Common Language Extension (Programmer's Guide)

Contents

Contents 2
1 Introduction 10
2 Install CLE 10
2.1 Install 10
2.1.1 windows & linux 10
2.1.2 android 11
2.1.3 ios 11
2.1.4 windows phone 8/8.1/10 11
2.2 Programming environment(for linux, windows) 11
2.2.1 Install starcore 12
2.2.2 C/C++ 12
2.2.3 Install python 12
2.2.4 Install ruby 13
2.2.5 Install java 13
2.2.6 Install .NET(skip) 13
2.2.7 Debug and compile(linux or windows) 14
2.2.7.1 Compile 14
2.2.7.2 Debug 18
2.2.8 Run(linux or windows) 19
2.3 starcore tools(linux and windows) 19
2.3.1 starapp:cle application running environment, which can load share library, lua/java/python/csharp scripts 19
2.3.2 starmodule 21
2.3.3 star2c/star2h,generate header file and code skeleton. 21
2.3.4 starsrvinstinfo 22
2.3.5 starsrvreg 22
2.3.6 starsrvparse/starsrvunparse 23
2.3.7 starsrvpack 23
3 Init CLE 24
3.1 Architecture of CLE 24
3.2 Init Cle using C++ 24
3.2.1 init type 1, create service group directly 25
3.2.2 init type 2, create service directly 25
3.2.3 init type 3, load libstarcore share library manually 25
3.2.4 init type 4, link with libstarcore share library 26
3.2.5 link errors for vsxx 27
3.3 Init Cle using lua 28
3.3.1 init type 1, create service group directly 28
3.3.2 init type 2, create service directly 28
3.3.3 init type 3, create service step by step 28
3.4 Init Cle using python 29
3.4.1 init type 1, create service group directly 29
3.4.2 init type 2, create service directly 29
3.4.3 init type 3, create service step by step 29
3.5 Init Cle using ruby 30
3.5.1 init type 1, create service group directly 30
3.5.2 init type 2, create service directly 30
3.5.3 init type 3, create service step by step 30
3.6 Init Cle using java 31

3.6.1 init type 1, create service group directly 31
3.6.2 init type 2, create service directly 31
3.6.3 init type 3, create service step by step 31
3.7 Init Cle using csharp/csharp4/csharp45/csharp451 32
3.7.1 init type 1, create service group directly 34
3.7.2 init type 2, create service directly 34
3.7.3 init type 3, create service step by step 35
3.7.4 compile using command line 35
3.8 Create object, define it's attributes and functions 35
3.8.1 python 35
3.8.2 lua 36
3.8.3 ruby 36
3.8.4 java 38
3.8.5 c# 39
3.8.6 c++ 40
3.8.7 Call object's function 43
3.8.7.1 python 43
3.8.7.2 lua 44
3.8.7.3 java 44
3.8.7.4 c# 44
3.8.7.5 c++ 44
3.9 Message loop in CLE 44
3.9.1 c#/java 45
3.9.2 android 45
3.10 Global Lock 46
3.11 Problems that need attention 47
3.12 CLE Environments 48
3.12.1 SRPHOME 48
3.12.2 SRPMODULE 48
3.13 language Locale 48
3.14 notes for android, ios， wp, winrt and windows 10 48
3.14.1 android 48
3.14.2 using ruby on android 50
3.14.3 using cle in native app 50
3.14.4 ios 52
3.14.5 wp or windows store or windows 10 53
3.14.6 winrt 53
3.14.7 win10 54
4 Capture output of CLE or other scripts 54
4.1 c/c++ 55
4.2 java 55
4.3 csharp 56
4.4 lua 56
4.5 python 56
4.6 ruby 57
5 C Interface 57
5.1 Init Cle using C 58
5.2 Using c interface function 58
6 Delphi Interface 61
6.1 Using cle with delphi on windows 61
6.1.1 Add "starcore.pas" 61
6.1.2 Init Cle 62
6.1.3 Using TSRPVariant to access object 62

6.1.4 Sample Code 64
6.1.5 Call Tensorflow 64
6.2 Using cle with delphi on android 67
6.2.1 Create Project and Add "starcore.pas" 67
6.2.2 Add cle share libraries. 68
6.2.3 Init Cle 69
6.2.4 Call java code 70
6.3 Using cle with delphi on ios 71
6.3.1 Create Project and Add "starcore.pas" 71
6.3.2 Set Link with stdc++ 72
6.3.3 Init Cle 73
6.3.4 Deploy files 74
6.4 Using cle with delphi on ios simulator 75
6.4.1 Create Project and Add "starcore.pas" 75
6.4.2 Add cle share libraries for simulator 75
6.4.3 Init Cle 76
6.4.4 Deploy files 76
6.4.5 Using python 76
6.5 Using CLEString 78
6.6 Interact with other scripts 79
6.6.1 Define object's callback 79
6.6.2 Create CLE Object 80
6.7 Capture print formation from cle 80
6.8 Using TSRPParaPkg, TSRPBinBuf, TSRPSXml, TSRPComm 81
7 C++Builder Interface 81
7.1 Using cle with c++ builder on windows 82
7.1.1 Init CLE 82
7.2 Using cle with c++ builder on android 83
7.2.1 Init CLE(First Method) 83
7.2.2 Init CLE(Second Method) 84
7.2.2.1 Deployment 84
7.2.2.2 Init CLE 90
7.2.2.3 Call android java code from lua 90
7.3 Using cle with c++ builder on ios 91
7.3.1 Init CLE 91
7.3.2 Use python 92
7.4 Using Variant to encapsulate cle object 94
7.4.1 How to use variant 95
7.4.2 Sample Code 96
7.4.3 Call tensorflow 96
7.5 Compile error for xe6/xe7 99
8 script raw operations 99
8.1 Special object and function for lua, python and c#. 100
8.2 Parameters mapping between scripts. 102
8.3 Parameters mapping between scripts as function input. 104
8.4 Callback from script. 104
8.5 extend script class. 105
8.6 script files to be called 109
8.6.1 testlua.lua 109
8.6.2 testpy.py 109
8.6.3 TestJava.java 110
8.6.4 test java proxy 111
8.6.5 test java class extend 112

8.6.6 testcs 113
8.6.7 test cs proxy 116
8.6.8 test cs class extend 116
8.7 c/c++ call other raw script functions 118
8.7.1 call lua 118
8.7.1.1 create project 118
8.7.1.2 source file 120
8.7.2 call python 121
8.7.2.1 create project 121
8.7.2.2 source file 121
8.7.3 call java 122
8.7.3.1 create project 122
8.7.3.2 source file 122
8.7.4 call java with callback 123
8.7.4.1 create project 123
8.7.4.2 source file 123
8.7.5 call java extend class 125
8.7.5.1 create project 125
8.7.5.2 source file 125
8.7.6 call cs 126
8.7.6.1 create project 126
8.7.6.2 source file 126
8.7.7 call cs with callback 127
8.7.7.1 create project 127
8.7.7.2 source file 128
8.7.8 call cs extend class 129
8.7.8.1 create project 129
8.7.8.2 source file 129
8.8 lua call other raw script functions 130
8.8.1 call c dll 130
8.8.2 call python 131
8.8.3 call java 132
8.8.4 call java with callback 132
8.8.5 call java extend class 133
8.8.6 call cs 134
8.8.7 call cs with callback 134
8.8.8 call cs extend class 135
8.9 python call other raw script functions 136
8.9.1 call c dll 136
8.9.2 call lua 137
8.9.3 call java 137
8.9.4 call java with callback 138
8.9.5 call java extend class 139
8.9.6 call cs 139
8.9.7 call cs with callback 140
8.9.8 call cs extend class 141
8.10 java call other raw script functions 142
8.10.1 call c dll 142
8.10.2 call lua 142
8.10.3 call python 143
8.10.4 call cs 144
8.10.5 call cs with callback 145
8.10.6 call cs extend class 146

8.11 cs call other raw script functions 147
8.11.1 call c dll 147
8.11.1.1 create project 147
8.11.1.2 source file 148
8.11.2 call lua 149
8.11.3 call python 150
8.11.4 call java 151
8.11.5 call java with callback 152
8.11.6 call java extend class 153
8.12 some examples 154
8.12.1 lua call java awt 154
8.12.2 lua call cs form 156
8.13 Errors and Exceptions. 158
8.14 Directly assign c/c++, c#, java and object-c object to lua,python and ruby 158
8.14.1 Assign c/c++ object to scripts 158
8.14.2 Assign java object to scripts 161
8.14.3 Assign c# object to scripts 164
8.14.4 Assign Object-C object to scripts 166
9 Calling lua, python or ruby on android, ios, wp,windows 10 168
9.1 using cle on ios 168
9.1.1 c++ calling lua 168
9.1.2 c++ calling python 172
9.1.3 c++ calling ruby 178
9.1.4 ObjectC bridge for scripts 182
9.2 using cle on android 185
9.2.1 java calling lua 186
9.2.2 java calling python 189
9.2.3 java calling ruby 194
9.3 using cle on wp, windows 10 200
9.3.1 native calling lua 200
9.3.2 c# calling lua 204
9.3.3 using lua to handle button event. 207
9.3.4 cs calling lua [windows 10] 207
9.3.5 cs calling python [windows 10] 210
9.3.6 cs calling ruby [windows 10] 214
9.3.7 notes 217
10 Multithreading 217
11 Binary data mapping 218
12 Double or Float as Native Function Parameter 219
13 Develop common extension. 219
13.1 Common extension 220
13.1.1 Develop common extension using python 220
13.1.2 Develop common extension using lua 220
13.1.3 Develop common extension using java 220
13.1.4 Develop common extension using C++ 221
13.1.5 Develop common extension using C# 221
13.2 Call common extension using C/C++ 222
13.3 Call common extension using lua 223
13.4 Call common extension using python 223
13.5 Call common extension using java 223
13.6 Call common extension using C# 224
13.7 passing complex data structures between languages 225
13.7.1 Extension module to be called 226

13.7.1.1 Develop common extension using python 226
13.7.1.2 Develop common extension using lua 227
13.7.1.3 Develop common extension using java 227
13.7.1.4 Develop common extension using C++ 228
13.7.1.5 Develop common extension using C# 229
13.7.2 Call common extension using C/C++ 230
13.7.3 Call common extension using lua 232
13.7.4 Call common extension using python 232
13.7.5 Call common extension using java 233
13.7.6 Call common extension using C# 233
13.8 A more complicated example 234
13.8.1 java swing window(Callback function) 234
13.8.1.1 Common extension developed by java to create a window using swing 234
13.8.1.2 Call using python 235
13.8.1.3 Call using C++ 236
13.8.1.4 Call using c# 237
13.8.2 call jsoup 238
13.8.2.1 Common extension developed by java to create an interface object to jsoup 238
13.8.2.2 Call using python 239
13.8.2.3 Call using C/C++ 240
13.8.2.4 Call using c# 240
13.8.3 c# form calls java 241
13.9 Direct call share library 242
13.9.1 lua calls MessageBox 243
13.9.2 Java calls MessageBox 243
13.9.3 c# calls MessageBox 244
13.10 Mixed script language programming 244
13.10.1 Module to be called 244
13.10.1.1 lua 244
13.10.1.2 python 245
13.10.1.3 java 245
13.10.1.4 c# 245
13.10.2 C/C++ call other script 246
13.10.3 lua call other script 247
13.10.4 python call other script 247
13.10.5 java call other script 247
13.10.6 c# call other script 248
13.11 ASP.NET call CLE extensions 248
14 CLE distributed function 249
14.1 TCP/UDP communication 249
14.1.1 TCP server 249
14.1.1.1 C 249
14.1.1.2 lua 253
14.1.1.3 python 254
14.1.1.4 java 255
14.1.1.5 c# 256
14.1.2 TCP client 257
14.1.2.1 C 257
14.1.2.2 lua 259
14.1.2.3 python 260
14.1.3 UDP server 261
14.1.3.1 C 261
14.1.3.2 lua 262
14.1.3.3 python 263

14.1.4 UDP client 264
14.1.4.1 C 264
14.1.4.2 lua 265
14.1.4.3 python 266
14.2 Remotecall 267
14.2.1 Create server side application 267
14.2.1.1 C 267
14.2.1.2 lua 271
14.2.1.3 python 272
14.2.2 Create client side application 272
14.2.2.1 Win32 272
14.2.2.2 linux 274
14.2.2.3 lua 274
14.2.2.4 python 274
14.2.3 Creating and using starcore service 275
14.2.3.1 Create starcore service 275
14.2.3.2 Using starcore service 282
14.3 Remotecall-complicate data type 286
14.3.1 Create server side application 287
14.3.1.1 C 287
14.3.1.2 lua 290
14.3.1.3 python 291
14.3.2 Create client side application 292
14.3.2.1 Win32 292
14.3.2.2 lua 293
14.3.2.3 python 294
14.3.3 Create and ust stand alone starcore service 295
14.3.3.1 Create starcore service 295
14.3.3.2 Export skeleton file 296
14.3.3.3 create module 297
14.3.4 called by LUA 298
14.3.5 called by Python 298
15 Webservice and http application 298
15.1 Http&HttpServer 298
15.1.1 Http download 298
15.1.1.1 C 298
15.1.1.2 lua 300
15.1.1.3 python 301
15.1.2 Http upload 302
15.1.2.1 C 302
15.1.2.2 lua 303
15.1.2.3 python 303
15.1.3 Simple HttpServer 304
15.1.3.1 C 304
15.1.3.2 lua 306
15.1.3.3 python 307
15.1.4 HttpServer local request 308
15.1.4.1 C 308
15.1.4.2 lua 311
15.1.4.3 python 312
15.2 WebService 313
15.2.1 Create WebService 313
15.2.1.1 WebService object 313
15.2.1.2 lua 313
15.2.1.3 python 314

15.2.1.4 C....315
15.2.2 Get WSDL of WebService....320
15.2.3 WebService client(gsoap)....321
15.2.3.1 Win32....322
15.2.4 Create and use stand alone starcore service.....323
15.2.4.1 Called by C....323
15.2.4.2 called by LUA....327
15.2.4.3 Called by python....328
15.3 WebService-compilcate data type....328
15.3.1 Create Web service using LUA....330
15.3.2 Get WSDL of WebService....330
16 Starcore application packing....332
16.1 starcore packing....332
16.1.1 Packing applications....332
16.1.2 Packing applications developed with c/c++....333
16.1.2.1 Win32....333
16.1.2.2 linux....336
16.1.2.3 Packing and testing....338
16.2 Data files in package....338
16.2.1 pack to single file....339
16.2.1.1 C....339
16.2.2 Pack to directory....341
17 How to register....343
17.1 Difference between two versions....343
17.2 Buy and Register....344
17.3 use cle in application on other computers.....344
18 Distributing cle with your products....345
19 Q&A....345
19.1 Create network server or client failed on android....345
19.2 load share library failed....346
19.3 RuntimeBinderException of using dynamic in c#....346
19.4 Java init failed on MAC OSX....346
19.5 vccorlib_lib_should_be_specified_before_msvcrt_lib_to_linker....346
19.6 Init ruby or call ruby raw function fails from java command on linux....347
19.7 Init ruby or call ruby raw function fails from java command on linux....347
19.8 Init python3.6 interface failed on windows....347
19.9 onDestroy event on the android platform....347
19.10 Problems when installing 32bit and 64bit ruby Simultaneously on windows platform....348
19.11 Load ruby share library failed for version 2.4 or above on windows platform....348
19.12 Specifing ruby runtime version....348
19.13 Print function of python and ruby in thread....348
20 About srplab....348

1 Introduction

There are many programming languages. In addition to traditional language C/C++, script languages such as JAVA ,PYTHON,RUBY,LUA,C# are also introduced. Applications may be developped with proper and efficient language. For example, GUI applications are developped with JAVA or C#, low-layer applications use C/C++, etc. Although it is convenient, but it also introduces some problems: how to call each other, how the libraries or codes developped with one language are used in other languages easily,or how to reuse existing development results.

For example, to develop a library module with C/C++, general method is to write kinds of extensions, such as python extension, lua extension, JAVA extension, and so on. In order to write these extensions, not only to study interfaces of different languages, but also to write interface codes, in which many problems may be encountered causing longer period and unstable of the products. In addition, this effort is only the accumulation of experience. The result almost can not be used in a new product. The procedure will be repeated again. Therefore, a common extension development environment is expected.

There are many languages to choice. Therefore, how to perform mixed calls between languages is a problem .There are some solutions. For JAVA calling PYTHON, JPYTHON may be used, etc. But developpers have to study, understand in order to use them.

CLE is a common extension paltform. Libraries developped with CLE may be called by any other languages supported. In addition, CLE also provides a general pattern for mixed calls. CLE is cross-platform. It supports win32, linux X86, android, IOS, windows phone 8. Developing libraries using interface provided by CLE, and calling these libraries also uses the same interfaces. Programmer need only study once to use CLE in different languages.

CLE supports distributed object technique, which objects as medium to implement the mixed call between languages. Object is stored in a structed memory and a list of function pointers. Through mapping from the structed memory and function pointers to different languages, the above idea is realized. CLE is a share library and simple. It does not impose any restriction on specific language, and may be used to develop kinds of distributed applications easily.

2 Install CLE

2.1 Install

2.1.1 windows & linux

Current version is for 32 bit paltform. On windows, the share library is named libstarcore.dll, and on linux, the share library is named libstarcoreX.X.so.

Lua language is embedded in starcore, which needs not install alone.

Pre-compiled interface library .pyd or .so is for python2.7. and for android is python2.6.
python 3.3 is supported from version 2.1.0, for linux and windows version

Pre-compiled interface library .so is for ruby 1.9, 2.0 or 2.1.

Starcore environment config file, is mainly used to config python or other script languages. If no needs, you may do not care about or change the config file.

File name: starenvcfg.xml,
For win32, the file is located in C:\ srplab.
For linux, the file is located in /usr/local/srplab.
File format :

```
<?xml version="1.0" encoding="utf-8" ?>
<StarCoreEvnConfig Python="">
Python:Python share library name. If not set, default is python27.dll for win32 and libpython2.7.so for linux.
   <ExternScript>
      <script name="" Module="dll/so" para="XXX"/>
   </ExternScript>
   ExternScript: script language interpreter, may be any script language except lua and python. If the last char of Module
     is'/'or'\',then it is a path name, the interpreter will be searched in the directory, which name should be
     star_XXX.dll/libstar_XXX.so
</StarCoreEvnConfig>
```

2.1.2 android

CLE for android is a zip package. You can simply download it from web site and unzip to a directory.

android version includes two architecture "armeabi" and "armeabi-v7a". Alias name in cle for the two architecture are "android" and "androidv7a". File name of libraries of c/c++ service should add postfix "_android.so" or "_androidv7a.so".

For android, cle supports java in calling c/c++, lua and python. If you want use python, you should install SL4A or add python libraries into the project.

2.1.3 ios

CLE for ios is a static library. It can be downloaded from appstore.

2.1.4 windows phone 8/8.1/10

CLE for wp8 is a zip file including share libraries, header files, document, and assemblies. For wp8, Star_csharp assembly is the interface for c#.
File name of libraries of c/c++ service should add postfix "_wpx86.dll" or " _wparm.dll".

2.2 Programming environment(for linux, windows)

2.2.1 Install starcore

For windows:
Running package:
starcore_win32.X.X.exe

For linux :
rpm –i –nodeps starcore-X.X-1.i386.rpm

2.2.2 C/C++

Development tools:
For windows 2000,XP,2003,Vista,windows7 is CBuilder or VC series.
For linux, is gcc++ or gdb.

Header files:
vs_shell.h
vscoreshell.h: including functions about registry, string coding conversion functions.
vsopenapi.h,
vsopencommtype.h,
vsopencoredll.h,
vsopendatatype.h,
vsopenmemorydisk.h,
vsopensyseventdef.h,
vsopennetlink.h

For general applications, you only need to include vsopenapi.h and vsopensyseventdef.h.

#include "vsopenapi.h"
#include "vsopensyseventdef.h"

On windows platform, libstarcore.dll may be linked by adding libstarcore.lib into your project.
On linux system, add -lstarcore in your makefile.

Share library may be loaded dynamically. For example:

```
VS_CHAR ModuleName[512];

sprintf(ModuleName,"libstarcore%s",VS_MODULEEXT);
hDllInstance = vs_dll_open( ModuleName );
    if( hDllInstance == NULL ){
        printf("load library [%s] error....\n",ModuleName);
        return -1;
}
```

2.2.3 Install python

If wants CLE to support python, you should install python package. Default version supported is python2.7 and python 3.3

Win32:

Install python-2.7.msi

linux:

First to unload previous version:

rpm –e –nodeps python

download Python-2.7.tar.bz2

tar –jxvf Python-2.7.tar.bz2

./configure --enable-shared

make

make install

under directory usr/lib, create a link to share library

ln –s /XXXX/libpython2.7.so.1.0 /usr/lib/libpython2.7.so

2.2.4 Install ruby

linux:

tar -zxvf ruby-1.9.3-p484.tar.gz

cd ruby-1.9.3-p484

./configure --enable-shared

make

make install

2.2.5 Install java

CLE supports java version higher than 1.5. Java package can be downloaded from sun website.

Config environment variables.

win32:

CLASSPATH,addX:\srplab\libs\starcore.jar

if you want to use ecllipse,then java library X:\srplab\libs\starcore.jar should be imported first.

linux:

CLASSPATH,add /usr/local/srplab/libs/starcore.jar.

2.2.6 Install .NET(skip)

.NET Version should be higher than 3.5.

cle .net interface library is Star_csharp.dll/Star_csharp4.dll/Star_csharp45.dll/Star_csharp451, which is installed in GAC:

C:\WINDOWS\assembly\GAC_32\Star_csharp\1.0.1.0__7bc3b413a7df63bc

In directory c:\srplab\libs, there is a copy of Star_csharp.dll/Star_csharp4.dll/Star_csharp45.dll/ Star_csharp451.dll, which may be used in C# programming environment.

2.2.7 Debug and compile(linux or windows)

2.2.7.1 Compile

On win32, compile is simple after set correct path for included files.

On linux, using g++, as follows:

g++ -Wall -Wno-format -g -DDEBUG -DENV_LINUX -I/usr/include/starcore -o c_call.o -c c_call.cpp

g++ -o c_call_linux -g c_call.o -ldl -lpthread -lrt /usr/lib/libstarlib.a /usr/lib/libuuid.a

If you want to generate share lib, then:

g++ -fPIC -Wall -Wno-format -g -DDEBUG -DENV_LINUX -I/usr/include/starcore -o AddFunction.o -c AddFunction.cpp

g++ -shared -o ../AddFunction.so -g AddFunction.o -ldl -lpthread -lrt /usr/lib/libstarlib.a /usr/lib/libuuid.a

You also may write MakeFile. Here gives a template.

```
#*****************************************************************************
#
# Makefile for StarCore.
# www.srplab.com
#*****************************************************************************
DEBUG         := YES
PROFILE       := NO
#*****************************************************************************
CC     := gcc
CXX    := g++
LD     := g++
AR     := ar
RANLIB := ranlib

DEBUG_CFLAGS     := -Wall -Wno-format -g -DDEBUG -DENV_LINUX
RELEASE_CFLAGS   := -Wall -Wno-unknown-pragmas -Wno-format -O3 -DENV_LINUX

LIBS := -ldl -lpthread -lrt
EXTRA_LIBS := /usr/lib/libstarlib.a /usr/lib/libuuid.a

DEBUG_CXXFLAGS   := ${DEBUG_CFLAGS}
RELEASE_CXXFLAGS := ${RELEASE_CFLAGS}
```

```
DEBUG_LDFLAGS    := -g
RELEASE_LDFLAGS  :=

ifeq (YES, ${DEBUG})
  CFLAGS      := ${DEBUG_CFLAGS}
  CXXFLAGS    := ${DEBUG_CXXFLAGS}
  LDFLAGS     := ${DEBUG_LDFLAGS}
else
  CFLAGS      := ${RELEASE_CFLAGS}
  CXXFLAGS    := ${RELEASE_CXXFLAGS}
  LDFLAGS     := ${RELEASE_LDFLAGS}
endif

ifeq (YES, ${PROFILE})
  CFLAGS   := ${CFLAGS} -pg -O3
  CXXFLAGS := ${CXXFLAGS} -pg -O3
  LDFLAGS  := ${LDFLAGS} -pg
endif

#*******************************************************************************
# Makefile code common to all platforms
#*******************************************************************************

CFLAGS   := ${CFLAGS}   ${DEFS}
CXXFLAGS := ${CXXFLAGS} ${DEFS}

#*******************************************************************************
# include source and paths
#*******************************************************************************

INCS_T := /usr/include/starcore
INCS   = $(addprefix -I,$(INCS_T))

C_CALL_CXXSRCS := c_call.cpp

#*******************************************************************************
C_CALL_CXXOBJS := $(C_CALL_CXXSRCS:%.cpp=%.o)

#*******************************************************************************
CXXOBJS := ${C_CALL_CXXOBJS}
COBJS :=

EXEC_C_CALL_OBJS := ${C_CALL_CXXOBJS}

#*******************************************************************************
```

```
# Targets of the build
#*****************************************************************************
OBJS_PATH = .

EXEC_C_CALL := ./c_call_linux

all: ${EXEC_C_CALL}

#*****************************************************************************
# Output
#*****************************************************************************

${EXEC_C_CALL}: ${EXEC_C_CALL_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_C_CALL_OBJS} ${LIBS} ${EXTRA_LIBS}



#*****************************************************************************
# common rules
#*****************************************************************************

${CXXOBJS} :
	${CXX} ${CXXFLAGS} ${INCS} $< -o $@ -c $*.cpp

${COBJS} :
	${CC} ${CFLAGS} ${INCS} -o $@ -c $*.c

clean:
	-rm -f core ${CXXOBJS} ${COBJS} ${EXEC_C_CALL}
```

If you want to generate share library, then the Makefile is :

```
#*****************************************************************************
#
# Makefile for StarCore.
# www.srplab.com
#*****************************************************************************
DEBUG		:= YES
PROFILE		:= NO
#*****************************************************************************
CC	:= gcc
CXX	:= g++
LD	:= g++
AR	:= ar
RANLIB := ranlib
```

```
DEBUG_CFLAGS    := -Wall -Wno-format -g -DDEBUG -DENV_LINUX
RELEASE_CFLAGS  := -Wall -Wno-unknown-pragmas -Wno-format -O3 -DENV_LINUX

LIBS := -ldl -lpthread -lrt
EXTRA_LIBS := /usr/lib/libstarlib.a /usr/lib/libuuid.a

DEBUG_CXXFLAGS  := ${DEBUG_CFLAGS}
RELEASE_CXXFLAGS := ${RELEASE_CFLAGS}

DEBUG_LDFLAGS   := -g
RELEASE_LDFLAGS :=

ifeq (YES, ${DEBUG})
  CFLAGS      := ${DEBUG_CFLAGS}
  CXXFLAGS    := ${DEBUG_CXXFLAGS}
  LDFLAGS     := ${DEBUG_LDFLAGS}
else
  CFLAGS      := ${RELEASE_CFLAGS}
  CXXFLAGS    := ${RELEASE_CXXFLAGS}
  LDFLAGS     := ${RELEASE_LDFLAGS}
endif

ifeq (YES, ${PROFILE})
  CFLAGS  := ${CFLAGS} -pg -O3
  CXXFLAGS := ${CXXFLAGS} -pg -O3
  LDFLAGS  := ${LDFLAGS} -pg
endif

#*******************************************************************************
# Makefile code common to all platforms
#*******************************************************************************

CFLAGS  := ${CFLAGS}  ${DEFS}
CXXFLAGS := ${CXXFLAGS} ${DEFS}

#*******************************************************************************
# include source and paths
#*******************************************************************************

INCS_T := /usr/include/starcore
INCS  = $(addprefix -I,$(INCS_T))

ADDFUNCTION_CXXSRCS := AddFunction.cpp

#*******************************************************************************
```

```
ADDFUNCTION_CXXOBJS := $(ADDFUNCTION_CXXSRCS:%.cpp=%.o)

#*****************************************************************************
CXXOBJS := ${ADDFUNCTION_CXXOBJS}
COBJS :=

EXEC_ADDFUNCTION_OBJS := ${ADDFUNCTION_CXXOBJS}

#*****************************************************************************
# Targets of the build
#*****************************************************************************
OBJS_PATH = .

EXEC_ADDFUNCTION := ../AddFunction.so

all: ${EXEC_ADDFUNCTION}

#*****************************************************************************
# Output
#*****************************************************************************

${EXEC_ADDFUNCTION}: ${EXEC_ADDFUNCTION_OBJS}
	${LD} -shared -o $@ ${LDFLAGS} ${EXEC_ADDFUNCTION_OBJS} ${LIBS} ${EXTRA_LIBS}



#*****************************************************************************
# common rules
#*****************************************************************************

${CXXOBJS} :
	${CXX} -fPIC ${CXXFLAGS} ${INCS} $< -o $@ -c $*.cpp

${COBJS} :
	${CC} -fPIC ${CFLAGS} ${INCS} -o $@ -c $*.c

clean:
	-rm -f core ${CXXOBJS} ${COBJS} ${EXEC_ADDFUNCTION}
```

2.2.7.2 Debug

On win32 ,SRPWatch is output window of CLE. Outputs from starcore will be displayed in the watch window. Outputs may also be configured to output to syslog, which may be captured by syslog server.

syslog parameter config(server address and port number),may be set through config file, or interface function SetOutputPort. The interface is provided for C/C++,lua,python, and other script languages.

The output information is coded to utf-8 format.

telnet:

cle may open its telnet port, which may be enabled by config file, or interface function SetTelnetPort.

If telnet port is enabled, users can login telenet through telnet client, using lua or python to interact with starcore.

string coding is utf-8.

2.2.8 Run(linux or windows)

1. Using starapp.exe/starapp9.exe to load CLE applications, which may be share library, script file,etc.For example:
   starapp –e "XXX.class?script=java"
2. For python,may use command like : python filename.
3. For java,may use command like : java class name.
4. for c#,may use command, or use starapp –e "XXX.exe/dll?script=csharp"
5. for python 3.3, on windows: starapp –e XXX.py?script=python33. on linux: starapp –e XXX.py?script=python33
6. for ruby, on windows: starapp –e XXX.rb?script=ruby –imodule "X:\\XX\\XX\\libstar_ruby.so". on linux: starapp –e XXX.rb?script=ruby. By default, the program search registers for ruby share library of version 1.9.3, which is installed by rubyinstaller. For others, you can set ruby share library name or version, for example. starapp –e XXX.rb?script=ruby –imodule "X:\\XX\\XX\\libstar_ruby.so" –ipara "-m X:\\XX\\XX\\msvcrt-ruby191.dll" or
   starapp –e XXX.rb?script=ruby –imodule "X:\\XX\\XX\\libstar_ruby.so" –ipara "-v 2.0.0"

   -v parameter is only valid on windows desktop. For linux, libruby.so is always loaded.

note:

starapp9.exe is build with vs2008.

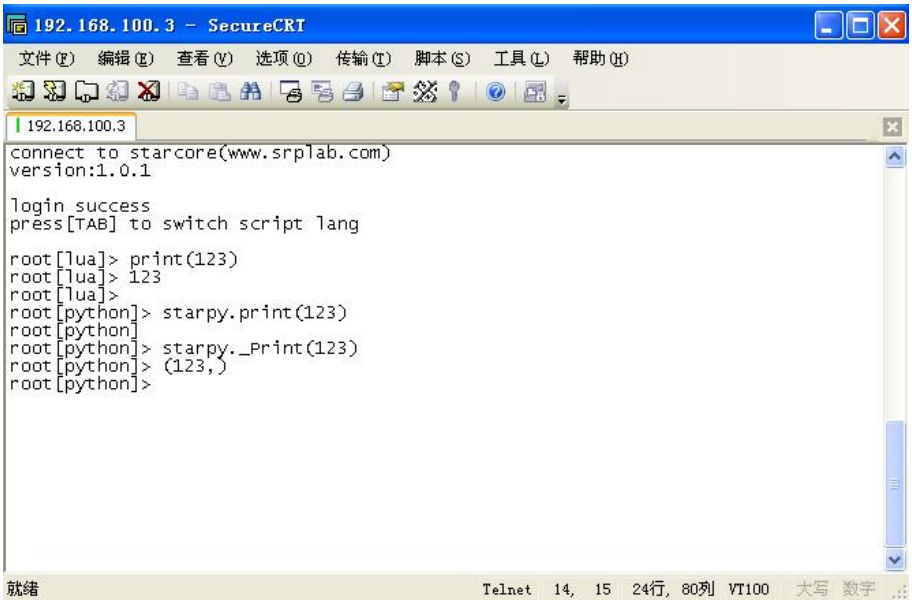
2.3 starcore tools(linux and windows)

2.3.1 starapp:cle application running environment, which can load share library, lua/java/python/csharp scripts.

starapp9.exe is build with vs2008.

```
starapp -e "share library"
starapp -e "XXXX.lua"
starapp -e "XXXX.py?script=python"
starapp -e "XXXX.class?script=java"
starapp -e "XXXX.exe/dll?script=csharp"
```

-t Telnet port. If the parameter is set, then you can use telnet client to connect to starcore. String is coded to utf-8,and supports lua and python language, using TAB to switch each other.



-w Web port. It the parameter is set, then you can use web browser to get staistic information or wsdl of cle application; External manager site can use http protocol to config and manage cle applications. Details is in later chapter. In addition, other applications can call webservice through the port.

-d Debug port. If the parameter is set, then service development tools (SRPDebug) may connect to starcore, to modify and create global object or its attributes

-c Client port, If the parameter is set, then client may connect to the application through the port.

-x xml config filename, which format is :

```
<?xml version="1.0" encoding="utf-8" ?>
<StarCoreConfig DynamicConfig="1" Host="127.0.0.1">
   <Config NotLoadModule="0" MinPortNumber="0" MaxPortNumber="0" />
   <Service NetPkgSize="0" UpLoadPkgSize="0" DownLoadPkgSize="0" DataUpPkgSize="0" DataDownPkgSize="0" />
   <Client Interface="" Port="0" ConnectionNumber="128" />
   <DebugServer Interface="" Port="0" />
   <Comm OutputHost="" OutputPort="0" TelnetPort="0" />
   <WebServer Port="0" ConnectionNumber="0" PostSize="0" />
   <StaticData CacheSize="10240000" Interface="" Port="0" ConnectionNumber="128" OverTimer="120" />
   <DataServer DirectConnect="0" Interface="" Host="" Port="0" />
   <RawSocket ServerNumber="0" ClientNumber="0" />
</StarCoreConfig>
```

DynamicConfig = 1 permit dynamic config,=0 not permit.

Host: IP address or domain name

Config:

NotLoadModule = 0, allows to load share library(dll/so),=1 not allow

MinPortNumber,MaxPortNumber: port number, =0 no limit, affect on RawSocket functions.

Service:

NetPkgSize,UpLoadPkgSize,DownLoadPkgSize,DataUpPkgSize,DataDownPkgSiz, it they eqaul to 0, then uses the value set by service,or else, uses these values.

Client:

Interface and Port are client connection parameters.

ConnectionNumber =0 means no limit.

DebugServer:

Interface and port for debugserver to connect.

Comm:

OutputHost,OutputPort: If they are set, then information will be print to the address, and coded to utf-8, syslog format. you can use syslog server to receive the information.

TelnetPort: telnet port number.

WebServer:

Port: port number

ConnectionNumber:number of pending connections

PostSize: upload file size, unit is KB.

StaticData: static data parameter

DataServer:data server parameter.

RawSocket:core raw socket parameter

--srpmodule XXX, set to load libstarcore.dll/so, example:

--srpmodule libstarcore

2.3.2 starmodule

starmodule is a tool to generate examples codes of cle module, which can be used in any language supported. This tool will help developer to write extensions with script language. It is released from version v2.5.0.

```
useage : starmodule modulename [-o output directory] [ [--example-all/--example-
c/--example-lua/--example-python/--example-java/--example-ruby/--example-cs] [--
class classname[--class classname[...]]] [--use-wrap/--use-raw] [--with-initpara
/--with-initpara-starcall] [--with-test] [--with-callback] ]

--example-all     :
--example-c       : default
--example-lua     :
--example-python  :
--example-java    :
--example-ruby    :
--example-cs      :
                  : generate module example code for c/lua/python/java/ruby/csha
rp
--class classname : set the class name in the module
--use-wrap        : generate code to wrap raw class
--use-raw         : generate code to wrap raw class, use raw functions, this fla
g must be used with --use-wrap or --use-raw
--with-initpara   : raw class with construct parameter
--with-initpara-starcall : raw class with construct parameter, call using method
 _StarCall. This flag enable use XXXX(XX,XX) to create instance for lua, python,
 or XXXX.new(XX,XX) for ruby. This flag must be used with --use-wrap or --use-ra
w.
--with-test       : generate test code and projects
--with-callback   : generate code for callback from module to apps
```

2.3.3 star2c/star2h,generate header file and code skeleton.

The two tools are used to create header files and code skeleton.

star2c, generates code skeleton, including header files, command line:

star2c {service url} { password of root user} [xml configfile]

Service url:maybe local path, local xml service file,or network path.

local path, example, service "aaa", under directory "d:\test", then the service url is: "d:\test\aaa"

network path, example, service "aaa", at http://www.XXX.com/XXX, then service url is: http://www.XXX.com/XXX/aaa.

xml config file may be omitted, which format is:

```
<?xml version="1.0" encoding="utf-8" ?>
<ExportModuleInfo  ExportModuleDir="..\project">
<TestModule>
   <TestClass/>
   <….>
</TestModule>
</ExportModuleInfo>
```

ExportModuleDir:output path

TestModule:module name which is defined in the service.

TestClass:Class included in the module.

star2h, only generate header file, command line:

star2h {service url} [-o output path] [-d dynamic service]

-d and -o may be omitted

2.3.4 starsrvinstinfo

Query starcore services registered, also can be used to unregister services. Command line:

starsrvinstinfo -s/-c/-d

Query registered services at server side (-s), cliet side(-c), or debug server side (-d).

starsrvinstinfo -s/-c/-d -u servicename

Unregistered services at server side, client side, or debug server side.

2.3.5 starsrvreg

Register starcore services, command line:

starsrvreg -s/-c/-d servicename

Service should locate on local disk.

For example, service aaa, in directory d:\test, then the servicename should be set to d:\test\aaa

you also can into directory d:\test, and run starsrvreg -s aaa

2.3.6 starsrvparse/starsrvunparse

Parse or unparse starcore service,

starsrvparse {xml service description file} [--o output path]

Input is service description file in xml format. It's syntax refers to service description document

starsrvunparse, convert starcore service to xml description file, command line:

starsrvunparse servicename {-u root password} {-o output filename} [-s ServicePath]

-s ServicePath, may be omitted.

2.3.7 starsrvpack

Pack service files to multiple files or single file, which is used to publish on network. The tools can also pack the files into executable file for win32.

Two formats:

1. starsrvpack {service name} {-s win32/linux} {-o output path}

   For example: public service for win32 + linux, then

   starsrvpack {service name} {-s win32} {-s linux} {-o output path}

   If –s is omitted, then default is packed for win32,linux,android, and all platforms supported

   Output file name is attached .bin postfix, which is used to solve download problems of website.

   For service published on network, if you want to create its header files, you can use command,

   star2h http://www.XXX.XXX/XXX/servicename.

2. starsrvpack {xml project file} {-s win32/linux} {-o output path}[-i pack to single file] [-f do not pack published starcore services] [-e pack to executable files on win32]

xml project file format,

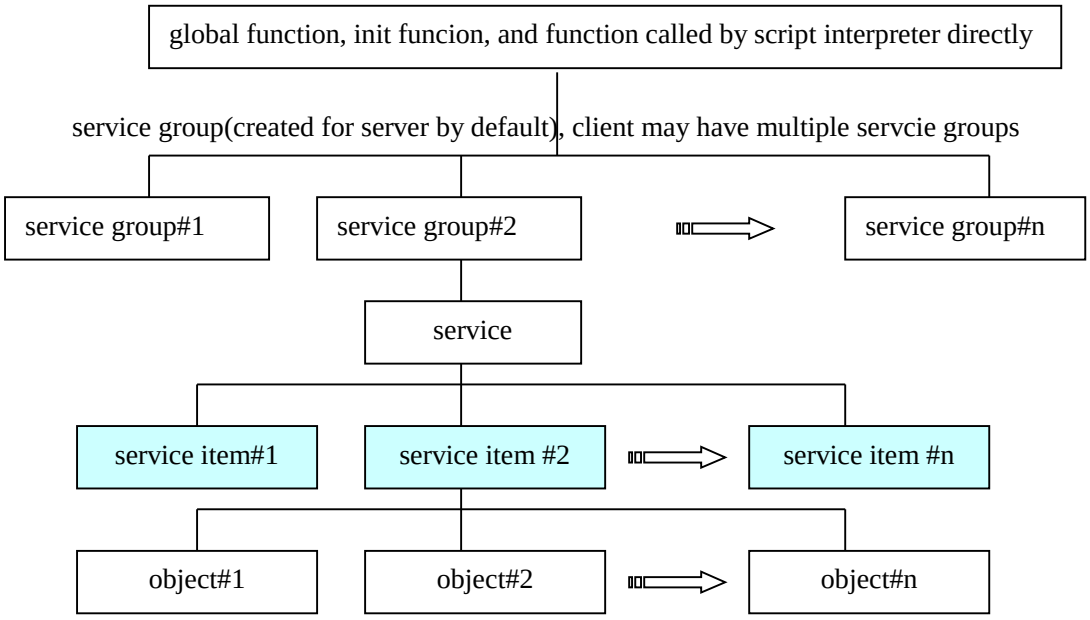
```
<?xml version="1.0" encoding="utf-8" ?>
<srpproject>
   <option>
      <name>GUIDemo</name>      note: output file name
      <output>..\output</output>      note: output path
      <script>lua/python</script>      note: script type, if not exist, default is lua.
   </option>
   <exec>
      <file name="GUIDll.dll" start="true/false" ostype="win32,linux" toutf8="true/false" />      note:start=true, indicates the file is a startup file, if ostype does not exist, then the file should support all platforms .
      toutf8 = true, when packing, the file is changed coding to utf8. If starup file is change to utf8, then after download complete, the file will be changed to local coding automatically by CLE and executed.
   </exec>
   <depend>
      <file name="SRPRenderEngine" />     note: depended service name
```

```
    <file name="D:\Work\starcore\service\irrlicht_srp\SRPIrrlichtEngine" />   note: depended service name
  </depend>
  <static>
     <file name="8.gif" />
    <path name="Media">              note: static data files will be downloaded before the service started.
       <file name="8.gif" />
       <file name="9.gif" />
       <file name="Back.jpg" />
    </path>
  </static>
  <dyna />                           note: dynamic data file, which will be downloaded on demand
</srpproject>
```

3 Init CLE

3.1 Architecture of CLE

Architecture of CLE is shown as follow:



Objects are grouped into four kinds : service group object, service object, service item object, object, where service item object may be not existed if you do not develop distributed applications.

Applications based on CLE is creating and managing the above four kinds of object. And then specific functions are provided by the objects.

3.2 Init Cle using C++

Headfiles used for C/C++ programming, is stored at X:\program files\srplab\starcore\files on win32, and /usr/include/starcore on linux. Project should link with starlib_vcm/ starlib_vcm9/ starlib_vcm10/ starlib_vcm11.lib[win32,for VC6,VC2008,VC2010,VC2012], and /usr/lib/libstarlib.a[linux]

3.2.1 init type 1, create service group directly

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
	VS_CORESIMPLECONTEXT Context;
	class ClassOfBasicSRPInterface *BasicSRPInterface;

	BasicSRPInterface = VSCore_InitSimpleEx(&Context, 0,0,NULL,0, NULL);
	//The last parameter should be  NULL.

	VSCore_TermSimple(&Context);
	return 0;
}
```

3.2.2 init type 2, create service directly

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
	VS_CORESIMPLECONTEXT Context;
	class ClassOfSRPInterface *SRPInterface;

	SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0, NULL);
	//The last parameter should be  NULL.

	VSCore_TermSimple(&Context);
	return 0;
}
```

3.2.3 init type 3, load libstarcore share library manually

```
#include "vsopenapi.h"

VS_HANDLE hDllInstance;
VSCore_InitProc VSInitProc;
VSCore_TermProc VSTermProc;
```

```
VSCore_QueryControlInterfaceProc QueryControlInterfaceProc;
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;

int main(int argc, char* argv[])
{
        VS_CHAR ModuleName[512];

        SRPControlInterface = NULL;
        BasicSRPInterface = NULL;
        sprintf(ModuleName,"libstarcore%s",VS_MODULEEXT);
        hDllInstance = vs_dll_open( ModuleName );
        if( hDllInstance == NULL ){
                printf("load library [%s] error....\n",ModuleName);
                return -1;
         }
         VSInitProc = (VSCore_InitProc)vs_dll_sym( hDllInstance, VSCORE_INIT_NAME );
         VSTermProc = (VSCore_TermProc)vs_dll_sym( hDllInstance, VSCORE_TERM_NAME );
          QueryControlInterfaceProc = (VSCore_QueryControlInterfaceProc)vs_dll_sym( hDllInstance,
VSCORE_QUERYCONTROLINTERFACE_NAME );
         VSInitProc( true, true, "", 0, "", 0,NULL);
        printf("init starcore success\n");
        SRPControlInterface = QueryControlInterfaceProc();
         BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);
       ….
        SRPControlInterface ->Release();
        BasicSRPInterface ->Release();
        VSTermProc();
        vs_dll_close(hDllInstance);
        return 0;
    }
```

3.2.4 init type 4, link with libstarcore share library

```
#include "vsopenapi.h"

static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;

int main(int argc, char* argv[])
{
         VSCore_Init( true, true, "", 0, "", 0,NULL);
        printf("init starcore success\n");
```

```
        SRPControlInterface = VSCore_QueryControlInterface();
         BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);


       …..
        SRPControlInterface ->Release();
        BasicSRPInterface ->Release();
        VSCore_Term();
        return 0;
}
```

Project should include libstarcore.lib on win32

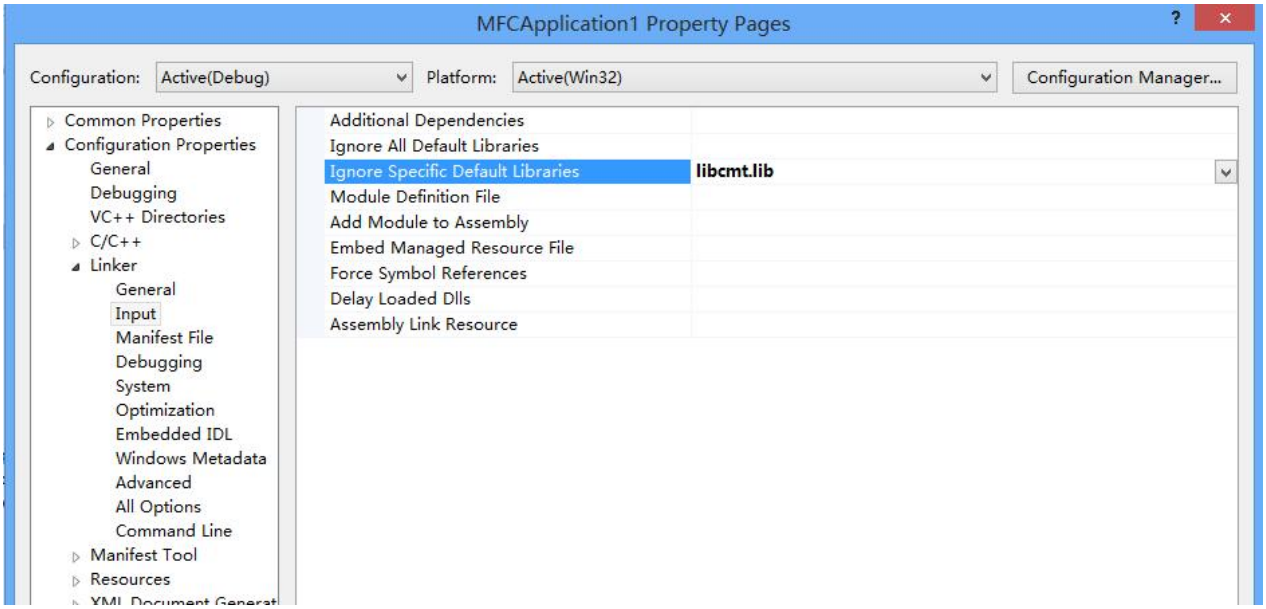
On linux should add library with –lstarcore.

3.2.5 link errors for vsxx

If there are link errors as follows:

```
1>  Generating Code...
1>LIBCMT.lib(sprintf.obj) : error LNK2005: _sprintf already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(invarg.obj) : error LNK2005: __invoke_watson already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(wsetloca.obj) : error LNK2005: __configthreadlocale already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(mlock.obj) : error LNK2005: __lock already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(mlock.obj) : error LNK2005: __unlock already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(crt0dat.obj) : error LNK2005: __amsg_exit already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(crt0dat.obj) : error LNK2005: __cexit already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(crt0dat.obj) : error LNK2005: __exit already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(crt0dat.obj) : error LNK2005: __initterm_e already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(crt0dat.obj) : error LNK2005: _exit already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(winapisupp.obj) : error LNK2005: ___crtGetShowWindowMode already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(winapisupp.obj) : error LNK2005: ___crtSetUnhandledExceptionFilter already defined in
msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(winapisupp.obj) : error LNK2005: ___crtTerminateProcess already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(winapisupp.obj) : error LNK2005: ___crtUnhandledException already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(winxfltr.obj) : error LNK2005: __XcptFilter already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(printf.obj) : error LNK2005: _printf already defined in msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(hooks.obj) : error LNK2005: "void __cdecl terminate(void)" (?terminate@@YAXXZ) already defined in
msvcrtd.lib(MSVCR110D.dll)
1>LIBCMT.lib(crt0init.obj) : error LNK2005: ___xi_a already defined in msvcrtd.lib(cinitexe.obj)
1>LIBCMT.lib(crt0init.obj) : error LNK2005: ___xi_z already defined in msvcrtd.lib(cinitexe.obj)
1>LIBCMT.lib(crt0init.obj) : error LNK2005: ___xc_a already defined in msvcrtd.lib(cinitexe.obj)
1>LIBCMT.lib(crt0init.obj) : error LNK2005: ___xc_z already defined in msvcrtd.lib(cinitexe.obj)
1>LIBCMT.lib(errmode.obj) : error LNK2005: ___set_app_type already defined in msvcrtd.lib(MSVCR110D.dll)
```

libcmt.lib must be ignored.



3.3 Init Cle using lua

3.3.1 init type 1, create service group directly

```
require "libstarcore"
SrvGroup=libstarcore._InitSimpleEx(0,0)
SrvGroup:_CreateService( "","" test", "123",5,0,0,0,0,0,"" )
Service = SrvGroup:_GetService("root","123")
…
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

3.3.2 init type 2, create service directly

```
require "libstarcore"
Service=libstarcore._InitSimple("test", "123",0,0)
..
Service._ServiceGroup:_ClearService()
libstarcore._ModuleExit()
```

3.3.3 init type 3, create service step by step

```
require "libstarcore"
libstarcore._InitCore(true,true,false,true,"",0,"",0)
SrvGroup = libstarcore:_GetSrvGroup()
```

```
SrvGroup:_CreateService( "","" test", "123",5,0,0,0,0,0,"" )
Service = SrvGroup:_GetService("root","123")
…
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

3.4 Init Cle using python

note : for python 3.3, the module name is libstar_pyhon33, the interface name is python33

3.4.1 init type 1, create service group directly

```
import libstarpy
SrvGroup= libstarpy._InitSimpleEx(0,0)
SrvGroup._CreateService( "","" test", "123",5,0,0,0,0,0,"" )
Service = SrvGroup._GetService("root","123")
…
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

for python3.3

```
import libstar_python33
SrvGroup= libstar_python33._InitSimpleEx(0,0)
SrvGroup._CreateService( "","" test", "123",5,0,0,0,0,0,"" )
Service = SrvGroup._GetService("root","123")
…
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

3.4.2 init type 2, create service directly

```
import libstarpy
Service= libstarpy._InitSimple("test", "123",0,0)
..
Service._ServiceGroup._ClearService()
libstarpy._ModuleExit()
```

3.4.3 init type 3, create service step by step

```
import libstarpy
libstarpy._InitCore(True,True,False,True,"",0,"",0)
SrvGroup = libstarpy._GetSrvGroup()
```

```
SrvGroup._CreateService( "","" test", "123",5,0,0,0,0,0,"" )
Service = SrvGroup._GetService("root","123")
…
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

3.5 Init Cle using ruby

note : libstar_ruby is not installed automatically. You can copy it to ruby ext path. for example, X:\Ruby193\lib\ruby\1.9.1\i386-mingw32

3.5.1 init type 1, create service group directly

```
if (defined? Libstar_ruby) == nil
#   require "D:\\Work\\starcore\\core\\starcore.ruby\\libruby193\\libstar_ruby.so"
  require "libstar_ruby"
end
$SrvGroup= $starruby._InitSimpleEx(0,0)
$SrvGroup._CreateService( "","" test", "123",5,0,0,0,0,0,"" )
$Service = $SrvGroup._GetService("root","123")
…
$SrvGroup._ClearService()
$starruby._ModuleExit()
```

3.5.2 init type 2, create service directly

```
if (defined? Libstar_ruby) == nil
#   require "D:\\Work\\starcore\\core\\starcore.ruby\\libruby193\\libstar_ruby.so"
  require "libstar_ruby"
end
$Service= $starruby._InitSimple("test", "123",0,0)
..
$Service._ServiceGroup._ClearService()
$starruby._ModuleExit()
```

3.5.3 init type 3, create service step by step

```
if (defined? Libstar_ruby) == nil
#   require "D:\\Work\\starcore\\core\\starcore.ruby\\libruby193\\libstar_ruby.so"
  require "libstar_ruby"
end
$starruby._InitCore(true,true,false,true,"",0,"",0)
$SrvGroup = $starruby._GetSrvGroup(0)
```

```
$SrvGroup._CreateService( "","test", "123",5,0,0,0,0,"" )
$Service = $SrvGroup._GetService("root","123")
…
$SrvGroup._ClearService()
$starruby._ModuleExit()
```

3.6 Init Cle using java

3.6.1 init type 1, create service group directly

```
import com.srplab.www.starcore.*;
public class test_server{
	public static void main(String[] args){
		StarCoreFactory starcore=StarCoreFactory.GetFactory();
		StarSrvGroupClass SrvGroup =starcore._InitSimpleEx(0,0);
..
		SrvGroup._ClearService();
		starcore._ModuleExit();
	}
}
```

3.6.2 init type 2, create service directly

```
import com.srplab.www.starcore.*;
public class test_server{
	public static void main(String[] args){
		StarCoreFactory starcore=StarCoreFactory.GetFactory();
		StarServiceClass Service=starcore._InitSimple("test","123",0,0);
		StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

..
		SrvGroup._ClearService();
		starcore._ModuleExit();
	}
}
```

3.6.3 init type 3, create service step by step

```
import com.srplab.www.starcore.*;
public class test_server{
	public static void main(String[] args){
		StarCoreFactory starcore=StarCoreFactory.GetFactory();
```

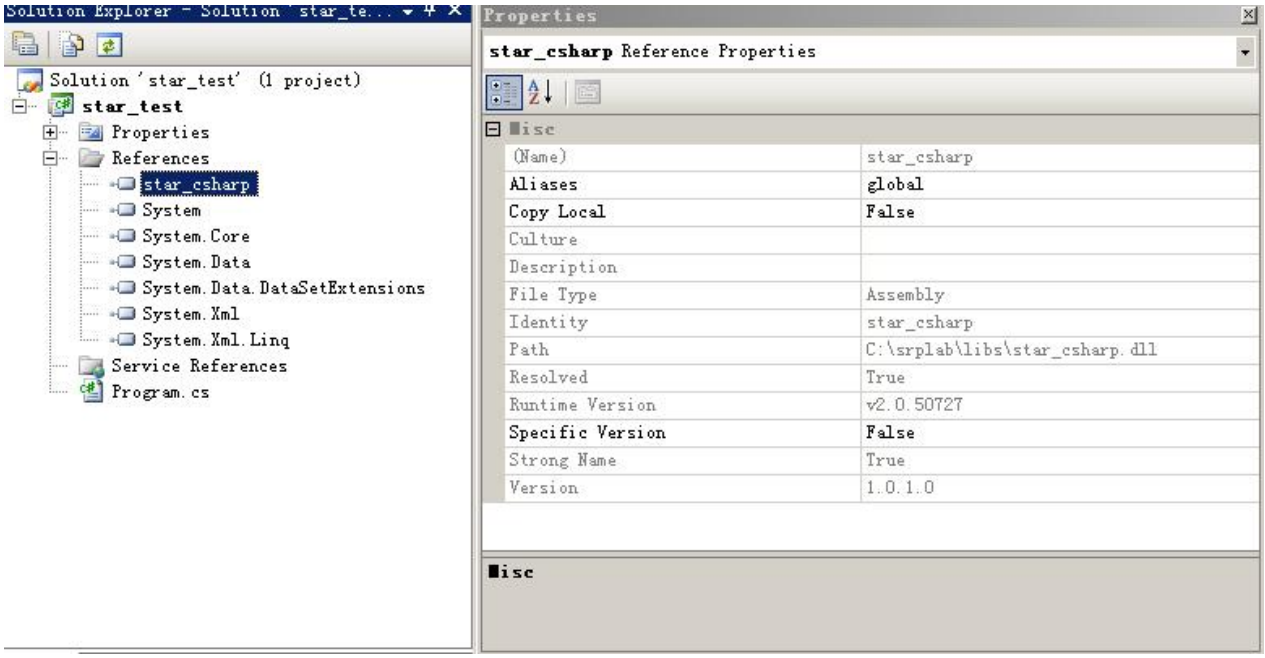
```
            starcore._InitCore(true,true,false,true,"",0,"",0);
            SrvGroup = starcore._GetSrvGroup();
            SrvGroup._CreateService( "","" test", "123",5,0,0,0,0,0,"" );
            StarServiceClass Service = SrvGroup._GetService("root","123");
..
            SrvGroup._ClearService();
            starcore._ModuleExit();
        }
}
```

3.7 Init Cle using csharp/csharp4/csharp45/csharp451

Note: for windows phone 8/8.1/windows store, Star_csharp should be used and StarCoreFactoryInit.Init() should be called before GetFactory function.

from windows phone 8.1 or windows store apps, "StarCoreFactoryInit.Init(this);" should be used other than "StarCoreFactoryInit.Init()"

Using C#, should set Perference of the project, as follows:



Star_csharp library is stored in directory c:\srplab\libs.

csharp4 is for .net4.0

csharp45 is for .net45

csharp451 is for .net451

If error "System.BadImageFormatException" occurs, the platform target should be changed to x86 to solve this error.

C# may be used to generate exe file or share library. File name should same with namespace. Cle will search and call Program.Main function. The function may be no argments or take string[] as argument, as follows:

The file name is case sensitive.

```
namespace socketserver
{
   class Program
   {
      static void Main(string[] args)
      {
            …
       }
  }
}
```

```
namespace socketserver
{
   class Program
   {
      static void Main()
      {
            …
       }
  }
}
```

3.7.1 init type 1, create service group directly

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace socketserver
{
   class Program
   {
       static void Main(string[] args){
              StarCoreFactory starcore=StarCoreFactory.GetFactory();
              StarSrvGroupClass SrvGroup =starcore._InitSimpleEx(0,0,null);


..
              SrvGroup._ClearService();
              starcore._ModuleExit();
       }
     }
}
```

3.7.2 init type 2, create service directly

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace socketserver
{
   class Program
   {
       static void Main(string[] args){
              StarCoreFactory starcore=StarCoreFactory.GetFactory();
              StarServiceClass Service=starcore._InitSimple("test","123",0,0,null);
              StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");


..
              SrvGroup._ClearService();
              starcore._ModuleExit();
```

```
        }
    }
}
```

3.7.3 init type 3, create service step by step

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace socketserver
{
   class Program
   {
       static void Main(String[] args){
                StarCoreFactory starcore=StarCoreFactory.GetFactory();
                starcore._InitCore(true,true,false,true,"",0,"",0);
                SrvGroup = starcore._GetSrvGroup();
                SrvGroup._CreateService( "","" test", "123",5,0,0,0,0,0,"" );
                StarServiceClass Service = SrvGroup._GetService("root","123");

..
                SrvGroup._ClearService();
                starcore._ModuleExit();
        }
     }
}
```

3.7.4 compile using command line

```
csc /reference:c:\srplab\libs\Star_csharp.dll /platform:x86 XXXX.cs
```

3.8 Create object, define it's attributes and functions

3.8.1 python

Method 1:

```
Object = Service._New("Test")   // Create object named Test
```

```
Object.Attr = 123;                  //  Define attribute Attr
def Object_Func(self,Para) :
      print(self, Para);
Object.Func = Object_Func;   //  Define function Func
```

Method 2:

```
MyObj = Service._New("Test")   // Create object named Test
MyObj.Attr = 123;                    //  Define attribute Attr
def InitCleObject(which) :
   def a_Func( cleobj, para ) :
      print( cleobj,para)
      return
   end
   which.Func = a_Func
end
InitCleObject(MyObj)
```

Method 3

```
class TestClass :
   def __init__(self) :
      self.IntValue = 1
      self.CharValue = "hello from Test"

   def Add(self,f1,f2) :
      return f1 + f2
obj2 = Service._New("Test2")
obj2._AttachRawObject(TestClass(),False)

print(obj2.Add(12,34))
print(obj2.IntValue)
print(obj2.CharValue)
```

3.8.2 lua

```
Object = Service:_New("Test")   // Create object named Test
Object.Attr = 123;                    //  Define attribute Attr
function Object:Func(Para)        //  Define function Func
      print(self, Para);
end
```

3.8.3 ruby

Method 1:

```
Object = Service._New("Test")   // Create object named Test
Object.Attr = 123;              //  Define attribute Attr
def a_Func( cleobj, para )      //  Define function Func
   puts(cleobj,para)
end
a. Func = method(:a_Func)
```

Method 2:

```
Object = Service._New("Test")   // Create object named Test
Object.Attr = 123;              //  Define attribute Attr
obj._RegScriptProc_P(Func) {|cleobj, Para | puts(cleobj, Para) }        //  Define function Func
```

Method 3:

```
MyObj = Service._New("Test")   // Create object named Test
MyObj.Attr = 123;              //  Define attribute Attr
def InitCleObject(which)
   def a_Func( cleobj, para )
      puts( cleobj,para)
      return
   end
   which.Func = method(:a_Func)
end
InitCleObject(MyObj)
```

Method 4:

```
class TestClass
   attr_accessor :IntValue
   attr_accessor :CharValue
   def initialize()
      @IntValue = 1
      @CharValue = "hello from Test"
   end
   def Add(f1,f2)
      return f1+f2
   end
end
obj2 = Service._New("Test2")
obj2._AttachRawObject(TestClass.new(),false)

puts(obj2.Add(12,34))
puts(obj2.IntValue)
puts(obj2.CharValue)
```

3.8.4 java

Method 1:

```
class MyObjectClass extends StarObjectClass{
  public void Func ( StarObjectClass self,int para )        //  Define function Func
  {
    System.out.println(self);
    System.out.println(para);
  }
}
public class XXXXX{
        public static void main(String[] args){
          …..
          StarObjectClass Object = new MyObjectClass(Service._New ("Test"));  // Create object named Test
          ……
       }
}
```

Method 2:

```
public class XXXXX{
        public static void main(String[] args){
          …..
          StarObjectClass MyObj = Service._New("Test");
          MyObj._RegScriptProc_P("FuncName", new StarObjectScriptProcInterface() {
                    public Object Invoke(Object CleObject, Object[] EventParas) {
                       return null;
                    }
                });
          ……
       }
}
```

Method 3:

```
class TestClass{
   public int IntValue;
   public String CharValue;
   public TestClass(){
      IntValue = 1;
      CharValue = "hello from Test";
   }
   public double Add(double f1, double f2 ){
      return f1+f2;
```

```
    }
}

    StarObjectClass obj2=Service._New("Test");
    obj2._AttachRawObject(new TestClass(),false);
    System.out.println(obj2._Call("Add",12,34));
    System.out.println(obj2._Get("IntValue"));
    System.out.println(obj2._Get("CharValue"));
```

3.8.5 c#

Method 1:

```
class MyObjectClass : StarObjectClass{
  public void Func ( StarObjectClass self,int para )        //  Define function Func
  {
    Console.WriteLine (self);
    Console.WriteLine (para);
  }
}
namespace XXXXX
{
   class Program
   {
       public static void main(String[] args){
          …..
          StarObjectClass Object = new MyObjectClass(Service._New ("Test"));  // Create object named Test
          ……
       }
   }
}
```

Method 2:

```
namespace XXXXX
{
   class Program
   {
       public static void main(String[] args){
          …..
          StarObjectClass MyObj = Service._New ("Test");  // Create object named Test
          MyObj._RegScriptProc_P("FuncName", delegate (Object CleObject, object[] EventParas){
                    return null;
              });
```

```
        ……
      }
   }
}
```

Method 3:

```
class TestClass
{
   public int IntValue;
   public String CharValue;
   public TestClass()
   {
      IntValue = 1;
      CharValue = "hello from Test";
   }
   public double Add(double f1, double f2)
   {
      return f1 + f2;
   }
}

      StarObjectClass obj2=Service._New("Test");
      obj2._AttachRawObject(new TestClass(),false);
      Console.WriteLine(obj2._Call("Add",12,34));
      Console.WriteLine(obj2._Get("IntValue"));
      Console.WriteLine(obj2._Get("CharValue"));
```

3.8.6 c++

```
static void Func(void *Object,VS_INT32 Para)
{
   printf( "%d\n",Para);
}

class ClassOfSRPInterface *SRPInterface;

int main(int argc, char* argv[])
{
   VS_CORESIMPLECONTEXT Context;
   void *AtomicClass,*Func_AtomicFunction;

   SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
      AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","Test","VS_INT32 Attr",NULL,NULL);    // Create
object named Test    Define attribute Attr
```

```
	Func_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"Func","void Func(VS_INT32
Para);",NULL,NULL,VS_FALSE,VS_FALSE);	// Define function Func
	SRPInterface -> SetAtomicFunction(Func_AtomicFunction,(void *)Func);	// Set function address

	....
}
```

Method 2:

```
class ClassOfTest{
public :
   VS_INT32 IntValue;
   VS_CHAR  CharValue[256];

public:
   ClassOfTest();
   ~ClassOfTest();

   VS_DOUBLE Add(VS_DOUBLE f1,VS_DOUBLE f2);
};

ClassOfTest::ClassOfTest()
{
   IntValue = 1;
   strcpy(CharValue,"hello from Test");
}

ClassOfTest::~ClassOfTest()
{

}

VS_DOUBLE ClassOfTest::Add(VS_DOUBLE f1,VS_DOUBLE f2)
{
   return f1 + f2;
}

static VS_INT32 SRPAPI TestClass_Obj_ScriptCallBack( void *L );
static VS_BOOL SRPAPI TestClass_Obj_LuaFuncFilter(void *Object,void *ForWhichObject,VS_CHAR
*FuncName,VS_UWORD Para);
static VS_BOOL SRPAPI TestClass_Obj_RegGetValue(void *Object,void *ForWhichObject,VS_CHAR *Name,VS_UWORD
Para,VS_BOOL GetAllRawAttributeFlag);
static VS_BOOL SRPAPI TestClass_Obj_RegSetValue(void *Object,void *ForWhichObject,VS_CHAR *Name,VS_INT32
Index,VS_UWORD Para);
```

```
struct StructOfTestClassLocalBuf{
   ClassOfTest *testobject;
};
      TestClassLocalBuf = (struct StructOfTestClassLocalBuf *)SRPInterface -> MallocPrivateBuf( Object, SRPInterface ->
GetLayer(Object),0,sizeof(struct StructOfTestClassLocalBuf) );
      vs_memset(TestClassLocalBuf,0,sizeof(struct StructOfTestClassLocalBuf));

      TestClassLocalBuf ->testobject = new ClassOfTest();
      SRPInterface -> RegLuaFunc( Object, NULL, (void*)TestClass_Obj_ScriptCallBack, (VS_UWORD)0 );
      SRPInterface -> RegLuaFuncFilter(Object,TestClass_Obj_LuaFuncFilter,(VS_UWORD)0);
      SRPInterface ->RegLuaGetValueFunc(Object,TestClass_Obj_RegGetValue,(VS_UWORD)0 );
      SRPInterface ->RegLuaSetValueFunc(Object,TestClass_Obj_RegSetValue,(VS_UWORD)0 );

static VS_BOOL TestClass_Obj_RegGetValue(void *Object,void *ForWhichObject,VS_CHAR *Name,VS_UWORD
Para,VS_BOOL GetAllRawAttributeFlag)
{
   struct StructOfTestClassLocalBuf  *TestClassLocalBuf;

   TestClassLocalBuf = (struct StructOfTestClassLocalBuf *)SRPInterface -> GetPrivateBuf( Object, SRPInterface ->
GetLayer(Object),0, NULL );
   if( strcmp(Name,"IntValue") == 0 ){
      SRPInterface ->LuaPushInt( TestClassLocalBuf ->testobject->IntValue );
      return VS_TRUE;
   }
   if( strcmp(Name,"CharValue") == 0 ){
      SRPInterface ->LuaPushString( TestClassLocalBuf ->testobject->CharValue );
      return VS_TRUE;
   }
   return VS_FALSE;
}
static VS_BOOL SRPAPI TestClass_Obj_RegSetValue(void *Object,void *ForWhichObject,VS_CHAR *Name,VS_INT32
Index,VS_UWORD Para)
{
   struct StructOfTestClassLocalBuf  *TestClassLocalBuf;

   TestClassLocalBuf = (struct StructOfTestClassLocalBuf *)SRPInterface -> GetPrivateBuf( Object, SRPInterface ->
GetLayer(Object),0, NULL );
   if( strcmp(Name,"IntValue") == 0 ){
      TestClassLocalBuf ->testobject->IntValue = SRPInterface ->LuaToInt(Index);
      return VS_TRUE;
   }else if( strcmp(Name,"CharValue") == 0 ){
      VS_CHAR *CharPtr = SRPInterface ->LuaToString(Index);
      if( CharPtr == NULL )
         TestClassLocalBuf ->testobject->CharValue[0] = 0;
      else
```

```
            strcpy( TestClassLocalBuf ->testobject->CharValue, CharPtr);
        return VS_TRUE;
    }
    return VS_FALSE;
}

static VS_BOOL SRPAPI TestClass_Obj_LuaFuncFilter(void *Object,void *ForWhichObject,VS_CHAR
*FuncName,VS_UWORD Para)
{
    if( strcmp(FuncName,"Add") == 0 )
        return VS_TRUE;
    return VS_FALSE;
}
static VS_INT32 TestClass_Obj_ScriptCallBack( void *L )
{
    struct StructOfTestClassLocalBuf  *TestClassLocalBuf;
    void *Object;
    VS_CHAR *ScriptName;

    ScriptName = SRPInterface -> LuaToString( SRPInterface -> LuaUpValueIndex(3) );
    Object = SRPInterface -> LuaToObject(1);
    /*first input parameter is started at index 2 */
    TestClassLocalBuf = (struct StructOfTestClassLocalBuf *)SRPInterface -> GetPrivateBuf( Object, SRPInterface ->
GetLayer(Object),0, NULL );
    if( strcmp(ScriptName,"Add") == 0 ){
        VS_DOUBLE f1 = SRPInterface ->LuaToNumber(2);
        VS_DOUBLE f2 = SRPInterface ->LuaToNumber(3);
        SRPInterface ->LuaPushNumber( TestClassLocalBuf ->testobject->Add(f1,f2) );
        return 1;
    }
    return 0;
}
```

3.8.7 Call object's function

3.8.7.1 python

```
Object = Service.Test._New()
a =  Object.Attr       //--Get object's attribute
Object.Func(123);   //--Call object's function
Object._Free();
```

3.8.7.2 lua

```
Object = Service.Test:_New()
a =  Object.Attr       //--Get object's attribute
Object:Func(123);   //--Call object's function
Object:_Free();
```

3.8.7.3 java

```
StarObjectClass Obj = Service._GetObject("Test")._New();
int a =  Obj._Get("Attr");       //--Get object's attribute
Obj._Call("Func",123);          //--Call object's function
Object._Free();
```

3.8.7.4 c#

```
StarObjectClass Obj = Service._GetObject("Test")._New();
int a =  Obj._Get("Attr");       //--Get object's attribute
Obj._Call("Func",123);          //--Call object's function
Object._Free();
```

3.8.7.5 c++

```
int main(int argc, char* argv[])
{
        void *Obj;
        VS_INT32 i;


   ......
        Class = SRPInterface ->GetObjectEx(NULL,"Test");
        Obj = SRPInterface -> MallocObjectL(SRPInterface->GetIDEx(Class),NULL,NULL);
        i = SRPInterface -> ScriptGetInt(Obj,"Attr");                    //--Get object's attribute
        SRPInterface -> ScriptCall(Obj,NULL,"Func","(i)",123);     //--Call object's function
        SRPInterface ->FreeObject(Obj);
        ....
}
```

3.9 Message loop in CLE

CLE is driven by message. For C++ language, interface ClassOfSRPControlInterface provides function "SRPDispatch". Each call to the function, one message in the queue of CLE will be processed.

In script language, additional function _MsgLoop is provided, which will continue to dispatch message until callback function returns true. Script interface also provides function _SRPDispatch, which may be used to dispatch message same as SRPDispatch.

3.9.1 c#/java

For Form application, a timer(10ms) should be created to drive the CLE, as follows:

```
…
using Star_csharp;


public partial class Form1 : Form
{
    public static StarCoreFactory a;
    public Form1()
    {
        InitializeComponent();
        a = null;
        timer1.Enabled = true;
    }
    private void Form1_Shown(object sender, EventArgs e)
    {

        a = StarCoreFactory.GetFactory();
         …
    }
    private void timer1_Tick(object sender, EventArgs e)
    {
        if (a != null)
        {
            while (a._SRPDispatch(false)) ;
        }
    }
}
```

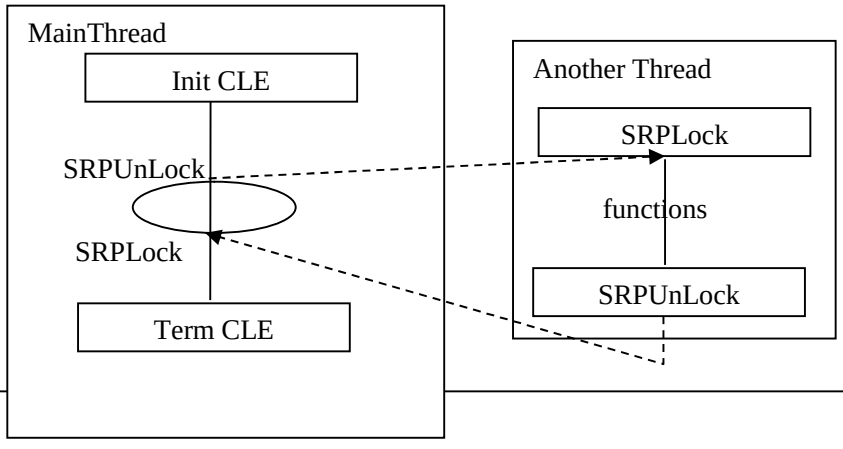
3.9.2 android

```
public class Test_serverActivity extends Activity {
   /** Called when the activity is first created. */
      StarCoreFactory starcore;
      StarSrvGroupClass SrvGroup;
```

```
    Timer timer;
    ----
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    --------
    timer = new Timer();

    final Handler handler = new Handler()
    {
      @Override
      public void handleMessage(Message msg)
      {
        while( starcore._SRPDispatch(false) == true );
      }
    };
    timer.scheduleAtFixedRate(new TimerTask()
    {
      @Override
      public void run()
      {
        Message message = handler.obtainMessage(); //handler is an instance of type Handler
        message.what = 0;
        message.sendToTarget();
      }
    }, 0, 10);
}
```

3.10 Global Lock

CLE maintains a global lock. It's status is locked after cle init.



Programmer may call SRPLock and SRPUnLock function to change its status. The global lock is important for multiple threads applications.

3.11 Problems that need attention

In the same process, load multiple instance of cle is not safe. This is because CLE supports many languages, which may not completely support multi-instance in the same process. Especially python, which is hard to unload completely after loaded.

For C/C++, if you want to load multiple instance of CLE, you should call function starlib_dll_open_starcore, which will check whether libstarcore share library is loaded or not. If the share library has been loaded, then the function will create a copy of libstarcore, and load it. CLE will try to support multiple instance, but for complicated environment, the effort does not always take effect.

For java, if package starcore.jar is not loaded into current ClassLoader, then CLE will load it into SystemClassLoader. At this time, if SystemClassLoader exists some limitions for security, the loading may be failed.

For c#, if you want to use CLE in multiple AppDomains, you should provide function which is a complete CLE procedure as follows:

```
{
   StarCoreFactory starcore = StarCoreFactory.GetFactory();
   StarServiceClass Service = starcore._InitSimple("test", "123", 0, 0, null);
   ….
   SrvGroup._ClearService();
   starcore._ModuleClear();  // use this function to clear CLE objects
}
```

An example with multiple appdomains is located in directory examples\cle.advanced\ csharp.appdomain

For python and lua, all global variables share the same script space. Therefore, variables with same name with replace the existing one, which should be careful.

In loop procedure, applications should call SRPDispatch function to consume internally generated message of cle.

For C++, set value of VS_STRING attributes of object should use the following two functions:

```
void SRPAPI SetVString(VS_VSTRING *Buf,VS_CHAR *Str);
VS_VSTRING *SRPAPI ToVString(VS_CHAR *Str);
```

For examples:

```
struct ParaClass{
    VS_VSTRING Para5;
};
ParaObj->Para5 = (*SRPInterface->ToVString("From caller"));   or
SRPInterface->SetVString(&ParaObj->Para5, "From caller");
```

3.12 CLE Environments

3.12.1 SRPHOME

This environment variable is used by starcore to set path for temp files or files downloaded from network.

3.12.2 SRPMODULE

This variable is used by script bridge to determine the name of share library name of starcore.
SRPMODULE = libstarcore

3.13 language Locale

for ios, android, and wp, default lang of cle is utf-8
for windows and linux, default is same with system
you can use _SetLocale and _GetLocale method to change default settings.
And use _ToAnsi and _FromAnsi method to change string of special charset to string of cle

3.14 notes for android, ios, wp, winrt and windows 10

3.14.1 android

cle for android has an additional class named "StarCoreFactoryPath", which has three static members :

```
    public static String StarCoreShareLibraryPath = null;
    public static String StarCoreCoreLibraryPath = null;
    public static String StarCoreOperationPath = null;
```

These three members are valid before call StarCoreFactory.GetFactory() function.
StarCoreCoreLibraryPath is the path for starcore share library path, if is not set, then these library should be located in /data/data/com.srplab.starcore/lib directory. If you add starcore library into the project, then you must change it to this.getApplicationInfo().nativeLibraryDir

starcore share library includes libstarcore.so, libstarpy.so, libstar_java.so

StarCoreShare libraryraryPath is the path for other share library.

StarCoreOperationPath is direcotry for srplab writeable path. If it is not set, then use /sdcard/srplab.
If it is set, it may be set to /data/data/packagename/files.
an example :

```
StarCoreFactoryPath.StarCoreCoreLibraryPath = this.getApplicationInfo().nativeLibraryDir;
StarCoreFactoryPath.StarCoreShareLibraryPath = this.getApplicationInfo().nativeLibraryDir;
StarCoreFactoryPath.StarCoreOperationPath = "/data/data/"+getPackageName()+"/files";
 StarCoreFactory starcore= StarCoreFactory.GetFactory();
```

StarCoreOperationPath has a function named "Install", which can be used to unzip file in assets into a directory on target. Its prototype is :

```
public static boolean Install(InputStream zipFileName, String outputDirectory,Boolean OverWriteFlag )
```

for example :

```
try{
 AssetManager assetManager = getAssets();
 InputStream dataSource = assetManager.open("returnraw.zip");
  StarCoreFactoryPath.CreatePath(Runtime.getRuntime(),"/data/data/"+getPackageName()+"/files/SRPFSEngine");
 StarCoreFactoryPath.Install(dataSource, "/data/data/"+getPackageName()+"/files",true );
}
catch(IOException e ){
}
```

StarCoreOperationPath has a function named " CreatePath", which can be used to create directory. Its prototype is :

```
public static boolean CreatePath(Runtime runtime,String Path)
```

for example :

```
StarCoreFactoryPath.CreatePath(Runtime.getRuntime(),"/data/data/"+getPackageName()+"/files/SRPFSEngine");
```

use the following code to set python module search patch :

```
SrvGroup._InitRaw("python",Service);
StarObjectClass python = Service._ImportRawContext("python","",false,"");
python._Call("import", "sys");
StarObjectClass pythonSys = python._GetObject("sys");
StarObjectClass pythonPath = (StarObjectClass)pythonSys._Get("path");
pythonPath._Call("insert",0,"/data/data/"+getPackageName()+"/files/python27.zip");
```

```
pythonPath._Call("insert",0,"/data/data/"+getPackageName()+"/lib");
pythonPath._Call("insert",0,"/data/data/"+getPackageName()+"/files");
```

3.14.2 using ruby on android

Ruby is supported on android, and some modules had been compile for android. Copy libstar_ruby.so and libruby.so to directory libs of the project. Copy rubylib_armeabi_r193.zip to dirctory assets.
Using the following code to extract rubylib_armeabi_r193.zip to files directory,

```
try{
 AssetManager assetManager = getAssets();

 dataSource = assetManager.open("rubylib_armeabi_r193.zip ");
 StarCoreFactoryPath.CreatePath(Runtime.getRuntime(),"/data/data/"+getPackageName()+"/files/ruby");
 StarCoreFactoryPath.Install(dataSource, "/data/data/"+getPackageName()+"/files/ruby",true );
}
catch(IOException e ){
}
```

Before init ruby, set the library path for ruby

```
starcore._SetScript("ruby","","-p
"+"/data/data/"+getPackageName()+"/files/ruby/lib;"+"/data/data/"+getPackageName()+"/files/ruby/lib/arm-linux");
```

The following code is to call raw ruby file testrb.rb

```
SrvGroup._InitRaw("ruby",Service);
SrvGroup._LoadRawModule("ruby","","/data/data/"+getPackageName()+"/files/testrb.rb",false);
StarObjectClass ruby = Service._ImportRawContext("ruby","",false,"");
```

use the following code to set ruby module search patch :

```
SrvGroup._InitRaw("ruby",Service);
StarObjectClass ruby = Service._ImportRawContext("ruby","",false,"");
StarObjectClass LOAD_PATH = (StarObjectClass)ruby._R("LOAD_PATH");
LOAD_PATH._Call("unshift", "/data/data/"+getPackageName()+"/files");
```

3.14.3 using cle in native app

In native app, core share library path should be set when init cle, using "VS_STARCONFIGEX" as follow,

```
VS_CORESIMPLECONTEXT Context;
class ClassOfSRPInterface *SRPInterface;
```

```
VS_STARCONFIGEX CleConfig;

memset(&CleConfig, 0, sizeof(CleConfig));
sprintf(CleConfig.CoreLibraryPath, "/data/data/com.cle_testandroid/lib");
SRPInterface = VSCore_InitSimpleWithCfg(&Context, &CleConfig,"test", "123", 0, 0, NULL, 0, NULL);
……

VSCore_TermSimple(&Context);
```

or,

```
VS_HANDLE hDllInstance;
VSCore_InitProc VSInitProc;
VSCore_TermProc VSTermProc;
VSCore_QueryControlInterfaceProc QueryControlInterfaceProc;
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;

VS_CHAR ModuleName[512];

SRPControlInterface = NULL;
BasicSRPInterface = NULL;
sprintf(ModuleName, "/data/data/com.cle_testandroid/lib/libstarcore%s", VS_MODULEEXT);
hDllInstance = vs_dll_open(ModuleName);
if (hDllInstance == NULL) {
        printf("load library [%s] error....\n", ModuleName);
        return;
}
VSInitProc = (VSCore_InitProc)vs_dll_sym(hDllInstance, VSCORE_INIT_NAME);
VSTermProc = (VSCore_TermProc)vs_dll_sym(hDllInstance, VSCORE_TERM_NAME);
QueryControlInterfaceProc = (VSCore_QueryControlInterfaceProc)vs_dll_sym(hDllInstance,
VSCORE_QUERYCONTROLINTERFACE_NAME);

VS_STARCONFIGEX CleConfig;

memset(&CleConfig, 0, sizeof(CleConfig));
sprintf(CleConfig.CoreLibraryPath, "/data/data/com.cle_testandroid/lib");
VSInitProc(true, true, "", 0, "", 0, &CleConfig);

printf("init starcore success\n");
SRPControlInterface = QueryControlInterfaceProc();
BasicSRPInterface = SRPControlInterface->QueryBasicInterface(0);
BasicSRPInterface->CreateService("", "test", NULL, "123", 0, 0, 0, 0, 0, 0);

class ClassOfSRPInterface *SRPInterface;
```

```
SRPInterface = BasicSRPInterface->GetSRPInterface("test", "root", "123");

....

SRPControlInterface->Release();
BasicSRPInterface->Release();
VSTermProc();
vs_dll_close(hDllInstance);
```

The app must linked with "libstarlib.a"

3.14.4 ios

Using the following code to init cle for ios

```
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,   NSUserDomainMask, YES);
NSString *documentsDirectory = [paths objectAtIndex:0];
const char* destDir = [documentsDirectory UTF8String];
NSString *respaths = [[NSBundle mainBundle] resourcePath];
const VS_CHAR *res_cpath = [respaths UTF8String];
VS_BOOL Result = StarCore_InitEx((VS_CHAR *)destDir,(VS_CHAR *)res_cpath);

VS_CORESIMPLECONTEXT Context;
SRPInterface = VSCoreLib_InitSimple(&Context,"test","123",0,0,NULL,0,,NULL);
```

If python is used on ios, then uses the following code to init cle

```
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,   NSUserDomainMask, YES);
NSString *documentsDirectory = [paths objectAtIndex:0];
const char* destDir = [documentsDirectory UTF8String];
NSString *respaths = [[NSBundle mainBundle] resourcePath];
const VS_CHAR *res_cpath = [respaths UTF8String];
VS_BOOL Result = StarCore_InitEx((VS_CHAR *)destDir,(VS_CHAR *)res_cpath);

VS_CHAR python_path[512];
VS_CHAR python_home[512];
sprintf(python_home,"%s",res_cpath);
sprintf(python_path,"%s",res_cpath);
VSCoreLib_InitPython((VS_CHAR*)python_home,(VS_CHAR *)python_path,NULL);

VS_CORESIMPLECONTEXT Context;
SRPInterface = VSCoreLib_InitSimple(&Context,"test","123",0,0,NULL,0,,NULL);
```

If python has no site, add following code before calling any python function.

```
Context.VSControlInterface -> SetScriptInterface("python","","-S -d");
```

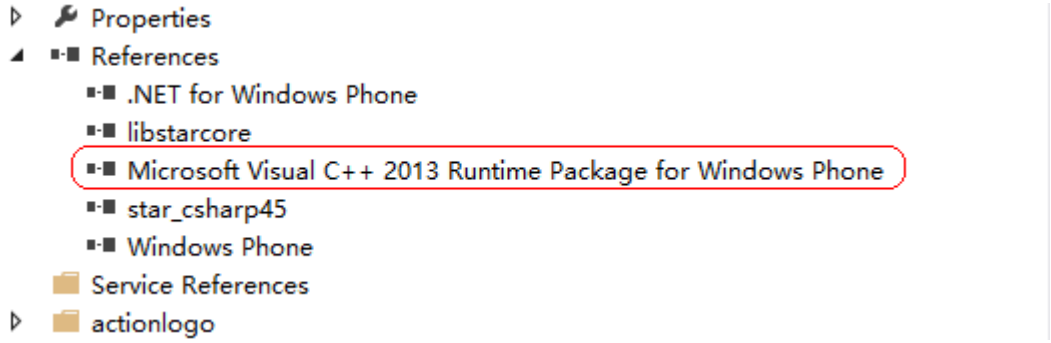
3.14.5 wp or windows store or windows 10

On windows phone, c# language is used. In this case, the following code should be called before GetFactory.

```
StarCoreFactoryInit.Init(this);  // for windows phone 8.1 or windows store 8.1, or windows 10
//StarCoreFactoryInit.Init();  // for windows phone 8.0

StarCoreFactory starcore = StarCoreFactory.GetFactory();
StarServiceClass Service = (StarServiceClass)starcore._InitSimple("test", "123", 0, 0,null);
```

Note for windows phone 8.1

Add reference "Libstarcore" and "star_csharp45", and "Microsoft Visusl C++ 2013 Runtime Package" to the project.



