



# **Common Language Extension (Programmer's Guide)**

# Contents

<b>Contents</b>	<b>2</b>
1 Introduction	7
2 Install CLE	7
2.1 Install	7
2.1.1 windows & linux	7
2.1.2 android	8
2.2 Programming environment	8
2.2.1 Install starcore	8
2.2.2 C/C++	9
2.2.3 Install python	9
2.2.4 Install php	12
2.2.5 Install java	13
2.2.6 Install .NET(skip)	13
2.2.7 Debug and compile	14
2.2.7.1 Compile	14
2.2.7.2 Debug	18
2.2.8 Run	19
2.3 starcore tools	19
2.3.1 starapp:cle application running environment,which can load sharelib, lua/java/python/csharp shcipts.	19
2.3.2 star2c/star2h,generate header file and code skeleton.	21
2.3.3 starsrvinstinfo	22
2.3.4 starsrvreg	22
2.3.5 starsrvparse/starsrvunparse	22
2.3.6 starsrvpack	22
2.3.7 starsrvcheck	23
3 Init CLE	23
3.1 Architecture of CLE	23
3.2 Init Cle using C/C++	24
3.2.1 init type 1, create service group directly	24
3.2.2 init type 2, create service directly	25
3.2.3 init type 3, load libstarcore sharelib manually	25
3.2.4 init type 4, link with libstarcore sharelib	26
3.3 Init Cle using lua	26
3.3.1 init type 1, create service group directly	27
3.3.2 init type 2, create service directly	27
3.3.3 init type 3, create service step by step	27
3.4 Init Cle using python	27
3.4.1 init type 1, create service group directly	27
3.4.2 init type 2, create service directly	28
3.4.3 init type 3, create service step by step	28
3.5 Init Cle using php	28
3.5.1 init type 1, create service group directly	28
3.5.2 init type 2, create service directly	28
3.5.3 init type 3, create service step by step	29
3.6 Init Cle using java	29
3.6.1 init type 1, create service group directly	29
3.6.2 init type 2, create service directly	29
3.6.3 init type 3, create service step by step	30

3.7	Init Cle using csharp .....	30
3.7.1	init type 1, create service group directly .....	31
3.7.2	init type 2, create service directly .....	32
3.7.3	init type 3, create service step by step .....	32
3.8	Create object, define its attributes and functions .....	33
3.8.1	python .....	33
3.8.2	lua .....	33
3.8.3	php .....	33
3.8.4	java.....	34
3.8.5	c#.....	34
3.8.6	c++ .....	35
3.8.7	Call object's function.....	35
3.8.7.1	python .....	35
3.8.7.2	lua .....	35
3.8.7.3	php .....	36
3.8.7.4	java.....	36
3.8.7.5	c#.....	36
3.8.7.6	c++ .....	36
3.9	Message loop in CLE.....	37
3.9.1	c#/java.....	37
3.9.2	android .....	38
3.10	Programming problems needed pay attention.....	38
4	Develop common extension. ....	39
4.1	Common extension .....	40
4.1.1	Develop common extension using python .....	40
4.1.2	Develop common extension using lua .....	40
4.1.3	Develop common extension using java .....	40
4.1.4	Develop common extension using C++ .....	41
4.1.5	Develop common extension using C# .....	41
4.2	Call common extension using C/C++ .....	42
4.3	Call common extension using lua .....	43
4.4	Call common extension using python.....	43
4.5	Call common extension using PHP.....	43
4.6	Call common extension using java .....	44
4.7	Call common extension using C# .....	44
4.8	passing complex data structures between languages .....	45
4.8.1	Extension module to be called .....	47
4.8.1.1	Develop common extension using python .....	47
4.8.1.2	Develop common extension using lua .....	47
4.8.1.3	Develop common extension using java .....	48
4.8.1.4	Develop common extension using C++ .....	48
4.8.1.5	Develop common extension using C# .....	50
4.8.2	Call common extension using C/C++ .....	51
4.8.3	Call common extension using lua .....	52
4.8.4	Call common extension using python.....	52
4.8.5	Call common extension using PHP.....	53
4.8.6	Call common extension using java .....	54
4.8.7	Call common extension using C# .....	54
4.9	A more complicated example .....	55
4.9.1	java swing window(Callback function) .....	55
4.9.1.1	Common extension developed by java to create a window using swing .....	55
4.9.1.2	Call using python.....	57
4.9.1.3	Call using C++ .....	57
4.9.1.4	Call using c# .....	58

4.9.2	call jsoup.....	59
4.9.2.1	Common extension developed by java to create an interface object to jsoup.....	59
4.9.2.2	Call using python.....	60
4.9.2.3	Call using C/C++.....	61
4.9.2.4	Call using c#.....	61
4.9.3	c# form calls java.....	62
4.9.4	php call c#.....	63
4.10	Direct call sharelib.....	65
4.10.1	lua calls MessageBox.....	65
4.10.2	Java calls MessageBox.....	65
4.10.3	c# calls MessageBox.....	66
4.11	Mixed script language programming.....	67
4.11.1	Module to be called.....	67
4.11.1.1	lua.....	67
4.11.1.2	python.....	67
4.11.1.3	java.....	67
4.11.1.4	c#.....	68
4.11.2	C/C++ call other script.....	68
4.11.3	lua call other script.....	69
4.11.4	python call other script.....	69
4.11.5	php call other script.....	70
4.11.6	java call other script.....	70
4.11.7	c# call other script.....	70
4.12	ASP.NET call CLE extensions.....	71
5	CLE distributed function.....	72
5.1	TCP/UDP communication.....	72
5.1.1	TCP server.....	72
5.1.1.1	C.....	72
5.1.1.2	lua.....	76
5.1.1.3	python.....	77
5.1.1.4	php.....	78
5.1.1.5	java.....	79
5.1.1.6	c#.....	80
5.1.2	TCP client.....	81
5.1.2.1	C.....	81
5.1.2.2	lua.....	82
5.1.2.3	python.....	83
5.1.3	UDP server.....	84
5.1.3.1	C.....	84
5.1.3.2	lua.....	86
5.1.3.3	python.....	87
5.1.4	UDP client.....	88
5.1.4.1	C.....	88
5.1.4.2	lua.....	89
5.1.4.3	python.....	90
5.2	Remotecall.....	91
5.2.1	Create server side application.....	91
5.2.1.1	C.....	91
5.2.1.2	lua.....	94
5.2.1.3	python.....	95
5.2.2	Create client side application.....	96
5.2.2.1	Win32.....	96
5.2.2.2	linux.....	97
5.2.2.3	lua.....	98
5.2.2.4	python.....	98

5.2.3	Create and use stand alone starcore service .....	99
5.2.3.1	Create starcore service .....	99
5.2.3.2	Using starcore service .....	105
5.3	Remotecall-complicate data type .....	110
5.3.1	Create server side application .....	111
5.3.1.1	C.....	111
5.3.1.2	lua .....	114
5.3.1.3	python .....	115
5.3.2	Create client side application .....	116
5.3.2.1	Win32 .....	116
5.3.2.2	lua .....	117
5.3.2.3	python .....	118
5.3.3	Create and ust stand alone starcore service.....	119
5.3.3.1	Create starcore service .....	119
5.3.3.2	Export skeleton file .....	120
5.3.3.3	create module.....	121
5.3.4	called by LUA.....	121
5.3.5	called by Python.....	122
6	Webservice and http appli cation .....	122
6.1	Http&HttpServer.....	122
6.1.1	Http download.....	122
6.1.1.1	C.....	122
6.1.1.2	lua .....	124
6.1.1.3	python .....	125
6.1.2	Http upload .....	126
6.1.2.1	C.....	126
6.1.2.2	lua .....	127
6.1.2.3	python .....	127
6.1.3	Simple HttpServer.....	128
6.1.3.1	C.....	128
6.1.3.2	lua .....	130
6.1.3.3	python .....	131
6.1.4	HttpServer local request.....	132
6.1.4.1	C.....	132
6.1.4.2	lua .....	135
6.1.4.3	python .....	136
6.2	WebService.....	137
6.2.1	Create WebService .....	137
6.2.1.1	WebService object .....	137
6.2.1.2	lua .....	137
6.2.1.3	python .....	138
6.2.1.4	C.....	139
6.2.2	Get WSDL of WebService.....	143
6.2.3	WebService client(gsoap) .....	145
6.2.3.1	Win32 .....	145
6.2.4	WebService client(php) .....	146
6.2.5	Create and use stand alone starcore service. ....	147
6.2.5.1	Called by C .....	147
6.2.5.2	called by LUA.....	151
6.2.5.3	Called by python.....	152
6.3	WebService-compilcate data type.....	152
6.3.1	Create Web service using LUA .....	154
6.3.2	Get WSDL of WebService.....	155
7	Starcore config and management .....	156
7.1	config and manage starcore .....	156

7.1.1	Message between control node and starcore .....	157
7.1.1.1	starcore request restart .....	157
7.1.1.2	Control config starcore with new manager parameter .....	158
7.1.2	Message between manage node and starcore.....	158
7.1.2.1	echo(starcore -> Manager).....	158
7.1.2.2	get config parameter(starcore -> Manager) .....	159
7.1.2.3	Get server Url (starcore -> Manager).....	159
7.1.2.4	Get service Url(starcore -> Manager) .....	159
7.1.2.5	Register service(starcore -> Manager).....	159
7.1.2.6	Request to alloc cooperator(starcore -> Manager).....	160
7.1.2.7	Request free cooperator(starcore -> Manager) .....	160
7.1.2.8	Request alloc cooperator(Manager ->starcore).....	160
7.1.2.9	Request free cooperator(Manager ->starcore) .....	161
7.1.2.10	Get cooperator iformation(Manager ->starcore) .....	161
7.1.2.11	Modify config(Manager ->starcore) .....	161
7.1.2.12	Get config(Manager ->starcore).....	161
7.1.2.13	Get statistic(Manager ->starcore).....	162
7.1.3	Message between manager node and user or control node .....	162
7.1.3.1	echo(control ->manager).....	162
7.1.3.2	Create service(manager/user ->control).....	162
7.1.3.3	Restart service(manager/user ->control) .....	163
7.1.3.4	stop service(manager/user ->control) .....	163
7.1.3.5	Start service (manager/user ->control).....	163
7.1.3.6	Get service info(manager/user ->control) .....	164
7.1.3.7	Set output port(user ->control).....	164
7.1.3.8	Get config(user ->control) .....	165
7.1.3.9	Get control node url(user ->manager).....	165
7.2	Simple control node:starctl .....	165
7.3	simple manager node :starmconsole .....	166
7.4	Simple user node :starmuser .....	168
8	Starcore packing and publishing .....	169
8.1	starcore packing .....	169
8.1.1	Packing applications .....	169
8.1.2	Packing applications developed with c/c++ .....	170
8.1.2.1	Win32 .....	170
8.1.2.2	linux .....	173
8.1.2.3	Packing and testing .....	175
8.2	Data files in package.....	175
8.2.1	pack to single file .....	176
8.2.1.1	C.....	176
8.2.2	Pack to directory .....	178
8.3	Publish .....	180
9	How to register .....	180
9.1	Difference between two versions.....	181
9.2	Buy and Register.....	182
9.2.1	single computer license.....	182
9.2.2	Single service license .....	182
9.2.3	Single computer service license.....	183
9.2.4	starcore registry operation.....	183
9.3	Object status event.....	183
10	Redistributing with your products .....	185

## 1 *Introduction*

There are many programming languages. In addition to traditional language C/C++, script languages such as JAVA,PHP,PYTHON,LUA,C# are also introduced. Applications may be developed with proper and efficient language. For example, websites are usually programmed with PHP, GUI applications are developed 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 modules developed with one language are used in other languages easily, or how to re-use 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, PHP extension, JAVA extension, and so on. In order to write these extensions, not only to study interfaces of different languages, but also to perform real programming, in which many problems may be encountered lead to 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 module. The procedure will be repeated again. Therefore, a common extension development environment or middleware is expected.

There many languages to choice, which will introduce problems of how to perform mixed calls between languages. There are some solutions. For JAVA calls PYTHON may use JPYTHON, PHP calls JAVA may use PHP\_JAVA extension, etc. But developers have to study, understand and use each of them.

CLE is a common extension platform. Modules developed on CLE may be call by any other supported languages. In addition, CLE also presents a general pattern for mixed calls. CLE is cross-platform, which supports win32 and linux X86. Developing extension using interface presented by CLE, and calling these extensions also use 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 presented as a structured memory and a list of function pointers. Through mapping the structured memory and function pointers to different languages, the above idea is realized. CLE is a share library, which does not impose any restrictions on specific language, and may be used to develop kinds of distributed applications.

## 2 *Install CLE*

### 2.1 *Install*

#### 2.1.1 windows & linux

Supports Windows and linux, current version is for 32 bit operating system. On windows, the share library is named libstarcore.dll, and on linux, the sharelibrary is named libstarcore.so.

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

python, current supports version 2.4/2.5/2.6/2.7. In order to support python, python environment should be installed from <http://www.python.org>.

Pre-compiled interface library .pyd or .so is for python2.7. For other version, you should compile .pyd or .so manually, and copy the library file to python directory for extensions.

Starcore environment config file, is mainly used to config python or other script languages. If do not need, you may do not care about or change the config file.

File name: starenv.cfg.xml,

For win32, the file is located in C:\srplab.

For linux, the file is located in /etc/srplab.

