

Getting Started with SLED and Lua

© 2015 Sony Computer Entertainment America LLC.
All Rights Reserved.

This document provides important information about getting started with the Tools & Technology ("TNT") SLED and Lua package.

Version	Revision Date	Author(s)	Comments
0.5	25-Oct-07	Patrick J. OLeary	"SLED – SCEA Lua Editor & Debugger: A Guide to Using SLED and LibLuaDebugger."
0.7	26-Oct-07	David T. Friedman	Rewrote, reorganized, and supplemented material from Version 0.5 to create this document.
0.7	27-Aug-08	Patrick J. OLeary	Minor edits as libluadebugger has been replaced by libscriptdebugger & libluaplugin.
2.0	31-Oct-08	DT Friedman	Major revision, bringing the document current with 2.0 release of SLED and Lua. Template and layout also updated.
3.1.0	30-Sept-09	Risa Galant	Minor updates: library and directory name changes, screen changes.
3.2.0	18-Dec-09	Risa Galant	Added Chapter 6: Using SLED as Your Game Debugger.
3.3.0	16-Feb-10	Risa Galant	Update for version number, copyright, filename.
3.5.0	Aug-10	Risa Galant	Major revision; bringing the document current with 3.5 release of SLED and Lua, wws re-org and re-naming.
5.0.0	Apr-14	Gary Staas	Update format. Update installation procedure for Package Manager; remove obsolete material on third party installs. Update file paths. Update UI figures.
5.1.0	Oct-14	Gary Staas	Update for 5.1.0.
5.1.2	Feb-15	Gary Staas	Open source version.

Material contained in this document may not be copied, reproduced, reduced to any electronic medium or machine readable form or otherwise duplicated and the information herein may not be used, disseminated or otherwise disclosed, except with the prior written consent of an authorized representative of Sony Computer Entertainment America, LLC.

Sony is a registered trademark of Sony Corporation.

All other trademarks are the properties of their respective owners.

See the accompanying License.txt for instructions on how you may use this copyrighted material.

Table of Contents

1 Introduction	4
Document Overview	4
Prerequisite Knowledge and Access	4
Knowledge	4
Access	4
2 SLED and Lua Files and Dependencies	5
3 Obtaining and Building SLED	6
Obtaining SLED	6
Obtaining Dependencies	6
Building SLED	7
Starting SLED	7
4 Building the Runtime Libraries	8
Building Libraries from Source Files	8
5 Confirming Successful Building and Setup	9
Running TestTarget	9
Windows	9
6 Example: Using SLED as Your Game Debugger	12
Requirements	12
Creating a SLED Project	12
Connect to Your Game and Go	17

1 Introduction

The SLED and Lua package allows users to develop, edit, and debug Lua scripts. At the heart of the package is the Script Language Editor and Debugger, or SLED. SLED is an IDE, similar in look and feel to Microsoft Visual Studio.

This package also encompasses several libraries, described in [SLED and Lua Files and Dependencies](#).

This document explains how to install and configure SLED, set up the rest of the SLED and Lua package, test the whole package using a small application provided for that purpose, and create a project to use SLED to debug Lua scripts.

Document Overview

In addition to this introduction, this guide comprises the following chapters.

- [SLED and Lua Files and Dependencies](#): lists the files that constitute the SLED and Lua package.
- [Obtaining and Building SLED](#): explains how to obtain and configure SLED.
- [Building the Runtime Libraries](#): explains how to set up the rest of the SLED and Lua package.
- [Confirming Successful Building and Setup](#): explains how to use the included test application to verify successful installation of the entire SLED and Lua package.
- [Example: Using SLED as Your Game Debugger](#): shows how to set up SLED as the debugger for your game's Lua scripts.

Prerequisite Knowledge and Access

To make the best use of this and other SLED documents, you should have the following knowledge and access.

Knowledge

For all targets:

- How to build an executable file for the target platform, using a Microsoft Visual Studio solution (.sln) file.

Access

For all targets:

- The home page for [SLED](#).

2 SLED and Lua Files and Dependencies

The SLED component package contains all the SLED and Lua files and dependencies necessary to build and run SLED. For your reference, the following table lists and describes the SLED and Lua files and dependencies that are part of the SLED and Lua distribution.

Download Package	Description
Scripting Language Editor & Debugger (SLED)	A full-featured IDE for editing and run-time debugging of scripts.
Lua 5.x libraries	Run-time interpreters for Lua scripts, enabling Lua script execution.
LibSledDebugger	The library that allows SLED to communicate with the target. For more information, see <i>SLED Plugin Guide</i> and <i>LibSledDebugger API Reference</i> .
LibSledLuaPlugin	The library that hooks Lua debug events, permitting SLED to be used as a Lua language debugger. For more information about SLED plugins, see <i>SLED Plugin Guide</i> .