If c# class is exposed to other languages such as lua, the _InjectClass function should be called. For example,

```
starcore._InjectClass("System.Windows.MessageBoxButton", typeof(System.Windows.MessageBoxButton));
starcore._InjectClass("System.Windows.MessageBox", typeof(System.Windows.MessageBox));
```

and then,

```
SrvGroup=_GetSrvGroup(0)
Service=SrvGroup:_GetService("","")
MessageBoxButton=Service:_ImportRawContext("csharp45","System.Windows.MessageBoxButton",true,"")
MessageBox=Service:_ImportRawContext("csharp45","System.Windows.MessageBox",true,"")
MessageBox:Show("eeee","eeee",MessageBoxButton.OK)
```

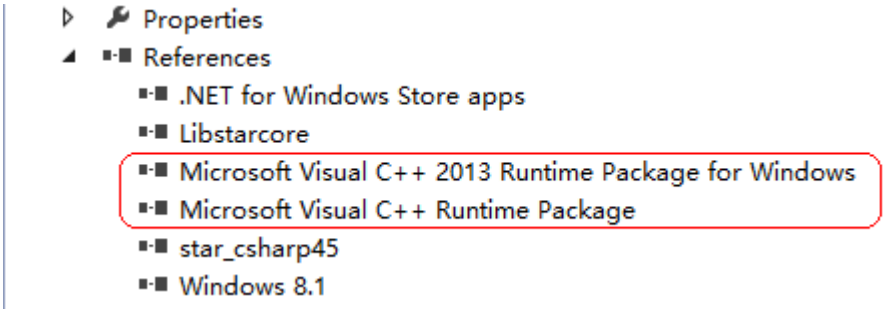
3.14.6 winrt

For windows store app, c# language is used. In this case, the following code should be called before GetFactory.

```
StarCoreFactoryInit.Init(this);

StarCoreFactory starcore = StarCoreFactory.GetFactory();
```

```
StarServiceClass Service = (StarServiceClass)starcore._InitSimple("test", "123", 0, 0,null);
```

Add reference "Libstarcore" and "star_csharp45", and "Microsoft Visusl C++ 2013 Runtime Package" to the project.



If c# class is exposed to other languages such as lua, the _InjectClass function should be called. For example,

```
starcore._InjectClass("System.Windows.MessageBoxButton", typeof(System.Windows.MessageBoxButton));
starcore._InjectClass("System.Windows.MessageBox", typeof(System.Windows.MessageBox));
```

and then,

```
SrvGroup=_GetSrvGroup(0)
Service=SrvGroup:_GetService("","")
MessageBoxButton=Service:_ImportRawContext("csharp45","System.Windows.MessageBoxButton",true,"")
MessageBox=Service:_ImportRawContext("csharp45","System.Windows.MessageBox",true,"")
MessageBox:Show("eeee","eeee",MessageBoxButton.OK)
```

3.14.7 win10

For windows 10 app, c# language is used. In this case, the following code should be called before GetFactory.

```
StarCoreFactoryInit.Init(this);

StarCoreFactory starcore = StarCoreFactory.GetFactory();
StarServiceClass Service = (StarServiceClass)starcore._InitSimple("test", "123", 0, 0,null);
```

Add reference "Libstarcore" and "Star_csharp"

4 Capture output of CLE or other scripts

Register callback function , apps can capture output of cle or other scripts.

4.1 c/c++

```
static VS_UWORD MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_UWORD wParam, VS_UWORD
lParam, VS_BOOL *IsProcessed, VS_UWORD Para )
{
   switch( uMsg ){
    case MSG_VSDISPMSG :
    case MSG_VSDISPLUAMSG :
          printf("[core]%s\n",(VS_CHAR *)wParam);
          break;
    case MSG_DISPMSG :
    case MSG_DISPLUAMSG :
          printf("%s\n",(VS_CHAR *)wParam);
          break;
   }
   return 0;
}

int main(int argc, char* argv[])
{
    class ClassOfSRPInterface *SRPInterface;
    class ClassOfBasicSRPInterface *BasicSRPInterface;
    VS_CORESIMPLECONTEXT Context;

    SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,NULL);
    ...
}
```

4.2 java

```
final StarCoreFactory starcore= StarCoreFactory.GetFactory();
starcore._RegMsgCallBack_P(new StarMsgCallBackInterface()
{
   public Object Invoke( int ServiceGroupID, int uMes, Object wParam, Object lParam)
   {
      if(uMes == starcore._GetInt("MSG_VSDISPMSG") || uMes == starcore._GetInt("MSG_VSDISPLUAMSG"))
      {
         System.out.println((String)wParam);
      }
      if (uMes == starcore._GetInt("MSG_DISPMSG") || uMes == starcore._GetInt("MSG_DISPLUAMSG"))
      {
         System.out.println("+++++++++++++++++" + (String)wParam);
      }
      return null;
```

```
    }
});
StarServiceClass Service=starcore._InitSimple("test","123",0,0);
StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");
```

4.3 csharp

```
starcore = StarCoreFactory.GetFactory();
starcore._RegMsgCallBack_P((int ServiceGroupID, int uMes, Object wParam, Object lParam) =>
{
    if (uMes == starcore._Getint("MSG_DISPMSG") || uMes == starcore._Getint("MSG_DISPLUAMSG") ||
        uMes == starcore._Getint("MSG_VSDISPMSG") || uMes == starcore._Getint("MSG_VSDISPLUAMSG"))
    {
        Debug.WriteLine((String)wParam);
    }
    return null;
 });
```

4.4 lua

```
Service=libstarcore._InitSimple("test","123",0,0,nil);
function MsgCallBack( ServiceGroupID, uMes, wParam, lParam )
    if( uMes == MSG_VSDISPMSG or uMes == MSG_VSDISPLUAMSG ) then
        print(wParam)
    end
    if( uMes == MSG_DISPMSG or uMes == MSG_DISPLUAMSG ) then
        -- print(wParam)  this will cause dead loop
    end
    return false
end
libstarcore._RegMsgCallBack_P(MsgCallBack)
SrvGroup = Service._ServiceGroup;
```

4.5 python

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0);
def MsgCallBack( ServiceGroupID, uMes, wParam, lParam ) :
    if( uMes == libstarpy.MSG_VSDISPMSG or uMes == libstarpy.MSG_VSDISPLUAMSG ) :
        print(wParam)
    if( uMes == libstarpy.MSG_DISPMSG or uMes == libstarpy.MSG_DISPLUAMSG ) :
        # print(wParam)  #this will cause dead loop
        pass
    return False
```

```
libstarpy._RegMsgCallBack_P(MsgCallBack)

SrvGroup = Service._ServiceGroup;
```

4.6 ruby

```
$Service=$starruby._InitSimple("test","123",0,0);
$starruby._RegMsgCallBack_P{ |serviceGroupID, uMes, wParam, lParam|
    if( uMes == $starruby.MSG_VSDISPMSG || uMes == $starruby.MSG_VSDISPLUAMSG )
      puts(wParam)
    end
    if( uMes == $starruby.MSG_DISPMSG || uMes == $starruby.MSG_DISPLUAMSG )
      puts(wParam)
    end
    false
}
$SrvGroup = $Service._ServiceGroup;
```

cle does not capture puts/print function of ruby.

5 C Interface

For version 2.1.0, cle supports plain c interface. The c functions correspond to c++ functions are defined in "vsopenapi_c.h".

For ClassOfSRPCommInterface, the c functions begin with prefix "SRPComm_";

For ClassOfSRPSXMLInterface, the c functions begin with prefix "SRPSXML_";

For ClassOfSRPControlInterface, the c functions begin with prefix "SRPControl_";

For ClassOfBasicSRPInterface, the c functions begin with prefix "SRPBasic_";

For ClassOfSRPBinBufInterface, the c functions begin with prefix "SRPBinBuf_";

For ClassOfSRPParaPackageInterface, the c functions begin with prefix "SRPParaPkg_";

For ClassOfSRPInterface, the c functions begin with prefix "SRPI_";

For ClassOfSRPMemoryFileInterface, the c functions begin with prefix "SRPMF_";

For ClassOfSRPFileDiskInterface, the c functions begin with prefix "SRPFD_";

For ClassOfCoreShellInterface, the c functions begin with prefix "SRPCS_";

For ClassOfSRPFunctionParaInterface, the c functions begin with prefix "SRPFP_";

For ClassOfSRPLockInterface, the c functions begin with prefix "SRPLock_";

For ClassOfStarCore, the c functions begin with prefix "StarCore_";

The C interface also exports the following three functions:

```
extern void SRPAPI SRPMM_memset(void *Buf,VS_INT8 c,VS_INT32 Len);
extern void SRPAPI SRPMM_memcpy(void *DesBuf,void *SrcBuf,VS_INT32 Len);
```

extern VS_INT32 SRPAPI SRPMM_strlen(VS_INT8 *Buf);

All c functions are stored in a function table, programmer can get them using "GetCFunctionTable" defined in SRPControl_GetCFunctionTable;

5.1 Init Cle using C

Link with libstarcore.lib on windows or libstarcore.so on linux or android

```
VSCore_RegisterCallBackInfo(MsgCallBack,0);
VSCore_Init( TRUE, TRUE, "", 0, "", 3008,NULL);
```

Dynamic load share library libstarcore.dll(window) or libstarcore.so(linux)

```
sprintf(ModuleName,"libstarcore%s",VS_MODULEEXT);
hDllInstance = vs_dll_open( ModuleName );
if( hDllInstance == NULL ){
    printf("load library [%s] error....\n",ModuleName);
    return -1;
}
Core_RegisterCallBackInfo =
(VSCore_RegisterCallBackInfoProc)vs_dll_sym(hDllInstance,"VSCore_RegisterCallBackInfo");
Core_Init = (VSCore_InitProc)vs_dll_sym(hDllInstance,"VSCore_Init");
Core_QueryControlInterface =
(VSCore_QueryControlInterfaceProc)vs_dll_sym(hDllInstance,"VSCore_QueryControlInterface");

Core_RegisterCallBackInfo(MsgCallBack,0);
Core_Init( VS_TRUE, VS_TRUE, "", 0, "", 3008,NULL);

printf("init starcore success\n");

SRPControlInterface = Core_QueryControlInterface();
Control_GetCFunctionTable =
(SRPControl_GetCFunctionTable_Proc)vs_dll_sym(hDllInstance,"SRPControl_GetCFunctionTable");
CoreFunctionTbl = (struct StructOfVSStarCoreInterfaceTable *)Control_GetCFunctionTable(SRPControlInterface);

BasicSRPInterface = CoreFunctionTbl->SRPControl_QueryBasicInterface(SRPControlInterface,0);
```

The struct "StructOfVSStarCoreInterfaceTable " holds all c functions address of libstarcore. You can also use dll_sym or GetProcAddess.

5.2 Using c interface function

```
#include "vsopenapi_c.h"

static VS_ULONG MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_ULONG wParam, VS_ULONG
lParam, VS_BOOL *IsProcessed, VS_ULONG Para )
```

```
{
   switch( uMsg ){
   case MSG_VSDISPMSG :
        case MSG_VSDISPLUAMSG :
                printf("[core]%s\n",(VS_CHAR *)wParam);
                break;
   case MSG_DISPMSG :
        case MSG_DISPLUAMSG :
                printf("%s\n",(VS_CHAR *)wParam);
       break;
        case MSG_EXIT :
                break;
   }
   return 0;
}

static VS_INT32 GetNumber(void *Object,VS_INT32 Para)
{
   printf( "Remote Call Number [%d]\n ",Para);
        return Para + 1;
}

static VS_CHAR *GetString(void *Object,VS_CHAR *Para)
{
        static VS_CHAR CharBuf[128];

   printf( "Remote Call String [%s]\n",Para);
        sprintf(CharBuf,"%sasdfsaf",Para);
   return CharBuf;
}

VS_PARAPKGPTR ParaPkgPtr;

static VS_PARAPKGPTR GetPkg(void *Object,VS_PARAPKGPTR Para)
{
   printf( "Remote Call Pkg [%d]",SRPParaPkg_GetInt(Para,0));
        SRPParaPkg_Clear(ParaPkgPtr);
        SRPParaPkg_InsertStr(ParaPkgPtr,0,"asdfsaf");
   return ParaPkgPtr;
}

//--------------------------------------------------------------------------------
void *SRPControlInterface;
void *BasicSRPInterface;
//--------------------------------------------------------------------------------
```

```
int main(int argc, char* argv[])
{
        void *SRPInterface;
        VS_UUID ClassID;
        void *AtomicClass,*AtomicFunction,*Object;
        VS_CHAR *ErrorInfo;
        VS_UUID ServiceID;

        VSCore_RegisterCallBackInfo(MsgCallBack,0);
        VSCore_Init( TRUE, TRUE, "", 0, "", 3008,NULL);
        printf("init starcore success\n");

        SRPControlInterface = VSCore_QueryControlInterface();
        BasicSRPInterface = SRPControl_QueryBasicInterface(SRPControlInterface,0);

   INIT_UUID( ServiceID );
   if( SRPBasic_CreateService( BasicSRPInterface, "","test",&ServiceID,"123",0,0,0,0,0,0 ) == VS_FALSE )
      return 0;
   SRPInterface = SRPBasic_GetSRPInterface(BasicSRPInterface, "test","root","123");
   if( SRPInterface == NULL )
      return 0;
        SRPI_CreateSysRootItem(SRPInterface,"TestItem","",NULL,NULL);
        SRPI_ActiveSysRootItem( SRPInterface,"TestItem" );
        //---Create Atomic Class, for define function, no attribute
        AtomicClass = SRPI_CreateAtomicObjectSimple(SRPInterface,"TestItem","TestClass",NULL,NULL,&ErrorInfo);
        AtomicFunction = SRPI_CreateAtomicFunctionSimple(SRPInterface,AtomicClass,"GetNumber","VS_INT32
GetNumber(VS_INT32 Para);",NULL,&ErrorInfo,VS_FALSE,VS_FALSE);
        SRPI_SetAtomicFunction(SRPInterface,AtomicFunction,(void *)GetNumber);
        AtomicFunction = SRPI_CreateAtomicFunctionSimple(SRPInterface,AtomicClass,"GetString","VS_CHAR
*GetString(VS_CHAR *Para);",NULL,&ErrorInfo,VS_FALSE,VS_FALSE);
        SRPI_SetAtomicFunction(SRPInterface,AtomicFunction,(void *)GetString);
        AtomicFunction = SRPI_CreateAtomicFunctionSimple(SRPInterface,AtomicClass,"GetPkg","VS_PARAPKGPTR
GetPkg(VS_PARAPKGPTR Para);",NULL,&ErrorInfo,VS_FALSE,VS_FALSE);
        SRPI_SetAtomicFunction(SRPInterface,AtomicFunction,(void *)GetPkg);

        ParaPkgPtr = SRPI_GetParaPkgInterface(SRPInterface);

        printf("create TestObject for remotecall..\n");
        SRPI_GetAtomicID(SRPInterface,AtomicClass,&ClassID);
       Object = SRPI_MallocGlobalObject(SRPInterface,SRPI_GetSysRootItem(SRPInterface,"TestItem"),0,&ClassID,0,NULL,0);
        SRPI_SetName(SRPInterface,Object,"TestObject");

        printf("finish,enter message loop..\n");
        while( 1 ){
                VS_INT32 Ch;
```

```
            Ch = _kbhit();
            if( Ch == 27 )
                break;
            SRPControl_SRPDispatch(SRPControlInterface,VS_FALSE);
        }
        SRPParaPkg_Release(ParaPkgPtr);
        SRPI_Release(SRPInterface);

        return 0;
}
```

6 Delphi Interface

For version 2.1.0, cle supports Delphi and can be used to program for android and windows. The ios platform may be supported in later version.

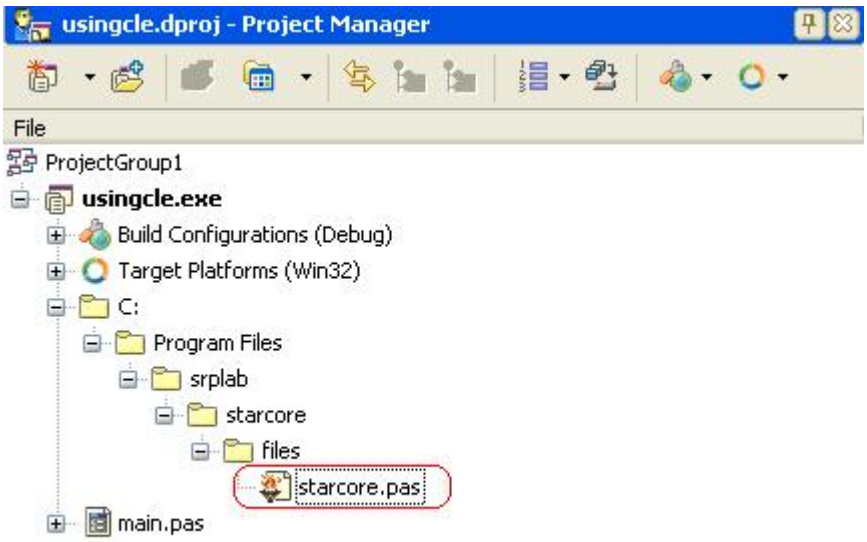
Version of Delphi supported must be Delphi 2010, Delphi xe, Delphi xe2, .., Delphi xe10 or above.

Programmer uses pascal the interact with cle. The Delphi interface is same with C Interface.

example projects located at "examples\delphi"

6.1 Using cle with delphi on windows

6.1.1 Add "starcore.pas"



uses

…,starcore;

6.1.2 Init Cle

```
function MsgCallBack( ServiceGroupID:VS_ULONG; uMsg:VS_ULONG; wParam:VS_UWORD; lParam:VS_UWORD;
IsProcessed:PVS_BOOL; Para:VS_UWORD ) : VS_UWORD; cdecl;
var
  str : string;
begin
  if ( uMsg = MSG_VSDISPLUAMSG) or (uMsg = MSG_VSDISPMSG ) or (uMsg = MSG_DISPMSG ) or (uMsg =
MSG_DISPLUAMSG ) then
  begin
      str := TOVS_STRING(PVS_CHAR(wParam));
      if not (str.Length = 0 ) then
      begin

      end;
  end;
  Result := 0;
end;

procedure TForm3.FormCreate(Sender: TObject);
var
 SRPControlInterface : Pointer;
 BasicSRPInterface : Pointer;
 ServiceID : VS_UUID;
begin
 Label1.Caption := 'Init Fail';
 StarCore_Init('libstarcore.dll');
 VSCore_RegisterCallBackInfo(@MsgCallBack,0);
 VSCore_Init( TRUE, TRUE, '', 0, '', 3008,nil);
SRPControlInterface := VSCore_QueryControlInterface();
BasicSRPInterface := SRPControl_QueryBasicInterface(SRPControlInterface,0);
 INIT_UUID( ServiceID );
 SRPBasic_ImportService( BasicSRPInterface, 'SRPFSEngine', true );
 if( SRPBasic_CreateService( BasicSRPInterface, ','test',@ServiceID,'123',0,0,0,0,0,0 ) = false ) then
    exit;
 Label1.Caption := 'Init OK';
end;
```

6.1.3 Using TSRPVariant to access object

TSRPVariant is an encapsulation of cle object on windows platform. It act as com object to provide convenient object access interface.

Before using TSRPVariant, **SRP_CLEInterface** must be set as service interface.

For example,

```
Var
  python : Pointer;
  varpython : variant;

begin

  StarCore_Init('libstarcore.dll');
  VSCore_RegisterCallBackInfo(@MsgCallBack,0);
  VSCore_Init( TRUE, TRUE, '', 0, '', 3008,nil);
        SRPControlInterface := VSCore_QueryControlInterface();
        BasicSRPInterface := SRPControl_QueryBasicInterface(SRPControlInterface,0);
  INIT_UUID( ServiceID );
  if( SRPBasic_CreateService( BasicSRPInterface, '','test',@ServiceID,'123',0,0,0,0,0,0 ) = false ) then
    exit;
  CLEInterface := SRPBasic_GetSRPInterface(BasicSRPInterface,'test','root','123');
  SRP_CLEInterface := CLEInterface

  SRPBasic_InitRaw(BasicSRPInterface,'python35',CLEInterface);
  python := SRPI_ImportRawContext(CLEInterface,'python','',false,NULL);
  varpython := SRPOBJECT_TOVARIANT (python,true);
  varpython.import('os');
  str := varpython.os.getcwd;
```

When call object's function with TSRPVariant, if the result is parapkg, binbuf, SXml, FunctionPara, CommInterface. The returned variant can not handle this object. You must get the pointer, using function "SRPVARIANT_TOPOINTER (varpython);", and the use the pointer as normal cle object.

You can alse create TSRPParaPkg, TSRPBinBuf, TSRPSXml, and TSRPComm instance to aid to call their methods.

For example

```
  pk := SRPVARIANT_TOPOINTER (xxx)
  Tp := TSRPParaPkg.create(pk,false);
  Tp.xxxx()
```

```
  function SRPOBJECT_TOVARIANT(X:Pointer;NeedRelease:Boolean):variant;
  function SRPPARAPKG_TOVARIANT(X:Pointer;NeedRelease:Boolean):variant;
  function SRPBINBUF_TOVARIANT(X:Pointer;NeedRelease:Boolean):variant;
  function SRPSXML_TOVARIANT(X:Pointer;NeedRelease:Boolean):variant;
  function SRPCOMM_TOVARIANT(X:Pointer;NeedRelease:Boolean):variant;
  function SRPFUNCPARA_TOVARIANT(X:Pointer;NeedRelease:Boolean):variant;
```

function SRPVARIANT_TOPOINTER(X:variant):Pointer;

TSRPVariant has three additional functions:

ID() : which is used to get the cle object's uuid.

Create() or create(arg1,arg2,…) : which is used to create instance of class.

ToString() : which is used to get string of cle object.

6.1.4 Sample Code

```
StarCore_Init('libstarcore.dll');
VSCore_RegisterCallBackInfo(@MsgCallBack,0);
VSCore_Init( TRUE, TRUE, '', 0, '', 3008,nil);
SRPControlInterface := VSCore_QueryControlInterface();
BasicSRPInterface := SRPControl_QueryBasicInterface(SRPControlInterface,0);
INIT_UUID( ServiceID );
if( SRPBasic_CreateService( BasicSRPInterface, '','test',@ServiceID,'123',0,0,0,0,0,0 ) = false ) then
  exit;
CLEInterface := SRPBasic_GetSRPInterface(BasicSRPInterface,'test','root','123');
SRP_CLEInterface := CLEInterface;

SRPBasic_InitRaw(BasicSRPInterface,'python35',CLEInterface);
python := SRPI_ImportRawContext(CLEInterface,'python','',false,NULL);
RetValue := SRPI_ScriptCall(CLEInterface,python,NULL,TOVS_CHAR('import'),TOVS_CHAR('(s)'),TOVS_CHAR('sys'));
sys := Pointer(SRPI_ScriptGetRawObject(CLEInterface,python,'sys',NULL));

tc := TSRPComm.create();
str := tc.FormatRspHeader('aaaa','bbbb','cccc','dddd',34456);

varpython := IDispatch(TSRPVariant.Create(python,true));
bbb := varpython.IsObject();
ssss := ISRPVariant(IDispatch(varpython)).ToObject();

varpython.import('os');
str := varpython.os.getcwd;
```

6.1.5 Call Tensorflow

```
unit main;

interface

uses
```

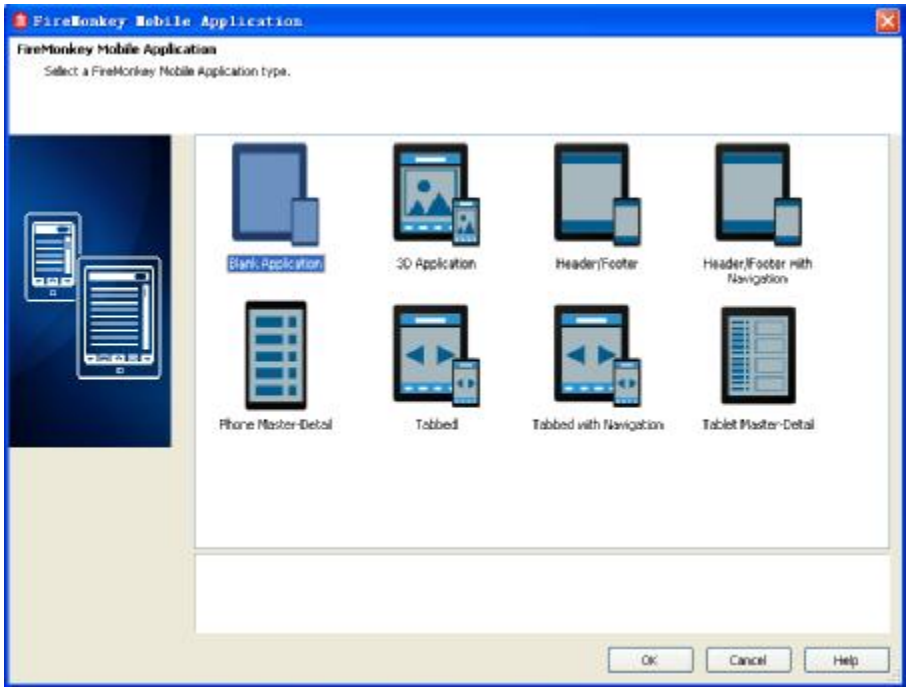
```
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs,starcore, Vcl.StdCtrls;

 type
  TForm1 = class(TForm)
    Memo1: TMemo;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

 var
  Form1: TForm1;

 implementation

 {$R *.dfm}

 function MsgCallBack( ServiceGroupID:VS_ULONG; uMsg:VS_ULONG; wParam:VS_UWORD; lParam:VS_UWORD;
IsProcessed:PVS_BOOL; Para:VS_UWORD ) : VS_UWORD; cdecl;
 var
   str : string;
 begin
   if ( uMsg = MSG_VSDISPLUAMSG) or (uMsg = MSG_VSDISPMSG ) or (uMsg = MSG_DISPMSG ) or (uMsg =
MSG_DISPLUAMSG ) then
    begin
       str := TOVS_STRING(PVS_CHAR(wParam));
       if not (Length(str) = 0 ) then
       begin
          Form1.Memo1.Lines.Add(str);
       end;
    end;
   Result := 0;
 end;

 procedure TForm1.FormCreate(Sender: TObject);
 var
  SRPControlInterface : Pointer;
  BasicSRPInterface : Pointer;
  ServiceID : VS_UUID;
  CLEInterface : Pointer;

  python : Pointer;
```

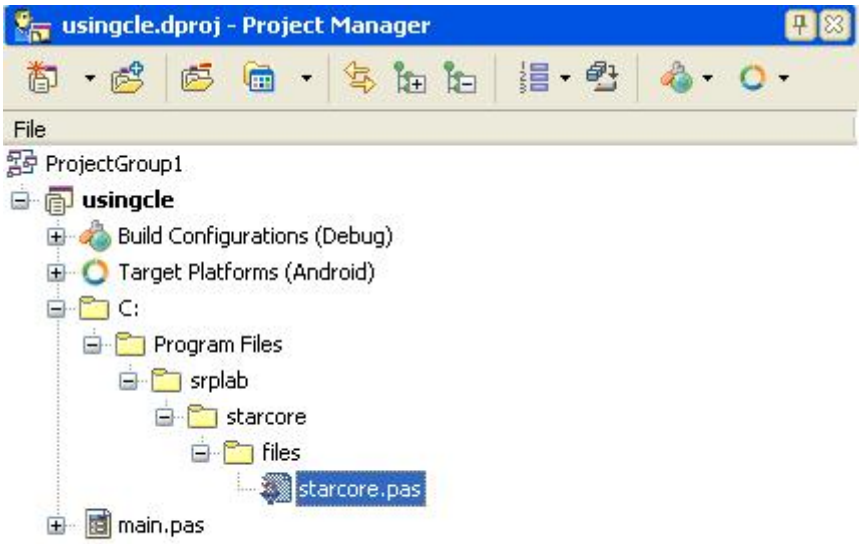
```
  tempobj : Pointer;
  varpython : variant;

  tf : variant;
  tv : string;
  str : string;

  a,b,c,sessclass,sess : variant;

  para,feed_dict : TSRPParaPkg;
  res : variant;

begin
  Memo1.Clear();
  StarCore_Init('libstarcore.dll');
  VSCore_RegisterCallBackInfo(@MsgCallBack,0);
  VSCore_Init( TRUE, TRUE, '', 0, '', 3008,nil);
      SRPControlInterface := VSCore_QueryControlInterface();
      BasicSRPInterface := SRPControl_QueryBasicInterface(SRPControlInterface,0);
  INIT_UUID( ServiceID );
  if( SRPBasic_CreateService( BasicSRPInterface, '','test',@ServiceID,'123',0,0,0,0,0,0 ) = false ) then
    exit;
  CLEInterface := SRPBasic_GetSRPInterface(BasicSRPInterface,'test','root','123');
  SRPBasic_InitRaw(BasicSRPInterface,'python35',CLEInterface);
  python := SRPI_ImportRawContext(CLEInterface,'python','',false,NULL);

  varpython := SRPOBJECT_TOVARIANT(python,true);
  varpython.import('sys');

  Memo1.Lines.Add(varpython.sys.ToString);

  varpython.eval('import tensorflow as tf');
  tf := varpython.tf;
  tv := tf.VERSION;



  Memo1.Lines.Add(tf.ToString);
  Memo1.Lines.Add(tv);

  //-- a = tf.add(2,5)
  a := tf.add(2, 5);
  Memo1.Lines.Add(a.ToString);
  //-- b = tf.multiply(a,5)
  b := tf.multiply(a, 3);
  Memo1.Lines.Add(b.ToString);
```

```
  //-- c = tf.constant(2,name="Node_c")
  para := TSRPParaPkg.Create;
  para.InsertStr(0,'name').InsertStr(1,'Node_c').AsDict(true);
  c := tf.constant(2,SRPPARAPKG_TOVARIANT(para,false));
  Memo1.Lines.Add(c.ToString);

  //-- result = sess.run(b,feed_dict={a:25});
  sess := tf.Session.create;

  para.Clear;
  para.InsertObject(0,SRPVARIANT_TOPOINTER(a)).InsertInt(1,25).AsDict(true);

  feed_dict := TSRPParaPkg.Create;
  feed_dict.InsertStr(0,'feed_dict').InsertParaPackage(1,para).AsDict(true);

  res := sess.run(b,SRPPARAPKG_TOVARIANT(feed_dict,false));
  Memo1.Lines.Add(res);

  //Memo1.Clear();
end;

end.
```

6.2 Using cle with delphi on android

6.2.1 Create Project and Add "starcore.pas"





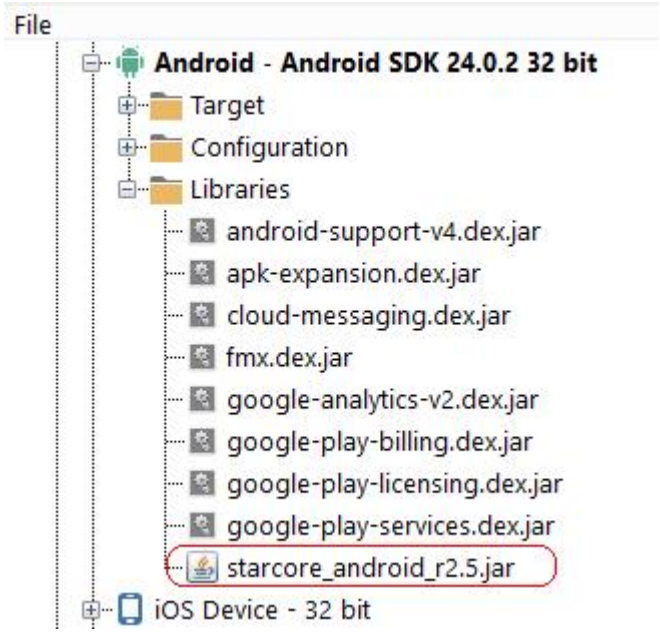
6.2.2 Add cle share libraries.

Open "Project -> Deployment "

Local Path	Local Name	Type	Platforms	Remote Path	Remote Name
Android\Debug\	libusingcle.so	ProjectOutput	[Android]	library\lib\armeabi-v7a\	libusingcle.so
$(BDS)\bin\Artwor...	FM_SplashImage_960...	Android_Spla...	[Android]	res\drawable-xlarge\	splash_image
$(BDS)\lib\android...	libnative-activity.so	AndroidLibna...	[Android]	library\lib\mips\	libusingcle.so
$(BDS)\bin\Artwor...	FM_SplashImage_640...	Android_Spla...	[Android]	res\drawable-large\	splash_image
..\starcore_for_and...	libstar_java.so	File	[Android]	library\lib\armeabi-v7a\	libstar_java.so
..\starcore_for_and...	libstarcore.so	File	[Android]	library\lib\armeabi-v7a\	libstarcore.so
$(BDS)\lib\android...	libnative-activity.so	AndroidLibna...	[Android]	library\lib\x86\	libusingcle.so
$(BDS)\bin\Artwor...	FM_LauncherIcon_144...	Android_Lau...	[Android]	res\drawable-xxhdpi\	ic_launcher.pr
C:\srplab\example...	classes.dex	AndroidClass...	[Android]	classes\	classes.dex
$(BDS)\bin\Artwor...	FM_LauncherIcon_48x...	Android_Lau...	[Android]	res\drawable-mdpi\	ic_launcher.pr
Android\Debug\	splash_image_def.xml	AndroidSplas...	[Android]	res\drawable\	splash_image
Android\Debug\	AndroidManifest.xml	ProjectAndro...	[Android]	.\	AndroidMani
$(BDS)\bin\Artwor...	FM_SplashImage_426...	Android_Spla...	[Android]	res\drawable-small\	splash_image
$(BDS)\bin\Artwor...	FM_LauncherIcon_96x...	Android_Lau...	[Android]	res\drawable-xhdpi\	ic_launcher.pr
Android\Debug\	styles.xml	AndroidSplas...	[Android]	res\values\	styles.xml
$(NDKBasePath)\p...	gdbserver	AndroidGDB...	[Android]	library\lib\armeabi-v7a\	gdbserver
$(BDS)\bin\Artwor...	FM_LauncherIcon_36x...	Android_Lau...	[Android]	res\drawable-ldpi\	ic_launcher.pr
$(BDS)\lib\android...	libnative-activity.so	AndroidLibna...	[Android]	library\lib\armeabi\	libusingcle.so
$(BDS)\bin\Artwor...	FM_LauncherIcon_72x...	Android_Lau...	[Android]	res\drawable-hdpi\	ic_launcher.pr
$(BDS)\bin\Artwor...	FM_SplashImage_470...	Android_Spla...	[Android]	res\drawable-normal\	splash_image
C:\xe8\android.xe8...	classes.dex	AndroidClass...	[Android]	classes\	classes.dex

If python script is used, libpython2.6.so and libstarpy.so should be added. These share library can be found from starcore for android package.

Add "starcore_android_rX.X.jar" to Libraries

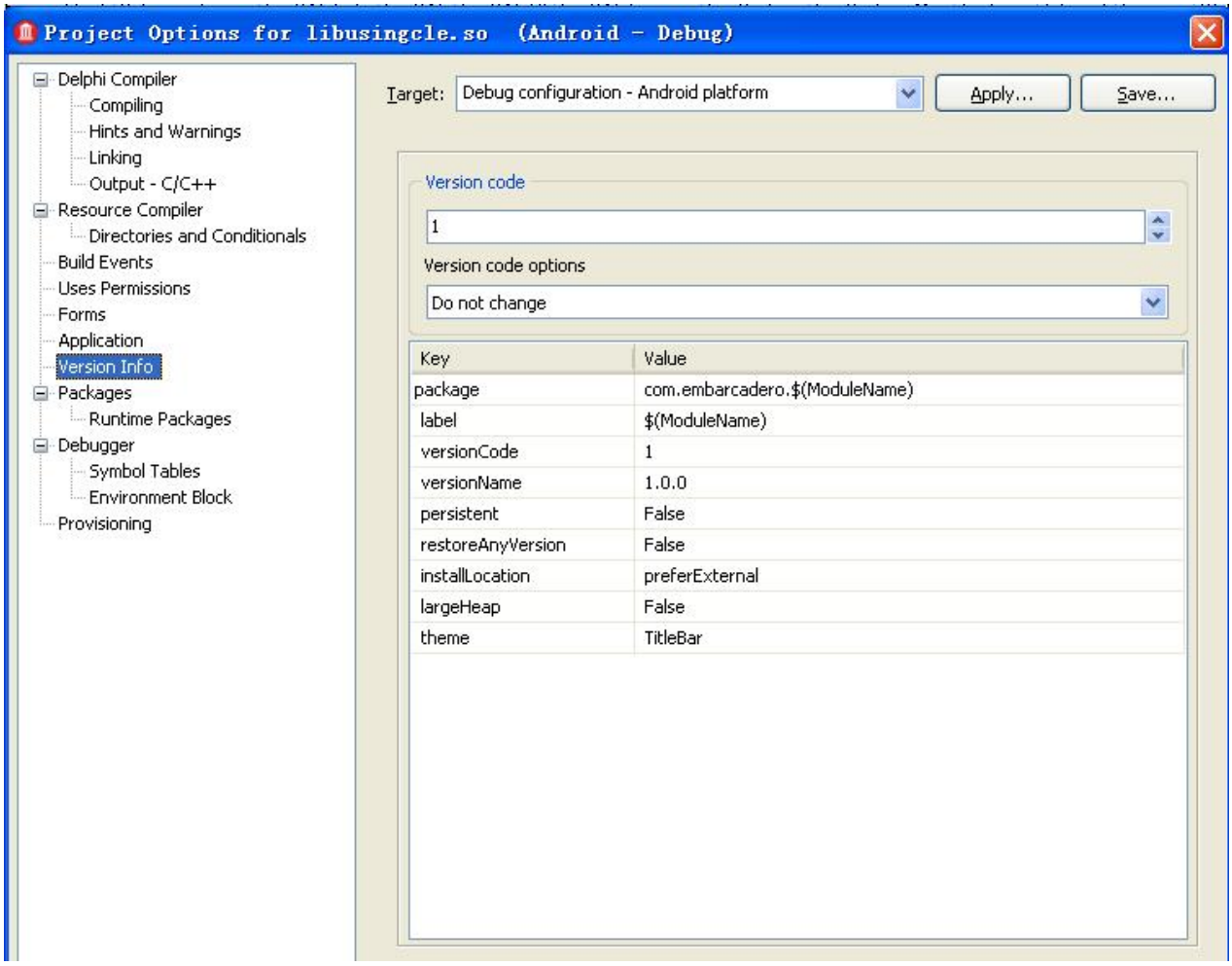


6.2.3 Init Cle

```
procedure TForm3.FormCreate(Sender: TObject);
var
   InitResult : VS_INT32;
   SRPControlInterface : Pointer;
   BasicSRPInterface : Pointer;
   ServiceID : VS_UUID;
   SRPInterface : Pointer;

   TestBuf : CLEString;
begin;
   Label1.Text := 'Init Failed';
{$IFDEF ANDROID}
   if( StarCore_Init('com.embarcadero.usingcle') <> true ) then
{$ELSE}
   if( StarCore_Init('libstarcore.dll') <> true ) then
{$ENDIF}
      exit;
   VSCore_Init(true, true, '', 0, '', 0,nil);
   SRPControlInterface := VSCore_QueryControlInterface();
        BasicSRPInterface := SRPControl_QueryBasicInterface(SRPControlInterface,0);
   INIT_UUID( ServiceID );
   if( SRPBasic_CreateService( BasicSRPInterface, '','test',@ServiceID,'123',0,0,0,0,0,0 ) = false ) then
      exit;
   SRPInterface := SRPBasic_GetSRPInterface(BasicSRPInterface, 'test','root','123');
   if( not assigned(SRPInterface) ) then
      exit;
   Label1.Text := 'Create Service Ok';
```

```
end;
```

'com.embarcadero.usingcle' is the package name, which can be found at options of the project.



6.2.4 Call java code

The activity object is cached in "**StarCoreFactoryPath. ActivityObject**"

Delphi call java:

```
 SRPBasic_InitRaw(BasicSRPInterface,'java',SRPInterface);
 StarCoreFactoryPath :=
SRPI_ImportRawContext(SRPInterface,'java','com/srplab/www/starcore/StarCoreFactoryPath',true,NULL);
 // get activity object
 ActivityObject := Pointer(SRPI_ScriptGetObject(SRPInterface,StarCoreFactoryPath,'ActivityObject',NULL));
 // get app title
 AndroidAppTitle := PVS_CHAR(SRPI_ScriptCall( SRPInterface, ActivityObject, NULL,
BSChar1.SetString('getTitle').ToVSChar,BSChar2.SetString('()s'). ToVSChar));
 // change PVS_CHAR to string
 Memo1.Text := BSChar1.SetVSChar(AndroidAppTitle).ToString();
```

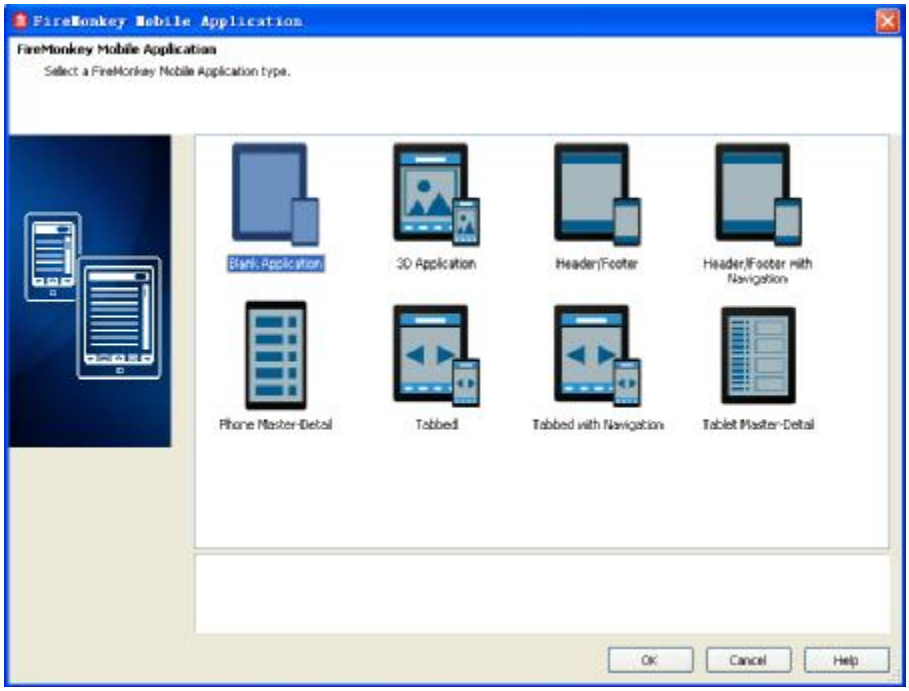
lua call java:

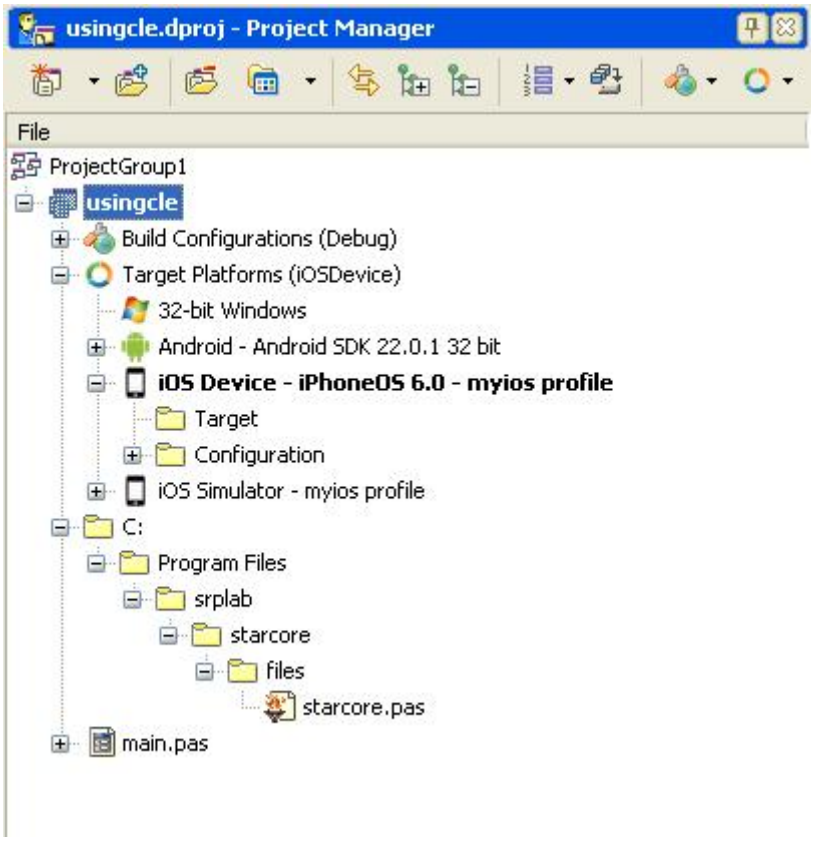
```
SrvGroup=libstarcore:_GetSrvGroup(0)
Service=SrvGroup:_GetService("","")
print(Service)
print(Service.TestClass)
print(Service.TestClass:Add(123.4,456.7))

SrvGroup:_InitRaw("java",Service);
TestJava = Service:_ImportRawContext("java","android/graphics/Color",true,nil);
print(TestJava)

ActivityJava = Service:_ImportRawContext("java","com/srplab/www/starcore/StarCoreFactoryPath",true,nil);
print(ActivityJava.StarCoreCoreLibraryPath)
print(ActivityJava.ActivityObject)

print(ActivityJava.ActivityObject:getTitle())
print(ActivityJava.ActivityObject:getPreferences(0))
```

6.3 Using cle with delphi on ios

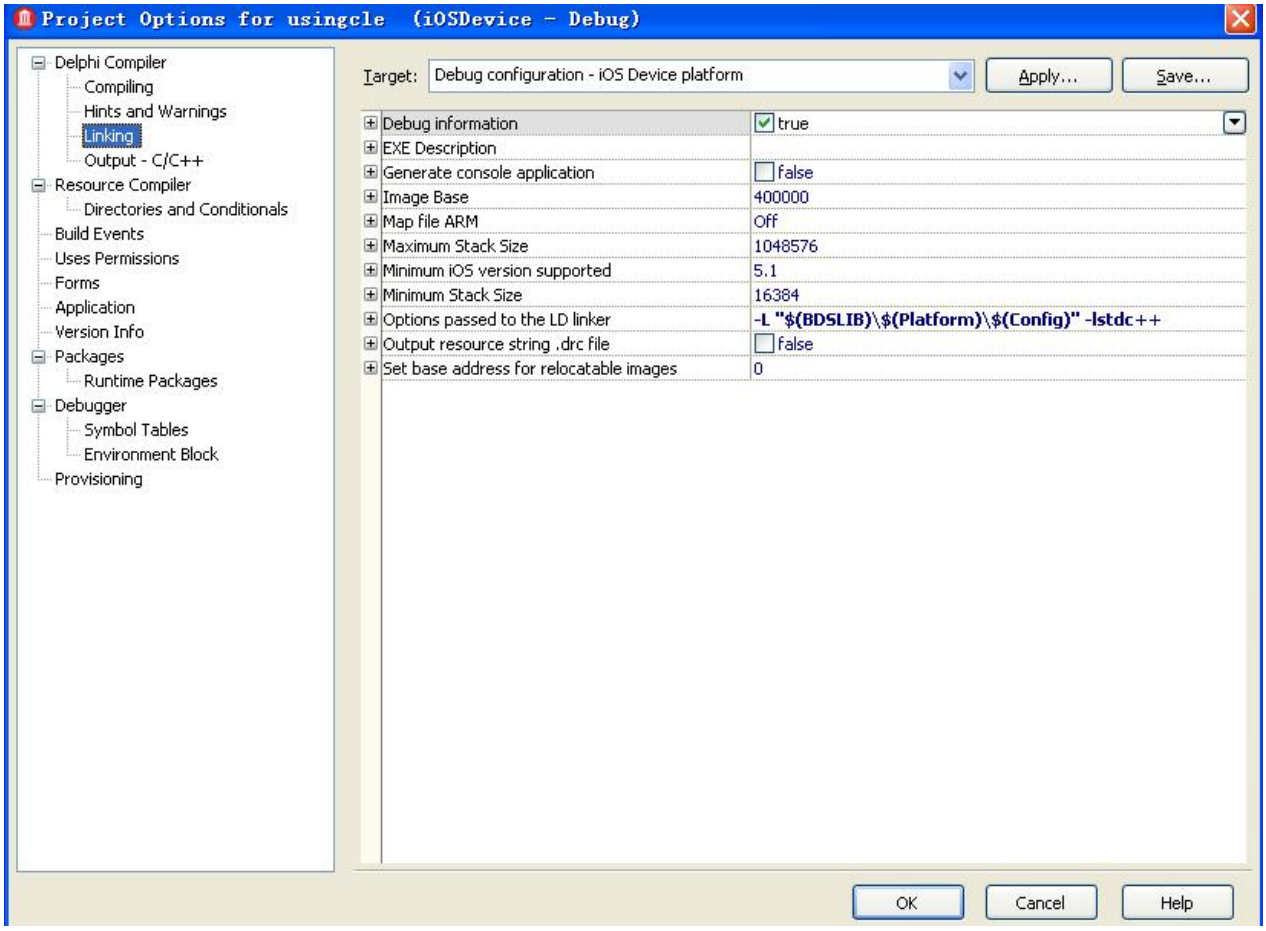
Note : Using CLE with delphi has great limit, for "dlsym" function always returns 0.

6.3.1 Create Project and Add "starcore.pas"





6.3.2 Set Link with stdc++



add link parameter: -L "$(BDSLIB)\$(Platform)\$(Config)" -lstdc++

The static library 'starcore.a' or 'starcorepy.a' can be found at starcore for ios package.

If static library needs add into the project, uses statements as follow,

```
{$IFDEF IOS}
{$IF DEFINED(CPUARM)}
{$link libXXXX.a}
{$ENDIF}
{$ENDIF}
```

6.3.3 Init Cle

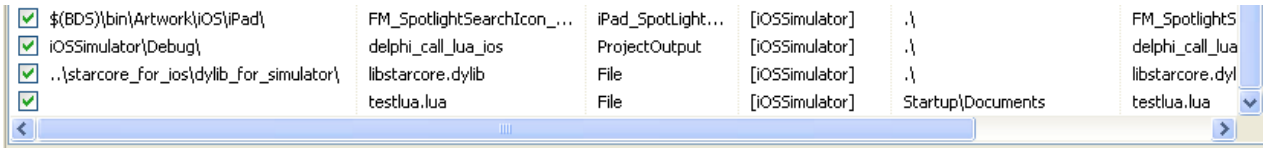
```
procedure TForm3.FormCreate(Sender: TObject);
var
   InitResult : VS_INT32;
   SRPControlInterface : Pointer;
   BasicSRPInterface : Pointer;
   ServiceID : VS_UUID;
   SRPInterface : Pointer;

   TestBuf : CLEString;
```

```
begin;
{$IFDEF ANDROID}
   if( StarCore_Init('com.embarcadero.usingcle') <> true ) then
{$ENDIF}
{$IFDEF MSWINDOWS}
   if( StarCore_Init('libstarcore.dll') <> true ) then
{$ENDIF}
{$IFDEF IOS}
   if( StarCore_Init('usingcle') <> true ) then    /* the input of StarCore_Init is project name */
{$ENDIF}
      exit;
   VSCore_Init( true, true, '', 0, '', 0,PVS_STARCONFIGEX(nil));
   SRPControlInterface := VSCore_QueryControlInterface();
        BasicSRPInterface := SRPControl_QueryBasicInterface(SRPControlInterface,0);
   INIT_UUID( ServiceID );
   if( SRPBasic_CreateService( BasicSRPInterface, '','test',@ServiceID,'123',0,0,0,0,0,0 ) = false ) then
      exit;
   SRPInterface := SRPBasic_GetSRPInterface(BasicSRPInterface, 'test','root','123');
end;
```

6.3.4 Deploy files

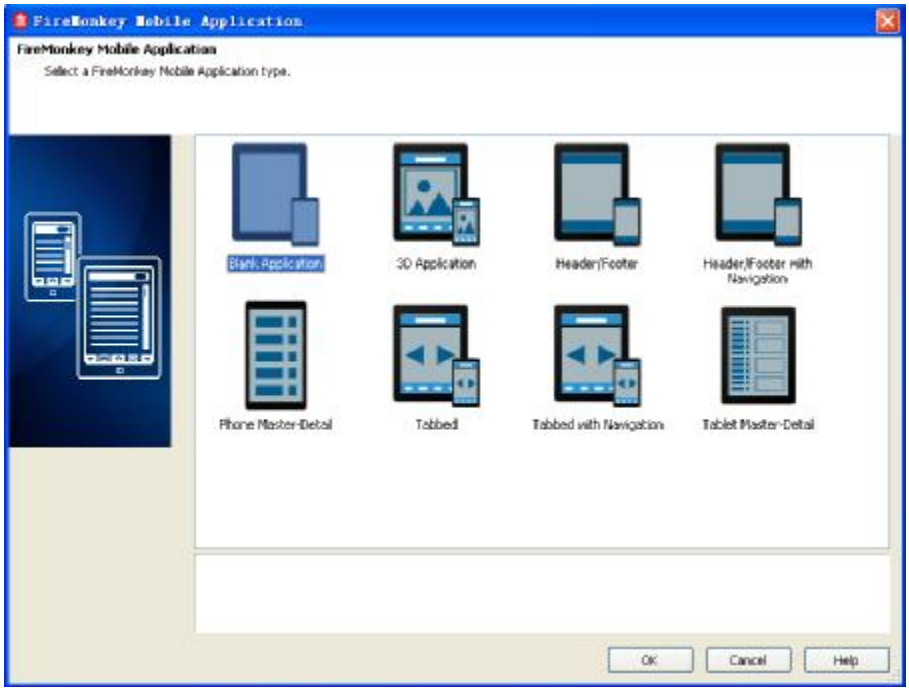
For resource files, remote path set to "StartUp\Documents\" and using the following code to get file path.

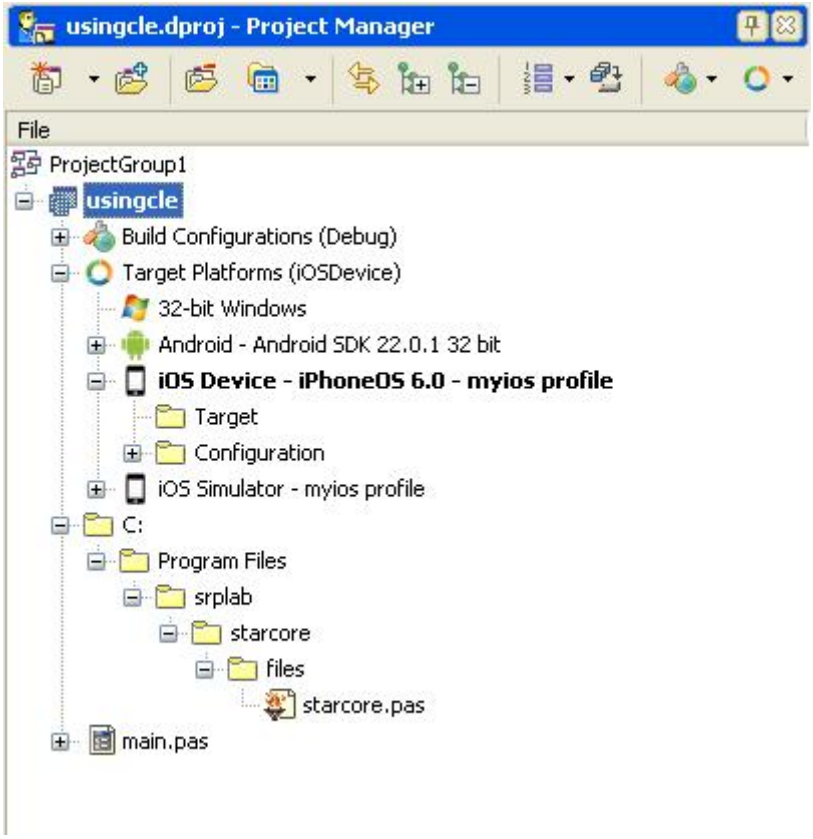
"GetHomePath + PathDelim + 'Documents' + PathDelim"



6.4 Using cle with delphi on ios simulator

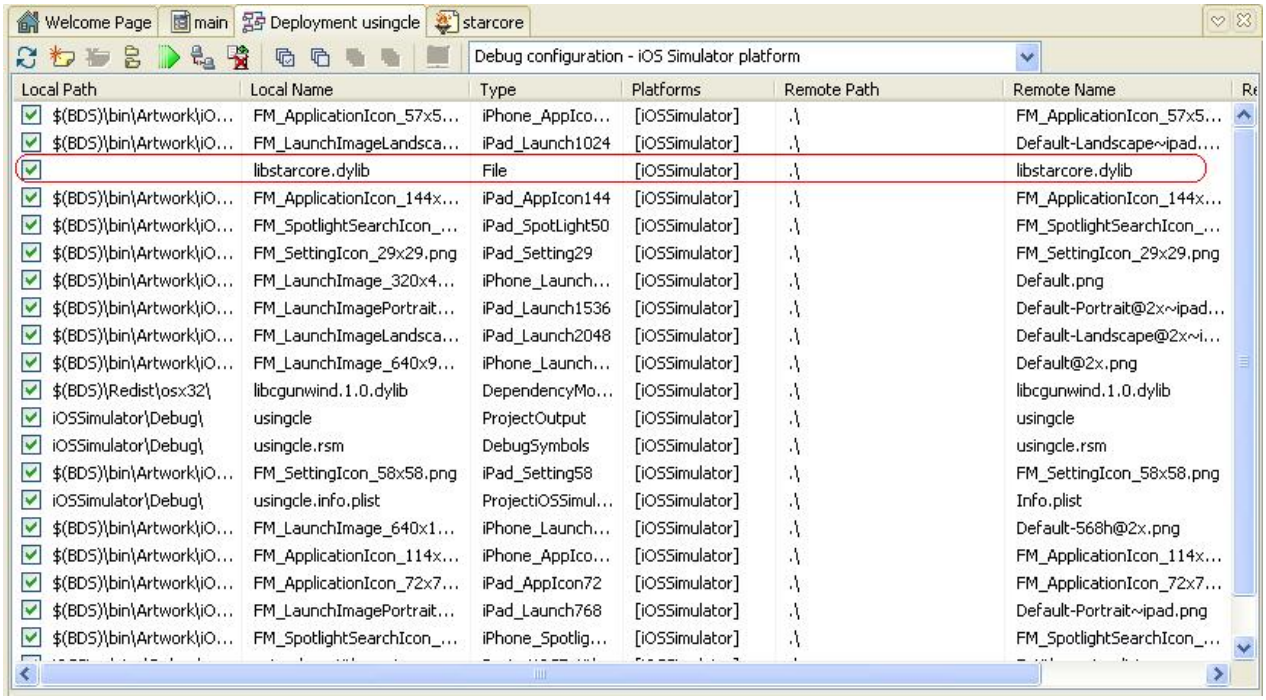
6.4.1 Create Project and Add "starcore.pas"





6.4.2 Add cle share libraries for simulator

Open "Project -> Deployment "



add "libstarcore.dylib" to the project.

The static library 'libstarcore.dylib' or 'libstarcorepy.dylib' can be found at starcore for ios package.

note: please use 32bit version.

6.4.3 Init Cle

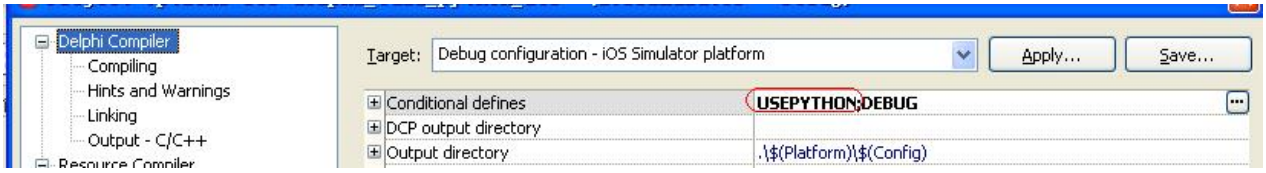
same as ios device.

6.4.4 Deploy files

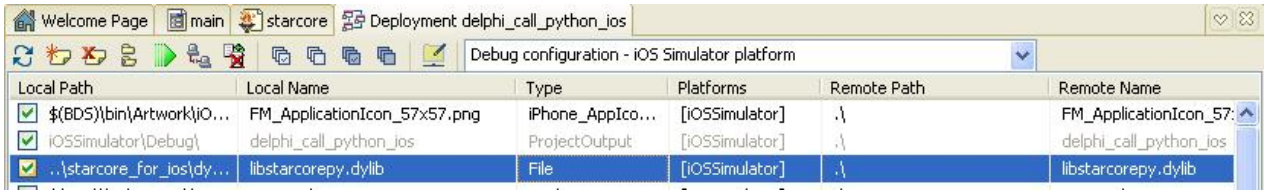
same as ios device.

6.4.5 Using python

Add conditional define "USEPYTHON"



Add libstarcorepy.dylib



Init Cle

```
function MsgCallBack( ServiceGroupID:VS_ULONG; uMsg:VS_ULONG; wParam:VS_UWORD; lParam:VS_UWORD;
IsProcessed:PVS_BOOL; Para:VS_UWORD ) : VS_UWORD; cdecl;
var
str : CLEString;
begin
  if (uMsg = MSG_VSDISPMSG) or (uMsg = MSG_VSDISPLUAMSG) or (uMsg = MSG_DISPMSG) or (uMsg =
MSG_DISPLUAMSG) then
  begin
     str := CLEString( PVS_CHAR(wParam) );
     Form1.Memo1.Lines.Add(str.ToString);
  end;

  Result := 0;
end;
```

```
{$IFDEF ANDROID}
 if( StarCore_Init('com.embarcadero.delphi_call_python_ios') <> true ) then
{$ENDIF}
{$IFDEF MSWINDOWS}
 if( StarCore_Init('libstarcore.dll') <> true ) then
{$ENDIF}
{$IFDEF IOS}
 if( StarCore_Init('delphi_call_python_ios') <> true ) then
{$ENDIF}
   exit;
 (* init python module path *)
 StarCore_InitPython(GetHomePath + '/XXXX.app/python',GetHomePath + '/XXXX.app/python2.7.zip',NULL);
(* XXXX is package name *)

 VSCore_Init( TRUE, TRUE, '', 0, '', 0, nil);
 VSCore_RegisterCallBackInfo(@MsgCallBack,0);
 SRPControlInterface := VSCore_QueryControlInterface();
 BasicSRPInterface := SRPControl_QueryBasicInterface(SRPControlInterface,0);
 INIT_UUID( ServiceID );
 SRPBasic_CreateService( BasicSRPInterface, '','test',@ServiceID,'123',0,0,0,0,0,0 );
 CLEInterface := SRPBasic_GetSRPInterface(BasicSRPInterface,'test','root','123');
```

```
(*----init python raw interface ---*)
SRPControl_SetScriptInterface(SRPControlInterface,'python','','-S -d');
BoolTemp := SRPBasic_InitRaw(BasicSRPInterface,'python',CLEInterface);
```

6.5 Using CLEString

Strings of delphi are unicode. When they are acted as prameters or return values of cle functions, the strings should be coverted to ansi. The record CLEString can be used to do this function.

```
type CLEString = record
  private
  TType : Integer;
  Value : PVS_CHAR;
  public
  class operator Implicit(InData : string) : CLEString;
  class operator Implicit(InData : PVS_CHAR) : CLEString;
  class operator Explicit(InData : string) : CLEString;
  class operator Explicit(InData : PVS_CHAR) : CLEString;
  public
  function SetString(InData : string) : CLEString;
  function SetVSChar(InData : PVS_CHAR) : CLEString;
  function SetPChar(InData : PChar) : CLEString;

  procedure Clear();
  public
  function ToString() : string;
  function ToVSChar() : PVS_CHAR;
end;
```

Using CLEString should be after cle initialized finished.

When using CLEString to conver string, it alloc memory to hold the converted ansi string. Programmers shoul use Clear to free the memory.

```
example:
TestBuf : CLEString;

 TestBuf.SetString('aaaaaaaaaaaaa');
 Button2.Text := TestBuf.ToString();

SRPI_ScriptCall(SRPInterface,CLEObject,nil, TOVS_CHAR('tt'), TOVS_CHAR ('(ss)p'), TOVS_CHAR('hello  '),
TOVS_CHAR('world')));
```

```
function TOVS_CHAR(X : string) : PVS_CHAR;
function TOVS_STRING(X : PVS_CHAR) : string;
```

The following two functions should be used when act as parameters to cle function.

```
function TOVS_FLOAT(X : VS_FLOAT) : VS_FLOAT;
function TOVS_DOUBLE(X : VS_DOUBLE) : VS_DOUBLE;
```

6.6 Interact with other scripts

Create CLE objects, define their callback functions, and then the objects can be accessed by other scripts. More detailed information will be listed in the following chapters. Here gives an simple example.

6.6.1 Define object's callback

```
function TestClass_ScriptCallBack( L : Pointer ) : VS_INT32; cdecl;
var
      CLEObject : Pointer;
      ScriptName : PVS_CHAR;
  BSCharTemp : CLEString;
  FloatTemp1 : VS_FLOAT;
  FloatTemp2 : VS_FLOAT;
begin
  ScriptName := SRPI_LuaToString( SRPInterface, SRPI_LuaUpValueIndex(SRPInterface,3) );
  CLEObject := SRPI_LuaToObject(SRPInterface,1);
  BSCharTemp.SetVSChar(ScriptName);
  if( BSCharTemp.ToString() = 'Add' ) then
  begin
    FloatTemp1 := SRPI_LuaToNumber(SRPInterface,2);
    FloatTemp2 := SRPI_LuaToNumber(SRPInterface,3);
    SRPI_LuaPushNumber(SRPInterface,FloatTemp1 + FloatTemp2);
    Result := 1;
  end
  else
  begin
    Result := 0;
  end;
  BSCharTemp.Clear();
end;

(* functions of TestClass  *)
function TestClass_LuaFuncFilter(CLEObject:Pointer; ForWhichObject:Pointer; FuncName:PVS_CHAR;
Para:VS_ULONG):VS_BOOL; cdecl;
var
  BSCharTemp : CLEString
begin
  BSCharTemp.SetVSChar(FuncName);
```

```
 if BSCharTemp.ToString() = 'Add' then
 begin
   Result := true;
 end
 else
 begin
   Result := false;
 end;
end;
```

6.6.2 Create CLE Object

```
 SRPI_CheckPassWord(SRPInterface,false);

 (* Create TestClass and exports to other script *)
  TestClass := SRPI_MallocObjectL(SRPInterface,NULL,0,NULL);
SRPI_SetName(SRPInterface,TestClass,'TestClass');
SRPI_RegLuaFunc( SRPInterface, TestClass, NULL, @TestClass_ScriptCallBack, 0 );
SRPI_RegLuaFuncFilter(SRPInterface,TestClass,@TestClass_LuaFuncFilter,0);
```

6.7 Capture print formation from cle

Register callback function of cle. As follow, for example

```
function MsgCallBack( ServiceGroupID:VS_ULONG; uMsg:VS_ULONG; wParam:VS_UWORD; lParam:VS_UWORD;
IsProcessed:PVS_BOOL; Para:VS_UWORD ) : VS_UWORD; cdecl;
var
 strcov:CLEString;
 str : string;
begin
   case uMsg of
   MSG_VSDISPMSG :
     begin
       strcov.SetVSChar(PVS_INT8(wParam));
       str := strcov.ToString;
       WriteLn(str);
     end;
         MSG_VSDISPLUAMSG :
     begin
       strcov.SetVSChar(PVS_INT8(wParam));
       str := strcov.ToString;
       WriteLn(str);
     end;
   MSG_DISPMSG :
```

```
    begin
      strcov.SetVSChar(PVS_INT8(wParam));
      str := strcov.ToString;
      WriteLn(str);
    end;
  MSG_DISPLUAMSG :
    begin
      strcov.SetVSChar(PVS_INT8(wParam));
      str := strcov.ToString;
      WriteLn(str);
    end;
  end;
  Result := 0;
end;

VSCore_RegisterCallBackInfo(@MsgCallBack,0);
```

6.8 Using TSRPParaPkg, TSRPBinBuf, TSRPSXml, TSRPComm

TSRPParaPkg, TSRPBinBuf, TSRPSXml, TSRPComm are delphi classes, which provide functions corresponding to the classes of cle platform.

Before using these cleaaes, **SRP_CLEInterface** must be set as service interface.

For example

```
Var
 Para : TSRPParaPkg;
Begin
StarCore_Init('libstarcore.dll');
 VSCore_RegisterCallBackInfo(@MsgCallBack,0);
 VSCore_Init( TRUE, TRUE, '', 0, '', 3008,nil);
      SRPControlInterface := VSCore_QueryControlInterface();
      BasicSRPInterface := SRPControl_QueryBasicInterface(SRPControlInterface,0);
 INIT_UUID( ServiceID );
 if( SRPBasic_CreateService( BasicSRPInterface, '','test',@ServiceID,'123',0,0,0,0,0,0 ) = false ) then
   exit;
 CLEInterface := SRPBasic_GetSRPInterface(BasicSRPInterface,'test','root','123');
 SRP_CLEInterface := CLEInterface;

 Para := TSRPParaPkg.create();
```

7 C++Builder Interface

.

For c++ builder xe6/xe7/xe8, programmers can write mobile applications using c++. Using cle with c++ builder xe6/ xe7 is more easier than Delphi.

7.1 Using cle with c++ builder on windows

7.1.1 Init CLE

Add "starlib_bc.lib" to the project. And using the following code to init cle.

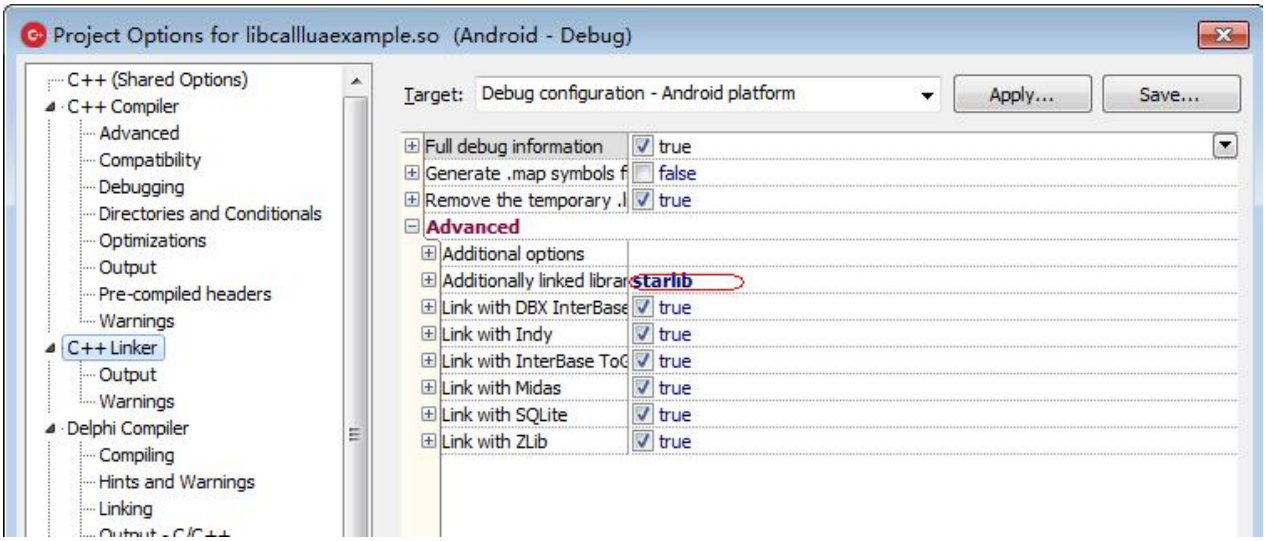
```
#include "vsopenapi.h"
extern "C"{
#include "vs_shell.h"
}
void __fastcall TForm1::FormCreate(TObject *Sender)
{
	VS_HANDLE hDllInstance;
	VSCore_InitProc VSInitProc;
	VSCore_TermProc VSTermProc;
	VSCore_QueryControlInterfaceProc QueryControlInterfaceProc;
	class ClassOfSRPControlInterface *SRPControlInterface = NULL;
	class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
	VS_CHAR ModuleName[512];

	SRPControlInterface = NULL;
	BasicSRPInterface = NULL;
	sprintf(ModuleName,"libstarcore%s",VS_MODULEEXT);
	hDllInstance = vs_dll_open( ModuleName );
	if( hDllInstance == NULL ){
		printf("load library [%s] error....\n",ModuleName);
		return;
	}
	VSInitProc = (VSCore_InitProc)vs_dll_sym( hDllInstance, VSCORE_INIT_NAME );
	VSTermProc = (VSCore_TermProc)vs_dll_sym( hDllInstance, VSCORE_TERM_NAME );
	QueryControlInterfaceProc = (VSCore_QueryControlInterfaceProc)vs_dll_sym( hDllInstance,
	VSCORE_QUERYCONTROLINTERFACE_NAME );
	VSInitProc( true, true, "", 0, "", 0,NULL);
	printf("init starcore success\n");
	SRPControlInterface = QueryControlInterfaceProc();
	BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);

	if( BasicSRPInterface != NULL ){
		BasicSRPInterface -> Release();
		SRPControlInterface -> Release();
	}
```

```
    VSTermProc();
    vs_dll_close(hDllInstance);
    return;
}
```

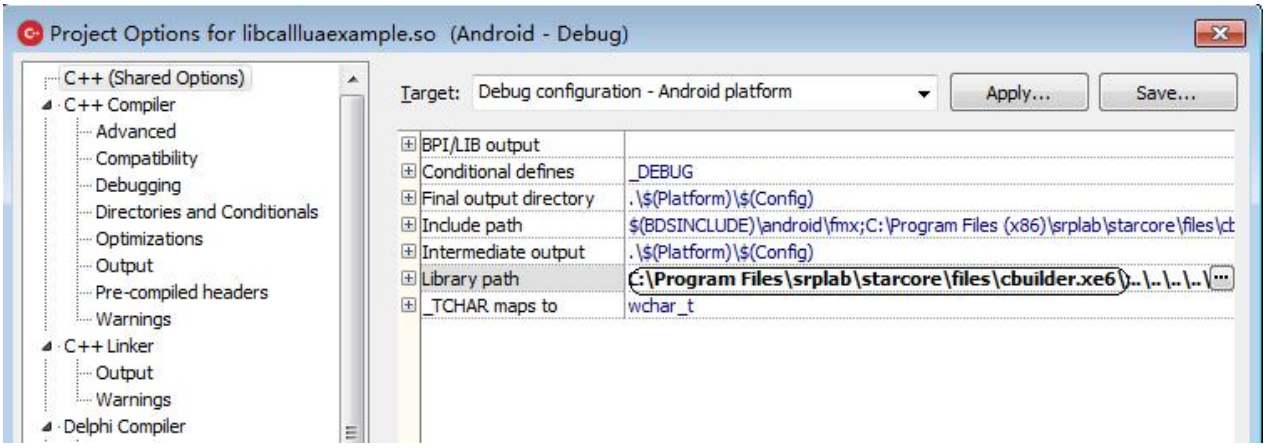
7.2 Using cle with c++ builder on android

7.2.1 Init CLE(First Method)

Add "libstarlib.a" to the project.



Set library search path



And using the following code to init cle.

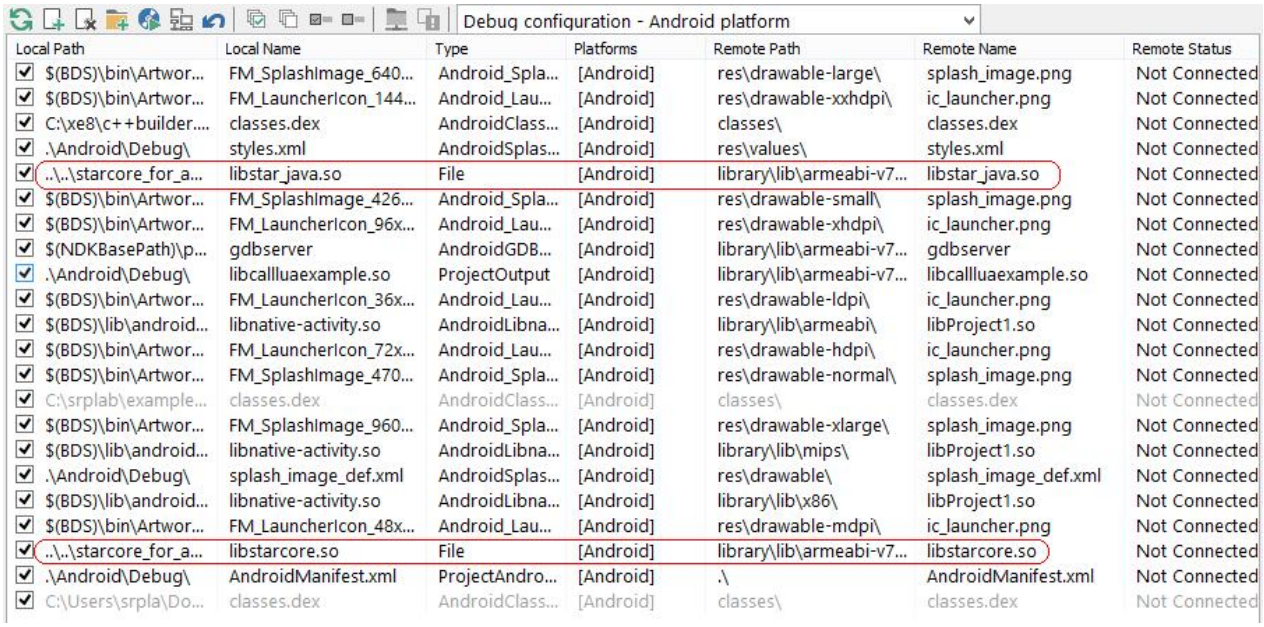
FirstMethod does not c++ or script language call android java code.