File format :

```
<?xml version="1.0" encoding="utf-8" ?>
<StarCoreEvnConfig Python="">
Python:Python sharelib 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 '\', the it is 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

Install package for android is starcore.apk and starcorev7a.apk, you can download or install from web site <http://www.srplab.com>

android is supported from version 1.1.1, including two architecture "armeabi" and "armeabi-v7a". Alias name in cle for the two architecture are "android" and "androidv7a". Extension module for c/c++ service should add postfix "\_android.so" or "\_androidv7a.so".

For android, cle only supports java in calling c/c++, lua and python. If you want use python, you should install SL4A first.

CLE requires permission of network and sdcard, you should add the following in AndroidManifest.xml of the project.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-
permission>
```

## 2.2 Programming environment

### 2.2.1 Install starcore

For windows:

Running package:

starcore\_win32.1.0.1.exe



For linux :

```
rpm -e --nodeps starcore-1.0.1-1.i386.rpm
```

Examples of the document is

[starcore.examples.1.0.1.tar.gz](http://starcore.examples.1.0.1.tar.gz)

### 2.2.2 C/C++

Windows 2000,XP,2003,Vista,windows7.Development tools:CBuilder,VC series.

linux,gcc + gdb.

Header files:

vs\_shell.h:linux/windows applications may use functions defined in this header file.

vscoreshell.h:linux/windows migration, which includes functions about registry,string coding conversion functions.

vsopenapi.h,vsopencommtype.h,vsopencoredll.h,vsopendatatype.h,vsopenmemorydisk.h,vsopensyseventdef.h,vsopennetlink.h

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

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

```
#include "vsopensyseventdef.h"
```

On windows platform, libstarcore.dll may be used by default link by adding libstarcore.lib into your project.

On linux system, add -lstarcore condition 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 starcore to support python, you should install Python package. Default version supported is python2.7, wich can be modified to support other versions.

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 sharelib

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

for other version python, for example 2.6, you should modified the config as follow:

**linux:**

vi /etc/srplab/starencvfg.xml

Change as:

```
<?xml version="1.0" encoding="utf-8" ?>
<StarCoreEvnConfig Python="libpython2.6.so">
</StarCoreEvnConfig>
```

Into directory /usr/local/srplab/starcore/srpspy, modify file Makefile  
vi Makefile

LIBS := -ldl -lpthread -lrt -lpython2.7 change to:-lpython2.6

INCS\_T := /usr/include/starcore ../../source/python/include.linux change to /usr/include/python2.6

run : make

copy the library file created to python directory for extensions.

**win32:**

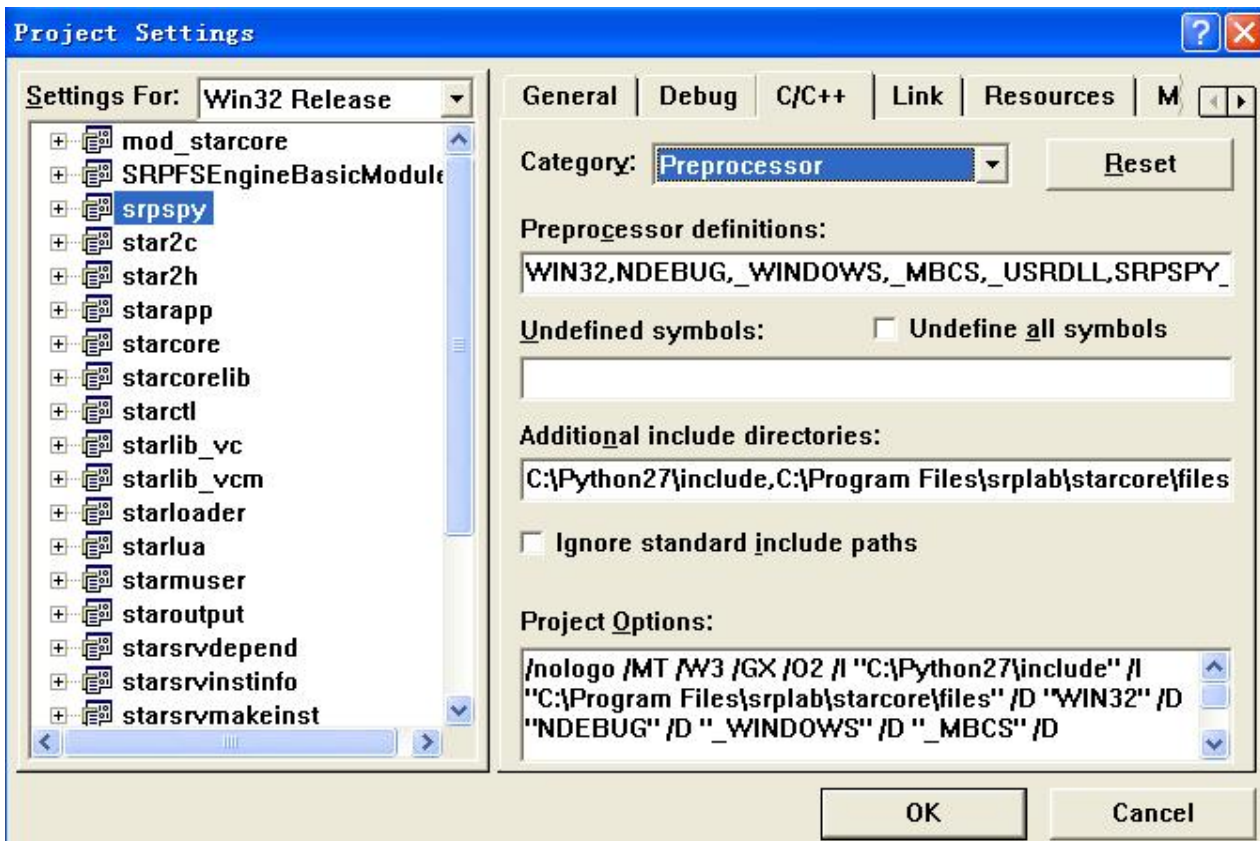
modify config file C:\srplab\starencvfg.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<StarCoreEvnConfig Python="python2.6.dll">
</StarCoreEvnConfig>
```

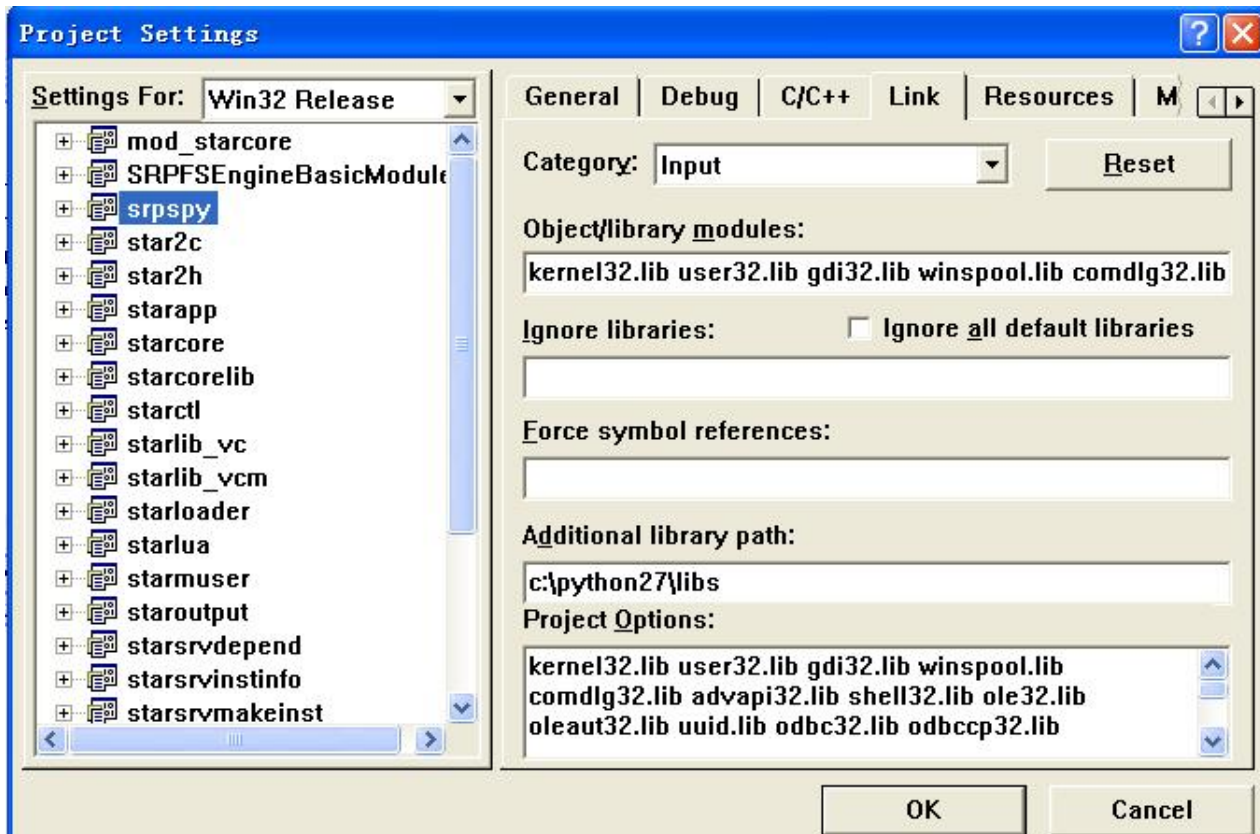
Into source code directory.

C:\rplab\starcore\srpspy

Open the project with VC, and modify included files:



Change c:\python27\include to c:\python26\include  
 Modify included libs:



Change c:\python27\libs to c:\python26\libs  
 Compile, and copy the starpy.pyd to X:\srplab\libs directory.

## 2.2.4 Install php

Current support 2.3.X version, PHP install package may be downloaded from website.

### **win32:**

php\_starcore is located in directory X:\srplab\libs,

edit php.ini, add starcore extension as follow:

```
[PHP_STARCORE]
```

```
extension=c:\srplab\libs\php_starcore.dll
```

```
starcore.java.class.path = ".;C:\Program Files\Java\jdk1.6.0_21\lib;C:\srplab\libs\starcore.jar"
```

```
starcore.java.sharelib = "C:\Program Files\Java\jdk1.6.0_21\jre\bin\client\jvm.dll"
```

Note: after install php, php.ini maybe not exist. In this case, you can change the file php.ini-development to php.ini.

if you want php to call service developed with java, you should set variables in php.ini as above.

source code of PHP extension is published with cle package, which locates in directory X:\srplab\starcore\php.

If you want to compile yourself, should use VC2008 and PHP source code, Steps lists as follows:

1. running buildconf.bat, generate .js script.
2. running configure.bat, generate header file config.w32.h in directory main.

In this period, some files may be missed for compiling PHP which should be downloaded from php website.

3. modify config.w32.h

```
/* Detected compiler version */
```

```
#define COMPILER "MSVC9 (Visual C++ 8.0)"
```

```
/* Compiler compatibility ID */
```

```
#define PHP_COMPILER_ID "VC9"
```

Change VC6 to VC9.

4. use VC2008 to open the project under directory ext\starcore, and compile.

### 5. compiling starcore needs jdk header files.

### **linux:**

Install starcore extension for linux, should compile the source code. Please first download php source code, and install php.

copy source code of starcore in directory /usr/local/srplab/starcore/php/ext/starcore to the directory of source code of php ext/starcore

runing the following commands:

```
/usr/local/php/bin/phpize  
./configure --enable-starcore --with-php-config=/usr/local/php/bin/php-config  
make  
make install
```

compiling starcore needs jdk header files, as follows:

```
PHP_ADD_INCLUDE(/usr/include/starcore)  
PHP_ADD_INCLUDE(/usr/java/jdk1.6.0_24/include)  
PHP_ADD_INCLUDE(/usr/java/jdk1.6.0_24/include/linux)  
STARCORELIB_CFLAGS="-DENV_LINUX"
```

modify php.ini, if not exist, copy one as follow:

```
cp php.ini-development /usr/local/php/etc/php.ini  
vi /usr/local/php/etc/php.ini  
add extension:
```

**extension=starcore.so**

**starcore.java.class.path = "/usr/java/jdk1.6.0\_24/lib:/usr/java/jdk1.6.0\_24/jre/lib"**

**starcore.java.sharelib = "/usr/java/jdk1.6.0\_24/jre/lib/i386/server/libjvm.so"**

if you want php to call service developed with java, you should set variables in php.ini as above.

## 2. 2. 5 Install java

Supports 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 eclipse,the java library X:\srplab\libs\starcore.jar should be imported first.

### **linux:**

CLASSPATH,add /etc/srplab/libs/starcore.jar.

## 2. 2. 6 Install .NET(skip)

Version higher than .NET3.5.

cle .net interface library is star\_csharp.dll, which will be 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, which may be used in C# programming environment.

## 2.2.7 Debug and compile

### 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, exmaple is as follow:

```
#####
#
# 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_CXXOBS := $(C_CALL_CXXSRCS:%.cpp=%.o)

#####
CXXOBS := ${C_CALL_CXXOBS}
COBJS :=

EXEC_C_CALL_OBJS := ${C_CALL_CXXOBS}

#####
# Targets of the build
#####
```

```

OBS_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 sharelib, 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_CXXOBS := $(ADDFUNCTION_CXXSRCS:%.cpp=%.o)

```