The following table lists and describes SLED system key project dependencies.

Dependency	Description
wws_lua	Version 5.1.4 of the Lua scripting language, up-to-date with all official patches.
unittest-cpp	A lightweight and flexible unit-testing framework for C++.
wws_atf	Authoring Tools Framework (ATF), used to build SLED and its plugins.

3 Obtaining and Building SLED

The SLED package contains the source code and files to build SLED. If you install the contents of the distribution together at a common location, you should be able to build everything without issues when you need to.

Obtaining SLED

SLED is an open source project on GitHub. The [SLED GitHub repository](#) contains the latest public release. Create a placeholder directory for the release, such as `C:\SledOSS`. You can either clone the project to your desktop or download a ZIP file from this SLED GitHub page. After getting the SLED distribution, place its `sce_sled` folder in `C:\SledOSS`.

After you obtain the SLED distribution, the `c:\SledOSS\sce_sled` folder contains several files and subfolders:

- Visual Studio solution files:
 - `runtime_vs[VS version].sln`: Solution to build SLED runtime, including the LibSledDebugger, and LibSledLuaPlugin plugin runtime libraries, as in `runtime_vs2010.sln`. It also includes projects for testing these libraries.
 - `tool_vs[VS version].sln`: Solution to build SLED and its plugins, as in `tool_vs2013.sln`.
- `doc`: SLED documentation.
- `src`: Sources for the SLED runtime, used by `runtime_vs[VS version].sln`.
- `tool`: Sources for SLED and its plugins, used by `tool_vs[VS version].sln`.
- `wws_lua`: Lua distribution.

Obtaining Dependencies

To build SLED, you also need:

- [Authoring Tools Framework \(ATF\)](#)
- [unittest-cpp, a lightweight unit testing framework for C++](#)

You can obtain these items from GitHub at the indicated links. Download them and place them in the `SledOSS` folder, so that the `wws_atf` and `unittest-cpp` folders are in `SledOSS`.

After you have downloaded SLED and added these dependencies, your `c:\SledOSS` folder should look like this:

```
c:\SledOSS
  sce_sled (SLED distribution)
  unittest-cpp (dependency)
  wws_atf (dependency)
```

Building SLED

Build both the `runtime_vs[VS version].sln` and `tool_vs[VS version].sln` solutions in `c:\SledOSS\sce_sled` with Visual Studio, using the same configuration. The executables are placed in the `c:\SledOSS\sce_sled\bin` folder. After building everything, `c:\SledOSS\sce_sled` looks like this (all files are not listed):

```
c:\SledOSS\sce_sled
  bin (built files)
    sce_sled (built files for SLED)
      runtime (folder with runtime folders for various configurations)
        win32_static_dcrt_vc100_release (static release folder)
          libsce_testtarget_5.1.4_sample.spf (test project file)
          libsce_testtarget-5.1.4_sample.exe (test app)
        ...
      SLED.vs[VS version] (SLED executable and related files)
        Plugins (folder with DLLs for SLED)
        Resources (folder with resources, such as skins)
        Sled.exe (SLED executable)
  lib (folder for built libraries for various configurations)
    anycpu_dotnet_clr4_debug (folder of library files for this config)
    ...
    win32_static_dcrt_vc100_release (folder of library files for this config)
  tmp (folder for temporary build files)
```

Starting SLED

The SLED executable lives in `c:\SledOSS\sce_sled\bin\sce_sled\SLED.vs[VS version]`. Folder names in `bin\sce_sled` describe the version of Visual Studio used to build the item, as in `SLED.vs2010`.

To verify the installation was successful, start SLED:

1. Navigate to the `c:\SledOSS\sce_sled\bin\sce_sled\SLED.vs[VS version]` folder, where `[VS version]` is the version of Visual Studio used to build SLED.
2. Double-click the `SLED.exe` application. Verify that SLED runs.

4 Building the Runtime Libraries

The SLED libraries are:

- `libsce_sleddebugger.lib`: Runtime LibSledDebugger library.
- `libsce_sledluaplugin-5.1.4.lib`: SLED-side LibSledLuaPlugin library for Lua 5.1.4.
- `liblua-5.1.4.lib`: SLED-side Lua 5.1.4 library.

The SLED runtime libraries are built with the `runtime_vs[VS version].sln` solution in the `c:\SledOSS\sce_sled` folder.