```
#include "vsopenapi.h"
extern "C"{
#include "vs_shell.h"
}
```

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
	VS_HANDLE hDllInstance;
	VSCore_InitProc VSInitProc;
	VSCore_TermProc VSTermProc;
	VSCore_QueryControlInterfaceProc QueryControlInterfaceProc;
	class ClassOfSRPControlInterface *SRPControlInterface = NULL;
	class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
	VS_CHAR ModuleName[512];

	SRPControlInterface = NULL;
	BasicSRPInterface = NULL;
	sprintf(ModuleName,"/data/data/com.embarcadero.usingcle/lib/libstarcore%s",VS_MODULEEXT);
	hDllInstance = vs_dll_open( ModuleName );
	if( hDllInstance == NULL ){
		printf("load library [%s] error....\n",ModuleName);
		return;
	}
	VSInitProc = (VSCore_InitProc)vs_dll_sym( hDllInstance, VSCORE_INIT_NAME );
	VSTermProc = (VSCore_TermProc)vs_dll_sym( hDllInstance, VSCORE_TERM_NAME );
	QueryControlInterfaceProc = (VSCore_QueryControlInterfaceProc)vs_dll_sym( hDllInstance,
	VSCORE_QUERYCONTROLINTERFACE_NAME );
	VSInitProc( true, true, "", 0, "", 0,NULL);
	printf("init starcore success\n");
	SRPControlInterface = QueryControlInterfaceProc();
	BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);

	if( BasicSRPInterface != NULL ){
		BasicSRPInterface -> Release();
		SRPControlInterface -> Release();
	}
	VSTermProc();
	vs_dll_close(hDllInstance);
	return;
}
```

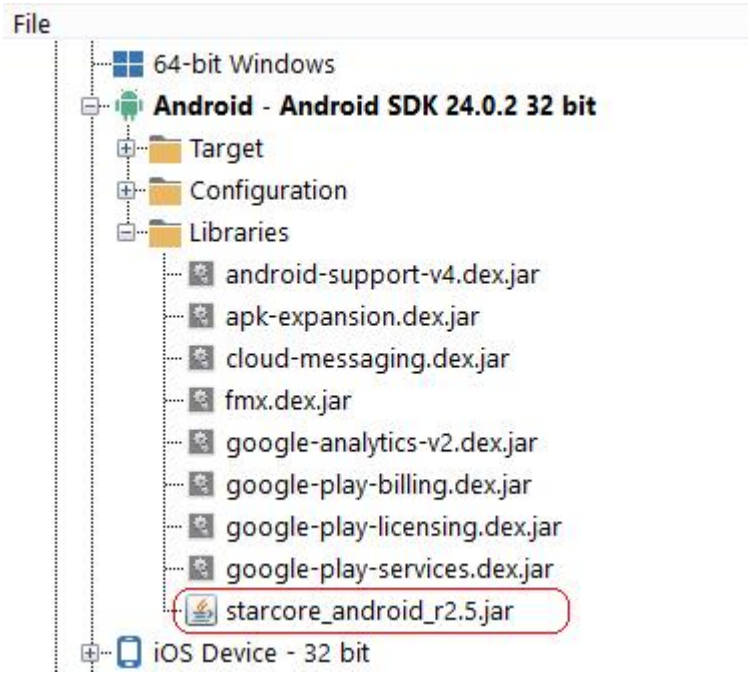
7.2.2 Init CLE(Second Method)

7.2.2.1 Deployment



Local Path	Local Name	Type	Platforms	Remote Path	Remote Name	Remote Status
$(BDS)\bin\Artwor...	FM_SplashImage_640...	Android_Spla...	[Android]	res\drawable-large\	splash_image.png	Not Connected
$(BDS)\bin\Artwor...	FM_LauncherIcon_144...	Android_Lau...	[Android]	res\drawable-xxhdpi\	ic_launcher.png	Not Connected
C:\xe8\c++builder....	classes.dex	AndroidClass...	[Android]	classes\	classes.dex	Not Connected
.\Android\Debug\	styles.xml	AndroidSplas...	[Android]	res\values\	styles.xml	Not Connected
..\..\starcore_for_a...	libstar_java.so	File	[Android]	library\lib\armeabi-v7...	libstar_java.so	Not Connected
$(BDS)\bin\Artwor...	FM_SplashImage_426...	Android_Spla...	[Android]	res\drawable-small\	splash_image.png	Not Connected
$(BDS)\bin\Artwor...	FM_LauncherIcon_96x...	Android_Lau...	[Android]	res\drawable-xhdpi\	ic_launcher.png	Not Connected
$(NDKBasePath)\p...	gdbserver	AndroidGDB...	[Android]	library\lib\armeabi-v7...	gdbserver	Not Connected
.\Android\Debug\	libcallluaexample.so	ProjectOutput	[Android]	library\lib\armeabi-v7...	libcallluaexample.so	Not Connected
$(BDS)\bin\Artwor...	FM_LauncherIcon_36x...	Android_Lau...	[Android]	res\drawable-ldpi\	ic_launcher.png	Not Connected
$(BDS)\lib\android...	libnative-activity.so	AndroidLibna...	[Android]	library\lib\armeabi\	libProject1.so	Not Connected
$(BDS)\bin\Artwor...	FM_LauncherIcon_72x...	Android_Lau...	[Android]	res\drawable-hdpi\	ic_launcher.png	Not Connected
$(BDS)\bin\Artwor...	FM_SplashImage_470...	Android_Spla...	[Android]	res\drawable-normal\	splash_image.png	Not Connected
C:\srplab\example...	classes.dex	AndroidClass...	[Android]	classes\	classes.dex	Not Connected
$(BDS)\bin\Artwor...	FM_SplashImage_960...	Android_Spla...	[Android]	res\drawable-xlarge\	splash_image.png	Not Connected
$(BDS)\lib\android...	libnative-activity.so	AndroidLibna...	[Android]	library\lib\mips\	libProject1.so	Not Connected
.\Android\Debug\	splash_image_def.xml	AndroidSplas...	[Android]	res\drawable\	splash_image_def.xml	Not Connected
$(BDS)\lib\android...	libnative-activity.so	AndroidLibna...	[Android]	library\lib\x86\	libProject1.so	Not Connected
$(BDS)\bin\Artwor...	FM_LauncherIcon_48x...	Android_Lau...	[Android]	res\drawable-mdpi\	ic_launcher.png	Not Connected
..\..\starcore_for_a...	libstarcore.so	File	[Android]	library\lib\armeabi-v7...	libstarcore.so	Not Connected
.\Android\Debug\	AndroidManifest.xml	ProjectAndro...	[Android]	.\	AndroidManifest.xml	Not Connected
C:\Users\srpla\Do...	classes.dex	AndroidClass...	[Android]	classes\	classes.dex	Not Connected

and add "starcore_android_rX-X.jar" to Libraries



Load libstarcore.so from java code.

```
//---------------------------------------------------------------------------
#include <fmx.h>
#pragma hdrstop

#if defined(__ANDROID__)
#include <Androidapi.JNI.GraphicsContentViewText.hpp>
#include <Androidapi.JNI.Net.hpp>
#include <Androidapi.Helpers.hpp>
#include <Androidapi.JNIBridge.hpp>
#include <Androidapi.JNI.JavaTypes.hpp>
#include <Androidapi.Jni.hpp>
```

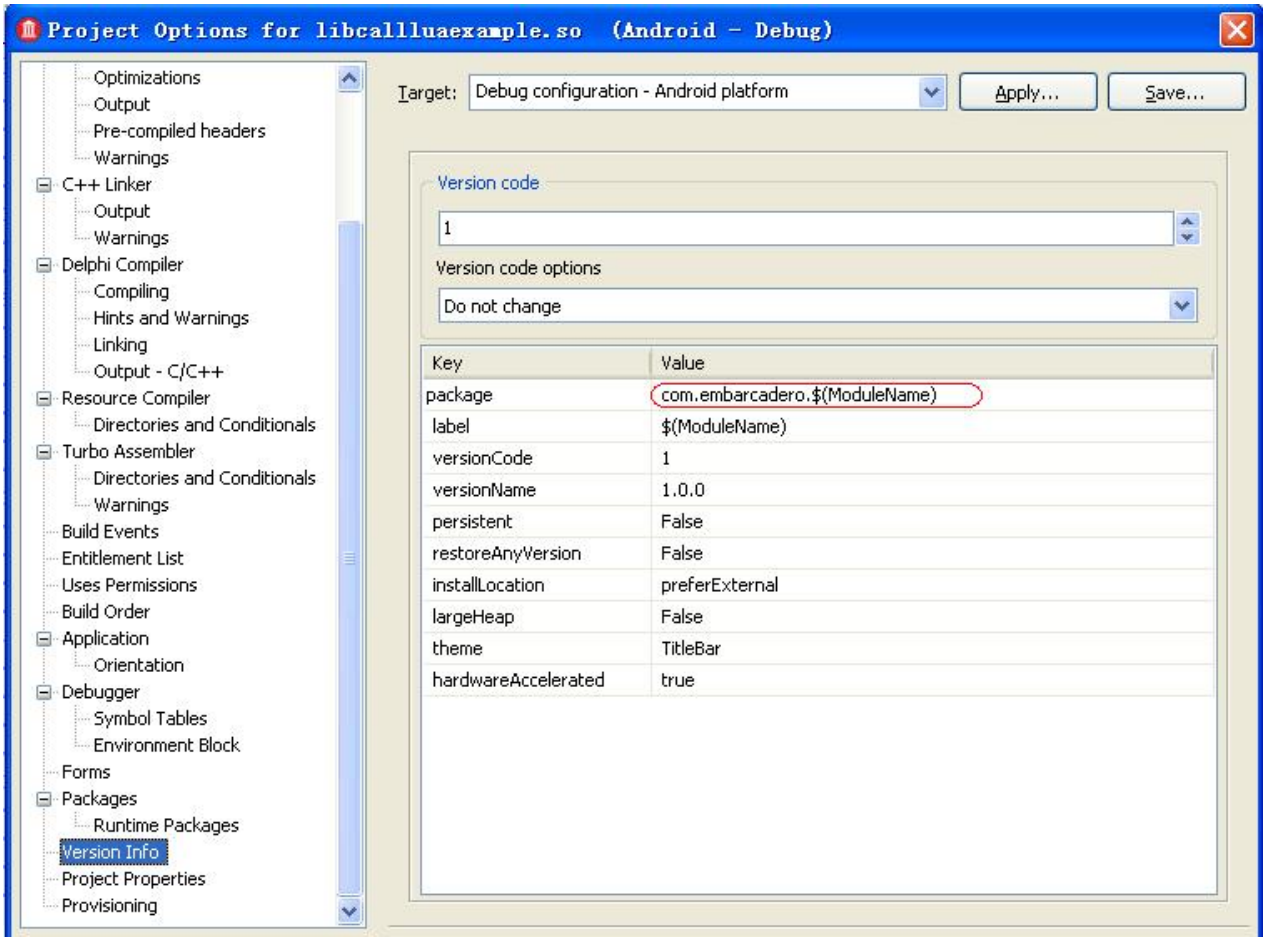
```
#include <Androidapi.JNI.Dalvik.hpp>
#include <FMX.Helpers.Android.hpp>

#include "starcore.h"

//PackageName : com.embarcadero.usingcle
//DexFileName : starcore_android_r2.2.dex
static JNIEnv *JavaEnv;
static jobject starcoreobj;
#if !defined(USE_DEXFILE)
static jclass jFactoryClass;
#else
static _di_Jlang_Class jFactoryClass;
#endif


#if !defined(USE_DEXFILE)
//---add starcore_android_rX.X.jar to "Android -> Libraries"
VS_HANDLE XE_StarCore_Init(const VS_CHAR *PackageName)
{
	VS_HANDLE hDllInstance;
	VS_CHAR Buf[256];

	JavaEnv = TJNIResolver::GetJNIEnv();

	jclass jLoadedClass = (jclass)TJNIResolver::ClassLoader->LoadClass("com/srplab/www/starcore/StarCoreFactoryPath");
	jFactoryClass = (jclass)TJNIResolver::ClassLoader->LoadClass("com/srplab/www/starcore/StarCoreFactory");
	//jclass jLoadedClass = JavaEnv->FindClass("com/srplab/www/starcore/StarCoreFactoryPath");
	//jFactoryClass = JavaEnv->FindClass("com/srplab/www/starcore/StarCoreFactory");
	if( jFactoryClass == NULL || jLoadedClass == NULL )
		return NULL;
	//------------------------------------------------------------
	sprintf(Buf,"/data/data/%s/lib",PackageName);
	jstring FieldName = JavaEnv->NewStringUTF(Buf);

	jfieldID jFieldID = JavaEnv->GetStaticFieldID(jLoadedClass,"StarCoreCoreLibraryPath","Ljava/lang/String;");
	JavaEnv->SetStaticObjectField(jLoadedClass,jFieldID,FieldName);

	jFieldID = JavaEnv->GetStaticFieldID(jLoadedClass,"StarCoreShareLibraryPath","Ljava/lang/String;");
	JavaEnv->SetStaticObjectField(jLoadedClass,jFieldID,FieldName);

	_di_JActivity activity = SharedActivity();
	jFieldID = JavaEnv->GetStaticFieldID(jLoadedClass,"ActivityObject","Ljava/lang/Object;");
	JavaEnv->SetStaticObjectField(jLoadedClass,jFieldID,(jobject)((_di_ILocalObject)activity) -> GetObjectID());
```

```
	//---call java init function
	jmethodID jMethodID = JavaEnv->GetStaticMethodID(jFactoryClass,"GetFactory","()Lcom/srplab/www/starcore/StarCoreFactory;");
	starcoreobj = JavaEnv -> CallStaticObjectMethod(jFactoryClass,jMethodID);
	if( starcoreobj == NULL )
		return NULL; //---failed
	jMethodID = JavaEnv->GetMethodID(jFactoryClass,"_CoreHandle","()J");
	VS_INT64 IntTemp = JavaEnv->CallLongMethod(starcoreobj,jMethodID);
	if( IntTemp == 0 )
		return NULL;
	hDllInstance = (VS_HANDLE)IntTemp;
	//----init starcore
	jMethodID = JavaEnv->GetMethodID(jFactoryClass,"_InitCore","(ZZZZLjava/lang/String;ILjava/lang/String;I)I");
	jvalue jvs[8];

	jvs[0].z = 1;
	jvs[1].z = 0;
	jvs[2].z = 0;
	jvs[3].z = 1;
	jvs[4].l = JavaEnv->NewStringUTF("");
	jvs[5].i = 0; //PortNumberForDebug;
	jvs[6].l = JavaEnv->NewStringUTF("");
	jvs[7].i = 0; //PortNumberForDirectClient;
	IntTemp = JavaEnv -> CallIntMethodA(starcoreobj,jMethodID,jvs);
	if( IntTemp == -1 )
		return NULL;
	return hDllInstance;
}
#else
VS_HANDLE XE_StarCore_Init(const VS_CHAR *PackageName,const VS_CHAR *DexFileName)
{
	VS_HANDLE hDllInstance;
	VS_CHAR Buf[256];

	JavaEnv = TJNIResolver::GetJNIEnv();
	_di_JContext context = SharedActivityContext();
	_di_JString dexpath_jstring,optimizedpath_jstring;
	sprintf(Buf,"/data/data/%s/files/%s",PackageName,DexFileName);
	dexpath_jstring = StringToJString(Buf);
	_di_JFile optimizedpath_jfile = context -> getDir(StringToJString("outdex"),TJContext::JavaClass->MODE_PRIVATE);
	optimizedpath_jstring = optimizedpath_jfile->getAbsolutePath();
	_di_JDexClassLoader cl = TJDexClassLoader::JavaClass->init(dexpath_jstring, optimizedpath_jstring,NULL,
TJDexClassLoader::JavaClass->getSystemClassLoader());
	if( cl == NULL )
		return NULL; //---failed
```

```
	_di_Jlang_Class jLoadedClass = cl -> loadClass(StringToJString("com/srplab/www/starcore/StarCoreFactoryPath"));
	jFactoryClass = cl -> loadClass(StringToJString("com/srplab/www/starcore/StarCoreFactory"));
	if( jFactoryClass == NULL )
		return NULL;
	//------------------------------------------------------------
	sprintf(Buf,"/data/data/%s/lib",PackageName);
	jstring FieldName = JavaEnv->NewStringUTF(Buf);

	jfieldID jFieldID = JavaEnv->GetStaticFieldID((jclass)((_di_ILocalObject)jLoadedClass) ->
GetObjectID(),"StarCoreCoreLibraryPath","Ljava/lang/String;");
	JavaEnv->SetStaticObjectField((jclass)((_di_ILocalObject)jLoadedClass) -> GetObjectID(),jFieldID,FieldName);

	jFieldID = JavaEnv->GetStaticFieldID((jclass)((_di_ILocalObject)jLoadedClass) ->
GetObjectID(),"StarCoreShareLibraryPath","Ljava/lang/String;");
	JavaEnv->SetStaticObjectField((jclass)((_di_ILocalObject)jLoadedClass) -> GetObjectID(),jFieldID,FieldName);

	_di_JActivity activity = SharedActivity();
	jFieldID = JavaEnv->GetStaticFieldID((jclass)((_di_ILocalObject)jLoadedClass) ->
GetObjectID(),"ActivityObject","Ljava/lang/Object;");
	JavaEnv->SetStaticObjectField((jclass)((_di_ILocalObject)jLoadedClass) ->
GetObjectID(),jFieldID,(jobject)((_di_ILocalObject)activity) -> GetObjectID());

	//---call java init function
	jmethodID jMethodID = JavaEnv->GetStaticMethodID((jclass)((_di_ILocalObject)jFactoryClass) ->
GetObjectID(),"GetFactory","()Lcom/srplab/www/starcore/StarCoreFactory;");
	starcoreobj = JavaEnv -> CallStaticObjectMethod((jclass)((_di_ILocalObject)jFactoryClass) -> GetObjectID(),jMethodID);
	if( starcoreobj == NULL )
		return NULL; //---failed
	jMethodID = JavaEnv->GetMethodID((jclass)((_di_ILocalObject)jFactoryClass) -> GetObjectID(),"_CoreHandle","()J");
	VS_INT64 IntTemp = JavaEnv->CallLongMethod(starcoreobj,jMethodID);
	if( IntTemp == 0 )
		return NULL;
	hDllInstance = (VS_HANDLE)IntTemp;
	//----init starcore
	jMethodID = JavaEnv->GetMethodID((jclass)((_di_ILocalObject)jFactoryClass) ->
GetObjectID(),"_InitCore","(ZZZZLjava/lang/String;ILjava/lang/String;I)I");
	jvalue jvs[8];

	jvs[0].z = 1;
	jvs[1].z = 0;
	jvs[2].z = 0;
	jvs[3].z = 1;
	jvs[4].l = JavaEnv->NewStringUTF("");
	jvs[5].i = 0; //PortNumberForDebug;
	jvs[6].l = JavaEnv->NewStringUTF("");
```

```
        jvs[7].i = 0; //PortNumberForDirectClient;
        IntTemp = JavaEnv -> CallIntMethodA(starcoreobj,jMethodID,jvs);
        if( IntTemp == -1 )
                return NULL;
        return hDllInstance;
}
#endif

void XE_VSCore_Term( )
{
#if !defined(USE_DEXFILE)
        jmethodID jMethodID = JavaEnv->GetMethodID(jFactoryClass,"_ModuleExit","()V");
#else
        jmethodID jMethodID = JavaEnv->GetMethodID((jclass)((_di_ILocalObject)jFactoryClass) ->
GetObjectID(),"_ModuleExit","()V");
#endif
        JavaEnv -> CallIntMethod(starcoreobj,jMethodID);
        return;
}

void XE_VSCore_TermEx( )
{
#if !defined(USE_DEXFILE)
        jmethodID jMethodID = JavaEnv->GetMethodID(jFactoryClass,"_ModuleExit","()V");
#else
        jmethodID jMethodID = JavaEnv->GetMethodID((jclass)((_di_ILocalObject)jFactoryClass) ->
GetObjectID(),"_ModuleExit","()V");
#endif
        JavaEnv -> CallIntMethod(starcoreobj,jMethodID);
        return;
}

#endif
```

PackageName is a string like " com.embarcadero.callluaexample". You can find it from project options.



7.2.2.2 Init CLE

Using the following code to init starcore.

```
#if defined(__ANDROID__)
	hDllInstance = XE_StarCore_Init("com.embarcadero.callluaexample");
	QueryControlInterfaceProc =
(VSCore_QueryControlInterfaceProc)vs_dll_sym( hDllInstance,VSCORE_QUERYCONTROLINTERFACE_NAME );
	SRPControlInterface = QueryControlInterfaceProc();
#endif
	BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);


	INIT_UUID( ServiceID );
	BasicSRPInterface -> CreateService( "","test",&ServiceID,"123",0,0,0,0,0,0 );
	SRPInterface = BasicSRPInterface -> GetSRPInterface("test","root","123");
```

7.2.2.3 Call android java code from lua

```
SrvGroup=libstarcore:_GetSrvGroup(0)
Service=SrvGroup:_GetService("","")
```

```
print(Service)
print(Service.TestClass)
Service.TestClass:Add(123.4,456.7)

SrvGroup:_InitRaw("java",Service)
TestJava = Service:_ImportRawContext("java","android/graphics/Color",true,nil);
print(TestJava)

ActivityJava = Service:_ImportRawContext("java","com/srplab/www/starcore/StarCoreFactoryPath",true,nil);
print(ActivityJava.StarCoreCoreLibraryPath)
print(ActivityJava.ActivityObject)

print(ActivityJava.ActivityObject:getTitle())
print(ActivityJava.ActivityObject:getPreferences(0))
```

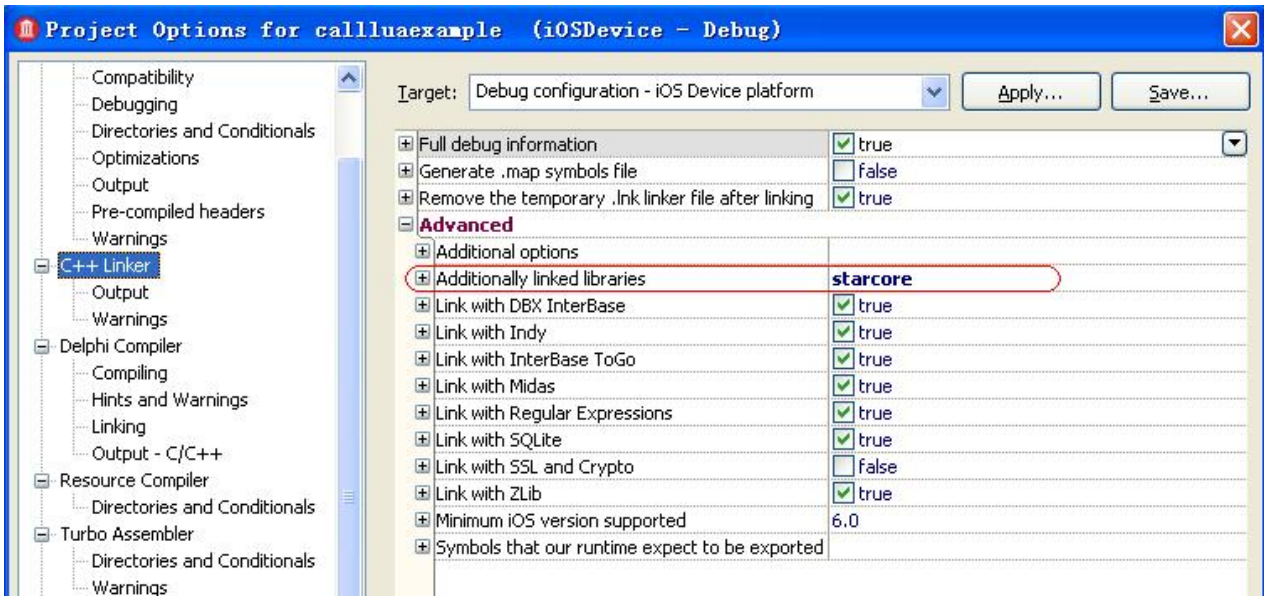
7.3 Using cle with c++ builder on ios

C++Builder does not support running iOS apps on the iOS Simulator

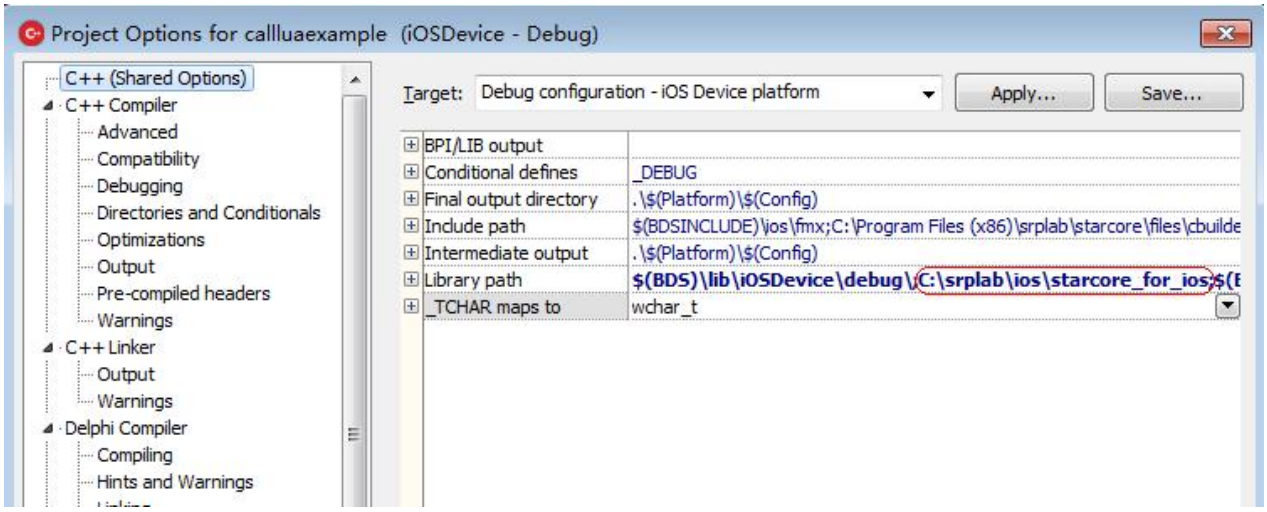
Note : Using CLE with c++ builder has great limit, for "dlsym" function always returns 0.

7.3.1 Init CLE

Add "starcore" to the project.



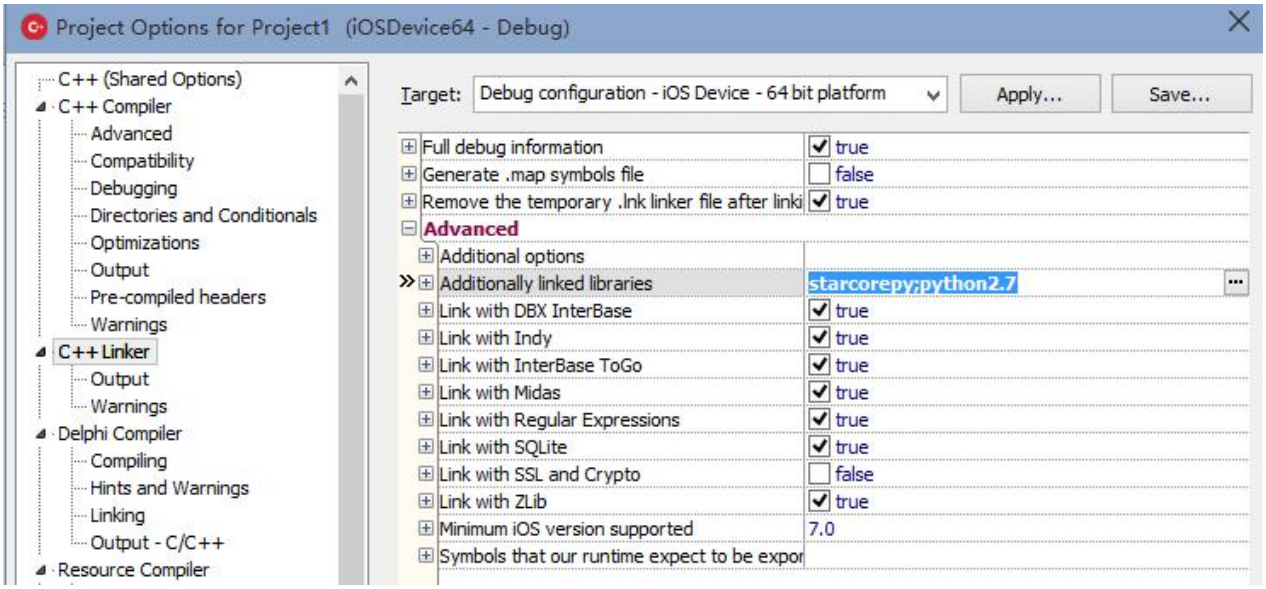
Set library search path :

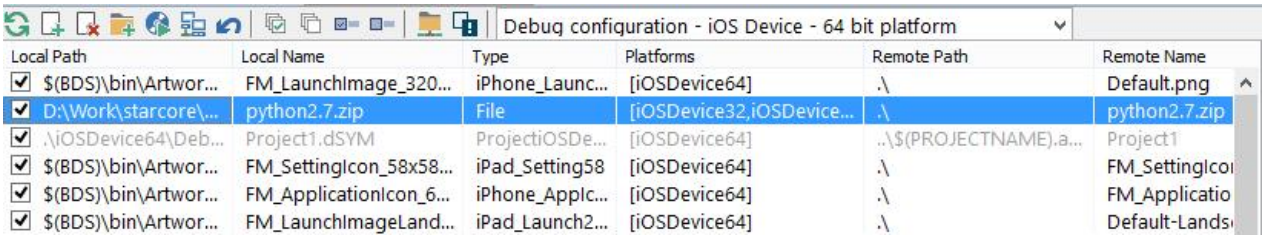


Using the following code to init cle.

```
#if defined(TARGET_OS_IPHONE)
	VS_CHAR documentpath[256];
	VS_CHAR coreDirectory[256];

	AnsiString Cstr = System::Ioutils::TPath::GetDocumentsPath();
	strcpy(documentpath,Cstr.c_str());

	Cstr = System::Ioutils::TPath::GetHomePath();
	sprintf(coreDirectory,"%s/callluaexample.app",Cstr.c_str());

	VS_BOOL Result = StarCore_InitEx(documentpath,coreDirectory);
	VSCore_Init( true, true, "", 0, "", 0,NULL);
	SRPControlInterface = VSCore_QueryControlInterface();
#endif
	BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);

	INIT_UUID( ServiceID );
	BasicSRPInterface -> CreateService( "","test",&ServiceID,"123",0,0,0,0,0,0 );
	SRPInterface = BasicSRPInterface -> GetSRPInterface("test","root","123");
```

7.3.2 Use python






```
#include "vsopenapi.h"

static VS_UWORD MsgCallBack(VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_UWORD wParam, VS_UWORD
lParam, VS_BOOL *IsProcessed, VS_UWORD Para)
{
   switch( uMsg ){
   case MSG_VSDISPMSG :
       case MSG_VSDISPLUAMSG :
              printf("[core]%s\n",(VS_CHAR *)wParam);
              Form1->Memo1->Lines->Add((VS_CHAR *)wParam);
      break;
   case MSG_DISPMSG :
       case MSG_DISPLUAMSG :
              printf("%s\n",(VS_CHAR *)wParam);
              Form1->Memo1->Lines->Add((VS_CHAR *)wParam);
      break;
   }
   return 0;
}

static class ClassOfSRPInterface *SRPInterface;
extern "C" SRPDLLEXPORT void *star_lrp_GetExportFunctionTable( );
```

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
	VS_CORESIMPLECONTEXT Context;

	VS_CHAR documentpath[256];
	VS_CHAR coreDirectory[256];

	AnsiString Cstr = System::Ioutils::TPath::GetDocumentsPath();
	strcpy(documentpath,Cstr.c_str());

	Cstr = System::Ioutils::TPath::GetHomePath();
	sprintf(coreDirectory,"%s/Test_lrp.app",Cstr.c_str());

	Form1->Memo1->Lines->Add(documentpath);
	Form1->Memo1->Lines->Add(coreDirectory);

	VS_BOOL Result = StarCore_InitEx(documentpath,coreDirectory);

	VS_CHAR python_path[512];
	VS_CHAR python_home[512];
	sprintf(python_home,"%s/python",coreDirectory);
	sprintf(python_path,"%s/python2.7.zip",coreDirectory);
	VSCoreLib_InitPython((VS_CHAR*)python_home,(VS_CHAR *)python_path,NULL);

	SRPInterface = VSCoreLib_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,NULL);
	SRPInterface -> CheckPassword(false);

	SRPInterface -> Release();
	VSCoreLib_TermSimple(&Context);
	return;
}
```

7.4 Using Variant to encapsulate cle object

Encapsulating cle object to Variant can simplify the script call. After change cle object to variant, you can use OleProcedure, OleFunction, OlePropertyGet, or OlePropertySet to call script function, get or set properties of script objects.

First, "StarCore_BC_Init(SRPInterface,CoreShellInterface) " should be called with service interface and shell interface. For example,

```
VS_CORESIMPLECONTEXT Context;
	SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,NULL);
	if( SRPInterface == NULL )
		return;
```

```
BasicSRPInterface = SRPInterface ->GetBasicInterface();
SRPControlInterface = BasicSRPInterface->GetSRPControlInterface();

CoreShellInterface = (class ClassOfCoreShellInterface *)SRPControlInterface->GetCoreShellInterface();

StarCore_BC_Init(SRPInterface,CoreShellInterface);
```

Encapsulating cle object to Variant

```
BasicSRPInterface->InitRaw("python35",SRPInterface);
void *python = SRPInterface->ImportRawContext("python","",false,NULL);

Variant varpython = SRPOBJECT_TOVARIANT(python,true);
```

Variant to cle object.

SRPVARIANT_TOPOINTER()

Variant has three additional functions:

ID() : which is used to get the cle object's uuid.

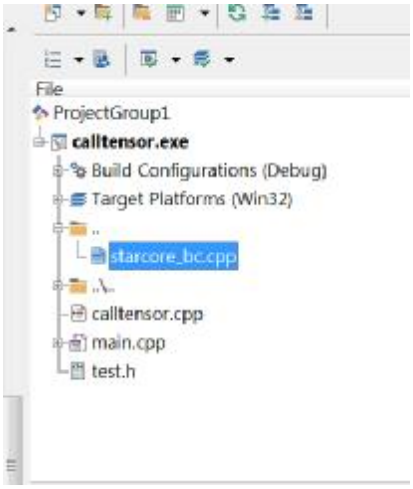
Create() or create(arg1,arg2,…) : which is used to create instance of class.

```
Variant ssss = varpython.OlePropertyGet("Multiply");
tt = ssss.OleFunction("create",33,44);
```

ToString() : which is used to get string of cle object.

7.4.1 How to use variant

Add starcore_bc.cpp to your project.



Sourc code file includes "starcore_bc.h "

```
#include "starcore_bc.h"
```

7.4.2 Sample Code

```
VS_CORESIMPLECONTEXT Context;
SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,NULL);
if( SRPInterface == NULL )
        return;
BasicSRPInterface = SRPInterface ->GetBasicInterface();
SRPControlInterface = BasicSRPInterface->GetSRPControlInterface();

CoreShellInterface = (class ClassOfCoreShellInterface *)SRPControlInterface->GetCoreShellInterface();

StarCore_BC_Init(SRPInterface,CoreShellInterface);

BasicSRPInterface->InitRaw("python35",SRPInterface);
void *python = SRPInterface->ImportRawContext("python","",false,NULL);

Variant varpython = SRPOBJECT_TOVARIANT(python,true);
String sss = "sys";
varpython.OleProcedure("import",sss);

varpython.OleProcedure("import","os");
Variant pythonos = varpython.OlePropertyGet("os");
sss =  pythonos.OleFunction("getcwd");

Variant sysvar = varpython.OlePropertyGet("sys");
Variant syspath = sysvar.OlePropertyGet("path");
sss = syspath.OleFunction("Get",0);

syspath.OleProcedure("Set",0,"aaaaaaaaaaaaaaaaaaaaaaaa");
sss = syspath.OleFunction("Get",0);

bool aaa = BasicSRPInterface->LoadRawModule("python","","..\\..\\testpy.py",false,NULL);
Variant tt = varpython.OleFunction("tt","hello ","world");

Variant cc = tt.OleFunction("Get",1);
```

7.4.3 Call tensorflow

```
//---------------------------------------------------------------------------

#include <fmx.h>
#pragma hdrstop
```

```
#include "main.h"
#include "starcore_bc.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.fmx"
TForm1 *Form1;


class ClassOfSRPControlInterface *SRPControlInterface = NULL;
class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
class ClassOfCoreShellInterface *CoreShellInterface = NULL;
class ClassOfSRPInterface *SRPInterface = NULL;

VS_UWORD MsgCallBack( VS_ULONG ServiceGroupID,VS_ULONG uMsg,VS_UWORD wParam,VS_UWORD
lParam,VS_BOOL* IsProcessed,VS_UWORD Para)
{
  String str;

  if( ( uMsg == MSG_VSDISPLUAMSG) || (uMsg == MSG_VSDISPMSG ) || (uMsg == MSG_DISPMSG ) || (uMsg ==
MSG_DISPLUAMSG ) )
  {
        str = TOVS_STRING((VS_CHAR *)wParam);
        if( str.Length() != 0 )
              Form1->Memo1->Lines->Add(str);
  }
  return 0;
}

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
      VS_CORESIMPLECONTEXT Context;
      SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,NULL);
      if( SRPInterface == NULL )
            return;
      BasicSRPInterface = SRPInterface ->GetBasicInterface();
      SRPControlInterface = BasicSRPInterface->GetSRPControlInterface();

      CoreShellInterface = (class ClassOfCoreShellInterface *)SRPControlInterface->GetCoreShellInterface();
```

```
    StarCore_BC_Init(SRPInterface,CoreShellInterface);

    BasicSRPInterface->InitRaw("python35",SRPInterface);
    void *python = SRPInterface->ImportRawContext("python","",false,NULL);

    Variant varpython = SRPOBJECT_TOVARIANT(python,true);

    varpython.OleProcedure("import","sys");

    Memo1->Lines->Add(varpython.OlePropertyGet("sys"));

    varpython.OleProcedure("eval","import tensorflow as tf");
    Variant tf = varpython.OlePropertyGet("tf");
    Memo1->Lines->Add(tf.OlePropertyGet("VERSION"));

//-- a = tf.add(2,5)
Variant a = tf.OleFunction("add",2, 5);
Memo1->Lines->Add(a);
//-- b = tf.multiply(a,5)
Variant b = tf.OleFunction("multiply",a, 3);
Memo1->Lines->Add(b);
//-- c = tf.constant(2,name="Node_c")
VS_PARAPKGPTR para = SRPInterface->GetParaPkgInterface();
para->InsertStr(0,"name");
para->InsertStr(1,"Node_c");
para->AsDict(true);
Variant c = tf.OleFunction("constant",2,SRPPARAPKG_TOVARIANT(para,false));
Memo1->Lines->Add(c);

//-- result = sess.run(b,feed_dict={a:25});
Variant sess = tf.OlePropertyGet("Session").OleFunction("create");

para->Clear;
para->InsertObject(0,SRPVARIANT_TOPOINTER(a));
para->InsertInt(1,25);
para->AsDict(true);

VS_PARAPKGPTR feed_dict = SRPInterface->GetParaPkgInterface();
feed_dict->InsertStr(0,"feed_dict");
feed_dict->InsertParaPackage(1,para);
feed_dict->AsDict(true);

Variant res = sess.OleFunction("run",b,SRPPARAPKG_TOVARIANT(feed_dict,false));
Memo1->Lines->Add(res);
```

```
}
//------------------------------------------------------------------------
```

7.5 Compile error for xe6/xe7

The file Posix.Errorno.hpp will cause compile error. You can annotate the following line.

```
//-- user supplied ---------------------------------------------------------

namespace Posix
{
namespace Errno
{
//-- type declarations ------------------------------------------------------
//-- var, const, procedure -------------------------------------------------
//static constexpr System::Int8 ENOTSUP = System::Int8(0x5f);
}       /* namespace Errno */
}       /* namespace Posix */
```

8 script raw operations

From version 2.0, cle supports script raw objects, which may be lua, python, java, or c# script objects. This feature can greatly simply operations between scripts. Modules or class libraries developed with scripts can be called by other scripts or c/c++ languages directly without encapsulating these raw script objects into cle objects.

The main difference of functions of raw objects and cle objects, in that for cle object's functions, the first parameter is always cle object self. For example,

void func(StarObjectClass self, int arg1) is a cle function.

void func(int arg1) is a raw script function.

In order to operate with scripts, the following steps should be followed.

1. call _InitRaw function to init special interface. The function has also a c version, which is included in SRPInterface class. The function should be called after cle service has been created.
2. call _LoadRawModule function to import script library.
3. call _ImportRawContext to get a cle object associated with a raw class or object.
4. The returned object can be operated same as normal cle object.

8.1 Special object and function for lua, python and c#.

lua = service:_ImportRawContext("lua","",false,"") to get global lua space

python = service:_ImportRawContext("pythn","",false,"") to get global python space

use _ImportRawContext("lua","{}",false,"") to create raw lua table

use _ImportRawContext("python","{}",false,"") to create raw python dict

use _ImportRawContext("python","[]XX",false,"") to create raw python list, XX is sizeof list

use _ImportRawContext("python","()XX",false,"") to create raw python tuple, XX is size of tuple

lua:

```
tab = Service:_ImportRawContext("lua", "{}", false, "");
tab[1] = "234"
tab["sadf"] = 345.66
```

python:

```
tab = Service._ImportRawContext("python", "{}", False, "");
tab._Set(1,"234")
tab._Set("sadf",345.66)
```

```
lis = Service._ImportRawContext("python", "[]", False, "");
lis._Set(0,"234")
lis._Set(1,345.66)
print(lis.__len__())
```

for each raw object of lua or python, there has a build-in function "_Eval", which input is a string.

for lua, a "return " statement will be added before executing the string. Example:

```
a=service:_ImportRawContext("lua","",false,"")
print( a:_Eval("2+2") )
```

for python, is same as PyRun_StringFlags with Py_eval_input as start parameter.

```
python=service._ImportRawContext("python","",False,"")
print(python._Eval("5 ** 2"))
```

For c#:

For csharp event attribute, cle will return a wrapper object which has two methods : Add and Remove. For example:

```
function button1:onClick(sender,e)
    print("Is Trigger");
    print(e.X);
```

```
    print(e.Y);
end
button1.Click:Add(button1:_NewRawProxyEx("","onClick","System.EventHandler"))
```

For lua:

there are also four build-in functions "eval","require","execute","executefile" for each lua raw object.

useage, for example:

lua:eval("2+2")

result = lua:execute("a=123")

result = lua:executefile("luafile.lua")

lua:require("os")

lua:eval equals to lua:_Eval

lua:execute(string) equals to SrvGroup:_RunScript("lua",string,"")

note: for lua, execute and executefile can return values.

For eval and execute command, the %@ of input string has special meaning. When the string is executed, the %@xxx will be replaced with the variable following the string one by one. For example,

Lua:eval("a=%@ccc",123)

1. Set global variable ccc to 123
2. Change the string to "a=ccc"
3. Executed the string.

For python:

there are also four build-in functions "eval","import","execute", "executefile" for each python raw object.

useage, for example:

python.eval("2+2")

python.execute("class Join:\n def __call__(self, *args):\n return '-'.join(args)")

python.executefile("pyfile.py")

python.import("os")

python.eval equals to python:_Eval

python.execute(string) equals to SrvGroup._RunScript("python",string,"")

For eval and execute command, the %@ of input string has special meaning. When the string is executed, the %@xxx will be replaced with the variable following the string one by one. For example,

1. Python.eval("a=%@ccc",123)
2. Set global variable ccc to 123

3. Change the string to "a=ccc"

Executed the string.

For ruby:

there are also four build-in functions "eval","require","execute", "executefile" for each ruby raw object.

useage, for example:

ruby.eval("2+2")

ruby.execute("aaa=123")

ruby.executefile("file.rb")

ruby.require("ssl")

ruby.eval equals to ruby._Eval

ruby.execute(string) equals to SrvGroup._RunScript("ruby",string,"")

For eval and execute command, the %@ of input string has special meaning. When the string is executed, the %@xxx will be replaced with the variable following the string one by one. For example,

1. Ruby:eval("a=%@ccc",123)
2. Set global variable ccc to 123
3. Change the string to "a=ccc"

Executed the string.

8.2 Parameters mapping between scripts.

int, float/double, bool, string, binbuf, parapkg, cle object are types for objects and functions of CLE.

For raw object, variable types are mapped into the above types.

for lua:

int <-> int

double <-> float

bool <-> bool

string <-> string /binbuf(for binary string)

table <-> parapkg/cle object(associated with raw object)

for python:

int <-> int

double <-> float

bool <-> bool

string <-> string /binbuf(for binary string)

tuple <-> parapkg

dict <-> cle object(associated with raw object)

list <-> cle object(associated with raw object)

object <-> cle object(associated with raw object)

note:

for parapkg, if it IsDict == true, then it will be mapped to dict.

Python set is not supported directly.

for ruby:

int <-> int

double <-> float

bool <-> bool

string <-> string /binbuf(for binary string)

array <-> parapkg

hash <-> cle object(associated with raw object)

object <-> cle object(associated with raw object)

note:

for parapkg, if it IsDict == true, then it will be mapped to hash.

for java:

int/byte/short/long <-> int

double <-> float

bool <-> bool

string <-> string

byte[] <-> binbuf

int[]/byte[]/bool[]/short[]/long[]/float[]/double[] <-> parapkg

object <-> cle object(associated with raw object)

for c#:

int/byte/short/long <-> int

double <-> float

bool <-> bool

string <-> string

byte[] <-> binbuf

int[]/byte[]/bool[]/short[]/long[]/float[]/double[] <-> parapkg

object <-> cle object(associated with raw object)

note : uint8 uint16 uint32 ulong are not supported for version 2.0

Instance of StarParaPkgClass and StarObjectClass is iterable.
For Star_csharp4/Star_csharp45/ Star_csharp451, instance of Star**Class is also dynamic object.

Note:
CLE object has a predefined attribute "_ReturnRawFlag", which is valid for raw object of lua and python. In normal case, lua table and python tuple is tried to be converted to parapkg. But if ReturnRawFlag == true, then, lua table and python tuple will be wrapped with cle object.

8.3 Parameters mapping between scripts as function input.

Ruby hash map to parapkg with IsDict == true
Python dict map to parapkg with IsDict == true
Java array map to parapkg with IsDict = false
C# array map to parapkg with IsDict =false

For example, ruby call python function
Python :

```
Def myfunc(a) :
   print a
```

ruby :
$python.myfunc({"aaaa"=>123})

{"aaaa"=>123} will be translate to dict as python function myfunc's input parameter.

8.4 Callback from script.

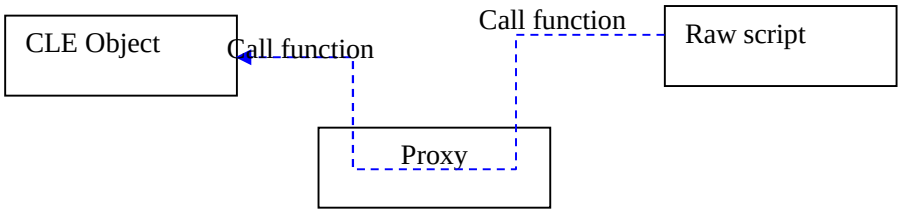
The callback of scripts may be a function for lua, function for python, an interface for java, a delegate for csharp. To handle the callbacks, proxy needs to be created, which can be called by raw script directly.

The proxy acts as a bridge from raw function to cle function. To create a proxy, a cle object should be created first.

Example:

```
object = Service:_New()
function object:click(arg)
   print(arg)
end
```

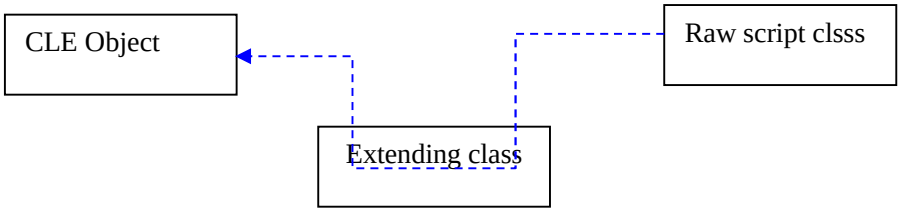
The relations of proxy, cle object, and raw script function is shown in below:



You can use the "_NewRawProxy" to create proxy for scripts. The function is explained in details in script interface document.

8.5 extend script class.

To extend class or override functions defined in raw script needs to create class dynamically. For some reason, the function is not supported for all scripts. Therefore, for version 2.0 of cle, extending classes using other scripts are not supported directly. If you want to extend class, you need to define extend class using original script, override its functions, and using cle object as a bridge to call other scripts.



An example of extending class of java or csharp is given in chapter "test java class extend" and "test cs class extend"

BaseClass

```
public class BaseClass
{
   public String getstr(String val)
   {
      return "ret from  base class";
   }
```

ExtendClass

```
package testsuper;
import com.srplab.www.starcore.*;

public class ExtendClass extends BaseClass
{
       private    StarServiceClass Service;
```

```
        private    String ObjectID;

        public ExtendClass(){
       StarCoreFactory starcore= StarCoreFactory.GetFactory();
       Service = starcore._StarCurrentService;  /*note: for c# (StarServiceClass)get_StarCurrentService() should be used instead.*/
       ObjectID = starcore._StarCurrentObject._GetStr("_ID"); /*note: for c# (StarObjectClass)get_StarCurrentObject() should be used instead.*/
       starcore._StarCurrentObject._LockGC();/*note: for c# (StarObjectClass)get_StarCurrentObject() should be used instead.*/
        }

    public void finalize() throws Throwable
    {
        try{
            StarObjectClass obj = Service._GetObjectEx(ObjectID);
            if( obj == null )
                return;
            obj._UnLockGC();
        }
        finally {
            super.finalize();
        }
    }

    public String _SuperStar_getstr(String val)
    {
        return super.getstr(val);
    }
    @Override
    public String getstr(String val){
        System.out.println("wrap class.....");
        StarObjectClass object = Service._GetObjectEx(ObjectID);
        if( object == null || object._IsFunctionDefined("_Star_getstr",true) == null )
            return super.getstr(val);
        return (String)object._Call("_Star_getstr",val);
    }
}
```

Bridge function for subclass to call super, should start with prefix "_SuperStar_"

function in subclass should start with "_Star_" for override function

For applications,

First: import ExtendClass.

ExtendClass = Service:_ImportRawContext("java","testextend/ExtendClass",true,nil);

then, override function.

```
function ExtendClass:_Star_getstr(input)
    print("cle class..... : ",input);
    return self:_SuperStar_getstr(input);
end
```

ExtendClass should define bridge functions with "_SuperStar_" prefix.

You can also create extending class code using function **_CreateRawProxyCode** and compile dynamically.This function is valid for python, java and csharp;

for example:

code = service._CreateRawProxyCode("python","","","ExtendBaseClass","BaseClass","__init__(self);getstr(self,val)","");

code = Service:_CreateRawProxyCode("java","","testextend.*","ExtendBaseClass","testextend/BaseClass","getstr,getmyclass,getstres,getmyclasses,getmyobjectes,getmyobject","testextend/ICallBack");

code = Service:_CreateRawProxyCode("csharp","","testextend","ExtendBaseClass","testextend.BaseClass","getstr,getmyclass,getstres,getmyclasses,getmyobjectes,getmyobject","testextend.ICallBack");

Example:

Compiling dynamically for csharp(written in lua)

```
Service=libstarcore._InitSimple("test","123",0,0,nil);
SrvGroup = Service._ServiceGroup;

csharp = "csharp"

SrvGroup:_InitRaw(csharp,Service);
SrvGroup:_LoadRawModule(csharp, "mscorlib", "", true);
SrvGroup:_LoadRawModule(csharp, "System", "", true);
SrvGroup:_LoadRawModule(csharp,"csextend","csextend.dll",false);
Simple = Service:_ImportRawContext(csharp,"Simple",true,nil);
s = Simple(10)
print(s)

delegateobj = Service:_New()
function delegateobj:X(i)
   return i + 100;
end
csdelegate = Service:_NewRawProxy(csharp,delegateobj,"X","Transformer",0);
```

```
print(s:Transform(csdelegate))

--create extend class
code = Service:_CreateRawProxyCode(csharp,"","","ExtendClass","Simple","getstr","");
BinBuf = SrvGroup:_NewBinBuf();
BinBuf:_Set(0,0,"S",code);
BinBuf:_SaveToFile("ExtendClass.cs",true);

--compile code dynamically
CSharpCodeProvider = Service:_ImportRawContext(csharp, "Microsoft.CSharp.CSharpCodeProvider", true, "");
CompilerParameters = Service:_ImportRawContext(csharp, "System.CodeDom.Compiler.CompilerParameters", true, "");

objCSharpCodePrivoder = CSharpCodeProvider();
objICodeCompiler = objCSharpCodePrivoder:CreateCompiler();
objCompilerParameters = CompilerParameters();

objCompilerParameters.ReferencedAssemblies:Add("System.dll");
--objCompilerParameters.ReferencedAssemblies:Add("System.Core.dll");
objCompilerParameters.ReferencedAssemblies:Add("c:\\srplab\\libs\\Star_csharp.dll");
objCompilerParameters.ReferencedAssemblies:Add("csextend.dll");
objCompilerParameters.GenerateExecutable = false;
objCompilerParameters.GenerateInMemory = true;

cr = objICodeCompiler:CompileAssemblyFromSource(objCompilerParameters, code);
if(cr.Errors.HasErrors == true ) then
   print("compile error.....",cr.Errors);
   err = cr.Errors
   for i=0, err.Count - 1 do
      print(err:get_Item(i).ErrorText)
   end
end

print( cr.CompiledAssembly)

--##########################################
Assembly = Service:_ImportRawContext(csharp, "System.Reflection.Assembly", true, "");
cleobject = Service:_New()
function cleobject:AssemblyResolve(sender, args)
   print("call back from cs");
   print(args.Name);
   return Assembly:LoadFrom("csextend.dll");
end
proxy = Service:_NewRawProxy(csharp,cleobject,"AssemblyResolve","System.ResolveEventHandler",0);
print(proxy)
AppDomain = Service:_ImportRawContext(csharp, "System.AppDomain", true, "");
```

```
currentDomain = AppDomain.CurrentDomain;
print(currentDomain);
DomainEvent = currentDomain.AssemblyResolve
print(DomainEvent)
DomainEvent:Add(proxy);

SrvGroup:_LoadRawModuleEx(csharp,"ExtendClass",cr.CompiledAssembly);

ExtendClass = Service:_ImportRawContext(csharp, "cle.ExtendClass", true, "");
print(ExtendClass);

SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

8.6 script files to be called

8.6.1 testlua.lua

```
function tt(a,b)
    print(a,b)
    return 6666,7777
end
g1 = 123
c={x=456}
function c:yy(a,b,z)
    print(self)
    print(a,b,z)
    return {33,Type="mytype"}
end
```

8.6.2 testpy.py

```
def tt(a,b) :
    print(a,b)
    return 666,777
g1 = 123
def yy(a,b,z) :
    print(a,b,z)
    return {'jack': 4098, 'sape': 4139}

class Multiply :
```

```
    def __init__(self,x,y) :
        self.a = x
        self.b = y

    def multiply(self,a,b):
        print("multiply....",self,a,b)
        return a * b
```

8.6.3 TestJava.java

```
package test;
public class TestJava
{
        public static int COUNT = 8;

        private String msg;
        private int[] counts;

        public TestJava(String msg,float num)
        {
                System.out.println("Demo...");
                System.out.println(num);
                this.msg = "default construct";
        }
        public String getMessage()
        {
                return msg;
        }
        public static String getHelloWorld()
        {
                return "Hello world!";
        }
        public String append(String str, int i)
        {
                return str + i;
        }
        public int[] getCounts()
        {
         return counts;
        }
        public void setCounts(int[] counts)
        {
         this.counts = counts;
        }
```

```
}
```

compile : javac test\TestJava.java

pack : jar cvf test.jar test*.class

The test file contains static field, static method, and normal fields and methods. Only public methods and fields can be accessed by other languages.

8.6.4 test java proxy

ICallBack.java

```
package testcallback;
public interface ICallBack
{
   void postExec();
   float getNum(float[] input);
}
```

TestCallBack.java

```
package testcallback;

public class TestCallBack
{
   private ICallBack callBack = null;
   public void setCallBack(ICallBack callBack){
      this.callBack = callBack;
   }
   public void postExec() throws RuntimeException{
      if(this.callBack == null)
         throw new RuntimeException("the call back must be defined~");
      this.callBack.postExec();
   }
   public float getNum(float[] input) throws RuntimeException{
      if(this.callBack == null)
         throw new RuntimeException("the call back must be defined~");
      Object Value = this.callBack.getNum(input);
      System.out.println(Value);
      return (Float)Value;
   }
   public void PrintInfo(Object...args){
      for( int i=0; i < args.length; i++ )
         System.out.println(""+args[i]);
   }
}
```

compile:

javac testcallback\ICallBack.java

javac testcallback\TestCallBack.java

pack to jar:

jar cvf testcallback.jar testcallback*.class

8.6.5 test java class extend

BaseClass

```
package testextend;
public class BaseClass
{
   public String getstr(String val)
   {
      System.out.println("base class.....");
      System.out.println(this);
      System.out.println(val);
      return "ret from  base class";
   }
}
```

ExtendClass

```
package testextend;
import com.srplab.www.starcore.*;

public class ExtendClass extends BaseClass
{
       private    StarServiceClass Service;
       private    String ObjectID;

       public ExtendClass(){
      StarCoreFactory starcore= StarCoreFactory.GetFactory();
      Service = starcore._StarCurrentService;
      ObjectID = starcore._StarCurrentObject._GetStr("_ID");
      starcore._StarCurrentObject._LockGC();
       }

   public void finalize() throws Throwable
   {
      try{
```

```
            StarObjectClass obj = Service._GetObjectEx(ObjectID);
            if( obj == null )
                return;
            obj._UnLockGC();
        }
        finally {
            super.finalize();
        }
    }

    public String _SuperStar_getstr(String val)
    {
        return super.getstr(val);
    }
    @Override
    public String getstr(String val){
        System.out.println("wrap class.....");
        StarObjectClass object = Service._GetObjectEx(ObjectID);
        if( object == null || object._IsFunctionDefined("_Star_getstr",true) == null )
            return super.getstr(val);
        return (String)object._Call("_Star_getstr",val);
    }
}
```

compile:

javac testextend\baseClass.java

javac testextend\ExtendClass.java

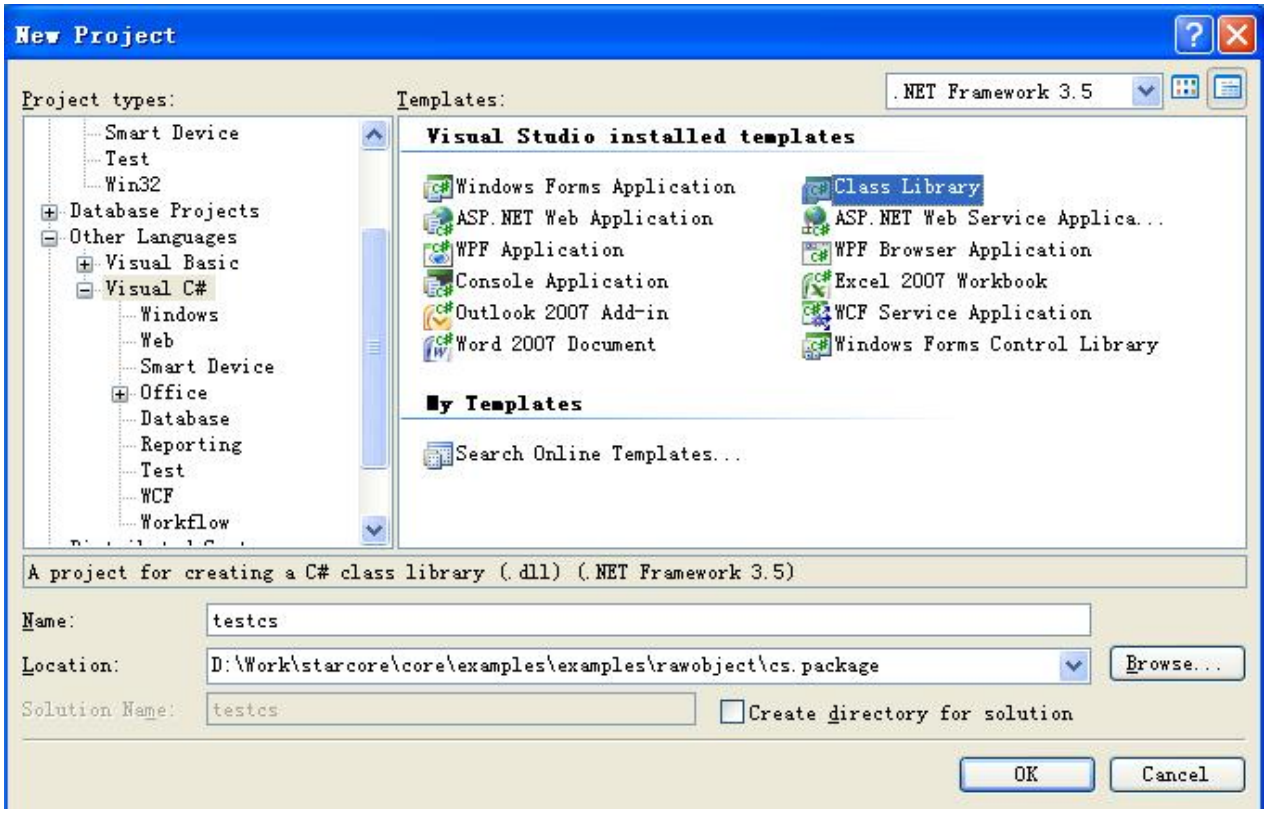
packtojar:

jar cvf testextend.jar testextend*.class

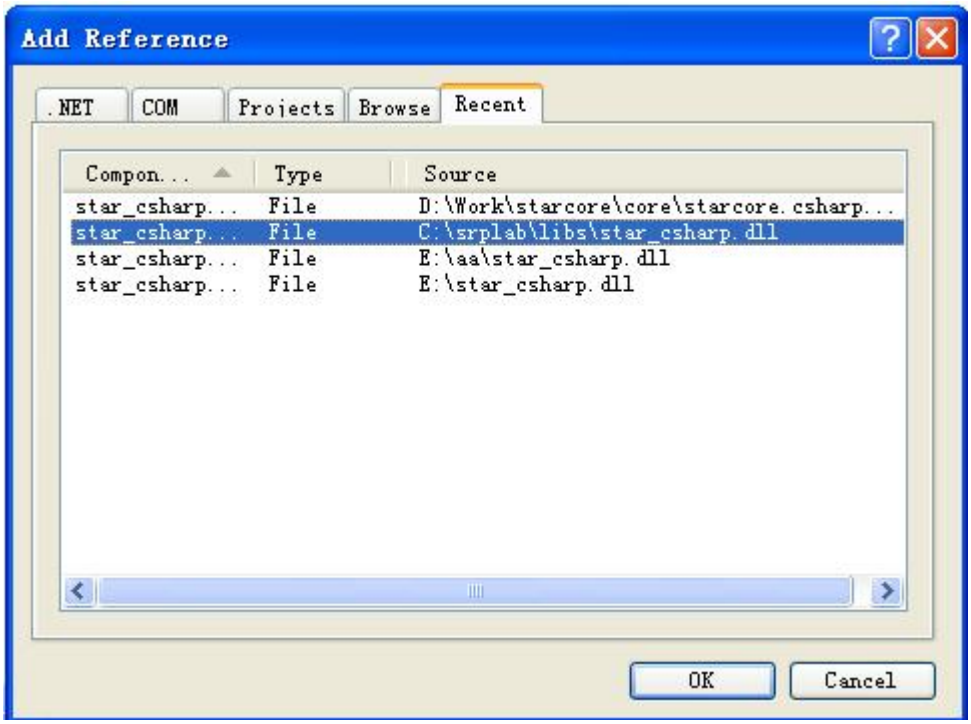
8.6.6 testcs

testcs is a class library of csharp to be called. Steps to create class library is as follow.

. Create project



. Add References



.source code

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using Star_csharp;

namespace testcs
{
   public class Class1
   {
          public static int COUNT = 8;

          private String msg;
          private int[] counts;

          public Class1(String msg,float num)
          {
                Console.WriteLine("Demo...");
           Console.WriteLine(num);
                this.msg = "default construct";
          }
          public String getMessage()
          {
                return msg;
          }
          public static String getHelloWorld()
          {
              return "Hello world!";
       }
       public String append(String str, int i)
          {
                return str + i;
          }
          public int[] getCounts()
          {
             return counts;
          }
          public void setCounts(int[] counts)
          {
             this.counts = counts;
          }
   }
}
```

command line:

```
csc /reference:c:\srplab\libs\Star_csharp.dll /platform:x86 XXXX.cs
```

8.6.7 test cs proxy

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace testcallback
{
    public delegate void postExec();
    public delegate float getNum(float[] input);

    public class Class1
    {
        private postExec callBack1 = null;
        private getNum callBack2 = null;
        public void setCallBack(postExec In_callBack1,getNum In_callBack2){
            this.callBack1 = In_callBack1;
            this.callBack2 = In_callBack2;
        }
        public void postExec(){
            if(this.callBack1 == null)
                throw new Exception("the call back must be definded~");
            this.callBack1();
        }
        public float getNum(float[] input){
            if(this.callBack2 == null)
                throw new Exception("the call back must be definded~");
            Object Value = this.callBack2(input);
            Console.WriteLine(Value);
            return (float)Value;
        }
    }
}
```

command line:

```
csc /reference:c:\srplab\libs\Star_csharp.dll /platform:x86 XXXX.cs
```

8.6.8 test cs class extend

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using Star_csharp;

namespace testextend
{
   public class BaseClass
   {
      public virtual String getstr(String val)
      {
         Console.WriteLine("base class.....");
         Console.WriteLine(this);
         Console.WriteLine(val);
         return "ret from  base class";
      }
   }

   public class ExtendClass : BaseClass
   {
      private StarServiceClass Service;
      private String ObjectID;

      public ExtendClass()
      {
         StarCoreFactory starcore = StarCoreFactory.GetFactory();
         Service = starcore.get_StarCurrentService();
         ObjectID = starcore.get_StarCurrentObject()._GetStr("_ID");
         starcore._StarCurrentObject._LockGC();
      }

      ~ExtendClass()
      {
         StarObjectClass obj = Service._GetObjectEx(ObjectID);
         if (obj == null)
            return;
         obj._UnLockGC();
      }

      public String _SuperStar_getstr(String val)
      {
         return base.getstr(val);
      }

      public override String getstr(String val)
      {
```

```
        Console.WriteLine("wrap class.....");
        StarObjectClass obj = Service._GetObjectEx(ObjectID);
        if (obj == null || obj._IsFunctionDefined("_Star_getstr", true) == null)
            return base.getstr(val);
        return (String)obj._Call("_Star_getstr", val);
      }
   }
}
```

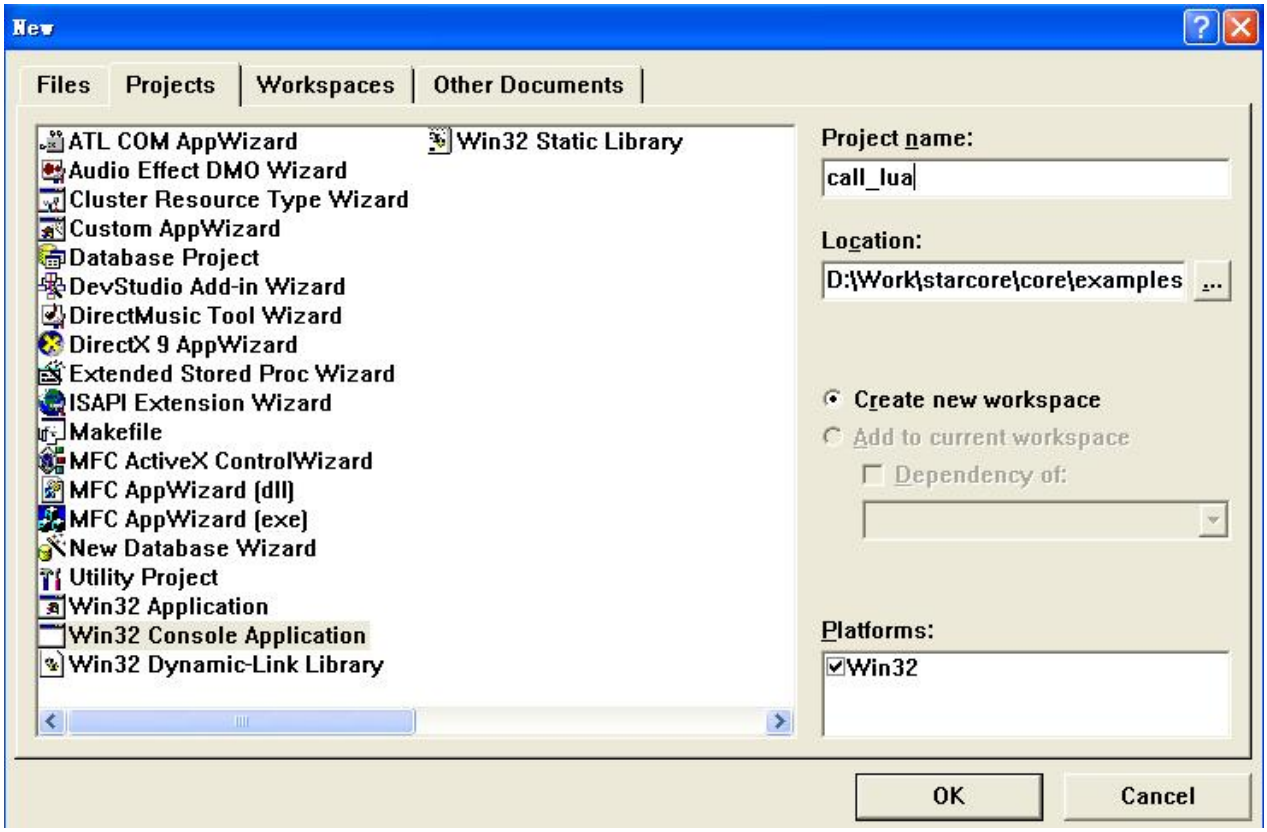
command line:

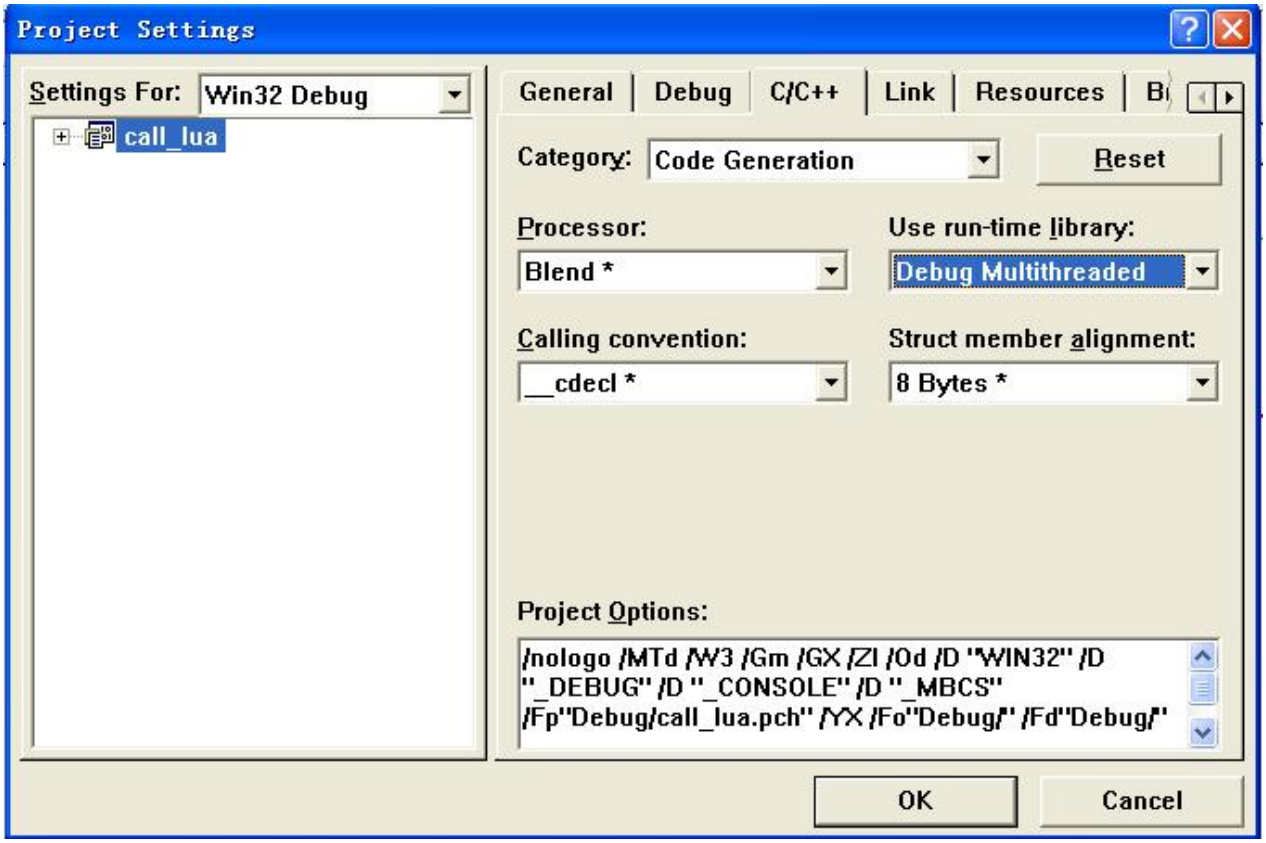
```
csc /reference:c:\srplab\libs\Star_csharp.dll /platform:x86 XXXX.cs
```

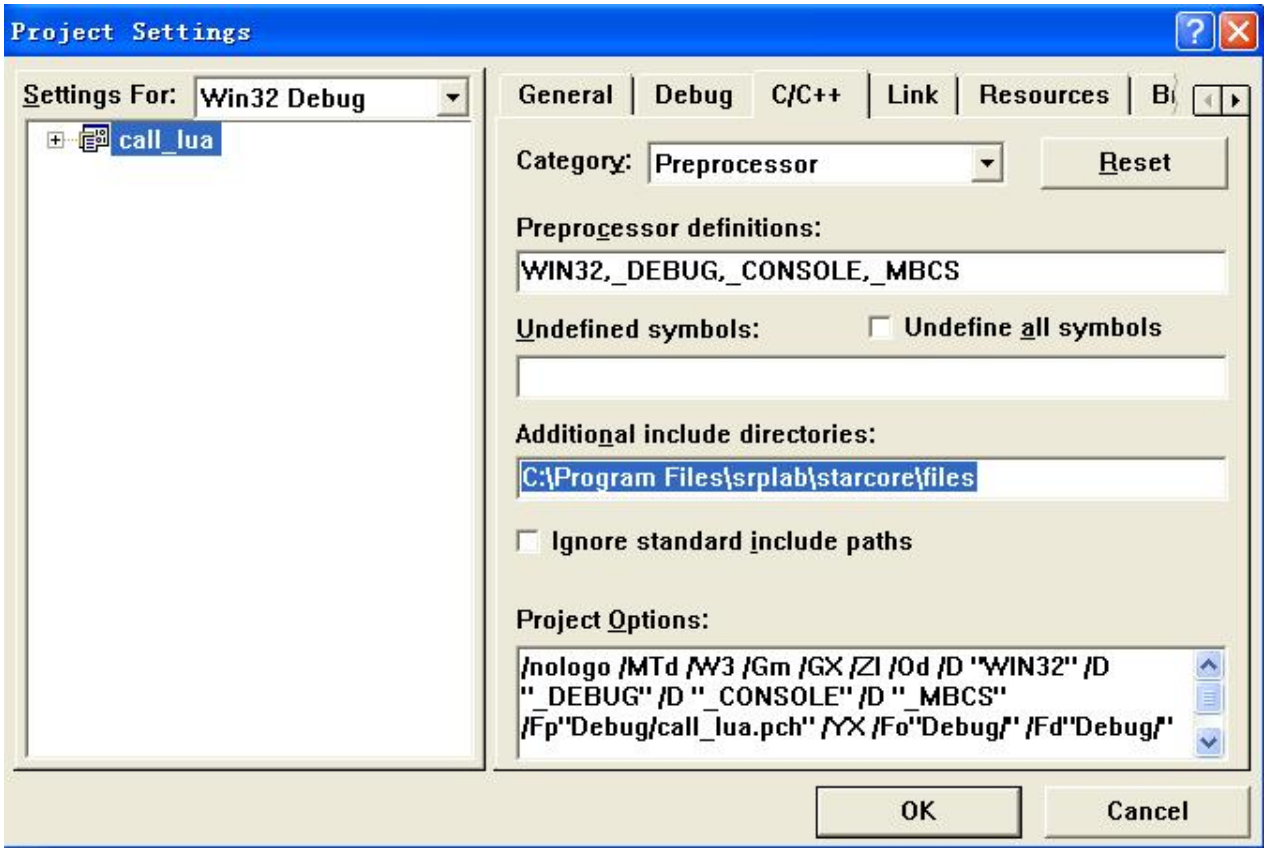
8.7 c/c++ call other raw script functions

8.7.1 call lua

8.7.1.1 create project




Project Settings
Settings For: Win32 Debug
call_lua
General
Debug
C/C++
Link
Resources
Category: Code Generation
Reset
Processor:
Blend *
Use run-time library:
Debug Multithreaded
Calling convention:
__cdecl *
Struct member alignment:
8 Bytes *
Project Options:
/nologo /MTd /W3 /Gm /GX /ZI /Od /D "WIN32" /D "_DEBUG" /D "_CONSOLE" /D "_MBCS" /Fp"Debug/call_lua.pch" /YX /Fo"Debug/" /Fd"Debug/"
OK
Cancel


Project Settings
Settings For: Win32 Debug
call_lua
General
Debug
C/C++
Link
Resources
Category: Preprocessor
Reset
Preprocessor definitions:
WIN32,_DEBUG,_CONSOLE,_MBCS
Undefined symbols:
Undefine all symbols
Additional include directories:
C:\Program Files\srplab\starcore\files
Ignore standard include paths
Project Options:
/nologo /MTd /W3 /Gm /GX /ZI /Od /D "WIN32" /D "_DEBUG" /D "_CONSOLE" /D "_MBCS" /Fp"Debug/call_lua.pch" /YX /Fo"Debug/" /Fd"Debug/"
OK
Cancel

Add one lib file from starlib_vcm/ starlib_vcm9/ starlib_vcm10/ starlib_vcm11.lib for VC6,VC2008,VC2010,VC2012 into the project.

For c++ builder, starlib_bc.lib file should be used.

8.7.1.2 source file

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
	class ClassOfSRPInterface *SRPInterface;
	class ClassOfBasicSRPInterface *BasicSRPInterface;
	VS_CORESIMPLECONTEXT Context;

	SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
	if( SRPInterface == NULL ){
		printf("init starcore fail\n");
		return -1;
	}
	printf("init starcore success\n");
	BasicSRPInterface = SRPInterface ->GetBasicInterface();
  /*----init lua raw interface ---*/
	BasicSRPInterface ->InitRaw("lua",SRPInterface);
	/*----load lua module ---*/
	BasicSRPInterface ->LoadRawModule("lua","","..\\lua.package\\testlua.lua",VS_FALSE,NULL);
	/*----attach object to global lua space ---*/
	void *Object = SRPInterface ->ImportRawContext("lua","",false,NULL);
	/*----call lua function tt, the return contains two integer, which will be packed into parapkg ---*/
	class ClassOfSRPParaPackageInterface *ParaPkg;
	ParaPkg = (class ClassOfSRPParaPackageInterface *)SRPInterface ->ScriptCall(Object,NULL,"tt","(ss)p","hello ","world");
	printf("ret from lua :  %d,  %d\n",ParaPkg->GetInt(0),ParaPkg->GetInt(1));
	/*----get global int value g1--------*/
	printf("lua value g1 :  %d\n",SRPInterface ->ScriptGetInt(Object,"g1"));
	/*----get global table value c, which is a table with function, it will be mapped to cle object --------*/
	void *c = (void *)SRPInterface ->ScriptGetObject(Object,"c",NULL);
  /*----get int value x from c--------*/
	printf("c value x :  %d\n",SRPInterface ->ScriptGetInt(c,"x"));
	/*----call c function yy, the return is a table, which will be mapped to cle object ---*/
	void *yy = (void *)SRPInterface ->ScriptCall(c,NULL,"yy","(osss)o",c,"hello ","world","!");
	printf("yy value [1] :  %d\n",SRPInterface ->ScriptGetIntIndex(yy,1));
	printf("yy value [Type] :  %s\n",SRPInterface ->ScriptGetStr(yy,"Type"));

	SRPInterface -> Release();
```

```
        VSCore_TermSimple(&Context);
        return 0;
}
```

8.7.2 call python

8.7.2.1 create project

skip

8.7.2.2 source file

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
        class ClassOfSRPInterface *SRPInterface;
        class ClassOfBasicSRPInterface *BasicSRPInterface;
        VS_CORESIMPLECONTEXT Context;

        SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
        if( SRPInterface == NULL ){
                printf("init starcore fail\n");
                return -1;
        }
        printf("init starcore success\n");
        BasicSRPInterface = SRPInterface ->GetBasicInterface();
   /*----init python raw interface ---*/
        BasicSRPInterface ->InitRaw("python",SRPInterface);
        /*----load python module ---*/
        BasicSRPInterface ->LoadRawModule("python","","..\\..\\python.package\\testpy.py",VS_FALSE,NULL);
        /*----attach object to global python space ---*/
        void *Object = SRPInterface ->ImportRawContext("python","",false,NULL);
        /*----call python function tt, the return contains two integer, which will be packed into parapkg ---*/
        class ClassOfSRPParaPackageInterface *ParaPkg;
        ParaPkg = (class ClassOfSRPParaPackageInterface *)SRPInterface ->ScriptCall(Object,NULL,"tt","(ss)p","hello ","world");
        printf("ret from python :  %d,  %d\n",ParaPkg->GetInt(0),ParaPkg->GetInt(1));
        /*----get global int value g1--------*/
        printf("python value g1 :  %d\n",SRPInterface ->ScriptGetInt(Object,"g1"));

        /*----call python function yy, the return is dict, which will be mapped to cle object ---*/
        void *yy = (void *)SRPInterface ->ScriptCall(Object,NULL,"yy","(ssi)o","hello ","world",123);
```

```
	/*----call dict __len__ function to get dict length ---*/
	printf("python value dict length :  %d\n",SRPInterface ->ScriptCall(yy,NULL,"__len__","()i"));

	/*----get global class Multiply --------*/
	void *Multiply = (void *)SRPInterface ->ImportRawContext("python","Multiply",VS_TRUE,NULL);
	/*----create instance of Multiply class, construct parameter should be packed in parapkg---*/
	ParaPkg = SRPInterface ->GetParaPkgInterface();
	ParaPkg ->InsertInt(0,33);
	ParaPkg ->InsertInt(1,44);
	void *multiply = SRPInterface ->IMallocObjectL(SRPInterface ->GetIDEx(Multiply),ParaPkg);
	/*----call instance method multiply --------*/
	printf("instance multiply = %d\n",SRPInterface ->ScriptCall(multiply,NULL,"multiply","(ii)i",11,22));

	SRPInterface -> Release();
	VSCore_TermSimple(&Context);
	return 0;
}
```

8.7.3 call java

8.7.3.1 create project

skip

8.7.3.2 source file

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
	class ClassOfSRPInterface *SRPInterface;
	class ClassOfBasicSRPInterface *BasicSRPInterface;
	VS_CORESIMPLECONTEXT Context;

	SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
	if( SRPInterface == NULL ){
		printf("init starcore fail\n");
		return -1;
	}
	printf("init starcore success\n");
	BasicSRPInterface = SRPInterface ->GetBasicInterface();
  /*----init java raw interface ---*/
```

```
	BasicSRPInterface ->InitRaw("java",SRPInterface);
	/*----load java module ---*/
	BasicSRPInterface ->LoadRawModule("java","","..\\..\\java.package\\test.jar",VS_FALSE,NULL);
	/*----attach object to global java space ---*/
	void *TestJava = SRPInterface ->ImportRawContext("java","test/TestJava",true,NULL);
	/*----get and set static field---*/
	SRPInterface ->ScriptSetInt(TestJava,"COUNT",7766);
	printf("java value COUNT :  %d\n",SRPInterface ->ScriptGetInt(TestJava,"COUNT"));
	/*----call static method---------*/
	printf("java getHelloWorld() :  %s\n",SRPInterface ->ScriptCall(TestJava,NULL,"getHelloWorld","()s"));

	/*----create instance of TestJava class----*/
	class ClassOfSRPParaPackageInterface *ParaPkg;
	ParaPkg = SRPInterface ->GetParaPkgInterface();
	ParaPkg ->InsertStr(0,"cle value");
	ParaPkg ->InsertInt(1,44);
	void *inst = SRPInterface ->IMallocObjectL(SRPInterface ->GetIDEx(TestJava),ParaPkg);

	/*----call normal function setCounts ---*/
	ParaPkg ->Clear();
	ParaPkg -> InsertInt(0,77);
	ParaPkg -> InsertInt(1,88);
	SRPInterface ->ScriptCall(inst,NULL,"setCounts","(p)",ParaPkg);

	class ClassOfSRPParaPackageInterface *ret;
	ret = (class ClassOfSRPParaPackageInterface *)SRPInterface ->ScriptCall(inst,NULL,"getCounts","()p");
	printf("ret from java :  %d,  %d\n",ParaPkg->GetInt(0),ParaPkg->GetInt(1));

	SRPInterface -> Release();
	VSCore_TermSimple(&Context);
	return 0;
}
```

8.7.4 call java with callback

8.7.4.1 create project

skip

8.7.4.2 source file

```
#include "vsopenapi.h"
```

```
void postExec(void *Object)
{
   printf("call back from java\n");
}

float getNum(void *Object,VS_PARAPKGPTR input)
{
   printf("call back [getNum]from java : %f,  %f\n",input->GetFloat(0),input->GetFloat(1));
       return input->GetFloat(0) + input->GetFloat(1);
}

int main(int argc, char* argv[])
{
       class ClassOfSRPInterface *SRPInterface;
       class ClassOfBasicSRPInterface *BasicSRPInterface;
       VS_CORESIMPLECONTEXT Context;

       SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
       if( SRPInterface == NULL ){
              printf("init starcore fail\n");
              return -1;
       }
       printf("init starcore success\n");
       BasicSRPInterface = SRPInterface ->GetBasicInterface();
   /*----init java raw interface ---*/
       BasicSRPInterface ->InitRaw("java",SRPInterface);
       /*----load java module ---*/
       BasicSRPInterface ->LoadRawModule("java","","..\\..\\java.package\\testcallback.jar",VS_FALSE,NULL);
       /*----attach object to testcallback/TestCallBack ---*/
       void *inst = SRPInterface ->IMallocObjectL(SRPInterface ->GetIDEx(TestCallBack),NULL);

       /*----create cle object associated with proxy ---*/

       /*---1 : create cleobjectclass and functions for proxy----*/
       void *cleobject = SRPInterface ->IMallocObjectL(NULL,NULL);
       void *AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(SRPInterface -
>ObjectToAtomic(cleobject),"postExec","()",NULL,NULL,VS_FALSE,VS_FALSE);
       SRPInterface ->SetAtomicFunction(AtomicFunction,(void *)postExec);
       AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(SRPInterface -
>ObjectToAtomic(cleobject),"getNum","(p)f",NULL,NULL,VS_FALSE,VS_FALSE);
       SRPInterface ->SetAtomicFunction(AtomicFunction,(void *)getNum);

       /*----create proxy for interface testcallback/ICallBack ---*/
       void *proxy = SRPInterface ->NewRawProxy("java",cleobject,NULL,"testcallback.ICallBack",0);
```

```
        /*----set the proxy to TestCallBack instance ---*/
        SRPInterface ->ScriptCall(inst,NULL,"setCallBack","(o)",proxy);
        /*----now proxy can be freed----*/
        SRPInterface ->FreeObject(proxy);

    /*----call inst function postExec----*/
        SRPInterface ->ScriptCall(inst,NULL,"postExec","()");
    /*----call inst function getNum----*/
        class ClassOfSRPParaPackageInterface *ParaPkg;
        ParaPkg = SRPInterface ->GetParaPkgInterface();
        ParaPkg ->InsertFloat(0,123);
        ParaPkg ->InsertFloat(1,456);
        printf("%f\n",SRPInterface ->ScriptFCall(inst,NULL,"getNum","(p)f",ParaPkg));

        SRPInterface -> Release();
        VSCore_TermSimple(&Context);
        return 0;
}
```

8.7.5 call java extend class

8.7.5.1 create project

skip

8.7.5.2 source file

```
#include "vsopenapi.h"

class ClassOfSRPInterface *SRPInterface;

char *_Star_getstr(void *Object,char *input)
{
    printf("cle class..... : %s\n",input);
        return (char *)SRPInterface ->ScriptCall(Object,NULL,"_SuperStar_getstr","(s)s",input);
}

int main(int argc, char* argv[])
{
        class ClassOfBasicSRPInterface *BasicSRPInterface;
        VS_CORESIMPLECONTEXT Context;
```

```
	SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
	if( SRPInterface == NULL ){
		printf("init starcore fail\n");
		return -1;
	}
	printf("init starcore success\n");
	BasicSRPInterface = SRPInterface ->GetBasicInterface();
  /*----init java raw interface ---*/
	BasicSRPInterface ->InitRaw("java",SRPInterface);
	/*----load java module ---*/
	BasicSRPInterface ->LoadRawModule("java","","..\\..\\java.package\\testextend.jar",VS_FALSE,NULL);
	/*----attach object to testextend/ExtendClass ---*/
	void *ExtendClass = SRPInterface ->ImportRawContext("java","testextend/ExtendClass",true,NULL);
	/*----create an instance of ExtendClass-----*/
	void *inst = SRPInterface ->IMallocObjectL(SRPInterface ->GetIDEx(ExtendClass),NULL);
	SRPInterface ->CreateAtomicFunctionSimpleEx(SRPInterface ->ObjectToAtomic(inst),"_Star_getstr","(s)s",(void
*)_Star_getstr,NULL);
	 /*----call function getstr---*/
	printf("%s\n",SRPInterface ->ScriptCall(inst,NULL,"getstr","(s)s","3333333"));

	SRPInterface -> Release();
	VSCore_TermSimple(&Context);
	return 0;
}
```

8.7.6 call cs

8.7.6.1 create project

skip

8.7.6.2 source file

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
	class ClassOfSRPInterface *SRPInterface;
	class ClassOfBasicSRPInterface *BasicSRPInterface;
	VS_CORESIMPLECONTEXT Context;

	SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
```

```
        if( SRPInterface == NULL ){
                printf("init starcore fail\n");
                return -1;
        }
        printf("init starcore success\n");
        BasicSRPInterface = SRPInterface ->GetBasicInterface();
    /*----init csharp raw interface ---*/
        BasicSRPInterface ->InitRaw("csharp",SRPInterface);
        /*----load csharp module ---*/
        BasicSRPInterface -
>LoadRawModule("csharp","testcs","..\\..\\cs.package\\testcs\\bin\\Debug\\testcs.dll",VS_FALSE,NULL);
        /*----attach object to testcs.Class1 ---*/
        void *Class1 = SRPInterface ->ImportRawContext("csharp","testcs.Class1",true,NULL);
        /*----get and set static field---*/
        SRPInterface ->ScriptSetInt(Class1,"COUNT",7766);
        printf("csharp value COUNT :  %d\n",SRPInterface ->ScriptGetInt(Class1,"COUNT"));
        /*----call static method---------*/
        printf("csharp getHelloWorld() :  %s\n",SRPInterface ->ScriptCall(Class1,NULL,"getHelloWorld","()s"));

        /*----create instance of Class1 class----*/
        class ClassOfSRPParaPackageInterface *ParaPkg;
        ParaPkg = SRPInterface ->GetParaPkgInterface();
        ParaPkg->Build("si","cle value",44);
        void *inst = SRPInterface ->IMallocObjectL(SRPInterface ->GetIDEx(Class1),ParaPkg);

        /*----call normal function setCounts ---*/
        ParaPkg->Build("ii",77,88);
        SRPInterface ->ScriptCall(inst,NULL,"setCounts","(p)",ParaPkg);

        ParaPkg ->Release();
        ParaPkg = (class ClassOfSRPParaPackageInterface *)SRPInterface ->ScriptCall(inst,NULL,"getCounts","()p");
        printf("ret from csharp :  %d,  %d\n",ParaPkg->GetInt(0),ParaPkg->GetInt(1));

        SRPInterface -> Release();
        VSCore_TermSimple(&Context);
        return 0;
}
```

8.7.7 call cs with callback

8.7.7.1 create project

skip

8.7.7.2 source file

```
#include "vsopenapi.h"

void postExec(void *Object)
{
   printf("call back from cs\n");
}

float getNum(void *Object,VS_PARAPKGPTR input)
{
   printf("call back [getNum]from cs : %f,  %f\n",input->GetFloat(0),input->GetFloat(1));
       return input->GetFloat(0) + input->GetFloat(1);
}

int main(int argc, char* argv[])
{
       class ClassOfSRPInterface *SRPInterface;
       class ClassOfBasicSRPInterface *BasicSRPInterface;
       VS_CORESIMPLECONTEXT Context;

       SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
       if( SRPInterface == NULL ){
              printf("init starcore fail\n");
              return -1;
       }
       printf("init starcore success\n");
       BasicSRPInterface = SRPInterface ->GetBasicInterface();
   /*----init cs raw interface ---*/
       BasicSRPInterface ->InitRaw("csharp",SRPInterface);
       /*----load cs module ---*/
       BasicSRPInterface -
>LoadRawModule("csharp","testcallback","..\\..\\cs.package\\testcallback\\bin\\Debug\\testcallback.dll",VS_FALSE,NULL);
       /*----attach object to testcallback.Class1 ---*/
       void *TestCallBack = SRPInterface ->ImportRawContext("csharp","testcallback.Class1",true,NULL);
       /*----create an instance of TestCallBack-----*/
       void *inst = SRPInterface ->IMallocObjectL(SRPInterface ->GetIDEx(TestCallBack),NULL);

       /*---1 : create cleobject and functions for proxy----*/
       void *cleobject = SRPInterface ->IMallocObjectL(NULL,NULL);
       SRPInterface ->CreateAtomicFunctionSimpleEx(SRPInterface ->ObjectToAtomic(cleobject),"postExec","()",(void
*)postExec,NULL);
```

```
        SRPInterface ->CreateAtomicFunctionSimpleEx(SRPInterface ->ObjectToAtomic(cleobject),"getNum","(p)f",(void
*)getNum,NULL);

        /*----create proxy for interface testcallback/ICallBack ---*/
        void *proxy1 = SRPInterface ->NewRawProxy("csharp",cleobject,"postExec","testcallback.postExec",0);
        void *proxy2 = SRPInterface ->NewRawProxy("csharp",cleobject,"getNum","testcallback.getNum",0);
        /*----set the proxy to TestCallBack instance ---*/
        SRPInterface ->ScriptCall(inst,NULL,"setCallBack","(oo)",proxy1,proxy2);
        /*----now proxy can be freed----*/
        SRPInterface ->FreeObject(proxy1);
        SRPInterface ->FreeObject(proxy2);

    /*----call inst function postExec----*/
        SRPInterface ->ScriptCall(inst,NULL,"postExec","()");
    /*----call inst function getNum----*/
        class ClassOfSRPParaPackageInterface *ParaPkg;
        ParaPkg = SRPInterface ->GetParaPkgInterface();
        ParaPkg ->Build("ff",123.0,456.0);
        printf("%f\n",SRPInterface ->ScriptFCall(inst,NULL,"getNum","(p)f",ParaPkg));

        SRPInterface -> Release();
        VSCore_TermSimple(&Context);
        return 0;
}
```

8.7.8 call cs extend class

8.7.8.1 create project

skip

8.7.8.2 source file

```
#include "vsopenapi.h"

class ClassOfSRPInterface *SRPInterface;

char *_Star_getstr(void *Object,char *input)
{
    printf("cle class..... : %s\n",input);
        return (char *)SRPInterface ->ScriptCall(Object,NULL,"_SuperStar_getstr","(s)s",input);
}
```

```
int main(int argc, char* argv[])
{
	class ClassOfBasicSRPInterface *BasicSRPInterface;
	VS_CORESIMPLECONTEXT Context;

	SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
	if( SRPInterface == NULL ){
		printf("init starcore fail\n");
		return -1;
	}
	printf("init starcore success\n");
	BasicSRPInterface = SRPInterface ->GetBasicInterface();
  /*----init cs raw interface ---*/
	BasicSRPInterface ->InitRaw("csharp",SRPInterface);
	/*----load csharp module ---*/
	BasicSRPInterface -
>LoadRawModule("csharp","testextend","..\\..\\cs.package\\testextend\\bin\\Debug\\testextend.dll",VS_FALSE,NULL);
	/*----attach object to testextend.ExtendClass ---*/
	void *ExtendClass = SRPInterface ->ImportRawContext("csharp","testextend.ExtendClass",true,NULL);
	/*----create an instance of ExtendClass-----*/
	void *inst = SRPInterface ->IMallocObjectL(SRPInterface ->GetIDEx(ExtendClass),NULL);
	SRPInterface ->CreateAtomicFunctionSimpleEx(SRPInterface ->ObjectToAtomic(inst),"_Star_getstr","(s)s",(void
*)_Star_getstr,NULL);
	 /*----call function getstr---*/
	printf("%s\n",SRPInterface ->ScriptCall(inst,NULL,"getstr","(s)s","3333333"));

	SRPInterface -> Release();
	VSCore_TermSimple(&Context);
	return 0;
}
```

8.8 lua call other raw script functions

8.8.1 call c dll

Lua can call simple dll functions directly. These functions's input parameters and output parameters are integer, boolean, float,or string.