```

#####
CXXOBS := ${ADDFUNCTION_CXXOBS}
COBS :=

EXEC_ADDFUNCTION_OBS := ${ADDFUNCTION_CXXOBS}

#####
# Targets of the build
#####
OBS_PATH = .

EXEC_ADDFUNCTION := ../AddFunction.so

all: ${EXEC_ADDFUNCTION}

#####
# Output
#####

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

#####
# common rules
#####

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

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

dist:
    bash makedistlinux

clean:
    -rm -f core ${CXXOBS} ${COBS} ${EXEC_ADDFUNCTION}

depend:
    #makedepend ${INCS} ${SRCS}

```

## 2. 2. 7. 2 Debug

On win32 ,SRPWatch may be started. Outputs from starcore will be displayed in the watch window.

Outputs may also be configed 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 fromat.

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. coding type is utf-8.

## 2. 2. 8 Run

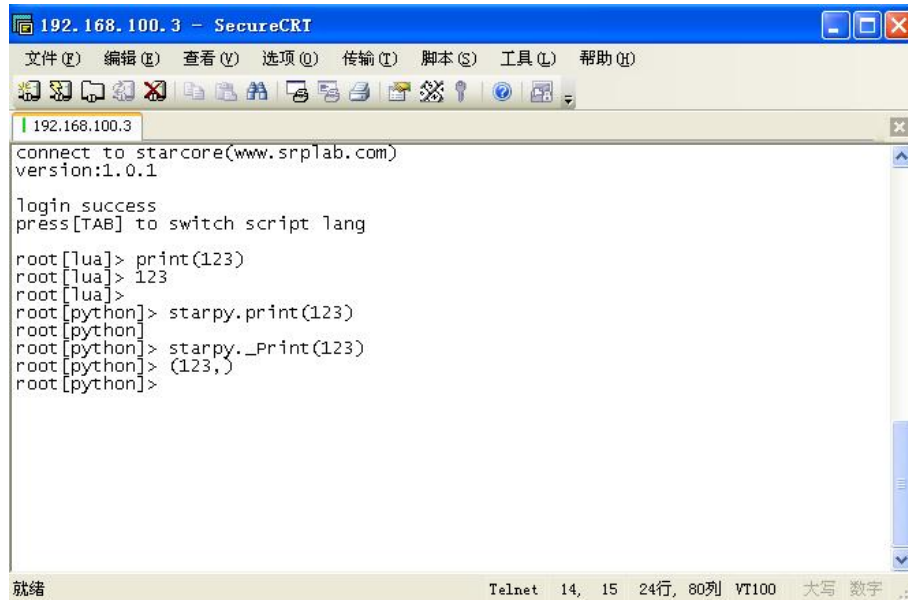
1. Using starapp.exe to load CLE applications, which may be share library, script file(excepy php),etc.For example:  
starapp -e "XXX.class?script=java"
2. For python,may use command like : python filename.
3. For php,may use command like : php -f filename, or put the file on website, and uses web browser to get the result.
4. Forjava,may use command like : java class name.
5. for c#,may use command, or use starapp -e "XXX.exe/dll?script=csharp"

## 2. 3 *starcore tools*

2. 3. 1 starapp:cle application running environment,which can load sharelib, lua/java/python/csharp shcipts.

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

-t Telnet port. If the parameter is set, then 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.



```
192.168.100.3 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
192.168.100.3
connect to starcore(www.srplab.com)
version:1.0.1

login success
press[TAB] to switch script lang

root[lua]> print(123)
root[lua]> 123
root[lua]>
root[python]> starpy.print(123)
root[python]>
root[python]> starpy._Print(123)
root[python]> (123,)
root[python]>
```

-w Web port. If 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 allow to load sharelib(dll/so),=1 not permit

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 nt 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

### 2. 3. 2 star2c/star2h,generate header file and code skeleton.

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

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

`star2c {service url} { password of root user } {xml configfile }`

Service url:maybe local path 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 format:

```
<?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

### 2.3.3 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), client side(-c), or debug server side (-d).

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

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

### 2.3.4 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

### 2.3.5 starsrvparse/starsrvunparse

Parse or unparse starcore service,

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

Input is service description file in xml format, 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.

### 2.3.6 starsrvpack

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

Output file name is attached .bin postfix, which is used to solve download problem 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/service> name.

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 linux/win32 .
    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 change 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>
```

### 2.3.7 starsrvcheck

starsrvcheck [serviceurl] [-u]

If serviceurl is omitted, then check update from starcore website. Otherwise check update from special website. Command line is:

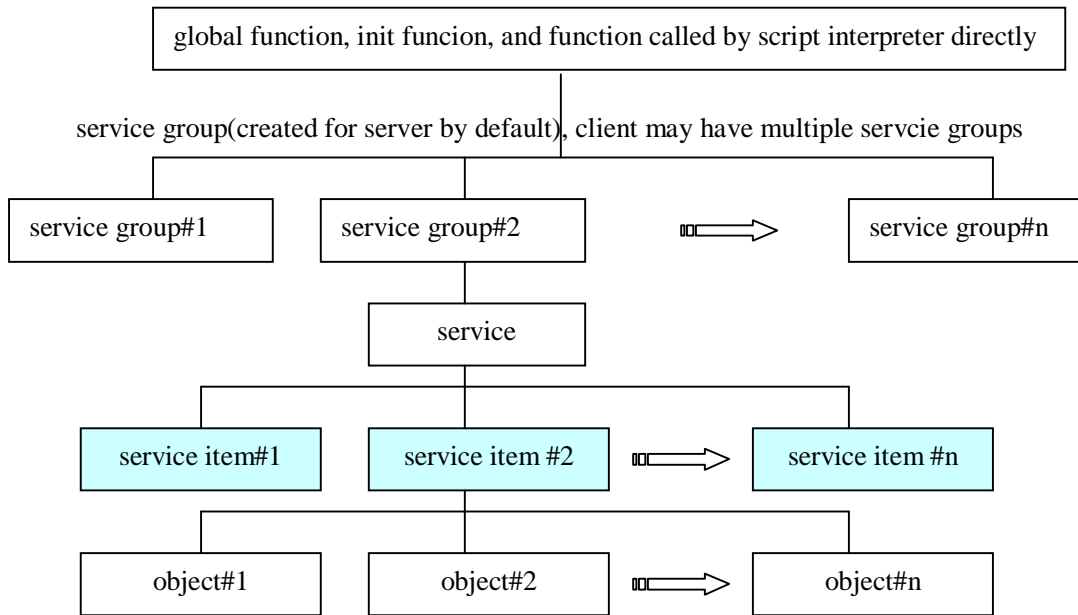
starssrvcheck or

starsrvcheck <http://www.XXXX.XX/XX>

## 3 Init CLE

### 3.1 Architecture of CLE

In architecture of CLE, application includes four levels of objects, 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, mainly is create and init the above four kinds of object. Specific functions is provided by object.

### 3.2 Init Cle using C/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.lib[win32], 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 sharelib 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 sharelib

```
#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.

## 3.3 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, "" )
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, "" )
Service = SrvGroup:_GetService("root","123")
...
SrvGroup:_ClearService()
libstarcore._ModuleExit()
```

## 3. 4 *Init Cle using python*

### 3. 4. 1 init type 1, create service group directly

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

### 3. 4. 2 init type 2, create service directly

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

### 3. 4. 3 init type 3, create service step by step

```
import starpy
starpy._InitCore(True,True,False,True,"",0,"",0)
SrvGroup = starpy._GetSrvGroup()
SrvGroup._CreateService( "", " test", "123",5,0,0,0,0,"" )
Service = SrvGroup._GetService("root","123")
...
SrvGroup._ClearService()
starpy._ModuleExit()
```

## 3.5 *Init Cle using php*

### 3. 5. 1 init type 1, create service group directly

```
<?php
$starcore = new StarCoreFactory();
$SrvGroup= $starcore->_InitSimpleEx(0,0);
$SrvGroup->_CreateService( "", " test", "123",5,0,0,0,0,"" );
$Service = $SrvGroup->_GetService("root","123");
...
$SrvGroup->_ClearService();
$starcore->_ModuleExit();
?>
```

### 3. 5. 2 init type 2, create service directly

```
<?php
$starcore = new StarCoreFactory();
$Service= $starcore->_InitSimple("test", "123",0,0);
```

```
..
$Service->_ServiceGroup->_ClearService();
$starcore->_ModuleExit();
?>
```

### 3.5.3 init type 3, create service step by step

```
<?php
$starcore = new StarCoreFactory();
$starcore->_InitCore(true,true,false,true,"",0,"",0);
$SrvGroup = $starcore->_GetSrvGroup();
$SrvGroup->_CreateService( "", " test", "123",5,0,0,0,0,"" );
$Service = $SrvGroup->_GetService("root","123");
...
$SrvGroup->_ClearService();
$starcore->_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

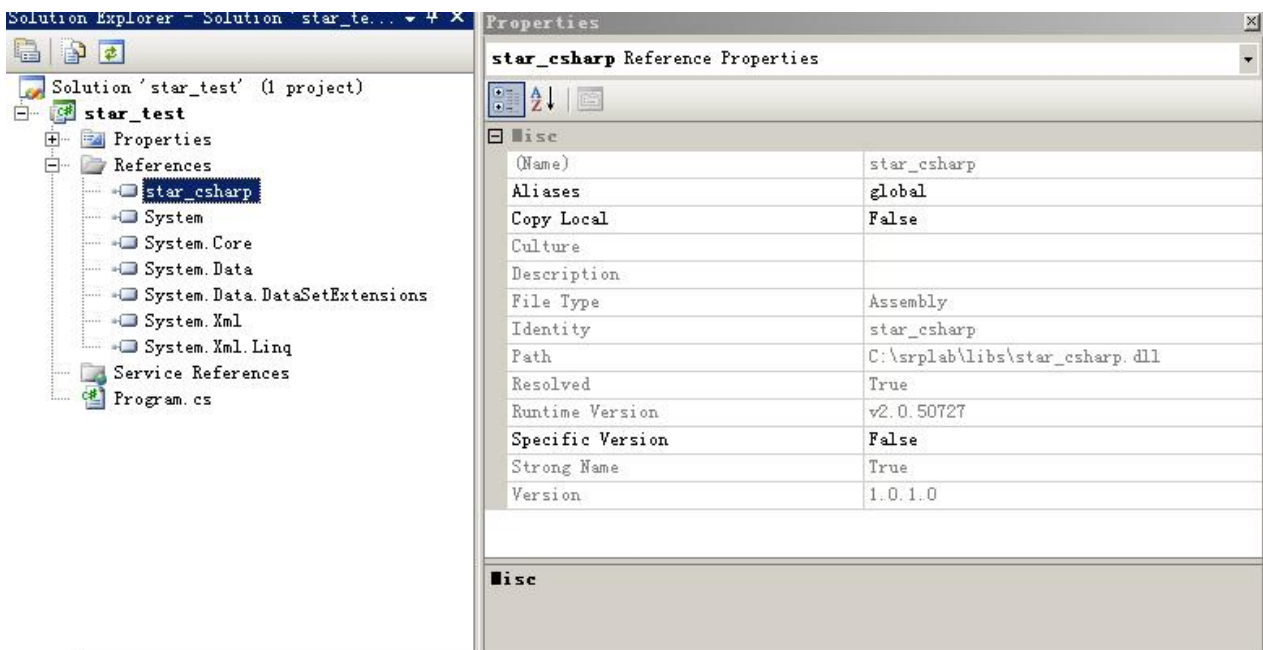
```

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,"" );
        StarServiceClass Service = SrvGroup._GetService("root","123");
        ..
        SrvGroup._ClearService();
        starcore._ModuleExit();
    }
}

```

## 3.7 Init Cle using csharp

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



star\_csharp library is stored in directory c:\srplab\libs.

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 arguments 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,"" );
    StarServiceClass Service = SrvGroup._GetService("root","123");

    ..

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

### 3.8 Create object, define its attributes and functions

Applications based on CLE, mainly is create and init the above four kinds of object. Specific functions is provided by object.

#### 3.8.1 python

```
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
```

#### 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 php

```
$Object = $Service->_New("Test") // Create object named Test
$Object->Attr = 123;           // Define attribute Attr
function Object_Func($self,$Para)
```

```
{
    echo $self, $Para;
}
$Object->Func = "Object_Func"; // Define function Func
```

### 3. 8. 4 java

```
class MyObjectClass extends StarObjectClass{
    public void Func ( StarObjectClass self,int para )    // Define function Func
    {
        System.out.println(self);
        System.out.println(para);
    }
    public MyObjectClass(StarObjectClass srcobj){
        super(srcobj);
        _Set("Attr",123);                                // Define attribute Attr
    }
}
public class XXXXX{
    public static void main(String[] args){
        .....
        StarObjectClass Object = new MyObjectClass(Service._New ("Test")); // Create object named Test
        .....
    }
}
```

### 3. 8. 5 c#

```
class MyObjectClass : StarObjectClass{
    public void Func ( StarObjectClass self,int para )    // Define function Func
    {
        Console.WriteLine (self);
        Console.WriteLine (para);
    }
    public MyObjectClass(StarObjectClass srcobj) :base(srcobj){
        _Set("Attr",123);                                // Define attribute Attr
    }
}
namespace XXXXX
{
    class Program
    {
        public static void main(String[] args){
```

```
.....
StarObjectClass Object = new MyObjectClass(Service._New ("Test")); // Create object named Test
.....
}
}
}
```

### 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

    ....
}
```

### 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 php

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

### 3. 8. 7. 4 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. 5 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. 6 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 `ClassOfSRPCControlInterface` 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 presented, which will continue to dispatch message until callback function returns true. Script also presents function `_SRPDispatch`, which may be used to dispatch message same as `SRPDispatch`.

#### 3.9.1 c#/java

For Form application, a timer(10ms) should create 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 mesasge = new Message();
                mesasge.what = 0;
                handler.sendMessage(mesasge);
            }
        }, 0, 10);
    }
}
```

### 3.10 Programming problems needed pay 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` sharelib is loaded or not. If the sharelib 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 limitations 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, variable 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 function:

```
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->ToVString(&ParaObj->Para5, "From caller");
```

