

Squirrel Remote Debugger 1.0

Alberto Demichelis

Squirrel Remote Debugger 1.0

Alberto Demichelis

Copyright © 2003-2005 Alberto Demichelis

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
 3. This notice may not be removed or altered from any source distribution.
-

Table of Contents

1. The Debugger	1
Overview	1
Integrating the debugger	1
The network protocol	2
2. API Reference	3
Debugger	3
Index	5

Chapter 1. The Debugger

This part of the document describes how integrate the squirrel remote debugger in a application.

Overview

SQDBG is a tiny C++ library that allows to easily expose squirrel's VM internal state to an external debugger/IDE application. The library is based on TCP/IP and exposes its functionalities through only 4 public functions.

Integrating the debugger

The debugger is designed to be non-intrusive, because of this is based on non blocking socket and won't spawn any thread or other OS objects(except for the TCP socket).

The debugger logical flow is extremely simple.

- The application initializes squirrel's VM.

Note that for step by step debugging, debug info's generation must be enabled(see `sq_enableddebuginfo()`).

- The application initializes SQDBG(`sq_rdbg_init()`).
- The application waits for an incoming connection from a client debugger(`sq_rdbg_waitforconnections()`).
- The application updates the debugger.

This can be done through `sq_rdbg_update()` or by enabling 'autoupdate' while initializing the debugger.

```
int main(int argc, char *argv[])
{
    if(argc < 2){
        sprintf(_SC("SQDBG error : no file specified"));
        return -1;
    }

    HSQUIRRELVm v = sq_open(1024);
    sqstd_seterrorhandlers(v);

    ///!! INITIALIZES THE DEBUGGER ON THE TCP PORT 1234
    ///!! ENABLES AUTOUPDATE
    HSQREMOTEDBG rdbg = sq_rdbg_init(v,1234,SQTrue);

    ///!! ENABLES DEBUG INFO GENERATION(for the compiler)
    sq_enableddebuginfo(v,SQTrue);

    sq_setprintfunc(v,printfunc);

    const SQChar *fname=NULL;
#ifdef _UNICODE
    SQChar sTemp[256];
    mbstowcs(sTemp,argv[1],(int)strlen(argv[1])+1);
```

```
        fname=sTemp;
#else
        fname=argv[1];
#endif

        sprintf(_SC("SQDBG file =%s\n"),fname);
        if(!rdbg){
            sprintf(_SC("error starting the debugger"));
            return -1;
        }

        ///!! SUSPENDS THE APP UNTIL THE DEBUGGER CLIENT CONNECTS
        if(SQ_SUCCEEDED(sq_rdbg_waitforconnections(rdbg))) {
            sprintf(_SC("connected\n"));

            ///!!EXECUTES A SCTIPT
            sq_pushroottable(v);
            sqstd_dofile(v,fname,SQFalse,SQTrue);
        }
        ///!! CLEANUP
        sq_rdbg_shutdown(rdbg);
        sq_close(v);
        return 0;
    }
}
```

The network protocol

TODO

Chapter 2. API Reference

Debugger

`sq_rdbg_init`

```
HSQREMOTEDBG sq_rdbg_init(HSQUIRRELVm v, unsigned short port, SQ-  
Bool autoupdate);
```

creates a new instance of the squirrel remote debugger.

parameters:

<i>HSQUIRRELVm v</i>	the target VM to be debugged
<i>unsigned short port</i>	TCP/IP port to listen at
<i>SQBool autoupdate</i>	enable/disable debugger autoupdate

return: an handle to a squirrel debugger

remarks: if autoupdate is true the debugger will automatically call `sq_rdbg_update()` every time a line is executed. Autoupdate is a good choice for applications that do not have a busy loop, however the `sq_rdbg_update()` performs IO operations and this will impact on the application performances. For applications like games is recommended to set autoupdate to false and call `sq_rdbg_update()` once per frame.

`sq_rdbg_shutdown`

```
HRESULT sq_rdbg_shutdown(HSQREMOTEDBG rdbg);
```

Suspends the execution of the specified vm.

parameters:

<i>HSQREMOTEDBG rdbg</i>	the target debugger
--------------------------	---------------------

return: an SRESULT

`sq_rdbg_update`

```
HSQREMOTEDBG sq_rdbg_update(HSQREMOTEDBG rdbg);
```

updates the internal state of the debugger. if the parameter 'autoupdate' is set to true at initialization(`sq_rdbg_init()`), this function is automatically invoked by the debugger everytime a line is executed.

parameters:

HSQREMOTEDBG rdbg the target debugger

return: a SQRESULT (if the result is different than SQ_OK, the debugger wont work again and needs to be reinitialized).

remarks: Autoupdate is a good choice for applications that do not have a busy loop, however the sq_rdbg_update() performs IO operations and this will impact on the application performances. For applications like games is raccomanded to set autoupdate to false and call sq_rdbg_update() once per frame.

<code>sq_rdbg_waitforconnections</code>

SQRESULT **sq_rdbg_waitforconnections**(*HSQREMOTEDBG rdbg*);

waits for incoming connections from a client debugger.

parameters:

HSQREMOTEDBG rdbg the target debugger

return: an SQRESULT

Index

S

sq_rdbg_init, 3
sq_rdbg_shutdown, 3
sq_rdbg_update, 3
sq_rdbg_waitforconnections, 4