```
Service=libstarcore._InitSimple("test","123",0,0,nil);
SrvGroup = Service._ServiceGroup;

object = Service:_New()
```

```
--create function description
Service:_CreateAtomicFunctionSimple(Service:_ObjectToAtomic(object),"MessageBoxA","(issI)i","",true,true);
--attach dynamic library to object
object:_AttachRawContext("c","user32.dll",false,"")

object:MessageBoxA(0,"123","123",1)

SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Run:

starapp –e call_c.lua

8.8.2 call python

```
Service=libstarcore._InitSimple("test","123",0,0,nil);
SrvGroup = Service._ServiceGroup;

--init python raw interface
SrvGroup:_InitRaw("python",Service);
--load python module
SrvGroup:_LoadRawModule("python","","..\\python.package\\testpy.py",false,nil);
--attach object to global python space
Object = Service:_ImportRawContext("python","",false,nil);
--call python function tt, the return contains two integer, which will be packed into parapkg
ParaPkg = Object:tt("hello ","world");
print("ret from python :  ",ParaPkg:_Get(0),ParaPkg:_Get(1));
--get global int value g1
print("python value g1 :  ",Object.g1);

--call python function yy, the return is dict, which will be mapped to cle object
yy = Object:yy("hello ","world",123);
--call dict __len__ function to get dict length
print("python value dict length :  ",yy:__len__());

--get global class Multiply
Multiply = Service:_ImportRawContext("python","Multiply",true,nil);
multiply = Multiply(33,44);
--call instance method multiply
print("instance multiply = ",multiply:multiply(11,22));

SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Run:

starapp –e call_python.lua

8.8.3 call java

```
Service=libstarcore._InitSimple("test","123",0,0,nil);
SrvGroup = Service._ServiceGroup;

--init java raw interface
SrvGroup:_InitRaw("java",Service);
--load java module
SrvGroup:_LoadRawModule("java","","..\\java.package\\test.jar",false,nil);
--attach object to global java space
TestJava = Service:_ImportRawContext("java","test/TestJava",true,nil);
--get and set static field
TestJava.COUNT = 7766
print("java value COUNT :  ",TestJava.COUNT);
--call static method
print("java getHelloWorld() :  ",TestJava:getHelloWorld());

--create instance of TestJava class
inst = TestJava("cle value",44);

--call normal function setCounts
inst:setCounts(SrvGroup:_NewParaPkg(77,88));
ret = inst:getCounts();
print("ret from java :  ",ret:_Get(0),ret:_Get(1));

SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

8.8.4 call java with callback

```
Service=libstarcore._InitSimple("test","123",0,0,nil);
SrvGroup = Service._ServiceGroup;

--init java raw interface ---*/
SrvGroup:_InitRaw("java",Service);
--load java module ---*/
SrvGroup:_LoadRawModule("java","","..\\java.package\\testcallback.jar",false,nil);
--attach object to testcallback/TestCallBack ---*/
```

```
TestCallBack = Service:_ImportRawContext("java","testcallback/TestCallBack",true,nil);
--create an instance of TestCallBack-----*/
inst = TestCallBack()

--create cle object associated with proxy ---*/

--1 : create cleobject and functions for proxy----*/
cleobject = Service:_New()
function cleobject:postExec()
   print("call back from java");
end

function cleobject:getNum(input)
   print("call back [getNum]from java : ",input:_Get(0),input:_Get(1));
       return input:_Get(0) + input:_Get(1);
end

--create proxy for interface testcallback/ICallBack ---*/
proxy = Service:_NewRawProxy("java",cleobject,nil,"testcallback.ICallBack",0);
--set the proxy to TestCallBack instance ---*/
inst:setCallBack(proxy);
--now proxy can be freed----*/
proxy:_Free();

--call inst function postExec----*/
inst:postExec();
--call inst function getNum----*/
print(inst:getNum(SrvGroup:_NewParaPkg(123,456)));

SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

8.8.5 call java extend class

```
Service=libstarcore._InitSimple("test","123",0,0,nil);
SrvGroup = Service._ServiceGroup;

--init java raw interface ---*/
SrvGroup:_InitRaw("java",Service);
--load java module ---*/
SrvGroup:_LoadRawModule("java","","..\\java.package\\testextend.jar",false,nil);
--attach object to testextend/ExtendClass ---*/
ExtendClass = Service:_ImportRawContext("java","testextend/ExtendClass",true,nil);
--create an instance of ExtendClass-----*/
```

```
inst = ExtendClass()
function inst:_Star_getstr(input)
    print("cle class..... : ",input);
        return self:_SuperStar_getstr(input);
end
--call function getstr---*/
print(inst:getstr("3333333"));

SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

8.8.6 call cs

```
Service=libstarcore._InitSimple("test","123",0,0,nil);
SrvGroup = Service._ServiceGroup;

--init csharp raw interface ---*/
SrvGroup:_InitRaw("csharp",Service);
--load csharp module ---*/
SrvGroup:_LoadRawModule("csharp","testcs","..\\cs.package\\testcs\\bin\\Debug\\testcs.dll",false,nil);
--attach object to testcs.Class1 ---*/
Class1 = Service:_ImportRawContext("csharp","testcs.Class1",true,nil);
--get and set static field---*/
Class1.COUNT = 7766
print("csharp value COUNT :  ",Class1.COUNT);
--call static method---------*/
print("csharp getHelloWorld() :  ",Class1:getHelloWorld());

--create instance of Class1 class----*/
inst = Class1("cle value",44);

--call normal function setCounts ---*/
inst:setCounts(SrvGroup:_NewParaPkg(77,88));
ParaPkg = inst:getCounts();
print("ret from csharp :  ",ParaPkg:_Get(0),ParaPkg:_Get(1));

SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

8.8.7 call cs with callback

```
Service=libstarcore._InitSimple("test","123",0,0,nil);
SrvGroup = Service._ServiceGroup;
```

```
--init cs raw interface ---*/
SrvGroup:_InitRaw("csharp",Service);
--load cs module ---*/
SrvGroup:_LoadRawModule("csharp","testcallback","..\\cs.package\\testcallback\\bin\\Debug\\testcallback.dll",false,nil);
--attach object to testcallback.Class1 ---*/
TestCallBack = Service:_ImportRawContext("csharp","testcallback.Class1",true,nil);
--create an instance of TestCallBack-----*/
inst = TestCallBack();

--create cle object associated with proxy ---*/

--1 : create cleobject and functions for proxy----*/
cleobject = Service:_New()
function cleobject:postExec()
   print("call back from cs");
end
function cleobject:getNum(input)
   print("call back [getNum]from cs : ",input:_Get(0),input:_Get(1));
       return input:_Get(0) + input:_Get(1);
end

--create proxy for interface testcallback/ICallBack ---*/
proxy1 = Service:_NewRawProxy("csharp",cleobject,"postExec","testcallback.postExec",0);
proxy2 = Service:_NewRawProxy("csharp",cleobject,"getNum","testcallback.getNum",0);
--set the proxy to TestCallBack instance ---*/
inst:setCallBack(proxy1,proxy2);
--now proxy can be freed----*/
proxy1:_Free();
proxy2:_Free();

--call inst function postExec----*/
inst:postExec();
--call inst function getNum----*/
print(inst:getNum(SrvGroup:_NewParaPkg(123.0,456.0)));

SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

8.8.8 call cs extend class

```
Service=libstarcore._InitSimple("test","123",0,0,nil);
SrvGroup = Service._ServiceGroup;
```

```
--init cs raw interface ---*/
SrvGroup:_InitRaw("csharp",Service);
--load csharp module ---*/
SrvGroup:_LoadRawModule("csharp","testextend","..\\cs.package\\testextend\\bin\\Debug\\testextend.dll",false,nil);
--attach object to testextend.ExtendClass ---*/
ExtendClass = Service:_ImportRawContext("csharp","testextend.ExtendClass",true,nil);
--create an instance of ExtendClass-----*/
inst = ExtendClass();
function inst:_Star_getstr(input)
    print("cle class..... : ",input);
        return self:_SuperStar_getstr(input);
end
--call function getstr---*/
print(inst:getstr("3333333"));

SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

8.9 python call other raw script functions

8.9.1 call c dll

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0,None);
SrvGroup = Service._ServiceGroup;

object = Service._New()
#--create function description
Service._CreateAtomicFunctionSimple(Service._ObjectToAtomic(object),"MessageBoxA","(issI)i","",True,True);
#--attach dynamic library to object
object._AttachRawContext("c","user32.dll",False,"")

object.MessageBoxA(0,"123","123",1)

SrvGroup._ClearService()
libstarpy._ModuleExit()
```

run:

python call_c.py or

starapp –e call_py.py?script=python

8.9.2 call lua

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0);
SrvGroup = Service._ServiceGroup;

#--init lua raw interface ---*/
SrvGroup._InitRaw("lua",Service);
#--load lua module ---*/
SrvGroup._LoadRawModule("lua","","..\\lua.package\\testlua.lua",False);
#--attach object to global lua space ---*/
Object = Service._ImportRawContext("lua","",False,"");
#--call lua function tt, the return contains two integer, which will be packed into parapkg ---*/
ParaPkg = Object.tt("hello ","world");
print("ret from lua : ",ParaPkg._Get(0),ParaPkg._Get(1));
#--get global int value g1--------*/
print("lua value g1 :  ",Object.g1);
#--get global table value c, which is a table with function, it will be mapped to cle object --------*/
c = Object.c;
#--get int value x from c--------*/
print("c value x :  ",c.x);
#--call c function yy, the return is a table, which will be mapped to cle object ---*/
yy = c.yy("hello ","world","!");
print("yy value [1] :  ",yy._Get("1"));
print("yy value [Type] :  ",yy._Get("Type"));

SrvGroup._ClearService()
libstarpy._ModuleExit()
```

8.9.3 call java

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0);
SrvGroup = Service._ServiceGroup;

#--init java raw interface
SrvGroup._InitRaw("java",Service);
#--load java module
SrvGroup._LoadRawModule("java","","..\\java.package\\test.jar",False);
#--attach object to global java space
TestJava = Service._ImportRawContext("java","test/TestJava",False,"");
#--get and set static field
TestJava.COUNT = 7766
```

```
print("java value COUNT :  ",TestJava.COUNT);
#--call static method
print("java getHelloWorld() :  ",TestJava.getHelloWorld());

#--create instance of TestJava class
inst = TestJava("cle value",44);

#--call normal function setCounts
inst.setCounts(SrvGroup._NewParaPkg(77,88));
ret = inst.getCounts();
print("ret from java :  ",ret._Get(0),ret._Get(1));

SrvGroup._ClearService()
libstarpy._ModuleExit()
```

8.9.4 call java with callback

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0);
SrvGroup = Service._ServiceGroup;

#--init java raw interface ---*/
SrvGroup._InitRaw("java",Service);
#--load java module ---*/
SrvGroup._LoadRawModule("java","","..\\java.package\\testcallback.jar",False);
#--attach object to testcallback/TestCallBack ---*/
TestCallBack = Service._ImportRawContext("java","testcallback/TestCallBack",True,"");
#--create an instance of TestCallBack-----*/
inst = TestCallBack()

#--create cle object associated with proxy ---*/

#--1 : create cleobject and functions for proxy----*/
cleobject = Service._New()
def cleobject_postExec(self) :
    print("call back from java");
cleobject.postExec = cleobject_postExec

def cleobject_getNum(self, input) :
    print("call back [getNum]from java : ",input._Get(0),input._Get(1));
    return input._Get(0) + input._Get(1);
cleobject.getNum = cleobject_getNum

#--create proxy for interface testcallback/ICallBack ---*/
```

```
proxy = Service._NewRawProxy("java",cleobject,"","testcallback.ICallBack",0);
#--set the proxy to TestCallBack instance ---*/
inst.setCallBack(proxy);
#--now proxy can be freed----*/
proxy._Free();

#--call inst function postExec----*/
inst.postExec();
#--call inst function getNum----*/
print(inst.getNum(SrvGroup._NewParaPkg(123,456)));

SrvGroup._ClearService()
libstarpy._ModuleExit()
```

8.9.5 call java extend class

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0);
SrvGroup = Service._ServiceGroup;

#--init java raw interface ---*/
SrvGroup._InitRaw("java",Service);
#--load java module ---*/
SrvGroup._LoadRawModule("java","","..\\java.package\\testextend.jar",False);
#--attach object to testextend/ExtendClass ---*/
ExtendClass = Service._ImportRawContext("java","testextend/ExtendClass",True,"");
#--create an instance of ExtendClass-----*/
inst = ExtendClass()
def inst_Star_getstr(self,input) :
    print("cle class..... : ",input);
    return self._SuperStar_getstr(input);
inst._Star_getstr = inst_Star_getstr
#--call function getstr---*/
print(inst.getstr("3333333"));

SrvGroup._ClearService()
libstarpy._ModuleExit()
```

8.9.6 call cs

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0);
SrvGroup = Service._ServiceGroup;
```

```
#--init csharp raw interface ---*/
SrvGroup._InitRaw("csharp",Service);
#--load csharp module ---*/
SrvGroup._LoadRawModule("csharp","testcs","..\\cs.package\\testcs\\bin\\Debug\\testcs.dll",False);
#--attach object to testcs.Class1 ---*/
Class1 = Service._ImportRawContext("csharp","testcs.Class1",True,"");
#--get and set static field---*/
Class1.COUNT = 7766
print("csharp value COUNT :  ",Class1.COUNT);
#--call static method---------*/
print("csharp getHelloWorld() :  ",Class1.getHelloWorld());

#--create instance of Class1 class----*/
inst = Class1("cle value",44);

#--call normal function setCounts ---*/
inst.setCounts(SrvGroup._NewParaPkg(77,88));
ParaPkg = inst.getCounts();
print("ret from csharp :  ",ParaPkg._Get(0),ParaPkg._Get(1));

SrvGroup._ClearService()
libstarpy._ModuleExit()
```

8.9.7 call cs with callback

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0);
SrvGroup = Service._ServiceGroup;

#--init cs raw interface ---*/
SrvGroup._InitRaw("csharp",Service);
#--load cs module ---*/
SrvGroup._LoadRawModule("csharp","testcallback","..\\cs.package\\testcallback\\bin\\Debug\\testcallback.dll",False);
#--attach object to testcallback.Class1 ---*/
TestCallBack = Service._ImportRawContext("csharp","testcallback.Class1",True,"");
#--create an instance of TestCallBack-----*/
inst = TestCallBack();

#--create cle object associated with proxy ---*/

#--1 : create cleobject and functions for proxy----*/
cleobject = Service._New()
class cleobjectclass :
```

```
    def postExec(self) :
        print("call back from cs");
    def getNum(self,input) :
        print("call back [getNum]from cs : ",input._Get(0),input._Get(1));
        return input._Get(0) + input._Get(1);
cleobject._Assign(cleobjectclass())

#--create proxy for interface testcallback/ICallBack ---*/
proxy1 = Service._NewRawProxy("csharp",cleobject,"postExec","testcallback.postExec",0);
proxy2 = Service._NewRawProxy("csharp",cleobject,"getNum","testcallback.getNum",0);
#--set the proxy to TestCallBack instance ---*/
inst.setCallBack(proxy1,proxy2);
#--now proxy can be freed----*/
proxy1._Free();
proxy2._Free();

#--call inst function postExec----*/
inst.postExec();
#--call inst function getNum----*/
print(inst.getNum(SrvGroup._NewParaPkg(123.0,456.0)));

SrvGroup._ClearService()
libstarpy._ModuleExit()
```

8.9.8 call cs extend class

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0);
SrvGroup = Service._ServiceGroup;

#--init cs raw interface ---*/
SrvGroup._InitRaw("csharp",Service);
#--load csharp module ---*/
SrvGroup._LoadRawModule("csharp","testextend","..\\cs.package\\testextend\\bin\\Debug\\testextend.dll",False);
#--attach object to testextend.ExtendClass ---*/
ExtendClass = Service._ImportRawContext("csharp","testextend.ExtendClass",True,"");
#--create an instance of ExtendClass-----*/
inst = ExtendClass();
def inst_Star_getstr(self,input) :
    print("cle class..... : ",input);
    return self._SuperStar_getstr(input);
inst._Star_getstr = inst_Star_getstr
#--call function getstr---*/
print(inst.getstr("3333333"));
```

```
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

8.10 java call other raw script functions

8.10.1 call c dll

```
import com.srplab.www.starcore.*;

public class call_c{
        public static void main(String[] args){
                StarCoreFactory starcore= StarCoreFactory.GetFactory();
                StarServiceClass Service=starcore._InitSimple("test","123",0,0);
                StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

        StarObjectClass object = Service._New();
        //--create function description
        Service._CreateAtomicFunctionSimple(Service._ObjectToAtomic(object),"MessageBoxA","(issI)i","",true,true);
        //--attach dynamic library to object
        object._AttachRawContext("c","user32.dll",false,"");

        object._Call("MessageBoxA",0,"123","123",1);

        SrvGroup._ClearService();
        starcore._ModuleExit();
    }
}
```

8.10.2 call lua

```
import com.srplab.www.starcore.*;

public class call_lua{
        public static void main(String[] args){
                StarCoreFactory starcore= StarCoreFactory.GetFactory();
                StarServiceClass Service=starcore._InitSimple("test","123",0,0);
                StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

                //--init lua raw interface ---*/
                SrvGroup._InitRaw("lua",Service);
```

```
            //--load lua module ---*/
            SrvGroup._LoadRawModule("lua","","..\\lua.package\\testlua.lua",false);
            //--attach object to global lua space ---*/
            StarObjectClass object = Service._ImportRawContext("lua","",false,"");
            //--call lua function tt, the return contains two integer, which will be packed into parapkg ---*/
            StarParaPkgClass ParaPkg = (StarParaPkgClass)object._Call("tt","hello ","world");
            System.out.println("ret from lua : "+ParaPkg._Get(0)+"  "+ParaPkg._Get(1));
            //--get global int value g1--------*/
            System.out.println("lua value g1 :  "+object._Get("g1"));
            //--get global table value c, which is a table with function, it will be mapped to cle object --------*/
            StarObjectClass c = object._GetObject("c");
            //--get int value x from c--------*/
            System.out.println("c value x :  "+c._Get("x"));
            //--call c function yy, the return is a table, which will be mapped to cle object ---*/
            StarObjectClass yy = (StarObjectClass)c._Call("yy","hello ","world","!");
            System.out.println("yy value [1] :  "+yy._Get("1"));
            System.out.println("yy value [Type] :  "+yy._Get("Type"));

            SrvGroup._ClearService();
            starcore._ModuleExit();
    }
}
```

8.10.3 call python

```
import com.srplab.www.starcore.*;

public class call_python{
        public static void main(String[] args){
            StarCoreFactory starcore= StarCoreFactory.GetFactory();
            StarServiceClass Service=starcore._InitSimple("test","123",0,0);
            StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

            //--init python raw interface
            SrvGroup._InitRaw("python",Service);
            //--load python module
            SrvGroup._LoadRawModule("python","","..\\python.package\\testpy.py",false);
            //--attach object to global python space
            StarObjectClass object = Service._ImportRawContext("python","",false,"");
            //--call python function tt, the return contains two integer, which will be packed into parapkg
            StarParaPkgClass ParaPkg = (StarParaPkgClass)object._Call("tt","hello ","world");
            System.out.println("ret from python :  "+ParaPkg._Get(0)+"   "+ParaPkg._Get(1));
            //--get global int value g1
            System.out.println("python value g1 :  "+object._Get("g1"));
```

```
            //--call python function yy, the return is dict, which will be mapped to cle object
            StarObjectClass yy = (StarObjectClass)object._Call("yy","hello ","world",123);
            //--call dict __len__ function to get dict length
            System.out.println("python value dict length :  "+yy._Call("__len__"));

            //--get global class Multiply
            StarObjectClass Multiply = Service._ImportRawContext("python","Multiply",true,null);
            StarObjectClass multiply = Multiply._Callobject("_StarCall",33,44);
            //--call instance method multiply
            System.out.println("instance multiply = "+multiply._Call("multiply",11,22));

            SrvGroup._ClearService();
            starcore._ModuleExit();
    }
}
```

8.10.4 call cs

```
import com.srplab.www.starcore.*;

public class call_cs{
    public static void main(String[] args){
            StarCoreFactory starcore= StarCoreFactory.GetFactory();
            StarServiceClass Service=starcore._InitSimple("test","123",0,0);
            StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

            //--init csharp raw interface ---*/
            SrvGroup._InitRaw("csharp",Service);
            //--load csharp module ---*/
            SrvGroup._LoadRawModule("csharp","testcs","..\\cs.package\\testcs\\bin\\Debug\\testcs.dll",false);
            //--attach object to testcs.Class1 ---*/
            StarObjectClass Class1 = Service._ImportRawContext("csharp","testcs.Class1",true,"");
            //--get and set static field---*/
            Class1._Set("COUNT",7766);
            System.out.println("csharp value COUNT :  "+Class1._Get("COUNT"));
            //--call static method---------*/
            System.out.println("csharp getHelloWorld() :  "+Class1._Call("getHelloWorld"));

            //--create instance of Class1 class----*/
            StarObjectClass inst = Class1._Callobject("_StarCall","cle value",44);

            //--call normal function setCounts ---*/
            inst._Call("setCounts",SrvGroup._NewParaPkg(77,88));
```

```
            StarParaPkgClass ParaPkg = (StarParaPkgClass)inst._Call("getCounts");
            System.out.println("ret from csharp :  "+ParaPkg._Get(0)+"   "+ParaPkg._Get(1));

            SrvGroup._ClearService();
            starcore._ModuleExit();
    }
}
```

8.10.5 call cs with callback

```
import com.srplab.www.starcore.*;

public class call_cs_callback{
        public static void main(String[] args){
            StarCoreFactory starcore= StarCoreFactory.GetFactory();
            StarServiceClass Service=starcore._InitSimple("test","123",0,0);
            StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

            //--init cs raw interface ---*/
            SrvGroup._InitRaw("csharp",Service);
            //--load cs module ---*/
            SrvGroup._LoadRawModule("csharp","testcallback","..\\cs.package\\testcallback\\bin\\Debug\\testcallback.dll",false);
            //--attach object to testcallback.Class1 ---*/
            StarObjectClass TestCallBack = Service._ImportRawContext("csharp","testcallback.Class1",true,"");
            //--create an instance of TestCallBack-----*/
            StarObjectClass inst = TestCallBack._Callobject("_StarCall");

            //--create cle object associated with proxy ---*/

            //--1 : create cleobject and functions for proxy----*/
            StarObjectClass cleobject = Service._New()._Assign(new StarObjectClass(){
               public void postExec(StarObjectClass self){
                      System.out.println("call back from cs");
               };
               public float getNum(StarObjectClass self,StarParaPkgClass input){
                  System.out.println("call back [getNum]from cs : "+input._Get(0)+"    "+input._Get(1));
                  return (float)(input._Getdouble(0) + input._Getdouble(1));
               };
            });

            //--create proxy for interface testcallback/ICallBack ---*/
            StarObjectClass proxy1 = Service._NewRawProxy("csharp",cleobject,"postExec","testcallback.postExec",0);
            StarObjectClass proxy2 = Service._NewRawProxy("csharp",cleobject,"getNum","testcallback.getNum",0);
            //--set the proxy to TestCallBack instance ---*/
```

```
            inst._Call("setCallBack",proxy1,proxy2);
            //--now proxy can be freed----*/
            proxy1._Free();
            proxy2._Free();

            //--call inst function postExec----*/
            inst._Call("postExec");
            //--call inst function getNum----*/
            System.out.println(inst._Call("getNum",SrvGroup._NewParaPkg(123.0,456.0)));

            SrvGroup._ClearService();
            starcore._ModuleExit();
    }
}
```

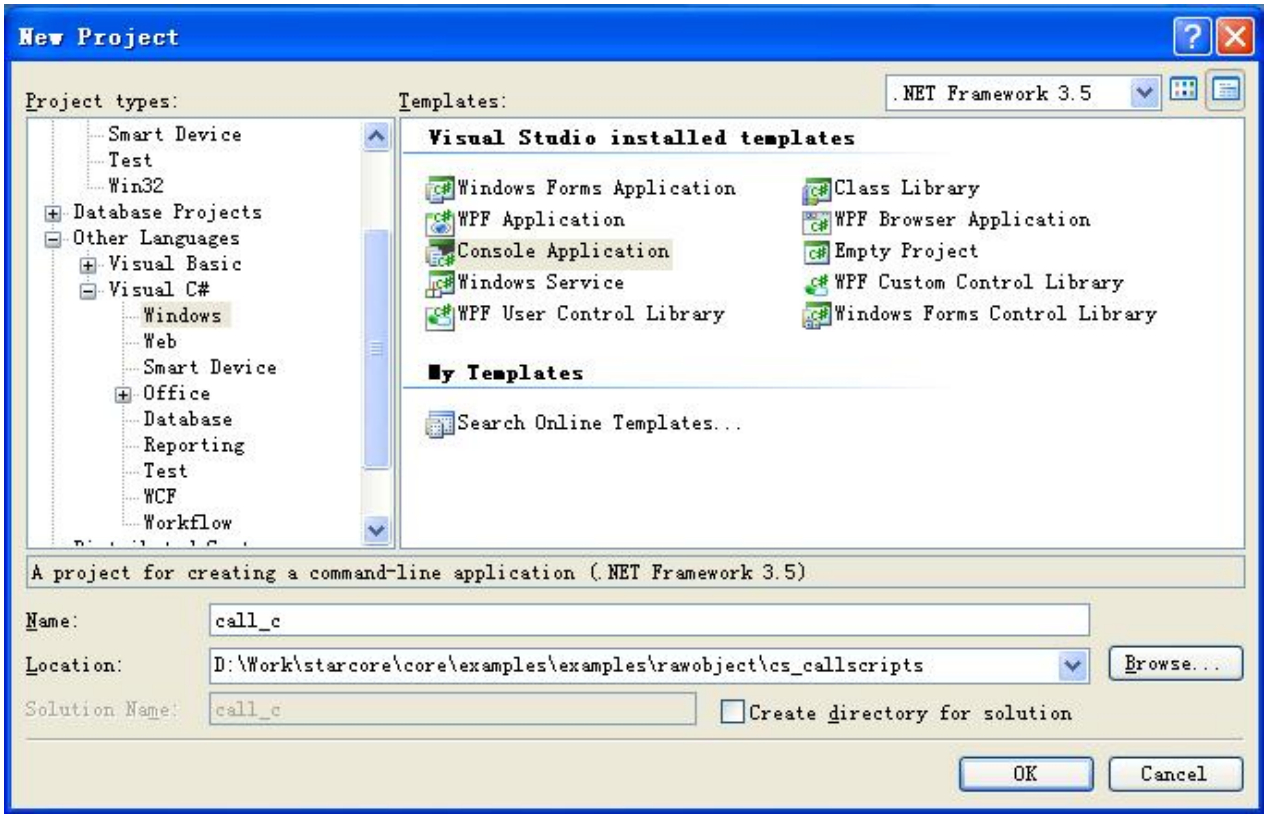
8.10.6 call cs extend class

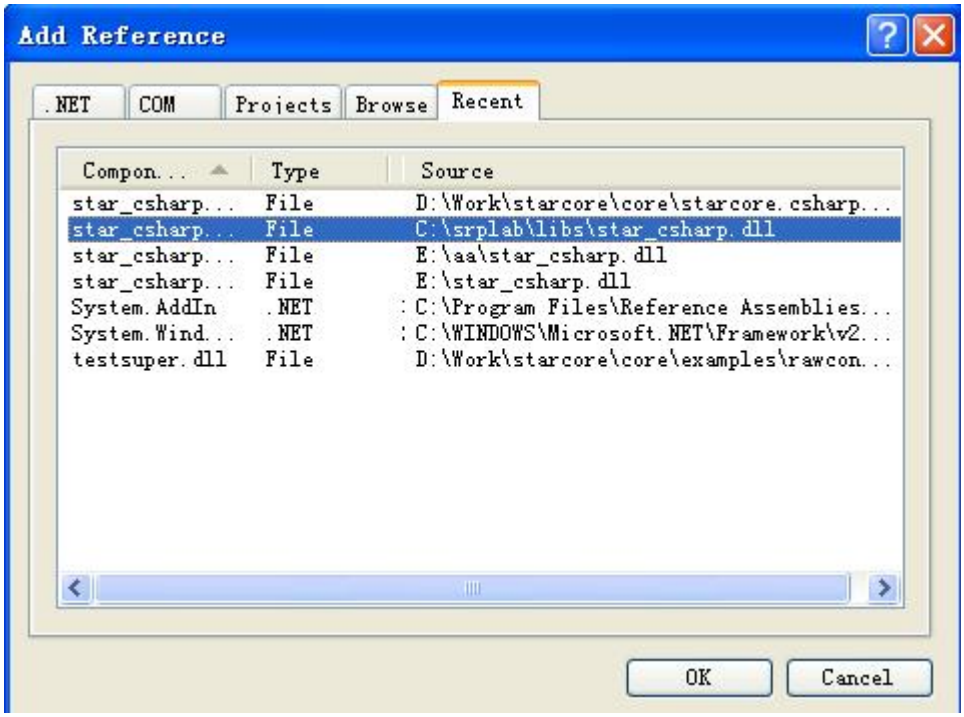
```
import com.srplab.www.starcore.*;

public class call_cs_extend{
    public static void main(String[] args){
            StarCoreFactory starcore= StarCoreFactory.GetFactory();
            StarServiceClass Service=starcore._InitSimple("test","123",0,0);
            StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");
            //--init cs raw interface ---*/
            SrvGroup._InitRaw("csharp",Service);
            //--load csharp module ---*/
            SrvGroup._LoadRawModule("csharp","testextend","..\\cs.package\\testextend\\bin\\Debug\\testextend.dll",false);
            //--attach object to testextend.ExtendClass ---*/
            StarObjectClass ExtendClass = Service._ImportRawContext("csharp","testextend.ExtendClass",true,"");
            //--create an instance of ExtendClass-----*/
            StarObjectClass inst = ExtendClass._Callobject("_StarCall")._Assign(new StarObjectClass(){
            public String _Star_getstr(StarObjectClass self,String input){
            System.out.println("call back [getstr]from cs : "+input);
            return (String)self._Call("_SuperStar_getstr",input);
        };
       });
            //--call function getstr---*/
            System.out.println(inst._Call("getstr","3333333"));

            SrvGroup._ClearService();
            starcore._ModuleExit();
    }
}
```

8.11 cs call other raw script functions

8.11.1 call c dll

8.11.1.1create project





8.11.1.2source file

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace call_c
{
   class Program
   {
      static void Main(string[] args)
      {
            StarCoreFactory starcore= StarCoreFactory.GetFactory();
            StarServiceClass Service=starcore._InitSimple("test","123",0,0);
               StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

         StarObjectClass obj = Service._New();
         //--create function description
         Service._CreateAtomicFunctionSimple(Service._ObjectToAtomic(obj), "MessageBoxA", "(issI)i", "", true, true);
         //--attach dynamic library to object
         obj._AttachRawContext("c", "user32.dll", false, "");

         obj._Call("MessageBoxA", 0, "123", "123", 1);
```

```
        SrvGroup._ClearService();
        starcore._ModuleExit();
      }
    }
}
```

8.11.2 call lua

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace call_lua
{
    class Program
    {
        static void Main(string[] args)
        {
                StarCoreFactory starcore= StarCoreFactory.GetFactory();
                StarServiceClass Service=starcore._InitSimple("test","123",0,0);
                StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

                //--init lua raw interface ---*/
                SrvGroup._InitRaw("lua",Service);
                //--load lua module ---*/
                SrvGroup._LoadRawModule("lua","","..\\..\\..\\..\\lua.package\\testlua.lua",false);
                //--attach object to global lua space ---*/
                StarObjectClass obj = Service._ImportRawContext("lua","",false,"");
                //--call lua function tt, the return contains two integer, which will be packed into parapkg ---*/
        StarParaPkgClass ParaPkg = (StarParaPkgClass)obj._Call("tt", "hello ", "world");
                Console.WriteLine("ret from lua : "+ParaPkg._Get(0)+"  "+ParaPkg._Get(1));
                //--get global int value g1--------*/
        Console.WriteLine("lua value g1 :  " + obj._Get("g1"));
                //--get global table value c, which is a table with function, it will be mapped to cle object --------*/
        StarObjectClass c = obj._GetObject("c");
        //--get int value x obj c--------*/
                Console.WriteLine("c value x :  "+c._Get("x"));
                //--call c function yy, the return is a table, which will be mapped to cle object ---*/
                StarObjectClass yy = (StarObjectClass)c._Call("yy","hello ","world","!");
                Console.WriteLine("yy value [1] :  "+yy._Get("1"));
                Console.WriteLine("yy value [Type] :  "+yy._Get("Type"));
```

```
            SrvGroup._ClearService();
            starcore._ModuleExit();
        }
    }
}
```

8.11.3 call python

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace call_python
{
    class Program
    {
        static void Main(string[] args)
        {
            StarCoreFactory starcore= StarCoreFactory.GetFactory();
            StarServiceClass Service=starcore._InitSimple("test","123",0,0);
            StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

            //--init python raw interface
            SrvGroup._InitRaw("python",Service);
            //--load python module
            SrvGroup._LoadRawModule("python","","..\\..\\..\\..\\python.package\\testpy.py",false);
            //--attach object to global python space
            StarObjectClass obj = Service._ImportRawContext("python","",false,"");
            //--call python function tt, the return contains two integer, which will be packed into parapkg
        StarParaPkgClass ParaPkg = (StarParaPkgClass)obj._Call("tt", "hello ", "world");
            Console.WriteLine("ret from python :  "+ParaPkg._Get(0)+"   "+ParaPkg._Get(1));
            //--get global int value g1
        Console.WriteLine("python value g1 :  " + obj._Get("g1"));

            //--call python function yy, the return is dict, which will be mapped to cle object
        StarObjectClass yy = (StarObjectClass)obj._Call("yy", "hello ", "world", 123);
            //--call dict __len__ function to get dict length
            Console.WriteLine("python value dict length :  "+yy._Call("__len__"));

            //--get global class Multiply
            StarObjectClass Multiply = Service._ImportRawContext("python","Multiply",true,null);
```

```
            StarObjectClass multiply = Multiply._Callobject("_StarCall",33,44);
            //--call instance method multiply
            Console.WriteLine("instance multiply = "+multiply._Call("multiply",11,22));

            SrvGroup._ClearService();
            starcore._ModuleExit();
        }
    }
}
```

8.11.4 call java

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace call_java
{
    class Program
    {
        static void Main(string[] args)
        {
            StarCoreFactory starcore = StarCoreFactory.GetFactory();
            StarServiceClass Service = starcore._InitSimple("test", "123", 0, 0);
            StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

            //--init java raw interface
            SrvGroup._InitRaw("java",Service);
            //--load java module
            SrvGroup._LoadRawModule("java","",".\\..\\..\\..\\java.package\\test.jar",false);
            //--attach object to global java space
            StarObjectClass TestJava = Service._ImportRawContext("java","test/TestJava",false,"");
            //--get and set static field
            TestJava._Set("COUNT", 7766);
            Console.WriteLine("java value COUNT :  "+TestJava._Get("COUNT"));
            //--call static method
            Console.WriteLine("java getHelloWorld() :  "+TestJava._Call("getHelloWorld"));

            //--create instance of TestJava class
            StarObjectClass inst = TestJava._New("","","cle value", 44);

            //--call normal function setCounts
```

```
        inst._Call("setCounts",SrvGroup._NewParaPkg(77,88));
        StarParaPkgClass ret = (StarParaPkgClass)inst._Call("getCounts");
        Console.WriteLine("ret from java :  "+ret._Get(0)+"   "+ret._Get(1));

        SrvGroup._ClearService();
        starcore._ModuleExit();
    }
  }
}
```

8.11.5 call java with callback

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace call_java_callback
{
  class MyStarObjectClass : StarObjectClass{
    public void postExec(StarObjectClass self){
     Console.WriteLine("call back from cs");
    }
    public float getNum(StarObjectClass self,StarParaPkgClass input){
     Console.WriteLine("call back [getNum]from cs : "+input._Get(0)+"    "+input._Get(1));
          return (float)(input._Getdouble(0) + input._Getdouble(1));
    }
  };

  class Program
  {
    static void Main(string[] args)
    {
      StarCoreFactory starcore = StarCoreFactory.GetFactory();
      StarServiceClass Service = starcore._InitSimple("test", "123", 0, 0);
      StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

      //--init java raw interface ---*/
      SrvGroup._InitRaw("java",Service);
      //--load java module ---*/
      SrvGroup._LoadRawModule("java","","..\\..\\..\\..\\java.package\\testcallback.jar",false);
      //--attach object to testcallback/TestCallBack ---*/
      StarObjectClass TestCallBack = Service._ImportRawContext("java","testcallback/TestCallBack",true,"");
```

```
        //--create an instance of TestCallBack-----*/
        StarObjectClass inst = TestCallBack._New();

        //--create cle object associated with proxy ---*/

        //--1 : create cleobject and functions for proxy----*/
            StarObjectClass cleobject = Service._New()._Assign(new MyStarObjectClass());

        //--create proxy for interface testcallback/ICallBack ---*/
        StarObjectClass proxy = Service._NewRawProxy("java",cleobject,"","testcallback.ICallBack",0);
        //--set the proxy to TestCallBack instance ---*/
        inst._Call("setCallBack",proxy);
        //--now proxy can be freed----*/
        proxy._Free();

        //--call inst function postExec----*/
        inst._Call("postExec");
        //--call inst function getNum----*/
        Console.WriteLine(inst._Call("getNum",SrvGroup._NewParaPkg(123,456)));

        SrvGroup._ClearService();
        starcore._ModuleExit();
    }
  }
}
```

8.11.6 call java extend class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace call_java_extend
{
  class MyStarObjectClass : StarObjectClass{
            public String _Star_getstr(StarObjectClass self,String input){
          Console.WriteLine("call back [getstr]from cs : " + input);
          return (String)self._Call("_SuperStar_getstr", input);
      }
  }

  class Program
```

```
  {
    static void Main(string[] args)
    {
      StarCoreFactory starcore = StarCoreFactory.GetFactory();
      StarServiceClass Service = starcore._InitSimple("test", "123", 0, 0);
      StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

      //--init java raw interface ---*/
      SrvGroup._InitRaw("java",Service);
      //--load java module ---*/
      SrvGroup._LoadRawModule("java","","..\\..\\..\\..\\java.package\\testextend.jar",false);
      //--attach object to testextend/ExtendClass ---*/
      StarObjectClass ExtendClass = Service._ImportRawContext("java","testextend/ExtendClass",true,"");
      //--create an instance of ExtendClass-----*/
      StarObjectClass inst = ExtendClass._New()._Assign(new MyStarObjectClass());
      //--call function getstr---*/
      Console.WriteLine(inst._Call("getstr","3333333"));

      SrvGroup._ClearService();
      starcore._ModuleExit();
    }
  }
}
```

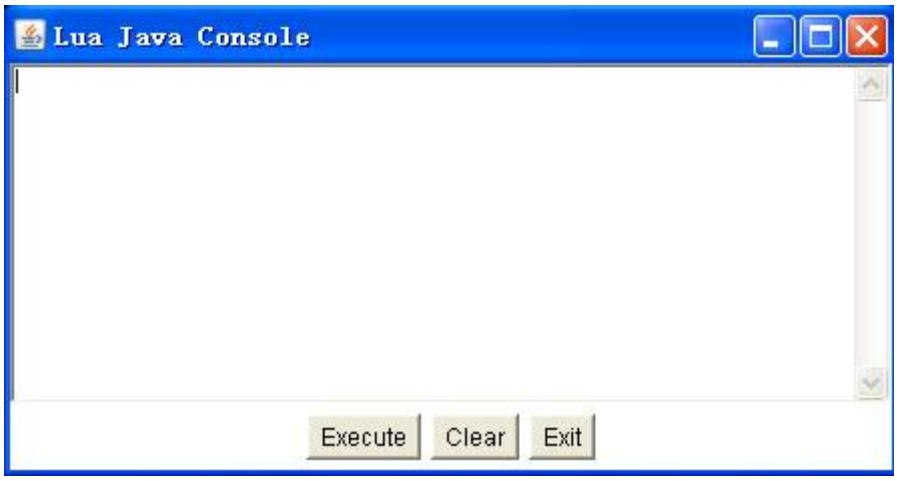
8.12 some examples

8.12.1 lua call java awt

```
SrvGroup=_GetSrvGroup()
SrvGroup:_CreateService("","test","123",0,0,0,0,0)
Service=SrvGroup:_GetService("root","123")

SrvGroup:_InitRaw("java",Service)
--import class
Frame = Service:_ImportRawContext("java","java/awt/Frame",true,"")
Console = Service:_ImportRawContext("java","java/awt/TextArea",true,"")
Panel = Service:_ImportRawContext("java","java/awt/Panel",true,"")
Button = Service:_ImportRawContext("java","java/awt/Button",true,"")
BorderLayout = Service:_ImportRawContext("java","java/awt/BorderLayout",true,"")

print(Frame,Console,Panel,Button,BorderLayout)

--create instance
frame = Frame("Lua Java Console");
```

```
console = Console();
buttons_pn = Panel();
execute_bt = Button("Execute");
clear_bt = Button("Clear");
exit_bt = Button("Exit");

print(frame,console,buttons_pn,execute_bt,clear_bt,exit_bt)

frame:setSize(600,300);
buttons_pn:add(execute_bt);
buttons_pn:add(clear_bt);
buttons_pn:add(exit_bt);

frame:add(BorderLayout.NORTH,console)
frame:add(BorderLayout.SOUTH,buttons_pn)

frame:pack()
frame:show()

---create event
luaobj = Service:_New()
function luaobj:actionPerformed(ev)
   print("execute");
   SrvGroup:_RunScript("lua",console:getText(),"");
end

jproxy = Service:_NewRawProxy("java",luaobj,"","java.awt.event.ActionListener",0);
execute_bt:addActionListener(jproxy);

luaobj1 = Service:_New()
function luaobj1:actionPerformed(ev)
   print("clear");
   console:setText("");
end
jproxy = Service:_NewRawProxy("java",luaobj1,"","java.awt.event.ActionListener",0);
clear_bt:addActionListener(jproxy);

luaobj2 = Service:_New()
function luaobj2:actionPerformed(ev)
   print("exit");
   frame:setVisible(false);
   frame:dispose();
   SrvGroup:_ClearService();
end
jproxy = Service:_NewRawProxy("java",luaobj2,"","java.awt.event.ActionListener",0);
```

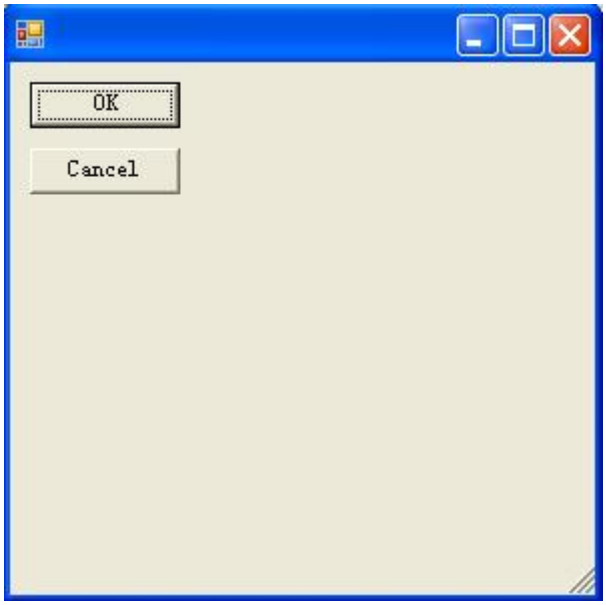
```
exit_bt:addActionListener(jproxy);

--winevent
luaobj3 = Service:_New()
function luaobj3:windowClosing(ev)
   print("close");
   frame:setVisible(false);
   frame:dispose();
   SrvGroup:_ClearService();
end
function luaobj3:windowActivated(ev)
   print("act");
end

jproxy = Service:_NewRawProxy("java",luaobj3,"","java.awt.event.WindowListener",0);
frame:addWindowListener(jproxy);
```



8.12.2 lua call cs form

```
SrvGroup=_GetSrvGroup()
SrvGroup:_CreateService("","test","123",0,0,0,0,0)
Service=SrvGroup:_GetService("root","123")

SrvGroup:_InitRaw("csharp",Service)
Result = SrvGroup:_LoadRawModule("csharp", "System", "", true);
Result = SrvGroup:_LoadRawModule("csharp", "System.Drawing", "", true);
Result = SrvGroup:_LoadRawModule("csharp", "System.Windows.Forms", "", true);

FormClass = Service:_New()
FormClass:_AttachRawContext("csharp", "System.Windows.Forms.Form", true, "");
ButtonClass = Service:_New()
```

```
ButtonClass:_AttachRawContext("csharp", "System.Windows.Forms.Button", true, "");
PointClass = Service:_New()
PointClass:_AttachRawContext("csharp", "System.Drawing.Point", true, "");

form1 = FormClass();
button1 = ButtonClass();
button2 = ButtonClass();
position = PointClass(10, 10);

button1.Text = "OK";
button2.Text = "Cancel";
button1.Location = position
button2.Location = PointClass:_New("", "", button1.Left, button1.Height + button1.Top + 10);

controls = form1.Controls;
controls:Add(button1);
controls:Add(button2);

function button1:onClick(sender,e)
   print("Is Trigger");
   print(e.X);
   print(e.Y);
end
button1.Click:Add(button1:_NewRawProxyEx("","onClick","System.EventHandler"))

button1:_DetachRawContext(true);
button2:_DetachRawContext(true);

form1:ShowDialog()

SrvGroup:_ClearService();
```



8.13 Errors and Exceptions.

For compatiability, CLE does not raise any exception. Applications can uses SrvGroup._GetLastError / Service._GetLastError() / Object._GetLastError() to retrieve the recent error code. And uses _GetLastErrorInfo to get detailed error information.

8.14 Directly assign c/c++, c#, java and object-c object to lua,python and ruby

8.14.1 Assign c/c++ object to scripts

1. To Python

C code

```
{
   BasicSRPInterface ->InitRaw((VS_CHAR*)"python35",SRPInterface);
   void *python = SRPInterface ->ImportRawContext((VS_CHAR*)"python35",(VS_CHAR*)"",false,NULL);
…
   void *CClass = SRPInterface -> MallocObjectL(NULL,0,NULL);
   SRPInterface -> SetName( CClass, "CClass");
   SRPInterface -> RegLuaFunc( CClass, NULL, (void*)CClass_Obj_ScriptCallBack, (VS_UWORD)0 );
   SRPInterface -> RegLuaFuncFilter(CClass,CClass_Obj_LuaFuncFilter,(VS_UWORD)0);
   SRPInterface -> ScriptSetObject(python,"CClass",VSTYPE_OBJPTR,(VS_UWORD)CClass);
}
```

```
static VS_BOOL SRPAPI CClass_Obj_LuaFuncFilter(void *Object,void *ForWhichObject,VS_CHAR
*FuncName,VS_UWORD Para)
{
   if( strcmp(FuncName,"getinfo") == 0 )
      return VS_TRUE;
   if( strcmp(FuncName,"_StarCall") == 0 )
      return VS_TRUE;
   if( strcmp(FuncName,"callback") == 0 )
      return VS_TRUE;
   if( strcmp(FuncName,"SetPythonObject") == 0 )
      return VS_TRUE;
   return VS_FALSE;
}
static VS_INT32 CClass_Obj_ScriptCallBack( void *L )
{
   struct StructOfCClassLocalBuf  *CClassLocalBuf;
   void *Object;
   VS_CHAR *ScriptName;

   ScriptName = SRPInterface -> LuaToString( SRPInterface -> LuaUpValueIndex(3) );
   Object = SRPInterface -> LuaToObject(1);
   /*first input parameter is started at index 2 */
   CClassLocalBuf = (struct StructOfCClassLocalBuf *)SRPInterface -> GetPrivateBuf( Object, SRPInterface ->
GetLayer(Object),0, NULL );
   if( strcmp(ScriptName,"getinfo") == 0 ){
      SRPInterface ->LuaPushString("this module is create by star_module");
      return 1;
   }else if( strcmp(ScriptName,"_StarCall") == 0 ){
      VS_CHAR *Info = SRPInterface ->LuaToString(2);
      printf("%s\n",Info);
      void *Inst = SRPInterface ->IMallocObjectL(SRPInterface->GetIDEx(Object),NULL);
      SRPInterface ->LuaPushObject(Inst);
      return 1;
   }else if( strcmp(ScriptName,"callback") == 0 ){
      if( SRPInterface ->LuaType(2) == VSLUATYPE_NUMBER ){
         double d = SRPInterface ->LuaToNumber(2);
         printf("%f\n",d);
      }else{
         printf("%s\n",SRPInterface->LuaToString(2));
      }
      return 0;
   }else if( strcmp(ScriptName,"SetPythonObject") == 0 ){
      void *raw = SRPInterface->LuaToObject(2);
      printf("%s\n",(char *)SRPInterface ->GetRawContextType(raw,NULL));
   }
```

```
    return 0;
}
```

Python code

```
print(CClass)

val = CClass("from python")
val.callback(1234.4564)
val.callback("sdfsdfsdfsdf")
val.SetPythonObject(json);
print("===========end=========")
```

2. To Ruby

C code

```
{
BasicSRPInterface ->InitRaw((VS_CHAR*)"ruby",SRPInterface);
    void *ruby = SRPInterface ->ImportRawContext((VS_CHAR*)"ruby",(VS_CHAR*)"",false,NULL);

void *CClass = SRPInterface -> MallocObjectL(NULL,0,NULL);
    SRPInterface -> SetName( CClass, "CClass");
    SRPInterface -> RegLuaFunc( CClass, NULL, (void*)CClass_Obj_ScriptCallBack, (VS_UWORD)0 );
    SRPInterface -> RegLuaFuncFilter(CClass,CClass_Obj_LuaFuncFilter,(VS_UWORD)0);
    SRPInterface -> ScriptSetObject(ruby,"CClass",VSTYPE_OBJPTR,(VS_UWORD)CClass);
}

static VS_BOOL SRPAPI CClass_Obj_LuaFuncFilter(void *Object,void *ForWhichObject,VS_CHAR
*FuncName,VS_UWORD Para)
{
    if( strcmp(FuncName,"getinfo") == 0 )
        return VS_TRUE;
    if( strcmp(FuncName,"_StarCall") == 0 )
        return VS_TRUE;
    if( strcmp(FuncName,"callback") == 0 )
        return VS_TRUE;
    if( strcmp(FuncName,"SetPythonObject") == 0 )
        return VS_TRUE;
    return VS_FALSE;
}
static VS_INT32 CClass_Obj_ScriptCallBack( void *L )
{
    struct StructOfCClassLocalBuf  *CClassLocalBuf;
    void *Object;
```

```
    VS_CHAR *ScriptName;

    ScriptName = SRPInterface -> LuaToString( SRPInterface -> LuaUpValueIndex(3) );
    Object = SRPInterface -> LuaToObject(1);
    /*first input parameter is started at index 2 */
    CClassLocalBuf = (struct StructOfCClassLocalBuf *)SRPInterface -> GetPrivateBuf( Object, SRPInterface ->
GetLayer(Object),0, NULL );
    if( strcmp(ScriptName,"getinfo") == 0 ){
        SRPInterface ->LuaPushString("this module is create by star_module");
        return 1;
    }else if( strcmp(ScriptName,"_StarCall") == 0 ){
        VS_CHAR *Info = SRPInterface ->LuaToString(2);
        printf("%s\n",Info);
        void *Inst = SRPInterface ->IMallocObjectL(SRPInterface->GetIDEx(Object),NULL);
        SRPInterface ->LuaPushObject(Inst);
        return 1;
    }else if( strcmp(ScriptName,"callback") == 0 ){
        if( SRPInterface ->LuaType(2) == VSLUATYPE_NUMBER ){
            double d = SRPInterface ->LuaToNumber(2);
            printf("%f\n",d);
        }else{
            printf("%s\n",SRPInterface->LuaToString(2));
        }
        return 0;
    }else if( strcmp(ScriptName,"SetPythonObject") == 0 ){
        void *raw = SRPInterface->LuaToObject(2);
        printf("%s\n",(char *)SRPInterface ->GetRawContextType(raw,NULL));
    }
    return 0;
}
```

Ruby code

```
puts $CClass

val = $CClass.new("from ruby")
puts(val)
val.callback(1234.4564)
val.callback("sdfsdfsdfsdf")
val.SetRubyObject(File);
puts("===========end=========")
```

8.14.2 Assign java object to scripts

1. To Python

Java code

```
{
SrvGroup._InitRaw("python",Service);
StarObjectClass python = Service._ImportRawContext("python","",false,"");

python._Set("JavaClass", CallBackClass.class);
}

public class CallBackClass {
        StarObjectClass PythonClass;
        public CallBackClass(String Info)
    {
                System.out.println(Info);
    }
    public void callback(float val)
    {
        System.out.println("" + val);
    }
    public void callback(String val)
    {
        System.out.println("" + val);
    }
    public void SetPythonObject(Object rb)
    {
       PythonClass = (StarObjectClass)rb; // Ruby File
       String aa = "";
       StarParaPkgClass data1 = MainActivity.Host.SrvGroup._NewParaPkg("b",789,"c",456,"a",123)._AsDict(true);
       Object d1 = PythonClass._Call("dumps", data1, MainActivity.Host.SrvGroup._NewParaPkg("sort_keys",
true)._AsDict(true));
       System.out.println("" + d1);
       Object d2 = PythonClass._Call("dumps", data1,null);
       System.out.println("" + d2);
       Object d3 = PythonClass._Call("dumps", data1, MainActivity.Host.SrvGroup._NewParaPkg("sort_keys", true,
"indent",4)._AsDict(true));
       System.out.println("" + d3);
    }
```

Python code:

```
print(JavaClass)

val = JavaClass("from python")
val.callback(1234.4564)
val.callback("sdfsdfsdfsdf")
val.SetPythonObject(json);
```

```
print("===========end=========")
```

2. To Ruby

Java code

```
{
SrvGroup._InitRaw("ruby",Service);
StarObjectClass ruby = Service._ImportRawContext("ruby","",false,"");

ruby._Set("$JavaClass", CallBackClass.class);
}

public class CallBackClass {
        StarObjectClass RBClass;
        public CallBackClass(String Info)
    {
                System.out.println(Info);
    }
    public void callback(float val)
    {
        System.out.println("" + val);
    }
    public void callback(String val)
    {
        System.out.println("" + val);
    }
    public void SetRubyObject(Object rb)
    {
       RBClass = (StarObjectClass)rb; // Ruby File
       StarObjectClass f = RBClass._New("", "", "/data/data/"+MainActivity.Host.getPackageName()+"/files" + "/test.txt", "w+");
       f._Call("puts", "I am Jack");
       f._Call("close");
    }
}
```

Ruby code

```
puts $JavaClass

val = $JavaClass.new("from ruby")
puts(val)
val.callback(1234.4564)
val.callback("sdfsdfsdfsdf")
val.SetRubyObject(File);
puts("===========end=========")
```

note : for java, inner class can not assign to script

8.14.3 Assign c# object to scripts

1. To Python

C# code

```
{
SrvGroup._InitRaw("python34", Service);
StarObjectClass python = Service._ImportRawContext("python", "", false, "");

python._Set("CSClass", typeof(CallBackClass));
}

public class CallBackClass
   {
      StarObjectClass PythonClass;
      public CallBackClass(String Info)
      {
         Debug.WriteLine(Info);
      }
      public void print(float val)
      {
         Debug.WriteLine("" + val);
      }
      public void print(String val)
      {
         Debug.WriteLine("" + val);
      }
      public void SetPythonObject(Object rb)
      {
         PythonClass = (StarObjectClass)rb; // Ruby File
         String aa = "";
         StarParaPkgClass data1 = MainPage.Host.SrvGroup._NewParaPkg("b",789,"c",456,"a",123)._AsDict(true);
         Object d1 = PythonClass._Call("dumps", data1, MainPage.Host.SrvGroup._NewParaPkg("sort_keys", true)._AsDict(true));
         Debug.WriteLine("" + d1);
         Object d2 = PythonClass._Call("dumps", data1,null);
         Debug.WriteLine("" + d2);
         Object d3 = PythonClass._Call("dumps", data1, MainPage.Host.SrvGroup._NewParaPkg("sort_keys", true,
"indent",4)._AsDict(true));
         Debug.WriteLine("" + d3);
      }
   }
```

Python code

```
print(CSClass)

val = CSClass("from python")
val.print(1234.4564)
val.print("sdfsdfsdfsdf")
val.SetPythonObject(json);
print("==========end=========")
```

2. To Ruby

C# code

```
{
SrvGroup._InitRaw("ruby", Service);
StarObjectClass ruby = Service._ImportRawContext("ruby", "", false, "");

ruby._Set("$CSClass", typeof(CallBackClass));
}

public class CallBackClass
   {
      StarObjectClass RBClass;
      public CallBackClass(String Info)
      {
         Debug.WriteLine(Info);
      }
      public void print(float val)
      {
         Debug.WriteLine(""+ val);
      }
      public void print(String val)
      {
         Debug.WriteLine("" + val);
      }
      public void SetRubyObject(Object rb)
      {
         RBClass = (StarObjectClass)rb; // Ruby File
         StarObjectClass f = RBClass._New("", "", ApplicationData.Current.LocalFolder.Path + "\\test.txt", "w+");
         f._Call("puts", "I am Jack");
         f._Call("close");
      }
   }
```

Ruby code

```
puts $CSClass
```

```
val = $CSClass.new("from ruby")
val.print(1234.4564)
val.print("sdfsdfsdfsdf")
val.SetRubyObject(File);
puts("===========end=========")
```

8.14.4 Assign Object-C object to scripts

1. To Python

Object-C code

```
{
BasicSRPInterface = SRPInterface ->GetBasicInterface();
   BasicSRPInterface ->InitRaw((VS_CHAR*)"python35",SRPInterface);
   void *python = SRPInterface ->ImportRawContext((VS_CHAR*)"python35",(VS_CHAR*)"",false,NULL);

Star_ObjectCBridge_Init(SRPInterface,NULL,NULL);
   /*---need include --#import <objc/runtime.h>-*/
   SRPInterface ->
ScriptSetObject(python,"CClass",VSTYPE_OBJPTR,(VS_UWORD)_FromObjectC(objc_getClass("TestSRPClass")));
}

@interface TestSRPClass : NSObject{
   NSString *_name;
}

@property (nonatomic, retain) NSString *name;
@property (nonatomic, retain) NSNumber *DoubleValue;
@property (nonatomic) NSInteger IntValue;

+(NSObject*)initTestSRPClass:(NSString *)initName;
-(id)usingPointer:(NSObject *)Which;

@end

@implementation TestSRPClass

@synthesize name;
@synthesize DoubleValue;
@synthesize IntValue;

+(NSObject*)initTestSRPClass:(NSString *)initName
{
```

```
    TestSRPClass *obj = [[TestSRPClass alloc]init];
    obj->name = initName;
    return obj;
}

-(id)usingPointer:(NSObject *)Which
{    return nil;
}
@end
```

Python code

```
print(CClass)

bb=CClass.initTestSRPClass("aaaaaaaaaaaaaa")
bb.usingPointer(Service._New())

print("===========end=========")
```

2. To Ruby

Object-C code

```
{
BasicSRPInterface ->InitRaw((VS_CHAR*)"ruby",SRPInterface);
    void *ruby = SRPInterface ->ImportRawContext((VS_CHAR*)"ruby",(VS_CHAR*)"",false,NULL);

Star_ObjectCBridge_Init(SRPInterface,NULL,NULL);
    /*---need include --#import <objc/runtime.h>-*/
SRPInterface ->
ScriptSetObject(ruby,"$CClass",VSTYPE_OBJPTR,(VS_UWORD)_FromObjectC(objc_getClass("TestSRPClass")));
}

@interface TestSRPClass : NSObject{
    NSString *_name;
}

@property (nonatomic, retain) NSString *name;
@property (nonatomic, retain) NSNumber *DoubleValue;
@property (nonatomic) NSInteger IntValue;

+(NSObject*)initTestSRPClass:(NSString *)initName;
-(id)usingPointer:(NSObject *)which;

@end
```

```
@implementation TestSRPClass

@synthesize name;
@synthesize DoubleValue;
@synthesize IntValue;

+(NSObject*)initTestSRPClass:(NSString *)initName
{
   TestSRPClass *obj = [[TestSRPClass alloc]init];
   obj->name = initName;
   return obj;
}

-(id)usingPointer:(NSObject *)which
{
   return nil;
}
@end
```

Ruby Code

```
print($CClass)

bb=$CClass.initTestSRPClass("aaaaaaaaaaaaaa")
bb.usingPointer(bb)

print("===========end=========")
```

9 Calling lua, python or ruby on android, ios, wp,windows 10

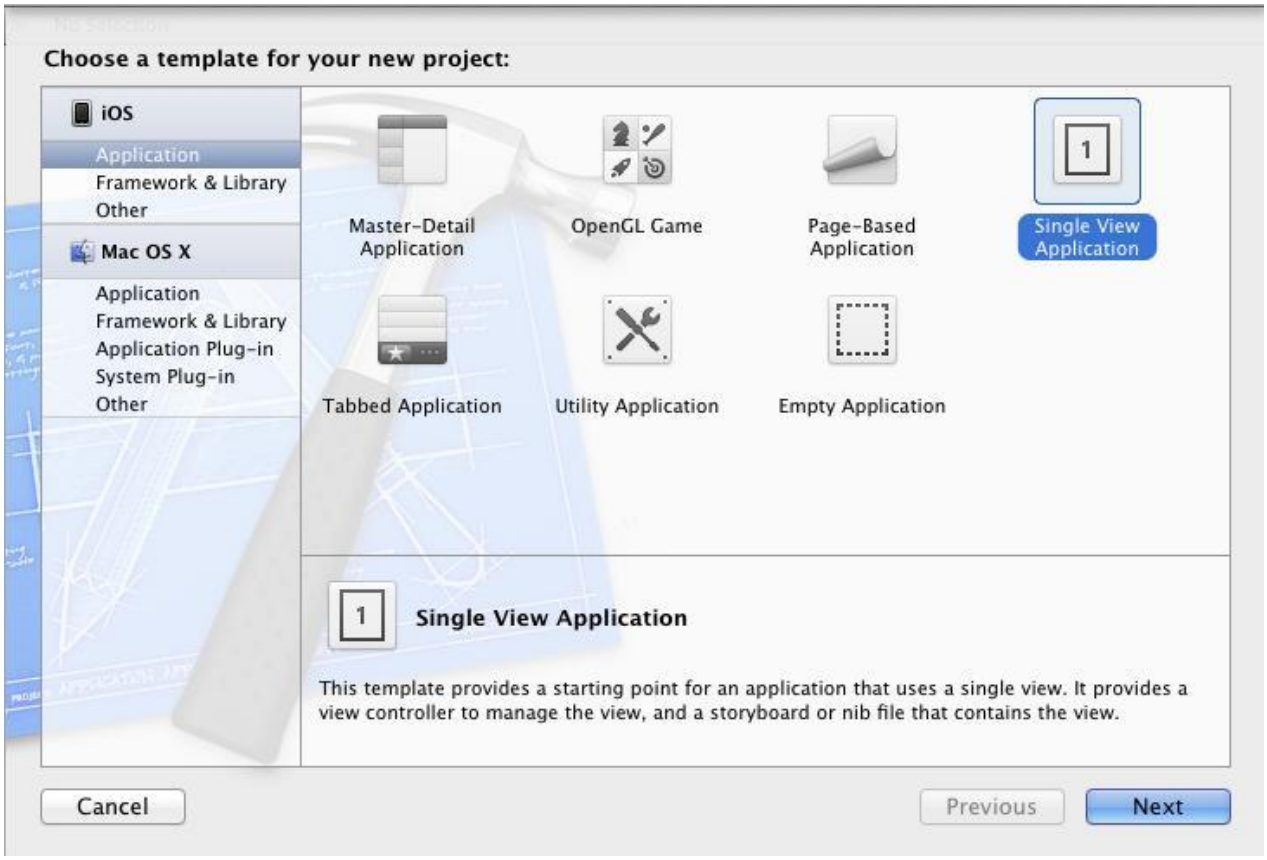
CLE supports android, ios and wp.

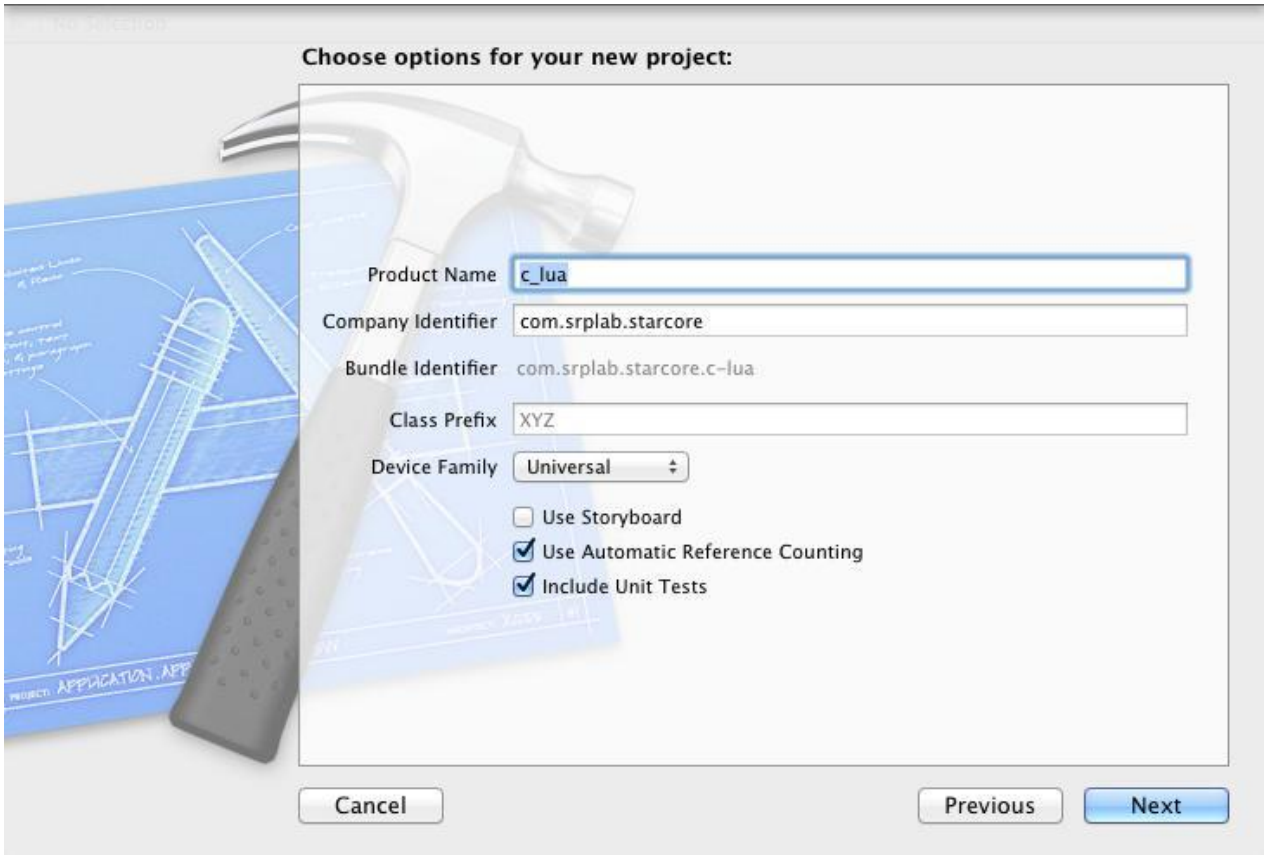
9.1 using cle on ios

For ios, cle supports lua and python script languages.

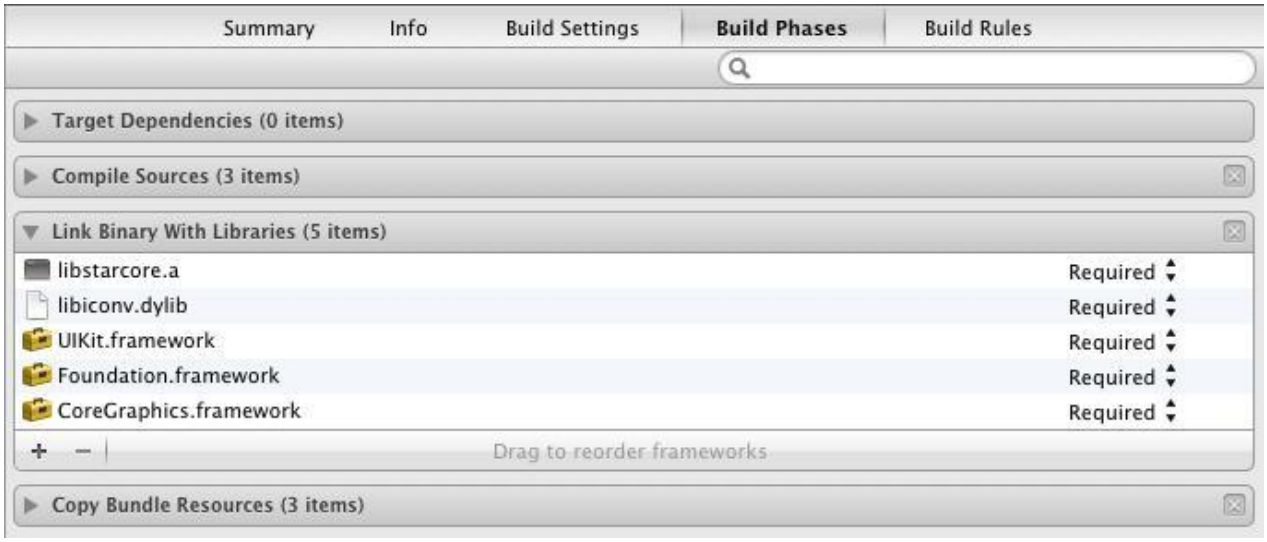
9.1.1 c++ calling lua

1. create project

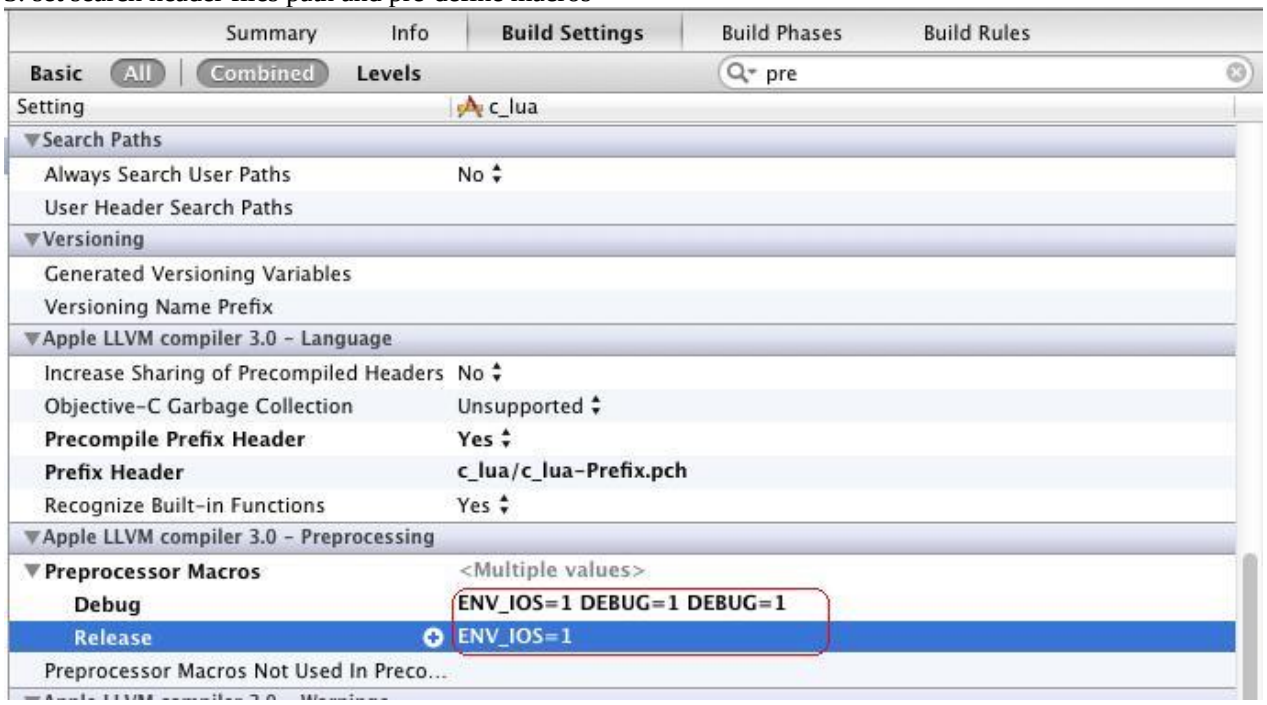




2. add libstarcore.a static library and libiconv.dylib into project



3. set search header files path and pre-define macros



1. source code

```
#include "vsopenapi.h"

static class ClassOfSRPInterface *SRPInterface;

static VS_UWORD MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_UWORD wParam, VS_UWORD
lParam, VS_BOOL *IsProcessed, VS_UWORD Para )
{
   switch( uMsg ){
      case MSG_VSDISPMSG :
      case MSG_VSDISPLUAMSG :
         printf("[core]%s\n",(VS_CHAR *)wParam);
```

```
            break;
        case MSG_DISPMSG :
        case MSG_DISPLUAMSG :
            printf("%s\n",(VS_CHAR *)wParam);
            break;
    }
    return 0;
}

// c function which will be called from lua
static VS_INT32 Add(void *Object,VS_INT32 x,VS_INT32 y)
{
    SRPInterface ->Print("Call From ios, %d,%d",x,y);
    return x + y;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
        // Do any additional setup after loading the view, typically from a nib.
    /* init cle */
    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,   NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];

    const char* destDir = [documentsDirectory UTF8String];
    VS_BOOL Result = StarCore_Init((VS_CHAR *)destDir);

    VS_CORESIMPLECONTEXT Context;

    SRPInterface = VSCoreLib_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,NULL);

    /* run simple lua script */
    VS_CHAR LuaBuf[512];
    sprintf(LuaBuf,"print(\"hello from lua\")");
    SRPInterface ->DoBuffer("lua",LuaBuf,strlen(LuaBuf),"", NULL, NULL, VS_FALSE);

    /* run lua script */
    sprintf(LuaBuf,"SrvGroup = libstarcore._GetSrvGroup()\n");
    strcat(LuaBuf,"Service = SrvGroup:_GetService(\"\",\"\")\n");
    strcat(LuaBuf,"Obj=Service:_New(\"TestClass\");\n");
    strcat(LuaBuf,"function Obj:Add(x,y)\n");
    strcat(LuaBuf,"  local cobj=self._Service.TestClassC:_New();\n");
    strcat(LuaBuf,"  print(cobj:Add(x,y))\n");
    strcat(LuaBuf,"  cobj:_Free()\n");
    strcat(LuaBuf,"  return x+y;\n");
```

```
    strcat(LuaBuf,"end\n");

    SRPInterface ->CheckPassword(VS_FALSE);
    SRPInterface ->DoBuffer("lua",LuaBuf,strlen(LuaBuf),"", NULL, NULL, VS_FALSE);

    /* create object and function which can be called from lua */
    void *AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClassC",NULL,NULL,NULL);
    void *Add_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"Add","VS_INT32 Add(VS_INT32
x,VS_INT32 y);",NULL,NULL,VS_FALSE,VS_FALSE);
    //---Set Function Address
    SRPInterface -> SetAtomicFunction(Add_AtomicFunction,(void *)Add);

    /* call lua function */
    void *Class,*Object;
    Class = SRPInterface ->GetObjectEx(NULL,"TestClass");
    Object = SRPInterface ->MallocObjectL( SRPInterface->GetIDEx(Class),0,NULL);
    printf("Call Function Ret = %lu\n",SRPInterface ->ScriptCall(Object,NULL,"Add","(ii)i",12,34));

    /* clear cle */
    SRPInterface -> Release();
   VSCoreLib_TermSimple(&Context);
}
```

9.1.2 c++ calling python

1. create project

same as above

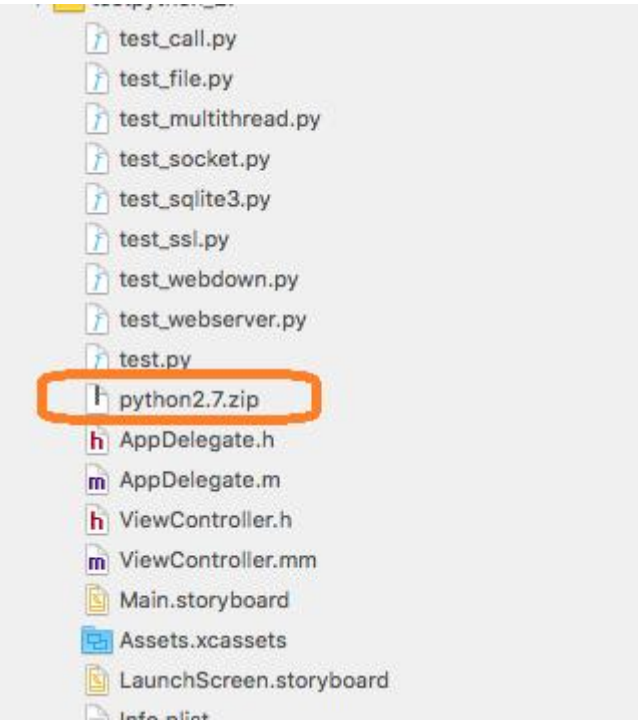
2. add libstarcore.a, libpython2.7.a,static library and libiconv.dylib, libsqlite3, libsqlite3.0 into project

If openssl is used, the libssl.a and libcrypto.a should be add to the project

3. set search header files path and pre-define macros

same as above

4. add python2.7.zip script to project.



5. source code

```
#include "vsopenapi.h"

static class ClassOfSRPInterface *SRPInterface;

static VS_UWORD MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_UWORD wParam, VS_UWORD
lParam, VS_BOOL *IsProcessed, VS_UWORD Para )
{
   switch( uMsg ){
      case MSG_VSDISPMSG :
      case MSG_VSDISPLUAMSG :
         printf("[core]%s\n",(VS_CHAR *)wParam);
         break;
      case MSG_DISPMSG :
      case MSG_DISPLUAMSG :
         printf("%s\n",(VS_CHAR *)wParam);
         break;
   }
   return 0;
}

// c function which will be called from lua
```

```
static VS_INT32 Add(void *Object,VS_INT32 x,VS_INT32 y)
{
   SRPInterface ->Print("Call From ios, %d,%d",x,y);
   return x + y;
}
- (void)viewDidLoad
{
   [super viewDidLoad];
      // Do any additional setup after loading the view, typically from a nib.

   NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,   NSUserDomainMask, YES);
   NSString *documentsDirectory = [paths objectAtIndex:0];

   const char* destDir = [documentsDirectory UTF8String];
   VS_BOOL Result = StarCore_Init((VS_CHAR *)destDir);

   NSString *respaths = [[NSBundle mainBundle] resourcePath];
   const VS_CHAR *res_cpath = [respaths UTF8String];
   VS_CHAR python_path[512];
   VS_CHAR python_home[512];
   sprintf(python_home,"%s/python",res_cpath);
   sprintf(python_path,"%s/python2.7.zip",res_cpath);

   VSCoreLib_InitPython((VS_CHAR*)python_home,(VS_CHAR *)python_path,NULL);

   VS_CORESIMPLECONTEXT Context;

   SRPInterface = VSCoreLib_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,NULL);

   VS_CHAR pyBuf[512];
   sprintf(pyBuf,"print(\"hello from python\")");
   SRPInterface ->DoBuffer("python",pyBuf,strlen(pyBuf),"", NULL, NULL, VS_FALSE);

   sprintf(pyBuf,"SrvGroup = libstarpy._GetSrvGroup()\n");
   strcat(pyBuf,"Service = SrvGroup._GetService(\"\",\"\")\n");
   strcat(pyBuf,"Obj=Service._New(\"TestClass\");\n");
   strcat(pyBuf,"def Obj_Add(self,x,y) :\n");
   strcat(pyBuf,"  cobj=self._Service.TestClassC._New();\n");
   strcat(pyBuf,"  print(cobj.Add(x,y))\n");
   strcat(pyBuf,"  cobj._Free()\n");
   strcat(pyBuf,"  return x+y;\n");
   strcat(pyBuf,"Obj.Add=Obj_Add\n");

   SRPInterface ->CheckPassword(VS_FALSE);
   SRPInterface ->DoBuffer("python",pyBuf,strlen(pyBuf),"", NULL, NULL, VS_FALSE);
```

```
    void *AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClassC",NULL,NULL,NULL);
    void *Add_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"Add","VS_INT32 Add(VS_INT32
x,VS_INT32 y);",NULL,NULL,VS_FALSE,VS_FALSE);
    //---Set Function Address
    SRPInterface -> SetAtomicFunction(Add_AtomicFunction,(void *)Add);

    void *Class,*Object;
        Class = SRPInterface ->GetObjectEx(NULL,"TestClass");
        Object = SRPInterface ->MallocObjectL( SRPInterface->GetIDEx(Class),0,NULL);
        printf("Call Function Ret = %lu\n",SRPInterface ->ScriptCall(Object,NULL,"Add","(ii)i",12,34));

        SRPInterface -> Release();
        VSCoreLib_TermSimple(&Context);

}
```

If python 3.4 version is used, the files named pyhton2.7 should be replaced with python3.4

If python 3.5 version is used, the files named pyhton2.7 should be replaced with python3.5

If _ssl is used, the _ssl module should be init as follows,

```
extern "C" void init_ssl(void);
extern "C" void init_hashlib(void);

@interface ViewController ()

@end
```

And

```
    /* if use openssl, add _hashlib and _ssl; and add libcrypto.a and libssl.a */
    VSImportPythonCModuleDef CModuleDef[]={{"_hashlib",(void*)init_hashlib},{"_ssl",(void*)init_ssl},{NULL,NULL}};
    VSCoreLib_InitPython((VS_CHAR*)python_home,(VS_CHAR *)python_path,CModuleDef);
```

Assign Object-C obejcts to python and called from python directly

```
@interface TestSRPClass : NSObject{
    NSString *_name;
}

@property (nonatomic, retain) NSString *name;
@property (nonatomic, retain) NSNumber *DoubleValue;
@property (nonatomic) NSInteger IntValue;

+(NSObject*)initTestSRPClass:(NSString *)initName;
```

```
-(id)usingPointer:(NSObject *)Which;

@end

@implementation TestSRPClass

@synthesize name;
@synthesize DoubleValue;
@synthesize IntValue;

+(NSObject*)initTestSRPClass:(NSString *)initName
{
    TestSRPClass *obj = [[TestSRPClass alloc]init];
    obj->name = initName;
    return obj;
}

-(id)usingPointer:(NSObject *)Which
{
    return nil;
}
@end
```

Assign TestSRPClass to python

```
void *python = SRPInterface ->ImportRawContext((VS_CHAR*)"python35",(VS_CHAR*)"",false,NULL);

//---test call NSObject
    Star_ObjectCBridge_Init(SRPInterface,NULL,NULL);
    /*---need include --#import <objc/runtime.h>-*/
    SRPInterface ->
ScriptSetObject(python,"CClass",VSTYPE_OBJPTR,(VS_UWORD)_FromObjectC(objc_getClass("TestSRPClass")));
    sprintf(FileBuf,"%s/test_callnsobject.py",res_cpath);
    SRPInterface->DoFile("python35", FileBuf, NULL,NULL,VS_FALSE);
```

Python script:

```
print(CClass)

bb=CClass.initTestSRPClass("aaaaaaaaaaaaaa")
bb.usingPointer(bb)

print("==========end=========")
```

Important:

Please use VSImportPythonCModuleDef CModuleDef[] to load c extension modules. For example,

```
extern "C" void init_imaging(void);
extern "C" void init_imagingmorph(void);
extern "C" void init_imagingft(void);
extern "C" void init_imagingmath(void);
```

```
static VSImportPythonCModuleDef
CModuleDef[]={{"_imaging",(void*)init_imaging},{"_imagingmorph",(void*)init_imagingmorph},{"_imagingft",(void*)init_im
agingft},{"_imagingmath",(void*)init_imagingmath},{NULL,NULL}};

  VSCoreLib_InitPython((VS_CHAR*)python_home,(VS_CHAR *)python_path,CModuleDef);
```

The c extension module is loaded in to gobal space. And the python script must import the module from global space.