## 4 Develop common extension.

examples in directory `examples\cle.basic\call.other`

## 4.1 Common extension

### 4.1.1 Develop common extension using python

```
#import python module
import starpy
#Init cle, and create service group and service
Service = starpy._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 finished. The first step is init cle, then create service group, create service, create service item, create object. Objects provide function Add.

### 4.1.2 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
```

### 4.1.3 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"));
}
}
```

#### 4. 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);
    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;
}
```

#### 4. 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"));
    }
}
}

```

## 4.2 Call common extension using C/C++

Call common extension, application may used interface functions presented 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

```

### 4.3 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()

```

### 4.4 Call common extension using python

```

import starpy
#Service=starpy._InitSimple("test","123",0,0,"files/AddFunction.lua?script=lua");
#Service=starpy._InitSimple("test","123",0,0,"files/AddFunction.py?script=python");
#Service=starpy._InitSimple("test","123",0,0,"files/AddFunction.class?script=java");
Service=starpy._InitSimple("test","123",0,0,"files/AddFunction.dll");
a = Service.TestClass:_New();
print(a:Add(12,34))
Service._ServiceGroup:_ClearService()
starpy._ModuleExit()

```

### 4.5 Call common extension using PHP

```

<?php
$starcore = new StarCoreFactory();

```

```

//$Service=$starcore->_InitSimple("test","123",0,0,"files/AddFunction.lua?script=lua");
//$Service=$starcore->_InitSimple("test","123",0,0,"files/AddFunction.py?script=python");
//$Service=$starcore->_InitSimple("test","123",0,0,"files/AddFunction.class?script=java");
$Service=$starcore->_InitSimple("test","123",0,0,"files/AddFunction.dll");
$a = $Service->TestClass->_New();
echo $a->Add(12,34);
$Service->_ServiceGroup->_ClearService();
$starcore->_ModuleExit();
?>

```

## 4.6 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();
    }
}

```

## 4.7 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();
}
}
}
```

## 4.8 *passing complex data structures between languages*

Examples in directory `examples\cle.advanced\call.other`.

Application can pass data structures with object as function parameter. Object may contain struct attributes. Attribute types supported by object and struct is listed below.

Type 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 :
TYPE_MEMORY(Reserved) :
```

Type 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 structure 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);
```

Corresponding to the c / c + + structure is as follows:

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

#### 4.8.1 Extension module to be called

##### 4.8.1.1 Develop common extension using python

```
import starpy  
Service = starpy._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;
```

##### 4.8.1.2 Develop 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
```

#### 4.8.1.3 Develop 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"));
    }
}
```

#### 4.8.1.4 Develop 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);
    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;  
}
```

#### 4.8.1.5 Develop 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"));  
        }  
    }  
}
```

#### 4. 8. 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
iParam, 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;
}

```

#### 4. 8. 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()

```

#### 4. 8. 4 Call common extension using python

```

import starpy
#Service=starpy._InitSimple("test","123",0,0,"files/Test.lua?script=lua");
#Service=starpy._InitSimple("test","123",0,0,"files/Test.py?script=python");
#Service=starpy._InitSimple("test","123",0,0,"files/Test.class?script=java");

```

```
Service=starpy._InitSimple("test","123",0,0,"files/Test.dll");
#Service=starpy._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()
starpy._ModuleExit()
```

#### 4. 8. 5 Call common extension using PHP

```
<?php
$starcore = new StarCoreFactory();
$Service=$starcore->_InitSimple("test","123",0,0,"files/Test.lua?script=lua");
// $Service=$starcore->_InitSimple("test","123",0,0,"files/Test.py?script=python");
//-----$Service=$starcore->_InitSimple("test","123",0,0,"files/Test.class?script=java"); // not supported
// $Service=$starcore->_InitSimple("test","123",0,0,"files/Test.dll");

if( $Service == null ){
    $starcore->_ModuleExit();
    return;
}

function StarCore_CallBack( $ServiceGroupID, $uMes, $wParam, $lParam )
{
    global $starcore;
    if( $uMes == $starcore->MSG_DISPMMSG || $uMes == $starcore->MSG_DISPLUAMSG ){
        echo $wParam;
    }
    return false;
}

$starcore->_RegMsgCallBack("StarCore_CallBack");
$a = $Service->TestClass->_New();
$ParaObj = $Service->ParaClass->_New();
$ParaObj->Para1 = 124;
$ParaObj->Para2 = $a->_ID;
$ParaObj->Para3 = 23456.78;
$ParaObj->Para4 = array(999,4444.55);
$ParaObj->Para5 = "From caller";
```

```
$a->PrintObj($ParaObj);

$Service->_ServiceGroup->_ClearService();
$starcore->_ModuleExit();
?>
```

#### 4. 8. 6 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();
    }
}
```

#### 4. 8. 7 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();
    }
}
}

```

## 4.9 A more complicated example

### 4.9.1 java swing window(Callback function)

The example is in directory `examples\cle.basic\call.javawin`

#### 4.9.1.1 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 JavaWinClass is created, which contains a function to create window CreateWindow, and object's callback function OnClose will be called when the window is closed.

#### 4.9.1.2 Call using python

```
import starpy
Service=starpy._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

starpy._MsgLoop( ExitProc )

Service._ServiceGroup._ClearService()
starpy._ModuleExit()
```

#### 4.9.1.3 Call 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;
}
```

#### 4.9.1.4 Call 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();
    }
}
}

```

#### 4. 9. 2 call jsoup

The example is in directory `examples\cle.basic\call.jsoup.java`

##### 4. 9. 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();
    }
}
```

Pack the above java program and jsoup into one jar file.

#### 4.9.2.2 Call using python

```
import starpy
Service=starpy._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()
starpy._ModuleExit()

```

#### 4.9.2.3 Call 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;
}

```

#### 4.9.2.4 Call 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();
        }
    }
}
```

#### 4. 9. 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();  
        }  
    }  
}
```

#### 4.9.4 php call c#

Examples in directory `examples\cle.basic\others\phpcall\call.csharp`

C# code:

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

namespace Test
{
    class MyObjectClass : StarObjectClass
    {
        public String GetStr(StarObjectClass self, String Str)
        {
            return "Return from csharp " + Str;
        }
        public MyObjectClass(StarObjectClass srcobj)
            : base(srcobj)
        {
        }
    }
// public class Class1
// {

    class Program
    {
        static void Main(string[] args)
        {
            StarCoreFactory starcore = StarCoreFactory.GetFactory();
            StarServiceClass Service = starcore._InitSimple("test", "123", 0, 0);
            MyObjectClass Obj = new MyObjectClass(Service._New("TestClass"));
        }
    }
}
```

php code:

```
<?php
$starcore = new StarCoreFactory();
$Service=$starcore->_InitSimple("testcall", "123",0,0,"Test.dll?script=csharp");
$a = $Service->TestClass->_New();
echo "result= ",$a->GetStr("Hello");
$starcore->_ModuleExit();
?>
```



Command line:

```
php -f callcsharp.php
```

You can copy files callsharp.php and Test.dll to your website directory, and test using web browser.

<http://127.0.0.1/callcsharp.php>

## 4. 10 Direct call sharelib

examples in directory `examples\cle.basic\call.sharelib`

Using CLE, you can directly call share library which has simple interface. For example, call MessageBox on win32:

### 4. 10. 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()
```

### 4. 10. 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();
}
}
```

#### 4. 10. 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();
}
}
}

```

## 4. 11 Mixed script language programming

examples in directory examples\cle.basic\embed.call.other

### 4. 11. 1 Module to be called

#### 4. 11. 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;

```

#### 4. 11. 1. 2python

```

SrvGroup = starpy._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;

```

#### 4. 11. 1. 3java

```

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);
}
}
```

#### 4. 11. 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);
    }
}
}
```

#### 4. 11. 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;
}
```

#### 4. 11. 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()
```

#### 4. 11. 4 python call other script

```
import starpy
Service=starpy._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()
starpy._ModuleExit()
```

#### 4. 11. 5 php call other script

```
<?php
    $starcore = new StarCoreFactory();
    $Service=$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","");
    $a = $Service->TestClass->_New();
    $a->Value = 100;
    echo $a->ChildValue;
    echo $a->Add(12,34);
    $Service->_ServiceGroup->_ClearService();
    $starcore->_ModuleExit();
?>
```

#### 4. 11. 6 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();
    }
}
```

#### 4. 11. 7 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();
        }
    }
}

```

#### 4.12 ASP.NET call CLE extensions

Because cle will be run in different AppDomains, for each call, function should perform a complete procedure which includes cle init, create service, and cle term as follow.

A simple code is presented 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();
    }
}

```

```
}
```

## 5 CLE distributed function

Examples in directory examples\comm.basic, which includes code of C++,lua,python,php,java,c#,etc.

### 5.1 TCP/UDP communication

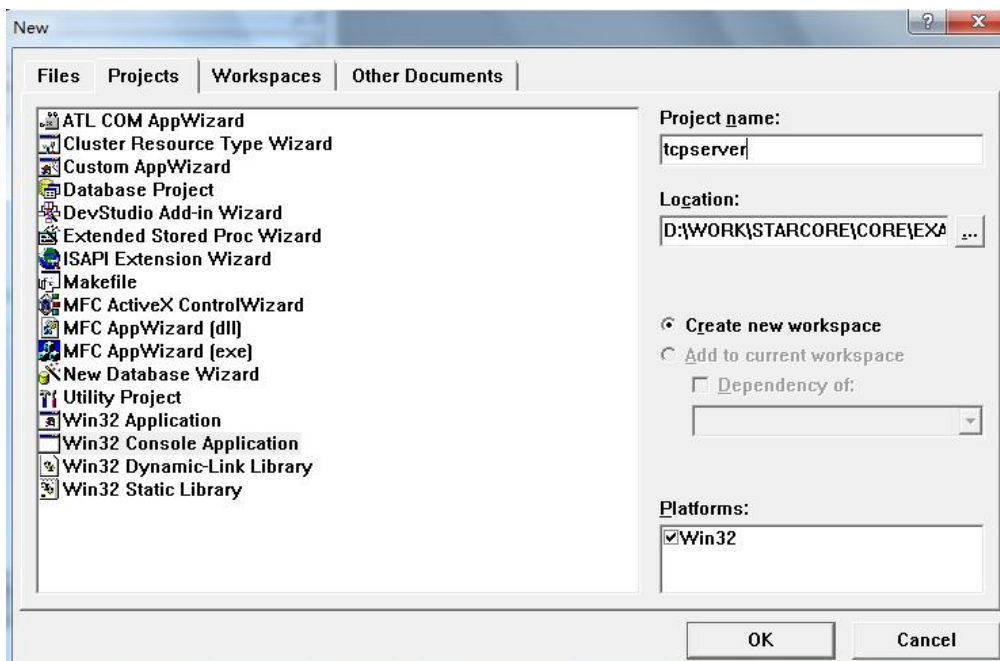
Using function presented by interface ClassOfSRPCCommInterface, applications can communicate with each other based on TCP/UDP.

#### 5.1.1 TCP server

##### 5.1.1.1 C

##### 5.1.1.1.1 Win32

##### 5.1.1.1.1.1 Create project(VC6)



Set header file path,

set to Multithread,

add starlib\_vcm.lib into the project

##### 5.1.1.1.1.2 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("create tcp server on port[%d] fail\n",atoi(argv[1]));
    CommInterface ->Release();
    VSCore_TermSimple(&Context);
    return -1;
}

printf("create 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 from %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;
}

```

#### 5.1.1.1.1.3 compile and run

tcpserver 3005

#### 5.1.1.1.2 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}

```

### 5. 1. 1. 2 lua

code is as follow:

```

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, "F0611A16-BFAA-4d0b-803F-807EC63BD265" )
get communicate interface

CommInterface = SrvGroup:_NewCommInterface()
create TCP server, using port 3005
CommInterface.ConnectionID = CommInterface:_TCPSetupServer(100, nil, 3005)
if CommInterface.ConnectionID == 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

**5.1.1.3 python**

code is as follow:

```
import sys
import starpy

starpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = starpy._GetSrvGroup()
SrvGroup._CreateService( "", "test", "123",5,0,0,0,0,"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 starpy._KeyPress() == 27 :
        return True
    return False

starpy._MsgLoop( ExitProc )

print("Exit...")
```

```
SrvGroup._ClearService()
starpy._ModuleExit()
```

Starapp -e tcpserver.py?script=python

#### 5. 1. 1. 4 php

code is as follow:

```
<?php
$starcore = new StarCoreFactory();
$starcore->_InitCore(true,true,false,true,"",0,"",0);

$SrvGroup = $starcore->_GetSrvGroup();
$SrvGroup->_CreateService( "", "test", "123",5,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" );

$CommInterface = $SrvGroup->_NewCommInterface();

$CommInterface->ConnetionID = $CommInterface->_TCPSetupServer(100,"",3005);
if($CommInterface->ConnetionID == 0 )
{
    echo "setup server on port 3005 fail";
    $starcore->_ModuleExit();
    return;
}

echo "setup server on port 3005 success";
$BinBuf = $SrvGroup->_NewBinBuf();
function CommInterface_MsgProc($self,$uMes,$Msg)
{
    global $BinBuf;
    $Size = 0;

    if($uMes == $self->TCP_ONCONNECT)
        echo "tcp connect from ",$self->_GetIP($Msg[3]);
    elseif($uMes == $self->TCP_ONREAD){
        $Size=$self->_TCPRecv($Msg[0],$BinBuf,0);
        echo "tcp read from ",$Msg[0],$Size,"\n";
        while($Size != 0 ){
            echo "receive from",$Msg[0],":",$BinBuf->_Get(0,0,'s');
            $Size=$self->_TCPRecv($Msg[0],$BinBuf,0);
        }
    }elseif($uMes == $self->TCP_ONCLOSE){
        echo "tcp connect close ",$Msg[0];
    }
}
$CommInterface->_MsgProc = "CommInterface_MsgProc";

function ExitProc()
{
    global $starcore;
    if($starcore -> _KeyPress() == 27){
        return true;
    }
    return false;
}

$starcore->_MsgLoop( "ExitProc" );

echo "Exit...";

$SrvGroup->_ClearService();
$starcore->_ModuleExit();
?>
```

## 5.1.1.5 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])+":"+_Toint(Msg[1])+_Toint(Msg[2]));
                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();
    }
}
```

### 5.1.1.6 c#

```

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])+":"+_Toint(Msg[1])+_Toint(Msg[2]));
                    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 ConnexionID = CommInterface._TCPSetupServer(100,"",3005);
            if(ConnexionID == 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();
    }
}
}

```

## 5.1.2 TCP client

### 5.1.2.1 C

#### 5.1.2.1.1 Win32

##### 5.1.2.1.1.1 create project(VC6)

refer to above.

