is not registered

Description

Unregister a `lua_State*` from the library. Lua states must be registered with the library if they are to be debugged.

See Also

luaPluginRegisterLuaState

sce::Sled::VarExcludeFlags namespace

Summary

sce::Sled::VarExcludeFlags

Namespace to scope variable exclude flags.

Definition

```
namespace VarExcludeFlags {}
```

Description

Namespace to scope variable exclude flags. Variable exclude flags exclude certain variable groups from being processed and sent to SLED when execution stops on a breakpoint.

Enumerated Types

Enum

Variable exclude flags for execution stops on a breakpoint.

Definition

```
#include <params.h>
namespace sce {
    namespace Sled {
        namespace VarExcludeFlags {
            enum Enum {
                kNone = 0,
                kGlobals = (1 << 1),
                kLocals = (1 << 2),
                kUpvalues = (1 << 3),
                kEnvironment = (1 << 4)
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kNone	0	Do not exclude any variable groups. This is the default behavior.
kGlobals	(1 << 1)	Exclude processing and sending global variables.
kLocals	(1 << 2)	Exclude processing and sending local variables.
kUpvalues	(1 << 3)	Exclude processing and sending upvalue variables.
kEnvironment	(1 << 4)	Exclude processing and sending environment table variables.

Description

Variable exclude flags exclude certain variable groups from being processed and sent to SLED when execution stops on a breakpoint.

sce::Sled::LuaPlugin class

Summary

sce::Sled::LuaPlugin

Class describing a Lua plugin instance.

Definition

```
#include <sledluaplugin_class.h>
class LuaPlugin {};
```

Description

Widely used class encapsulating the internals of a Lua plugin instance. Instantiate a `LuaPlugin` from a `LuaPluginConfig`.

This class is closed, and its internal data is not accessible.

The following are the main functions handling `SledDebugger`:

`sce::Sled::luaPluginCreate()`: Create `LuaPlugin` instance.

`sce::Sled::luaPluginRequiredMemory()`: Calculate size in bytes required for `LuaPlugin` instance based on configuration structure.

`sce::Sled::luaPluginShutdown()`: Shut down `LuaPlugin` instance.

`sce::Sled::luaPluginGetId()`: Get ID of plugin.

`sce::Sled::luaPluginRegisterLuaState()`: Register `lua_State*` with library.

`sce::Sled::luaPluginUnregisterLuaState()`: Unregister `lua_State*` from library.

`sce::Sled::luaPluginIsProfilerRunning()`: Check whether profiler is running.

`sce::Sled::luaPluginIsMemoryTracerRunning()`: Check whether memory tracer is running.

`sce::Sled::luaPluginDebuggerBreak()`: Force breakpoint on specific `lua_State` and send data via TTY.

`sce::Sled::luaPluginMemoryTraceNotify()`: Provide way to report Lua allocations to library for tracking with memory tracer.

`sce::Sled::debuggerAddLuaPlugin()`: Add `LuaPlugin` to `SledDebugger`.

For the full list of `LuaPlugin` functions, see `sce::Sled`.

sce::Sled::LuaPluginConfig struct

Summary

sce::Sled::LuaPluginConfig

LuaPlugin configuration parameters.

Definition

```
#include <params.h>
struct LuaPluginConfig {};
```

Description

Configuration parameters for LuaPlugin.

Fields

Public Instance Fields

uint16_t *maxBreakpoints*	Maximum number of breakpoints.
uint16_t *maxEditAndContinueEntryLen*	Maximum length of an edit and continue entry.
uint16_t *maxEditAndContinues*	Maximum number of scripts that can be modified when stopped on a breakpoint while debugging.
uint16_t *maxLuaStateNameLen*	Maximum length of a Lua state name.
uint16_t *maxLuaStates*	Maximum number of Lua states that can be debugged.
uint32_t *maxMemTraces*	Maximum number of memory traces to hold in memory.
uint16_t *maxNumVarFilters*	Maximum number of variable filters.
uint16_t *maxPatternsPerVarFilter*	Maximum number of patterns per filter.
uint16_t *maxProfileCallStackDepth*	Maximum call stack depth.
uint16_t *maxProfileFunctions*	Maximum number of functions to profile.
uint32_t *maxSendBufferSize*	Maximum size, in bytes, of the send buffer (1024 recommended at a minimum)
uint16_t *maxVarFilterPatternLen*	Maximum length of a single pattern in a filter.
uint32_t *maxWorkBufferSize*	Maximum size of the work buffer (1024 recommended at a minimum)
int32_t *numPathChopChars*	The number of characters to strip off the beginning of a path string.
void **pEditAndContinueUserData*	Edit and continue callback userdata.
ChopCharsCallback *pfnChopCharsCallback*	Modified path to compare breakpoint against.
EditAndContinueCallback *pfnEditAndContinueCallback*	Edit and continue callback function.
EditAndContinueFinishCallback *pfnEditAndContinueFinishCallback*	Edit and continue finish callback function.

Methods Summary

Methods	Description
LuaPluginConfig	LuaPluginConfig constructor.

Constructors and Destructors

LuaPluginConfig

`LuaPluginConfig` constructor.

Definition

```
#include <params.h>
inline LuaPluginConfig();
```

Return Values

None

Description

Constructor to initialize items.