"From XXX import _imaging" will failed.

You must use

"import _imaging"

The above limitation may cause a little change to your script.

Unsupported Modules:

Python2.7:

```
_bsddb          _curses          _curses_panel
_tkinter        bsddb185         bz2
dbm             dl               gdbm
imageop         linuxaudiodev    nis
ossaudiodev     readline         spwd
sunaudiodev     _ctypes
```

Python3.4

```
_bz2            _curses          _curses_panel
_dbm            _gdbm            _lzma
_tkinter        nis              ossaudiodev
readline        spwd     _ctypes
```

Python3.5

```
_bz2            _curses          _curses_panel
_dbm            _gdbm            _lzma
_tkinter        nis              ossaudiodev
readline        spwd    _ctypes
```

9.1.3 c++ calling ruby

1. create project

same as above

2. add libstarcore.a, libruby-static.a, libruby-exts.a, libtrans.a, libstar_ruby.a static library and libiconv.dylib, libsqlite3, libsqlite3.0 into project

If Openssl is used, the libssl.a and libcrypt.a should be add to the project.

3. set search header files path and pre-define macros

same as above

4. source code

```
#import "ViewController.h"
#include "vsopenapi.h"

static class ClassOfSRPInterface *SRPInterface;

static VS_UWORD MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_UWORD wParam, VS_UWORD
lParam, VS_BOOL *IsProcessed, VS_UWORD Para )
{
   switch( uMsg ){
      case MSG_VSDISPMSG :
      case MSG_VSDISPLUAMSG :
         printf("[core]%s\n",(VS_CHAR *)wParam);
         break;
      case MSG_DISPMSG :
      case MSG_DISPLUAMSG :
         printf("%s\n",(VS_CHAR *)wParam);
         break;
   }
```

```
    return 0;
}

extern "C" void ruby_init_ext(const char *name, void (*init)(void));
extern "C" void Init_socket();

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,   NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];

    const char* destDir = [documentsDirectory UTF8String];
    VS_BOOL Result = StarCore_Init((VS_CHAR *)destDir);

    VSCoreLib_InitRuby();

    VS_CORESIMPLECONTEXT Context;

    SRPInterface = VSCoreLib_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,NULL);
    SRPInterface ->CheckPassword(VS_FALSE);

    //---set ruby search path
    NSString *respaths = [[NSBundle mainBundle] resourcePath];
    const VS_CHAR *res_cpath = [respaths UTF8String];

    class ClassOfBasicSRPInterface *BasicSRPInterface;

    BasicSRPInterface = SRPInterface ->GetBasicInterface();
    BasicSRPInterface ->InitRaw("ruby", SRPInterface);
    BasicSRPInterface ->Release();

    void *ruby = SRPInterface -> ImportRawContext("ruby", "", VS_FALSE, "");
    void *LOAD_PATH = (void *)SRPInterface -> ScriptGetObject(ruby,"LOAD_PATH", NULL);
    SRPInterface->ScriptCall(LOAD_PATH,NULL, "unshift", "(s)",res_cpath);

    ruby_init_ext("socket.so",Init_socket);

    VS_CHAR rbBuf[512];
```

```
    //sprintf(rbBuf,"puts(\"hello from ruby\")");
    sprintf(rbBuf,"puts($starruby)");
    SRPInterface ->DoBuffer((VS_CHAR*)"ruby",(VS_CHAR*)rbBuf,strlen(rbBuf),(VS_CHAR*)"", NULL, NULL,
VS_FALSE);



    VS_CHAR filename[512];
    sprintf(filename,"%s/test.rb",res_cpath);
    SRPInterface ->DoFile("ruby",filename,NULL,NULL,VS_FALSE);



    SRPInterface -> Release();
    VSCoreLib_TermSimple(&Context);
}
```

If openssl is used, it should be init as follows,

```
//if use md5 sha1 sha2 openssl or rmd160

extern "C" void ruby_init_ext(const char*,void*);
extern "C" void Init_md5(void);
extern "C" void Init_rmd160(void);
extern "C" void Init_sha1(void);
extern "C" void Init_sha2(void);
extern "C" void Init_openssl(void);



@interface ViewController ()

@end
```

```
    BasicSRPInterface ->InitRaw((VS_CHAR*)"ruby",SRPInterface);
    void *ruby = SRPInterface ->ImportRawContext((VS_CHAR*)"ruby",(VS_CHAR*)"",false,NULL);

    ruby_init_ext("openssl.so",(void *)Init_openssl);
    ruby_init_ext("digest/md5.so",(void *)Init_md5);
    ruby_init_ext("digest/rmd160.so",(void *)Init_rmd160);
    ruby_init_ext("digest/sha1.so",(void *)Init_sha1);
    ruby_init_ext("digest/sha2.so",(void *)Init_sha2);
```

Assign Object-C obejcts to ruby and called from ruby directly

```
@interface TestSRPClass : NSObject{
    NSString *_name;
}
```

```
@property (nonatomic, retain) NSString *name;
@property (nonatomic, retain) NSNumber *DoubleValue;
@property (nonatomic) NSInteger IntValue;

+(NSObject*)initTestSRPClass:(NSString *)initName;
-(id)usingPointer:(NSObject *)Which;

@end

@implementation TestSRPClass

@synthesize name;
@synthesize DoubleValue;
@synthesize IntValue;

+(NSObject*)initTestSRPClass:(NSString *)initName
{
   TestSRPClass *obj = [[TestSRPClass alloc]init];
   obj->name = initName;
   return obj;
}

-(id)usingPointer:(NSObject *)Which
{
   return nil;
}
@end
```

Assign TestSRPClass to ruby

```
void *python = SRPInterface ->ImportRawContext((VS_CHAR*)"ruby",(VS_CHAR*)"",false,NULL);

//---test call NSObject
   Star_ObjectCBridge_Init(SRPInterface,NULL,NULL);
   /*---need include --#import <objc/runtime.h>-*/
   SRPInterface ->
ScriptSetObject(ruby,"$CClass",VSTYPE_OBJPTR,(VS_UWORD)_FromObjectC(objc_getClass("TestSRPClass")));
   sprintf(FileBuf,"%s/test_callnsobject.rb",res_cpath);
   SRPInterface->DoFile("ruby", FileBuf, NULL,NULL,VS_FALSE);
```

Ruby script:

```
print($CClass)

bb=$CClass.initTestSRPClass("aaaaaaaaaaaaaa")
```

```
bb.usingPointer(bb)


print("===========end=========")
```

Important:

For c extension modules, please compile to static library,and import as follow,

```
extern "C" void ruby_init_ext(const char*,void*);
extern "C" void Init_sha2(void);
extern "C" void Init_openssl(void);
```

```
   void *ruby = SRPInterface ->ImportRawContext((VS_CHAR*)"ruby",(VS_CHAR*)"",false,NULL);

   ruby_init_ext("openssl.so",(void *)Init_openssl);
   ruby_init_ext("digest/md5.so",(void *)Init_md5);
   ruby_init_ext("digest/rmd160.so",(void *)Init_rmd160);
   ruby_init_ext("digest/sha1.so",(void *)Init_sha1);
   ruby_init_ext("digest/sha2.so",(void *)Init_sha2);
```

Unsupported Modules:

```
Gdbm  tk  tk/tkutil  win32  win32ole fiddle  readline
```

9.1.4 ObjectC bridge for scripts

From cle version 2.50.0, an objectc bridge is provided, which enables scripts to directly access objectc class or instance. After finishing create service, using function "Star_ObjectCBridge_Init" to init the bridge.

```
   VS_CORESIMPLECONTEXT Context;

   SRPInterface = VSCoreLib_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,NULL);
   SRPInterface ->CheckPassword(VS_FALSE);

   Star_ObjectCBridge_Init(SRPInterface,NULL,NULL);
```

Two functions "_FromObjectC" and "_ToObjectC" can be used to wrap objectc to cle object, or get objectc wrapped with cle object. By default, this two callback functions should be set to NULL.

ObjectC bridge can also be initialized from script, using "InitRaw" function with interface name set to "objectc". For example,

```
SrvGroup=_GetSrvGroup(0);
Service = SrvGroup:_GetService("","")

SrvGroup:_InitRaw("objectc",Service);
```

For script languages, use Service.XXX to get obeect c class, for example,

lua:

```
SrvGroup=_GetSrvGroup(0);
Service = SrvGroup:_GetService("","")

dd=Service.NSMutableDictionary()
dd:setObject_forKey("dddddddd","123");
print(dd:objectForKey("123"))
```

ruby:

```
SrvGroup=$starruby._GetSrvGroup(0);
Service = SrvGroup._GetService("","")

dd=Service.NSMutableDictionary.new()
dd.setObject_forKey("dddddddd","123");
print(dd.objectForKey("123"))
```

python:

```
SrvGroup=libstarpy._GetSrvGroup(0);
Service = SrvGroup._GetService("","")

dd=Service.NSMutableDictionary()
dd.setObject_forKey("dddddddd","123");
print(dd.objectForKey("123"))
```

The objectc function being called can push values to lua stack. In this case, the value will be captured by bridge and returned to the caller, and the real return value will be ignored. For example,

```
-(id)usingObject:(NSObject *)obj
{
    /*---return a parapkg to scripts */
    VS_PARAPKGPTR ParaPkg = SRPInterface ->GetParaPkgInterface();
    ParaPkg ->InsertStr(0,"Hello From ObjectC");
    SRPInterface->LuaPushParaPkg(ParaPkg,VS_TRUE);

    return nil;
}
```

lua script:

```
cc = bb:usingObject({text="Hello World"})
print(cc[0])
```

The following types are supported by CLE:

```
_C_CHR
_C_INT
_C_SHT
_C_UCHR
_C_UINT
_C_USHT
_C_LNG
_C_LNG_LNG
_C_ULNG
_C_ULNG_LNG
_C_FLT
_C_DBL
_C_BOOL
_C_CHARPTR

_C_ID
_C_CLASS
_C_SEL
_C_PTR
```

The following types are not supported

```
_C_ARY_B
_C_ARY_E
_C_BFLD
_C_STRUCT_B
_C_STRUCT_E
_C_UNION_E
```

When call methods of object c class, the ':' of method selector must be replaced with '_', for example "colorWithRed_green_blue_alpha" corresponding to "colorWithRed:green:blue:alpha"

Note: for CLE, Method name of Object-C class must not start with '_';

An example:

```
@interface TestSRPClass : NSObject{
    NSString *_name;
}

@property (nonatomic, retain) NSString *name;
@property (nonatomic, retain) NSNumber *DoubleValue;
@property (nonatomic) NSInteger IntValue;
```

```
+(NSObject*)initTestSRPClass:(NSString *)initName;
-(id)usingPointer:(NSObject *)Which;

@end
```

```
@implementation TestSRPClass

@synthesize name;
@synthesize DoubleValue;
@synthesize IntValue;

+(NSObject*)initTestSRPClass:(NSString *)initName
{
   TestSRPClass *obj = [[TestSRPClass alloc]init];
   obj->name = initName;
   return obj;
}

-(id)usingPointer:(NSObject *)Which
{
   return nil;
}
@end
```

Python script:

```
bb=Service.TestSRPClass.initTestSRPClass("aaaaaaaaaaaaaa")
bb.usingPointer(bb)
```

You can also set Object-C object to script global space and use it in script directly, for example.

```
void *python = SRPInterface ->ImportRawContext((VS_CHAR*)"python35",(VS_CHAR*)"",false,NULL);

/Star_ObjectCBridge_Init(SRPInterface,NULL,NULL);
SRPInterface ->
ScriptSetObject(python,"CClass",VSTYPE_OBJPTR,(VS_UWORD)Self_FromObjectC(objc_getClass("TestSRPClass")));
```

Python:

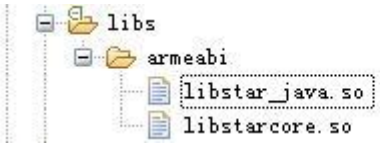
```
bb=CClass.initTestSRPClass("aaaaaaaaaaaaaa")
bb.usingPointer(bb)
```

9.2 using cle on android

For android, cle supports lua and python script languages, and java is the host language for developing apps.

1. copy starcore_android_r2.X.jar to the libs directory of the project.

2. copy libstar_java.so, libstarcore.so and libstarpy.so to libs directory, as follows:

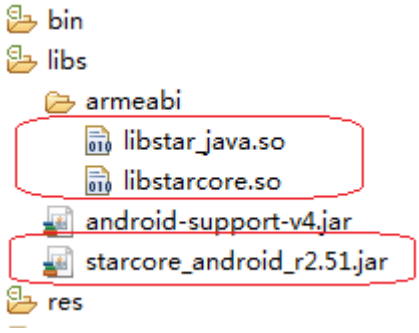


and using the following code to init cle

```
import com.srplab.www.starcore.*;

StarCoreFactory starcore;
StarServiceClass Service;

StarCoreFactoryPath.StarCoreCoreLibraryPath = this.getApplicationInfo().nativeLibraryDir;
StarCoreFactoryPath.StarCoreShare libraryraryPath = this.getApplicationInfo().nativeLibraryDir;
StarCoreFactoryPath.StarCoreOperationPath = "/data/data/"+getPackageName()+"/files";

starcore= StarCoreFactory.GetFactory();
Service=starcore._InitSimple("test","123",0,0,"");
```

9.2.1 java calling lua

1. create project and add libs.



2. lua code to be called.

Testlua.lua

```
function tt(a,b)
   print(a,b)
   return 6666,7777
end
g1 = 123
c={x=456}
```

```
function c:yy(a,b,z)
   print(self)
   print(a,b,z)
   return {33,Type="mytype"}
end
```

Test_CallJava.lua

```
print(JavaClass)


val = JavaClass("from lua")
val:callback(1234.4564)
val:callback("sdfsdfsdfsdf")
val:SetLuaObject({"aaa","bbb"});
print("==========end=========")
```

3. java code.

```
public class MainActivity extends Activity {
    public static MainActivity Host;
    public StarSrvGroupClass SrvGroup;

    private void copyFile(Activity c, String Name,String desPath) throws IOException {
        File outfile = null;
        if( desPath != null )
            outfile = new File("/data/data/"+getPackageName()+"/files/"+desPath+Name);
        else
            outfile = new File("/data/data/"+getPackageName()+"/files/"+Name);
        //if (!outfile.exists()) {
            outfile.createNewFile();
            FileOutputStream out = new FileOutputStream(outfile);
            byte[] buffer = new byte[1024];
            InputStream in;
            int readLen = 0;
            if( desPath != null )
                in = c.getAssets().open(desPath+Name);
            else
                in = c.getAssets().open(Name);
            while((readLen = in.read(buffer)) != -1){
                out.write(buffer, 0, readLen);
            }
            out.flush();
            in.close();
            out.close();
        //}
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Host = this;
```

```
        File destDir = new File("/data/data/"+getPackageName()+"/files");
        if(!destDir.exists())
            destDir.mkdirs();
        try{
            copyFile(this,"testlua.lua",null);
            copyFile(this,"test_calljava.lua",null);
        }
        catch(Exception e){
            System.out.println(e);
        }
        /*----init starcore----*/
        StarCoreFactoryPath.StarCoreCoreLibraryPath =
this.getApplicationInfo().nativeLibraryDir;
        StarCoreFactoryPath.StarCoreShareLibraryPath =
this.getApplicationInfo().nativeLibraryDir;
        StarCoreFactoryPath.StarCoreOperationPath =
"/data/data/"+getPackageName()+"/files";

        StarCoreFactory starcore= StarCoreFactory.GetFactory();
        StarServiceClass Service=starcore._InitSimple("test","123",0,0);
        SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");
        Service._CheckPassword(false);

        /*----run lua code----*/
        SrvGroup._InitRaw("lua",Service);
        StarObjectClass lua = Service._ImportRawContext("lua","",false,"");

        String CorePath = "/data/data/"+getPackageName()+"/files";
        //--load lua module ---*/
        SrvGroup._LoadRawModule("lua","",CorePath+"/testlua.lua",false);
        //--call lua function tt, the return contains two integer, which will be
wrapped into StarObjectClass
        StarObjectClass retobj = (StarObjectClass)lua._Call("tt","hello ","world");
        System.out.println("ret from lua :  "+retobj._Get(1)+"   "+retobj._Get(2));
        //--get global int value g1--------*/
        System.out.println("lua value g1 :  "+lua._Get("g1"));
        //--get global table value c, which is a table with function, it will be
mapped to cle object --------*/
        StarObjectClass c = lua._GetObject("c");
        //--get int value x from c--------*/
        System.out.println("c value x :  "+c._Get("x"));
        //--call c function yy, the return is a table, which will be mapped to cle
object ---*/
        StarObjectClass yy = (StarObjectClass)c._Call("yy",c,"hello ","world","!");
        System.out.println("yy value [1] :  "+yy._Get(1));
        System.out.println("yy value [Type] :  "+yy._Get("Type"));

        /*-------*/
        lua._Set("JavaClass", CallBackClass.class);
        Service._DoFile("lua", CorePath + "/test_calljava.lua", "");  //should not
use null

    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
```

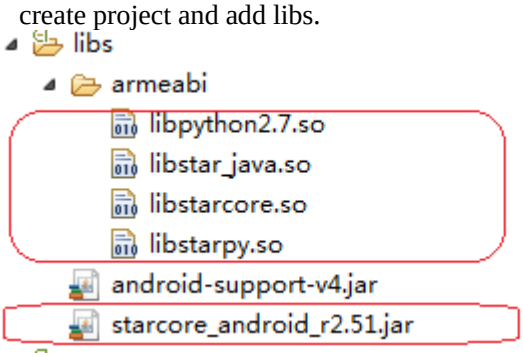
```
        return true;
    }

}
```

CallBackClass.java

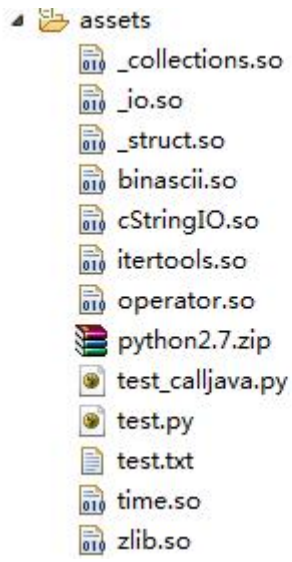
```
public class CallBackClass {
    StarObjectClass LuaClass;
    public CallBackClass(String Info)
    {
        System.out.println(Info);
    }
    public void callback(float val)
    {
        System.out.println("" + val);
    }
    public void callback(String val)
    {
        System.out.println("" + val);
    }
    public void SetLuaObject(Object[] rb)
    {
        for(Object Item : rb){
            System.out.println("" + Item);
        }
    }
}
```

9.2.2 java calling python

1. create project and add libs.



2. add python extensions and files to be call to assets folder



3. python code to be called

test.py

```
from __future__ import division
print(division)

import sys
print(sys.path)
import zipfile
import os
print os
print os.uname()

def testread(name) :
    text_file = open(name, "rt")
    print text_file.readline()
    text_file.close()
```

test_calljava.py

```
import imp  #test load path

import json
print(JavaClass)

val = JavaClass("from python")
val.callback(1234.4564)
val.callback("sdfsdfsdfsdf")
val.SetPythonObject(json);
print("==========end=========")
```

4. java code

```
public class MainActivity extends Activity {

    public static MainActivity Host;
    public StarSrvGroupClass SrvGroup;

    private void copyFile(Activity c, String Name,String desPath) throws IOException {
        File outfile = null;
        if( desPath != null )
            outfile = new File("/data/data/"+getPackageName()+"/files/"+desPath+Name);
        else
            outfile = new File("/data/data/"+getPackageName()+"/files/"+Name);
        if (!outfile.exists()) {
            outfile.createNewFile();
            FileOutputStream out = new FileOutputStream(outfile);
            byte[] buffer = new byte[1024];
            InputStream in;
            int readLen = 0;
            if( desPath != null )
                in = c.getAssets().open(desPath+Name);
            else
                in = c.getAssets().open(Name);
            while((readLen = in.read(buffer)) != -1){
                out.write(buffer, 0, readLen);
            }
            out.flush();
            in.close();
            out.close();
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Host = this;

        File destDir = new File("/data/data/"+getPackageName()+"/files");
        if(!destDir.exists())
            destDir.mkdirs();
        java.io.File python2_7_libFile = new
java.io.File("/data/data/"+getPackageName()+"/files/python2.7.zip");
        if( !python2_7_libFile.exists() ){
            try{
                copyFile(this,"python2.7.zip",null);
            }
            catch(Exception e){
            }
        }
        try{
            copyFile(this,"zlib.so",null);
            copyFile(this,"_struct.so",null);
            copyFile(this,"time.so",null);
            copyFile(this,"binascii.so",null);
            copyFile(this,"cStringIO.so",null);
            copyFile(this,"_collections.so",null);
            copyFile(this,"operator.so",null);
            copyFile(this,"itertools.so",null);
```

```
            copyFile(this,"_io.so",null);
        }
        catch(Exception e){
            System.out.println(e);
        }
        //----a test file to be read using python, we copy it to files directory
        try{
            copyFile(this,"test.txt","");
            copyFile(this,"test_calljava.py","");
        }
        catch(Exception e){
            System.out.println(e);
        }
        /*----load test.py----*/
        String pystring = null;
        try{
            AssetManager assetManager = getAssets();
            InputStream dataSource = assetManager.open("test.py");
            int size=dataSource.available();
            byte[] buffer=new byte[size];
            dataSource.read(buffer);
            dataSource.close();
            pystring=new String(buffer);
        }
        catch(IOException e ){
            System.out.println(e);
        }
        /*----init starcore----*/
        StarCoreFactoryPath.StarCoreCoreLibraryPath =
this.getApplicationInfo().nativeLibraryDir;
        StarCoreFactoryPath.StarCoreShareLibraryPath =
this.getApplicationInfo().nativeLibraryDir;
        StarCoreFactoryPath.StarCoreOperationPath =
"/data/data/"+getPackageName()+"/files";

        StarCoreFactory starcore= StarCoreFactory.GetFactory();
        StarServiceClass Service=starcore._InitSimple("test","123",0,0);
        SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");
        Service._CheckPassword(false);

        /*----run python code----*/
        SrvGroup._InitRaw("python",Service);
        StarObjectClass python = Service._ImportRawContext("python","",false,"");
        python._Call("import", "sys");

        StarObjectClass pythonSys = python._GetObject("sys");
        StarObjectClass pythonPath = (StarObjectClass)pythonSys._Get("path");

   pythonPath._Call("insert",0,"/data/data/"+getPackageName()+"/files/python2.7.zip");
        pythonPath._Call("insert",0,"/data/data/"+getPackageName()+"/lib");
        pythonPath._Call("insert",0,"/data/data/"+getPackageName()+"/files");

        python._Call("execute",pystring);
        python._Call("testread", "/data/data/"+getPackageName()+"/files/test.txt");

        String CorePath = "/data/data/"+getPackageName()+"/files";
        python._Set("JavaClass", CallBackClass.class);
        Service._DoFile("python", CorePath + "/test_calljava.py", "");
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}
```

CallBackClass.java

```
public class CallBackClass {
    StarObjectClass PythonClass;
    public CallBackClass(String Info)
    {
        System.out.println(Info);
    }
    public void callback(float val)
    {
        System.out.println("" + val);
    }
    public void callback(String val)
    {
        System.out.println("" + val);
    }
    public void SetPythonObject(Object rb)
    {
        PythonClass = (StarObjectClass)rb;
        String aa = "";
        StarParaPkgClass data1 =
MainActivity.Host.SrvGroup._NewParaPkg("b",789,"c",456,"a",123)._AsDict(true);
        Object d1 = PythonClass._Call("dumps", data1,
MainActivity.Host.SrvGroup._NewParaPkg("sort_keys", true)._AsDict(true));
        System.out.println("" + d1);
        Object d2 = PythonClass._Call("dumps", data1,null);
        System.out.println("" + d2);
        Object d3 = PythonClass._Call("dumps", data1,
MainActivity.Host.SrvGroup._NewParaPkg("sort_keys", true, "indent",4)._AsDict(true));
        System.out.println("" + d3);
    }
}
```

Note for Python3.X

The python core library must be load mannualy before any python code is called, for example,

```
        try{
            //--load python34 core library first;
            System.load("/data/data/"+getPackageName()+"/lib/libpython3.4m.so");
        }
        catch(UnsatisfiedLinkError ex)
        {
```

```
            System.out.println(ex.toString());
        }
        /*----init starcore----*/
        StarCoreFactoryPath.StarCoreCoreLibraryPath = this.getApplicationInfo().nativeLibraryDir;
        StarCoreFactoryPath.StarCoreShareLibraryPath = this.getApplicationInfo().nativeLibraryDir;
        StarCoreFactoryPath.StarCoreOperationPath = "/data/data/"+getPackageName()+"/files";
```

Unsupported Modules:

Python2.7:

```
_bsddb          _curses          _curses_panel
_tkinter        bsddb185         bz2
dbm             dl               gdbm
imageop         linuxaudiodev    nis
ossaudiodev     readline         spwd
sunaudiodev
```

Python3.4

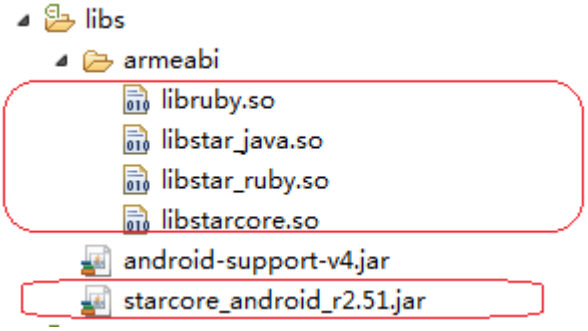
```
_bz2            _curses          _curses_panel
_dbm            _gdbm            _lzma
_tkinter        nis              readline
spwd
```

Python3.5

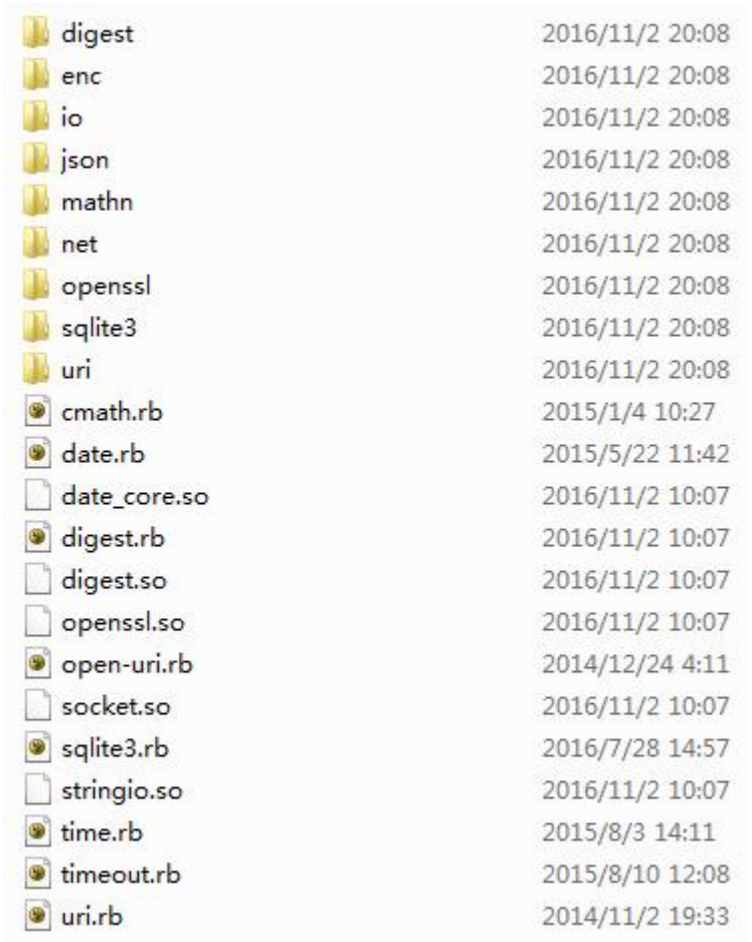
```
_bz2            _curses          _curses_panel
_dbm            _gdbm            _lzma
_tkinter        nis              readline
spwd
```

9.2.3 java calling ruby

1. create project and add libs.



2. put the ruby modules into one folder, and pack them to zip file.



When activity starts, unzip the files with directory to the phone.

```
private static boolean CreatePath(String Path){
            File destCardDir = new File(Path);
    if(!destCardDir.exists()){
     int Index = Path.lastIndexOf(File.separator.charAt(0));
     if( Index < 0 ){
            if( destCardDir.mkdirs() == false )
                return false;
     }else{
            String ParentPath = Path.substring(0, Index);
            if( CreatePath(ParentPath) == false )
                return false;
            if( destCardDir.mkdirs() == false )
                return false;
     }
    }
    return true;
  }
  private static boolean unzip(InputStream zipFileName, String outputDirectory,Boolean OverWriteFlag ) {
    try {
      ZipInputStream in = new ZipInputStream(zipFileName);
```

```
        ZipEntry entry = in.getNextEntry();
        byte[] buffer = new byte[1024];
        while (entry != null) {
           File file = new File(outputDirectory);
           file.mkdir();
           if (entry.isDirectory()) {
              String name = entry.getName();
              name = name.substring(0, name.length() - 1);
              if( CreatePath(outputDirectory + File.separator + name) == false )
                    return false;
           } else {
             String name = outputDirectory + File.separator + entry.getName();
             int Index = name.lastIndexOf(File.separator.charAt(0));
             if( Index < 0 ){
                    file = new File(outputDirectory + File.separator + entry.getName());
             }else{
                    String ParentPath = name.substring(0, Index);
                    if( CreatePath(ParentPath) == false )
                          return false;
                    file = new File(outputDirectory + File.separator + entry.getName());
             }
              if( !file.exists() || OverWriteFlag == true){
                    file.createNewFile();
                    FileOutputStream out = new FileOutputStream(file);
                    int readLen = 0;
               while((readLen = in.read(buffer)) != -1){
                  out.write(buffer, 0, readLen);
               }
                    out.close();
              }
           }
           entry = in.getNextEntry();
        }
        in.close();
        return true;
     } catch (FileNotFoundException e) {
        e.printStackTrace();
        return false;
     } catch (IOException e) {
        e.printStackTrace();
        return false;
     }
  }
```

```
    File destDir = new File("/data/data/"+getPackageName()+"/files");
```

```
if(!destDir.exists())
 destDir.mkdirs();
//----unzip the assets to files
try{
   InputStream in = getAssets().open("assets.zip");   // assets.zip is ruby module file package
   unzip(in, "/data/data/"+getPackageName()+"/files", true);
}
catch(Exception ex)
{

}
```

3. ruby file to be called.

Testrb.rb

```
def tt(a,b)
   puts(a,b)
   return 666,777
end
$g1 = 123
def yy(a,b,z)
   puts(a,b,z)
   return {'jack'=> 4098, 'sape'=> 4139}
end
class Multiply
   def initialize(x,y)
      @a = x
      @b = y
   end

   def multiply(a,b)
      puts("multiply....",self,a,b)
      return a * b
   end
end
```

test_calljava.rb

```
puts $JavaClass

val = $JavaClass.new("from ruby")
puts(val)
val.callback(1234.4564)
val.callback("sdfsdfsdfsdf")
val.SetRubyObject(File);
```

```
puts("==========end=========")
```

4. java code

```
public class MainActivity extends Activity {

    public static MainActivity Host;
    public StarSrvGroupClass SrvGroup;

    private void copyFile(Activity c, String Name,String desPath) throws IOException {
        File outfile = null;
        if( desPath != null )
            outfile = new File("/data/data/"+getPackageName()+"/files/"+desPath+Name);
        else
            outfile = new File("/data/data/"+getPackageName()+"/files/"+Name);
        //if (!outfile.exists()) {
            outfile.createNewFile();
            FileOutputStream out = new FileOutputStream(outfile);
            byte[] buffer = new byte[1024];
            InputStream in;
            int readLen = 0;
            if( desPath != null )
               in = c.getAssets().open(desPath+Name);
            else
               in = c.getAssets().open(Name);
            while((readLen = in.read(buffer)) != -1){
                out.write(buffer, 0, readLen);
            }
            out.flush();
            in.close();
            out.close();
        //}
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Host = this;

        File destDir = new File("/data/data/"+getPackageName()+"/files");
        if(!destDir.exists())
            destDir.mkdirs();
        try{
            copyFile(this,"testrb.rb",null);
            copyFile(this,"cmath.rb",null);
            copyFile(this,"test_calljava.rb",null);
        }
        catch(Exception e){
            System.out.println(e);
        }

        try{
            System.load("/data/data/"+getPackageName()+"/lib/libruby.so");
        }
        catch(UnsatisfiedLinkError ex)
        {
```

```
            System.out.println(ex.toString());
        }

        /*----init starcore----*/
        StarCoreFactoryPath.StarCoreCoreLibraryPath =
this.getApplicationInfo().nativeLibraryDir;
        StarCoreFactoryPath.StarCoreShareLibraryPath =
this.getApplicationInfo().nativeLibraryDir;
        StarCoreFactoryPath.StarCoreOperationPath =
"/data/data/"+getPackageName()+"/files";

        StarCoreFactory starcore= StarCoreFactory.GetFactory();
        StarServiceClass Service=starcore._InitSimple("test","123",0,0);
        SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");
        Service._CheckPassword(false);

        SrvGroup._InitRaw("ruby",Service);
        StarObjectClass ruby = Service._ImportRawContext("ruby","",false,"");
        System.out.println(ruby._Get("LOAD_PATH"));
        System.out.println(ruby._Get("File"));
/*
        StarParaPkgClass para = (StarParaPkgClass)ruby._Get("LOAD_PATH");

        for (Object obj : para )
            System.out.println(obj);
*/
        StarObjectClass LOAD_PATH = (StarObjectClass)ruby._R("LOAD_PATH");
        System.out.println(LOAD_PATH);
        LOAD_PATH._Call("unshift", "/data/data/"+getPackageName()+"/files");

        StarObjectClass para = (StarObjectClass)ruby._Get("LOAD_PATH");

        for (Object obj : para )
            System.out.println(obj);

        ruby._Call("require","cmath");
        System.out.println(ruby._Get("CMath"));

        //--load ruby module ---*/

    SrvGroup._LoadRawModule("ruby","","/data/data/"+getPackageName()+"/files/testrb.rb
",false);
        //--attach object to global ruby space ---*/
        StarObjectClass object = Service._ImportRawContext("ruby","",false,"");
        //--call ruby function tt, the return contains two integer, which will be
packed into parapkg ---*/
        StarObjectClass RetObj = (StarObjectClass)object._Call("tt","hello ","world");
        System.out.println("ret from ruby : "+RetObj._Get(0)+"  "+RetObj._Get(1));
        //--get global int value g1--------*/
        System.out.println("ruby value g1 :  "+object._Get("g1"));
        //--get global class Multiply
        StarObjectClass Multiply =
Service._ImportRawContext("ruby","Multiply",true,"");
        StarObjectClass multiply = Multiply._New("","",33,44);
        //--call instance method multiply
        System.out.println("instance multiply = "+multiply._Call("multiply",11,22));

        String CorePath = "/data/data/"+getPackageName()+"/files";
        ruby._Set("$JavaClass", CallBackClass.class);
```

```
        Service._DoFile("ruby", CorePath + "/test_calljava.rb", "");  //should not
use null

    }
```

CallBackClass.java

```
public class CallBackClass {
    StarObjectClass RBClass;
    public CallBackClass(String Info)
    {
        System.out.println(Info);
    }
    public void callback(float val)
    {
        System.out.println("" + val);
    }
    public void callback(String val)
    {
        System.out.println("" + val);
    }
    public void SetRubyObject(Object rb)
    {
        RBClass = (StarObjectClass)rb; // Ruby File
        StarObjectClass f = RBClass._New("", "",
"/data/data/"+MainActivity.Host.getPackageName()+"/files" + "/test.txt", "w+");
        f._Call("puts", "I am Jack");
        f._Call("close");
    }
}
```

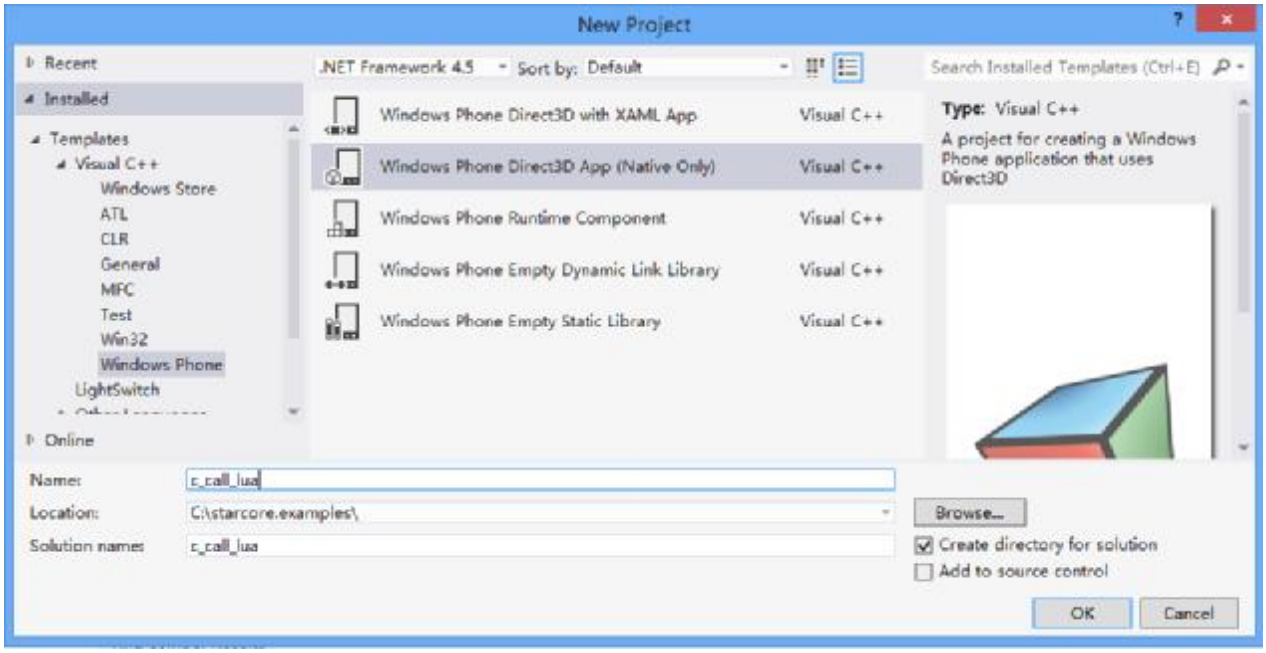
Unsupported Modules:

Dbm gdbm readline tk tk/tkutil win32 win32ole fiddle

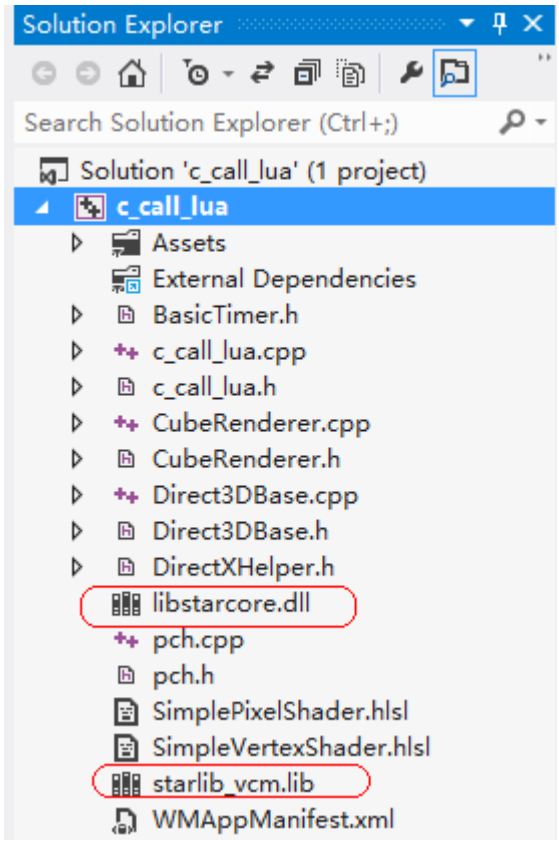
9.3 using cle on wp, windows 10

9.3.1 native calling lua

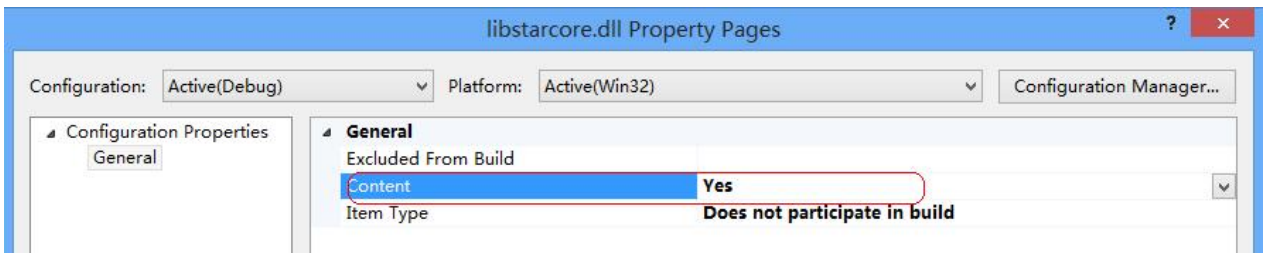
1. Create native project



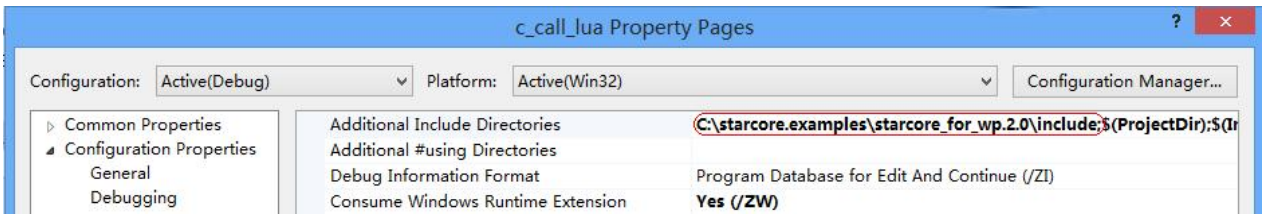
2. Add libstarcore.dll and starlib_vcm.lib to the project

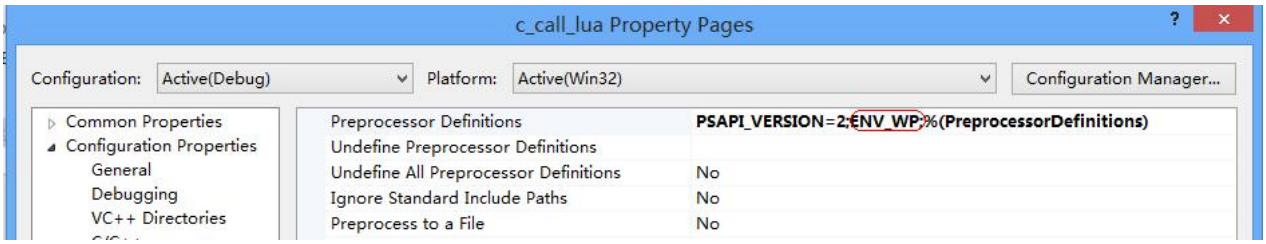


set property of libstarcore.dll.



3. set include directories and predefined macro



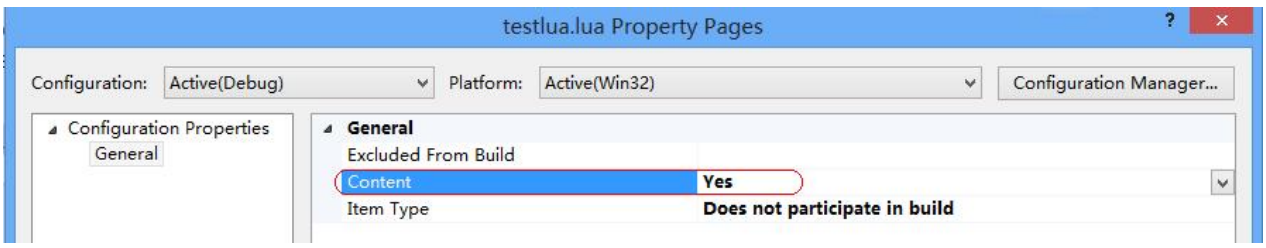


4. add lua files to be called

testlua.lua

```
function tt(a,b)
   print(a,b)
   return 6666,7777
end
g1 = 123
c={x=456}
function c:yy(a,b,z)
   print(self)
   print(a,b,z)
   return {33,Type="mytype"}
end
```

set property of testlua.lua



5. edit source code

```
#include "vsopenapi.h"
```

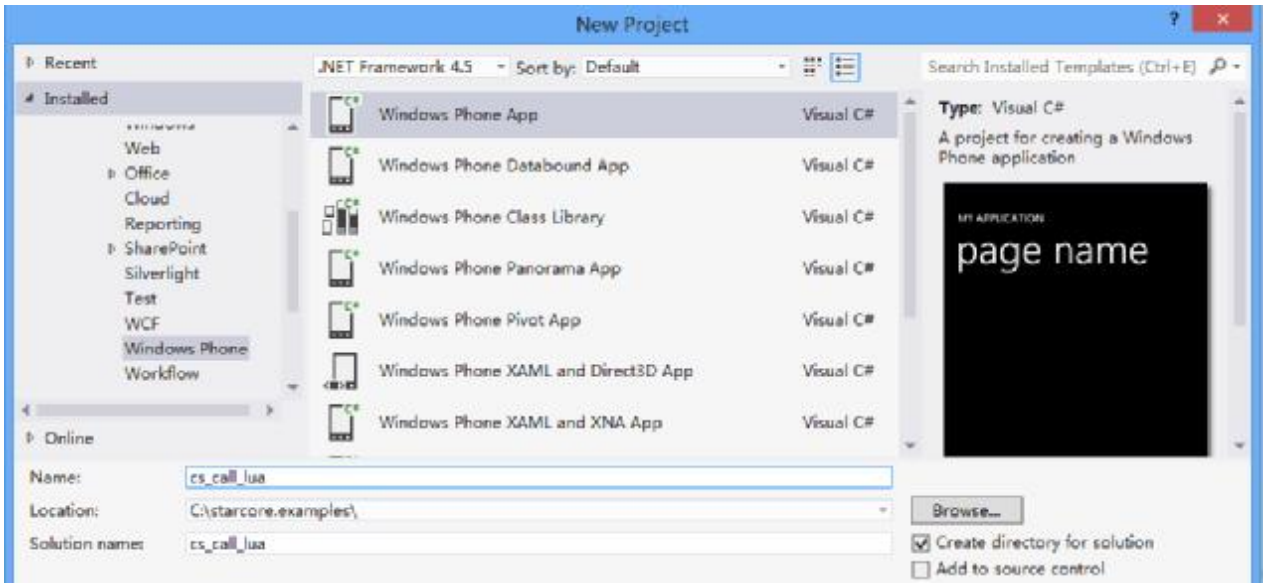
```
int test_main()
{
    class ClassOfSRPInterface *SRPInterface;
    class ClassOfBasicSRPInterface *BasicSRPInterface;
    VS_CORESIMPLECONTEXT Context;

    SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
    if( SRPInterface == NULL ){
        printf("init starcore fail\n");
        return -1;
    }
    OutputDebugString(L"init starcore success\n");
    BasicSRPInterface = SRPInterface ->GetBasicInterface();

    BasicSRPInterface ->Print("%s",BasicSRPInterface->GetCorePath());
    BasicSRPInterface ->Print("%s",BasicSRPInterface->GetUserPath());
    BasicSRPInterface ->Print("%s",BasicSRPInterface->GetLocalIP());
    /*----init lua raw interface ---*/
    BasicSRPInterface ->InitRaw("lua",SRPInterface);
    /*----load lua module ---*/
    VS_CHAR TempBuf[512];
    sprintf(TempBuf,"%s\\testlua.lua",BasicSRPInterface->GetCorePath());
    VS_BOOL LoadResult = BasicSRPInterface -
>LoadRawModule("lua","testlua",TempBuf,VS_FALSE,NULL);
    /*----attach object to global lua space ---*/
    void *Object = SRPInterface ->ImportRawContext("lua","",false,NULL);
    /*----call lua function tt, the return contains two integer, which will be packed into
parapkg ---*/
    class ClassOfSRPParaPackageInterface *ParaPkg;
    ParaPkg = (class ClassOfSRPParaPackageInterface *)SRPInterface -
>ScriptCall(Object,NULL,"tt","(ss)p","hello ","world");
    BasicSRPInterface -> Print("ret from lua :  %d,  %d",ParaPkg->GetInt(0),ParaPkg-
>GetInt(1));
    /*----get global int value g1--------*/
    BasicSRPInterface -> Print("lua value g1 :  %d",SRPInterface -
>ScriptGetInt(Object,"g1"));
    /*----get global table value c, which is a table with function, it will be mapped to
cle object --------*/
    void *c = (void *)SRPInterface ->ScriptGetObject(Object,"c",NULL);
    /*----get int value x from c--------*/
    BasicSRPInterface -> Print("c value x :  %d",SRPInterface ->ScriptGetInt(c,"x"));
    /*----call c function yy, the return is a table, which will be mapped to cle object --
-*/
    void *yy = (void *)SRPInterface ->ScriptCall(c,NULL,"yy","(osss)o",c,"hello
","world","!");
    BasicSRPInterface -> Print("yy value [1] :  %d",SRPInterface ->ScriptGetInt(yy,"1"));
    BasicSRPInterface -> Print("yy value [Type] :  %s",SRPInterface -
>ScriptGetStr(yy,"Type"));

    SRPInterface -> Release();
    VSCore_TermSimple(&Context);
    return 0;
}
```

add above code to any where, and run the program

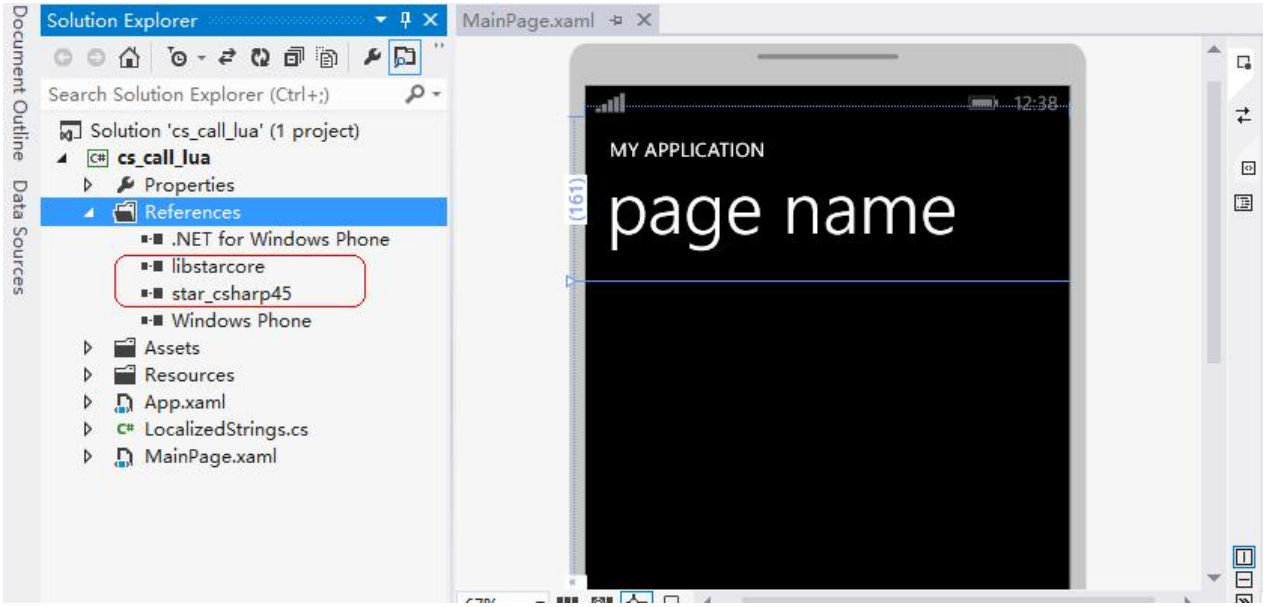
For windows 10, libstarcore and star_csharp45 should be replaced with Libstarcore and Star_csharp

9.3.2 c# calling lua

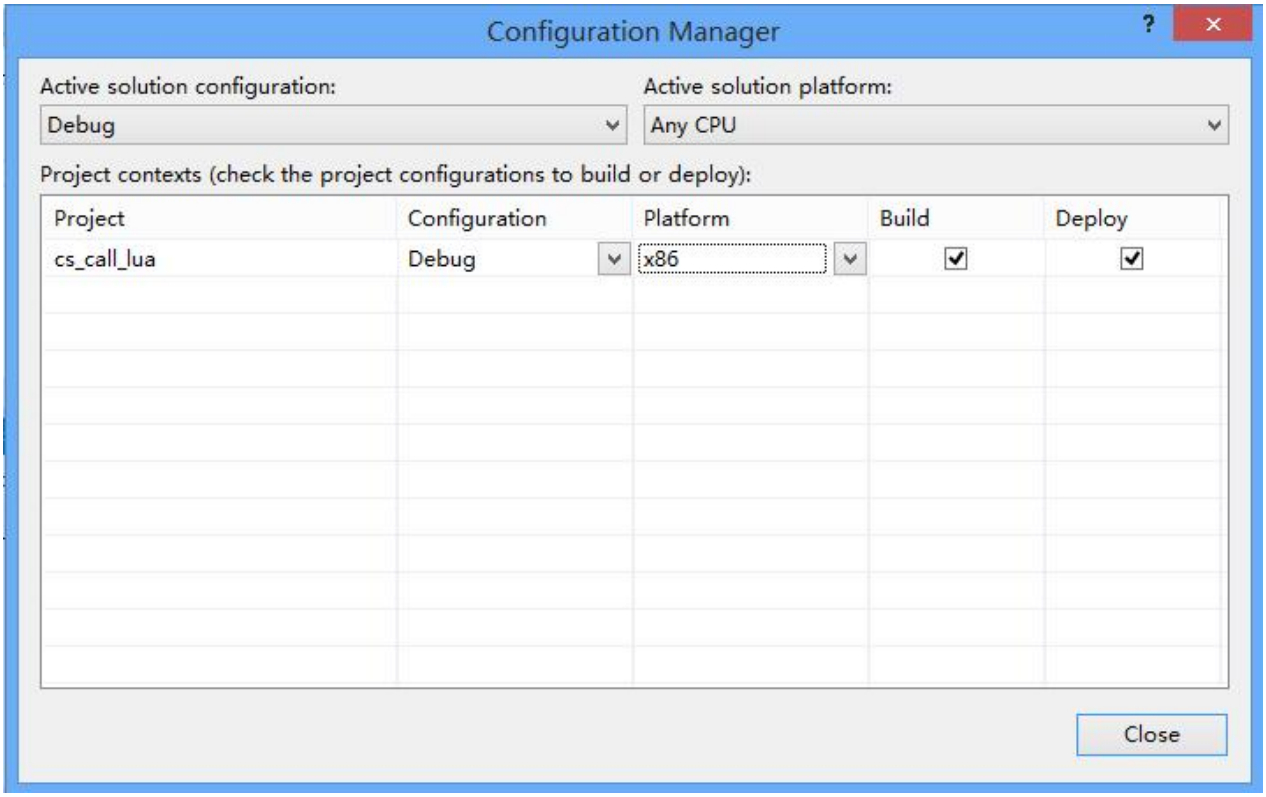
1. Create c# project



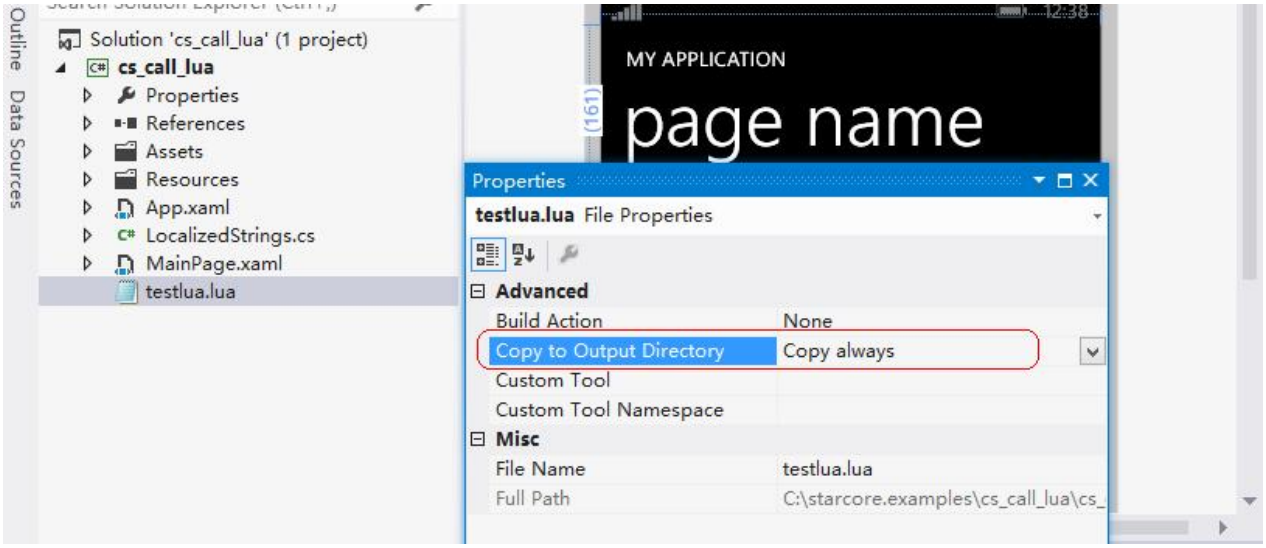
2. Add reference libstarcore and Star_csharp45



change target platform to x86 or arm.



3. add lua files to be called



4. edit source code

....

```
using libstarcore;
using Star_csharp45;

namespace cs_call_lua
{
```

```
    public class StarCoreContext
    {
        public static StarCoreFactory starcore = null;
        public static StarServiceClass Service = null;
        public static StarSrvGroupClass SrvGroup = null;
        public static string Path = null;
    }

    class MyStarCallBackClass : StarCallBackClass
    {
        public MyStarCallBackClass(StarCoreFactory starcore) : base(starcore)
{ starcore._RegMsgCallBack(this, "CallBack"); }
        public Object[] CallBack(Int32 ServiceGroupID, Int32 uMes, Object wParam, Object
lParam)
        {
            if (uMes == _GetInt("MSG_VSDISPMSG") || uMes == _GetInt("MSG_VSDISPLUAMSG"))
            {
                Debug.WriteLine((String)wParam);
            }
            if (uMes == _GetInt("MSG_DISPMSG") || uMes == _GetInt("MSG_DISPLUAMSG"))
            {
                Debug.WriteLine("+++++++++++++++++" + (Int32)(lParam) + "   " +
(String)wParam);
            }
            return null;
        }
    }

    public partial class App : Application
    {
        /// <summary>
        /// Provides easy access to the root frame of the Phone Application.
        /// </summary>
        /// <returns>The root frame of the Phone Application.</returns>
        public static PhoneApplicationFrame RootFrame { get; private set; }

        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {

            //  StarCoreFactoryInit.Init();  for window phone 8.0
     StarCoreFactoryInit.Init(this.GetType().GetTypeInfo().Assembly);  for windows phone
8.1 or windows store

            StarCoreFactory starcore = StarCoreFactory.GetFactory();
            StarServiceClass Service = (StarServiceClass)starcore._InitSimple("test",
"123", 0, 0,null);
            StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

            StarCoreContext.starcore = starcore;
            StarCoreContext.Service = Service;
            StarCoreContext.SrvGroup = SrvGroup;
            StarCoreContext.Path = SrvGroup._GetCorePath();
            MyStarCallBackClass CallBack = new MyStarCallBackClass(starcore);
            Service._CheckPassword(false);

            //--init lua raw interface ---*/
            SrvGroup._InitRaw("lua", Service);
            //--load lua module ---*/
```

```
            String CorePath = SrvGroup._GetCorePath();
            bool Result = SrvGroup._LoadRawModule("lua", "", CorePath + "\\testlua.lua",
false);
            //--attach object to global lua space ---*/
            dynamic obj = Service._ImportRawContext("lua", "", false, "");
            //--call lua function tt, the return contains two integer, which will be
packed into parapkg ---*/
            dynamic ParaPkg = obj.tt("hello ", "world");
            string result = "ret from lua : " + ParaPkg._Number + "   " + ParaPkg[0] + "
" + ParaPkg[0];
            Debug.WriteLine(result);
            //--get global int value g1--------*/
            Debug.WriteLine((string)("lua value g1 :  " + obj.g1));
            //--get global table value c, which is a table with function, it will be
mapped to cle object --------*/
            dynamic c = obj.c;
            //--get int value x obj c--------*/
            result = "c value x :  " + c.x;
            Debug.WriteLine(result);
            //--call c function yy, the return is a table, which will be mapped to cle
object ---*/
            dynamic yy = c.yy(c, "hello ", "world", "!");
            result = "yy value [1] :  " + yy[1];
            Debug.WriteLine(result);
            result = "yy value [Type] :  " + yy["Type"];
            Debug.WriteLine(result);

            SrvGroup._ClearService();
            starcore._ModuleExit();
```

For windows 10, libstarcore and star_csharp45 should be replaced with Libstarcore and Star_csharp

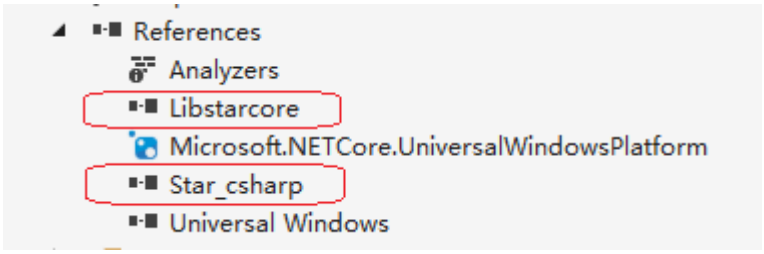
9.3.3 using lua to handle button event.

the following is code segment of lua to handle click event of button.

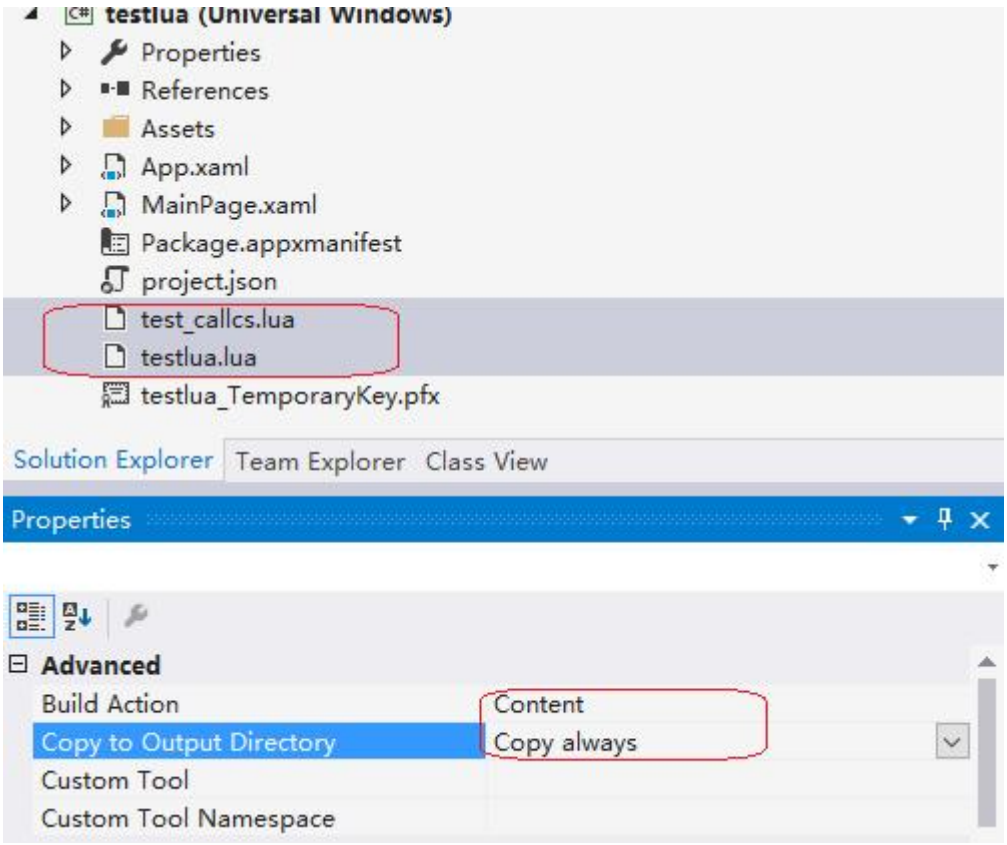
```
SrvGroup = _GetSrvGroup()
Service = SrvGroup:_GetService("","");
function main(context)
    button = context:FindName("mybutton")
    function button:MyClick(sender,e)
        print(sender,e)
  end
    --proxy
    proxy = Service:_NewRawProxy("csharp45",button,"MyClick","System.Windows.RoutedEventHandler",0);
    button.Click:Add(proxy);
end
```

9.3.4 cs calling lua [windows 10]

1. create project and add libs.



2. add lua files to be called



Testlua.lua

```
function tt(a,b)
   print(a,b)
   return 6666,7777
end
g1 = 123
c={x=456}
function c:yy(a,b,z)
   print(self)
   print(a,b,z)
   return {33,Type="mytype"}
end
```

Test_CallCS.lua

```
print(CSClass)


val = CSClass("from lua")
val:callback(1234.4564)
val:callback("sdfsdfsdfsdf")
val:SetLuaObject({"aaa","bbb"});
print("==========end=========")
```

3. c# code

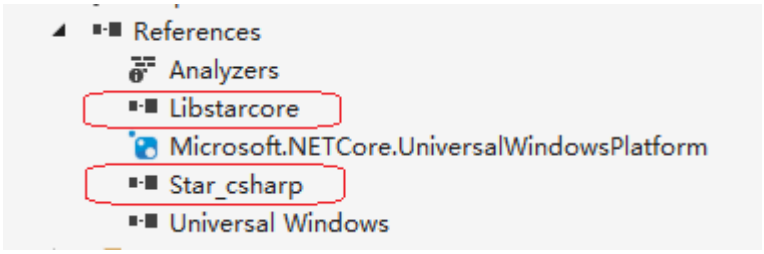
```
public sealed partial class MainPage : Page
    {
        public static MainPage Host;
        public StarSrvGroupClass SrvGroup;

        public MainPage()
        {
            this.InitializeComponent();

            Host = this;

            StarCoreFactoryInit.Init(this);
            StarCoreFactory starcore = StarCoreFactory.GetFactory();
            starcore._RegMsgCallBack_P(new StarMsgCallBackInterface(delegate (int
ServiceGroupID, int uMes, object wParam, object lParam)
            {
                if (uMes == starcore._Getint("MSG_VSDISPMSG") || uMes ==
starcore._Getint("MSG_VSDISPLUAMSG") || uMes == starcore._Getint("MSG_DISPMSG") || uMes ==
starcore._Getint("MSG_DISPLUAMSG"))
                {
                    Debug.WriteLine((string)wParam);
                }
                return null;
            }));
            StarServiceClass Service = (StarServiceClass)starcore._InitSimple("test",
"123", 0, 0, null);
            SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

            Service._CheckPassword(false);

            /*----run lua code----*/
            SrvGroup._InitRaw("lua", Service);
            StarObjectClass lua = Service._ImportRawContext("lua", "", false, "");

            String CorePath = Package.Current.InstalledLocation.Path;
            //--load lua module ---*/
            SrvGroup._LoadRawModule("lua", "", CorePath + "\\testlua.lua", false);
            //--call lua function tt, the return contains two integer, which will be
wrapped into StarObjectClass
            StarObjectClass retobj = (StarObjectClass)lua._Call("tt", "hello ", "world");
            Debug.WriteLine("ret from lua :  " + retobj._Get(1) + "   " + retobj._Get(2));
            //--get global int value g1--------*/
            Debug.WriteLine("lua value g1 :  " + lua._Get("g1"));
            //--get global table value c, which is a table with function, it will be
mapped to cle object --------*/
            StarObjectClass c = lua._GetObject("c");
            //--get int value x from c--------*/
            Debug.WriteLine("c value x :  " + c._Get("x"));
```

```
            //--call c function yy, the return is a table, which will be mapped to cle
object ---*/
            StarObjectClass yy = (StarObjectClass)c._Call("yy", c, "hello ", "world", "!");
            Debug.WriteLine("yy value [1] :  " + yy._Get(1));
            Debug.WriteLine("yy value [Type] :  " + yy._Get("Type"));

            /*-------*/
            lua._Set("CSClass", typeof(CallBackClass));
            Service._DoFile("lua", CorePath + "\\test_callcs.lua", "");  //should not use
null
        }

        public class CallBackClass
        {
            StarObjectClass PythonClass;
            public CallBackClass(String Info)
            {
                Debug.WriteLine(Info);
            }
            public void callback(float val)
            {
                Debug.WriteLine("" + val);
            }
            public void callback(String val)
            {
                Debug.WriteLine("" + val);
            }
            public void SetLuaObject(Object[] rb)
            {
                foreach (Object Item in rb)
                {
                    Debug.WriteLine("" + Item);
                }
            }
        }
    }
```

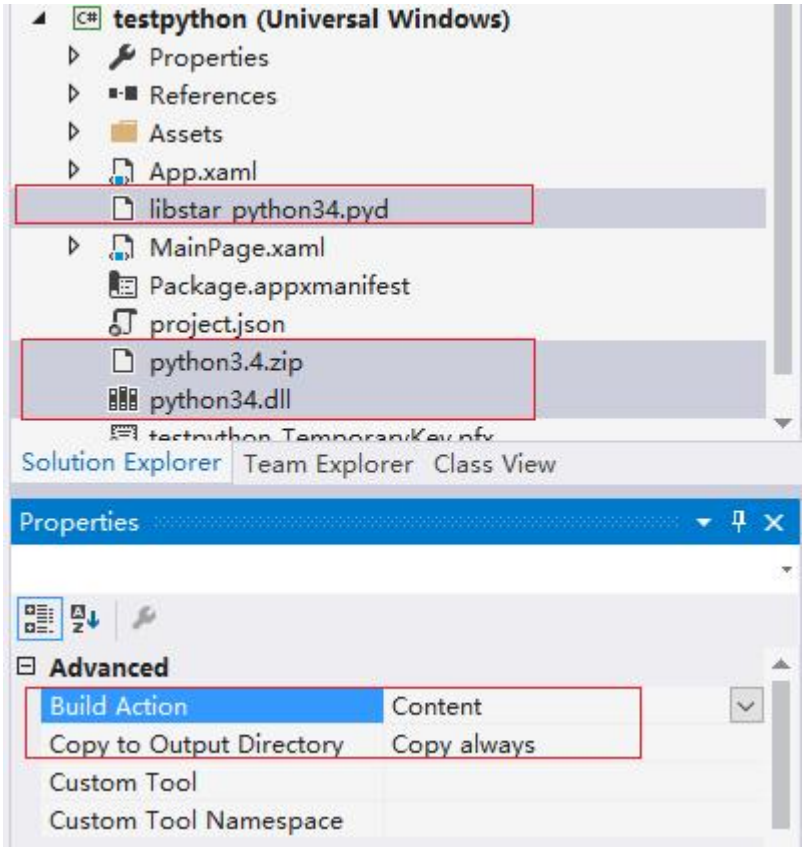
9.3.5 cs calling python [windows 10]

Note : Only python3.X is supported

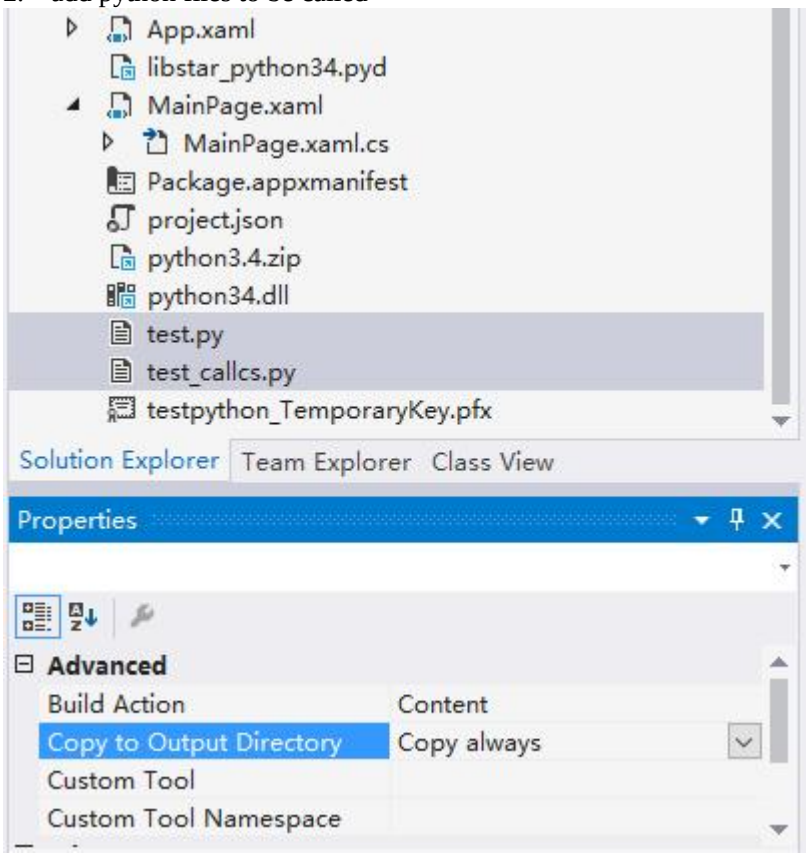
1. create project and add libs.



Add python libraries.



2. add python files to be called



Test.py

```
from __future__ import division
print(division)

import sys
print(sys.path)
import zipfile
import os
print(os)
```

test_callcs.py

```
import json
print(CSClass)

val = CSClass("from python")
val.callback(1234.4564)
val.callback("sdfsdfsdfsdf")
val.SetPythonObject(json);
print("==========end=========")
```

3. c# files

```
public sealed partial class MainPage : Page
    {
        public static MainPage Host;
        public StarSrvGroupClass SrvGroup;

        public MainPage()
        {
            this.InitializeComponent();
            Host = this;

            StarCoreFactoryInit.Init(this);
            StarCoreFactory starcore = StarCoreFactory.GetFactory();
            starcore._RegMsgCallBack_P(new StarMsgCallBackInterface(delegate (int
ServiceGroupID, int uMes, object wParam, object lParam)
            {
                if (uMes == starcore._Getint("MSG_VSDISPMSG") || uMes ==
starcore._Getint("MSG_VSDISPLUAMSG") || uMes == starcore._Getint("MSG_DISPMSG") || uMes ==
starcore._Getint("MSG_DISPLUAMSG"))
                {
                    Debug.WriteLine((string)wParam);
                }
                return null;
            }));
            StarServiceClass Service = (StarServiceClass)starcore._InitSimple("test",
"123", 0, 0, null);
            SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

            bool Result = SrvGroup._InitRaw("python34", Service);
            StarObjectClass python = Service._ImportRawContext("python", "", false, "");

            string CorePath = SrvGroup._GetCorePath();
            Service._DoFile("python", CorePath + "\\test.py", "");
```

```
            python._Set("CSClass", typeof(CallBackClass));
            Service._DoFile("python", CorePath + "\\test_callcs.py", "");  //should not
use null

            python._Call("import", "sys");
            StarObjectClass pythonSys = python._GetObject("sys");
            StarObjectClass pythonPath = (StarObjectClass)pythonSys._Get("path");
            //pythonPath._Call("insert", 0, CorePath + "\\Django-1.10.2-py3.4.egg.zip");
        }
    }

    public class CallBackClass
    {
        StarObjectClass PythonClass;
        public CallBackClass(String Info)
        {
            Debug.WriteLine(Info);
        }
        public void callback(float val)
        {
            Debug.WriteLine("" + val);
        }
        public void callback(String val)
        {
            Debug.WriteLine("" + val);
        }
        public void SetPythonObject(Object rb)
        {
            PythonClass = (StarObjectClass)rb; // Ruby File
            String aa = "";
            StarParaPkgClass data1 = MainPage.Host.SrvGroup._NewParaPkg("b", 789, "c", 456,
"a", 123)._AsDict(true);
            Object d1 = PythonClass._Call("dumps", data1,
MainPage.Host.SrvGroup._NewParaPkg("sort_keys", true)._AsDict(true));
            Debug.WriteLine("" + d1);
            Object d2 = PythonClass._Call("dumps", data1, null);
            Debug.WriteLine("" + d2);
            Object d3 = PythonClass._Call("dumps", data1,
MainPage.Host.SrvGroup._NewParaPkg("sort_keys", true, "indent", 4)._AsDict(true));
            Debug.WriteLine("" + d3);
        }
    }
```

Note:

Because the limitation of windows 10, the following extensions are not supported

_winapi

asyncio

_overlapped

_ctypes

_multiprocessing

_msi

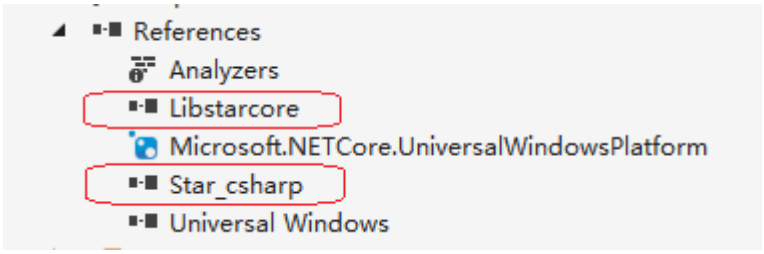
_tkinter

Subprocess

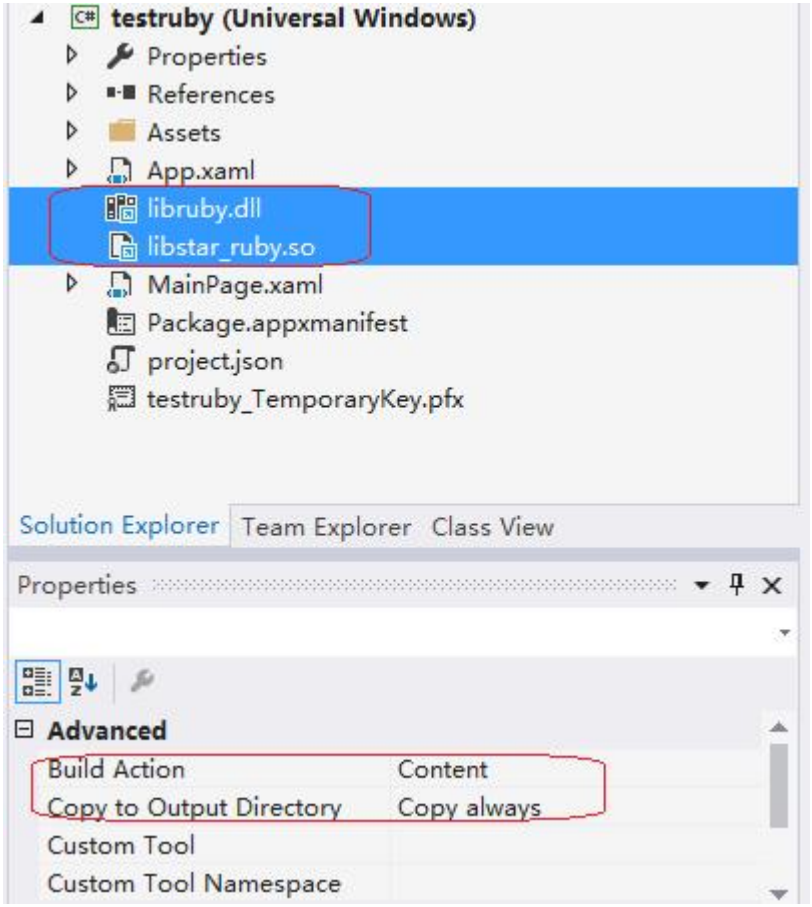
9.3.6 cs calling ruby [windows 10]

Note : ruby2.2.5 is supported, higher version may be supported in future.

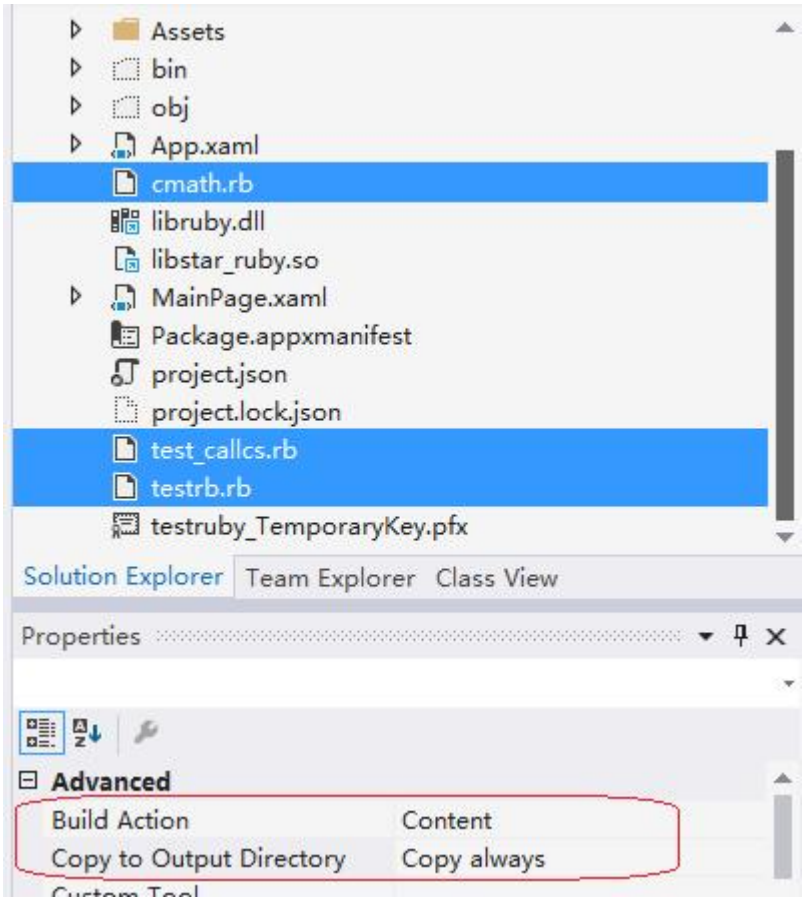
1. create project and add libs.



Add ruby libraries.