##### 5.1.2.1.1.2 Create and edit source file

Create source file tcpclient,add to project,code is 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,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("create 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;
}

```

#### 5.1.2.1.1.3 Compile and run

tcpclient 127.0.0.1 3005

#### 5.1.2.1.2 linux

write Makefile,skip

#### 5.1.2.2 lua

Code is as follow:

```

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

### 5.1.2.3 python

Code is as follow:

```

import sys
import starpy

```

```

starpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = starpy._GetSrvGroup()
SrvGroup._CreateService( "", "test", "123",5,0,0,0,0,"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()
    starpy._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 starpy._KeyPress() == 27 :
        return True
    return False

starpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()

```

Starapp -e tcpclient.py?script=python

Python tcpclient.py

### 5. 1. 3 UDP server

#### 5. 1. 3. 1 C

##### 5.1.3.1.1 Win32

##### 5.1.3.1.1.1 Create project(VC6)

See above

##### 5.1.3.1.1.2 Create and edit source file

Code is 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 UdpConnectionID;

    BasicSRPInterface = VS_Core_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();
        VS_Core_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;
}

```

#### 5.1.3.1.1.3 Compile and run

udpserver 3005

#### 5.1.3.1.2 linux

Write makefile(skip)

#### 5.1.3.2 lua

Code is as follow

```

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:_UDPSSetupServer(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
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

### 5.1.3.3 python

Code is as follow

```
import sys
import starpy

starpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = starpy._GetSrvGroup()
SrvGroup._CreateService( "", "test", "123",5,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )

CommInterface = SrvGroup._NewCommInterface()

CommInterface.ConnetionID = CommInterface._UDPSSetupServer(100,"",3005)
if CommInterface.ConnetionID == 0 :
    print("setup udp server on port 3005 fail")
    SrvGroup._ClearService()
    starpy._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 starpy._KeyPress() == 27 :
        return True
    return False

starpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()
```

Starapp -e udpserver.py?script=python

Python udpserver.py

#### 5.1.4 UDP client

##### 5.1.4.1 C

##### 5.1.4.1.1 Win32

##### 5.1.4.1.1.1 Create project(VC6)

See above

##### 5.1.4.1.1.2 Create and edit source file

Code is 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 UdpConnectionID,UdpSendTimerID,Index;

    BasicSRPInterface = VS_Core_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();
        VS_Core_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;
}

```

#### 5.1.4.1.1.3 Compile and run

udpclient 127.0.0.1 3005

#### 5.1.4.1.2 linux

Write makefile(skip)

#### 5. 1. 4. 2 lua

Code is as follow

```

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:_UDPSSetupClient(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:_UDPSSetSockAddr("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.ConnexionID,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
end
Message loop
function ExitProc()
    if libstarcore._KeyPress() == 27 then
        return true
    end
    return false
end
end

libstarcore._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup:_ClearService()
libstarcore._ModuleExit()

```

Starapp -e udpclient.lua

### 5. 1. 4. 3 python

Code is as follow

```

import sys
import starpy

starpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = starpy._GetSrvGroup()
SrvGroup._CreateService( "", "test", "123",5,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" )

CommInterface = SrvGroup._NewCommInterface()

CommInterface.ConnexionID = CommInterface._UDPSSetupClient(100)
if CommInterface.ConnexionID == 0 :
    print("setup udp client fail")
    SrvGroup._ClearService()
    starpy._ModuleExit()
    sys.exit()

print("setup udp client success")
CommInterface._SetupTimer(100,0)
CommInterface.Index = 0
print(CommInterface.ConnexionID)

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.ConnectionID,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 starpy._KeyPress() == 27 :
        return True
    return False

starpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()

```

Starapp -e udpclient.py?script=python

## 5.2 Remotecal I

### 5.2.1 Create server side application

#### 5.2.1.1 C

Examples in [directore examples\comm.basic\remotecall.c](#)

##### 5.2.1.1.1 Win32

##### 5.2.1.1.1.1 Create project(VC6)

skip

##### 5.2.1.1.1.2 Create and edit source file

Create source file test\_server,add to project,Code is as follow

```

#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_VSDISPMMSG :
        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, "%sadsaf", 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, "adsaf");
    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, &ErrorInfo, 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;
}

```

### 5.2.1.1.1.3 Compile and run

test\_server

### 5.2.1.1.2 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_CXXOBS := $(TEST_SERVER_CXXSRCS:%.cpp=%.o)

#*****
CXXOBS := ${TEST_SERVER_CXXOBS}
COBS :=

EXEC_TEST_SERVER_OBS := ${TEST_SERVER_CXXOBS}

#*****
# Targets of the build
#*****
OBS_PATH = output/linux

EXEC_TEST_SERVER := ${OBS_PATH}/test_server

all: ${EXEC_TEST_SERVER}

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

${EXEC_TEST_SERVER}: ${EXEC_TEST_SERVER_OBS}
    ${LD} -o $@ ${LDFLAGS} ${EXEC_TEST_SERVER_OBS} ${LIBS} ${EXTRA_LIBS}

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

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

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

dist:
    bash makedistlinux

clean:
    -rm -f core ${CXXOBS} ${COBS} ${EXEC_TEST_SERVER}

depend:
    #makedepend ${INCS} ${SRCS}

```

### 5. 2. 1. 2 lua

Examples in [directoryexamples\comm.basic\remotecall.lua](#)

Code is as follow

```

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()

```

### 5.2.1.3 python

Examples in directoryexamples\comm.basic\remotecall.python

Code is as follow

```

import sys
import starpy
initstarcore(cle)
starpy._InitCore(True,True,False,True,"",0,"",3008)
get service group 0, and create service

```

```

SrvGroup = starpy._GetSrvGroup()
#--create service
SrvGroup._CreateService( "", "RemoteCallServer", "123",5,0,0,0,0,"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 starpy._KeyPress() == 27 :
        return True
    return False

starpy._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()

```

Run:

```
starapp -e test_server.py?script=python
```

```
python test_server.py
```

## 5. 2. 2 Create client side application

### 5. 2. 2. 1 Wi n32

#### 5.2.2.1.1 Create project(VC6)

See above

#### 5.2.2.1.2 Create and edit source file

Create source file test\_client,add to project,Code is as follow



```

#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("usage 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;
}

```

### 5.2.2.1.3 Compile and run

test\_client

### 5.2.2.2 linux

Write Makefile,see above

### 5.2.2.3 lua

Code is as follow

```
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()
```

### 5.2.2.4 python

Code is as follow

```
import sys
import starpy
initstarcore(cle)
starpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = starpy._CreateSrvGroup(1,starpy.VS_CLIENT_USER);

ret = SrvGroup._SConnect("", "127.0.0.1",3008, "", "")
if ret == 0 :
    print( "Fail To Connect..." )
```

```

    SrvGroup._ClearService()
    starpy._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()
starpy._ModuleExit()

```

starapp -e test\_client.py?script=python

### 5.2.3 Create and use stand alone starcore service

#### 5.2.3.1 Create starcore service

Examples in directoryexamples\comm.basic\service

##### 5.2.3.1.1 Create starcore service data file

###### 5.2.3.1.1.1 C

###### 5.2.3.1.1.1.1 Win32

###### 5.2.3.1.1.1.1.1 Create project(VC6)

skip

###### 5.2.3.1.1.1.1.2 create source file

Create new file create\_service.cpp,Code is as follow

```

#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_VSDISPMMSG :
            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 = VS_Core_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);
    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");

    VS_Core_TermSimple(&Context);
    return 0;
}

```

### 5.2.3.1.1.1.3 Compile and run

Run create\_RemoetCallServe

### 5.2.3.1.1.1.2 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

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_REMOTE_CALLSERVER_CXXSRCS := create_RemoteCallServer.cpp

#####
CREATE_REMOTE_CALLSERVER_CXXOBS := ${CREATE_REMOTE_CALLSERVER_CXXSRCS:%.cpp=%.o}

#####
CXXOBS := ${CREATE_REMOTE_CALLSERVER_CXXOBS}
COBS :=

EXEC_CREATE_REMOTE_CALLSERVER_OBS := ${CREATE_REMOTE_CALLSERVER_CXXOBS}

#####
# 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

#### 5.2.3.1.1.2 Create service using lua

Code is as follow

```

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,"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()
```

### 5.2.3.1.1.3 Create service using python

Code is as follow

```
import sys
import starpy

starpy._InitCore(True,True,False,True,"",0,"",3008)

SrvGroup = starpy._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()
starpy._ModuleExit()
```

### 5.2.3.1.2 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.

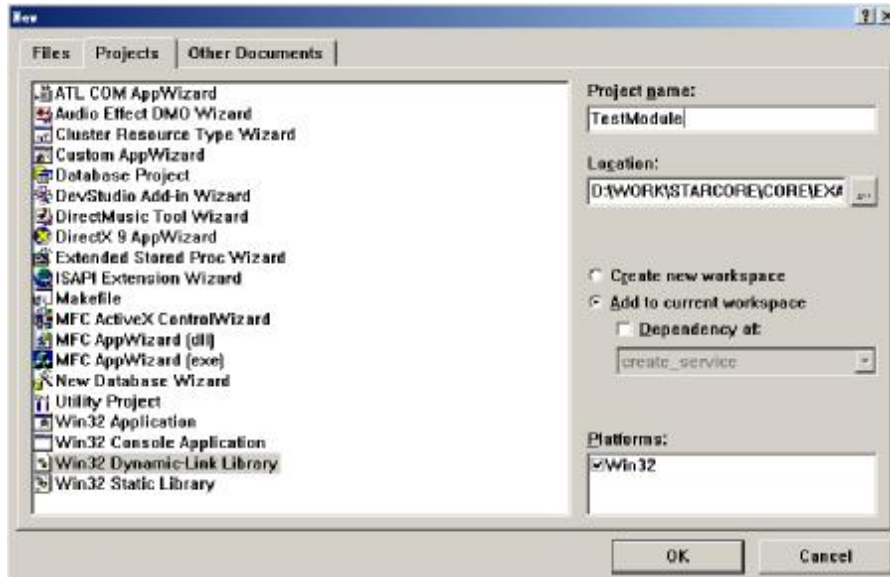
#### 5.2.3.1.3 Create module

Module is share library.

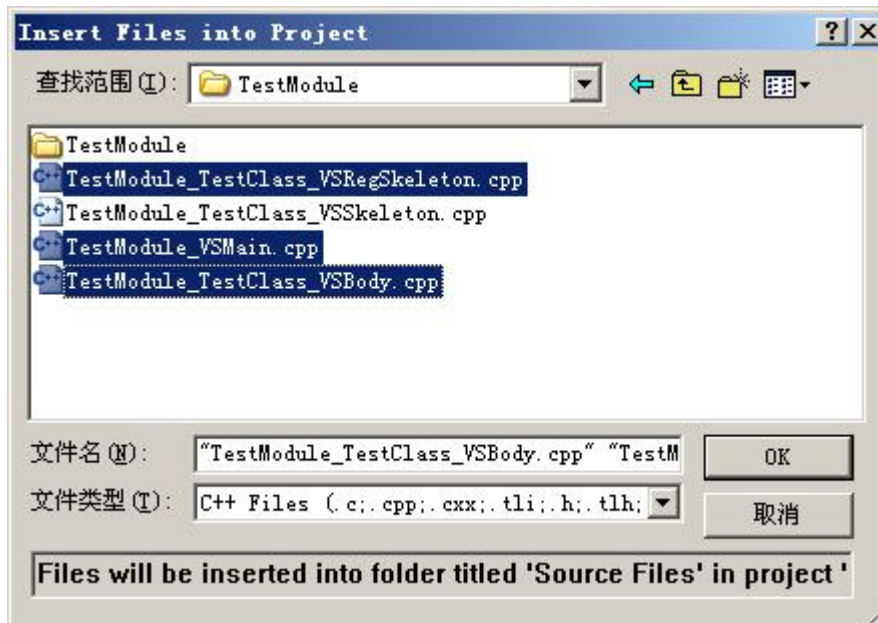
##### 5.2.3.1.3.1 Win32

##### 5.2.3.1.3.1.1 Create project(VC6)

Create new project:



Project name should 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



#### 5.2.3.1.3.1.2 Edit 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;
}

```

#### 5.2.3.1.3.1.3 Compile

After compile, TestModule.DLL will be generated.

#### 5.2.3.1.3.2 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

#### 5.2.3.2 Using starcore service

## Examples in directoryexamples\comm.basic\remotecall.service

### 5.2.3.2.1 Call by C++

#### 5.2.3.2.1.1 Win32

##### 5.2.3.2.1.1.1 Create Console project(VC6)

skip

##### 5.2.3.2.1.1.2 Create 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 dollowing 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,Code is as follow