Building Libraries from Source Files

The SLED and Lua package supports building with Microsoft Visual Studio (version 2010 or 2013).

Library source files, header files, and Visual Studio project files reside in `c:\SledOSS\sce_sled\src` in its `sleddebugger` and `sledluaplugin` folders.

If you already opened `runtime_vs[VS version].sln` and built all the projects as described in [Building SLED](#), these libraries are already built for you.

To build the libraries, open the `runtime_vs[VS version].sln` solution in Visual Studio. Build the following projects in the solution:

- `libsce_sleddebugger`
- `libsce_sledluaplugin-5.1.4`
- `lua-5.1.4`

5 Confirming Successful Building and Setup

The SLED and Lua package includes a sample application called `TestTarget` that you can use to confirm successful installation and configuration of the package. The `runtime_vs[VS version].sln` solution's `libsce_testtarget-5.1.4_sample` project builds `TestTarget`. If you built the entire solution as discussed in [Building SLED](#), you've already built `TestTarget`, so you can run it.

Running TestTarget

This section explains how to run `TestTarget`, open the `TestTarget` sample project in SLED, and connect to the `TestTarget` executable.

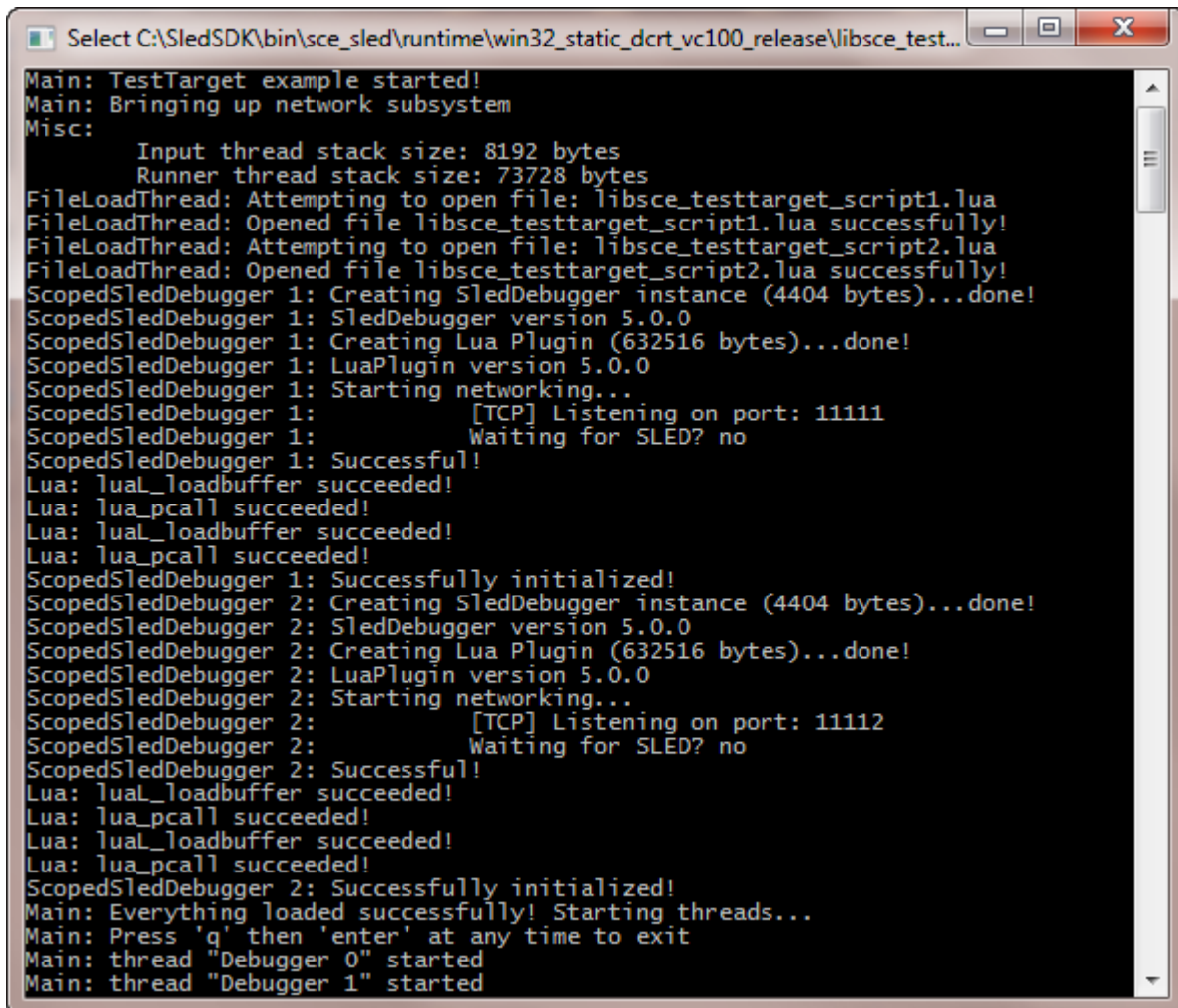
After building, `TestTarget` and related files are in `c:\SledOSS\sce_sled\bin\sce_sled\runtime\[configuration]` for the various supported configurations.