2. add ruby files to be called



Testrb.rb

```
def tt(a,b)
   puts(a,b)
   return 666,777
end
$g1 = 123
def yy(a,b,z)
   puts(a,b,z)
   return {'jack'=> 4098, 'sape'=> 4139}
end
class Multiply
   def initialize(x,y)
      @a = x
      @b = y
   end

   def multiply(a,b)
      puts("multiply....",self,a,b)
      return a * b
   end
end
```

test_callcs.rb

```
puts $CSClass


val = $CSClass.new("from ruby")
puts(val)
val.callback(1234.4564)
val.callback("sdfsdfsdfsdf")
val.SetRubyObject(File);
puts("===========end=========")
```

3. c# code

```
public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();

            StarCoreFactoryInit.Init(this);

            StarCoreFactory starcore = StarCoreFactory.GetFactory();
            StarServiceClass Service = (StarServiceClass)starcore._InitSimple("test",
"123", 0, 0, null);
            StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");
            starcore._RegMsgCallBack_P(new StarMsgCallBackInterface(delegate (int
ServiceGroupID, int uMes, object wParam, object lParam) {
                if (uMes == starcore._Getint("MSG_VSDISPMSG") || uMes ==
starcore._Getint("MSG_VSDISPLUAMSG") || uMes == starcore._Getint("MSG_DISPMSG") || uMes ==
starcore._Getint("MSG_DISPLUAMSG"))
                {
                    Debug.WriteLine((string)wParam);
                }
                return null;
            }));

            bool InitRawFlag = SrvGroup._InitRaw("ruby", Service);
            //---set module path
            StarObjectClass ruby = Service._ImportRawContext("ruby", "", false, "");
            StarObjectClass RbPath = (StarObjectClass)ruby._Get("$LOAD_PATH");

            string CorePath = SrvGroup._GetCorePath();
            RbPath._Call("unshift", CorePath);

            ruby._Call("require", "cmath");
            Debug.WriteLine(ruby._Get("CMath"));

            //--load ruby module ---*/
            SrvGroup._LoadRawModule("ruby", "", Package.Current.InstalledLocation.Path +
"\\testrb.rb", false);
            //--attach object to global ruby space ---*/
            StarObjectClass Obj = Service._ImportRawContext("ruby", "", false, "");
            //--call ruby function tt, the return contains two integer, which will be
packed into parapkg ---*/
            StarObjectClass RetObj = (StarObjectClass)Obj._Call("tt", "hello ", "world");
            Debug.WriteLine("ret from ruby : " + RetObj._Get(0) + "  " + RetObj._Get(1));
            //--get global int value g1--------*/
            Debug.WriteLine("ruby value g1 :  " + Obj._Get("g1"));
```

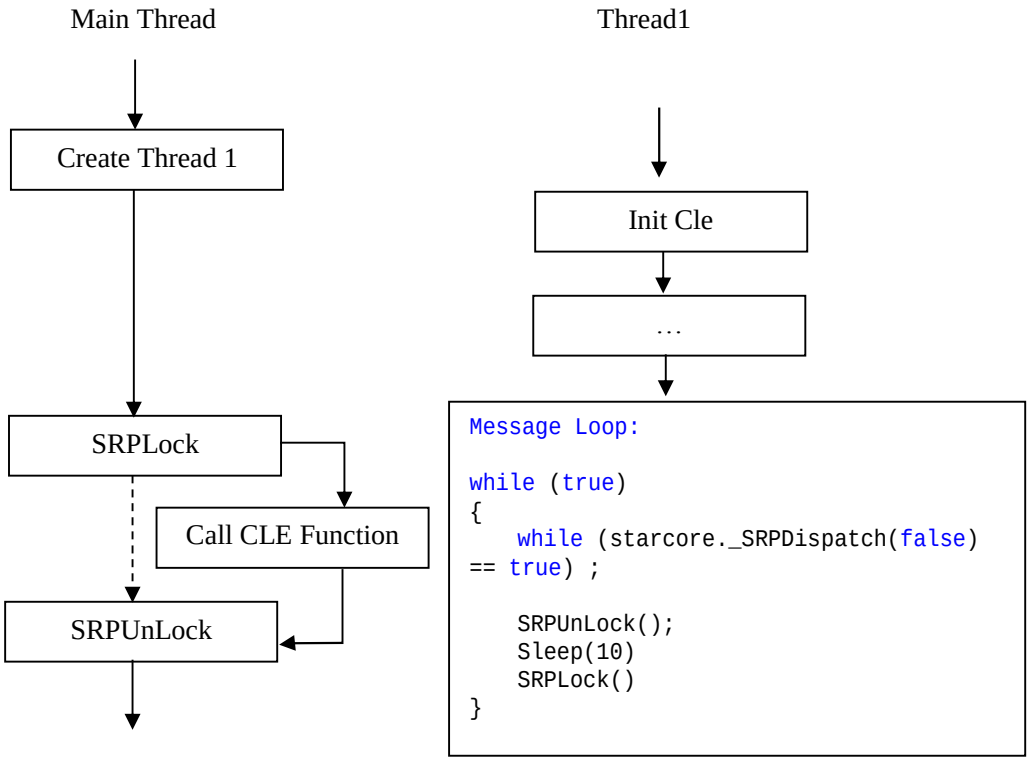
```
            //--get global class Multiply
            StarObjectClass Multiply = Service._ImportRawContext("ruby", "Multiply", true,
"");
            StarObjectClass multiply = Multiply._New("", "", 33, 44);
            //--call instance method multiply
            Debug.WriteLine("instance multiply = " + multiply._Call("multiply", 11, 22));

            ruby._Set("$CSClass", typeof(CallBackClass));
            Service._DoFile("ruby", CorePath + "\\test_callcs.rb", "");  //should not use
null

        }
    }

    public class CallBackClass
    {
        StarObjectClass RBClass;
        public CallBackClass(String Info)
        {
            Debug.WriteLine(Info);
        }
        public void callback(float val)
        {
            Debug.WriteLine("" + val);
        }
        public void callback(String val)
        {
            Debug.WriteLine("" + val);
        }
        public void SetRubyObject(Object rb)
        {
            RBClass = (StarObjectClass)rb; // Ruby File
            StarObjectClass f = RBClass._New("", "",
ApplicationData.Current.LocalFolder.Path + "\\test.txt", "w+");
            f._Call("puts", "I am Jack");
            f._Call("close");
        }
    }
```

9.3.7 notes

When bulild apps for different architecture, such as x86, arm, x64, the reference and native dlls must be replaced with the binary compiled for the corresponding architecture.

10 Multithreading

CLE maintains a global lock. It's status is locked after cle init, it runs in the thread which initialize the cle platform. Some script function, such as network operation, file operation, are time consume, these function may take a little long time to finish. If cle is initialize in the main thread, these operation may block the main thread. Therefore, we recommend to use a separate thread to run cle and scripts.



1. Main thread creates thread1
2. Thread1 starts and init cle, then enter message loop.
3. Main thread continue run, when it need call cle functions or script functions.
   a) _SRPLock
   b) Call cle function or script function
   c) _SRPUnLock

11 Binary data mapping

Because different script languages have different levels of support for binary data, direct use of binary data requires attention to the type of mapping. We recommend that you use BinBuf to process and save binary data. Unless a scripting language raw function is called.

The following table is a binary data mapping table

Call Script Raw Function

	c/c++	lua	Python2.7	Python3.x	ruby	Java	C#
c/c++ binbuf	X	string	string	bytes	string	Byte[]	Byte[]
Lua string	LuaToLString	X	string	String bytes	String	String Byte[]	String Byte[]
Python2.7 string	LuaToLString	string	X	X	string	String Byte[]	String Byte[]
Python 3.x bytes	BinBuf	string	X	X	string	Byte[]	Byte[]

Ruby string	LuaToLString	string	string	String bytes	X	String Byte[]	String Byte[]
Java byte[]	BinBuf	string	string	bytes	string	Byte[]	Byte[]
C# byte[]	BinBuf	string	string	bytes	string	Byte[]	Byte[]

Return value from script function

caller / return	c/c++	lua	Python2.7	Python3.x	ruby	Java	C#
c/c++ binbuf	X	binbuf	Binbuf if FromRaw=false String if FromRaw=true	Binbuf if FromRaw=false bytes if FromRaw=true	Binbuf if FromRaw=false String if FromRaw=true	Binbuf if FromRaw=false Byte[] if FromRaw=true	Binbuf if FromRaw=false Byte[] if FromRaw=true
Lua string	LuaToLString	X	string	String bytes	String	String Byte[]	String Byte[]
Python2.7 string	LuaToLString	string	X	X	string	String Byte[]	String Byte[]
Python 3.x bytes	BinBuf	binbuf	X	X	binbuf	Byte[]	Byte[]
Ruby string	LuaToLString	string	string	String bytes	X	String Byte[]	String Byte[]
Java byte[]	BinBuf	binbuf	string	bytes	string	Byte[]	Byte[]
C# byte[]	BinBuf	binbuf	string	bytes	string	Byte[]	Byte[]

12 Double or Float as Native Function Parameter

When native function parameter type is float or double, VS_FLOAT_F or VS_DOUBLE_F should be used, for example,

```
static void Add(void *Object,VS_FLOAT_F xx,VS_DOUBLE_F yy)
{
   VS_FLOAT x = FROM_VS_FLOAT_F(xx);    /*TO_VS_FLOAT_F convert VS_FLOAT to VS_FLOAT_F*/
   VS_DOUBLE y = FROM_VS_DOUBLE_F(yy);  /*TO_VS_DOUBLE_F convert VS_DOUBLE to VS_DOUBLE_F*/
   SRPInterface ->Print(".....%f,%f",x,y);
   return;
}


SRPInterface ->CreateAtomicFunctionSimpleEx(AtomicClass,(VS_CHAR*)"Add",(VS_CHAR*)"void Add(VS_FLOAT_F
x,VS_DOUBLE_F y);",(void *)Add,&ErrInfo);
```

13 Develop common extension.

examples in directory examples\cle.basic\call.other

13.1 Common extension

13.1.1 Develop common extension using python

```
#import python module
import libstarpy
#Init cle, and create  service group and service
Service = libstarpy._InitSimple("AddFunctionService","123",0,0);
#create object[service item is omitted]
Obj=Service._New("TestClass");
#define object function
def Obj_Add(self,x,y) :
   return x+y;
Obj.Add = Obj_Add;
```

As above, a simple common extension is created. The first step is init cle, then get service group, create service, create service item, create object.

13.1.2 Develop common extension using lua

```
require "libstarcore"
Service = libstarcore._InitSimple("AddFunctionService","123",0,0);
Obj=Service:_New("TestClass");
function Obj:Add(x,y)
  return x+y;
end
```

13.1.3 Develop common extension using java

```
import com.srplab.www.starcore.*;

class MyObjectClass extends StarObjectClass{
        public int Add(StarObjectClass self,int x,int y)
        {
                return x+y;
        }
        public MyObjectClass(StarObjectClass srcobj){
                super(srcobj);
        }
}

public class AddFunction{
        public static void main(String[] args){
```

```
            StarCoreFactory starcore=StarCoreFactory.GetFactory();
            StarServiceClass Service=starcore._InitSimple("AddFunctionService","123",0,0);
            MyObjectClass Obj = new MyObjectClass(Service._New("TestClass"));
    }
}
```

13.1.4 Develop common extension using C++

```
#include "vsopenapi.h"

static VS_INT32 Add(void *Object,VS_INT32 x,VS_INT32 y)
{
    return x + y;
}

VS_BOOL StarCoreService_Init(class ClassOfStarCore *starcore)
{
    void *AtomicClass,*Add_AtomicFunction;
    class ClassOfBasicSRPInterface *BasicSRPInterface;
    class ClassOfSRPInterface *SRPInterface;

    //--init star core
    BasicSRPInterface = starcore ->GetBasicInterface();
    BasicSRPInterface ->CreateService("","AddFunctionService",NULL,"123",0,0,0,0,0,0);
    SRPInterface = BasicSRPInterface ->GetSRPInterface("AddFunctionService","root","123");

    //---Create Atomic Class, for define function, no attribute
    AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClass",NULL,NULL,NULL);
    Add_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"Add","VS_INT32 Add(VS_INT32
x,VS_INT32 y);",NULL,NULL,VS_FALSE,VS_FALSE);
  //---Set Function Address
    SRPInterface -> SetAtomicFunction(Add_AtomicFunction,(void *)Add);
    SRPInterface -> Release();
    return VS_TRUE;
}

void StarCoreService_Term(class ClassOfStarCore *starcore)
{
    return;
}
```

13.1.5 Develop common extension using C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace AddFunction_Csharp
{
class MyObjectClass : StarObjectClass{
        public int Add(StarObjectClass self,int x,int y)
        {
                return x+y;
        }
        public MyObjectClass(StarObjectClass srcobj):base(srcobj){
        }
}
   class Program
   {
      static void Main(string[] args)
      {
         StarCoreFactory starcore =StarCoreFactory.GetFactory();
         StarServiceClass Service = starcore._InitSimple("AddFunctionService", "123", 0, 0);
         MyObjectClass Obj = new MyObjectClass(Service._New("TestClass"));
      }
   }
}
```

13.2 Call common extension using C/C++

Call common extension, application can use interface functions provided by CLE.

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
        VS_CORESIMPLECONTEXT Context;
        class ClassOfSRPInterface *SRPInterface;
        void *Class,*Object;

/*--------call as service  */
        SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,"files/AddFunction.lua?script=lua",NULL);
//  SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,"files/AddFunction.py?script=python",NULL);
//      SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,"files/AddFunction.class?script=java",NULL);
//      SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,"files/AddFunction.dll",NULL);
        //Get class : TestClass
```

```
	Class = SRPInterface ->GetObjectEx(NULL,"TestClass");
	//Create instance
	Object = SRPInterface ->MallocObjectL( SRPInterface->GetIDEx(Class),0,NULL);
	//Call object method : Add
	printf("Call Function Ret = %d\n",SRPInterface ->ScriptCall(Object,NULL,"Add","(ii)i",12,34));

	SRPInterface -> Release();
	VSCore_TermSimple(&Context);
	return 0;
}
```

Compiled on linux:

g++ -Wall -Wno-format -g -DDEBUG -DENV_LINUX -I/usr/include/starcore -o c_call.o -c c_call.cpp

g++ -o c_call_linux -g c_call.o -ldl -lpthread -lrt /usr/lib/libstarlib.a /usr/lib/libuuid.a

13.3 Call common extension using lua

```
require "libstarcore"
--Service=libstarcore._InitSimple("test","123",0,0,"files/AddFunction.lua?script=lua");
--Service=libstarcore._InitSimple("test","123",0,0,"files/AddFunction.py?script=python");
Service=libstarcore._InitSimple("test","123",0,0,"files/AddFunction.class?script=java");
--Service=libstarcore._InitSimple("test","123",0,0,"files/AddFunction.dll");
a = Service.TestClass:_New();
print(a:Add(12,34))
Service._ServiceGroup:_ClearService()
libstarcore._ModuleExit()
```

13.4 Call common extension using python

```
import libstarpy
#Service=libstarpy._InitSimple("test","123",0,0,"files/AddFunction.lua?script=lua");
#Service=libstarpy._InitSimple("test","123",0,0,"files/AddFunction.py?script=python");
#Service=libstarpy._InitSimple("test","123",0,0,"files/AddFunction.class?script=java");
Service=libstarpy._InitSimple("test","123",0,0,"files/AddFunction.dll");
a = Service.TestClass._New();
print(a.Add(12,34))
Service._ServiceGroup._ClearService()
libstarpy._ModuleExit()
```

13.5 Call common extension using java

```
import com.srplab.www.starcore.*;
```

```
public class java_call{
        public static void main(String[] args){
                StarCoreFactory starcore=StarCoreFactory.GetFactory();
//              StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/AddFunction.lua?script=lua");
//              StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/AddFunction.py?script=python");
//              StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/AddFunction.class?script=java");
//              StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/AddFunction.dll");
                StarServiceClass Service=starcore._InitSimple("test","123",0,0,"../files/AddFunction_Csharp.exe?script=csharp");
                StarObjectClass a = Service._GetObject("TestClass")._New();
        System.out.println(a._Call("Add",12,34));
        starcore._ModuleExit();
    }
}
```

13.6 Call common extension using C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace csharp_call
{
    class Program
    {
        static void Main(string[] args)
        {
                StarCoreFactory starcore=StarCoreFactory.GetFactory();
//              StarServiceClass Service=starcore._InitSimple("test","123",0,0,"../files/AddFunction.lua?script=lua");
//       StarServiceClass Service=starcore._InitSimple("test","123",0,0,"../files/AddFunction.py?script=python");
//       StarServiceClass Service=starcore._InitSimple("test","123",0,0,"../files/AddFunction.class?script=java");
//       StarServiceClass Service = starcore._InitSimple("test", "123", 0, 0, "../files/AddFunction.dll");
        StarServiceClass Service=starcore._InitSimple("test","123",0,0,"../files/AddFunction_Csharp.exe?script=csharp");
                StarObjectClass a = Service._GetObject("TestClass")._New();
        Console.WriteLine(a._Call("Add",12,34));
        starcore._ModuleExit();
        }
    }
}
```

13.7 passing complex data structures between languages

Examples in directory examples\ cle.advanced\call.other.

Application can pass data structues with object as function parameter. Object may contain struct attributes.Attibute types supported by object is listed bellow.

Types supported by struct :

```
TYPE_BOOL :
TYPE_INT8 :
TYPE_UINT8 :
TYPE_INT16 :
TYPE_UINT16 :
TYPE_INT32 :
TYPE_UINT32 :
TYPE_FLOAT :
TYPE_LONG :
TYPE_ULONG :
TYPE_CHAR :
TYPE_COLOR :
TYPE_RECT :
TYPE_FONT :
TYPE_TIME :
TYPE_UUID :
```

Types supported by object :

```
TYPE_BOOL :
TYPE_INT8 :
TYPE_UINT8 :
TYPE_INT16 :
TYPE_UINT16 :
TYPE_INT32 :
TYPE_UINT32 :
TYPE_FLOAT :
TYPE_LONG :
TYPE_ULONG :
TYPE_LONGHEX :
TYPE_ULONGHEX :
TYPE_VSTRING :
TYPE_PTR :
TYPE_STRUCT :
TYPE_CHAR :
TYPE_COLOR :
TYPE_RECT :
```

```
        TYPE_FONT :
        TYPE_TIME :
        TYPE_UUID :
        TYPE_STATICID :
```

Application can also use Parapkg to pass structured data.

For better mapping, application should define object attributes as follows :

Script code(python)：

```
Service._CreateAtomicStructSimple("ParaStruct","VS_INT32 Para1;VS_FLOAT Para2;","");
Service._CreateAtomicObjectSimple("ServiceItem","ParaClass","VS_INT32 Para1;VS_UUID Para2;VS_FLOAT Para3;struct
ParaStruct Para4;VS_VSTRING Para5;","");
```

c/c++ code

```
    SRPInterface ->CreateAtomicStructSimple("ParaStruct","VS_INT32 Para1;VS_FLOAT Para2;",NULL,NULL);
    SRPInterface ->CreateAtomicObjectSimple("TestItem","ParaClass","VS_INT32 Para1;VS_UUID Para2;VS_FLOAT
Para3;struct ParaStruct Para4;VS_VSTRING Para5;",NULL,NULL);
```

The corresponding to the c / c + + structure is shown below:

```
//--define struct
struct ParaStruct{
    VS_INT32 Para1;
    VS_FLOAT Para2;
};

struct ParaClass{
    VS_INT32 Para1;
    VS_UUID Para2;
    VS_FLOAT Para3;
    struct ParaStruct Para4;
    VS_VSTRING Para5;
};
```

13.7.1 Extension module to be called

13.7.1.1Develop common extension using python

```
import libstarpy
Service = libstarpy._InitSimple("TestService","123",0,0);
#--define struct
Service._CreateAtomicStructSimple("ParaStruct","VS_INT32 Para1;VS_FLOAT Para2;","");
Service._CreateAtomicObjectSimple("ServiceItem","ParaClass","VS_INT32 Para1;VS_UUID Para2;VS_FLOAT Para3;struct
ParaStruct Para4;VS_VSTRING Para5;","");
```

```
Obj=Service._New("TestClass");
def Obj_PrintObj(self,ParaObj) :
  print("ParaObj.Para1=",ParaObj.Para1);
  print("ParaObj.Para2=",ParaObj.Para2);
  print("ParaObj.Para3=",ParaObj.Para3);
  print("ParaObj.Para4.Para1=",ParaObj.Para4.Para1);
  print("ParaObj.Para4.Para2=",ParaObj.Para4.Para2);
  print("ParaObj.Para5=",ParaObj.Para5);
Obj.PrintObj = Obj_PrintObj;
```

13.7.1.2Develop common extension using lua

```
require "libstarcore"
Service = libstarcore._InitSimple("TestService","123",0,0);
--define struct
Service:_CreateAtomicStructSimple("ParaStruct","VS_INT32 Para1;VS_FLOAT Para2;","");
Service:_CreateAtomicObjectSimple("ServiceItem","ParaClass","VS_INT32 Para1;VS_UUID Para2;VS_FLOAT Para3;struct
ParaStruct Para4;VS_VSTRING Para5;","");

Obj=Service:_New("TestClass");
function Obj:PrintObj(ParaObj)
  print("ParaObj.Para1=",ParaObj.Para1);
  print("ParaObj.Para2=",ParaObj.Para2);
  print("ParaObj.Para3=",ParaObj.Para3);
  print("ParaObj.Para4.Para1=",ParaObj.Para4.Para1);
  print("ParaObj.Para4.Para2=",ParaObj.Para4.Para2);
  print("ParaObj.Para5=",ParaObj.Para5);
end
```

13.7.1.3Develop common extension using java

```
import com.srplab.www.starcore.*;

class MyObjectClass extends StarObjectClass{
    public void PrintObj(StarObjectClass self,StarObjectClass ParaObj)
    {
        System.out.println("ParaObj.Para1="+ParaObj._GetInt("Para1"));
        System.out.println("ParaObj.Para2="+ParaObj._GetStr("Para2"));
        System.out.println("ParaObj.Para3="+ParaObj._GetDouble("Para3"));
        System.out.println("ParaObj.Para4.Para1="+((StarStructClass)ParaObj._Get("Para4"))._Get("Para1"));
        System.out.println("ParaObj.Para4.Para2="+((StarStructClass)ParaObj._Get("Para4"))._Get("Para2"));
        System.out.println("ParaObj.Para5="+ParaObj._GetStr("Para5"));
```

```
    }
        public MyObjectClass(StarObjectClass srcobj){
                super(srcobj);
        }
}

public class Test{
        public static void main(String[] args){
                StarCoreFactory starcore= StarCoreFactory.GetFactory();
                StarServiceClass Service=starcore._InitSimple("TestService","123",0,0);

       Service._CreateAtomicStructSimple("ParaStruct","VS_INT32 Para1;VS_FLOAT Para2;","");
       Service._CreateAtomicObjectSimple("TestItem","ParaClass","VS_INT32 Para1;VS_UUID Para2;VS_FLOAT Para3;struct
ParaStruct Para4;VS_VSTRING Para5;","");

                MyObjectClass Obj = new MyObjectClass(Service._New("TestClass"));
    }
}
```

13.7.1.4Develop common extension using C++

```
#include "vsopenapi.h"

static class ClassOfSRPInterface *SRPInterface;

//--define struct
struct ParaStruct{
        VS_INT32 Para1;
        VS_FLOAT Para2;
};

struct ParaClass{
        VS_INT32 Para1;
        VS_UUID Para2;
        VS_FLOAT Para3;
        struct ParaStruct Para4;
        VS_VSTRING Para5;
};

static void PrintObj(void *Object,struct ParaClass *ParaObj)
{
        printf("ParaObj.Para1=%d\n",ParaObj->Para1);
        printf("ParaObj.Para2=%s\n",SRPInterface->UuidToString(&ParaObj->Para2));
        printf("ParaObj.Para3=%f\n",ParaObj->Para3);
```

```
        printf("ParaObj.Para4.Para1=%d\n",ParaObj->Para4.Para1);
        printf("ParaObj.Para4.Para2=%f\n",ParaObj->Para4.Para2);
    printf("ParaObj.Para5=%s\n",ParaObj->Para5.Buf);
}

VS_BOOL StarCoreService_Init(class ClassOfStarCore *starcore)
{
        void *AtomicClass,*PrintObjFunction;
        class ClassOfBasicSRPInterface *BasicSRPInterface;

        //--init star core
        BasicSRPInterface = starcore ->GetBasicInterface();
        BasicSRPInterface ->CreateService("","TestService",NULL,"123",0,0,0,0,0,0);
        SRPInterface = BasicSRPInterface ->GetSRPInterface("TestService","root","123");

        //---Create Atomic Class, for define function, no attribute
        SRPInterface ->CreateAtomicStructSimple("ParaStruct","VS_INT32 Para1;VS_FLOAT Para2;",NULL,NULL);
        SRPInterface ->CreateAtomicObjectSimple("TestItem","ParaClass","VS_INT32 Para1;VS_UUID Para2;VS_FLOAT
Para3;struct ParaStruct Para4;VS_VSTRING Para5;",NULL,NULL);



        AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClass",NULL,NULL,NULL);
        PrintObjFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"PrintObj","void PrintObj(VS_OBJPTR
ParaObj);",NULL,NULL,VS_FALSE,VS_FALSE);
    //---Set Function Address
        SRPInterface -> SetAtomicFunction(PrintObjFunction,(void *)PrintObj);
        return VS_TRUE;
}

void StarCoreService_Term(class ClassOfStarCore *starcore)
{
        SRPInterface -> Release();
        return;
}
```

13.7.1.5Develop common extension using C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace Test_Csharp
```

```
{
class MyObjectClass : StarObjectClass{
   public void PrintObj(StarObjectClass self,StarObjectClass ParaObj)
   {
      Console.WriteLine("ParaObj.Para1="+ParaObj._GetInt("Para1"));
      Console.WriteLine("ParaObj.Para2="+ParaObj._GetStr("Para2"));
      Console.WriteLine("ParaObj.Para3="+ParaObj._GetDouble("Para3"));
      Console.WriteLine("ParaObj.Para4.Para1="+((StarStructClass)ParaObj._Get("Para4"))._Get("Para1"));
      Console.WriteLine("ParaObj.Para4.Para2="+((StarStructClass)ParaObj._Get("Para4"))._Get("Para2"));
      Console.WriteLine("ParaObj.Para5=" + ParaObj._GetStr("Para5"));
   }
       public MyObjectClass(StarObjectClass srcobj):base(srcobj){
       }
}
   class Program
   {
      static void Main(string[] args)
      {
         StarCoreFactory starcore= StarCoreFactory.GetFactory();
         StarServiceClass Service = starcore._InitSimple("TestService", "123", 0, 0);

         Service._CreateAtomicStructSimple("ParaStruct", "VS_INT32 Para1;VS_FLOAT Para2;", "");
         Service._CreateAtomicObjectSimple("TestItem", "ParaClass", "VS_INT32 Para1;VS_UUID Para2;VS_FLOAT
Para3;struct ParaStruct Para4;VS_VSTRING Para5;", "");

         MyObjectClass Obj = new MyObjectClass(Service._New("TestClass"));
      }
   }
}
```

13.7.2 Call common extension using C/C++

```
#include "vsopenapi.h"

struct ParaStruct{
       VS_INT32 Para1;
       VS_FLOAT Para2;
};

struct ParaClass{
       VS_INT32 Para1;
       VS_UUID Para2;
       VS_FLOAT Para3;
       struct ParaStruct Para4;
```

```
        VS_VSTRING Para5;
};

static VS_ULONG MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_ULONG wParam, VS_ULONG
lParam, VS_BOOL &IsProcessed, VS_ULONG Para )
{
   switch( uMsg ){
   case MSG_DISPMSG :
        case MSG_DISPLUAMSG :
                printf("%s\n",(VS_CHAR *)wParam);
       break;
   }
   return 0;
}

int main(int argc, char* argv[])
{
        VS_CORESIMPLECONTEXT Context;
        class ClassOfSRPInterface *SRPInterface;
        void *Class,*Object,*m_ParaClass;
        struct ParaClass *ParaObj;

        /*--------call as service  */
        //SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,"files/Test.lua?script=lua",NULL);
        SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,MsgCallBack,0,"files/Test.py?script=python",NULL);
        //SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,"files/Test.class?script=java",NULL);
        //SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,"files/Test.dll",NULL);
        //SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,"files/Test_Csharp.exe?script=csharp",NULL);

        Class = SRPInterface ->GetObjectEx(NULL,"TestClass");
        Object = SRPInterface ->MallocObjectL( SRPInterface->GetIDEx(Class),0,NULL);

        m_ParaClass = SRPInterface ->GetObjectEx(NULL,"ParaClass");
        ParaObj = (struct ParaClass *)SRPInterface ->MallocObjectL( SRPInterface->GetIDEx(m_ParaClass),0,NULL);
        ParaObj->Para1 = 124;
        ParaObj->Para2 = (*SRPInterface->GetIDEx(Object));
        ParaObj->Para3 = 23456.78;
        ParaObj->Para4.Para1 = 999;
        ParaObj->Para4.Para2 = 4444.55;
        ParaObj->Para5 = (VS_VSTRING)"From caller";
        SRPInterface ->ScriptCall(Object,NULL,"PrintObj","(o)",ParaObj);

        SRPInterface -> Release();
        VSCore_TermSimple(&Context);
        return 0;
```

```
}
```

13.7.3 Call common extension using lua

```
require "libstarcore"
--Service=libstarcore._InitSimple("test","123",0,0,"files/Test.lua?script=lua");
--Service=libstarcore._InitSimple("test","123",0,0,"files/Test.py?script=python");
--Service=libstarcore._InitSimple("test","123",0,0,"files/Test.class?script=java");
--Service=libstarcore._InitSimple("test","123",0,0,"files/Test.dll");
Service=libstarcore._InitSimple("test","123",0,0,"files/Test_Csharp.exe?script=csharp");

a = Service.TestClass:_New();
ParaObj = Service.ParaClass:_New();
ParaObj.Para1 = 124;
ParaObj.Para2 = a._ID;
ParaObj.Para3 = 23456.78;
ParaObj.Para4 = {999,4444.55};
ParaObj.Para5 = "From caller";
a:PrintObj(ParaObj)
Service._ServiceGroup:_ClearService()
libstarcore._ModuleExit()
```

13.7.4 Call common extension using python

```
import libstarpy
#Service=libstarpy._InitSimple("test","123",0,0,"files/Test.lua?script=lua");
#Service=libstarpy._InitSimple("test","123",0,0,"files/Test.py?script=python");
#Service=libstarpy._InitSimple("test","123",0,0,"files/Test.class?script=java");
Service=libstarpy._InitSimple("test","123",0,0,"files/Test.dll");
#Service=libstarpy._InitSimple("test","123",0,0,"files/Test_Csharp.exe?script=csharp");

a = Service.TestClass._New();
ParaObj = Service.ParaClass._New();
ParaObj.Para1 = 124;
ParaObj.Para2 = a._ID;
ParaObj.Para3 = 23456.78;
ParaObj.Para4 = (999,4444.55);
ParaObj.Para5 = "From caller";
a.PrintObj(ParaObj)

Service._ServiceGroup._ClearService()
libstarpy._ModuleExit()
```

13.7.5 Call common extension using java

```
import com.srplab.www.starcore.*;

public class java_call{
        public static void main(String[] args){
                StarCoreFactory starcore= StarCoreFactory.GetFactory();
//              StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/Test.lua?script=lua");
//              StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/Test.py?script=python");
                StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/Test.class?script=java");
//              StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/Test.dll");

                StarObjectClass a = Service._GetObject("TestClass")._New();
                StarObjectClass ParaObj = Service._GetObject("ParaClass")._New();
                ParaObj._Set("Para1",124);
                ParaObj._Set("Para2",a._Get("_ID"));
                ParaObj._Set("Para3",23456.78);
                ParaObj._Set("Para4",new Object[]{999,4444.55});
                ParaObj._Set("Para5","From caller");
                a._Call("PrintObj",ParaObj);

        starcore._ModuleExit();
    }
}
```

13.7.6 Call common extension using C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace csharp_call
{
    class MyStarCallBackClass : StarCallBackClass{
            public Object[] CallBack( int ServiceGroupID, int uMes, Object wParam, Object lParam )
            {
                if( uMes == _Getint("MSG_DISPMSG") || uMes == _Getint("MSG_DISPLUAMSG") ){
                    Console.WriteLine((String)wParam);
                    }
                return null;
            }
```

```
        public MyStarCallBackClass(StarCoreFactory starcore) : base(starcore){starcore._RegMsgCallBack(this,"CallBack");}
  }
  class Program
  {
    static void Main(string[] args)
    {
      StarCoreFactory starcore = StarCoreFactory.GetFactory();
      //StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/Test.lua?script=lua");
      //StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/Test.py?script=python");
      //StarServiceClass Service = starcore._InitSimple("test", "123", 0, 0, "files/Test.class?script=java");
      //StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/Test.dll");
      StarServiceClass Service = starcore._InitSimple("test", "123", 0, 0, "files/Test_Csharp.exe?script=csharp");
      MyStarCallBackClass CallBack = new MyStarCallBackClass(starcore);

      StarObjectClass a = Service._GetObject("TestClass")._New();
      StarObjectClass ParaObj = Service._GetObject("ParaClass")._New();
      ParaObj._Set("Para1", 124);
      ParaObj._Set("Para2", a._Get("_ID"));
      ParaObj._Set("Para3", 23456.78);
      ParaObj._Set("Para4", new Object[] { 999, 4444.55 });
      ParaObj._Set("Para5", "From caller");
      a._Call("PrintObj", ParaObj);

      starcore._ModuleExit();
    }
  }
}
```

13.8 A more complicated example

13.8.1 java swing window(Callback function)

The example is in directory examples\cle.basic\call.javawin

13.8.1.1 Common extension developed by java to create a window using swing

```
import java.awt.*;
import javax.swing.*;
import com.srplab.www.starcore.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```

```
class FrameListener extends WindowAdapter
{
        private StarObjectClass WhichObj;
    public void windowClosing(WindowEvent e)
    {
       WhichObj._Call("OnClose");  //--call extrn script
    }
    public FrameListener(StarObjectClass obj){
        WhichObj = obj;
    }
}

class MyObjectClass extends StarObjectClass{
        public void CreateWindow(StarObjectClass self,int Width,int Height,String Caption)
        {
                JFrame ab = new JFrame(Caption);
                ab.setSize(Width,Height);
                ab.setVisible(true);
                ab.addWindowListener(new FrameListener(self));
                self._Set("WinObj",ab);
        }
        public MyObjectClass(StarObjectClass srcobj){
                super(srcobj);
        }
}

public class SimpleWin{
        public static void main(String[] args){
                StarCoreFactory starcore=StarCoreFactory.GetFactory();
                StarServiceClass Service=starcore._InitSimple("SimpleWinService","123",0,0);

                Object[] AtomicClassArray = Service._CreateAtomicObjectSimple("TestItem","JavaWinClass","","");
                //--Define function to enable C++ to set the address for callback from java, for script language, it is not needed.
                 Service._CreateAtomicFunctionSimple(Service._Toint(AtomicClassArray[0]),"OnClose","void
OnClose();","",false,false);
                MyObjectClass ImgObj = new MyObjectClass(Service._AtomicToObject(Service._Toint(AtomicClassArray[0])));    }
}
```

On above example, a class named JavaWinClass is created, which contains a function to create window "CreateWindow" and object's callback function "OnClose" which will be called when the window is closed.

13.8.1.2Call using python

```
import libstarpy
Service=libstarpy._InitSimple("test", "123",0,0,"files/SimpleWin.class?script=java");

a = Service.JavaWinClass._New("python object")
a.CreateWindow(640,480,"call from python");
def a_OnClose(self) :
   global ExitFlag
   ExitFlag = True;
a.OnClose = a_OnClose;

ExitFlag = False

def ExitProc() :
   return ExitFlag

libstarpy._MsgLoop( ExitProc )

Service._ServiceGroup._ClearService()
libstarpy._ModuleExit()
```

13.8.1.3Call using C++

```
#include "vsopenapi.h"

static VS_BOOL ExitFlag;

void OnClose(void *Object)
{
   ExitFlag = VS_TRUE;
}

int main(int argc, char* argv[])
{
        VS_CORESIMPLECONTEXT Context;
        class ClassOfSRPInterface *SRPInterface;
        void *Class,*Object;
        VS_UUID FunctionID;

        SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,"files\\SimpleWin.class?script=java",NULL);

        Class = SRPInterface ->GetObjectEx(NULL,"JavaWinClass");
        //get ID of callback function
        SRPInterface ->GetFunctionID(Class,"OnClose",&FunctionID);
```

```
        //Set callback function address
        SRPInterface ->SetFunction(&FunctionID,OnClose);

        Object = SRPInterface ->MallocObjectL( SRPInterface->GetIDEx(Class),0,NULL);
        SRPInterface ->ScriptCall(Object,NULL,"CreateWindow","(iis)",640,480,"window from c++");
        //--MsgLoop
        while(ExitFlag == VS_FALSE)
                Context.VSControlInterface->SRPDispatch(VS_TRUE);

        SRPInterface -> Release();
        VSCore_TermSimple(&Context);
        return 0;
}
```

13.8.1.4Call using c#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace csharp_call
{
class MyStarCallBackClass : StarCallBackClass{
        public MyStarCallBackClass(StarCoreFactory starcore):base(starcore){}
        public Boolean ExitProc()
   {
     return Program.ExitFlag;
   }
}

class MyObjectClass : StarObjectClass{
  public void OnClose( StarObjectClass self )
  {
     Program.ExitFlag = true;
  }
        public MyObjectClass(StarObjectClass srcobj):base(srcobj){
        }
}

   class Program
   {
      public static Boolean ExitFlag = false;
```

```
    static void Main(string[] args)
    {
            StarCoreFactory starcore= StarCoreFactory.GetFactory();
            MyStarCallBackClass CallBack = new MyStarCallBackClass(starcore);
            StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/SimpleWin.class?script=java");
            StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

    StarObjectClass a = new MyObjectClass(Service._GetObject("JavaWinClass")._New("csharp object"));
    a._Call("CreateWindow",640,480,"call from csharp");

    starcore._MsgLoop(CallBack,"ExitProc");

        Console.WriteLine("Exit...");
        SrvGroup._ClearService();
    starcore._ModuleExit();
    }
  }
}
```

13.8.2 call jsoup

The example is in directory examples\cle.basic\call.jsoup.java

13.8.2.1 Common extension developed by java to create an interface object to jsoup

```
import com.srplab.www.starcore.*;
import org.jsoup.*;
import org.jsoup.nodes.*;

class MyObjectClass extends StarObjectClass{
  public void Parse( StarObjectClass self, String HtmlStr){
       Document doc;
    doc = Jsoup.parse(HtmlStr);
    if( doc != null )
      self._Set("Document",doc);
  }
  public Boolean ParseUrl( StarObjectClass self, String In_Url){
       Document doc;
       try{
      doc = Jsoup.connect(In_Url).get();
      if( doc != null ){
        self._Set("Document",doc);
        return true;
```

```
        }
        return false;
    }
    catch(Exception e){
        return false;
    }
  }
  public String GetTitle(StarObjectClass self){
        Document doc;

        doc = (Document)self._Get("Document");
    if( doc == null )
        return null;
    return doc.title();
  }
  public MyObjectClass(StarObjectClass srcobj){
        super(srcobj);
  }
}
public class jsoup_wrap{
       public static void main(String[] args){
              StarCoreFactory starcore= StarCoreFactory.GetFactory();
              StarServiceClass Service=starcore._InitSimple("jsoup_cle_service","123",0,0);
      MyObjectClass ObjClass = new MyObjectClass( Service._New("JSoupClass") );  // JSoupClass is name of the interface
object.
      starcore._ModuleExit();
   }
}
```

Packing the above java program and jsoup into one jar file.

13.8.2.2Call using python

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0,"files/jsoup_wrap.jar?script=java");
a = Service.JSoupClass._New();
a.Parse("<html><head><title> test title </title></head>"+"<body><p> this is test of jsoup</p></body></html>");
print( a.GetTitle() );

b = Service.JSoupClass._New();
if( b.ParseUrl("http://127.0.0.1/index.htm") == True ) :
  print( b.GetTitle() );
```

```
Service._ServiceGroup._ClearService()
libstarpy._ModuleExit()
```

13.8.2.3Call using C/C++

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
	VS_CORESIMPLECONTEXT Context;
	class ClassOfSRPInterface *SRPInterface;
	void *Class,*Object,*Object1;

/*--------call as service  */
	SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,"files/jsoup_wrap.jar?script=java",NULL);

	Class = SRPInterface ->GetObjectEx(NULL,"JSoupClass");
	Object = SRPInterface ->MallocObjectL( SRPInterface->GetIDEx(Class),0,NULL);
	SRPInterface ->ScriptCall(Object,NULL,"Parse","(s)","<html><head><title> test title </title></head><body><p> this is test
of jsoup</p></body></html>");
	printf("%s\n",SRPInterface ->ScriptCall(Object,NULL,"GetTitle","()s"));

	Object1 = SRPInterface ->MallocObjectL( SRPInterface->GetIDEx(Class),0,NULL);
	if( (VS_BOOL)SRPInterface ->ScriptCall(Object1,NULL,"ParseUrl","(s)z","http://127.0.0.1/index.htm") == VS_TRUE )
		printf("%s\n",SRPInterface ->ScriptCall(Object1,NULL,"GetTitle","()s"));

	SRPInterface -> Release();
	VSCore_TermSimple(&Context);
	return 0;
}
```

13.8.2.4Call using c#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace csharp_call
{
   class Program
   {
```

```
    static void Main(string[] args)
    {
            StarCoreFactory starcore= StarCoreFactory.GetFactory();
            StarServiceClass Service=starcore._InitSimple("test","123",0,0,"files/jsoup_wrap.jar?script=java");
    StarObjectClass a = Service._GetObject("JSoupClass")._New();
    a._Call("Parse","<html><head><title> test title </title></head><body><p> this is test of jsoup</p></body></html>");
    Console.WriteLine(a._Call("GetTitle"));

    StarObjectClass b = Service._GetObject("JSoupClass")._New();
    if( (Boolean)b._Call("ParseUrl","http://127.0.0.1/index.htm") == true )
    {
      Console.WriteLine( b._Call("GetTitle"));
    }
    starcore._ModuleExit();
    }
  }
}
```

13.8.3 c# form calls java

Examples in directory examples\ cle.basic\others\csharp_call\csharp_form_call_java

java code:

```
import com.srplab.www.starcore.*;

class MyObjectClass extends StarObjectClass{
  public String getString(StarObjectClass self) {
    return "Hello, test!";
  }
     public MyObjectClass(StarObjectClass srcobj){
            super(srcobj);
     }
}

public class Test{
     public static void main(String[] args){
            StarCoreFactory starcore= StarCoreFactory.GetFactory();
            StarServiceClass Service=starcore._InitSimple("Test","123",0,0);
            MyObjectClass Obj = new MyObjectClass(Service._New("TestClass"));
  }
}
```

javac Test.java

c# form code:

```
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Star_csharp;

namespace CallTest
{
   public partial class Form1 : Form
   {
      private StarCoreFactory starcore;
      public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         starcore = StarCoreFactory.GetFactory();
         StarServiceClass Service = starcore._InitSimple("calltest", "123", 0, 0, "Test.class?script=java");
         StarObjectClass a = Service._GetObject("TestClass")._New();
         Text = (string)a._Call("getString");
      }

      private void Form1_FormClosed(object sender, FormClosedEventArgs e)
      {
         starcore._ModuleExit();
      }
   }
}
```

13.9 Direct call share library

examples in directory examples\ cle.basic\call.share library

Using CLE, you can directly call share library which has simple interface. For example, call MessageBox on win32:

13.9.1 lua calls MessageBox

```
require "libstarcore"
Service=libstarcore._InitSimple("Test","123",0,0)
Service:_CreateSysRootItemEx("TestItem","")

AtomicClass = Service:_CreateAtomicObjectSimple("TestItem","TestClass","","");
//define function prototype.
Service:_CreateAtomicFunctionSimple(AtomicClass,"MessageBoxA","VS_INT32 MessageBoxA(VS_INT32 hWnd,VS_CHAR
*Text,VS_CHAR *Caption,VS_UINT32 Type);","",true,true);  //stdcall
//attach share library
Service:_AtomicAttach(AtomicClass,"user32.dll")

a = Service.TestClass:_New()
print(a:MessageBoxA(0,"123","123",1) )

Service._ServiceGroup:_ClearService()
libstarcore._ModuleExit()
```

13.9.2 Java calls MessageBox

```
import com.srplab.www.starcore.*;

public class call_messagebox{
        public static void main(String[] args){
                StarCoreFactory starcore=StarCoreFactory.GetFactory();
                StarServiceClass Service = starcore._InitSimple("Test","123",0,0);
                Service._CreateSysRootItemEx("TestItem","",null,null);

        Object[] AtomicClass = Service._CreateAtomicObjectSimple("TestItem","TestClass","","");
        // define function prototype.
        Service._CreateAtomicFunctionSimple(Service._Toint(AtomicClass[0]),"MessageBoxA","VS_INT32
MessageBoxA(VS_INT32 hWnd,VS_CHAR *Text,VS_CHAR *Caption,VS_UINT32 Type);","",true,true);  //stdcall
       // attach share library
        Service._AtomicAttach(Service._Toint(AtomicClass[0]),"user32.dll");

        StarObjectClass a = ((StarObjectClass)Service._Get("TestClass"))._New();
        System.out.println(a._Call("MessageBoxA",0,"123","123",1));

            ((StarSrvGroupClass)Service._Get("_ServiceGroup"))._ClearService();
        starcore._ModuleExit();
    }
}
```

13.9.3 c# calls MessageBox

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace call_messagebox
{
   class Program
   {
      static void Main(string[] args)
      {
            StarCoreFactory starcore=StarCoreFactory.GetFactory();
            StarServiceClass Service = starcore._InitSimple("Test","123",0,0,null);
            Service._CreateSysRootItemEx("TestItem","",null,null);

      Object[] AtomicClass = Service._CreateAtomicObjectSimple("TestItem","TestClass","","");
      Service._CreateAtomicFunctionSimple(Service._Toint(AtomicClass[0]),"MessageBoxA","VS_INT32
MessageBoxA(VS_INT32 hWnd,VS_CHAR *Text,VS_CHAR *Caption,VS_UINT32 Type);","",true,true);
      Service._AtomicAttach(Service._Toint(AtomicClass[0]),"user32.dll");

      StarObjectClass a = ((StarObjectClass)Service._Get("TestClass"))._New();
      Console.WriteLine(a);
      Console.WriteLine(a._Call("MessageBoxA",0,"123","123",1));

          ((StarSrvGroupClass)Service._Get("_ServiceGroup"))._ClearService();
      starcore._ModuleExit();
      }
   }
}
```

13.10 Mixed script language programming

examples in directory examples\cle.basic\embed.call.other

13.10.1 Module to be called

13.10.1.1 lua

```
SrvGroup = libstarcore._GetSrvGroup()
Service = SrvGroup:_GetService("root","123")
Obj=Service:_New("TestClass");
```

```
function Obj:Add(x,y)
  return x+y+self.Value;  --Value is defined by caller
end
Obj.ChildValue = 200;
```

13.10.1.2 python

```
SrvGroup = libstarpy._GetSrvGroup()
Service = SrvGroup._GetService("root","123")
Obj=Service._New("TestClass");
def Obj_Add(self,x,y) :
   return x+y+self.Value;  # Value is defined by caller
Obj.Add = Obj_Add;
Obj.ChildValue = 200;
```

13.10.1.3 java

```
import com.srplab.www.starcore.*;

class MyObjectClass extends StarObjectClass{
        public int Add(StarObjectClass self,int x,int y)
        {
                return x+y+_Toint(self._Get("Value"));  //Value is defined by caller
        }
        public MyObjectClass(StarObjectClass srcobj){
                super(srcobj);
        }
}

public class AddFunction{
        public static void main(String[] args){
                StarCoreFactory starcore= StarCoreFactory.GetFactory();
                StarSrvGroupClass SrvGroup = starcore._GetSrvGroup(0);
       StarServiceClass Service = SrvGroup._GetService("root","123");
                MyObjectClass Obj = new MyObjectClass(Service._New("TestClass"));
                Obj._Set("ChildValue",200);
   }
}
```

13.10.1.4 c#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace AddFunction_Csharp
{
```

```
class MyObjectClass : StarObjectClass{
        public int Add(StarObjectClass self,int x,int y)
        {
                return x+y+_Toint(self._Get("Value"));  //Value is defined by caller
        }
        public MyObjectClass(StarObjectClass srcobj):base(srcobj){
        }
}
   class Program
   {
      static void Main(string[] args)
      {
         StarCoreFactory starcore= StarCoreFactory.GetFactory();
         StarSrvGroupClass SrvGroup = starcore._GetSrvGroup(0);
         StarServiceClass Service = SrvGroup._GetService("root", "123");
         MyObjectClass Obj = new MyObjectClass(Service._New("TestClass"));
         Obj._Set("ChildValue", 200);
      }
   }
}
```

13.10.2 C/C++ call other script

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
        VS_CORESIMPLECONTEXT Context;
        class ClassOfSRPInterface *SRPInterface;
        void *Class,*Object;

        /*--------call as service  */
        SRPInterface = VSCore_InitSimple(&Context,"test","123",0,0,NULL,0,NULL);
        //SRPInterface ->DoFile("lua","Files/AddFunction.lua",NULL,NULL,VS_FALSE);
        SRPInterface ->DoFile("python","Files/AddFunction.py",NULL,NULL,VS_FALSE);
        //SRPInterface ->DoFile("java","Files/AddFunction.class",NULL,NULL,VS_FALSE);
        //SRPInterface ->DoFile("csharp","Files/AddFunction_Csharp.exe",NULL,NULL,VS_FALSE);

        Class = SRPInterface ->GetObjectEx(NULL,"TestClass");
        Object = SRPInterface ->MallocObjectL( SRPInterface->GetIDEx(Class),0,NULL);
        SRPInterface ->ScriptSetInt(Object,"Value",100);
   printf("ChildValue = %d\n",SRPInterface ->ScriptGetInt(Object,"ChildValue"));
        printf("Call Function Ret = %d\n",SRPInterface ->ScriptCall(Object,NULL,"Add","(ii)i",12,34));

        SRPInterface -> Release();
```

```
	VSCore_TermSimple(&Context);
	return 0;
}
```

13.10.3 lua call other script

```
require "libstarcore"
Service=libstarcore._InitSimple("test","123",0,0);
Service:_DoFile("lua","Files/AddFunction.lua","");
--Service:_DoFile("python","Files/AddFunction.py","");
--Service:_DoFile("java","Files/AddFunction.class","");
--Service:_DoFile("csharp","Files/AddFunction_Csharp.exe","");
a = Service.TestClass:_New();
a.Value = 100
print(a.ChildValue)
print(a:Add(12,34))
Service._ServiceGroup:_ClearService()
libstarcore._ModuleExit()
```

13.10.4 python call other script

```
import libstarpy
Service=libstarpy._InitSimple("test","123",0,0);
#Service._DoFile("lua","Files/AddFunction.lua","");
#Service._DoFile("python","Files/AddFunction.py","");
#Service._DoFile("java","Files/AddFunction.class","");
Service._DoFile("csharp","Files/AddFunction_Csharp.exe","");
a = Service.TestClass._New();
a.Value = 100
print(a.ChildValue)
print(a.Add(12,34))
Service._ServiceGroup._ClearService()
libstarpy._ModuleExit()
```

13.10.5 java call other script

```
import com.srplab.www.starcore.*;


public class java_call{
	public static void main(String[] args){
		StarCoreFactory starcore= StarCoreFactory.GetFactory();
		StarServiceClass Service=starcore._InitSimple("test","123",0,0);
	//Service._DoFile("lua","Files/AddFunction.lua","");
		//Service._DoFile("python","Files/AddFunction.py","");
		Service._DoFile("java","Files/AddFunction.class","");
		//Service._DoFile("csharp","Files/AddFunction_Csharp.exe","");
		StarObjectClass a = Service._GetObject("TestClass")._New();
	a._Set("Value",100);
	System.out.println(a._Get("ChildValue"));
```

```
        System.out.println(a._Call("Add",12,34));
        starcore._ModuleExit();
    }
}
```

13.10.6 c# call other script

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace csharp_call
{
    class Program
    {
        static void Main(string[] args)
        {
                StarCoreFactory starcore= StarCoreFactory.GetFactory();
                StarServiceClass Service=starcore._InitSimple("test","123",0,0);
        //Service._DoFile("lua","Files/AddFunction.lua","");
        Service._DoFile("python","Files/AddFunction.py","");
        //Service._DoFile("java", "Files/AddFunction.class", "");
        //Service._DoFile("csharp","Files/AddFunction_Csharp.exe","");
                StarObjectClass a = Service._GetObject("TestClass")._New();
        a._Set("Value",100);
        Console.WriteLine(a._Get("ChildValue"));
        Console.WriteLine(a._Call("Add",12,34));
        starcore._ModuleExit();
        }
    }
}
```

13.11 ASP.NET call CLE extensions

Because cle will be run in different AppDomains, for each call, the function should perform a complete procedure which includes cle init, create service, and cle term as follow.

A simple code is provided below which calls java function to sum two numbers.

Examples in directory examples\ cle.advanced\ csharp.asp

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        StarCoreFactory starcore = StarCoreFactory.GetFactory();
```

```
        StarServiceClass Service = starcore._InitSimple("test", "123", 0, 0, "/srplab/examples/
cle.advanced\csharp.asp/files/AddFunction.class?script=java");
        StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

        StarObjectClass a = Service._GetObject("TestClass")._New();
        Response.Write("<H1> ObjectID = " + a._GetStr("_ID") + "</H1>");
        Response.Write(a._Call("Add", 12, 34));

        SrvGroup._ClearService();
        starcore._ModuleClear();
    }
  }
```

14 CLE distributed function

Examples in directory examples\comm.basic, which include code of C++,lua,python ,java,c# ,etc.

14.1 TCP/UDP communication

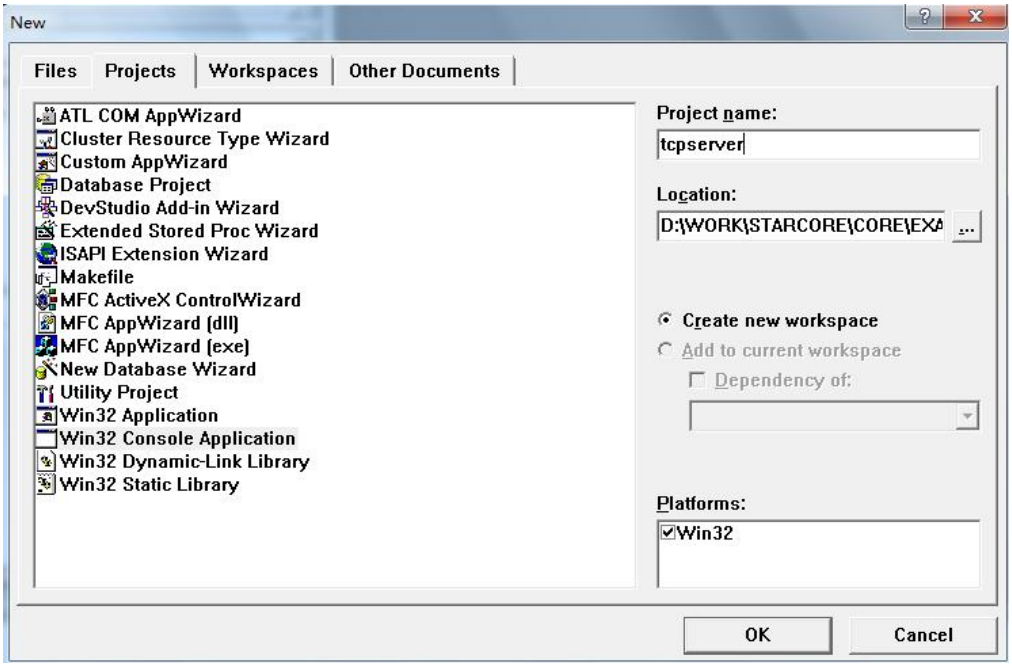
Using function provided by interface ClassOfSRPCommInterface, applications can communicate with each other based on TCP/UDP.

14.1.1 TCP server

14.1.1.1C

14.1.1.1.1 Win32

14.1.1.1.1.1 Create project(VC6)



Set header file path,

set to Multithread,

add starlib_vcm.lib into the project

14.1.1.1.1.2 Create and edit source code

create source file tcpserver,and add to the project,its source code as follow:

```
#include "vsopenapi.h"

//------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
	VS_CORESIMPLECONTEXT Context;
	class ClassOfBasicSRPInterface *BasicSRPInterface;
	class ClassOfSRPCommInterface *CommInterface;
	VS_HANDLE MsgHandle;
	VS_ULONG TcpConnectionID;

	BasicSRPInterface = VSCore_InitSimpleEx(&Context,0,0,NULL,0,NULL);
	if( BasicSRPInterface == NULL ){
		printf("init starcore fail\n");
		return -1;
	}
	printf("init starcore success\n");

	if( argc < 2 ){
		printf("Usage tcpserver portnumber\n");
		return -1;
	}
	CommInterface = Context.VSControlInterface ->GetCommInterface();

   MsgHandle = CommInterface ->CreateMsgQueue(256,256);
	TcpConnectionID = CommInterface -> TCPSetupServer(MsgHandle,100,NULL,atoi(argv[1]),0,0,NULL);

	if( TcpConnectionID == VS_COMM_INVALIDCONNECTION ){
		printf("cretate tcp server on port[%d] fail\n",atoi(argv[1]));
		CommInterface ->Release();
		VSCore_TermSimple(&Context);
		return -1;
	}

	printf("cretate tcp server on port[%d] success\n",atoi(argv[1]));
	printf("finish,enter message loop..\n");
	while( 1 ){
		VS_INT32 Ch;
		Ch = vs_kbhit();
		if( Ch == 27 )
			break;
		{
			struct StructOfSRPCommMessage *CommMessage;
			struct StructOfSRPComm_TCPOnConnect *TCPOnConnect;
			struct StructOfSRPComm_TCPOnClose *TCPOnClose;
			struct StructOfSRPComm_TCPOnRead *TCPOnRead;
			//struct StructOfSRPComm_TCPOnWrite *TCPOnWrite;
			VS_CHAR Buf[256];
			VS_INT32 Size;

			CommMessage = (struct StructOfSRPCommMessage *)CommInterface -
>GetMsgFromQueue(MsgHandle,VS_FALSE);
			if( CommMessage != NULL ){
				switch(CommMessage ->OperateCode){
				case SRPCOMM_TCP_ONCONNECT :
					TCPOnConnect = (struct StructOfSRPComm_TCPOnConnect *)CommMessage->Buf;
					printf("tcp connect[%u] setup fron %d.%d.%d.%d:%d\n",TCPOnConnect->ConnectionID,
```

```
                                                ((struct _in_addr *)&TCPOnConnect->PeerSockAddr.sin_addr)->S_un.S_un_b.s_b1,
                                                ((struct _in_addr *)&TCPOnConnect->PeerSockAddr.sin_addr)->S_un.S_un_b.s_b2,
                                                ((struct _in_addr *)&TCPOnConnect->PeerSockAddr.sin_addr)->S_un.S_un_b.s_b3,
                                                ((struct _in_addr *)&TCPOnConnect->PeerSockAddr.sin_addr)->S_un.S_un_b.s_b4,
                                                vs_ntohs(TCPOnConnect->PeerSockAddr.sin_port));
                                break;
                        case SRPCOMM_TCP_ONREAD :
                                TCPOnRead = (struct StructOfSRPComm_TCPOnRead *)CommMessage->Buf;
                                Size = CommInterface ->TCPRecv(TCPOnRead->ConnectionID,255,Buf);
                                while(Size != 0 ){
                                        Buf[Size] = 0;
                                        printf("receive from[%u] : %s\n",TCPOnRead->ConnectionID,Buf);
                                        Size = CommInterface ->TCPRecv(TCPOnRead->ConnectionID,255,Buf);
                                }
                                break;
                        case SRPCOMM_TCP_ONWRITE :
                                break;
                        case SRPCOMM_TCP_ONCLOSE :
                                TCPOnClose = (struct StructOfSRPComm_TCPOnClose *)CommMessage->Buf;
                                printf("tcp connect[%u] close\n",TCPOnClose ->ConnectionID );
                                break;
                        }
                        CommInterface ->FreeMsgBuf(MsgHandle,(VS_INT8 *)CommMessage);
                }
        }
        while( Context.VSControlInterface -> SRPDispatch(VS_FALSE) == VS_TRUE );
    }
    CommInterface ->TCPRelease(TcpConnectionID);
    CommInterface -> Release();
  VSCore_TermSimple(&Context);
    return 0;
}
```

14.1.1.1.1.3 compile and run

tcpserver 3005

14.1.1.1.2 linux

write Makefile

```
#*******************************************************************************
#
# Makefile for StarCore.
# www.srplab.com
#*******************************************************************************
DEBUG         := YES
PROFILE       := NO
#*******************************************************************************
CC    := gcc
CXX   := g++
LD    := g++
AR    := ar
RANLIB := ranlib

DEBUG_CFLAGS     := -Wall -Wno-format -g -DDEBUG -DENV_LINUX
```

```
RELEASE_CFLAGS   := -Wall -Wno-unknown-pragmas -Wno-format -O3 -DENV_LINUX

LIBS := -ldl -lpthread -lrt
EXTRA_LIBS := /usr/lib/libstarlib.a /usr/lib/libuuid.a          libstarlib.as should be include in the makefile

DEBUG_CXXFLAGS   := ${DEBUG_CFLAGS}
RELEASE_CXXFLAGS := ${RELEASE_CFLAGS}

DEBUG_LDFLAGS    := -g
RELEASE_LDFLAGS  :=

ifeq (YES, ${DEBUG})
  CFLAGS      := ${DEBUG_CFLAGS}
  CXXFLAGS    := ${DEBUG_CXXFLAGS}
  LDFLAGS     := ${DEBUG_LDFLAGS}
else
  CFLAGS      := ${RELEASE_CFLAGS}
  CXXFLAGS    := ${RELEASE_CXXFLAGS}
  LDFLAGS     := ${RELEASE_LDFLAGS}
endif

ifeq (YES, ${PROFILE})
  CFLAGS   := ${CFLAGS} -pg -O3
  CXXFLAGS := ${CXXFLAGS} -pg -O3
  LDFLAGS  := ${LDFLAGS} -pg
endif

#*******************************************************************************
# Makefile code common to all platforms
#*******************************************************************************

CFLAGS   := ${CFLAGS}   ${DEFS}
CXXFLAGS := ${CXXFLAGS} ${DEFS}

#*******************************************************************************
# include source and paths
#*******************************************************************************

INCS_T := /usr/include/starcore
INCS   = $(addprefix -I,$(INCS_T))

TCPSERVER_CXXSRCS := tcpserver.cpp
TCPCLIENT_CXXSRCS := tcpclient.cpp
UDPSERVER_CXXSRCS := udpserver.cpp
UDPCLIENT_CXXSRCS := udpclient.cpp

#*******************************************************************************
TCPSERVER_CXXOBJS := $(TCPSERVER_CXXSRCS:%.cpp=%.o)
TCPCLIENT_CXXOBJS := $(TCPCLIENT_CXXSRCS:%.cpp=%.o)
UDPSERVER_CXXOBJS := $(UDPSERVER_CXXSRCS:%.cpp=%.o)
UDPCLIENT_CXXOBJS := $(UDPCLIENT_CXXSRCS:%.cpp=%.o)

#*******************************************************************************
CXXOBJS := ${TCPSERVER_CXXOBJS} ${TCPCLIENT_CXXOBJS} ${UDPSERVER_CXXOBJS}
${UDPCLIENT_CXXOBJS}
COBJS :=

EXEC_TCPSERVER_OBJS := ${TCPSERVER_CXXOBJS}
EXEC_TCPCLIENT_OBJS := ${TCPCLIENT_CXXOBJS}
EXEC_UDPSERVER_OBJS := ${UDPSERVER_CXXOBJS}
EXEC_UDPCLIENT_OBJS := ${UDPCLIENT_CXXOBJS}

#*******************************************************************************
# Targets of the build
#*******************************************************************************
OBJS_PATH = .
```

```
EXEC_TCPSERVER := ${OBJS_PATH}/tcpserver_linux
EXEC_TCPCLIENT := ${OBJS_PATH}/tcpclient_linux
EXEC_UDPSERVER := ${OBJS_PATH}/udpserver_linux
EXEC_UDPCLIENT := ${OBJS_PATH}/udpclient_linux

all: ${EXEC_TCPSERVER} ${EXEC_TCPCLIENT} ${EXEC_UDPSERVER} ${EXEC_UDPCLIENT}

#*****************************************************************************
# Output
#*****************************************************************************

${EXEC_TCPSERVER}: ${EXEC_TCPSERVER_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_TCPSERVER_OBJS} ${LIBS} ${EXTRA_LIBS}

${EXEC_TCPCLIENT}: ${EXEC_TCPCLIENT_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_TCPCLIENT_OBJS} ${LIBS} ${EXTRA_LIBS}

${EXEC_UDPSERVER}: ${EXEC_UDPSERVER_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_UDPSERVER_OBJS} ${LIBS} ${EXTRA_LIBS}

${EXEC_UDPCLIENT}: ${EXEC_UDPCLIENT_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_UDPCLIENT_OBJS} ${LIBS} ${EXTRA_LIBS}

#*****************************************************************************
# common rules
#*****************************************************************************

${CXXOBJS} :
	${CXX} ${CXXFLAGS} ${INCS} $< -o $@ -c $*.cpp

${COBJS} :
	${CC} ${CFLAGS} ${INCS} -o $@ -c $*.c

dist:
	bash makedistlinux

clean:
	-rm -f core ${CXXOBJS} ${COBJS} ${EXEC_TCPSERVER} ${EXEC_TCPCLIENT} ${EXEC_UDPSERVER}
${EXEC_UDPCLIENT}

depend:
	#makedepend ${INCS} ${SRCS}
```

14.1.1.2 lua

:

```
require "libstarcore"

initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
SrvGroup:_CreateService( "","test", "123",5,0,0,0, 0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )
get communicate interface

CommInterface = SrvGroup:_NewCommInterface()
create TCP server, using port 3005
CommInterface.ConnetionID = CommInterface:_TCPSetupServer(100,nil,3005)
if CommInterface.ConnetionID == 0 then
  print("setup server on port 3005 fail")
  return
end
create binbuf to receive data
```

```
BinBuf = SrvGroup:_NewBinBuf()
message process function of the interface
function CommInterface:_MsgProc(uMes,Msg)
  local Size

  if uMes == self.TCP_ONCONNECT then
TCP connection setup message
     print("tcp connect from ",self:_GetIP(Msg[4]))
  elseif uMes == self.TCP_ONREAD then
Data read message
     Size=self:_TCPRecv(Msg[1],BinBuf)
     while Size ~= 0 do
        print( "receive from",Msg[1],":",BinBuf:_Get(0,0,'s'))
        Size=self:_TCPRecv(Msg[1],BinBuf)
     end
  elseif uMes == self.TCP_ONCLOSE then
Connection close message
     print("tcp connect close ",Msg[1])
  end
end

Message loop
function ExitProc()
    if libstarcore._KeyPress() == 27 then
       return true
    end
    return false
end

libstarcore._MsgLoop( ExitProc )

exit, clear service and starcore
print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Run:

Starapp -e tcpserver.lua

14.1.1.3python

:

```
import sys
import libstarpy

libstarpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = libstarpy._GetSrvGroup()
SrvGroup._CreateService( "","test", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )

CommInterface = SrvGroup._NewCommInterface()

CommInterface.ConnetionID = CommInterface._TCPSetupServer(100,"",3005)
if CommInterface.ConnetionID == 0 :
   print("setup server on port 3005 fail")

print("setup server on port 3005 success")
BinBuf = SrvGroup._NewBinBuf()
def CommInterface_MsgProc(self,uMes,Msg) :
  Size = 0

  if uMes == self.TCP_ONCONNECT :
     print("tcp connect from ",self._GetIP(Msg[3]))
  elif uMes == self.TCP_ONREAD :
```

```
    Size=self._TCPRecv(Msg[0],BinBuf,0)
    while Size != 0 :
       print( "receive from",Msg[0],":",BinBuf._Get(0,0,'s'))
       Size=self._TCPRecv(Msg[0],BinBuf,0)
 elif uMes == self.TCP_ONCLOSE :
   print("tcp connect close ",Msg[0])
CommInterface._MsgProc = CommInterface_MsgProc

def ExitProc() :
   if libstarpy._KeyPress() == 27 :
      return True
   return False

libstarpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

Starapp -e tcpserver.py?script=python

14.1.1.4 java

```
import com.srplab.www.starcore.*;

class MyStarCallBackClass extends StarCallBackClass{
        MyStarCallBackClass(StarCoreFactory starcore){super(starcore);}
        public boolean ExitProc()
   {
     if(StarCore._KeyPress() == 27){
       return true;
     }
     return false;
   }
}

class TCP_CommInterface extends StarCommInterfaceClass{
        private StarBinBufClass BinBuf;
        private StarSrvGroupClass SrvGroup;
   public void _MsgProc(int uMes,Object[] Msg){
    if(uMes == _Getint("TCP_ONCONNECT")){
       System.out.println( "tcp connect from "+_GetIP((StarBinBufClass)Msg[3]));
    }else if(uMes == _Getint("TCP_ONREAD")){
       int Size=_TCPRecv(_Toint(Msg[0]),BinBuf,0);
        System.out.println("tcp read from "+_Toint(Msg[0])+Size);
       while(Size != 0 ){
        System.out.println("receive from"+_Toint(Msg[0])+":"+(String)BinBuf._Get(0,0,"s"));
        Size=_TCPRecv(_Toint(Msg[0]),BinBuf,0);
       }
    }else if(uMes == _Getint("TCP_ONCLOSE")){
       System.out.println("tcp connect close "+_Toint(Msg[0]));
    }
   }
        public TCP_CommInterface(StarSrvGroupClass In_SrvGroup,StarCommInterfaceClass srcobj){
                super(srcobj);
                SrvGroup = In_SrvGroup;
                BinBuf = SrvGroup._NewBinBuf();
        }
}

public class tcpserver{
        public static void main(String[] args){
```

```
            StarCoreFactory starcore= StarCoreFactory.GetFactory();
            MyStarCallBackClass CallBack = new MyStarCallBackClass(starcore);
            StarServiceClass Service=starcore._InitSimple("test","123",0,0);
            StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

            TCP_CommInterface CommInterface = new TCP_CommInterface(SrvGroup,SrvGroup._NewCommInterface());
      int ConnetionID = CommInterface._TCPSetupServer(100,"",3005);
      if(ConnetionID == 0 ){
        System.out.println("setup server on port 3005 fail");
        starcore._ModuleExit();
        return;
      }
      System.out.println("setup server on port 3005 success");

      starcore._MsgLoop(CallBack,"ExitProc");

          System.out.println("Exit...");
          SrvGroup._ClearService();
      starcore._ModuleExit();
   }
}
```

14.1.1.5c#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Star_csharp;

namespace tcpserver
{
class MyStarCallBackClass : StarCallBackClass{
        public MyStarCallBackClass(StarCoreFactory starcore):base(starcore){}
   public Boolean ExitProc()
   {
    if(StarCore._KeyPress() == 27){
      return true;
    }
    return false;
   }
}

class TCP_CommInterface : StarCommInterfaceClass{
        private StarBinBufClass BinBuf;
        private StarSrvGroupClass SrvGroup;
   public void _MsgProc(int uMes,Object[] Msg){
    if(uMes == _Getint("TCP_ONCONNECT")){
      Console.WriteLine( "tcp connect from "+_GetIP((StarBinBufClass)Msg[3]));
   }else if(uMes == _Getint("TCP_ONREAD")){
      int Size=_TCPRecv(_Toint(Msg[0]),BinBuf,0);
       Console.WriteLine("tcp read from "+_Toint(Msg[0])+Size);
      while(Size != 0 ){
       Console.WriteLine("receive from"+_Toint(Msg[0])+":"+(String)BinBuf._Get(0,0,"s"));
       Size=_TCPRecv(_Toint(Msg[0]),BinBuf,0);
      }
   }else if(uMes == _Getint("TCP_ONCLOSE")){
      Console.WriteLine("tcp connect close "+_Toint(Msg[0]));
   }
   }
        public TCP_CommInterface(StarSrvGroupClass In_SrvGroup,StarCommInterfaceClass srcobj):base(srcobj){
            SrvGroup = In_SrvGroup;
            BinBuf = SrvGroup._NewBinBuf();
        }
```

```
}

   class Program
   {
      static void Main(string[] args)
      {
          StarCoreFactory starcore= StarCoreFactory.GetFactory();
          MyStarCallBackClass CallBack = new MyStarCallBackClass(starcore)
              StarServiceClass Service=starcore._InitSimple("test","123",0,0,null);
              StarSrvGroupClass SrvGroup = (StarSrvGroupClass)Service._Get("_ServiceGroup");

              TCP_CommInterface CommInterface = new TCP_CommInterface(SrvGroup,SrvGroup._NewCommInterface());
      int ConnetionID = CommInterface._TCPSetupServer(100,"",3005);
      if(ConnetionID == 0 ){
        Console.WriteLine("setup server on port 3005 fail");
        starcore._ModuleExit();
        return;
      }
      Console.WriteLine("setup server on port 3005 success");

      starcore._MsgLoop(CallBack,"ExitProc");

          Console.WriteLine("Exit...");
          SrvGroup._ClearService();
      starcore._ModuleExit();
      }
   }
}
```

14.1.2 TCP client

14.1.2.1C

14.1.2.1.1 Win32

14.1.2.1.1.1 create project(VC6)

refer to above.

14.1.2.1.1.2 Create and edit source file

Create source file tcpclient,add to project. It's code is shown below.

```
#include "vsopenapi.h"

//------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
        VS_CORESIMPLECONTEXT Context;
        class ClassOfBasicSRPInterface *BasicSRPInterface;
        class ClassOfSRPCommInterface *CommInterface;
        VS_HANDLE MsgHandle;
        VS_ULONG TcpConnectionID,TcpSendTimerID,ConnectFlag,Index;

        BasicSRPInterface = VSCore_InitSimpleEx(&Context,0,0,NULL,0,NULL);
        if( BasicSRPInterface == NULL ){
                printf("init starcore fail\n");
                return -1;
        }
        printf("init starcore success\n");

        if( argc < 3 ){
                printf("Usage tcpclient serverip portnumber\n");
                return -1;
        }
        CommInterface = Context.VSControlInterface ->GetCommInterface();
```

```
  MsgHandle = CommInterface ->CreateMsgQueue(256,256);
    TcpConnectionID = CommInterface -> TCPSetupClient(MsgHandle,100,argv[1],atoi(argv[2]),0,0);
    if( TcpConnectionID == VS_COMM_INVALIDCONNECTION ){
        printf("create tcp client on port[%s:%d] fail\n",argv[1],atoi(argv[2]));
        CommInterface ->Release();
        VSCore_TermSimple(&Context);
        return -1;
    }
    TcpSendTimerID = CommInterface ->SetupTimer(100,0,MsgHandle,0,0);
    ConnectFlag = 0;

    printf("cretate tcp client on port[%s:%d] success\n",argv[1],atoi(argv[2]));
    printf("finish,enter message loop..\n");
    Index = 0;
    while( 1 ){
        VS_INT32 Ch;
        Ch = vs_kbhit();
        if( Ch == 27 )
            break;
        {
            struct StructOfSRPCommMessage *CommMessage;
            struct StructOfSRPComm_TCPOnConnect *TCPOnConnect;
            struct StructOfSRPComm_TCPOnClose *TCPOnClose;
//          struct StructOfSRPComm_TCPOnRead *TCPOnRead;
//          struct StructOfSRPComm_TCPOnWrite *TCPOnWrite;
            VS_CHAR Buf[256];

            sprintf(Buf,"test data  [%d].....",Index);
            CommMessage = (struct StructOfSRPCommMessage *)CommInterface -
>GetMsgFromQueue(MsgHandle,VS_FALSE);
            if( CommMessage != NULL ){
                switch(CommMessage ->OperateCode){
                case SRPCOMM_TCP_ONCONNECT :
                    TCPOnConnect = (struct StructOfSRPComm_TCPOnConnect *)CommMessage->Buf;
                    if( TCPOnConnect ->Result == 0 ){
                        printf("tcp connect[%u] setup\n",TCPOnConnect->ConnectionID);
                        ConnectFlag = 1;
                    }else{
                        printf("tcp connect[%u] success\n",TCPOnConnect->ConnectionID);
                    }
                    break;
                case SRPCOMM_TCP_ONREAD :
                    break;
                case SRPCOMM_TCP_ONWRITE :
                    break;
                case SRPCOMM_TCP_ONCLOSE :
                    TCPOnClose = (struct StructOfSRPComm_TCPOnClose *)CommMessage->Buf;
                    printf("tcp connect[%u] close\n",TCPOnClose ->ConnectionID );
                    ConnectFlag = 0;
                    break;
                case SRPCOMM_TIMER :
                    if( ConnectFlag == 0 )
                        break;
                    CommInterface -> TCPSend(TcpConnectionID,strlen(Buf),Buf,VS_TRUE);
                    printf("Send Packet to server [%d]\n",Index);
                    Index ++;
                    break;
                }
                CommInterface ->FreeMsgBuf(MsgHandle,(VS_INT8 *)CommMessage);
            }
        }
        while( Context.VSControlInterface -> SRPDispatch(VS_FALSE) == VS_TRUE );
    }
    CommInterface -> KillTimer(TcpSendTimerID);
    CommInterface ->TCPRelease(TcpConnectionID);
    CommInterface -> Release();
    VSCore_TermSimple(&Context);
```

```
        return 0;
}
```

14.1.2.1.1.3 Compile and run

tcpclient 127.0.0.1 3005

14.1.2.1.2 linux

write Makefile,skip

14.1.2.2 lua

:

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
SrvGroup:_CreateService( "","test", "123",5,0,0,0, 0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )
get communicate interface
CommInterface = SrvGroup:_NewCommInterface()
Setup TCP client
CommInterface.ConnetionID = CommInterface:_TCPSetupClient(100,"127.0.0.1",3005)
if CommInterface.ConnetionID == 0 then
  print("setup client to server 127.0.0.1:3005 fail")
  return
end
Setup time to send data
CommInterface:_SetupTimer(100,0)
CommInterface.Index = 0
Create binbuf to receive data.
BinBuf = SrvGroup:_NewBinBuf()
Message processing function of the CommInterface
function CommInterface:_MsgProc(uMes,Msg)
 local Size

 if uMes == self.TCP_ONCONNECT then
Setup connection, ) mens succeed
    if Msg[5] == 0 then
      print("tcp connect success ")
    end
 elseif uMes == self.TIMER then
Receive data
    BinBuf:_Clear()
    BinBuf:_Set(0,0,'s',string.format("test data  [%d].....",self.Index));
    self:_TCPSend(self.ConnetionID,BinBuf,true)
    print(string.format("Send Packet to server [%d]",self.Index))
    self.Index = self.Index + 1
 elseif uMes == self.TCP_ONCLOSE then
Close connection
    print("tcp connect close ",Msg[1])
 end
end

Message loop
function ExitProc()
   if libstarcore._KeyPress() == 27 then
```

```
        return true
    end
    return false
end

libstarcore._MsgLoop( ExitProc )

print("Exit...")
Clear service and starcore
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Starapp -e tcpclient.lua

14.1.2.3python

:

```
import sys
import libstarpy

libstarpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = libstarpy._GetSrvGroup()
SrvGroup._CreateService( "","test", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )

CommInterface = SrvGroup._NewCommInterface()

CommInterface.ConnetionID = CommInterface._TCPSetupClient(100,"127.0.0.1",3005)
if CommInterface.ConnetionID == 0 :
   print("setup client to server 127.0.0.1:3005 fail")
   SrvGroup._ClearService()
   libstarpy._ModuleExit()
   sys.exit()

CommInterface._SetupTimer(100,0)
CommInterface.Index = 0

BinBuf = SrvGroup._NewBinBuf()
def CommInterface_MsgProc(self,uMes,Msg) :
  Size = 0

  if uMes == self.TCP_ONCONNECT :
     if Msg[4] == 0 :
        print("tcp connect success ")
  elif uMes == self.TIMER :
     BinBuf._Clear()
     BinBuf._Set(0,0,'s',"test data [{0}].....".format(self.Index))
     self._TCPSend(self.ConnetionID,BinBuf,0,True)
     print("test data [{0}].....".format(self.Index))
     self.Index = self.Index + 1
  elif uMes == self.TCP_ONCLOSE :
     print("tcp connect close ",Msg[0])
CommInterface._MsgProc = CommInterface_MsgProc

def ExitProc() :
    if libstarpy._KeyPress() == 27 :
        return True
    return False

libstarpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
```

```
libstarpy._ModuleExit()
```

Starapp -e tcpclient.py?script=python

Python tcpclient.py

14.1.3 UDP server

14.1.3.1C

14.1.3.1.1 Win32

14.1.3.1.1.1 Create project(VC6)

See above

14.1.3.1.1.2 Create and edit source file

```
#include "vsopenapi.h"

//------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
	VS_CORESIMPLECONTEXT Context;
	class ClassOfBasicSRPInterface *BasicSRPInterface;
	class ClassOfSRPCommInterface *CommInterface;
	VS_HANDLE MsgHandle;
	VS_ULONG UdpConnectionID;

	BasicSRPInterface = VSCore_InitSimpleEx(&Context,0,0,NULL,0,NULL);
	if( BasicSRPInterface == NULL ){
		printf("init starcore fail\n");
		return -1;
	}
	printf("init starcore success\n");

	if( argc < 2 ){
		printf("Usage udpserver portnumber\n");
		return -1;
	}

	CommInterface = Context.VSControlInterface ->GetCommInterface();

   MsgHandle = CommInterface ->CreateMsgQueue(256,256);
	UdpConnectionID = CommInterface -> UDPSetupServer(MsgHandle,100,NULL,atoi(argv[1]),0,0,NULL);


	if( UdpConnectionID == VS_COMM_INVALIDCONNECTION ){
		printf("create udp server on port[%d] fail\n",atoi(argv[1]));
		CommInterface ->Release();
		VSCore_TermSimple(&Context);
		return -1;
	}

	printf("create udp server on port[%d] success\n",atoi(argv[1]));
	printf("finish,enter message loop..\n");
	while( 1 ){
		VS_INT32 Ch;
		Ch = vs_kbhit();
		if( Ch == 27 )
			break;
		{
			struct StructOfSRPCommMessage *CommMessage;
			struct StructOfSRPComm_UDPOnRead *UDPOnRead;
			VS_CHAR Buf[256];
			VS_INT32 Size;
```

```
                CommMessage = (struct StructOfSRPCommMessage *)CommInterface -
>GetMsgFromQueue(MsgHandle,VS_FALSE);
                if( CommMessage != NULL ){
                    switch(CommMessage ->OperateCode){
                    case SRPCOMM_UDP_ONREAD :
                        UDPOnRead = (struct StructOfSRPComm_UDPOnRead *)CommMessage->Buf;
                        Size = 255;
                        if( CommInterface ->UDPRecv(UDPOnRead->ConnectionID,&Size,Buf,NULL) == VS_FALSE ){
                            printf("buf size is small, need[%d]\n",Size);
                            Size = 0;
                        }
                        while( Size != 0 ){
                            Buf[Size] = 0;
                            printf("receive from[%u] : %s\n",UDPOnRead->ConnectionID,Buf);
                            Size = 255;
                            if( CommInterface ->UDPRecv(UDPOnRead->ConnectionID,&Size,Buf,NULL) ==
VS_FALSE ){
                                printf("buf size is small, need[%d]\n",Size);
                                Size = 0;
                            }
                        }
                        break;
                    }
                    CommInterface ->FreeMsgBuf(MsgHandle,(VS_INT8 *)CommMessage);
                }
            }
            while( Context.VSControlInterface -> SRPDispatch(VS_FALSE) == VS_TRUE );
        }
        CommInterface ->UDPRelease(UdpConnectionID);
        CommInterface -> Release();
        VSCore_TermSimple(&Context);
        return 0;
}
```

14.1.3.1.1.3 Compile and run

udpserver 3005

14.1.3.1.2 linux

Write makefile(skip)

14.1.3.2 lua

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
SrvGroup:_CreateService( "","test", "123",5,0,0,0, 0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )
get communicate interface
CommInterface = SrvGroup:_NewCommInterface()

CommInterface.ConnetionID = CommInterface:_UDPSetupServer(100,nil,3005)
if CommInterface.ConnetionID == 0 then
  print("setup udp server on port 3005 fail")
  return
```

```
end

print("setup udp server on port 3005 success")
Create binbuf
BinBuf = SrvGroup:_NewBinBuf()
BinBuf_IP = SrvGroup:_NewBinBuf()
Message processing function of the comminterface
function CommInterface:_MsgProc(uMes,Msg)
  local Size
  if uMes == self.UDP_ONREAD then
     Size=self:_UDPRecv(Msg[1],BinBuf,BinBuf_IP)
     while Size ~= 0 do
        print( "receive from",Msg[1],":",BinBuf:_Get(0,0,'s'),self:_GetIP(BinBuf_IP),self:_GetPort(BinBuf_IP) )
        self:_UDPSend(self.ConnetionID,BinBuf,BinBuf_IP)
        Size=self:_UDPRecv(Msg[1],BinBuf,BinBuf_IP)
     end
  end
end
Message loop
function ExitProc()
    if libstarcore._KeyPress() == 27 then
       return true
    end
    return false
end

libstarcore._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Starapp -e udpserver.lua

14.1.3.3python

```
import sys
import libstarpy

libstarpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = libstarpy._GetSrvGroup()
SrvGroup._CreateService( "","test", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )

CommInterface = SrvGroup._NewCommInterface()

CommInterface.ConnetionID = CommInterface._UDPSetupServer(100,"",3005)
if CommInterface.ConnetionID == 0 :
   print("setup udp server on port 3005 fail")
   SrvGroup._ClearService()
   libstarpy._ModuleExit()
   sys.exit()

print("setup udp server on port 3005 success")

BinBuf = SrvGroup._NewBinBuf()
BinBuf_IP = SrvGroup._NewBinBuf()
def CommInterface_MsgProc(self,uMes,Msg) :
  Size  = 0
  if uMes == self.UDP_ONREAD :
     print( "Receive....")
     Size=self._UDPRecv(Msg[0],BinBuf,BinBuf_IP)
     print( Size )
     while Size != 0 :
```

```
        print( "receive from",Msg[0],":",BinBuf._Get(0,0,'s'),self._GetIP(BinBuf_IP),self._GetPort(BinBuf_IP) )
        self._UDPSend(self.ConnetionID,BinBuf,BinBuf_IP)
        Size=self._UDPRecv(Msg[0],BinBuf,BinBuf_IP)
CommInterface._MsgProc = CommInterface_MsgProc

def ExitProc() :
    if libstarpy._KeyPress() == 27 :
        return True
    return False

libstarpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

Starapp -e udpserver.py?script=python

Python udpserver.py

14.1.4 UDP client

14.1.4.1C

14.1.4.1.1 Win32

14.1.4.1.1.1 Create project(VC6)

See above

14.1.4.1.1.2 Create and edit source file

```
#include "vsopenapi.h"