```
#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;
}

```

#### 5.2.3.2.1.2 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_REMOTEALLSERVER_CXXSRCS := test_server_RemoteCallServer.cpp

#####
TEST_SERVER_REMOTEALLSERVER_CXXOBS :=
$(TEST_SERVER_REMOTEALLSERVER_CXXSRCS:%.cpp=%o)

#####
CXXOBS := ${TEST_SERVER_REMOTEALLSERVER_CXXOBS}
COBS :=

EXEC_TEST_SERVER_REMOTEALLSERVER_OBS := ${TEST_SERVER_REMOTEALLSERVER_CXXOBS}

#####
# Targets of the build
#####
EXEC_TEST_SERVER_REMOTEALLSERVER := test_server_RemoteCallServer_linux

all: ${EXEC_TEST_SERVER_REMOTEALLSERVER}

#####
# Output
#####

${EXEC_TEST_SERVER_REMOTEALLSERVER}: ${TEST_SERVER_REMOTEALLSERVER_CXXOBS}
    ${LD} -o $@ ${LDFLAGS} ${TEST_SERVER_REMOTEALLSERVER_CXXOBS} ${LIBS} ${EXTRA_LIBS}

#####
# common rules
#####

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

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

dist:
    bash makedistlinux

clean:
    -rm -f core ${CXXOBS} ${COBS} ${EXEC_TEST_SERVER_REMOTEALLSERVER}

depend:
    #makedepend ${INCS} ${SRCS}

```

### 5.2.3.2.2 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()

```

### 5.2.3.2.3 Call by python

```

import sys
import starpy
initstarcore(cle)
starpy._InitCore(True,True,False,True,"",0,"",3008)
get service group 0, and create service
SrvGroup = starpy._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 starpy._KeyPress() == 27 :
        return True
    return False

starpy._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()

```

### 5.3 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 :

Mapping 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

### 5.3.1 Create server side application

#### 5.3.1.1 C

Examples in [directoryexamples\comm.advanced\remotecall.c](#)

##### 5.3.1.1.1 Win32

##### 5.3.1.1.1.1 Create project(VC6)

skip,add libstarcore.lib to project.

##### 5.3.1.1.1.2 Create and edit source file

Create source file test\_server,add to project,Code is as follow

```
#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
IParam, 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);
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;
}

```

### 5.3.1.1.1.3 Compile and run

test\_server

### 5.3.1.2 lua

Examples in [directoryexamples\comm.advanced\remotecall.lua](#)

Code is as follow

```

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()
--创建服务
SrvGroup:_CreateService( "", "RemoteCallServer", "123",5,0,0,0,0,"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
end

```

```
libstarcore._MsgLoop( ExitProc )

print("Exit...")
exit, clear service and starcore
SrvGroup._ClearService()
libstarcore._ModuleExit()
```

### 5.3.1.3 python

Examples in directory `examples\comm.advanced\remotecall.python`

Code is as follow

```
import sys
import starpy
initstarcore(cle)

starpy._InitCore(True,True,False,True,"",0,"",3008)
get service group 0, and create service
SrvGroup = starpy._GetSrvGroup()
#--create service
SrvGroup._CreateService( "", "RemoteCallServer", "123",5,0,0,0,0,"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 starpy._KeyPress() == 27 :
        return True
    return False

starpy._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()
```

### 5.3.2 Create client side application

#### 5.3.2.1 Win32

Examples in `directoryexamples\comm.advanced\remotecall.c`

##### 5.3.2.1.1 Create project(VC6)

See above

##### 5.3.2.1.2 Create and edit source file

Create source file `test_client`, add to project, Code is as follow

```
#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;
}

```

### 5.3.2.1.3 Compile and run

test\_client

### 5.3.2.2 lua

Examples in [directoryexamples\comm.advanced\remotecall.lua](#)

Code is as follow

```

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()

```

### 5. 3. 2. 3 python

Examples in directory `examples\comm.advanced\remotecall.python`

Code is as follow

```

import sys
import starpy
initstarcore(cle)
starpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = starpy._CreateSrvGroup(1,starpy.VS_CLIENT_USER);

ret = SrvGroup._SConnect("", "127.0.0.1",3008,"","")
if ret == 0 :
    print( "Fail To Connect..." )
    SrvGroup._ClearService()
    starpy._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()
starpy._ModuleExit()
```

### 5.3.3 Create and ust stand alone starcore service

#### 5.3.3.1 Create starcore service

Examples in [directoryexamples\comm.advanced\service](#)

##### 5.3.3.1.1 create data file of service starcore

###### 5.3.3.1.1.1 C

###### 5.3.3.1.1.1.1 Win32

###### 5.3.3.1.1.1.1.1 Create project(VC6)

skip

###### 5.3.3.1.1.1.1.2 edit source code

create new file,create\_service.cpp,Code is as follow

```
#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
iParam, 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);
    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;,_UIDPTR("e65fa0b1-5684-429c-8075-4ca2ee685e6d"),&ErrorInfo);
    AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","ParaObject","VS_INT32  Para1;VS_UID
Para2;VS_FLOAT  Para3;struct  ParaStruct  Para4;VS_VSTRING  Para5;,_UIDPTR("f85b85b9-8109-4c02-b6f6-
4ad23f1cba38"),&ErrorInfo);
    //--Create Atomic Class, for define function, no attribute
    AtomicClass = SRPInterface ->CreateAtomicObjectSimple("TestItem","TestClass",NULL,_UIDPTR("07bdb329-
e9a4-41b4-b79d-10132210cd44"),&ErrorInfo);
    SRPInterface ->CreateAtomicFunctionSimple(AtomicClass,"GetRemoteObject","struct          ParaObject
*GetRemoteObject(struct          ParaObject          *Para);,_UIDPTR("2ef3218c-31e8-4d45-8002-
b8b29a91d837"),&ErrorInfo,VS_FALSE,VS_FALSE);

    SRPInterface ->          CreateAtomicModule("TestModule",VSMODULE_SERVER_SERVER          |
VSMODULE_SERVER_USER,_UIDPTR("a0164199-f3bd-4ad3-83ef-33d0b0939687"));

    SRPInterface -> SaveService("../..\\script");

    printf("save service to ../..\\script \n");

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

```

### 5.3.3.1.1.1.3 Compile and run

Run : create\_RemoteCallServe

### 5.3.3.2 Export 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.

### 5.3.3.3 create module

module is share library

#### 5.3.3.3.1 Win32

##### 5.3.3.3.1.1 Create project(VC6)

skip

##### 5.3.3.3.1.2 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;
}
```

##### 5.3.3.3.1.3 Compile

skip

### 5.3.4 called by LUA

need not change

### 5.3.5 called by Python

need not change

## 6 Webservice and http application

Examples in directory `examples\comm.basic`, include C++,lua,python,php,java,c# source code.

### 6.1 Http&HttpServer

Examples in directory `examples\comm.basic\ http_webserver`

#### 6.1.1 Http download

##### 6.1.1.1 C

##### 6.1.1.1.1 Win32

##### 6.1.1.1.1.1 Create project(VC6)

see above

##### 6.1.1.1.1.2 Create and edit source file

Code is as follow

```
#include "vsopenapi.h"

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

    if( VS_Core_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_strchr(argv[1],'/')+1,VS_TRUE,NULL,0);
    VS_Core_TermSimple(&Context);
    return 0;
}
```

##### 6.1.1.1.1.3 Compile and run

http\_download <http://127.0.0.1/index.html>

##### 6.1.1.1.2 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_CXXOBS := $(HTTP_DOWNLOAD_CXXSRCS:%.cpp=%.o)
HTTP_UPLOAD_CXXOBS := $(HTTP_UPLOAD_CXXSRCS:%.cpp=%.o)
SIMPLE_WEBSERVER_CXXOBS := $(SIMPLE_WEBSERVER_CXXSRCS:%.cpp=%.o)

#####
CXXOBS := ${HTTP_DOWNLOAD_CXXOBS} ${HTTP_UPLOAD_CXXOBS} ${SIMPLE_WEBSERVER_CXXOBS}

```

```

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}

```

### 6.1.1.2 lua

Code is as follow

```

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=_strchr(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\""
```

### 6.1.1.3 python

Code is as follow

```

import sys
if hasattr(sys, "argv") :
    if len(sys.argv) > 1 :
        Url = sys.argv[1]
import starpy

Service=starpy._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()
    starpy._ModuleExit()
    raise Exception("")

Pos=starpy._strchr(Url, '/')
if Pos == -1 :
    print("not find download filename")
    SrvGroup._ClearService()
    starpy._ModuleExit()
    raise Exception("")

FileName=Url[Pos+1:]

CommInterface._FileDownload(Url, FileName, True, None)

print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()

```

```
starapp -e "http_download.py?script=python;Url=\"http://127.0.0.1/zoc.rar\""
```

## 6.1.2 Http upload

### 6.1.2.1 C

#### 6.1.2.1.1 Win32

##### 6.1.2.1.1.1 Create project(VC6)

see above

##### 6.1.2.1.1.2 Create and edit source file

Code is as follow

```
#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;
}
```

##### 6.1.2.1.1.3 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"];
}
?>
```

### 6.1.2.1.2 linux

Write makefile(skip)

### 6.1.2.2 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\\\" File Name=\\\"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\\\" File Name=\\\"zoc.rar\\\""
```

### 6.1.2.3 python

Code is as follow

```
import sys
if hasattr(sys,"argv") :
    if len(sys.argv) > 2 :
        Url = sys.argv[1]
        FileName = sys.argv[2]
import starpy

Service=starpy._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\\\";File Name=\\\"XXX\\\"\" or")
    print("python http_upload.py http://127.0.0.1/XXX File Name")
    SrvGroup:_ClearService()
    starpy._ModuleExit()
    raise Exception("")

CommInterface._FileUpload(Url,FileName, FileName,"",True,"",True,None)

print("Exit...")
SrvGroup:_ClearService()
starpy._ModuleExit()
```

```
starapp -e "http_upload.py?script=python;Url=\"http://192.168.75.1/upload_file.php\";FileName=\"zoc.rar\""
```

### 6.1.3 Simple HttpServer

Implement simple WebServer, does not support dynamic script, and only support Get and Post operation.

#### 6.1.3.1 C

##### 6.1.3.1.1 Win32

##### 6.1.3.1.1.1 Create project(VC6)

skip

##### 6.1.3.1.1.2 Create and edit source file

Code is as follow

```
#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;
}

```

### 6.1.3.1.1.3 Compile and run

simple\_webserver 3040

### 6.1.3.1.2 linux

Write makefile(skip)

### 6.1.3.2 lua

Code is as follow

```
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()

```

### 6.1.3.3 python

Code is as follow

```

import sys
if hasattr(sys,"argv") :
  if len(sys.argv) > 1 :
    Port = atoi(sys.argv[1])
import starpy

starpy._InitCore(True,True,False,True,"",0,"",0)

SrvGroup = starpy._GetSrvGroup()
SrvGroup._CreateService( "", "test", "123",5,0,0,0,0,"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()
  starpy._ModuleExit()
  raise Exception("")

CommInterface.ConnetionID = CommInterface._HttpServer("",Port,100)
if CommInterface.ConnetionID == 0 :
  print("create webserver ",Port,"fail")
  SrvGroup._ClearService()
  starpy._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 starpy._KeyPress() == 27 :
        return True
    return False

starpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()

```

#### 6.1.4 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.

##### 6.1.4.1 C

##### 6.1.4.1.1 Win32

##### 6.1.4.1.1.1 Create project(VC6)

see above.

##### 6.1.4.1.1.2 Create and edit source file

Code is as follow

```

#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 STANDWEBSETERER
//--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;
}

```

#### 6.1.4.1.2 linux

Write makefile(skip)

## 6.1.4.2 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.HTTPPREQUEST_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()

```

### 6.1.4.3 python

Code is as follow

```
import sys
if hasattr(sys, "argv") :
    if len(sys.argv) > 1 :
        Port = atoi(sys.argv[1])
import starpy

starpy._InitCore(True, True, False, True, "", 0, "", 0)

SrvGroup = starpy._GetSrvGroup()
SrvGroup._CreateService( "", "test", "123", 5, 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 starpy._KeyPress() == 27 :
        return True
    if ExitFlag == 1 :
        CommInterface1._HttpLocalRequest(CommInterface1.HTTPREQUEST_GET, "/test", "", "", "")
        return True
    return False

starpy._MsgLoop( ExitProc )

print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()
```



## 6.2 WebService

To support WebService,you should open WebServer port. There are three types:

### 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

### 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

#### 6.2.1 Create WebService

##### 6.2.1.1 WebService object

1. 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](http://127.0.0.1:XXX/_ServiceName/wsdl)

WebService does not support VS\_PARAPKGPTR as parameter or return value, for it is not a structed data.