Windows

1. Start SLED, as show in [Starting SLED](#).
2. From the SLED menu, select **Target > Manage Targets**. The **Targets** dialog box opens.
3. Check the box for the target that you set as localhost in Setting up Targets. (If it is already checked, leave it unchanged.)
4. Click **OK**.
5. Open the `TestTarget` project in SLED if it isn't already open: **File > Project > Open**. Navigate to `c:\SledOSS\sce_sled\bin\sce_sled\runtime\[configuration]\libsce_testtarget_5.1.4_sample.spf` and select it.
6. In Windows Explorer, navigate to the appropriate `c:\SledOSS\sce_sled\bin\sce_sled\runtime\[configuration]`, where `[configuration]` is the configuration of Visual Studio used to build `TestTarget`. Double-click `libsce_testtarget-5.1.4_sample.exe` to run it. A screen similar to the following appears.

Figure 3

TestTarget Execution Screen in Windows

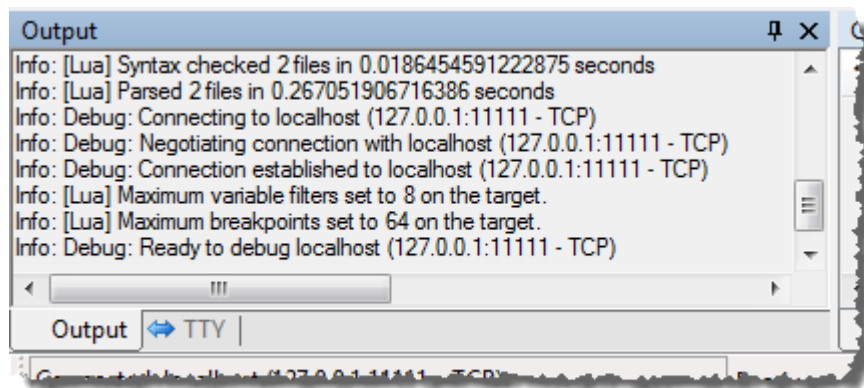


```
Select C:\SledSDK\bin\sce_sled\runtime\win32_static_dcr...
Main: TestTarget example started!
Main: Bringing up network subsystem
Misc:
    Input thread stack size: 8192 bytes
    Runner thread stack size: 73728 bytes
FileLoadThread: Attempting to open file: libsce_testtarget_script1.lua
FileLoadThread: Opened file libsce_testtarget_script1.lua successfully!
FileLoadThread: Attempting to open file: libsce_testtarget_script2.lua
FileLoadThread: Opened file libsce_testtarget_script2.lua successfully!
ScopedSledDebugger 1: Creating SledDebugger instance (4404 bytes)...done!
ScopedSledDebugger 1: SledDebugger version 5.0.0
ScopedSledDebugger 1: Creating Lua Plugin (632516 bytes)...done!
ScopedSledDebugger 1: LuaPlugin version 5.0.0
ScopedSledDebugger 1: Starting networking...
ScopedSledDebugger 1: [TCP] Listening on port: 11111
ScopedSledDebugger 1: Waiting for SLED? no
ScopedSledDebugger 1: Successful!
Lua: lua_loadbuffer succeeded!
Lua: lua_pcall succeeded!
Lua: lua_loadbuffer succeeded!
Lua: lua_pcall succeeded!
ScopedSledDebugger 1: Successfully initialized!
ScopedSledDebugger 2: Creating SledDebugger instance (4404 bytes)...done!
ScopedSledDebugger 2: SledDebugger version 5.0.0
ScopedSledDebugger 2: Creating Lua Plugin (632516 bytes)...done!
ScopedSledDebugger 2: LuaPlugin version 5.0.0
ScopedSledDebugger 2: Starting networking...
ScopedSledDebugger 2: [TCP] Listening on port: 11112
ScopedSledDebugger 2: Waiting for SLED? no
ScopedSledDebugger 2: Successful!
Lua: lua_loadbuffer succeeded!
Lua: lua_pcall succeeded!
Lua: lua_loadbuffer succeeded!
Lua: lua_pcall succeeded!
ScopedSledDebugger 2: Successfully initialized!
Main: Everything loaded successfully! Starting threads...
Main: Press 'q' then 'enter' at any time to exit
Main: thread "Debugger 0" started
Main: thread "Debugger 1" started
```

7. From the SLED menu, select **Debug > Connect**. Text similar to the following figure appears in SLED's **Output** tab.