//---------------------------------------------------------------------------
int main(int argc, char* argv[])
{
        VS_CORESIMPLECONTEXT Context;
        class ClassOfBasicSRPInterface *BasicSRPInterface;
        class ClassOfSRPCommInterface *CommInterface;
        VS_HANDLE MsgHandle;
        VS_ULONG UdpConnectionID,UdpSendTimerID,Index;

        BasicSRPInterface = VSCore_InitSimpleEx(&Context,0,0,NULL,0,NULL);
        if( BasicSRPInterface == NULL ){
                printf("init starcore fail\n");
                return -1;
        }
        printf("init starcore success\n");

        if( argc < 3 ){
                printf("Usage udpclient serverip portnumber\n");
                return -1;
        }
        CommInterface = Context.VSControlInterface ->GetCommInterface();

    MsgHandle = CommInterface ->CreateMsgQueue(256,256);
        UdpConnectionID = CommInterface -> UDPSetupClient(MsgHandle,100,0,0);

        if( UdpConnectionID == VS_COMM_INVALIDCONNECTION ){
                printf("create udp client fail\n");
                CommInterface ->Release();
                VSCore_TermSimple(&Context);
                return -1;
        }

        UdpSendTimerID = CommInterface ->SetupTimer(100,0,MsgHandle,0,0);
```

```
	Index = 0;

	printf("create udp client success\n");
	printf("finish,enter message loop..\n");
	while( 1 ){
		VS_INT32 Ch;
		Ch = vs_kbhit();
		if( Ch == 27 )
			break;
		{
			struct StructOfSRPCommMessage *CommMessage;
			VS_CHAR Buf[256];
			VSSOCKADDR_IN SocketAddr;

			CommMessage = (struct StructOfSRPCommMessage *)CommInterface -
>GetMsgFromQueue(MsgHandle,VS_FALSE);
			if( CommMessage != NULL ){
				switch(CommMessage ->OperateCode){
				case SRPCOMM_TIMER :
					sprintf(Buf,"test data  [%d].....",Index);
					CommInterface ->UDPSetSockAddr(argv[1],atoi(argv[2]),&SocketAddr);
					CommInterface -> UDPSend(UdpConnectionID,strlen(Buf),Buf,&SocketAddr);
					printf("Send Packet to server [%d]\n",Index);
					Index ++;
					break;
				}
				CommInterface ->FreeMsgBuf(MsgHandle,(VS_INT8 *)CommMessage);
			}
		}
		while( Context.VSControlInterface -> SRPDispatch(VS_FALSE) == VS_TRUE );
	}
	CommInterface -> KillTimer(UdpSendTimerID);
	CommInterface ->UDPRelease(UdpConnectionID);
	CommInterface -> Release();
	VSCore_TermSimple(&Context);
	return 0;
}
```

14.1.4.1.1.3 Compile and run

udpclient 127.0.0.1 3005

14.1.4.1.2 linux

Write makefile(skip)

14.1.4.2 lua

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
SrvGroup:_CreateService( "","test", "123",5,0,0,0, 0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )
get communicate interface
CommInterface = SrvGroup:_NewCommInterface()
```

```
CommInterface.ConnetionID = CommInterface:_UDPSetupClient(100)
if CommInterface.ConnetionID == 0 then
  print("setup udp client fail")
  return
end
print("setup udp client success")
Create timer
CommInterface:_SetupTimer(100,0)
CommInterface.Index = 0
print(CommInterface.ConnetionID)
Create binbuf
BinBuf = SrvGroup:_NewBinBuf()
BinBuf_IP = SrvGroup:_NewBinBuf()
CommInterface:_UDPSetSockAddr("127.0.0.1",3005,BinBuf_IP)
Message processing function of the comminterface
function CommInterface:_MsgProc(uMes,Msg)
 local Size

 if uMes == self.TIMER then
   BinBuf:_Clear()
   BinBuf:_Set(0,0,'s',string.format("test data  [%d].....",self.Index));
   self:_UDPSend(self.ConnetionID,BinBuf,BinBuf_IP)
   print(string.format("Send Packet to server [%d]",self.Index))
   self.Index = self.Index + 1
 end
 if uMes == self.UDP_ONREAD then
   Size=self:_UDPRecv(Msg[1],BinBuf,BinBuf_IP)
   while Size ~= 0 do
      print( "receive from",Msg[1],":",BinBuf:_Get(0,0,'s'),self:_GetIP(BinBuf_IP),self:_GetPort(BinBuf_IP) )
      Size=self:_UDPRecv(Msg[1],BinBuf,BinBuf_IP)
   end
 end
end
Message loop
function ExitProc()
   if libstarcore._KeyPress() == 27 then
      return true
   end
   return false
end

libstarcore._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Starapp -e udpclient.lua

14.1.4.3python

```
import sys
import libstarpy

libstarpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = libstarpy._GetSrvGroup()
SrvGroup._CreateService( "","test", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )

CommInterface = SrvGroup._NewCommInterface()

CommInterface.ConnetionID = CommInterface._UDPSetupClient(100)
if CommInterface.ConnetionID == 0 :
  print("setup udp client fail")
```

```
  SrvGroup._ClearService()
  libstarpy._ModuleExit()
  sys.exit()

print("setup udp client success")
CommInterface._SetupTimer(100,0)
CommInterface.Index = 0
print(CommInterface.ConnetionID)

BinBuf = SrvGroup._NewBinBuf()
BinBuf_IP = SrvGroup._NewBinBuf()
CommInterface._UDPSetSockAddr("127.0.0.1",3005,BinBuf_IP)

def CommInterface_MsgProc(self, uMes,Msg) :
 Size = 0
 if uMes == self.TIMER :
    BinBuf._Clear()
    BinBuf._Set(0,0,'s',"test data  [{0}].....".format(self.Index));
    self._UDPSend(self.ConnetionID,BinBuf,BinBuf_IP)
    print("test data  [{0}].....".format(self.Index))
    self.Index = self.Index + 1
 if uMes == self.UDP_ONREAD :
    Size=self._UDPRecv(Msg[0],BinBuf,BinBuf_IP)
    while Size != 0 :
       print( "receive from",Msg[0],":",BinBuf._Get(0,0,'s'),self._GetIP(BinBuf_IP),self._GetPort(BinBuf_IP) )
       Size=self._UDPRecv(Msg[0],BinBuf,BinBuf_IP)
CommInterface._MsgProc = CommInterface_MsgProc

def ExitProc() :
   if libstarpy._KeyPress() == 27 :
      return True
   return False

libstarpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

Starapp -e udpclient.py?script=python

14.2 Remotecall

14.2.1 Create server side application

14.2.1.1C

Examples in directore examples\comm.basic\remotecall.c

14.2.1.1.1 Win32

14.2.1.1.1.1 Create project(VC6)

skip

14.2.1.1.1.2 Create and edit source file

Create source file test_server,add to project. It's

```
#include "vsopenapi.h"

static VS_ULONG MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_ULONG wParam, VS_ULONG
lParam, VS_BOOL &IsProcessed, VS_ULONG Para )
```

```
{
   switch( uMsg ){
   case MSG_VSDISPMSG :
       case MSG_VSDISPLUAMSG :
              printf("[core]%s\n",(VS_CHAR *)wParam);
              break;
   case MSG_DISPMSG :
       case MSG_DISPLUAMSG :
              printf("%s\n",(VS_CHAR *)wParam);
      break;
       case MSG_EXIT :
              break;
   }
   return 0;
}

static VS_INT32 GetNumber(void *Object,VS_INT32 Para)
{
   printf( "Remote Call Number [%d]\n ",Para);
       return Para + 1;
}

static VS_CHAR *GetString(void *Object,VS_CHAR *Para)
{
       static VS_CHAR CharBuf[128];

   printf( "Remote Call String [%s]\n",Para);
       sprintf(CharBuf,"%sasdfsaf",Para);
   return CharBuf;
}

VS_PARAPKGPTR ParaPkgPtr;

static VS_PARAPKGPTR GetPkg(void *Object,VS_PARAPKGPTR Para)
{
   printf( "Remote Call Pkg [%d]",Para ->GetInt(0));
       ParaPkgPtr ->Clear();
       ParaPkgPtr ->InsertStr(0,"asdfsaf");
   return ParaPkgPtr;
}

//---------------------------------------------------------------------------
int main(int argc, char* argv[])
{
       VS_CORESIMPLECONTEXT Context;
   class ClassOfSRPInterface *SRPInterface;
       VS_UUID ClassID;
       void *AtomicClass,*GetPkg_AtomicFunction,*Object,*GetNumber_AtomicFunction,*GetString_AtomicFunction;
       VS_CHAR *ErrorInfo;

       SRPInterface = VSCore_InitSimple(&Context,"RemoteCallServer","123",3008,0,NULL,0,NULL);
       if( SRPInterface == NULL ){
              printf("init starcore fail\n");
              return -1;
       }
       printf("init starcore success\n");
       SRPInterface ->CreateSysRootItem("TestItem","",NULL,NULL);
       SRPInterface ->ActiveSysRootItem( "TestItem" );
       //---Create Atomic Class, for define function, no attribute
       AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClass",NULL,NULL,&ErrorInfo);
       GetNumber_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetNumber","VS_INT32
GetNumber(VS_INT32 Para);",NULL,&ErrorInfo,VS_FALSE,VS_FALSE);
       GetString_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetString","VS_CHAR
*GetString(VS_CHAR *Para);",NULL,&ErrorInfo,VS_FALSE,VS_FALSE);
       GetPkg_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetPkg","VS_PARAPKGPTR
GetPkg(VS_PARAPKGPTR Para);",NULL,&ErrorInfoo,VS_FALSE,VS_FALSE);
   //---Set Function Address
```

```
	SRPInterface -> SetAtomicFunction(GetNumber_AtomicFunction,(void *)GetNumber);
	SRPInterface -> SetAtomicFunction(GetString_AtomicFunction,(void *)GetString);
	SRPInterface -> SetAtomicFunction(GetPkg_AtomicFunction,(void *)GetPkg);

	ParaPkgPtr = SRPInterface -> GetParaPkgInterface();

	printf("create TestObject for remotecall..\n");
	SRPInterface ->GetAtomicID(AtomicClass,&ClassID);
	Object = SRPInterface ->MallocGlobalObject(SRPInterface->GetSysRootItem("TestItem"),0,&ClassID,0,NULL,0);
	SRPInterface ->SetName(Object,"TestObject");

	printf("finish,enter message loop..\n");
	while( 1 ){
		VS_INT32 Ch;
		Ch = vs_kbhit();
		if( Ch == 27 )
			break;
		Context.VSControlInterface -> SRPDispatch(VS_FALSE);
	}
	ParaPkgPtr -> Release();
	SRPInterface -> Release();
	VSCore_TermSimple(&Context);
	return 0;
}
```

14.2.1.1.1.3 Compile and run

test_server

14.2.1.1.2 linux

Write Makefile,as follows:

```
#*******************************************************************************
#
# Makefile for StarCore.
# www.srplab.com
#*******************************************************************************
DEBUG		:= YES
PROFILE		:= NO
#*******************************************************************************
CC	:= gcc
CXX	:= g++
LD	:= g++
AR	:= ar
RANLIB := ranlib

DEBUG_CFLAGS	:= -Wall -Wno-format -g -DDEBUG -DENV_LINUX
RELEASE_CFLAGS	:= -Wall -Wno-unknown-pragmas -Wno-format -O3 -DENV_LINUX

LIBS := -ldl -lpthread -lrt
EXTRA_LIBS := ../../output/linux/libstarlib.a /usr/lib/libuuid.a

DEBUG_CXXFLAGS	:= ${DEBUG_CFLAGS}
RELEASE_CXXFLAGS := ${RELEASE_CFLAGS}

DEBUG_LDFLAGS	:= -g
RELEASE_LDFLAGS	:=

ifeq (YES, ${DEBUG})
  CFLAGS	:= ${DEBUG_CFLAGS}
  CXXFLAGS	:= ${DEBUG_CXXFLAGS}
  LDFLAGS	:= ${DEBUG_LDFLAGS}
```

```
else
  CFLAGS      := ${RELEASE_CFLAGS}
  CXXFLAGS    := ${RELEASE_CXXFLAGS}
  LDFLAGS     := ${RELEASE_LDFLAGS}
endif

ifeq (YES, ${PROFILE})
  CFLAGS   := ${CFLAGS} -pg -O3
  CXXFLAGS := ${CXXFLAGS} -pg -O3
  LDFLAGS  := ${LDFLAGS} -pg
endif

#*******************************************************************************
# Makefile code common to all platforms
#*******************************************************************************

CFLAGS   := ${CFLAGS}   ${DEFS}
CXXFLAGS := ${CXXFLAGS} ${DEFS}

#*******************************************************************************
# include source and paths
#*******************************************************************************

INCS_T := /usr/include/starcore
INCS   = $(addprefix -I,$(INCS_T))

TEST_SERVER_CXXSRCS := test_server.cpp

#*******************************************************************************
TEST_SERVER_CXXOBJS := $(TEST_SERVER_CXXSRCS:%.cpp=%.o)

#*******************************************************************************
CXXOBJS := ${TEST_SERVER_CXXOBJS}
COBJS :=

EXEC_TEST_SERVER_OBJS := ${TEST_SERVER_CXXOBJS}

#*******************************************************************************
# Targets of the build
#*******************************************************************************
OBJS_PATH = output/linux

EXEC_TEST_SERVER := ${OBJS_PATH}/test_server

all: ${EXEC_TEST_SERVER}

#*******************************************************************************
# Output
#*******************************************************************************

${EXEC_TEST_SERVER}: ${EXEC_TEST_SERVER_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_TEST_SERVER_OBJS} ${LIBS} ${EXTRA_LIBS}

#*******************************************************************************
# common rules
#*******************************************************************************

${CXXOBJS} :
	${CXX} ${CXXFLAGS} ${INCS} $< -o $@ -c $*.cpp

${COBJS} :
	${CC} ${CFLAGS} ${INCS} -o $@ -c $*.c

dist:
	bash makedistlinux

clean:
```

```
	-rm -f core ${CXXOBJS} ${COBJS} ${EXEC_TEST_SERVER}

depend:
	#makedepend ${INCS} ${SRCS}
```

14.2.1.2 lua

Examples in directoryexamples\comm.basic\remotecall.lua

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",3008) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
--create service
SrvGroup:_CreateService( "","RemoteCallServer", "123",5,0,0,0, 0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )
Service = SrvGroup:_GetService("root","123")

--create service item(object group)
Service:_CreateSysRootItem("TestItem","")
active service item
Service:_ActiveSysRootItem("TestItem")
SrvItem = Service:_GetSysRootItem( "TestItem" )

print(Service,SrvItem)
a = Service:_NewGlobal(SrvItem)
a._Name = "TestObject"
a.___Value = "Global Attribute"

function a:GetNumber( para )
   print( "Remote Call Number ",para)
   return para + 1
end

function a:GetString( para )
   print( "Remote Call String ",para)
   return para .. "asdfsaf"
end

function a:GetPkg( para )
   print( a.___Value, "Remote Call Pkg ",para[0])
   ParaPkg = SrvGroup:_NewParaPkg()
   ParaPkg[0] = "asdfsaf"
   return ParaPkg
end

--
print( "Server Start ok....")
Message loop
function ExitProc()
   if libstarcore._KeyPress() == 27 then
      return true
   end
   return false
end

libstarcore._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

14.2.1.3python

Examples in directoryexamples\comm.basic\remotecall.python

```
import sys
import libstarpy
initstarcore(cle)
libstarpy._InitCore(True,True,False,True,"",0,"",3008)
get service group 0, and create service
SrvGroup = libstarpy._GetSrvGroup()
#--create service
SrvGroup._CreateService( "","RemoteCallServer", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )
Service = SrvGroup._GetService("root","123")

#--create service item(object group)
Service._CreateSysRootItem("TestItem","")
active service item
Service._ActiveSysRootItem("TestItem")
SrvItem = Service._GetSysRootItem( "TestItem" )

print(Service,SrvItem)
a = Service._NewGlobal(SrvItem)
a._Name = "TestObject"
a.___Value = "Global Attribute"

def a_GetNumber( self, para ) :
    print( "Remote Call Number ",para)
    return para + 1
a.GetNumber = a_GetNumber

def a_GetString( self, para ) :
    print( "Remote Call String ",para)
    return para+"asdfsaf"
a.GetString = a_GetString

def a_GetPkg( self, para ) :
    print( a.___Value, "Remote Call Pkg ",para._Get(0))
    ParaPkg = SrvGroup._NewParaPkg()
    ParaPkg._Set(0,"asdfsaf")
    return ParaPkg
a.GetPkg = a_GetPkg

#--
print( "Server Start ok....")
Message loop
def ExitProc() :
    if libstarpy._KeyPress() == 27 :
        return True
    return False

libstarpy._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

Run:

starapp -e test_server.py?script=python

python test_server.py

14.2.2 Create client side application

14.2.2.1Win32

14.2.2.1.1 Create project(VC6)

See above

14.2.2.1.2 Create and edit source file

Create source file test_client,add to project. It's code is shown below.

```
#include "vsopenapi.h"

//------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
	VS_CORESIMPLECONTEXT Context;
	class ClassOfBasicSRPInterface *BasicSRPInterface;
   class ClassOfSRPInterface *SRPInterface;
	VS_UUID FunctionID;
	void *SysRootItem,*Object;
	VS_ULONG RetCode;
	VS_PARAPKGPTR ReqParaPkg,RetParaPkg;

	BasicSRPInterface = VSCore_InitSimpleEx(&Context,0,0,NULL,0,NULL);
	if( BasicSRPInterface == NULL ){
		printf("init starcore fail\n");
		return -1;
	}
	printf("init starcore success\n");

	if( argc < 2 ){
		printf("useage test_client serverip\n");
		return -1;
	}

	BasicSRPInterface = Context.VSControlInterface ->CreateBasicInterface(1,VS_CLIENT_USER);
	if( BasicSRPInterface ->SConnect("",argv[1],3008,NULL,NULL,NULL) == 0 ){
		printf("Fail to connect to server\n");
		BasicSRPInterface ->Release();
		VSCore_TermSimple(&Context);
		return 0;
	}
	BasicSRPInterface ->WaitServiceSync(0);
	printf( "Success To Connect...\n" );
	SRPInterface = BasicSRPInterface ->GetSRPInterface(NULL,NULL,NULL);
	SysRootItem = SRPInterface ->GetSysRootItem("TestItem");
	SRPInterface ->WaitSysRootItemSync(SysRootItem);

	Object = SRPInterface ->GetObjectEx(NULL,"TestObject");
	SRPInterface ->GetFunctionID(Object,"GetNumber",&FunctionID);
	printf( "%d\n",SRPInterface->SRemoteCall(0,0,&RetCode,Object,&FunctionID,123));
	SRPInterface ->GetFunctionID(Object,"GetString",&FunctionID);
	printf( "%s\n",SRPInterface->SRemoteCall(0,0,&RetCode,Object,&FunctionID,"Hello"));
	SRPInterface ->GetFunctionID(Object,"GetPkg",&FunctionID);
	ReqParaPkg = SRPInterface ->GetParaPkgInterface();
  ReqParaPkg ->InsertInt(0,123);
	RetParaPkg = (VS_PARAPKGPTR)SRPInterface->SRemoteCall(0,0,&RetCode,Object,&FunctionID,ReqParaPkg);
	printf("%s\n",RetParaPkg->GetStr(0));
	RetParaPkg->Release();
	ReqParaPkg->Release();

	BasicSRPInterface ->Release();
	VSCore_TermSimple(&Context);
	return 0;
}
```

14.2.2.1.3 Compile and run

test_client

14.2.2.2 linux

Write Makefile,see above

14.2.2.3 lua

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
    return
end
SrvGroup = libstarcore._CreateSrvGroup(1,libstarcore.VS_CLIENT_USER);

print(SrvGroup,libstarcore.VS_CLIENT_USER);
ret = SrvGroup:_SConnect("","127.0.0.1",3008,"","")
if ret == 0 then
    print( "Fail To Connect..." )
    SrvGroup:_ClearService()
    libstarcore._ModuleExit()
    return
end
Wait service to sync, and then get service item, and wait service item to sync, and the can get object
SrvGroup:_WaitServiceSync()

print( "Success To Connect..." )

Service = SrvGroup:_GetService("root","123")
active service item
Service:_ActiveSysRootItem("TestItem")
SrvItem = Service:_GetSysRootItem("TestItem")
wait service item to sync
SrvItem:_WaitSync()

print( Service.TestObject:_SRemoteCall(0,0,"GetNumber",123) )
print( Service.TestObject:_SRemoteCall(0,0,"GetString","Hello") )

Para = Service._ServiceGroup:_NewParaPkg()
Para[0] = 123
RetCode,RetValue = Service.TestObject:_SRemoteCall(0,0,"GetPkg",Para)

print( RetValue[0] )
exit, clear service and starcore
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

14.2.2.4 python

```
import sys
import libstarpy
initstarcore(cle)
libstarpy._InitCore(True,True,False,True,"",0,"",0)
```

```
SrvGroup = libstarpy._CreateSrvGroup(1,libstarpy.VS_CLIENT_USER);

ret = SrvGroup._SConnect("","127.0.0.1",3008,"","")
if ret == 0 :
    print( "Fail To Connect..." )
    SrvGroup._ClearService()
    libstarpy._ModuleExit()
    raise Exception("")
Wait service to sync, and then get service item, and wait service item to sync, and the can get object
SrvGroup._WaitServiceSync()

print( "Success To Connect..." )

Service = SrvGroup._GetService("","")
active service item
Service._ActiveSysRootItem("TestItem")
SrvItem = Service._GetSysRootItem("TestItem")
wait service item to sync
SrvItem._WaitSync()

print(Service.TestObject)

print( Service.TestObject._SRemoteCall(0,0,"GetNumber",123) )
print( Service.TestObject._SRemoteCall(0,0,"GetString","Hello") )

Para = Service._ServiceGroup._NewParaPkg()
Para._Set(0, 123 )
RetCode,RetValue = Service.TestObject._SRemoteCall(0,0,"GetPkg",Para)
exit, clear service and starcore
print( RetValue._Get(0) )
SrvGroup._ClearServiceEx()
libstarpy._ModuleExit()
```

starapp -e test_client.py?script=python

14.2.3 Creating and using starcore service

14.2.3.1Create starcore service

Examples in directoryexamples\comm.basic\service

14.2.3.1.1 Create starcore service data file

14.2.3.1.1.1 C

14.2.3.1.1.1.1Win32

14.2.3.1.1.1.1.1 Create project(VC6)

skip

14.2.3.1.1.1.1.2 create source file

Create new file create_service.cpp,

```
    #include "vsopenapi.h"
```

```
	static VS_ULONG MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_ULONG wParam, VS_ULONG
lParam, VS_BOOL &IsProcessed, VS_ULONG Para )
	{
	   switch( uMsg ){
	   case MSG_VSDISPMSG :
	       case MSG_VSDISPLUAMSG :
	            printf("[core]%s\n",(VS_CHAR *)wParam);
	            break;
	   case MSG_DISPMSG :
	       case MSG_DISPLUAMSG :
	            printf("%s\n",(VS_CHAR *)wParam);
	          break;
	       case MSG_EXIT :
	            break;
	   }
	   return 0;
	}

	//---------------------------------------------------------------------------
	int main(int argc, char* argv[])
	{
	       VS_CORESIMPLECONTEXT Context;
	       class ClassOfBasicSRPInterface *BasicSRPInterface;
	   class ClassOfSRPInterface *SRPInterface;
	       void *AtomicClass;
	       VS_CHAR *ErrorInfo;

	       BasicSRPInterface = VSCore_InitSimpleEx(&Context,0,0,MsgCallBack,0,NULL);
	       if( BasicSRPInterface == NULL ){
	            printf("init starcore fail\n");
	            return -1;
	       }
	       printf("init starcore success\n");

	       BasicSRPInterface                ->CreateService("","RemoteCallServer",_UUIDPTR("5D0465E1-4203-4d44-9860-
8B56C4790BC2"),"123",0,0,0,0,0,0);
	       SRPInterface = BasicSRPInterface ->GetSRPInterface("RemoteCallServer","root","123");
	       SRPInterface ->CreateSysRootItem("TestItem","",NULL,NULL);
	       SRPInterface ->ActiveSysRootItem( "TestItem" );
	       //---Create Atomic Class, for define function, no attribute
	       AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClass",NULL,_UUIDPTR("3547400A-
AFCF-4434-8341-D4FF93D73AAE"),&ErrorInfo);
	       SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetNumber","VS_INT32 GetNumber(VS_INT32
Para);",_UUIDPTR("5859F990-57FE-4af7-86B6-9E47CED15444"),&ErrorInfo,VS_FALSE,VS_FALSE);
	       SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetString","VS_CHAR *GetString(VS_CHAR
*Para);",_UUIDPTR("0DA48468-A90E-4351-BB38-7BDD520451FE"),&ErrorInfo,VS_FALSE,VS_FALSE);
	       SRPInterface                ->CreateAtomicFunctionSimple(AtomicClass,"GetPkg","VS_PARAPKGPTR
GetPkg(VS_PARAPKGPTR Para);",_UUIDPTR("9C8BFB8F-6C48-4e32-A89A-
D41DE3A9627E"),&ErrorInfo,VS_FALSE,VS_FALSE);

	       SRPInterface -> CreateAtomicModule("TestModule",0,_UUIDPTR("0E63CE93-7C1C-4a41-857A-5824E1482023"));

	       SRPInterface -> SaveService("..\\..\\script");

	       printf("save service to ..\\..\\script \n");

	       VSCore_TermSimple(&Context);
	       return 0;
	}
```

14.2.3.1.1.1.3 Compile and run

Run create_RemoetCallServe

14.2.3.1.1.1.2linux

Write Makefile

```
#*******************************************************************************
#
# Makefile for StarCore.
# www.srplab.com
#*******************************************************************************
DEBUG         := YES
PROFILE       := NO
#*******************************************************************************
CC    := gcc
CXX   := g++
LD    := g++
AR    := ar
RANLIB := ranlib

DEBUG_CFLAGS     := -Wall -Wno-format -g -DDEBUG -DENV_LINUX
RELEASE_CFLAGS   := -Wall -Wno-unknown-pragmas -Wno-format -O3 -DENV_LINUX

LIBS := -ldl -lpthread -lrt
EXTRA_LIBS := /usr/lib/libstarlib.a /usr/lib/libuuid.a

DEBUG_CXXFLAGS   := ${DEBUG_CFLAGS}
RELEASE_CXXFLAGS := ${RELEASE_CFLAGS}

DEBUG_LDFLAGS    := -g
RELEASE_LDFLAGS  :=

ifeq (YES, ${DEBUG})
  CFLAGS      := ${DEBUG_CFLAGS}
  CXXFLAGS    := ${DEBUG_CXXFLAGS}
  LDFLAGS     := ${DEBUG_LDFLAGS}
else
  CFLAGS      := ${RELEASE_CFLAGS}
  CXXFLAGS    := ${RELEASE_CXXFLAGS}
  LDFLAGS     := ${RELEASE_LDFLAGS}
endif

ifeq (YES, ${PROFILE})
  CFLAGS   := ${CFLAGS} -pg -O3
  CXXFLAGS := ${CXXFLAGS} -pg -O3
  LDFLAGS  := ${LDFLAGS} -pg
endif

#*******************************************************************************
# Makefile code common to all platforms
#*******************************************************************************

CFLAGS   := ${CFLAGS}   ${DEFS}
CXXFLAGS := ${CXXFLAGS} ${DEFS}

#*******************************************************************************
# include source and paths
#*******************************************************************************

INCS_T := /usr/include/starcore
INCS   = $(addprefix -I,$(INCS_T))

CREATE_REMOTECALLSERVER_CXXSRCS := create_RemoteCallServer.cpp

#*******************************************************************************
CREATE_REMOTECALLSERVER_CXXOBJS := $(CREATE_REMOTECALLSERVER_CXXSRCS:%.cpp=%.o)

#*******************************************************************************
CXXOBJS := ${CREATE_REMOTECALLSERVER_CXXOBJS}
```

```
COBJS :=

EXEC_CREATE_REMOTECALLSERVER_OBJS := ${CREATE_REMOTECALLSERVER_CXXOBJS}

#*****************************************************************************
# Targets of the build
#*****************************************************************************
EXEC_CREATE_REMOTECALLSERVER := create_RemoteCallServer_linux

all: ${EXEC_CREATE_REMOTECALLSERVER}

#*****************************************************************************
# Output
#*****************************************************************************

${EXEC_CREATE_REMOTECALLSERVER}: ${EXEC_CREATE_REMOTECALLSERVER_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_CREATE_REMOTECALLSERVER_OBJS} ${LIBS} ${EXTRA_LIBS}


#*****************************************************************************
# common rules
#*****************************************************************************

${CXXOBJS} :
	${CXX} ${CXXFLAGS} ${INCS} $< -o $@ -c $*.cpp

${COBJS} :
	${CC} ${CFLAGS} ${INCS} -o $@ -c $*.c

dist:
	bash makedistlinux

clean:
	-rm -f core ${CXXOBJS} ${COBJS} ${EXEC_CREATE_REMOTECALLSERVER}

depend:
	#makedepend ${INCS} ${SRCS}
```

Run make to generate executable file

14.2.3.1.1.2 Create service using lua

```
require "libstarcore"

if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
   return
end

SrvGroup = libstarcore:_GetSrvGroup()
--Create service
SrvGroup:_CreateService( "","RemoteCallServer", "123",5,0,0,0,0,0,"5D0465E1-4203-4d44-9860-8B56C4790BC2" )
Service = SrvGroup:_GetService("root","123")

--create service item(object group)
Service:_CreateSysRootItem("TestItem","")
Service:_ActiveSysRootItem("TestItem")
SrvItem = Service:_GetSysRootItem( "TestItem" )
--Create Atomic Class, for define function, no attribute
AtomicClass = Service:_CreateAtomicObjectSimple("TestItem","TestClass","","3547400A-AFCF-4434-8341-
D4FF93D73AAE");
Service:_CreateAtomicFunctionSimple(AtomicClass,"GetNumber","VS_INT32 GetNumber(VS_INT32 Para);","5859F990-
57FE-4af7-86B6-9E47CED15444",false,false);
```

```
	Service:_CreateAtomicFunctionSimple(AtomicClass,"GetString","VS_CHAR *GetString(VS_CHAR *Para);","0DA48468-A90E-4351-BB38-7BDD520451FE",false,false);
	Service:_CreateAtomicFunctionSimple(AtomicClass,"GetPkg","VS_PARAPKGPTR GetPkg(VS_PARAPKGPTR Para);","9C8BFB8F-6C48-4e32-A89A-D41DE3A9627E",false,false);
	Service:_CreateAtomicModule("TestModule",0,"0E63CE93-7C1C-4a41-857A-5824E1482023");
	Service:_Save("..\\..\\script");

	print("save service to ..\\..\\script")

	SrvGroup:_ClearService()
	libstarcore._ModuleExit()
```

14.2.3.1.1.3 Create service using python

```
	import sys
	import libstarpy

	libstarpy._InitCore(True,True,False,True,"",0,"",3008)

	SrvGroup = libstarpy._GetSrvGroup()
	#--Create service
	SrvGroup._CreateService( "","RemoteCallServer", "123",5,0,0,0,0,0,"5D0465E1-4203-4d44-9860-8B56C4790BC2" )
	Service = SrvGroup._GetService("root","123")

	#--create service item(object group)
	Service._CreateSysRootItem("TestItem","")
	Service._ActiveSysRootItem("TestItem")
	SrvItem = Service._GetSysRootItem( "TestItem" )
	#--Create Atomic Class, for define function, no attribute
	AtomicClass,ErrorInfo = Service._CreateAtomicObjectSimple("TestItem","TestClass","","3547400A-AFCF-4434-8341-D4FF93D73AAE");
	Service._CreateAtomicFunctionSimple(AtomicClass,"GetNumber","VS_INT32 GetNumber(VS_INT32 Para);","5859F990-57FE-4af7-86B6-9E47CED15444",False,False);
	Service._CreateAtomicFunctionSimple(AtomicClass,"GetString","VS_CHAR *GetString(VS_CHAR *Para);","0DA48468-A90E-4351-BB38-7BDD520451FE",False,False);
	Service._CreateAtomicFunctionSimple(AtomicClass,"GetPkg","VS_PARAPKGPTR GetPkg(VS_PARAPKGPTR Para);","9C8BFB8F-6C48-4e32-A89A-D41DE3A9627E",False,False);
	Service._CreateAtomicModule("TestModule",0,"0E63CE93-7C1C-4a41-857A-5824E1482023")
	Service._Save("..\\..\\script");

	print("save service to ..\\..\\script")

	SrvGroup._ClearService()
	libstarpy._ModuleExit()
```

14.2.3.1.2 Exmport C code skeleton

1. Write config file servicecfg.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<ExportModuleInfo  ExportModuleDir="project">
<TestModule>
   <TestClass/>
</TestModule>
</ExportModuleInfo>
```

2. Generate code skeleton

Run : star2c RemoteCallServer 123 servicecfg.xml
in directory project, header files and source files will be created.

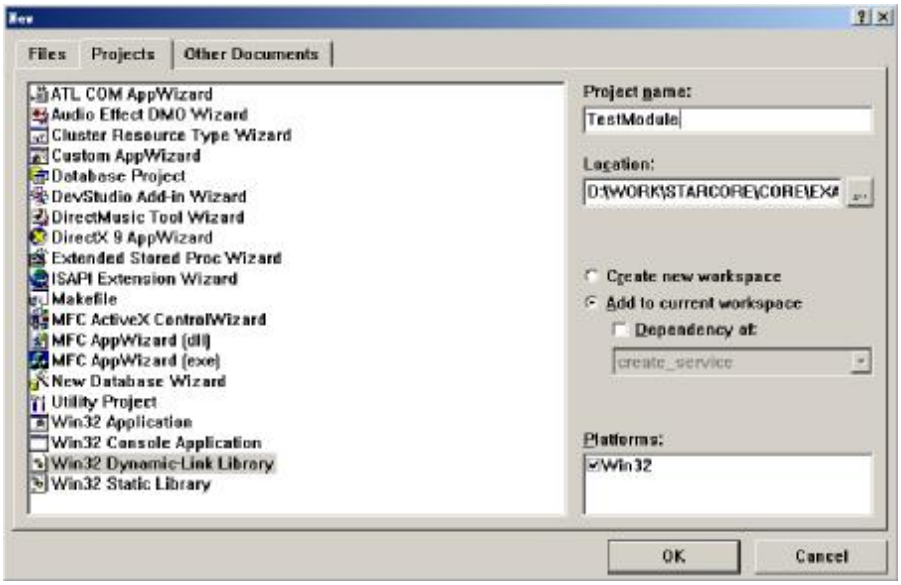
14.2.3.1.3 Create module

Module is share library.

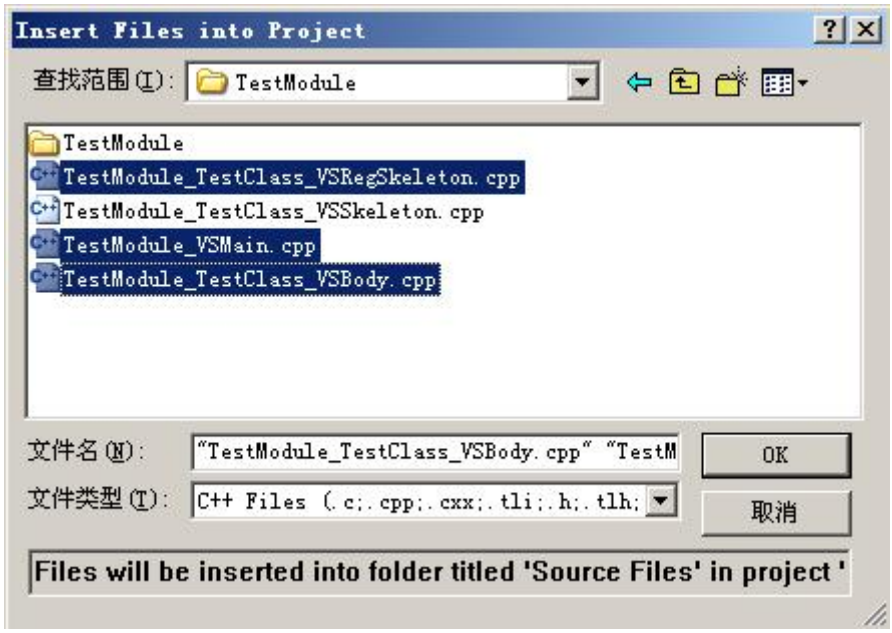
14.2.3.1.3.1 Win32

14.2.3.1.3.1.1 Create project(VC6)

Create new project:



Project name should be same with module name defined in the service. It is case sensitive.



Add skeleton file into the project. TestModule_TestClass_VSSkeleton.cpp is skeleton of the class TestClass,which will be changed to add new code. To avoid being overlay, here change its name to TestModule_TestClass_VSBody.cpp.

The project also needs UUID define file: RemoteCallServer_UUIDDef.cpp

14.2.3.1.3.1.2Edit source code

Open TestModule_TestClass_VSBody.cpp,edit code,as follows:

```
/*-------------------------------------------------*/
/*VirtualSociety System ServiceModuleTemplate Main File*/
/*CreateBy SRPLab               */
/*CreateDate: 2010-11-15  */
/*-------------------------------------------------*/
#include "RemoteCallServer_VSHeader.H"

VS_INT32 SRPAPI TestClass_GetNumber(void *Object,VS_INT32 Para)
{
        printf( "Remote Call Number [%d]\n ",Para);
        return Para+1;
}

VS_CHAR * SRPAPI TestClass_GetString(void *Object,VS_CHAR * Para)
{
        static VS_CHAR CharBuf[128];

    printf( "Remote Call String [%s]\n",Para);
        sprintf(CharBuf,"%sasdfsaf",Para);
    return CharBuf;
}

VS_PARAPKGPTR ParaPkgPtr = NULL;

VS_PARAPKGPTR SRPAPI TestClass_GetPkg(void *Object,VS_PARAPKGPTR Para)
{
        if( ParaPkgPtr == NULL )
                ParaPkgPtr = pSRP -> GetParaPkgInterface();
    printf( "Remote Call Pkg [%d]",Para ->GetInt(0));
        ParaPkgPtr ->Clear();
        ParaPkgPtr ->InsertStr(0,"asdfsaf");
    return ParaPkgPtr;
}
```

14.2.3.1.3.1.3Compile

After compile, TestModule.DLL will be generated.

14.2.3.1.3.2 linux

Makefile.ori has been created in directory of the module when create skeleton files.

You can change it to Makefile and change as follows:

1. MODULE_CXXSRCS := TestModule_TestClass_VSBody.cpp

TestModule_TestClass_VSRegSkeleton.cpp TestModule_VSMain.cpp ../RemoteCallServer_UUIDDef.cpp

MODULE_CSRCS :=

2. Modify output path, for example: OBJS_PATH = ../../../RemoteCallServer

Run make

14.2.3.2Using starcore service

Examples in directoryexamples\comm.basic\remotecall.service

14.2.3.2.1 Call by C++

14.2.3.2.1.1 Win32

####### 14.2.3.2.1.1.1Create Console project(VC6)

skip

####### 14.2.3.2.1.1.2Create and edit source file

1. Generate RemoteCallServer service header file

Into directory remotecall.service\c

Run:star2h ..\..\service\script\RemoteCallServer . ,then in current directory, the following files are generated.

RemoteCallServer.h

RemoteCallServer_UUIDDef.cpp

RemoteCallServer_VSClass.cpp

RemoteCallServer_VSClass.H

RemoteCallServer_VSDHeader.H

Add RemoteCallServer_UUIDDef.cpp to the project

Create file test_server.cpp,add to project,

```
#include " RemoteCallServer_VSDHeader.h"
#include "vsopenapi.h"

static VS_ULONG MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_ULONG wParam, VS_ULONG
lParam, VS_BOOL &IsProcessed, VS_ULONG Para )
{
   switch( uMsg ){
   case MSG_VSDISPMSG :
       case MSG_VSDISPLUAMSG :
              printf("[core]%s\n",(VS_CHAR *)wParam);
              break;
   case MSG_DISPMSG :
       case MSG_DISPLUAMSG :
              printf("%s\n",(VS_CHAR *)wParam);
      break;
       case MSG_EXIT :
              break;
   }
   return 0;
```

```
}

//------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
	VS_CORESIMPLECONTEXT Context;
   class ClassOfSRPInterface *SRPInterface;
	VS_UUID ClassID;
	void *Object;

	SRPInterface =
VSCore_InitSimple(&Context,"testserver","123",3008,0,MsgCallBack,0,"..\\..\\service\\script\\RemoteCallServer",NULL);
	if( SRPInterface == NULL ){
		printf("init starcore fail\n");
		return -1;
	}
	SRPInterface ->CreateSysRootItem( "TestItem","",NULL,NULL );
	SRPInterface ->ActiveSysRootItem( "TestItem" );
	printf("create TestObject for remotecall..\n");
	SRPInterface -> GetID(SRPInterface ->GetObjectEx(NULL,"TestClass"),&ClassID);
	Object = SRPInterface ->MallocGlobalObject(SRPInterface->GetSysRootItem("TestItem"),0,&ClassID,0,NULL,0);
	SRPInterface ->SetName(Object,"TestObject");

	printf("finish,enter message loop..\n");
	while( 1 ){
		VS_INT32 Ch;
		Ch = vs_kbhit();
		if( Ch == 27 )
			break;
		Context.VSControlInterface -> SRPDispatch(VS_FALSE);
	}
	VSCore_TermSimple(&Context);
	return 0;
}
```

14.2.3.2.1.2 linux

Write Makefile

```
#*******************************************************************************
#
# Makefile for StarCore.
# www.srplab.com
#*******************************************************************************
DEBUG		:= YES
PROFILE		:= NO
#*******************************************************************************
CC	:= gcc
CXX	:= g++
LD	:= g++
AR	:= ar
RANLIB := ranlib

DEBUG_CFLAGS	:= -Wall -Wno-format -g -DDEBUG -DENV_LINUX
RELEASE_CFLAGS	:= -Wall -Wno-unknown-pragmas -Wno-format -O3 -DENV_LINUX

LIBS := -ldl -lpthread -lrt
EXTRA_LIBS := /usr/lib/libstarlib.a /usr/lib/libuuid.a

DEBUG_CXXFLAGS	:= ${DEBUG_CFLAGS}
RELEASE_CXXFLAGS := ${RELEASE_CFLAGS}

DEBUG_LDFLAGS	:= -g
RELEASE_LDFLAGS	:=
```

```
ifeq (YES, ${DEBUG})
  CFLAGS      := ${DEBUG_CFLAGS}
  CXXFLAGS    := ${DEBUG_CXXFLAGS}
  LDFLAGS     := ${DEBUG_LDFLAGS}
else
  CFLAGS      := ${RELEASE_CFLAGS}
  CXXFLAGS    := ${RELEASE_CXXFLAGS}
  LDFLAGS     := ${RELEASE_LDFLAGS}
endif

ifeq (YES, ${PROFILE})
  CFLAGS   := ${CFLAGS} -pg -O3
  CXXFLAGS := ${CXXFLAGS} -pg -O3
  LDFLAGS  := ${LDFLAGS} -pg
endif

#*******************************************************************************
# Makefile code common to all platforms
#*******************************************************************************

CFLAGS   := ${CFLAGS}   ${DEFS}
CXXFLAGS := ${CXXFLAGS} ${DEFS}

#*******************************************************************************
# include source and paths
#*******************************************************************************

INCS_T := /usr/include/starcore
INCS   = $(addprefix -I,$(INCS_T))

TEST_SERVER_REMOTECALLSERVER_CXXSRCS := test_server_RemoteCallServer.cpp

#*******************************************************************************
TEST_SERVER_REMOTECALLSERVER_CXXOBJS :=
$(TEST_SERVER_REMOTECALLSERVER_CXXSRCS:%.cpp=%.o)

#*******************************************************************************
CXXOBJS := ${TEST_SERVER_REMOTECALLSERVER_CXXOBJS}
COBJS :=

EXEC_TEST_SERVER_REMOTECALLSERVER_OBJS := ${TEST_SERVER_REMOTECALLSERVER_CXXOBJS}

#*******************************************************************************
# Targets of the build
#*******************************************************************************
EXEC_TEST_SERVER_REMOTECALLSERVER := test_server_RemoteCallServer_linux

all: ${EXEC_TEST_SERVER_REMOTECALLSERVER}

#*******************************************************************************
# Output
#*******************************************************************************

${EXEC_TEST_SERVER_REMOTECALLSERVER}: ${TEST_SERVER_REMOTECALLSERVER_CXXOBJS}
	${LD} -o $@ ${LDFLAGS} ${TEST_SERVER_REMOTECALLSERVER_CXXOBJS} ${LIBS} ${EXTRA_LIBS}

#*******************************************************************************
# common rules
#*******************************************************************************

${CXXOBJS} :
	${CXX} ${CXXFLAGS} ${INCS} $< -o $@ -c $*.cpp

${COBJS} :
	${CC} ${CFLAGS} ${INCS} -o $@ -c $*.c
```

```
dist:
	bash makedistlinux

clean:
	-rm -f core ${CXXOBJS} ${COBJS} ${EXEC_TEST_SERVER_REMOTECALLSERVER}

depend:
	#makedepend ${INCS} ${SRCS}
```

14.2.3.2.2 Call by LUA

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",3008) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
--Create service
load depended service
SrvGroup:_ImportServiceWithPath("..\\..\\service\\script","RemoteCallServer",VS_TRUE)
SrvGroup:_CreateService( "","testserver", "123",5,0,0,0, 0,0,"B07427AF-3C8B-4e88-9F06-535831EF46EF" )
Service = SrvGroup:_GetService("root","123")
create service item
Service:_CreateSysRootItem("TestItem","")
active service item
Service:_ActiveSysRootItem("TestItem")
SrvItem = Service:_GetSysRootItem( "TestItem" )

a = Service.TestClass:_NewGlobal(SrvItem)
a._Name = "TestObject"

print( "Server Start ok....")
Message loop
function ExitProc()
   if libstarcore._KeyPress() == 27 then
      return true
   end
   return false
end

libstarcore._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

14.2.3.2.3 Call by python

```
import sys
import libstarpy
initstarcore(cle)
libstarpy._InitCore(True,True,False,True,"",0,"",3008)
get service group 0, and create service
SrvGroup = libstarpy._GetSrvGroup()
#--create service
load depended service
SrvGroup._ImportServiceWithPath("..\\..\\service\\script","RemoteCallServer",True)
SrvGroup._CreateService( "","testserver", "123",5,0,0,0,0,0,"B07427AF-3C8B-4e88-9F06-535831EF46EF" )
```

```
Service = SrvGroup._GetService("root","123")
create service item
Service._CreateSysRootItem("TestItem","")
active service item
Service._ActiveSysRootItem("TestItem")
SrvItem = Service._GetSysRootItem( "TestItem" )

a = Service.TestClass._NewGlobal(SrvItem)
a._Name = "TestObject"

#--
print( "Server Start ok....")
Message loop
def ExitProc() :
    if libstarpy._KeyPress() == 27 :
        return True
    return False

libstarpy._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

14.3 Remotecall-complicate data type

In Remotecall, complicate data can be delivered by VSTYPE_OBJPTR

In WebService, complicate data can be delivered by struct or VSTYPE_OBJPTR.

Data types supported by object is little more than struct, for example, it supports variable length.

Data types supported list below:

For object attribute:

VSTYPE_BOOL :
VSTYPE_INT8 :
VSTYPE_UINT8 :
VSTYPE_INT16 :
VSTYPE_UINT16 :
VSTYPE_INT32 :
VSTYPE_UINT32 :
VSTYPE_FLOAT :
VSTYPE_LONG :
VSTYPE_ULONG :
VSTYPE_VSTRING :
VSTYPE_STRUCT :
VSTYPE_CHAR :
VSTYPE_COLOR :
VSTYPE_RECT :
VSTYPE_FONT :
VSTYPE_TIME :
VSTYPE_UUID :
VSTYPE_STATICID :

For struct attribute:

VSTYPE_BOOL :
VSTYPE_INT8 :

VSTYPE_UINT8 :
VSTYPE_INT16 :
VSTYPE_UINT16 :
VSTYPE_INT32 :
VSTYPE_UINT32 :
VSTYPE_FLOAT :
VSTYPE_LONG :
VSTYPE_ULONG :
VSTYPE_CHAR :
VSTYPE_COLOR :
VSTYPE_RECT :
VSTYPE_FONT :
VSTYPE_TIME :
VSTYPE_UUID :

Maping between data type and xml

VSTYPE_BOOL	:	xsd:boolean
VSTYPE_INT8	:	xsd:byte
VSTYPE_UINT8	:	xsd:unsignedByte
VS_INT16	:	xsd:short
VSTYPE_UINT16	:	xsd:unsignedShort
VSTYPE_INT32	:	xsd:int
VSTYPE_UINT32	:	xsd:unsignedInt
VSTYPE_FLOAT	:	xsd:float
VSTYPE_LONG	:	xsd:long
VSTYPE_ULONG	:	xsd:unsignedLong
VSTYPE_LONGHEX	:	xsd:long
VSTYPE_ULONGHEX	:	xsd:unsignedLong
VSTYPE_VSTRING	:	xsd:string
VSTYPE_COLOR	:	xsd:unsignedLong
VSTYPE_RECT	:	xsd:string "left,top,right,bottom"
VSTYPE_FONT	:	xsd:string "height,size,charset,style,name"
VSTYPE_TIME	:	xsd:dateTime
VSTYPE_CHAR	:	xsd:string
VSTYPE_UUID	:	xsd:string
VSTYPE_STATICID	:	xsd:unsignedLong
VSTYPE_CHARPTR	:	xsd:string

14.3.1 Create server side application

14.3.1.1C

Examples in directoryexamples\comm.advanced\remotecall.c

14.3.1.1.1 Win32

14.3.1.1.1.1 Create project(VC6)

skip.

14.3.1.1.1.2 Create and edit source file

Create source file test_server and add to project,

```
#include "vsopenapi.h"
extern "C"{
	#include "vs_shell.h"
};
```

```
//-------------------------------------------------------------------------------
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
static class ClassOfSRPInterface *SRPInterface = NULL;
Callback function, to display some information
static VS_ULONG MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_ULONG wParam, VS_ULONG
lParam, VS_BOOL &IsProcessed, VS_ULONG Para )
{
   switch( uMsg ){
   case MSG_VSDISPMSG :
       case MSG_VSDISPLUAMSG :
              printf("[core]%s\n",(VS_CHAR *)wParam);
              break;
   case MSG_DISPMSG :
       case MSG_DISPLUAMSG :
              printf("%s\n",(VS_CHAR *)wParam);
      break;
       case MSG_EXIT :
              break;
   }
   return 0;
}

//-------------------------------------------------------------------------------
//--Complex Data Types
//--include struct,subobject types
#pragma pack(4)

define struct
struct StructOfParaStruct{
       VS_INT32 Para1;
       VS_FLOAT Para2;
};

struct StructOfParaObject{
       VS_INT32 Para1;
       VS_UUID Para2;
       VS_FLOAT Para3;
       struct StructOfParaStruct Para4;
       VS_VSTRING Para5;
};
#pragma pack()

struct StructOfParaObject *LocalRetObject;


Define function being called, input and output are object
static void *GetRemoteObject(void *Object,void *ParaObject)
{
       struct StructOfParaObject *RequestPara;

       RequestPara = (struct StructOfParaObject *)ParaObject;
       printf("Para1 = %d\n",RequestPara ->Para1);
       printf("Para2 = %s\n",SRPInterface->UuidToString(&RequestPara ->Para2));
       printf("Para3 = %f\n",RequestPara ->Para3);
       printf("Para4.Para1 = %d\n",RequestPara ->Para4.Para1);
       printf("Para4.Para2 = %f\n",RequestPara ->Para4.Para2);
       printf("Para5 = %s\n",RequestPara ->Para5.Buf);

       LocalRetObject ->Para1 = 123 + RequestPara ->Para1;
       SRPInterface -> StringToUuid("1E2929C6-7DDA-468f-BBAD-E303A1B3C826",&LocalRetObject ->Para2);
       LocalRetObject ->Para3 = 456.0 + RequestPara ->Para3;
       LocalRetObject ->Para4.Para1 = 234 + RequestPara ->Para4.Para1;
       LocalRetObject ->Para4.Para2 = 567.0 + RequestPara ->Para4.Para2;
       SRPInterface ->DupVString( &(VS_VSTRING)("server return"), &LocalRetObject ->Para5 );

       return LocalRetObject;
```

```
}

//-------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
        VS_UUID ServiceID,ClassID,RetClassID;
        void *AtomicClass,*GetObject_AtomicFunction,*Object;
        VS_CHAR *ErrorInfo;

        SRPControlInterface = NULL;
        BasicSRPInterface = NULL;
    //-----------------------------------------------------------------------
        //--init star core
callback function, to display information
        VSCore_RegisterCallBackInfo(MsgCallBack,0);
        VSCore_Init( true, true, "", 0, "", 3008,NULL);

        printf("init starcore success\n");

        SRPControlInterface = VSCore_QueryControlInterface();
get basic service interface
    BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);
create service
        BasicSRPInterface ->StringToUuid("F0611A16-BFAA-4d0b-803F-807EC63BD265",&ServiceID);
        BasicSRPInterface ->CreateService("","RemoteCallServer",&ServiceID,"123",0,0,0,0,0,0);
get service interface
        SRPInterface = BasicSRPInterface ->GetSRPInterface("RemoteCallServer","root","123");
create service item
        SRPInterface ->CreateSysRootItem("TestItem","",NULL,NULL);
active service item
        SRPInterface ->ActiveSysRootItem( "TestItem" );
//---Create Parameter Class
create struct
        SRPInterface ->CreateAtomicStructSimple("ParaStruct","VS_INT32 Para1;VS_FLOAT Para2;",NULL,&ErrorInfo);
create atomic object class
        AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","ParaObject","VS_INT32 Para1;VS_UUID
Para2;VS_FLOAT Para3;struct ParaStruct Para4;VS_VSTRING Para5;", NULL,&ErrorInfo);
get atomic object class ID,which is used to create instance
        SRPInterface -> GetAtomicID(AtomicClass,&RetClassID);

        //---Create Atomic Class, for define function, no attribute
create atomic object class
        AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClass",NULL, NULL,&ErrorInfo);
create function of class
        GetObject_AtomicFunction = SRPInterface -
>CreateAtomicFunctionSimple(AtomicClass,"GetRemoteObject","VS_OBJPTR GetNumber(VS_OBJPTR ParaObject);",
NULL,&ErrorInfo,VS_FALSE, VS_FALSE);
//---Set Function Address
   set function address, which should be called after all functions are created finish
        SRPInterface -> SetAtomicFunction(GetObject_AtomicFunction,(void *)GetRemoteObject);

        //---Create RetObject, and set value
        LocalRetObject = (struct StructOfParaObject *)SRPInterface -> MallocObjectL(&RetClassID,NULL,0);

        printf("create TestObject for remotecall..\n");
get atomic object class ID,which is used to create instance
        SRPInterface ->GetAtomicID(AtomicClass,&ClassID);
create globalobject, which will by sync to client
        Object = SRPInterface ->MallocGlobalObject(SRPInterface->GetSysRootItem("TestItem"),0,&ClassID,0,NULL,0);
set object name, then can be find object by name
        SRPInterface ->SetName(Object,"TestObject");

        printf("finish,enter message loop..\n");
        while( 1 ){
ESC is pressed? if so, exit
                VS_INT32 Ch;
                Ch = vs_kbhit();
```

```
            if( Ch == 27 )
                break;
Message loop, should be called in main loop to drive starcore
            if( SRPControlInterface -> SRPDispatch(VS_FALSE) == VS_FALSE ){
                SRPControlInterface -> SRPIdle();
                SRPControlInterface -> SRPDispatch(VS_TRUE);
            }
        }
        SRPInterface -> Release();
        SRPControlInterface ->Release();
        BasicSRPInterface ->Release();
        VSCore_Term();
        return 0;
}
```

14.3.1.1.1.3 Compile and run

test_server

14.3.1.2 lua

Examples in directoryexamples\comm.advanced\remotecall.lua

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",3008) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
--create service
SrvGroup:_CreateService( "","RemoteCallServer", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )
Service = SrvGroup:_GetService("root","123")

--create service item(object group)
Service:_CreateSysRootItem("TestItem","")
active service item
Service:_ActiveSysRootItem("TestItem")
SrvItem = Service:_GetSysRootItem( "TestItem" )

--Create Parameter Object
Service:_CreateAtomicStructSimple("ParaStruct","VS_INT32 Para1;VS_FLOAT Para2;", "");
create atomic object class
Service:_CreateAtomicObjectSimple("TestItem","ParaObject","VS_INT32 Para1;VS_UUID Para2;VS_FLOAT Para3;struct
ParaStruct Para4;VS_VSTRING Para5;", "");

print(Service,SrvItem)
a = Service:_NewGlobal(SrvItem)
a._Name = "TestObject"
a.___Value = "Global Attribute"

b = Service.ParaObject:_New()
function a:GetRemoteObject( para )
   para:_V()
   b.Para1 = 123 + para.Para1
   b.Para2 = "1E2929C6-7DDA-468f-BBAD-E303A1B3C826"
   b.Para3 = 456.0 + para.Para3
   b.Para4 = {para.Para4.Para1 + 234,para.Para4.Para2+567.0}
   b.Para5 = "server return"
   return b
end
```

```
print( "Server Start ok....")
Message loop
function ExitProc()
    if libstarcore._KeyPress() == 27 then
        return true
    end
    return false
end

libstarcore._MsgLoop( ExitProc )

print("Exit...")
exit, clear service and starcore
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

14.3.1.3python

Examples in directoryexamples\comm.advanced\remotecall.python

```
import sys
import libstarpy
initstarcore(cle)

libstarpy._InitCore(True,True,False,True,"",0,"",3008)
get service group 0, and create service
SrvGroup = libstarpy._GetSrvGroup()
#--create service
SrvGroup._CreateService( "","RemoteCallServer", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )
Service = SrvGroup._GetService("root","123")

#--create service item(object group)
Service._CreateSysRootItem("TestItem","")
active service item
Service._ActiveSysRootItem("TestItem")
SrvItem = Service._GetSysRootItem( "TestItem" )

#--Create Parameter Object
Service._CreateAtomicStructSimple("ParaStruct","VS_INT32 Para1;VS_FLOAT Para2;", "");
create atomic object class
Service._CreateAtomicObjectSimple("TestItem","ParaObject","VS_INT32 Para1;VS_UUID Para2;VS_FLOAT Para3;struct
ParaStruct Para4;VS_VSTRING Para5;", "");

a = Service._NewGlobal(SrvItem)
a._Name = "TestObject"
a.___Value = "Global Attribute"

b = Service.ParaObject._New()
def a_GetRemoteObject( self, para ) :
    para._V()
    b.Para1 = 123 + para.Para1
    b.Para2 = "1E2929C6-7DDA-468f-BBAD-E303A1B3C826"
    b.Para3 = 456.0 + para.Para3
    b.Para4 = (para.Para4.Para1 + 234,para.Para4.Para2+567.0)
    b.Para5 = "server return"
    return b
a.GetRemoteObject = a_GetRemoteObject

#--
print( "Server Start ok....")
Message loop
def ExitProc() :
    if libstarpy._KeyPress() == 27 :
        return True
```

```
   return False

libstarpy._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

14.3.2 Create client side application

14.3.2.1Win32

Examples in directoryexamples\comm.advanced\remotecall.c

14.3.2.1.1 Create project(VC6)

See above

14.3.2.1.2 Create and edit source file

Create source file test_client and add it to project,

```
#include "vsopenapi.h"
extern "C"{
	#include "vs_shell.h"
};

//-----------------------------------------------------------------------------
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
static class ClassOfSRPInterface *SRPInterface = NULL;

#pragma pack(4)

struct StructOfParaStruct{
	VS_INT32 Para1;
	VS_FLOAT Para2;
};

struct StructOfParaObject{
	VS_INT32 Para1;
	VS_UUID Para2;
	VS_FLOAT Para3;
	struct StructOfParaStruct Para4;
	VS_VSTRING Para5;
};
#pragma pack()

struct StructOfParaObject *LocalRequestObject;

//-----------------------------------------------------------------------------
int main(int argc, char* argv[])
{
	VS_UUID FunctionID,ParaObjectID;
	void *SysRootItem,*Object;
	VS_ULONG RetCode;

	if( argc < 2 ){
		printf("useage test_client serverip\n");
		return -1;
	}
	SRPControlInterface = NULL;
	BasicSRPInterface = NULL;
	//--init star core
   VSCore_Init( true, true, "", 0, "", 3008,NULL);

	printf("init starcore success\n");
```

```
        SRPControlInterface = VSCore_QueryControlInterface();
Client create service group, beacuse service group 0 created by default can be only used as server
    BasicSRPInterface = SRPControlInterface ->CreateBasicInterface(1,VS_CLIENT_USER);
        if( BasicSRPInterface ->SConnect("",argv[1],3008,NULL,NULL,NULL) == 0 ){
                printf("Fail to connect to server\n");
                SRPControlInterface ->Release();
                BasicSRPInterface ->Release();
                VSCore_Term();
                return 0;
        }
        BasicSRPInterface ->WaitServiceSync(0);
        printf( "Success To Connect...\n" );
get service interface
        SRPInterface = BasicSRPInterface ->GetSRPInterface(NULL,NULL,NULL);
        SysRootItem = SRPInterface ->GetSysRootItem("TestItem");
wait service item to sync
        SRPInterface ->WaitSysRootItemSync(SysRootItem);
get global object by name at client
        Object = SRPInterface ->GetObjectEx(NULL,"ParaObject");
        SRPInterface ->GetID(Object,&ParaObjectID);
        LocalRequestObject = (struct StructOfParaObject *)SRPInterface ->MallocObjectL(&ParaObjectID,NULL,0);
        LocalRequestObject ->Para1 = 123;
        SRPInterface -> StringToUuid("1E2929C6-7DDA-468f-BBAD-E303A1B3C826",&LocalRequestObject ->Para2);
        LocalRequestObject ->Para3 = 456.0;
        LocalRequestObject ->Para4.Para1 = 234;
        LocalRequestObject ->Para4.Para2 = 567.0;
        SRPInterface ->DupVString( &(VS_VSTRING)("client request"), &LocalRequestObject ->Para5 );
get global object by name at client
        Object = SRPInterface ->GetObjectEx(NULL,"TestObject");
get function ID by function name, and init the remotecall
        SRPInterface ->GetFunctionID(Object,"GetRemoteObject",&FunctionID);

        struct StructOfParaObject *RetObject;
        RetObject = (struct StructOfParaObject *)SRPInterface-
>SRemoteCall(0,0,&RetCode,Object,&FunctionID,LocalRequestObject);
        if( RetObject != NULL ){
                printf("Para1 = %d\n",RetObject ->Para1);
                printf("Para2 = %s\n",SRPInterface->UuidToString(&RetObject ->Para2));
                printf("Para3 = %f\n",RetObject ->Para3);
                printf("Para4.Para1 = %d\n",RetObject ->Para4.Para1);
                printf("Para4.Para2 = %f\n",RetObject ->Para4.Para2);
                printf("Para5 = %s\n",RetObject ->Para5.Buf);
        }

        SRPControlInterface ->Release();
        BasicSRPInterface ->Release();
        VSCore_Term();
        return 0;
}
```

14.3.2.1.3 Compile and run

test_client

14.3.2.2 lua

Examples in directoryexamples\comm.advanced\remotecall.lua

```
require "libstarcore"
initstarcore(cle)
```

```
if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
    return
end
SrvGroup = libstarcore._CreateSrvGroup(1,libstarcore.VS_CLIENT_USER);

print(SrvGroup,libstarcore.VS_CLIENT_USER);
ret = SrvGroup:_SConnect("","127.0.0.1",3008,"","")
if ret == 0 then
    print( "Fail To Connect..." )
    SrvGroup:_ClearService()
    libstarcore._ModuleExit()
    return
end
Wait service to sync, and then get service item, and wait service item to sync, and the can get object
SrvGroup:_WaitServiceSync()

print( "Success To Connect..." )

Service = SrvGroup:_GetService("root","123")
active service item
Service:_ActiveSysRootItem("TestItem")
SrvItem = Service:_GetSysRootItem("TestItem")
wait service item to sync
SrvItem:_WaitSync()

b = Service.ParaObject:_New()
b.Para1 = 123
b.Para2 = "1E2929C6-7DDA-468f-BBAD-E303A1B3C826"
b.Para3 = 456.0
b.Para4 = {123,456.0}
b.Para5 = "client request"

RetCode,RetValue = Service.TestObject:_SRemoteCall(0,0,"GetRemoteObject",b)
RetValue:_V()
exit, clear service and starcore
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

14.3.2.3python

Examples in directoryexamples\comm.advanced\remotecall.python

```
import sys
import libstarpy
initstarcore(cle)
libstarpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = libstarpy._CreateSrvGroup(1,libstarpy.VS_CLIENT_USER);

ret = SrvGroup._SConnect("","127.0.0.1",3008,"","")
if ret == 0 :
    print( "Fail To Connect..." )
    SrvGroup._ClearService()
    libstarpy._ModuleExit()
    raise Exception("")
Wait service to sync, and then get service item, and wait service item to sync, and the can get object
SrvGroup._WaitServiceSync()

print( "Success To Connect..." )

Service = SrvGroup._GetService("","")
active service item
Service._ActiveSysRootItem("TestItem")
SrvItem = Service._GetSysRootItem("TestItem")
```

```
wait service item to sync
SrvItem._WaitSync()

b = Service.ParaObject._New()
b.Para1 = 123
b.Para2 = "1E2929C6-7DDA-468f-BBAD-E303A1B3C826"
b.Para3 = 456.0
b.Para4 = (123,456.0)
b.Para5 = "client request"

RetCode,RetValue = Service.TestObject._SRemoteCall(0,0,"GetRemoteObject",b)
RetValue._V()
exit, clear service and starcore
SrvGroup._ClearServiceEx()
libstarpy._ModuleExit()
```

14.3.3 Create and ust stand alone starcore service

14.3.3.1Create starcore service

Examples in directoryexamples\comm.advanced\service

14.3.3.1.1 create data file of service starcore

14.3.3.1.1.1 C

14.3.3.1.1.1Win32

14.3.3.1.1.1.1 Create project(VC6)

skip

14.3.3.1.1.1.2 edit source code

create new file,create_service.cpp,

```
#include "vsopenapi.h"

//------------------------------------------------------------------------------
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
static class ClassOfSRPInterface *SRPInterface = NULL;


static VS_ULONG MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_ULONG wParam, VS_ULONG
lParam, VS_BOOL &IsProcessed, VS_ULONG Para )
{
   switch( uMsg ){
   case MSG_VSDISPMSG :
       case MSG_VSDISPLUAMSG :
            printf("[core]%s\n",(VS_CHAR *)wParam);
            break;
   case MSG_DISPMSG :
       case MSG_DISPLUAMSG :
            printf("%s\n",(VS_CHAR *)wParam);
      break;
       case MSG_EXIT :
            break;
   }
```

```
    return 0;
}

//-------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
    VS_UUID ServiceID;
    void *AtomicClass;
    VS_CHAR *ErrorInfo;

    SRPControlInterface = NULL;
    BasicSRPInterface = NULL;
  //-------------------------------------------------------------------------
    //--init star core
    VSCore_RegisterCallBackInfo(MsgCallBack,0);
  VSCore_Init( true, true, "", 0, "", 3008,NULL);

    printf("init starcore success\n");

    SRPControlInterface = VSCore_QueryControlInterface();
  BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);

    BasicSRPInterface ->StringToUuid("5D0465E1-4203-4d44-9860-8B56C4790BC2",&ServiceID);
    BasicSRPInterface ->CreateService("","RemoteCallServer",&ServiceID,"123",0,0,0,0,0,0);
    SRPInterface = BasicSRPInterface ->GetSRPInterface("RemoteCallServer","root","123");
    SRPInterface ->CreateSysRootItem("TestItem","",NULL,NULL);
    SRPInterface ->ActiveSysRootItem( "TestItem" );
  //---Create Parameter Class
    SRPInterface ->CreateAtomicStructSimple("ParaStruct","VS_INT32 Para1;VS_FLOAT
Para2;",_UUIDPTR("e65fa0b1-5684-429c-8075-4ca2ee685e6d"),&ErrorInfo);
    AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","ParaObject","VS_INT32 Para1;VS_UUID
Para2;VS_FLOAT Para3;struct ParaStruct Para4;VS_VSTRING Para5;",_UUIDPTR("f85b85b9-8109-4c02-b6f6-
4ad23f1cba38"),&ErrorInfo);
    //---Create Atomic Class, for define function, no attribute
    AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClass",NULL,_UUIDPTR("07bdb329-
e9a4-41b4-b79d-10132210cd44"),&ErrorInfo);
    SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetRemoteObject","struct ParaObject
*GetRemoteObject(struct ParaObject *Para);",_UUIDPTR("2ef3218c-31e8-4d45-8002-
b8b29a91d837"),&ErrorInfo,VS_FALSE,VS_FALSE);

    SRPInterface -> CreateAtomicModule("TestModule",VSMODULE_SERVER_SERVER |
VSMODULE_SERVER_USER,_UUIDPTR("a0164199-f3bd-4ad3-83ef-33d0b0939687"));

    SRPInterface -> SaveService("..\\..\\script");

    printf("save service to ..\\..\\script \n");

    SRPControlInterface ->Release();
    BasicSRPInterface ->Release();
    VSCore_Term();
    return 0;
}
```

14.3.3.1.1.1.3 Compile and run

Run : create_RemoteCallServe

14.3.3.2Export skeleton file

1. write config file servicecfg.xml

<?xml version="1.0" encoding="utf-8" ?>

```
<ExportModuleInfo  ExportModuleDir="project">
<TestModule>
   <TestClass/>
</TestModule>
</ExportModuleInfo>
```

2. generate skeleton

Into directory script

Run : star2c RemoteCallServer 123 RemoteCallServercfg.xml

In directory project, header and skeleton file will be generated.

14.3.3.3create module

module is share library

14.3.3.3.1 Win32

14.3.3.3.1.1 Create project(VC6)

skip

14.3.3.3.1.2 edit source code

Open TestModule_TestClass_VSBody.cpp,edit source code, as follows:

```
/*-------------------------------------------------------*/
/*VirtualSociety System ServiceModuleTemplate Main File*/
/*CreateBy SRPLab                */
/*CreateDate: 2010-11-15  */
/*-------------------------------------------------------*/
#include "RemoteCallServer_VSHeader.H"

VS_OBJPTR SRPAPI TestClass_GetRemoteObject(void *Object,VS_OBJPTR Para)
{
   struct StructOfParaObject *RequestPara,*LocalRetObject;

       LocalRetObject = (struct StructOfParaObject *)pSRP ->MallocObjectL(&VSOBJID_ParaObject,NULL,0);
       RequestPara = (struct StructOfParaObject *)Para;
       printf("Para1 = %d\n",RequestPara ->Para1);
       printf("Para2 = %s\n",pSRP->UuidToString(&RequestPara ->Para2));
       printf("Para3 = %f\n",RequestPara ->Para3);
       printf("Para4.Para1 = %d\n",RequestPara ->Para4.Para1);
       printf("Para4.Para2 = %f\n",RequestPara ->Para4.Para2);
       printf("Para5 = %s\n",RequestPara ->Para5.Buf);

       LocalRetObject ->Para1 = 123 + RequestPara ->Para1;
       pSRP -> StringToUuid("1E2929C6-7DDA-468f-BBAD-E303A1B3C826",&LocalRetObject ->Para2);
       LocalRetObject ->Para3 = 456.0 + RequestPara ->Para3;
       LocalRetObject ->Para4.Para1 = 234 + RequestPara ->Para4.Para1;
       LocalRetObject ->Para4.Para2 = 567.0 + RequestPara ->Para4.Para2;
       pSRP ->DupVString( &(VS_VSTRING)("server return"), &LocalRetObject ->Para5 );
       pSRP ->DeferFreeObject(LocalRetObject);  //defer free, which will be freed by cle

       return LocalRetObject;
}
```

14.3.3.3.1.3 Compile

skip

14.3.4 called by LUA

need not change

14.3.5 called by Python

need not change

15 Webservice and http application

Examples in directory examples\comm.basic,include C++,lua,python ,java,c# source code.

15.1 Http&HttpServer

Examples in directoryexamples\comm.basic\ http_webserver

15.1.1 Http download

15.1.1.1C

15.1.1.1.1 Win32

15.1.1.1.1.1 Create project(VC6)

see above

15.1.1.1.1.2 Create and edit source file

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
	VS_CORESIMPLECONTEXT Context;
	class ClassOfSRPCommInterface *CommInterface;

	if( VSCore_InitSimpleEx(&Context,0,0,NULL,0,NULL) == NULL ){
		printf("init starcore fail\n");
		return -1;
	}
	printf("init starcore success\n");

	if( argc < 2 ){
		printf("Usage http_download url\n");
		return -1;
	}
	CommInterface = Context.VSControlInterface ->GetCommInterface();
	CommInterface ->FileDownLoad(argv[1],vs_file_strrchr(argv[1],'/')+1,VS_TRUE,NULL,0);
	VSCore_TermSimple(&Context);
	return 0;
}
```

15.1.1.1.1.3 Compile and run

http_download http://127.0.0.1/index.html

15.1.1.1.2 linux

Makefile:

```
#*******************************************************************************
#
# Makefile for StarCore.
# www.srplab.com
#*******************************************************************************
DEBUG         := YES
PROFILE       := NO
#*******************************************************************************
CC    := gcc
CXX   := g++
LD    := g++
AR    := ar
RANLIB := ranlib

DEBUG_CFLAGS     := -Wall -Wno-format -g -DDEBUG -DENV_LINUX
RELEASE_CFLAGS   := -Wall -Wno-unknown-pragmas -Wno-format -O3 -DENV_LINUX

LIBS := -ldl -lpthread -lrt
EXTRA_LIBS := ../../../output/linux/libstarlib.a /usr/lib/libuuid.a

DEBUG_CXXFLAGS   := ${DEBUG_CFLAGS}
RELEASE_CXXFLAGS := ${RELEASE_CFLAGS}

DEBUG_LDFLAGS    := -g
RELEASE_LDFLAGS  :=

ifeq (YES, ${DEBUG})
  CFLAGS      := ${DEBUG_CFLAGS}
  CXXFLAGS    := ${DEBUG_CXXFLAGS}
  LDFLAGS     := ${DEBUG_LDFLAGS}
else
  CFLAGS      := ${RELEASE_CFLAGS}
  CXXFLAGS    := ${RELEASE_CXXFLAGS}
  LDFLAGS     := ${RELEASE_LDFLAGS}
endif

ifeq (YES, ${PROFILE})
  CFLAGS   := ${CFLAGS} -pg -O3
  CXXFLAGS := ${CXXFLAGS} -pg -O3
  LDFLAGS  := ${LDFLAGS} -pg
endif

#*******************************************************************************
# Makefile code common to all platforms
#*******************************************************************************

CFLAGS   := ${CFLAGS}   ${DEFS}
CXXFLAGS := ${CXXFLAGS} ${DEFS}

#*******************************************************************************
# include source and paths
#*******************************************************************************

INCS_T := /usr/include/starcore
INCS   = $(addprefix -I,$(INCS_T))

HTTP_DOWNLOAD_CXXSRCS := http_download.cpp
HTTP_UPLOAD_CXXSRCS := http_upload.cpp
SIMPLE_WEBSERVER_CXXSRCS := simple_webserver.cpp
```

```
#*****************************************************************************
HTTP_DOWNLOAD_CXXOBJS := $(HTTP_DOWNLOAD_CXXSRCS:%.cpp=%.o)
HTTP_UPLOAD_CXXOBJS := $(HTTP_UPLOAD_CXXSRCS:%.cpp=%.o)
SIMPLE_WEBSERVER_CXXOBJS := $(SIMPLE_WEBSERVER_CXXSRCS:%.cpp=%.o)

#*****************************************************************************
CXXOBJS := ${HTTP_DOWNLOAD_CXXOBJS} ${HTTP_UPLOAD_CXXOBJS} ${SIMPLE_WEBSERVER_CXXOBJS}
COBJS :=

EXEC_HTTP_DOWNLOAD_OBJS := ${HTTP_DOWNLOAD_CXXOBJS}
EXEC_HTTP_UPLOAD_OBJS := ${HTTP_UPLOAD_CXXOBJS}
EXEC_SIMPLE_WEBSERVER_OBJS := ${SIMPLE_WEBSERVER_CXXOBJS}

#*****************************************************************************
# Targets of the build
#*****************************************************************************
OBJS_PATH = .

EXEC_HTTP_DOWNLOAD := ${OBJS_PATH}/http_download_linux
EXEC_HTTP_UPLOAD := ${OBJS_PATH}/http_upload_linux
EXEC_SIMPLE_WEBSERVER := ${OBJS_PATH}/simple_webserver_linux

all: ${EXEC_HTTP_DOWNLOAD} ${EXEC_HTTP_UPLOAD} ${EXEC_SIMPLE_WEBSERVER}

#*****************************************************************************
# Output
#*****************************************************************************

${EXEC_HTTP_DOWNLOAD}: ${EXEC_HTTP_DOWNLOAD_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_HTTP_DOWNLOAD_OBJS} ${LIBS} ${EXTRA_LIBS}

${EXEC_HTTP_UPLOAD}: ${EXEC_HTTP_UPLOAD_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_HTTP_UPLOAD_OBJS} ${LIBS} ${EXTRA_LIBS}

${EXEC_SIMPLE_WEBSERVER}: ${EXEC_SIMPLE_WEBSERVER_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_SIMPLE_WEBSERVER_OBJS} ${LIBS} ${EXTRA_LIBS}

#*****************************************************************************
# common rules
#*****************************************************************************

${CXXOBJS} :
	${CXX} ${CXXFLAGS} ${INCS} $< -o $@ -c $*.cpp

${COBJS} :
	${CC} ${CFLAGS} ${INCS} -o $@ -c $*.c

dist:
	bash makedistlinux

clean:
	-rm -f core ${CXXOBJS} ${COBJS} ${EXEC_HTTP_DOWNLOAD} ${EXEC_HTTP_UPLOAD}
${EXEC_SIMPLE_WEBSERVER}

depend:
	#makedepend ${INCS} ${SRCS}
```

15.1.1.2 lua

```
require "libstarcore"
Service=libstarcore._InitSimple("test","123",0,0)
SrvGroup = Service._ServiceGroup
```

```
CommInterface = SrvGroup:_NewCommInterface()

if SrvGroup._EnvInputPara ~= nil then
   SrvGroup:_RunScript("",SrvGroup._EnvInputPara[0] ,"")
end
if Url == nil then
  print("starapp -e http_download.lua?Url=\\\"http://127.0.0.1/XXX\\\" or")
  SrvGroup:_ClearService()
  libstarcore._ModuleExit()
  return
end

Pos=_strrchr(Url,'/')
if Pos == -1 then
  print("not find download filename")
  SrvGroup:_ClearService()
  libstarcore._ModuleExit()
  return
end
FileName=string.sub(Url,Pos+1)

CommInterface:_FileDownLoad(Url,FileName,true,nil)

SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Run:

starapp -e "http_download.lua?Url=\"http://127.0.0.1/zoc.rar\""

15.1.1.3python

```
import sys
if hasattr(sys,"argv") :
   if len(sys.argv) > 1 :
      Url = sys.argv[1]
import libstarpy

Service=libstarpy._InitSimple("test","123",0,0)
SrvGroup = Service._ServiceGroup

CommInterface = SrvGroup._NewCommInterface()

if SrvGroup._EnvInputPara != None :
   SrvGroup._RunScript("python",SrvGroup._EnvInputPara._Get(0) ,"")

if "Url" not in dir() or Url == "" or Url == None :
  print("starapp -e http_download.py?script=python;Url=\\\"http://127.0.0.1/XXX\\\" or")
  print("python http_download.py http://127.0.0.1/XXX")
  SrvGroup._ClearService()
  libstarpy._ModuleExit()
  raise Exception("")

Pos=libstarpy._strrchr(Url,'/')
if Pos == -1 :
  print("not find download filename")
  SrvGroup._ClearService()
  libstarpy._ModuleExit()
  raise Exception("")

FileName=Url[Pos+1:]
```

```
CommInterface._FileDownLoad(Url,FileName,True,None)

print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

starapp -e "http_download.py?script=python;Url=\"http://127.0.0.1/zoc.rar\""

15.1.2 Http upload

15.1.2.1C

15.1.2.1.1 Win32

15.1.2.1.1.1 Create project(VC6)

see above

15.1.2.1.1.2 Create and edit source file

```
#include "vsopenapi.h"

int main(int argc, char* argv[])
{
	VS_CORESIMPLECONTEXT Context;
	class ClassOfSRPCommInterface *CommInterface;

	if( VSCore_InitSimpleEx(&Context,0,0,NULL,0,NULL) == NULL ){
		printf("init starcore fail\n");
		return -1;
	}
	printf("init starcore success\n");

	if( argc < 3 ){
		printf("Usage http_upload url FileName\n");
		return -1;
	}
	CommInterface = Context.VSControlInterface ->GetCommInterface();
	CommInterface ->FileUpLoad(argv[1],argv[2],argv[2],NULL,VS_TRUE,NULL,VS_TRUE,NULL,0);
	VSCore_TermSimple(&Context);
	return 0;
}
```

15.1.2.1.1.3 Compile and run

http_upload http://127.0.0.1/upload.php XXX

Upload message format conforms to php. You can use php code to receive the upload file, as follows:

```
<?php
if ($_FILES["file"]["error"] > 0)
{
  echo "Error: " . $_FILES["file"]["error"] . "<br />";
}
else
{
  echo "Upload: " . $_FILES["file"]["name"] . "<br />";
  echo "Type: " . $_FILES["file"]["type"] . "<br />";
  echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
```

```
 move_uploaded_file($_FILES["file"]["tmp_name"],"/upload/" . $_FILES["file"]["name"]);
 echo "Stored in: " . "/upload/" . $_FILES["file"]["name"];
}
?>
```

15.1.2.1.2 linux

Write makefile(skip)

15.1.2.2 lua

```
require "libstarcore"
Service=libstarcore._InitSimple("test","123",0,0)
SrvGroup = Service._ServiceGroup

if SrvGroup._EnvInputPara ~= nil then
   SrvGroup:_RunScript("",SrvGroup._EnvInputPara[0] ,"")
end
if Url == nil or FileName == nil then
  print("starapp -e \"http_download.lua?Url=\\\"http://127.0.0.1/XXX\\\" FileName=\\\"XXX\\\"\" or")
  SrvGroup:_ClearService()
  libstarcore._ModuleExit()
  return
end

CommInterface = SrvGroup:_NewCommInterface()
CommInterface:_FileUpLoad(Url,FileName, FileName,nil,true,"",true,nil)

print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Run:

starapp -e "http_upload.lua?Url=\"http://192.168.75.1/upload_file.php\" FileName=\"zoc.rar\""

15.1.2.3python

```
import sys
if hasattr(sys,"argv") :
   if len(sys.argv) > 2 :
      Url = sys.argv[1]
      FileName = sys.argv[2]
import libstarpy

Service=libstarpy._InitSimple("test","123",0,0)
SrvGroup = Service._ServiceGroup
CommInterface = SrvGroup._NewCommInterface()

if SrvGroup._EnvInputPara != None :
   SrvGroup._RunScript("python",SrvGroup._EnvInputPara._Get(0) ,"")

if "Url" not in dir() or Url == "" or Url == None :
  print("starapp -e \"http_upload.py?script=python;Url=\\\"http://127.0.0.1/XXX\\\";FileName=\\\"XXX\\\"\" or")
  print("python http_upload.py http://127.0.0.1/XXX  FileName")
  SrvGroup._ClearService()
  libstarpy._ModuleExit()
  raise Exception("")
```

```
CommInterface._FileUpLoad(Url,FileName, FileName,"",True,"",True,None)

print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

starapp -e "http_upload.py?script=python;Url=\"http://192.168.75.1/upload_file.php\";FileName=\"zoc.rar\""

15.1.3 Simple HttpServer

Implement simple WebServer, does not support dynamic script, and only support Get and Post operation.