##### 6.2.1.2 lua

Examples in directoryexamples\comm.basic\ webservice.lua

```
a = Service.TestClass:_New()
```

```
a._Name = "TestObject"
```

```
a._WebServiceFlag=true
```

Code is as follows

```
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
```

### 6.2.1.3 python

Examples in directoryexamples\comm.basic\ webservice.python

```

a = Service.TestClass._New()
a._Name = "TestObject"
a._WebServiceFlag=True

```

Code is as follows:

```

import sys
import starpy
initstarcore(cle)
starpy._InitCore(True,True,False,True,"",0,"",0)

```

```

SrvGroup = starpy._GetSrvGroup()
#--create service
SrvGroup._CreateService( "", "WebServiceCallServer", "123",5,0,0,0,0,"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 starpy._KeyPress() == 27 :
        return True
    return False

starpy._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()

```

#### 6.2.1.4 C

Examples in [directoryexamples\comm.basic\ webservice.c](#)

##### 6.2.1.4.1 Win32

##### 6.2.1.4.1.1 Create project(VC6)

##### 6.2.1.4.1.2 Create and edit source file

Create source file test\_server,add to project,Code is as follow

```

#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
iParam, 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, "%sdfsaf", 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);
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;
}

```

#### 6.2.1.4.1.3 Compile and run

test\_server

#### 6.2.1.4.2 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}

```

### 6.2.2 Get WSDL of WebService

From url:

<http://127.0.0.1:3040/wsdl>

or

[http://127.0.0.1:3040/\\_WebServiceCallServer/wsdl](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="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>

```



```

</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>

```

### 6. 2. 3 WebService client(gsoap)

Examples in directory `examples\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, includes the following files:

soapC.cpp

soapH.h

soapStub.h

soapTestObjectProxy.cpp

soapTestObjectProxy.h

TestObject.nsmmap

#### 6. 2. 3. 1 Win32

##### 6.2.3.1.1 Create project(VC6)

Create new project:

Include the following files into the project.

soapTestObjectProxy.cpp  
soapC.cpp  
stdsoap2.cpp

#### 6.2.3.1.2 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;
}
```

#### 6.2.3.1.3 Compile and run

WebServiceCallServer\_Client 127.0.0.1:3040

#### 6. 2. 4 WebService client(php)

Examples in directory examples\comm.basic\ php.call.webservice.

code:

```
<?php
try{
    $client = new SoapClient('http://127.0.0.1:3040/wsdl');
    $client->__setLocation('http://127.0.0.1:3040/___WebServiceCallServer/webservice/TestObject');
```

```

$params = array('Para' => 123);
$result = $client->GetNumber($params);
print_r($result->RetValue);
}
catch(Exception $ex){
    echo $ex->getMessage();
}
?>

```

### 6.2.5 Create and use stand alone starcore service.

Examples in `directoryexamples\comm.basic\ webservice.service`

How to create service, please refer to chapter before.

#### 6.2.5.1 Called by C

##### 6.2.5.1.1 Win32

##### 6.2.5.1.1.1 create console application(VC6)

skip

##### 6.2.5.1.1.2 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 = (VS_Core_RegisterCallBackInfoProc)vs_dll_sym( hDllInstance,
VSCORE_REGISTERCALLBACKINFO_NAME );
    VSInitProc = (VS_Core_InitProc)vs_dll_sym( hDllInstance, VSCORE_INIT_NAME );
    VSTermProc = (VS_Core_TermProc)vs_dll_sym( hDllInstance, VSCORE_TERM_NAME );
    QueryControlInterfaceProc = (VS_Core_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);
    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;
}

```

### 6.2.5.1.1.3 Compile and run

test\_server\_RemoteCallServer

### 6.2.5.1.2 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_REMOTEALLSERVER_CXXSRCS := test_server_RemoteCallServer.cpp
TEST_SERVER_REMOTEALLSERVERDEFER_CXXSRCS := test_server_RemoteCallServerDefer.cpp

#####
TEST_SERVER_REMOTEALLSERVER_CXXOBJS :=
$(TEST_SERVER_REMOTEALLSERVER_CXXSRCS:%.cpp=%.o)
TEST_SERVER_REMOTEALLSERVERDEFER_CXXOBJS :=
$(TEST_SERVER_REMOTEALLSERVERDEFER_CXXSRCS:%.cpp=%.o)

#####
CXXOBJS := ${TEST_SERVER_REMOTEALLSERVER_CXXOBJS}
${TEST_SERVER_REMOTEALLSERVERDEFER_CXXOBJS}
COBJS :=

EXEC_TEST_SERVER_REMOTEALLSERVER_OBJS := ${TEST_SERVER_REMOTEALLSERVER_CXXOBJS}
EXEC_TEST_SERVER_REMOTEALLSERVERDEFER_OBJS :=
${TEST_SERVER_REMOTEALLSERVERDEFER_CXXOBJS}

#####
# Targets of the build
#####
EXEC_TEST_SERVER_REMOTEALLSERVER := test_server_RemoteCallServer_linux
EXEC_TEST_SERVER_REMOTEALLSERVERDEFER := test_server_RemoteCallServerDefer_linux

all: ${EXEC_TEST_SERVER_REMOTEALLSERVER} ${EXEC_TEST_SERVER_REMOTEALLSERVERDEFER}

#####

```

```

# Output
#####

${EXEC_TEST_SERVER_REMOTEALLSERVER}: ${TEST_SERVER_REMOTEALLSERVER_CXXOBJS}
    ${LD} -o $@ ${LDFLAGS} ${TEST_SERVER_REMOTEALLSERVER_CXXOBJS} ${LIBS} ${EXTRA_LIBS}

${EXEC_TEST_SERVER_REMOTEALLSERVERDEFER}:
${EXEC_TEST_SERVER_REMOTEALLSERVERDEFER_OBJS}
    ${LD} -o $@ ${LDFLAGS} ${EXEC_TEST_SERVER_REMOTEALLSERVERDEFER_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_REMOTEALLSERVER}
    ${EXEC_TEST_SERVER_REMOTEALLSERVERDEFER}

depend:
    #makedepend ${INCS} ${SRCS}

```

#### 6.2.5.2 called by LUA

Code is as follow

```

require "libstarcorn"
initstarcorn(cle)
if libstarcorn._InitCore(true,true,false,true,"",0,"",3008) == false then
    return
end
get service group 0, and create service
SrvGroup = libstarcorn:_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 libstarcorn._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
```

### 6.2.5.3 Called by python

Code is as follow

```

import sys
import starpy
initstarcore(cle)
starpy._InitCore(True,True,False,True,"",0,"",0)
get service group 0, and create service
SrvGroup = starpy._GetSrvGroup()
#--create service
load depended service
SrvGroup._ImportServiceWithPath("../..\\service\\script", "RemoteCallServer", True)
SrvGroup._CreateService( "", "WebServiceCallServer", "123", 5, 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 starpy._KeyPress() == 27 :
        return True
    return False

starpy._MsgLoop( ExitProc )
exit, clear service and starcore
print("Exit...")
SrvGroup._ClearService()
starpy._ModuleExit()

```

Run

```
starapp -w 3040 -e "XXXX.py?script=python"
```

## 6.3 WebService-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 :

Mapping 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

```

### 6.3.1 Create Web service using LUA

Here directly use the starcore service created in remotecall chapter.

Code is as follow

```

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
```

### 6.3.2 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 GetWsdll

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>
</definitions>
```

```

</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>

```

## 7 Starcore config and management

### 7.1 config and manage starcore

starcore is configurable. First by defines object's attribute and functions, object is transparent to other applications; Second, resources such as port number, url, and so on, is configurable.

There are two methods to config cle. One is that application uses script or c interface to config cle, which parameter includes port of telnet, output, webserver, client, and debug. The other is that application uses config files to init cle or modify config parameters by http post request.

The format of config file is as follows:

```

<?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>

```

In init function:

```
VS_Core_Init ( VS_BOOL, VS_BOOL, VS_CHAR *, VS_UINT16, VS_CHAR, VS_UINT16,
VS_STARCONFIGEX *ConfigEx );
```

Application can pass the parameter to starcore through struct VS\_STARCONFIGEX

```
typedef struct StructOfStarCoreConfigEx{
    VS_CHAR InitConfigFile[512]; //XML File
    VS_CHAR ManagerServerUrl[512];
    VS_CHAR ControlServerUrl[512];
    VS_CHAR ManagerPath[256]; // "/XXXXXX"
    VS_CHAR ControlPath[256]; // "/XXXXXX"
    VS_UUID EndPointID;
    VS_CHAR EndPointAlias[256];
    VS_UINT8 EndPointType;
    VS_INT8 Reserved[3];
    VS_CHAR EnvTag[64]; //EnvTag predefined: "" "nolooop" "activex",
    VS_INT8 Reserved1[192];
}VS_STARCONFIGEX;
```

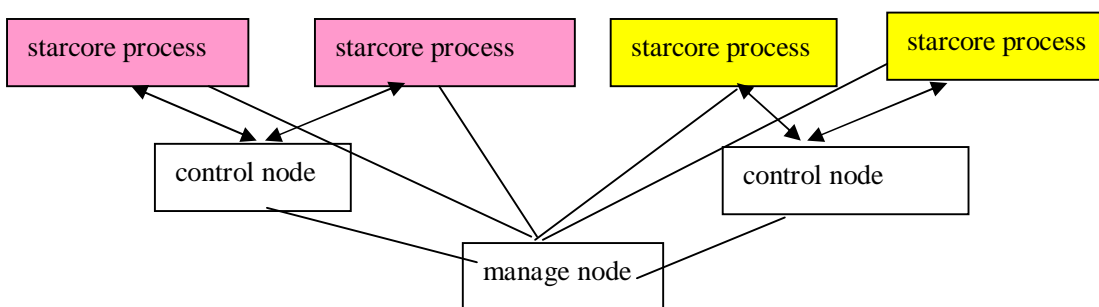
InitConfigFile:config file name, may be set to"";

ManagerServerUrl/ ManagerPath:Manager server address and directory which is any string without '/' and '\\'.

ControlServerUrl/ ControlPath:Control server address and directory, which is on local machine to control starting or stopping multiple starcore process.

EnvTag:Environment Tag, three types are predefined which are ""; nolooop(should not do msg loop); activex(application running in ActiveX).

EndPointID,EndPointAlias,EndPointType:node information, EndPointID is unique id of the starcore process in management domain. The model of management is as follows:



Control node and manage node, may be developed using any language, which communicates with starcore using http post message.

### 7.1.1 Message between control node and starcore

#### 7.1.1.1 starcore request restart

```
URL : http://ControlServerUrl/ControlPath/restart
Request : <?xml version="1.0" encoding="utf-8" ?>
  <routerinfo>
    <info source="endpoint" ID="id">
  </routerinfo>
Response : no
```

starcore process requires to restart, the message is sent from starcore to control.

Application can use function ProgramRestart() to request the process to restart. For example, the process is allocated to user A, when A upload a new version, then it can call this function to restart the process to make the new version take effect.

How to process the message depends on control node.

#### 7.1.1.2 Control config starcore with new manager parameter

```
URL : http://ControlServerUrl/ControlPath/managerconfig
Request : <?xml version="1.0" encoding="utf-8" ?>
  <routerinfo>
    <info source="control" ID="id">
  </routerinfo>
  <body>
    ManagerServerUrl
    <info managerserverurl="XXX" managerpath="XXX"/> //host may be changed by router
  </body>
Response : no
```

Control node change starcore manage node parameter. Example, for security, control node will get information from manage node and then modify the starcore.

### 7.1.2 Message between manage node and starcore

#### 7.1.2.1 echo(starcore -> Manager)

If manager parameter is valid, the starcore send echo message periodically (3 seconds) to manager node to report its status.

```
URL : http://ManagerServerUrl/ManagerPath/echo
Request : <?xml version="1.0" encoding="utf-8" ?>
  <routerinfo>
    <info source="endpoint" ID="id"/>
  </routerinfo>
  <body>
    pushflag=true,manager node should process immediately
    <info pushflag="true/false" alias="XXX" host="XXX" osalias="XXX cooperaturnum="XXX" servicename="XXX"
serviceid="XXXXX"/> //host may be changed by router
  </body>
Response : no
```

The message includes host address, os type, cooperator number, service name and service ID.

#### 7.1.2.2 get config parameter(starcore -> Manager)

```
URL : http://ManagerServerUrl/ManagerPath/getconfig
Request : <?xml version="1.0" encoding="utf-8" ?>
<routerinfo>
  <info source="endpoint" ID="id">
</routerinfo>
Response: <?xml version="1.0" encoding="utf-8" ?>
<StarCoreConfig DynamicConfig=0/1>
..
</StarCoreConfig>
```

#### 7.1.2.3 Get server Url (starcore -> Manager)

```
starcore -> manager : POST
URL : http://ManagerServerUrl/ManagerPath/serverurl
Request : <?xml version="1.0" encoding="utf-8" ?>
<routerinfo>
  <info source="endpoint" ID="id">
</routerinfo>
<body>
  <info servicename="XXX" osalias="XXX"/>
</body>
Response: <?xml version="1.0" encoding="utf-8" ?>
<body>
  <info interface="XXX" host="XXX" port="XXX"/>
</body>
```

When client calls function ConnectEx/ SConnectEx/ ConnectEx2, which only inputs servicename, to connect to server, starcore requires manager server to parse the service name and return the address of server to be connected.

#### 7.1.2.4 Get service Url (starcore -> Manager)

```
URL : http://ManagerServerUrl/ManagerPath/serviceurl --used at import service
Request : <?xml version="1.0" encoding="utf-8" ?>
<routerinfo>
  <info source="endpoint" ID="id">
</routerinfo>
<body>
  <info servicename="XXX" osalias="XXX"/>
</body>
Response: <?xml version="1.0" encoding="utf-8" ?>
<body>
  <info host="XXX"/> //http://www.XXX.XXX/XXX
</body>
```

When application calls import service function with only service name, then starcore will send request to manager node to get address where the service located

#### 7.1.2.5 Register service(starcore -> Manager)

```
URL : http://ManagerServerUrl/ManagerPath/registerserver
Request : <?xml version="1.0" encoding="utf-8" ?>
  <routerinfo>
    <info source="endpoint" ID="id">
  </routerinfo>
  <body>
    <info servicename="XXX" host="XXX" interface="XXX" port="XXX"/>
  </body>
Response: no
```

Application finishes creating service, it can call function RegisterServer to register the service to manager node.

In this case, starcore sends this message to manager node.

#### 7.1.2.6 Request to alloc cooperator(starcore -> Manager)