Figure 4

SLED Output Tab Display on Successful Connection with TestTarget



These messages confirm that SLED has been successfully configured to work with the target platform.

6 Example: Using SLED as Your Game Debugger

To use SLED to debug your game, you need to create a SLED project that corresponds or maps to your game. This chapter shows how to create a SLED project for your game using the sample application TestTarget. For instructions on building and running TestTarget, see [Confirming Successful Building and Setup](#).

Requirements

To enable SLED to connect to your game and to allow you to debug your game's Lua scripts, the SLED project that corresponds to your game must be open, loaded, and running on the SLED side, and the game must be active and running on the target machine.

The target machine is one of the supported game platforms: Windows.

The project file (the `.spf` file) enables debugging your Lua scripts.

SLED needs the following to allow you to use it to debug your game's Lua scripts:

- Set up a target. For instructions, see [Confirming Successful Building and Setup](#).
- Create a SLED project for your game. For instructions, see the next section, [Creating a SLED Project](#).
- Ensure that you include the SLED libraries, LibSledDebugger and LibSledLuaPlugin, in your game code so that SLED can connect and work with the game.
- Run your game on the target machine. For this example, run the appropriate version of `libsce_testtarget-5.1.4_sample.exe`. For details, see [Confirming Successful Building and Setup](#).
- Connect to your running game from SLED. You must have created a SLED project for your game and the SLED project must be loaded and running in SLED before you can connect. For this example, connect to TestTarget. For details, see [Confirming Successful Building and Setup](#).
- Set breakpoints. You must have created a SLED project for your game and the SLED project must be loaded and running in SLED before you can set breakpoints. Use the SLED **Debug** menu item to set and toggle breakpoints. See the "Breakpoints" section in the *SLED and Lua User's Guide* for more information.

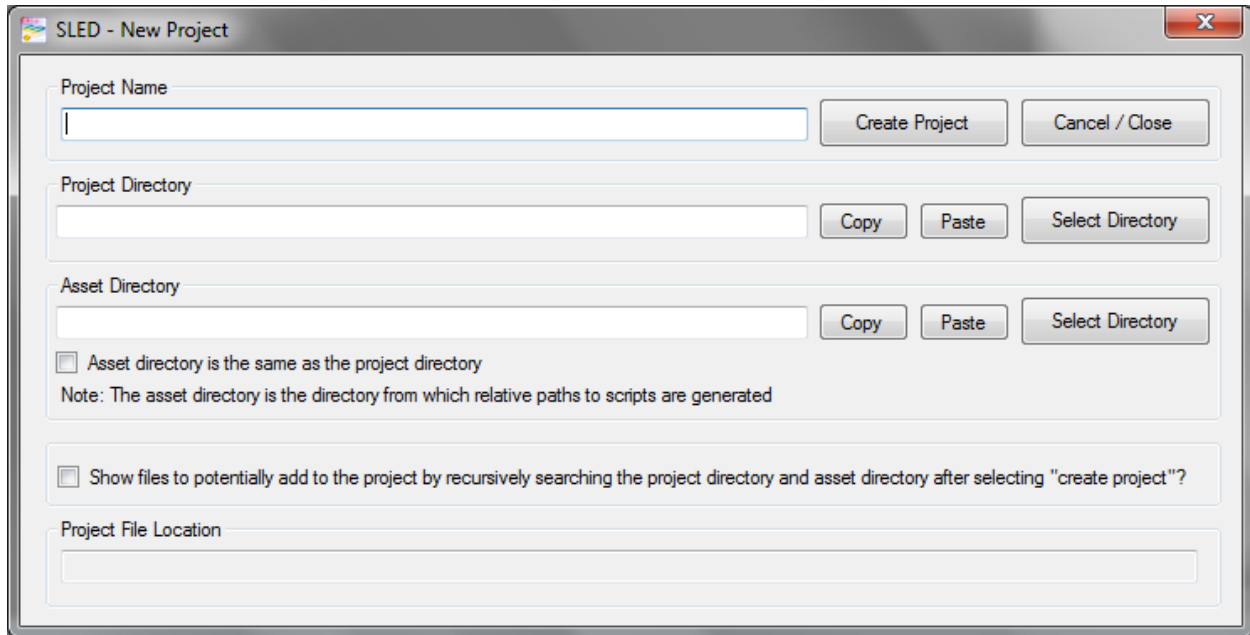
Creating a SLED Project

This section explains how to create a SLED project so you can debug your game using SLED. Here, you create a SLED project for the sample application TestTarget.