15.1.3.1C

15.1.3.1.1 Win32

15.1.3.1.1.1 Create project(VC6)

skip

15.1.3.1.1.2 Create and edit source file

```
#include "vsopenapi.h"
extern "C"{
	#include "vs_shell.h"
};

//------------------------------------------------------------------------------
VS_HANDLE hDllInstance;
VSCore_RegisterCallBackInfoProc RegisterCallBackInfoProc;
VSCore_InitProc VSInitProc;
VSCore_TermProc VSTermProc;
VSCore_QueryControlInterfaceProc QueryControlInterfaceProc;
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfSRPCommInterface *CommInterface = NULL;

//------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
	VS_CHAR ModuleName[512];
	VS_HANDLE MsgHandle;

	if( argc < 2 ){
		printf("Usage http_upload url FileName\n");
		return -1;
	}

	SRPControlInterface = NULL;
	CommInterface = NULL;
  //------------------------------------------------------------------------
  load library
	sprintf(ModuleName,"libstarcore%s",VS_MODULEEXT);
  hDllInstance = vs_dll_open( ModuleName );
	if( hDllInstance == NULL ){
		printf("load library [%s] error....\n",ModuleName);
		return -1;
  }
  get export functions from library
  RegisterCallBackInfoProc = (VSCore_RegisterCallBackInfoProc)vs_dll_sym( hDllInstance,
VSCORE_REGISTERCALLBACKINFO_NAME );
  VSInitProc = (VSCore_InitProc)vs_dll_sym( hDllInstance, VSCORE_INIT_NAME );
  VSTermProc = (VSCore_TermProc)vs_dll_sym( hDllInstance, VSCORE_TERM_NAME );
```

```
   QueryControlInterfaceProc = (VSCore_QueryControlInterfaceProc)vs_dll_sym( hDllInstance,
VSCORE_QUERYCONTROLINTERFACE_NAME );
        //--init star core
initstarcore(cle)
   VSInitProc( true, true, "", 0, "", 0,NULL);

        printf("init starcore success\n");
get control interface, controlinterface
        SRPControlInterface = QueryControlInterfaceProc();
get communicate interface
        CommInterface = SRPControlInterface ->GetCommInterface();
create message queue
   MsgHandle = CommInterface ->CreateMsgQueue(256,256);
create http server
        if( CommInterface ->HttpServer( MsgHandle,NULL,atoi(argv[1]),0,0,NULL,100) ==
VS_COMM_INVALIDCONNECTION ){
                printf("create webserver [%d] fail\n",atoi(argv[1]));
                CommInterface -> Release();
                SRPControlInterface ->Release();
                VSTermProc();
                vs_dll_close(hDllInstance);
                return -1;
        }
        printf("create webserver [%d] success\n",atoi(argv[1]));
        printf("finish,enter message loop..\n");
        while( 1 ){
ESC is pressed? if so, exit
                VS_INT32 Ch;
                Ch = vs_kbhit();
                if( Ch == 27 )
                        break;
                {
                        struct StructOfSRPCommMessage *CommMessage;
                        struct StructOfSRPComm_HttpOnRequest *HttpOnRequest;
                        VS_CHAR Buf[256];
Has message? it has, then the return value is not NULL
                        CommMessage = (struct StructOfSRPCommMessage *)CommInterface -
>GetMsgFromQueue(MsgHandle,VS_FALSE);
                        if( CommMessage != NULL ){
                                switch(CommMessage ->OperateCode){
process message bases on message type.
                                case SRPCOMM_HTTP_ONREQUEST :
receive http request, send simple response.
                                        HttpOnRequest = (struct StructOfSRPComm_HttpOnRequest *)CommMessage->Buf;
                                        if( HttpOnRequest ->RequestType == VS_HTTPREQUEST_GET ){
                                                printf("http get request : %s\n",HttpOnRequest ->FileName);
                                       CommInterface->FormatRspHeader("200 OK",NULL,NULL,NULL,0,Buf);
                                                CommInterface->HttpSend(HttpOnRequest->ConnectionID,strlen(Buf),Buf,VS_TRUE);
                                                sprintf(Buf,"test response data");
                                                CommInterface->HttpSend(HttpOnRequest->ConnectionID,strlen(Buf),Buf,VS_FALSE);
                                        }else if( HttpOnRequest ->RequestType == VS_HTTPREQUEST_GET ){
                                                printf("http post request : %s\n",HttpOnRequest ->FileName);
                                    CommInterface->FormatRspHeader("400 Bad Request",NULL,"close",NULL,0,Buf);
                                                CommInterface->HttpSend(HttpOnRequest->ConnectionID,strlen(Buf),Buf,VS_FALSE);
                                        }
                                        break;
                                }
after the message has been processed, it should be released.
                                CommInterface ->FreeMsgBuf(MsgHandle,(VS_INT8 *)CommMessage);
                        }
                }
Message loop, should be called in main loop to drive starcore
                while( SRPControlInterface -> SRPDispatch(VS_FALSE) == VS_TRUE );
                SRPControlInterface -> SRPIdle();
        }
        CommInterface -> Release();
        SRPControlInterface ->Release();
```

```
close starcore
        VSTermProc();
unload library
        vs_dll_close(hDllInstance);
        return 0;
}
```

15.1.3.1.1.3 Compile and run

simple_webserver 3040

15.1.3.1.2 linux

Write makefile(skip)

15.1.3.2 lua

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
    return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
SrvGroup:_CreateService( "","test", "123",5,0,0,0, 0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )
get communicate interface
CommInterface = SrvGroup:_NewCommInterface()
get input parameter
print(SrvGroup._EnvInputPara[0])
if SrvGroup._EnvInputPara ~= nil then
   SrvGroup: _RunScript("",SrvGroup._EnvInputPara[0] ,"")
end
If Port is not define, then exit
if Port == nil then
  print("starapp -e simple_webserver.lua?Port=3040 or")
  SrvGroup:_ClearService()
  libstarcore._ModuleExit()
  return
end

CommInterface.ConnetionID = CommInterface:_HttpServer(nil,Port,100)
if CommInterface.ConnetionID == 0 then
   print("create webserver ",Port,"fail")
   SrvGroup:_ClearService()
   libstarcore._ModuleExit()
   return
end

print("create webserver ",Port,"success")
Create binbuf
BinBuf = SrvGroup:_NewBinBuf()
Message processing function of the comminterface
function CommInterface:_MsgProc(uMes,Msg)
  if uMes == self.HTTP_ONREQUEST then
     if Msg[3] == self.HTTPREQUEST_GET then
        local a
```

```
      a = self:_FormatRspHeader("200 OK",nil,nil,nil,0)
      BinBuf:_Clear()
      BinBuf:_Set(0,0,'S',a)
      self:_HttpSend(Msg[1],BinBuf,0,true)
      BinBuf:_Clear()
      BinBuf:_Set(0,0,'S',"test response data")
      self:_HttpSend(Msg[1],BinBuf,0,false)
    elseif Msg[3] == self.HTTPREQUEST_POST then
      local a

      a = self:_FormatRspHeader("400 Bad Request",nil,"Close",nil,0)
      BinBuf:_Clear()
      BinBuf:_Set(0,0,'S',a)
      self:_HttpSend(Msg[1],BinBuf,0,false)
    end
  end
end
Message loop
function ExitProc()
   if ExitFlag == true or libstarcore._KeyPress() == 27 then
      return true
   end
   return false
end

libstarcore._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

15.1.3.3python

```
import sys
if hasattr(sys,"argv") :
  if len(sys.argv) > 1 :
    Port = atoi(sys.argv[1])
import libstarpy

libstarpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = libstarpy._GetSrvGroup()
SrvGroup._CreateService( "","test", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )

CommInterface = SrvGroup._NewCommInterface()

if SrvGroup._EnvInputPara != None :
  SrvGroup._RunScript("python",SrvGroup._EnvInputPara._Get(0) ,"")

if "Port" not in dir() or Port == 0 or Port == None :
  print("starapp -e simple_webserver.py?script=python;Port=3040 or")
  print("python simple_webserver.py 3040")
  SrvGroup._ClearService()
  libstarpy._ModuleExit()
  raise Exception("")

CommInterface.ConnetionID = CommInterface._HttpServer("",Port,100)
if CommInterface.ConnetionID == 0 :
  print("create webserver ",Port,"fail")
  SrvGroup._ClearService()
  libstarpy._ModuleExit()
  raise Exception("")
```

```
print("create webserver ",Port,"success")

BinBuf = SrvGroup._NewBinBuf()
def CommInterface_MsgProc(self,uMes,Msg) :
  if uMes == self.HTTP_ONREQUEST :
     if Msg[2] == self.HTTPREQUEST_GET :
        a = self._FormatRspHeader("200 OK","","","",0)
        BinBuf._Clear()
        BinBuf._Set(0,0,'S',a)
        self._HttpSend(Msg[0],BinBuf,0,True)
        BinBuf._Clear()
        BinBuf._Set(0,0,'S',"test response data")
        self._HttpSend(Msg[0],BinBuf,0,False)
     elif Msg[2] == self.HTTPREQUEST_POST :
        if Msg[3]!= 0 :
           PartLength,PartOffset,PartHeader=self._HttpGetMultiPart(Msg[11],0,Msg[3],Msg[9])
           FileName = self._HttpGetNVValue( self._HttpGetHeaderItem(PartHeader,0,"Content-Disposition:"),"filename")
           a = Msg[11]._Get(PartOffset,PartLength,'r')
           a._SaveToFile(FileName,False)
        a = self._FormatRspHeader("200 OK","","Close","",0)
        BinBuf._Clear()
        BinBuf._Set(0,0,'S',a)
        self._HttpSend(Msg[0],BinBuf,0,False)
CommInterface._MsgProc = CommInterface_MsgProc

def ExitProc() :
    if libstarpy._KeyPress() == 27 :
       return True
    return False

libstarpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

15.1.4 HttpServer local request.

After set the port of HttpServer, application can extend the function of starcore by register webpage process functions.

If WebServer port is not set, pages defined in starcore or in the extension can be obtained by function HttpLocalRequest. Using this function, you can combine starcore with other Webserver, such as apache.

15.1.4.1C

15.1.4.1.1 Win32

15.1.4.1.1.1 Create project(VC6)

see above.

15.1.4.1.1.2 Create and edit source file

```
#include "vsopenapi.h"
extern "C"{
        #include "vs_shell.h"
};

//------------------------------------------------------------------------------
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
static class ClassOfSRPCommInterface *CommInterface = NULL;
```

```
static VS_BOOL Local_WebServerMsg(VS_HANDLE MsgHandle,class ClassOfSRPCommInterface *CommInterface,struct
StructOfSRPCommMessage *Mes,VS_ULONG Para,void *AttachBuf,VS_BOOL *ContinueFlag);
static VS_BOOL UnRegisterFlag;
//-------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
        SRPControlInterface = NULL;
        CommInterface = NULL;
    //-------------------------------------------------------------------------
        //--init star core
    VSCore_Init( true, true, "", 0, "", 0,NULL);
        printf("init starcore success\n");

        SRPControlInterface = VSCore_QueryControlInterface();
get communicate interface
        CommInterface = SRPControlInterface ->GetCommInterface();
get basic service interface
        BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);

        //---Runs in Kernel Process
Register page process function, which is running in starcore thread
    CommInterface ->RegWebServerMsgProc(Local_WebServerMsg,0,VS_TRUE,0);

#ifdef STANDWEBSERER
        //--stand webserver
        BasicSRPInterface ->SetWebServerPort("",3040,100,100);
        printf("use : http://127.0.0.1:3040/test\n");
        printf("finish,enter message loop..\n");
        while( 1 ){
ESC is pressed? if so, exit
                VS_INT32 Ch;
                Ch = vs_kbhit();
                if( Ch == 27 )
                        break;
Message loop, should be called in main loop to drive starcore
                while( SRPControlInterface -> SRPDispatch(VS_FALSE) == VS_TRUE );
                SRPControlInterface -> SRPIdle();
        }
#else
        //---local format request and get response
Local request /test page.
        {
                struct StructOfSRPComm_HttpOnRead *HttpOnRead;
                struct StructOfSRPCommMessage *CommMessage;
                VS_ULONG ConnectionID;
                VS_HANDLE MsgHandle;
                VS_CHAR Buf[1024];
                VS_INT32 ReadSize;
create message queue
                //--Create Msg Queue
                MsgHandle = CommInterface ->CreateMsgQueue(256,256);
                ConnectionID = CommInterface -
>HttpLocalRequest(MsgHandle,0,0,VS_HTTPREQUEST_GET,0,0,"/test","","",NULL,"","");
                while( 1 ){
ESC is pressed? if so, exit
                        VS_INT32 Ch;
                        Ch = vs_kbhit();
                        if( Ch == 27 )
                                break;
Has message? it has, then the return value is not NULL
                        CommMessage = (struct StructOfSRPCommMessage *)CommInterface -
>GetMsgFromQueue(MsgHandle,VS_FALSE);
                        if( CommMessage != NULL ){
                                switch(CommMessage ->OperateCode){
process message bases on message type.
                                case SRPCOMM_HTTP_ONREAD : //--receive result;
                                        HttpOnRead = (struct StructOfSRPComm_HttpOnRead *)CommMessage ->Buf;
```

```
                                ReadSize = CommInterface ->HttpRecv(HttpOnRead ->ConnectionID,1024,Buf);
                                Buf[ReadSize] = 0;
                            printf("%s\n",Buf);
                                break;
                            case SRPCOMM_HTTP_ONFINISH :
                                printf("get result finish\n");
                                goto Exit_Lab;
                                break;
                            }
after the message has been processed, it should be released.
                            CommInterface ->FreeMsgBuf(MsgHandle,(VS_INT8 *)CommMessage);
                        }
Message loop, should be called in main loop to drive starcore
                        while( SRPControlInterface -> SRPDispatch(VS_FALSE) == VS_TRUE );
                        SRPControlInterface -> SRPIdle();
                    }
                }
Exit_Lab:

#endif
        UnRegisterFlag = VS_FALSE;
        CommInterface ->UnRegWebServerMsgProc(Local_WebServerMsg,0);
        while( UnRegisterFlag == VS_FALSE ){
Message loop, should be called in main loop to drive starcore
            while( SRPControlInterface -> SRPDispatch(VS_FALSE) == VS_TRUE );
            SRPControlInterface -> SRPIdle();
        }
        BasicSRPInterface ->Release();
        CommInterface -> Release();
        SRPControlInterface ->Release();
        VSCore_Term();
        return 0;
}

the page process function
//---http://127.0.0.1/test
VS_BOOL Local_WebServerMsg(VS_HANDLE MsgHandle,class ClassOfSRPCommInterface *CommInterface,struct
StructOfSRPCommMessage *Mes,VS_ULONG Para,void *AttachBuf,VS_BOOL *ContinueFlag)
{
        struct StructOfSRPComm_HttpOnRequest *HttpOnRequest;
        VS_CHAR Buf[256];

    switch( Mes -> OperateCode ){
    case SRPCOMM_HTTP_ONREQUEST :
receive http request, then returns VS_TRUE, indicates the function processes the request, the kernel will not continue dispatch.
        HttpOnRequest = (struct StructOfSRPComm_HttpOnRequest *)Mes ->Buf;
        if( HttpOnRequest -> RequestType != VS_HTTPREQUEST_GET || vs_string_stricmp( HttpOnRequest -> FileName,
"/test" ) != 0 )
            return VS_FALSE;  // not our url
                HttpOnRequest = (struct StructOfSRPComm_HttpOnRequest *)Mes->Buf;
                if( HttpOnRequest ->RequestType == VS_HTTPREQUEST_GET ){
                    printf("http get request : %s\n",HttpOnRequest ->FileName);

                    CommInterface->FormatRspHeader("200 OK",NULL,NULL,NULL,0,Buf);
                    CommInterface->HttpSend(HttpOnRequest->ConnectionID,strlen(Buf),Buf,VS_TRUE);
                    sprintf(Buf,"test response data");
                    CommInterface->HttpSend(HttpOnRequest->ConnectionID,strlen(Buf),Buf,VS_FALSE);  //--finish
                }else if( HttpOnRequest ->RequestType == VS_HTTPREQUEST_GET ){
                    printf("http post request : %s\n",HttpOnRequest ->FileName);
                    CommInterface->FormatRspHeader("400 Bad Request",NULL,"close",NULL,0,Buf);
                    CommInterface->HttpSend(HttpOnRequest->ConnectionID,strlen(Buf),Buf,VS_FALSE);
                }
                (*ContinueFlag) = VS_TRUE;
                break;
        case SRPCOMM_HTTP_ONWEBSERVERUNREG :
                UnRegisterFlag = VS_TRUE;
                break;
```

```
        }
        return VS_TRUE;
}
```

15.1.4.1.2 linux

Write makefile(skip)

15.1.4.2 lua

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
SrvGroup:_CreateService( "","test", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )

--create web page
get communicate interface
CommInterface = SrvGroup:_NewCommInterface()
function CommInterface:_WebServerProc(uMes,Msg)
  print(uMes,Msg)
  if uMes == self.HTTP_ONREQUEST then
    print(Msg)
    if Msg[7] == "/test" then
       local a

       print("receive http request.....")
       a = self:_FormatRspHeader("200 OK",nil,nil,nil,0)
       BinBuf:_Clear()
       BinBuf:_Set(0,0,'S',a)
       self:_HttpSend(Msg[1],BinBuf,0,true)
       BinBuf:_Clear()
       BinBuf:_Set(0,0,'S',"test response data")
       self:_HttpSend(Msg[1],BinBuf,0,false)
    end
  end
  print("return.....")
  return false,false
end

--create local request
get communicate interface
CommInterface1 = SrvGroup:_NewCommInterface()
Create binbuf
BinBuf = SrvGroup:_NewBinBuf()
ExitFlag = 0
Message processing function of the comminterface
function CommInterface1:_MsgProc(uMes,Msg)
  if uMes == self.HTTP_ONREAD then
     BinBuf:_Clear()
     self:_HttpRecv(Msg[1],BinBuf,0)
     print(BinBuf:_Get(0,0,"a"))
  elseif uMes == self.HTTP_ONFINISH then
     ExitFlag = 1
  end
end
CommInterface1:_HttpLocalRequest(CommInterface1.HTTPREQUEST_GET,"/test", "", "", nil)
Message loop
function ExitProc()
   if ExitFlag == 1 then
```

```
        return true
    end
    return false
end

libstarcore._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

15.1.4.3python

```
import sys
if hasattr(sys,"argv") :
  if len(sys.argv) > 1 :
    Port = atoi(sys.argv[1])
import libstarpy

libstarpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = libstarpy._GetSrvGroup()
SrvGroup._CreateService( "","test", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )

#--create web page
CommInterface = SrvGroup._NewCommInterface()
def CommInterface_WebServerProc(self,uMes,Msg) :
 global BinBuf
 print(uMes,Msg)
 if uMes == self.HTTP_ONREQUEST :
   print(Msg)
   if Msg[6] == "/test" :
      print("receive http request.....")
      a = self._FormatRspHeader("200 OK","","","",0)
      BinBuf._Clear()
      BinBuf._Set(0,0,'S',a)
      self._HttpSend(Msg[0],BinBuf,0,True)
      BinBuf._Clear()
      BinBuf._Set(0,0,'S',"test response data")
      self._HttpSend(Msg[0],BinBuf,0,False)
      return True,True
CommInterface._WebServerProc = CommInterface_WebServerProc

#--create local request
CommInterface1 = SrvGroup._NewCommInterface()
BinBuf = SrvGroup._NewBinBuf()
ExitFlag = 0
def CommInterface1_MsgProc(self,uMes,Msg) :
 global ExitFlag
 if uMes == self.HTTP_ONREAD :
   BinBuf._Clear()
   self._HttpRecv(Msg[0],BinBuf,0)
   print(BinBuf._Get(0,0,"a"))
 elif uMes == self.HTTP_ONFINISH :
   ExitFlag = 1
CommInterface1._MsgProc = CommInterface1_MsgProc

CommInterface1._HttpLocalRequest(CommInterface1.HTTPREQUEST_GET,"/test", "", "", "")

def ExitProc() :
   global ExitFlag
   if libstarpy._KeyPress() == 27 :
      return True
   if ExitFlag == 1 :
```

```
    CommInterface1._HttpLocalRequest(CommInterface1.HTTPREQUEST_GET,"/test", "", "", "")
    return True
  return False

libstarpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

15.2 WebService

To support WebService,you should open WebServer port. There are three methods:

1. Command line

   starapp -w XXX ;Uses parameter –w to set WebServer port number.

2. Script

   Script may call function _SetWebServerPort to open or close Web service. For example:

   _SetWebServerPort (Host,Portnumber, ConnectionNumber, PostSize)

Host Url name, in normal case, should be set to ""

Portnumber:port number

ConnectionNumber:max number of connections supported

PostSize:max size uploaded in kbytes

3. c/c++ language

VS_BOOL SRPAPI SetWebServerPort(VS_CHAR *WebServerHost,VS_UINT16 WebServerPortNumber,VS_INT32 ConnectionNumber,VS_ULONG PostSize);
WebServerHost is set to NULL

15.2.1 Create WebService

15.2.1.1WebService object

1. If object's attribute "_WebServiceFlag" is set to true, then the object can be called through WebService

a = Service.TestClass:_New()

a._Name = "TestObject"

a._WebServiceFlag=true

In WebService, WebService object acts as PortType, functions defined in the object acts as Operation;

WSDL will be generated by starcore automatically
Url: http://127.0.0.1:XXX/wsdl

or:http://127.0.0.1:XXX/ ServiceName /wsdl

WebService does not support VS_PARAPKGPTR as parameter or return value, for it is not a structured data.

15.2.1.2lua

Examples in directoryexamples\comm.basic\ webservice.lua

a = Service.TestClass:_New()

a._Name = "TestObject"

a._WebServiceFlag=true

s

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
--Create service
SrvGroup:_CreateService( "",WebServiceCallServer", "123",5,0,0,0, 0,0," E124266B-C66D-4fc3-B287-6D0B4C5F90AD" )
Service = SrvGroup:_GetService("root","123")

--create service item(object group)
Service:_CreateSysRootItem("TestItem","")
SrvItem = Service:_GetSysRootItem( "TestItem" )

--Create Atomic Class, for define function, no attribute
create atomic object class
AtomicClass = Service:_CreateAtomicObjectSimple("TestItem","TestClass",nil, "");
create function of class
Service:_CreateAtomicFunctionSimple(AtomicClass,"GetNumber","VS_INT32 GetNumber(VS_INT32 Para);", "",false,false);
create function of class
Service:_CreateAtomicFunctionSimple(AtomicClass,"GetString","VS_CHAR *GetString(VS_CHAR *Para);", "",false,false);
function Service.TestClass:GetNumber(Para)
   return Para+1;
end
function Service.TestClass:GetString(Para)
   return Para .. "asdfsaf";
end

Create object and set its WebService flag.
a = Service.TestClass:_New()
a._Name = "TestObject"
a._WebServiceFlag=true

print( "Server Start ok....")
Message loop
function ExitProc()
   if libstarcore._KeyPress() == 27 then
      return true
   end
   return false
end

libstarcore._MsgLoop( ExitProc )
exit, clear service and starcore
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Run:

starapp -w 3040 -e XXXX.lua

15.2.1.3python

Examples in directoryexamples\comm.basic\ webservice.python

a = Service.TestClass._New()

a._Name = "TestObject"

a._WebServiceFlag=True

s:

```
import sys
import libstarpy
initstarcore(cle)
libstarpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = libstarpy._GetSrvGroup()
#--create service
SrvGroup._CreateService( "","WebServiceCallServer", "123",5,0,0,0,0,0,"E124266B-C66D-4fc3-B287-6D0B4C5F90AD" )
Service = SrvGroup._GetService("root","123")

#--create service item(object group)
Service._CreateSysRootItem("TestItem","")
SrvItem = Service._GetSysRootItem( "TestItem" )

#--Create Atomic Class, for define function, no attribute
create atomic object class
AtomicClass,ErrorInfo = Service._CreateAtomicObjectSimple("TestItem","TestClass","","");
create function of class
Service._CreateAtomicFunctionSimple(AtomicClass,"GetNumber","VS_INT32 GetNumber(VS_INT32 Para);", "",False,False);
create function of class
Service._CreateAtomicFunctionSimple(AtomicClass,"GetString","VS_CHAR *GetString(VS_CHAR *Para);", "",False,False);

def Service_TestClass_GetNumber(self,Para) :
   return Para+1;
Service.TestClass.GetNumber = Service_TestClass_GetNumber;

def Service_TestClass_GetString(self,Para) :
   return Para+"asdfsaf";
Service.TestClass.GetString = Service_TestClass_GetString;

a = Service.TestClass._New()
a._Name = "TestObject"
a._WebServiceFlag=True

#--
print( "Server Start ok....")
Message loop
def ExitProc() :
   if libstarpy._KeyPress() == 27 :
      return True
   return False

libstarpy._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

15.2.1.4C

Examples in directoryexamples\comm.basic\ webservice.c

15.2.1.4.1 Win32

15.2.1.4.1.1 Create project(VC6)

15.2.1.4.1.2 Create and edit source file

Create source file test_server,add to project,

```
#include "vsopenapi.h"
extern "C"{
	#include "vs_shell.h"
};

//-------------------------------------------------------------------------------
VS_HANDLE hDllInstance;
VSCore_RegisterCallBackInfoProc RegisterCallBackInfoProc;
VSCore_InitProc VSInitProc;
VSCore_TermProc VSTermProc;
VSCore_QueryControlInterfaceProc QueryControlInterfaceProc;
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
static class ClassOfSRPInterface *SRPInterface = NULL;

callback function, to display information
static VS_ULONG MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_ULONG wParam, VS_ULONG
lParam, VS_BOOL &IsProcessed, VS_ULONG Para )
{
   switch( uMsg ){
   case MSG_VSDISPMSG :
	case MSG_VSDISPLUAMSG :
		printf("[core]%s\n",(VS_CHAR *)wParam);
		break;
   case MSG_DISPMSG :
	case MSG_DISPLUAMSG :
		printf("%s\n",(VS_CHAR *)wParam);
	  break;
	case MSG_EXIT :
		break;
   }
   return 0;
}

static VS_INT32 GetNumber(void *Object,VS_INT32 Para)
{
   printf( "Remote Call Number [%d]\n ",Para);
	return Para + 1;
}

static VS_CHAR *GetString(void *Object,VS_CHAR *Para)
{
	static VS_CHAR CharBuf[128];

   printf( "Remote Call String [%s]\n",Para);
	sprintf(CharBuf,"%sasdfsaf",Para);
   return CharBuf;
}

//-------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
	VS_CHAR ModuleName[512];
	VS_UUID ServiceID,ClassID;
	void *AtomicClass,*Object,*GetNumber_AtomicFunction,*GetString_AtomicFunction;
	VS_CHAR *ErrorInfo;

	SRPControlInterface = NULL;
	BasicSRPInterface = NULL;
   //-----------------------------------------------------------------------
   load library
	sprintf(ModuleName,"libstarcore%s",VS_MODULEEXT);
   hDllInstance = vs_dll_open( ModuleName );
	if( hDllInstance == NULL ){
		printf("load library [%s] error....\n",ModuleName);
		return -1;
   }
```

```
   get export functions of the library
   RegisterCallBackInfoProc = (VSCore_RegisterCallBackInfoProc)vs_dll_sym( hDllInstance,
VSCORE_REGISTERCALLBACKINFO_NAME );
   VSInitProc = (VSCore_InitProc)vs_dll_sym( hDllInstance, VSCORE_INIT_NAME );
   VSTermProc = (VSCore_TermProc)vs_dll_sym( hDllInstance, VSCORE_TERM_NAME );
   QueryControlInterfaceProc = (VSCore_QueryControlInterfaceProc)vs_dll_sym( hDllInstance,
VSCORE_QUERYCONTROLINTERFACE_NAME );
      //--init star core
callback function, to display information
      RegisterCallBackInfoProc(MsgCallBack,0);
init starcore
   VSInitProc( true, true, "", 0, "", 3008,NULL);

      printf("init starcore success\n");
get control interface controlinterface
      SRPControlInterface = QueryControlInterfaceProc();
get basic service interface
   BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);
create service
      BasicSRPInterface ->StringToUuid("E124266B-C66D-4fc3-B287-6D0B4C5F90AD",&ServiceID);
      BasicSRPInterface ->CreateService("","WebServiceCallServer",&ServiceID,"123",0,0,0,0,0,0);
get service interface
      SRPInterface = BasicSRPInterface ->GetSRPInterface("WebServiceCallServer","root","123");
create service item
      SRPInterface ->CreateSysRootItem("TestItem","",NULL,NULL);
active service item
      SRPInterface ->ActiveSysRootItem( "TestItem" );
      //---Create Atomic Class, for define function, no attribute
create atomic object class
      AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClass",NULL, NULL,&ErrorInfo);
create function of class
      GetNumber_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetNumber","VS_INT32
GetNumber(VS_INT32 Para);", NULL,&ErrorInfo,VS_FALSE,VS_FALSE);
create function of class
      GetString_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetString","VS_CHAR
*GetString(VS_CHAR *Para);", NULL,&ErrorInfo,VS_FALSE,VS_FALSE);
//---Set Function Address
   set function address, which should be called after all functions are created finish
      SRPInterface -> SetAtomicFunction(GetNumber_AtomicFunction,(void *)GetNumber);
      SRPInterface -> SetAtomicFunction(GetString_AtomicFunction,(void *)GetString);

      printf("create TestObject for webservice..\n");
get atomic object class ID,which is used to create instance
      SRPInterface ->GetAtomicID(AtomicClass,&ClassID);
      Object = SRPInterface ->MallocObjectL(&ClassID,0,NULL);  //---need not create global object
set object name, then can be find object by name
      SRPInterface ->SetName(Object,"TestObject");
      SRPInterface ->SetWebServiceFlag(Object,VS_TRUE);

      BasicSRPInterface ->SetWebServerPort(NULL,3040,100,200);

      printf("finish,enter message loop..\n");
      while( 1 ){
ESC is pressed? if so, exit
            VS_INT32 Ch;
            Ch = vs_kbhit();
            if( Ch == 27 )
                  break;
Message loop, should be called in main loop to drive starcore
            if( SRPControlInterface -> SRPDispatch(VS_FALSE) == VS_FALSE ){
                  SRPControlInterface -> SRPIdle();
                  SRPControlInterface -> SRPDispatch(VS_TRUE);
            }
      }
      SRPControlInterface ->Release();
      BasicSRPInterface ->Release();
close starcore
```

```
        VSTermProc();
unload library
        vs_dll_close(hDllInstance);
        return 0;
}
```

15.2.1.4.1.3 Compile and run

test_server

15.2.1.4.2 linux

Write Makefile

```
#*******************************************************************************
#
# Makefile for StarCore.
# www.srplab.com
#*******************************************************************************
DEBUG         := YES
PROFILE       := NO
#*******************************************************************************
CC     := gcc
CXX    := g++
LD     := g++
AR     := ar
RANLIB := ranlib

DEBUG_CFLAGS     := -Wall -Wno-format -g -DDEBUG -DENV_LINUX
RELEASE_CFLAGS   := -Wall -Wno-unknown-pragmas -Wno-format -O3 -DENV_LINUX

LIBS := -ldl -lpthread -lrt
EXTRA_LIBS := ../../output/linux/libstarlib.a /usr/lib/libuuid.a

DEBUG_CXXFLAGS   := ${DEBUG_CFLAGS}
RELEASE_CXXFLAGS := ${RELEASE_CFLAGS}

DEBUG_LDFLAGS    := -g
RELEASE_LDFLAGS  :=

ifeq (YES, ${DEBUG})
  CFLAGS      := ${DEBUG_CFLAGS}
  CXXFLAGS    := ${DEBUG_CXXFLAGS}
  LDFLAGS     := ${DEBUG_LDFLAGS}
else
  CFLAGS      := ${RELEASE_CFLAGS}
  CXXFLAGS    := ${RELEASE_CXXFLAGS}
  LDFLAGS     := ${RELEASE_LDFLAGS}
endif

ifeq (YES, ${PROFILE})
  CFLAGS   := ${CFLAGS} -pg -O3
  CXXFLAGS := ${CXXFLAGS} -pg -O3
  LDFLAGS  := ${LDFLAGS} -pg
endif

#*******************************************************************************
# Makefile code common to all platforms
#*******************************************************************************

CFLAGS   := ${CFLAGS}   ${DEFS}
```

```
CXXFLAGS := ${CXXFLAGS} ${DEFS}

#*****************************************************************************
# include source and paths
#*****************************************************************************

INCS_T := /usr/include/starcore
INCS   = $(addprefix -I,$(INCS_T))

TEST_SERVER_CXXSRCS := test_server.cpp
TEST_SERVER_DEFER_CXXSRCS := test_server_defer.cpp

#*****************************************************************************
TEST_SERVER_CXXOBJS := $(TEST_SERVER_CXXSRCS:%.cpp=%.o)
TEST_SERVER_DEFER_CXXOBJS := $(TEST_SERVER_DEFER_CXXSRCS:%.cpp=%.o)

#*****************************************************************************
CXXOBJS := ${TEST_SERVER_CXXOBJS} ${TEST_SERVER_DEFER_CXXOBJS}
COBJS :=

EXEC_TEST_SERVER_OBJS := ${TEST_SERVER_CXXOBJS}
EXEC_TEST_SERVER_DEFER_OBJS := ${TEST_SERVER_DEFER_CXXOBJS}

#*****************************************************************************
# Targets of the build
#*****************************************************************************
OBJS_PATH = output/linux

EXEC_TEST_SERVER := ${OBJS_PATH}/test_server
EXEC_TEST_SERVER_DEFER := ${OBJS_PATH}/test_server_defer

all: ${EXEC_TEST_SERVER} ${EXEC_TEST_SERVER_DEFER}

#*****************************************************************************
# Output
#*****************************************************************************

${EXEC_TEST_SERVER}: ${EXEC_TEST_SERVER_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_TEST_SERVER_OBJS} ${LIBS} ${EXTRA_LIBS}

${EXEC_TEST_SERVER_DEFER}: ${EXEC_TEST_SERVER_DEFER_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_TEST_SERVER_DEFER_OBJS} ${LIBS} ${EXTRA_LIBS}


#*****************************************************************************
# common rules
#*****************************************************************************

${CXXOBJS} :
	${CXX} ${CXXFLAGS} ${INCS} $< -o $@ -c $*.cpp

${COBJS} :
	${CC} ${CFLAGS} ${INCS} -o $@ -c $*.c

dist:
	bash makedistlinux

clean:
	-rm -f core ${CXXOBJS} ${COBJS} ${EXEC_TEST_SERVER} ${EXEC_TEST_SERVER_DEFER}

depend:
	#makedepend ${INCS} ${SRCS}
```

15.2.2 Get WSDL of WebService

From url:

http://127.0.0.1:3040/wsdl

or

http://127.0.0.1:3040/__WebServiceCallServer/wsdl

You also can use script or C function GetWsdl

Example of wsdl is as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions targetNamespace="urn:starcore-WebServiceCallServer" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="urn:starcore-WebServiceCallServer" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:MIME="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:DIME="http://schemas.xmlsoap.org/ws/2002/04/dime/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <xsd:schema targetNamespace="urn:starcore-WebServiceCallServer">
      <xsd:element name="TestClassGetNumberreq">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Para" type="xsd:int" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="TestClassGetNumberrsp">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="RetValue" type="xsd:int" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="TestClassGetStringreq">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Para" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="TestClassGetStringrsp">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="RetValue" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <message name="coreempty" />
  <message name="coreerror" />
  <message name="TestClassGetNumberrequest">
    <part name="parameter" element="tns:TestClassGetNumberreq" />
  </message>
  <message name="TestClassGetNumberresponse">
    <part name="parameter" element="tns:TestClassGetNumberrsp" />
  </message>
  <message name="TestClassGetStringrequest">
    <part name="parameter" element="tns:TestClassGetStringreq" />
```

```
  </message>
  <message name="TestClassGetStringresponse">
    <part name="parameter" element="tns:TestClassGetStringrsp" />
  </message>
  <portType name="TestObjectPortType">
    <operation name="GetNumber">
      <input message="tns:TestClassGetNumberrequest" />
      <output message="tns:TestClassGetNumberresponse" />
    </operation>
    <operation name="GetString">
      <input message="tns:TestClassGetStringrequest" />
      <output message="tns:TestClassGetStringresponse" />
    </operation>
  </portType>
  <binding name="TestObject" type="tns:TestObjectPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="GetNumber">
      <soap:operation style="document" soapAction="urn:GetNumber" />
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
    <operation name="GetString">
      <soap:operation style="document" soapAction="urn:GetString" />
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
  <service name="WebServiceCallServer">
    <port name="TestObject" binding="tns:TestObject">
      <soap:address location="http://127.0.0.1:3040/__WebServiceCallServer/webservice/TestObject" />
    </port>
  </service>
</definitions>
```

15.2.3 WebService client(gsoap)

Examples in directoryexamples\comm.basic\ webservice.client

Based on WSDL, service can be called with standard SOAP message. Here the client is written using gsoap

Run:

wsdl2h -s -o WebServiceCallServer.h WebServiceCallServer.wsdl

generate header file WebServiceCallServer.h

Run:

soapcpp2 -i -C WebServiceCallServer.h

Generate client skeleton, which includs the following files:

soapC.cpp

soapH.h

soapStub.h

soapTestObjectProxy.cpp

soapTestObjectProxy.h

TestObject.nsmap

15.2.3.1Win32

15.2.3.1.1 Create project(VC6)

Create new project:

Include the following files into the project.

soapTestObjectProxy.cpp

soapC.cpp

stdsoap2.cpp

15.2.3.1.2 Create and edit source file

Create source file clientmain.cpp,add to project.

```
#include "soapTestObjectProxy.h"
#include "TestObject.nsmap"

char server[256];

int main(int argc, char **argv)
{
	TestObjectProxy TestObject;
	_ns1__TestClassGetNumberreq q1;
	_ns1__TestClassGetStringreq q2;
	_ns1__TestClassGetNumberrsp s1;
	_ns1__TestClassGetStringrsp s2;

	if( argc < 2 ){
		printf("usage ServerUrl\n");
		return -1;
	}
	sprintf(server,"http://%s/__WebServiceCallServer/webservice/TestObject",argv[1]);
	TestObject.soap_endpoint = server;
	q1.Para=123;
	TestObject.GetNumber(&q1,&s1);
	if (TestObject.error)
		TestObject.soap_stream_fault(std::cerr);
	else
		printf("result = %d\n", s1.RetValue);

	q2.Para="Hello";
	TestObject.GetString(&q2,&s2);
	if (TestObject.error)
		TestObject.soap_stream_fault(std::cerr);
	else
		printf("result = %s\n", s2.RetValue);

  return 0;
}
```

15.2.3.1.3 Compile and run

WebServiceCallServer_Client 127.0.0.1:3040

15.2.4 Create and use stand alone starcore service.

Examples in directoryexamples\comm.basic\ webservice.service

How to create service, please refer to chapters before.

15.2.4.1Called by C

15.2.4.1.1 Win32

15.2.4.1.1.1 create console application(VC6)

skip

15.2.4.1.1.2 Create and edit source file

1. Create service RemoteCallServer header file

Run:star2h service\script\RemoteCallServer . ,then, in local directory, will generate.

RemoteCallServer.h

RemoteCallServer_UUIDDef.cpp

RemoteCallServer_VSClass.cpp

RemoteCallServer_VSClass.H

RemoteCallServer_VSDHeader.H

2. Create source file ,add project

```
#include "vsopenapi.h"
extern "C"{
	#include "vs_shell.h"
};

//------------------------------------------------------------------------------
VS_HANDLE hDllInstance;
VSCore_RegisterCallBackInfoProc RegisterCallBackInfoProc;
VSCore_InitProc VSInitProc;
VSCore_TermProc VSTermProc;
VSCore_QueryControlInterfaceProc QueryControlInterfaceProc;
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
static class ClassOfSRPInterface *SRPInterface = NULL;

callback function, to display information
static VS_ULONG MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_ULONG wParam, VS_ULONG
lParam, VS_BOOL &IsProcessed, VS_ULONG Para )
{
   switch( uMsg ){
   case MSG_VSDISPMSG :
      case MSG_VSDISPLUAMSG :
            printf("[core]%s\n",(VS_CHAR *)wParam);
            break;
   case MSG_DISPMSG :
      case MSG_DISPLUAMSG :
            printf("%s\n",(VS_CHAR *)wParam);
      break;
      case MSG_EXIT :
```

```
                break;
    }
    return 0;
}

//------------------------------------------------------------------------------
int main(int argc, char* argv[])
{
        VS_CHAR ModuleName[512];
        VS_UUID ServiceID,ClassID;
        void *Object;

        SRPControlInterface = NULL;
        BasicSRPInterface = NULL;
    //--------------------------------------------------------------------
    load library
        sprintf(ModuleName,"libstarcore%s",VS_MODULEEXT);
    hDllInstance = vs_dll_open( ModuleName );
        if( hDllInstance == NULL ){
                printf("load library [%s] error....\n",ModuleName);
                return -1;
    }
    get export function of the library
    RegisterCallBackInfoProc = (VSCore_RegisterCallBackInfoProc)vs_dll_sym( hDllInstance,
VSCORE_REGISTERCALLBACKINFO_NAME );
    VSInitProc = (VSCore_InitProc)vs_dll_sym( hDllInstance, VSCORE_INIT_NAME );
    VSTermProc = (VSCore_TermProc)vs_dll_sym( hDllInstance, VSCORE_TERM_NAME );
    QueryControlInterfaceProc = (VSCore_QueryControlInterfaceProc)vs_dll_sym( hDllInstance,
VSCORE_QUERYCONTROLINTERFACE_NAME );
        //--init star core
callback function, to display information
        RegisterCallBackInfoProc(MsgCallBack,0);
init starcore
    VSInitProc( true, true, "", 0, "", 3008,NULL);

        printf("init starcore success\n");
get control interface controlinterface
        SRPControlInterface = QueryControlInterfaceProc();
get basic service interface
    BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);

        //---import service
load depended service
        if( BasicSRPInterface ->ImportServiceWithPath("..\\..\\service\\script","RemoteCallServer",VS_TRUE) == VS_FALSE ){
       printf("import service [..\\..\\service\\script\\RemoteCallServer] fail\n");
                SRPControlInterface ->Release();
                BasicSRPInterface ->Release();
                VSTermProc();
                vs_dll_close(hDllInstance);
                return -1;
        }
        //---create service
create service
        BasicSRPInterface ->StringToUuid("B07427AF-3C8B-4e88-9F06-535831EF46EF",&ServiceID);
        BasicSRPInterface ->CreateService("","WebServiceCallServer",&ServiceID,"123",0,0,0,0,0,0);
get service interface
        SRPInterface = BasicSRPInterface ->GetSRPInterface("WebServiceCallServer","root","123");
create service item
        SRPInterface ->CreateSysRootItem( "TestItem","",NULL,NULL );
active service item
        SRPInterface ->ActiveSysRootItem( "TestItem" );
        printf("create TestObject for webservice..\n");
        SRPInterface -> GetID(SRPInterface ->GetObjectEx(NULL,"TestClass"),&ClassID);
create globalobject, which will by sync to client
        Object = SRPInterface ->MallocGlobalObject(SRPInterface->GetSysRootItem("TestItem"),0,&ClassID,0,NULL,0);
set object name, then can be find object by name
        SRPInterface ->SetName(Object,"TestObject");
```

```
        SRPInterface ->SetWebServiceFlag(Object,VS_TRUE);
        BasicSRPInterface ->SetWebServerPort(NULL,3040,100,200);

        printf("finish,enter message loop..\n");
        while( 1 ){
ESC is pressed? if so, exit
                VS_INT32 Ch;
                Ch = vs_kbhit();
                if( Ch == 27 )
                        break;
Message loop, should be called in main loop to drive starcore
                if( SRPControlInterface -> SRPDispatch(VS_FALSE) == VS_FALSE ){
                        SRPControlInterface -> SRPIdle();
                        SRPControlInterface -> SRPDispatch(VS_TRUE);
                }
        }
        SRPControlInterface ->Release();
        BasicSRPInterface ->Release();
close starcore
        VSTermProc();
unload library
        vs_dll_close(hDllInstance);
        return 0;
}
```

15.2.4.1.1.3 Compile and run

test_server_RemoteCallServer

15.2.4.1.2 linux

Write Makefile

```
#*****************************************************************************
#
# Makefile for StarCore.
# www.srplab.com
#*****************************************************************************
DEBUG         := YES
PROFILE       := NO
#*****************************************************************************
CC     := gcc
CXX    := g++
LD     := g++
AR     := ar
RANLIB := ranlib

DEBUG_CFLAGS     := -Wall -Wno-format -g -DDEBUG -DENV_LINUX
RELEASE_CFLAGS   := -Wall -Wno-unknown-pragmas -Wno-format -O3 -DENV_LINUX

LIBS := -ldl -lpthread -lrt
EXTRA_LIBS := /usr/lib/libstarlib.a /usr/lib/libuuid.a

DEBUG_CXXFLAGS   := ${DEBUG_CFLAGS}
RELEASE_CXXFLAGS := ${RELEASE_CFLAGS}

DEBUG_LDFLAGS    := -g
RELEASE_LDFLAGS  :=

ifeq (YES, ${DEBUG})
```

```
  CFLAGS      := ${DEBUG_CFLAGS}
  CXXFLAGS    := ${DEBUG_CXXFLAGS}
  LDFLAGS     := ${DEBUG_LDFLAGS}
else
  CFLAGS      := ${RELEASE_CFLAGS}
  CXXFLAGS    := ${RELEASE_CXXFLAGS}
  LDFLAGS     := ${RELEASE_LDFLAGS}
endif

ifeq (YES, ${PROFILE})
  CFLAGS   := ${CFLAGS} -pg -O3
  CXXFLAGS := ${CXXFLAGS} -pg -O3
  LDFLAGS  := ${LDFLAGS} -pg
endif

#*******************************************************************************
# Makefile code common to all platforms
#*******************************************************************************

CFLAGS   := ${CFLAGS}   ${DEFS}
CXXFLAGS := ${CXXFLAGS} ${DEFS}

#*******************************************************************************
# include source and paths
#*******************************************************************************

INCS_T := /usr/include/starcore
INCS   = $(addprefix -I,$(INCS_T))

TEST_SERVER_REMOTECALLSERVER_CXXSRCS := test_server_RemoteCallServer.cpp
TEST_SERVER_REMOTECALLSERVERDEFER_CXXSRCS := test_server_RemoteCallServerDefer.cpp

#*******************************************************************************
TEST_SERVER_REMOTECALLSERVER_CXXOBJS :=
$(TEST_SERVER_REMOTECALLSERVER_CXXSRCS:%.cpp=%.o)
TEST_SERVER_REMOTECALLSERVERDEFER_CXXOBJS :=
$(TEST_SERVER_REMOTECALLSERVERDEFER_CXXSRCS:%.cpp=%.o)

#*******************************************************************************
CXXOBJS := ${TEST_SERVER_REMOTECALLSERVER_CXXOBJS}
${TEST_SERVER_REMOTECALLSERVERDEFER_CXXOBJS}
COBJS :=

EXEC_TEST_SERVER_REMOTECALLSERVER_OBJS := ${TEST_SERVER_REMOTECALLSERVER_CXXOBJS}
EXEC_TEST_SERVER_REMOTECALLSERVERDEFER_OBJS :=
${TEST_SERVER_REMOTECALLSERVERDEFER_CXXOBJS}

#*******************************************************************************
# Targets of the build
#*******************************************************************************
EXEC_TEST_SERVER_REMOTECALLSERVER := test_server_RemoteCallServer_linux
EXEC_TEST_SERVER_REMOTECALLSERVERDEFER := test_server_RemoteCallServerDefer_linux

all: ${EXEC_TEST_SERVER_REMOTECALLSERVER} ${EXEC_TEST_SERVER_REMOTECALLSERVERDEFER}

#*******************************************************************************
# Output
#*******************************************************************************

${EXEC_TEST_SERVER_REMOTECALLSERVER}: ${TEST_SERVER_REMOTECALLSERVER_CXXOBJS}
	${LD} -o $@ ${LDFLAGS} ${TEST_SERVER_REMOTECALLSERVER_CXXOBJS} ${LIBS} ${EXTRA_LIBS}

${EXEC_TEST_SERVER_REMOTECALLSERVERDEFER}:
${EXEC_TEST_SERVER_REMOTECALLSERVERDEFER_OBJS}
	${LD} -o $@ ${LDFLAGS} ${EXEC_TEST_SERVER_REMOTECALLSERVERDEFER_OBJS} ${LIBS}
${EXTRA_LIBS}
```

```
#*****************************************************************************
# common rules
#*****************************************************************************

${CXXOBJS} :
	${CXX} ${CXXFLAGS} ${INCS} $< -o $@ -c $*.cpp

${COBJS} :
	${CC} ${CFLAGS} ${INCS} -o $@ -c $*.c

dist:
	bash makedistlinux

clean:
	-rm -f core ${CXXOBJS} ${COBJS} ${EXEC_TEST_SERVER_REMOTECALLSERVER}
${EXEC_TEST_SERVER_REMOTECALLSERVERDEFER}

depend:
	#makedepend ${INCS} ${SRCS}
```

15.2.4.2called by LUA

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",3008) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
create service
load depended service
SrvGroup:_ImportServiceWithPath("..\\..\\service\\script","RemoteCallServer",true)
SrvGroup:_CreateService( "","WebServiceCallServer", "123",5,0,0,0, 0,0,"B07427AF-3C8B-4e88-9F06-535831EF46EF" )
Service = SrvGroup:_GetService("root","123")
create service item
Service:_CreateSysRootItem("TestItem","")
active service item
Service:_ActiveSysRootItem("TestItem")
SrvItem = Service:_GetSysRootItem( "TestItem" )

a = Service.TestClass:_NewGlobal(SrvItem)
a._Name = "TestObject"

a._WebServiceFlag=true

print( "Server Start ok....")
Message loop
function ExitProc()
   if libstarcore._KeyPress() == 27 then
      return true
   end
   return false
end

libstarcore._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Run

starapp -w 3040 -e XXXX.lua

15.2.4.3Called by python

```
import sys
import libstarpy
initstarcore(cle)
libstarpy._InitCore(True,True,False,True,"",0,"",0)
get service group 0, and create service
SrvGroup = libstarpy._GetSrvGroup()
#--create service
load depended service
SrvGroup._ImportServiceWithPath("..\\..\\service\\script","RemoteCallServer",True)
SrvGroup._CreateService( "","WebServiceCallServer", "123",5,0,0,0,0,0,"B07427AF-3C8B-4e88-9F06-535831EF46EF" )
Service = SrvGroup._GetService("root","123")
create service item
Service._CreateSysRootItem("TestItem","")
active service item
Service._ActiveSysRootItem("TestItem")
SrvItem = Service._GetSysRootItem( "TestItem" )

a = Service.TestClass._NewGlobal(SrvItem)
a._Name = "TestObject"

a._WebServiceFlag=True

#--
print( "Server Start ok....")
Message loop
def ExitProc() :
    if libstarpy._KeyPress() == 27 :
        return True
    return False

libstarpy._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup._ClearService()
libstarpy._ModuleExit()
```

Run

starapp -w 3040 -e "XXXX.py?script=python"

15.3 WebService-compilcate data type

In Remotecall, complicate data can be delivered by VSTYPE_OBJPTR

In WebService, complicate data can be delivered by struct or VSTYPE_OBJPTR.

Data types supported by object is little more than struct, for example, it supports variable length.

Data types supported list below:

For object attribute:

VSTYPE_BOOL :
VSTYPE_INT8 :
VSTYPE_UINT8 :
VSTYPE_INT16 :
VSTYPE_UINT16 :
VSTYPE_INT32 :
VSTYPE_UINT32 :
VSTYPE_FLOAT :
VSTYPE_LONG :
VSTYPE_ULONG :
VSTYPE_VSTRING :
VSTYPE_STRUCT :
VSTYPE_CHAR :
VSTYPE_COLOR :
VSTYPE_RECT :
VSTYPE_FONT :
VSTYPE_TIME :
VSTYPE_UUID :
VSTYPE_STATICID :

For struct attribute:

VSTYPE_BOOL :
VSTYPE_INT8 :
VSTYPE_UINT8 :
VSTYPE_INT16 :
VSTYPE_UINT16 :
VSTYPE_INT32 :
VSTYPE_UINT32 :
VSTYPE_FLOAT :
VSTYPE_LONG :
VSTYPE_ULONG :
VSTYPE_CHAR :
VSTYPE_COLOR :
VSTYPE_RECT :
VSTYPE_FONT :
VSTYPE_TIME :
VSTYPE_UUID :

Maping between data type and xml

VSTYPE_BOOL : xsd:boolean
VSTYPE_INT8 : xsd:byte
VSTYPE_UINT8 : xsd:unsignedByte
VS_INT16 : xsd:short
VSTYPE_UINT16 : xsd:unsignedShort
VSTYPE_INT32 : xsd:int
VSTYPE_UINT32 : xsd:unsignedInt
VSTYPE_FLOAT : xsd:float
VSTYPE_LONG : xsd:long
VSTYPE_ULONG : xsd:unsignedLong
VSTYPE_LONGHEX : xsd:long
VSTYPE_ULONGHEX : xsd:unsignedLong
VSTYPE_VSTRING : xsd:string
VSTYPE_COLOR : xsd:unsignedLong
VSTYPE_RECT : xsd:string "left,top,right,bottom"
VSTYPE_FONT : xsd:string "height,size,charset,style,name"
VSTYPE_TIME : xsd:dateTime
VSTYPE_CHAR : xsd:string

VSTYPE_UUID : xsd:string
VSTYPE_STATICID : xsd:unsignedLong
VSTYPE_CHARPTR : xsd:string

15.3.1 Create Web service using LUA

Here directly use the starcore service created in remotecall chapter.

```
require "libstarcore"
initstarcore(cle)
if libstarcore._InitCore(true,true,false,true,"",0,"",3008) == false then
   return
end
get service group 0, and create service
SrvGroup = libstarcore:_GetSrvGroup()
--create service
load depended service
SrvGroup:_ImportServiceWithPath("..\\..\\service\\script","RemoteCallServer",true)
SrvGroup:_CreateService( "","WebServiceCallServer", "123",5,0,0,0, 0,0,"B07427AF-3C8B-4e88-9F06-535831EF46EF" )
Service = SrvGroup:_GetService("root","123")
create service item
Service:_CreateSysRootItem("TestItem","")
active service item
Service:_ActiveSysRootItem("TestItem")
SrvItem = Service:_GetSysRootItem( "TestItem" )

a = Service.TestClass:_NewGlobal(SrvItem)
a._Name = "TestObject"

a._WebServiceFlag=true

print( "Server Start ok....")
Message loop
function ExitProc()
   if libstarcore._KeyPress() == 27 then
      return true
   end
   return false
end

libstarcore._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

Run

starapp -w 3040 -e XXXX.lua

15.3.2 Get WSDL of WebService

from url:

http://127.0.0.1:3040/wsdl

or

http://127.0.0.1:3040/__WebServiceCallServer/wsdl

also can use script or C function GetWsdl

Example of wsdl is as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions targetNamespace="urn:starcore-WebServiceCallServer" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="urn:starcore-WebServiceCallServer" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:MIME="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:DIME="http://schemas.xmlsoap.org/ws/2002/04/dime/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <xsd:schema targetNamespace="urn:starcore-WebServiceCallServer">
      <xsd:element name="TestClassGetRemoteObjectreq">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="tns:SOAPClassOfParaObject" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="SOAPClassOfParaObject">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Para1" type="xsd:int" />
            <xsd:element name="Para2" type="xsd:string" />
            <xsd:element name="Para3" type="xsd:float" />
            <xsd:element ref="tns:SOAPStructOfParaStruct" />
            <xsd:element name="Para5" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="SOAPStructOfParaStruct">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Para1" type="xsd:int" />
            <xsd:element name="Para2" type="xsd:float" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="TestClassGetRemoteObjectrsp">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="tns:SOAPClassOfParaObject" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <message name="coreempty" />
  <message name="coreerror" />
  <message name="TestClassGetRemoteObjectrequest">
    <part name="parameter" element="tns:TestClassGetRemoteObjectreq" />
  </message>
  <message name="TestClassGetRemoteObjectresponse">
    <part name="parameter" element="tns:TestClassGetRemoteObjectrsp" />
  </message>
  <portType name="TestObjectPortType">
    <operation name="GetRemoteObject">
      <input message="tns:TestClassGetRemoteObjectrequest" />
      <output message="tns:TestClassGetRemoteObjectresponse" />
    </operation>
  </portType>
  <binding name="TestObject" type="tns:TestObjectPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="GetRemoteObject">
      <soap:operation style="document" soapAction="urn:GetRemoteObject" />
```

```
        <input>
          <soap:body use="literal" />
        </input>
        <output>
          <soap:body use="literal" />
        </output>
      </operation>
    </binding>
    <service name="WebServiceCallServer">
      <port name="TestObject" binding="tns:TestObject">
        <soap:address location="http://127.0.0.1:3040/__WebServiceCallServer/webservice/TestObject" />
      </port>
    </service>
</definitions>
```

16 Starcore application packing

16.1 starcore packing

Using starsrvpack, you can pack application and publish it on web site.

Examples in directoryexamples\service.publish

16.1.1 Packing applications

write config file, and then use starsrvpack to pack. For example,

remotecall_lua

```
<?xml version="1.0" encoding="utf-8" ?>
<srpproject>
  <option>
    <name>remotecall_lua</name>
    <output></output>
    <script>lua</script>
  </option>
  <exec>
    <file name="../comm.basic/remotecall.lua/test_server.lua" start="true" />
  </exec>
  <depend />
  <static />
  <dyna />
</srpproject>
```

remotecall_python

```
<?xml version="1.0" encoding="utf-8" ?>
<srpproject>
  <option>
    <name>remotecall_python</name>
    <output></output>
```

```
      <script>python</script>
   </option>
   <exec>
      <file name="../comm.basic/remotecall.python/test_server.py" start="true"/>
   </exec>
   <depend />
   <static />
   <dyna />
</srpproject>
```

Packing:

starsrvpack remotecall_lua.srprj -i

starsrvpack remotecall_python.srprj -i

test:

starapp -e remotecall_python.srb

starapp -e remotecall_lua.srb

16.1.2 Packing applications developed with c/c++

examples\service.publish\webservice.c

Applications developped with c/c++, should be compiled into share libraries.

Here takes websevice as an example:

The share library should exports two function which prototype is defined in vsopenapi.h.

VS_BOOL StarCoreService_Init(class ClassOfStarCore *StarCore);

Init function, is called when share library is loaded.

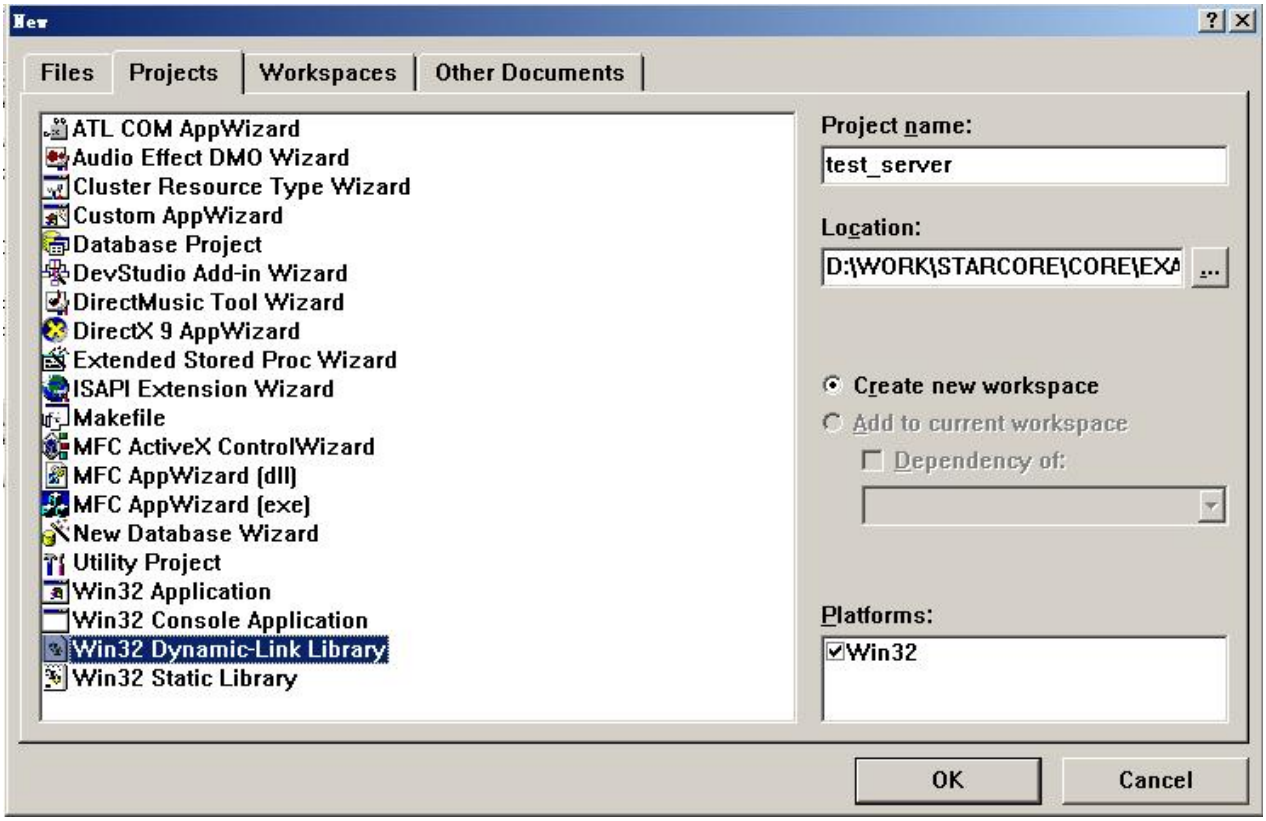
Using SRPControlInterface = StarCore ->GetControlInterface() ->Dup(); to get control interface, and then to get other interface.

void StarCoreService_Term(class ClassOfStarCore *StarCore);

Terminating function, called when share library is unloaded.

16.1.2.1Win32

16.1.2.1.1 Create project(VC6)



16.1.2.1.2 Create and edit source file

Modify as follows. You can compare it with the previous example in 8.2.1.4.

```
#include "vsopenapi.h"
extern "C"{
        #include "vs_shell.h"
};

//------------------------------------------------------------------------------
/*
VS_HANDLE hDllInstance;
VSCore_RegisterCallBackInfoProc RegisterCallBackInfoProc;
VSCore_InitProc VSInitProc;
VSCore_TermProc VSTermProc;
VSCore_QueryControlInterfaceProc QueryControlInterfaceProc;
*/
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
static class ClassOfSRPInterface *SRPInterface = NULL;


static VS_INT32 GetNumber(void *Object,VS_INT32 Para)
{
   printf( "Remote Call Number [%d]\n ",Para);
        return Para + 1;
}

static VS_CHAR *GetString(void *Object,VS_CHAR *Para)
{
        static VS_CHAR CharBuf[128];
```

```
    printf( "Remote Call String [%s]\n",Para);
        sprintf(CharBuf,"%sasdfsaf",Para);
    return CharBuf;
}

//---------------------------------------------------------------------------
/*
int main(int argc, char* argv[])
{
    VS_CHAR ModuleName[512];
    VS_UUID ServiceID,ClassID;
    void *AtomicClass,*Object,*GetNumber_AtomicFunction,*GetString_AtomicFunction;
    VS_CHAR *ErrorInfo;

    SRPControlInterface = NULL;
    BasicSRPInterface = NULL;
  //-------------------------------------------------------
    sprintf(ModuleName,"libstarcore%s",VS_MODULEEXT);
  hDllInstance = vs_dll_open( ModuleName );
    if( hDllInstance == NULL ){
        printf("load library [%s] error....\n",ModuleName);
        return -1;
  }
  RegisterCallBackInfoProc = (VSCore_RegisterCallBackInfoProc)vs_dll_sym( hDllInstance,
VSCORE_REGISTERCALLBACKINFO_NAME );
  VSInitProc = (VSCore_InitProc)vs_dll_sym( hDllInstance, VSCORE_INIT_NAME );
  VSTermProc = (VSCore_TermProc)vs_dll_sym( hDllInstance, VSCORE_TERM_NAME );
  QueryControlInterfaceProc = (VSCore_QueryControlInterfaceProc)vs_dll_sym( hDllInstance,
VSCORE_QUERYCONTROLINTERFACE_NAME );
    //--init star core
    RegisterCallBackInfoProc(MsgCallBack,0);
  VSInitProc( true, true, "", 0, "", 3008,NULL);

    printf("init starcore success\n");
*/

VS_BOOL SRPAPI StarCoreService_Init(class ClassOfStarCore *StarCore)
{
    VS_UUID ServiceID,ClassID;
    void *AtomicClass,*Object,*GetNumber_AtomicFunction,*GetString_AtomicFunction;
    VS_CHAR *ErrorInfo;

    SRPControlInterface = StarCore ->GetControlInterface() ->Dup();
  BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);

    BasicSRPInterface ->StringToUuid("E124266B-C66D-4fc3-B287-6D0B4C5F90AD",&ServiceID);
    BasicSRPInterface ->CreateService("","WebServiceCallServer",&ServiceID,"123",0,0,0,0,0,0);
    SRPInterface = BasicSRPInterface ->GetSRPInterface("WebServiceCallServer","root","123");
    SRPInterface ->CreateSysRootItem("TestItem","",NULL,NULL);
    SRPInterface ->ActiveSysRootItem( "TestItem" );
    //---Create Atomic Class, for define function, no attribute
    AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClass",NULL, NULL,&ErrorInfo);
    GetNumber_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetNumber","VS_INT32
GetNumber(VS_INT32 Para);", NULL,&ErrorInfo,VS_FALSE,VS_FALSE);
    GetString_AtomicFunction = SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetString","VS_CHAR
*GetString(VS_CHAR *Para);", NULL,&ErrorInfo,VS_FALSE,VS_FALSE);
  //---Set Function Address
    SRPInterface -> SetAtomicFunction(GetNumber_AtomicFunction,(void *)GetNumber);
    SRPInterface -> SetAtomicFunction(GetString_AtomicFunction,(void *)GetString);

    printf("create TestObject for webservice..\n");
    SRPInterface ->GetAtomicID(AtomicClass,&ClassID);
    Object = SRPInterface ->MallocObjectL(&ClassID,0,NULL);  //---need not alloc global object
    SRPInterface ->SetName(Object,"TestObject");
    SRPInterface ->SetWebServiceFlag(Object,VS_TRUE);

    return VS_TRUE;
```

```
}

/*
	BasicSRPInterface ->SetWebServerPort(NULL,3040,100,200);

	printf("finish,enter message loop..\n");
	while( 1 ){
		VS_INT32 Ch;
		Ch = vs_kbhit();
		if( Ch == 27 )
			break;
		if( SRPControlInterface -> SRPDispatch(VS_FALSE) == VS_FALSE ){
			SRPControlInterface -> SRPIdle();
			SRPControlInterface -> SRPDispatch(VS_TRUE);
		}
	}
	SRPControlInterface ->Release();
	BasicSRPInterface ->Release();
	VSTermProc();
	vs_dll_close(hDllInstance);
	return 0;
}
*/

void SRPAPI StarCoreScript_Term()
{
	SRPControlInterface ->Release();
	BasicSRPInterface ->Release();
	SRPInterface ->Release();
}
```

16.1.2.2 linux

```
#*******************************************************************************
#
# Makefile for StarCore.
# www.srplab.com
#*******************************************************************************
DEBUG		:= YES
PROFILE		:= NO
#*******************************************************************************
CC	:= gcc
CXX	:= g++
LD	:= g++
AR	:= ar
RANLIB := ranlib

DEBUG_CFLAGS	:= -Wall -Wno-format -g -DDEBUG -DENV_LINUX
RELEASE_CFLAGS	:= -Wall -Wno-unknown-pragmas -Wno-format -O3 -DENV_LINUX

LIBS := -ldl -lpthread -lrt
EXTRA_LIBS := /usr/lib/libstarlib.a /usr/lib/libuuid.a

DEBUG_CXXFLAGS	:= ${DEBUG_CFLAGS}
RELEASE_CXXFLAGS := ${RELEASE_CFLAGS}

DEBUG_LDFLAGS	:= -g
RELEASE_LDFLAGS	:=

ifeq (YES, ${DEBUG})
  CFLAGS	:= ${DEBUG_CFLAGS}
  CXXFLAGS	:= ${DEBUG_CXXFLAGS}
  LDFLAGS	:= ${DEBUG_LDFLAGS}
else
```

```
  CFLAGS      := ${RELEASE_CFLAGS}
  CXXFLAGS    := ${RELEASE_CXXFLAGS}
  LDFLAGS     := ${RELEASE_LDFLAGS}
endif

ifeq (YES, ${PROFILE})
  CFLAGS   := ${CFLAGS} -pg -O3
  CXXFLAGS := ${CXXFLAGS} -pg -O3
  LDFLAGS  := ${LDFLAGS} -pg
endif

#*******************************************************************************
# Makefile code common to all platforms
#*******************************************************************************

CFLAGS   := ${CFLAGS}   ${DEFS}
CXXFLAGS := ${CXXFLAGS} ${DEFS}

#*******************************************************************************
# include source and paths
#*******************************************************************************

INCS_T := /usr/include/starcore
INCS   = $(addprefix -I,$(INCS_T))

TEST_SERVER_CXXSRCS := test_server.cpp

#*******************************************************************************
TEST_SERVER_CXXOBJS := $(TEST_SERVER_CXXSRCS:%.cpp=%.o)

#*******************************************************************************
CXXOBJS := ${TEST_SERVER_CXXOBJS}
COBJS :=

EXEC_TEST_SERVER_OBJS := ${TEST_SERVER_CXXOBJS}

#*******************************************************************************
# Targets of the build
#*******************************************************************************
OBJS_PATH = .

EXEC_TEST_SERVER := ${OBJS_PATH}/test_server.so

all: ${EXEC_TEST_SERVER}

#*******************************************************************************
# Output
#*******************************************************************************

${EXEC_TEST_SERVER}: ${EXEC_TEST_SERVER_OBJS}
	${LD} -shared -o $@ ${LDFLAGS} ${EXEC_TEST_SERVER_OBJS} ${LIBS} ${EXTRA_LIBS}


#*******************************************************************************
# common rules
#*******************************************************************************

${CXXOBJS} :
	${CXX} -fPIC ${CXXFLAGS} ${INCS} $< -o $@ -c $*.cpp

${COBJS} :
	${CC} -fPIC ${CFLAGS} ${INCS} -o $@ -c $*.c

clean:
	-rm -f ${CXXOBJS} ${COBJS} ${EXEC_TEST_SERVER}
```

16.1.2.3Packing and testing

For binary module of C/C++, it is different on win32 and linux. Therefore should set startup file seperately. The project is as:

```
<?xml version="1.0" encoding="utf-8" ?>
<srpproject>
   <option>
      <name>webservice_c</name>
      <output></output>
      <start>test_server_RemoteCallServer.lua</start>
      <script>lua</script>
   </option>
   <exec>
      <file name="webservice.c/test_server.dll" start="true" ostype="win32"/>
      <file name="webservice.c/test_server.so" start="true" ostype="linux"/>
   </exec>
   <depend/>
   <static />
   <dyna />
</srpproject>
```

Run

starsrvpack webservice_c.srprj -i

Packing to single file.

If only pack for win32, the command is starsrvpack webservice_c.srprj -i -s win32

test:

starapp -e webservice_c.srb

Upload webservice_c.srb to web site, publishing for win32 and linux is finished

16.2 Data files in package

Examples in directoryexamples\service.publish\packdata

In package, there are three type of files:

```
   <exec>
      <file name="../comm.basic/remotecall.python/test_server.py" toutf8="true/false" />
   </exec>
   <static />
   <dyna />
```

exec, executable file, usually is lua/python script file or dll/so share library file.

static : static file, these files will be downloaded before loading service.

dyna: dynamic files, these files does not download before service start. They will be downloaded on demand.

For dynamic file, if being packed into single file, they are same as static files.

If toutf8 = true,then the file will be changed to utf8 when packing. If start file is utf8, then cle will convert it to local coding before service is started.

For other files, cle does not convert, the application should use the function defined in binbuf interface.

About how applications to get data from package is list below.

16.2.1 pack to single file

After starcore loads the application, it will set interface class ClassOfSRPMemoryFileInterface, which points to the files in the package. Application may use GetEnvMemoryFile to get the interface.

Examples:

```
<?xml version="1.0" encoding="utf-8" ?>
<srpproject>
   <option>
      <name>testsingle_service</name>
      <output></output>
      <script>lua</script>
   </option>
   <exec>
      <file name="testsingle_service.lua" start="true"/>
      <file name="e1.txt"/>
      <path name="aaa">
         <file name="e2.txt"/>
      </path>
   </exec>
   <depend />
   <static>
      <file name="s1.txt"/>
      <file name="s2.txt"/>
   </static>
   <dyna>
      <file name="d1.txt"/>
      <file name="d2.txt"/>
   </dyna>
</srpproject>
```

16.2.1.1C

16.2.1.1.1 Win32

16.2.1.1.1.1 Create project(VC6)

skip

16.2.1.1.1.2 Create and edit source file

```
#include "vsopenapi.h"
extern "C"{
        #include "vs_shell.h"
};

//-------------------------------------------------------------------------------
VS_HANDLE hDllInstance;
VSCore_RegisterCallBackInfoProc RegisterCallBackInfoProc;
VSCore_InitProc VSInitProc;
VSCore_TermProc VSTermProc;
VSCore_QueryControlInterfaceProc QueryControlInterfaceProc;
static class ClassOfSRPControlInterface *SRPControlInterface = NULL;
static class ClassOfBasicSRPInterface *BasicSRPInterface = NULL;
```

```
static VS_ULONG MsgCallBack( VS_ULONG ServiceGroupID, VS_ULONG uMsg, VS_ULONG wParam, VS_ULONG
lParam, VS_BOOL &IsProcessed, VS_ULONG Para )
{
   switch( uMsg ){
   case MSG_VSDISPMSG :
       case MSG_VSDISPLUAMSG :
              printf("[core]%s\n",(VS_CHAR *)wParam);
              break;
   case MSG_DISPMSG :
       case MSG_DISPLUAMSG :
              printf("%s\n",(VS_CHAR *)wParam);
      break;
       case MSG_EXIT :
              break;
   }
   return 0;
}

void PrintFile(class ClassOfSRPMemoryFileInterface *MemoryFileInterface,VS_CHAR *FileName)
{
       VS_CHAR Buf[128];
       VS_ULONG Size;

       Size = MemoryFileInterface ->GetSize(FileName);
       MemoryFileInterface ->Read(FileName,(VS_UINT8 *)Buf);
       Buf[Size] = 0;
       printf("File %s : size=%d, : %s\n",FileName,Size,Buf);
}

//---------------------------------------------------------------------------
int main(int argc, char* argv[])
{
       class ClassOfSRPMemoryFileInterface *MemoryFileInterface;
        class ClassOfSRPInterface *SRPInterface;
       VS_CHAR ModuleName[512];

       SRPControlInterface = NULL;
       BasicSRPInterface = NULL;
   //---------------------------------------------------------------------------
       sprintf(ModuleName,"libstarcore%s",VS_MODULEEXT);
   hDllInstance = vs_dll_open( ModuleName );
       if( hDllInstance == NULL ){
              printf("load library [%s] error....\n",ModuleName);
              return -1;
   }
   RegisterCallBackInfoProc = (VSCore_RegisterCallBackInfoProc)vs_dll_sym( hDllInstance,
VSCORE_REGISTERCALLBACKINFO_NAME );
   VSInitProc = (VSCore_InitProc)vs_dll_sym( hDllInstance, VSCORE_INIT_NAME );
   VSTermProc = (VSCore_TermProc)vs_dll_sym( hDllInstance, VSCORE_TERM_NAME );
   QueryControlInterfaceProc = (VSCore_QueryControlInterfaceProc)vs_dll_sym( hDllInstance,
VSCORE_QUERYCONTROLINTERFACE_NAME );
       //--init star core
       RegisterCallBackInfoProc(MsgCallBack,0);
   VSInitProc( true, true, "", 0, "", 3008,NULL);

       printf("init starcore success\n");

       SRPControlInterface = QueryControlInterfaceProc();
   BasicSRPInterface = SRPControlInterface ->QueryBasicInterface(0);

       if( BasicSRPInterface -> RunFromUrl("test.srb",VS_FALSE,VS_TRUE) != SRPLOADPROCESS_OK ){
              SRPControlInterface ->Release();
              BasicSRPInterface ->Release();
              VSTermProc();
              vs_dll_close(hDllInstance);
              return -1;
       }
```

```
        SRPInterface = BasicSRPInterface ->GetSRPInterface(BasicSRPInterface->QueryActiveService(NULL),"root","123");
        MemoryFileInterface = SRPInterface ->GetEnvMemoryFile();
        PrintFile(MemoryFileInterface,"e1.txt");
        PrintFile(MemoryFileInterface,"e2.txt");
        PrintFile(MemoryFileInterface,"aaa\\e2.txt");
        PrintFile(MemoryFileInterface,"s1.txt");
        PrintFile(MemoryFileInterface,"s2.txt");
        PrintFile(MemoryFileInterface,"d1.txt");
        PrintFile(MemoryFileInterface,"d2.txt");

        SRPControlInterface ->Release();
        BasicSRPInterface ->Release();
        VSTermProc();
        vs_dll_close(hDllInstance);
        return 0;
}
```

16.2.1.1.1.3 Compile and run

starsrvpack testsingle_pack.srprj -s win32 -i

Run:

testsingle

16.2.1.1.2 linux

Write Makefile

16.2.2 Pack to directory

Pack to directory, static data file will be download into current directory. For dynamic files, which should be associated with objects to trigger the download process.

Calling interface function GetStaticDataEx with token set to file name

You can use service "SRPFSEngine", may simplify the procedure.

Packing xml config :

```
<?xml version="1.0" encoding="utf-8" ?>
<srpproject>
   <option>
      <name>testdir_service</name>
      <output></output>
      <start>testdir_service.lua</start>
      <script>lua</script>
   </option>
   <exec>
      <file name="testdir_service.lua" />
      <file name="e1.txt"/>
      <path name="aaa">
         <file name="e2.txt"/>
      </path>
   </exec>
   <depend />
   <static>
      <file name="s1.txt"/>
```

```
      <file name="s2.txt"/>
   </static>
   <dyna>
      <file name="d1.txt"/>
      <path name="bbb">
         <file name="d2.txt"/>
      </path>
   </dyna>
</srpproject>
```

Includeing two dynamic files : d1.txt and bbb/d2.txt

The following code loads "SRPFSEngine" service:

```
require "libstarcore"

if libstarcore._InitCore(true,true,false,true,"",0,"",0) == false then
   return
end

SrvGroup = libstarcore:_GetSrvGroup()
--Create service
SrvGroup:_ImportService("SRPFSEngine")
SrvGroup:_CreateService( "","test", "123",5,0,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )
Service = SrvGroup:_GetService("root","123")

Create a virtual disk
VDisk=Service.DriveClass:_New()
VDisk._Name="VDisk"
Load network file, namely, the dynamic files in the project
VDisk:Lua_LoadWebFile("d1.txt","d1.txt")
VDisk:Lua_LoadWebFile("d2.txt","bbb\\d2.txt")
start download
VDisk:Lua_DownLoad("d1.txt")
VDisk:Lua_DownLoad("d2.txt")
whether the download is finished.
function ExitProc()
   if VDisk:Lua_GetFileStatus("d1.txt") == 0 and VDisk:Lua_GetFileStatus("d2.txt")==0 then
      print("download finish.....")
      return true
   end
   return false
end

libstarcore._MsgLoop( ExitProc )

libstarcore._ModuleExit()
```

16.2.2.1.1.1 Run

starsrvpack testdir_pack.srprj

Copy directory testdir_service to website

Run:

starapp -e "http://XXX/testdir_service"

17 How to register

17.1 Difference between two versions

Share version provides basic and popular functions and has limitions about the total number of cle objects being created. **From version 2.5.1, The maximum number of objects can be created for unregistered version is 48, which meet most application scenarios. The objects include objects created using the "_New" function or objects returned from other scripts that encapsulate native data types. You can release unused objects with "_Free" function to make the total number of objects less than 48. Function "GetObjectNum" is added to get current object number.**

We recommend that you use the registered version.

Registered version provides some special features, such as object event registration, atomic object operation, and supports service description languag, conversion between service and xml. Details of the difference is listed below:

Functions	Explanation
SetSysEvent	*Object system event process*
RegSysEventFunction	*Object system event process for c/c++ and scripts.*
Active	*Active object*
ActiveClient	*Active object*
CreateNameScript	*Obejct name script, which is lua, may be syncto client.*
CreateNameScriptEx	
SetPrivateValue	*Object private variable*
SaveToBuf	*Object serialization*
SaveToFile	
Redirect	*Service redirec*
SetClientQos	*Set client Qos*
GetClientInfo	*Get client info*
RegClientOpFunction	*Register client operation callback*
XmlToSysRootItem	*Service parse and unparse*
XmlToObject	
ServiceToXml	
SysRootItemToXml	
ObjectToXml	
CreateAtomicStruct	*Atomic functions*
CreateAtomicObject	

CreateAtomicAttachAttribute	
CreateAtomicAttribute	
CreateAtomicFuncRetAttribute	
CreateAtomicFuncParaAttribute	
CreateAtomicStructAttribute	
SetAtomicAttributeLength	
SetAtomicAttributeStruct	
SetAtomicAttributeCombobox	
SetAtomicAttributeSyncFlag	
CreateAtomicScript	
CreateAtomicFunction	
CreateAtomicOvlFunction	
CreateAtomicLuaFunction	
CreateAtomicFunctionEx	
CreateAtomicInEvent	
CreateAtomicOutEvent	
SetLog	*Objec toperation logs. used for high avaliablity*
SetLogFile	
GetLogFile	
ClearLog	
ApplyLog	
SetNameBoolValue	*Object name value*
SetNameIntValue	
SetNameFloatValue	
SetNameBinValue	
SetNameStrValue	
SetNameTimeValue	

17.2 Buy and Register

Please visit http://www.srplab.com/products.htm to buy a license of cle. After a little pay, you will receive an email with a registration code. Then,

you may run starregister in cmd line window, for example:

Run: starregister [register code]

You can also use "_SetRegisterCode" for script or " SetRegisterCode " for c/c++ applications dynamically.

You should not redistribute the registration code to others, for any purpose.

17.3 use cle in application on other computers.

Using "_SetRegisterCode" function to authorize the application, in order to run on other computers.

for example:

```
import com.srplab.www.starcore.*;

public class java_call{
	public static void main(String[] args){
		StarCoreFactory starcore= StarCoreFactory.GetFactory();
		starcore._InitCore(true,true,false,true,"",0,"",0);   // should init starcore before call  _PreAuthorize
		starcore._SetRegisterCode("XXXXXXXXXXXXXXXXXXXX",false);
		StarServiceClass Service=starcore._InitSimple1("test","","123",0,0);
		....
		starcore._ModuleExit();
   }
}
```

18 Distributing cle with your products

CLE is permited to be distributed with your products. You should include the following files in your product installing package according to the languages used

win32:

libstarcore.dll Located at directory X:\windows\system32

java:star_java.dll Located at directory X:\windows\system32

python:libstarpy.pyd Located at directory python27/DLLS

python:libstar_python34.pyd Located at directory python34/DLLS

python:libstar_python35.pyd Located at directory python35/DLLS

ruby:libstar_ruby.so Located at directory X:\srplab\libs

c#:Star_csharp/ Star_csharp4/ Star_csharp45.dll/ Star_csharp451.dll Located at directory X:\srplab\libs

linux:

libstarcore.so Located at directory /usr/lib

java:libstar_java.so Located at directory /usr/lib

python:libstarpy.so Located at directory of python27

python:libstar_python34.so Located at directory of python34

python:libstar_python35.so Located at directory of python35

ruby:libstar_ruby.so Located at directory /usr/lib

For c++ and lua, you need only include libstarcore.dll or libstarcore.so

19 Q&A

19.1 Create network server or client failed on android

Please make sure port number is unused and permission is set in androidmanifest.xml file as follow :

<uses-permission android:name="android.permission.INTERNET" />

19.2 load share library failed

The error may be occurred on linux. Its reason may be some symbols can not be located. Please uses ldd –r method to check the share library.

19.3 RuntimeBinderException of using dynamic in c#

When using dynamic object in c# for cle object, the following error will be printed :

```
A first chance exception of type 'Microsoft.CSharp.RuntimeBinder.RuntimeBinderException' occurred in
Unknown Module.
A first chance exception of type 'Microsoft.CSharp.RuntimeBinder.RuntimeBinderException' occurred in
Microsoft.CSharp.ni.dll
```

Do not need warry about the exceptions.

19.4 Java init failed on MAC OSX

please check file " /Library/Java/JavaVirtualMachines/XXX.jdk/Contents/Info.plist" to see whether it contains "JNI" or not,

```
<key>JVMCapabilities</key>
<array>
    <string>CommandLine</string>
    <string>JNI</string>
</array>
```

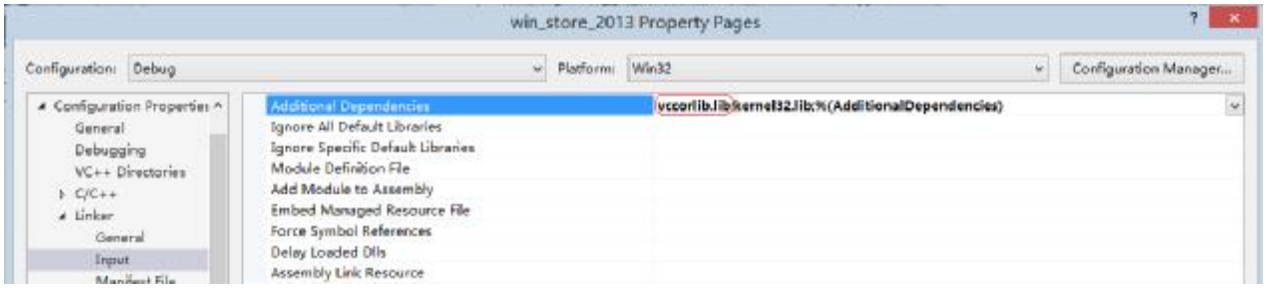
19.5 vccorlib_lib_should_be_specified_before_msvcrt_lib_to_linker

Using visual studio 2013 or later, if compiled with following message

1>vccorlibd.lib(compiler.obj) : error LNK2038: mismatch detected for 'vccorlib_lib_should_be_specified_before_msvcrt_lib_to_linker': value '1' doesn't match value '0' in MSVCRTD.lib(appinit.obj)

1>vccorlibd.lib(compiler.obj) : error LNK2005: ___crtWinrtInitType already defined in MSVCRTD.lib(appinit.obj)

Add vccorlib.lib to the project may solve this problem, like this,



19.6 Init ruby or call ruby raw function fails from java command on linux

On linux, for stack reason, using java command to run java files which call ruby raw function may be get abnormal result.

Please try different java version. Or using starapp to run java files.

19.7 Init ruby or call ruby raw function fails from java command on linux

On linux, for stack reason, using java command to run java files which call ruby raw function may be get abnormal result.

Please try different java version. Or using starapp to run java files.

19.8 Init python3.6 interface failed on windows

Got error message :

[core]load library [star_python36.so] error....[126]

Please add python 3.6 to the path variable.

19.9 onDestroy event on the android platform

Because script engine can not be unloaded completely, it is better for the app to exit when receive onDestroy event.

```
@Override
protected void onDestroy() {
    super.onDestroy();
    System.exit(0);
}
```

19.10 Problems when installing 32bit and 64bit ruby Simultaneously on windows platform

On windows platform, RubyInstaller uses same registry key for 64bit package and 32bit package.
If 64bit is installed after 32bit, there is no information about 32bit package. In this case, starcore cannot find the share library of ruby core, which results run ruby script failed.
If you do so, you need to specify the ruby shared library. For example,

```
_SetScript("ruby","", "-m C:\\Ruby24\\bin\\msvcrt-ruby240.dll")
SrvGroup._InitRaw("ruby",Service);
```

19.11 Load ruby share library failed for version 2.4 or above on windows platform

For ruby version 2.4 or above, the depended libraries are located in "X:\Ruby24\bin\ruby_builtin_dlls".
If it is not in the path, the ruby core share library will be loaded failed.
Please add it to path envoriment variable.

19.12 Specifing ruby runtime version

For windows desktop, you can set version before initialize ruby script , for example,

```
_SetScript("ruby","", "-v 2.3.0")
SrvGroup._InitRaw("ruby",Service);
```

Or using starapp

```
Starapp –ipara "-v 2.3.0" –e xxx.rb ?script=ruby
```

-v parameter is only valid on windows desktop. For linux, libruby.so is always loaded.

19.13 Print function of python and ruby in thread

When calling "print" function in python thread, _SRPLock and _SRPUnLock must be used.
When calling "print" or "puts" function in ruby thread, _SRPLock and _SRPUnLock must be used.

20 About srplab

If there are any questions, please contact using email freely:

srplab.cn@hotmail.com