1. Open SLED if it is not already open.
2. Choose **File > Project > New**. The **New Project** window appears, as shown in the following figure.

Figure 6

New Project Window

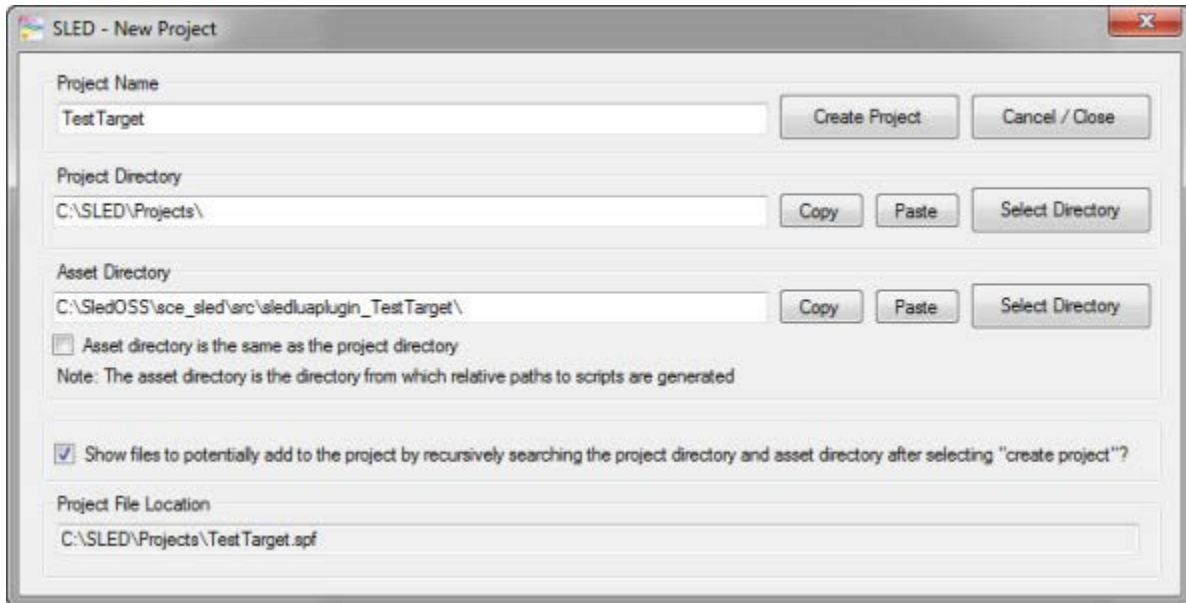


The image shows a Windows-style dialog box titled "SLED - New Project". It contains several input fields and buttons. The "Project Name" field is at the top, followed by "Project Directory" and "Asset Directory" fields. Each of these fields has "Copy", "Paste", and "Select Directory" buttons to its right. Below the "Asset Directory" field, there is a checkbox labeled "Asset directory is the same as the project directory" and a note: "Note: The asset directory is the directory from which relative paths to scripts are generated". Further down, there is another checkbox labeled "Show files to potentially add to the project by recursively searching the project directory and asset directory after selecting 'create project'?". At the bottom, there is a "Project File Location" field. On the right side of the dialog, there are two buttons: "Create Project" and "Cancel / Close".

3. Enter a project name in the **Project Name** field. For this example, enter TestTarget.
4. Click **Select Directory** in the **Project Directory** group pane. A **Browse for Folder** window appears.
5. Browse to and select any directory you'd like for this example. After you select the project directory, click **OK**.
6. Click **Select Directory** in the **Asset Directory** group box. A **Browse for Folder** window appears.
7. Navigate to `c:\SledOSS\sce_sled\src` and select the folder `sledluaplugin_TestTarget`. This directory contains TestTarget's assets.
8. Check **Show files to potentially add ...?** check box. The **New Project** window should be similar to the following figure.

Figure 7

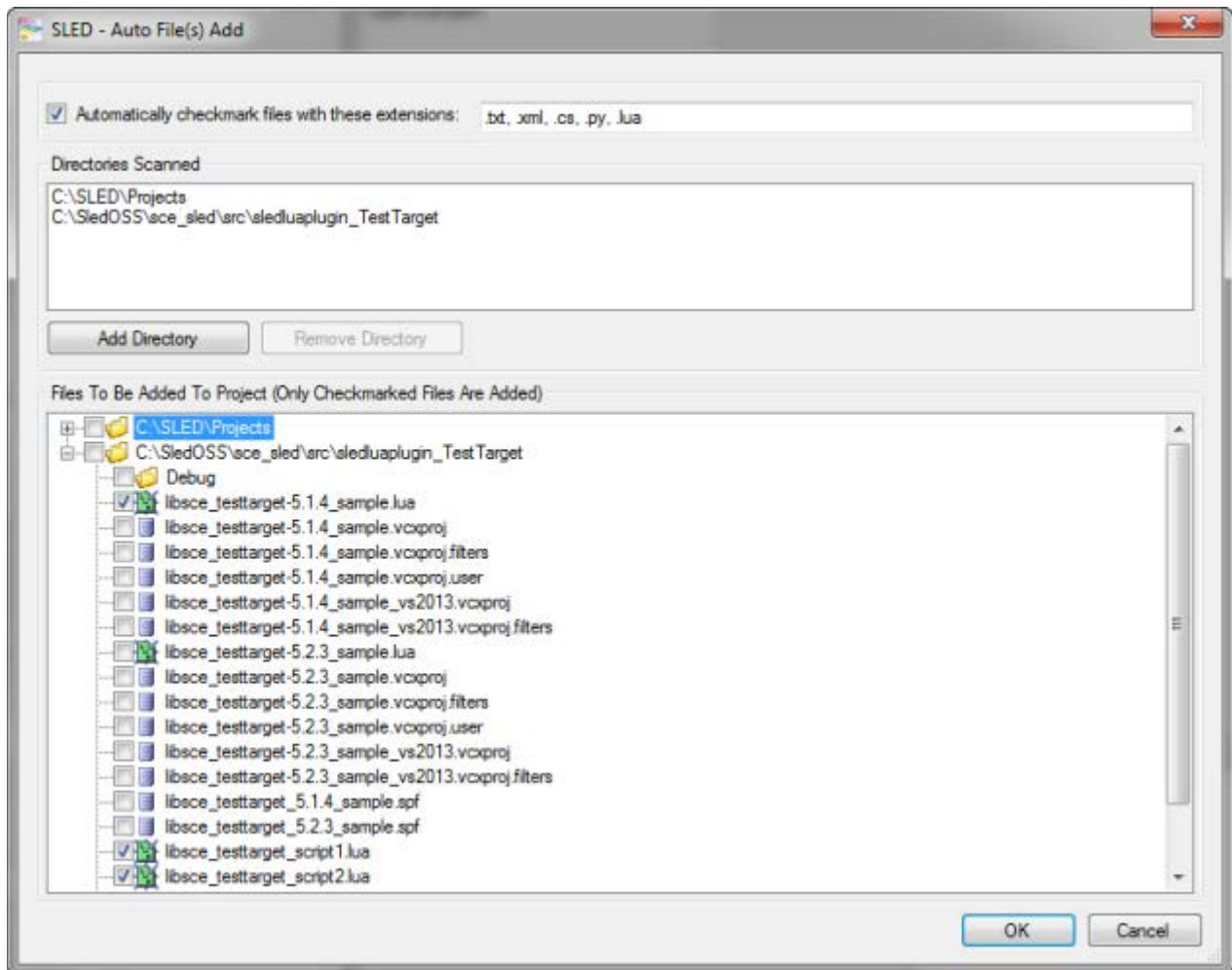
Completed New Project Window



9. Click **Create Project**. The **SLED Auto File(s) Add** window appears. The **Directories Scanned** field shows the Project and Assets folders. You can add additional directories by clicking the **Add Directory** button. The **Files To Be Added to Project** field lists all files in these directories.
10. Checking **Automatically checkmark files with these extensions** automatically checks all files that have one of the extensions listed in the field adjacent to this checkbox. The **Auto File(s) Add** dialog should now be similar to the following figure.

Figure 8

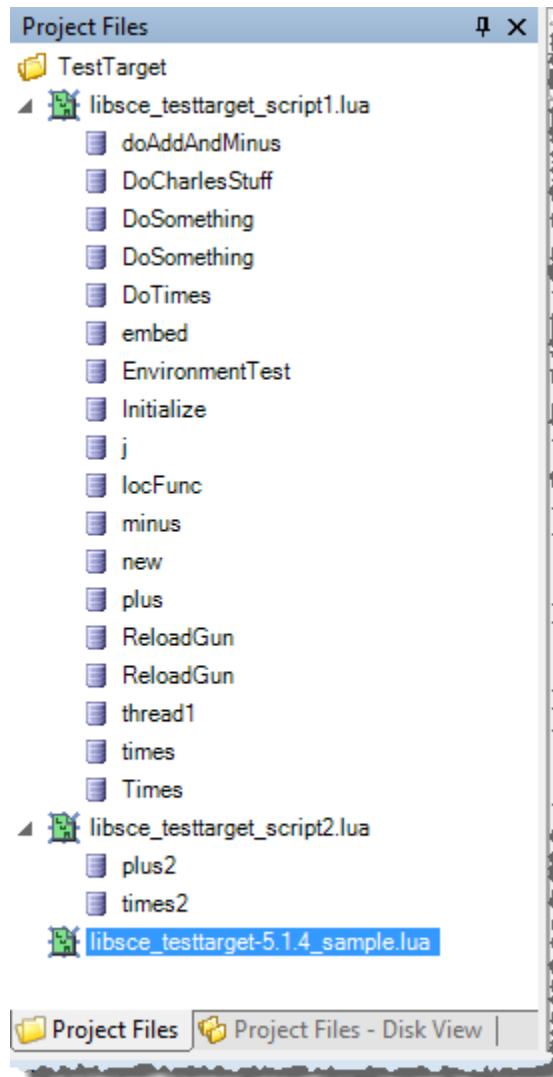
Completed Auto File(s) Add Window



11. Ensure that only the files you want in the project are checked in the **Files To Be Added to Project** list box. Typically you select .lua files.
12. Click **OK**. SLED opens the newly created project. The **Project Files** window lists the `libsce_testtarget_script1.lua`, `libsce_testtarget_script2.lua`, and `libsce_testtarget-5.1.4_sample.lua` files, similar to the following figure:

Figure 9

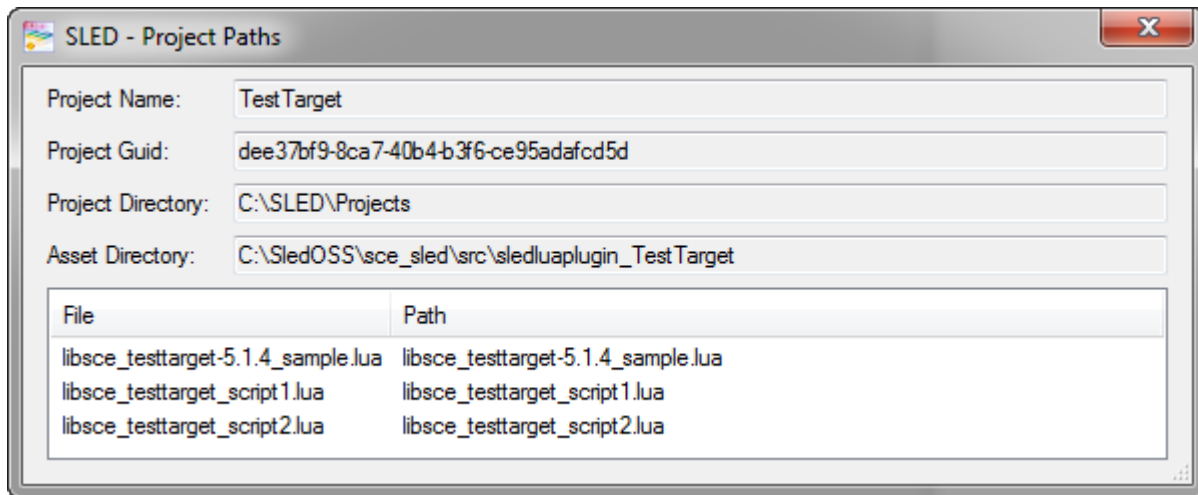
Test Target Project Files



13. Check that the project's relative paths match the relative paths used in LibSledLuaPlugin. The relative paths must match, so that SLED hits your project's breakpoints. To compare the relative paths, select **Project > View Project Paths**. The **Project Paths** window appears, similar to the following figure:

Figure 10

SLED Project Paths Window



Connect to Your Game and Go

Now that you have created your project, you can connect to your game and debug its Lua scripts. For details on doing this, see the *SLED and Lua User's Guide*.