```
URL : http://ManagerServerUrl/ManagerPath/alloccooperator
Request : <?xml version="1.0" encoding="utf-8" ?>
  <routerinfo>
    <info source="endpoint" ID="id">
  </routerinfo>
  <body>
    <info servicename="XXX" host="XXX" interface="XXX" port="XXX"/>
  </body>
Response: no
```

For distributed application, which needs lots of cooperator to work together to finish the job, application can call AllocCooperator to alloc a cooperator. At this time, starcore send this message to manager node. The manager node should select proper node, send message to direct the node to connect the request application as cooperator.

#### 7.1.2.7 Request free cooperator(starcore -> Manager)

```
URL : http://ManagerServerUrl/ManagerPath/freecooperator
Request : <?xml version="1.0" encoding="utf-8" ?>
  <routerinfo>
    <info source="endpoint" ID="id">
  </routerinfo>
  <body>
    <info servicename="XXX" host="XXX" interface="XXX" port="XXX"/>
  </body>
Response: no
```

Application calls FreeCooperator

#### 7.1.2.8 Request alloc cooperator(Manager ->starcore)

```
URL : http://XXXXXX/ManagerPath/alloccooperator
Request : <?xml version="1.0" encoding="utf-8" ?>
  <routerinfo>
    <info source="manager" ID="id">
  </routerinfo>
```



```
<body>
  <info host="XXX" interface="XXX" port="XXX"/>
</body>
```

Response: no

Manager node send connect information to starcore, starcore will create new client service group and connect to the url.

#### 7.1.2.9 Request free cooperatort(Manager ->starcore)

URL : <http://XXXXXX/ManagerPath/freecooperator>

Request : <?xml version="1.0" encoding="utf-8" ?>

```
<routerinfo>
  <info source="manager" ID="id">
</routerinfo>
<body>
  <info host="XXX" interface="XXX" port="XXX"/>
</body>
```

Response: no

#### 7.1.2.10Get cooperatort information(Manager ->starcore)

URL : <http://XXXXXX/ManagerPath/getcooperator>

Request : <?xml version="1.0" encoding="utf-8" ?>

```
<routerinfo>
  <info source="manager" ID="id">
</routerinfo>
```

Response: <?xml version="1.0" encoding="utf-8" ?>

```
<body>
  <cooperator host="XXX" interface="XXX" port="XXX" connect="true/false"/>
</body>
```

#### 7.1.2.11Modi fy confi g(Manager ->starcore)

URL : <http://XXXXXX/ManagerPath/setconfig>

Request : <?xml version="1.0" encoding="utf-8" ?>

```
<routerinfo>
  <info source="manager" ID="id">
</routerinfo>
<StarCoreConfig DynamicConfig=0/1 Host="127.0.0.1">
  ..
</StarCoreConfig>
```

Response: <?xml version="1.0" encoding="utf-8" ?>

```
<body>
  <DebugCfgResult>true/false</DebugCfgResult>
  <DirectClientCfgResult>true/false</DirectClientCfgResult>
  <TelnetCfgResult>true/false</TelnetCfgResult>
  <WebServerCfgResult>true/false</WebServerCfgResult>
</body>
```

#### 7.1.2.12Get confi g(Manager ->starcore)

URL : <http://XXXXXX/ManagerPath/getconfig>

Request : <?xml version="1.0" encoding="utf-8" ?>

```
<routerinfo>
  <info source="manager" ID="id">
```

```

</routerinfo>
Response: <?xml version="1.0" encoding="utf-8" ?>
<StarCoreConfig DynamicConfig=0/1 Host="127.0.0.1">
..
</StarCoreConfig>

```

### 7.1.2.13 Get statistic(Manager ->starcore)

```

manager -> starcore : POST
URL : http://XXXXXX/ManagerPath/statistic
Request : <?xml version="1.0" encoding="utf-8" ?>
<routerinfo>
<info source="manager" ID="id">
</routerinfo>
Response: <?xml version="1.0" encoding="utf-8" ?>
<starcore>
..
</starcore>

```

### 7.1.3 Message between manager node and user or control node

Message between control node and manager node defines here is only a reference, not restriction.

#### 7.1.3.1 echo(control ->manager)

```

URL : http://ManagerServerUrl/ManagerPath/echo
Request : <?xml version="1.0" encoding="utf-8" ?>
<routerinfo>
<info source="control" ID="id">
</routerinfo>
<body>
pushflag=true,manager node should process immediately
<info pushflag="true/false" alias="XXX" host="XXX" osalias="XXX"/> //host may be changed by router
<router>
<endpoint ID="XXX" Alias="XXX" osalias="XXX" source="endpoint"/>
<endpoint ID="XXX" Alias="XXX" osalias="XXX" source="control"/>
</router>
<ability enableconfig="true/false" enablecontrol="true/false"/>
</body>
Response : <?xml version="1.0" encoding="utf-8" ?>
<body>
<router>
<endpoint ID="XXX" userid="XXX"/>
<endpoint ID="XXX" userid="XXX"/>
</router>
</body>

```

The response will alloc ID for each user. After that, user can use the ID to visit control node.

#### 7.1.3.2 Create service(manager/user ->control)

```

URL : http://ControlServerUrl/ManagerPath/createservice
Request : <?xml version="1.0" encoding="utf-8" ?>
<routerinfo>
<info source="manager" ID="id">

```

```

</routerinfo>
<body>
  <service ID="XXX" cmd="XXX"/>
</body>

```

Response : no

```

user -> control : POST
URL : http://ControlServerUrl/userid/createservice
Request : <?xml version="1.0" encoding="utf-8" ?>
  <body>
    <service cmd="XXX" local="true/false"/>
  </body>
Response : no

```

ID is EndPointID, which indicates the process of starcore should be loaded.

local=true,use local computer http server or ftp server address. If the address is not a public address, you should config address mapping at network gateway properly.

#### 7.1.3.3 Restart service(manager/user ->control)

```

URL : http://ControlServerUrl/ManagerPath/restartservice
Request : <?xml version="1.0" encoding="utf-8" ?>
  <routerinfo>
    <info source="manager" ID="id">
  </routerinfo>
  <body>
    <service ID="XXX" cmd="XXX"/>
  </body>
Response : no

```

```

URL : http://ControlServerUrl/userid/restartservice
Request : <?xml version="1.0" encoding="utf-8" ?>
  <body>
    <service local="true/false">
  </body>
Response : no

```

local=true,use local computer http server or ftp server address. if the address is not a public address, you should config address mapping at network gateway properly.

#### 7.1.3.4 stop service(manager/user ->control)

```

URL : http://ControlServerUrl/ManagerPath/stopservice
Request : <?xml version="1.0" encoding="utf-8" ?>
  <routerinfo>
    <info source="manager" ID="id">
  </routerinfo>
  <body>
    <service ID="XXX"/>
  </body>
Response : no

```

```

URL : http://ControlServerUrl/userid/stopservice
Request : <?xml version="1.0" encoding="utf-8" ?>
  <body/>
Response : no

```

#### 7.1.3.5 Start service (manager/user ->control)

URL : `http://ControlServerUrl/ManagerPath/startservice`

Request : `<?xml version="1.0" encoding="utf-8" ?>`

```
<routerinfo>
  <info source="manager" ID="id">
</routerinfo>
<body>
  <service ID="XXX"/>
</body>
```

Response : no

URL : `http://ControlServerUrl/userid/startservice`

Request : `<?xml version="1.0" encoding="utf-8" ?>`

```
<body>
  <service local="true/false">
</body>
```

Response : no

local=true,use local computer http server or ftp server address. if the address is not a public address, you should config address mapping at network gateway properly.

#### 7.1.3.6 Get service info(manager/user ->control)

URL : `http://ManagerServerUrl/ManagerPath/getservice`

Request : `<?xml version="1.0" encoding="utf-8" ?>`

```
<routerinfo>
  <info source="manager" ID="id">
</routerinfo>
```

Response: `<?xml version="1.0" encoding="utf-8" ?>`

```
<body>
  <service ID="XXX" dynaflag="true/false" extern="true/false" maxcooperatornum="XXX" curcooperatornum="XXX"
cmd="XXX" status="run/fail/stop" servicename="XXX" serviceid="XXXX"/>
  <service ID="XXX" dynaflag="true/false" extern="true/false" maxcooperatornum="XXX" curcooperatornum="XXX"
cmd="XXX" status="run/fail/stop" servicename="XXX" serviceid="XXXX"/>
  <service ID="XXX" dynaflag="true/false" extern="true/false" maxcooperatornum="XXX" curcooperatornum="XXX"
cmd="XXX" status="run/fail/stop" servicename="XXX" serviceid="XXXX"/>
  ...
</body>
```

URL : `http://ControlServerUrl/userid/getservice`

Request : `<?xml version="1.0" encoding="utf-8" ?>`

```
<body/>
```

Response: `<?xml version="1.0" encoding="utf-8" ?>`

```
<body>
  <service cmd="XXX" status="run/fail/stop" servicename="XXX" serviceid="XXXX"/>
</body>
```

#### 7.1.3.7 Set output port(user ->control)

URL : `http://ControlServerUrl/userid/setoutport`

Request : `<?xml version="1.0" encoding="utf-8" ?>`

```
<body>
  <output host="XXX" port="XXX"/> // if host not apperance, use source ip address as host
</body>
```

Response : no

### 7.1.3.8 Get config(user ->control)

URL : http://ControlServerUrl/userid/getcfginfo  
 Request : <?xml version="1.0" encoding="utf-8" ?>  
           <body/>  
 Response : no

### 7.1.3.9 Get control node url (user ->manager)

URL : http://ManagerServerUrl/userid/userregister  
 Request : <?xml version="1.0" encoding="utf-8" ?>  
           <body/>  
 Response: <?xml version="1.0" encoding="utf-8" ?>  
           <body>  
             <service osalias="XXX host=""/>  
           </body>

## 7.2 Simple control node:starctl

A simple control node is presented by starcore, which name is starctl.  
 starctl command.

On win32/linux, as background process.

starctl start

starctl stop

Run in console.

starctl

starctl will read config file starctlcfg.xml.

For win32 xp,the file is stored in directory X:\srplab\serverapp

For linux,the file is stored in directory /etc/srplab/serverapp.

```
<?xml version="1.0" encoding="utf-8" ?>
starcore config parameter for the control node
<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="0" />
  <DebugServer Interface="" Port="0" />
  <Comm OutputHost="" OutputPort="0" TelnetPort="0" />
  <WebServer Port="3050" 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>
local config of control node
<LocalConfig>
  node ID and name, different node should use different ID. In manager network, ID uniquely mark a node
  <SiteInfo ID="B8D5567B-7C2B-41d8-9CD0-1BB59FA1539C" Alias="star control" />
```

```

control directory
<ControlInfo Path="aaa" />
Whether service is under control of manager node, whether permit manager node to change the service config.
<ability EnableControl="true" EnableConfig="true" />
Service parameter, may be modify manually, or by manager server
<DynamicService>
  Service is allocated which user, its userid and command line. which may be modified by manager node
  <service ID="XXX" userid="XXX" cmd="XXX"/>
</DynamicService>
</LocalConfig>
manager node config
<ManagerConfig>
  manager address
  <SiteInfo Host="127.0.0.1:3060" />
  manager directory
  <ManagerInfo Path="bbb" />
</ManagerConfig>
service config info
<ServiceConfig>
  each item is run as a process
  cmd may be set"" indicates is dynamic service, which may be allocated to user by manager node
  <service ID="3BF1FE17-0B3F-451d-AF8B-B882EBC72EAC" Alias="service" extern="false" maxcoopeatornum="2"
cmd="">
  starcore config parameter of the service
  <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="0" />
    <DebugServer Interface="" Port="0" />
    <Comm OutputHost="192.168.100.3" OutputPort="514" TelnetPort="0" />
    <WebServer Port="3051" 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>
</service>
The second service.
<service ID="F26A778E-8EBA-4e8c-8AA5-74ADA941FF19" Alias="service" extern="false" maxcoopeatornum="2"
cmd="">
  <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="0" />
    <DebugServer Interface="" Port="0" />
    <Comm OutputHost="192.168.100.3" OutputPort="514" TelnetPort="3010" />
    <WebServer Port="3052" 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>
</service>
</ServiceConfig>

```

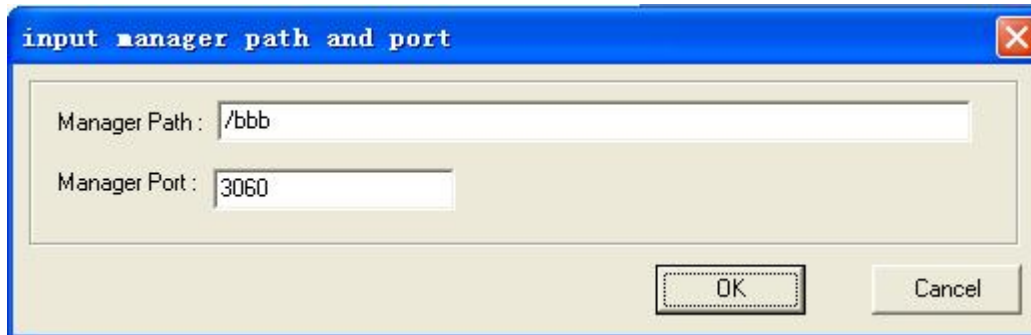
The following config, may be modified based on actual environment

Error infotmation and init information will be written into starreg.xml

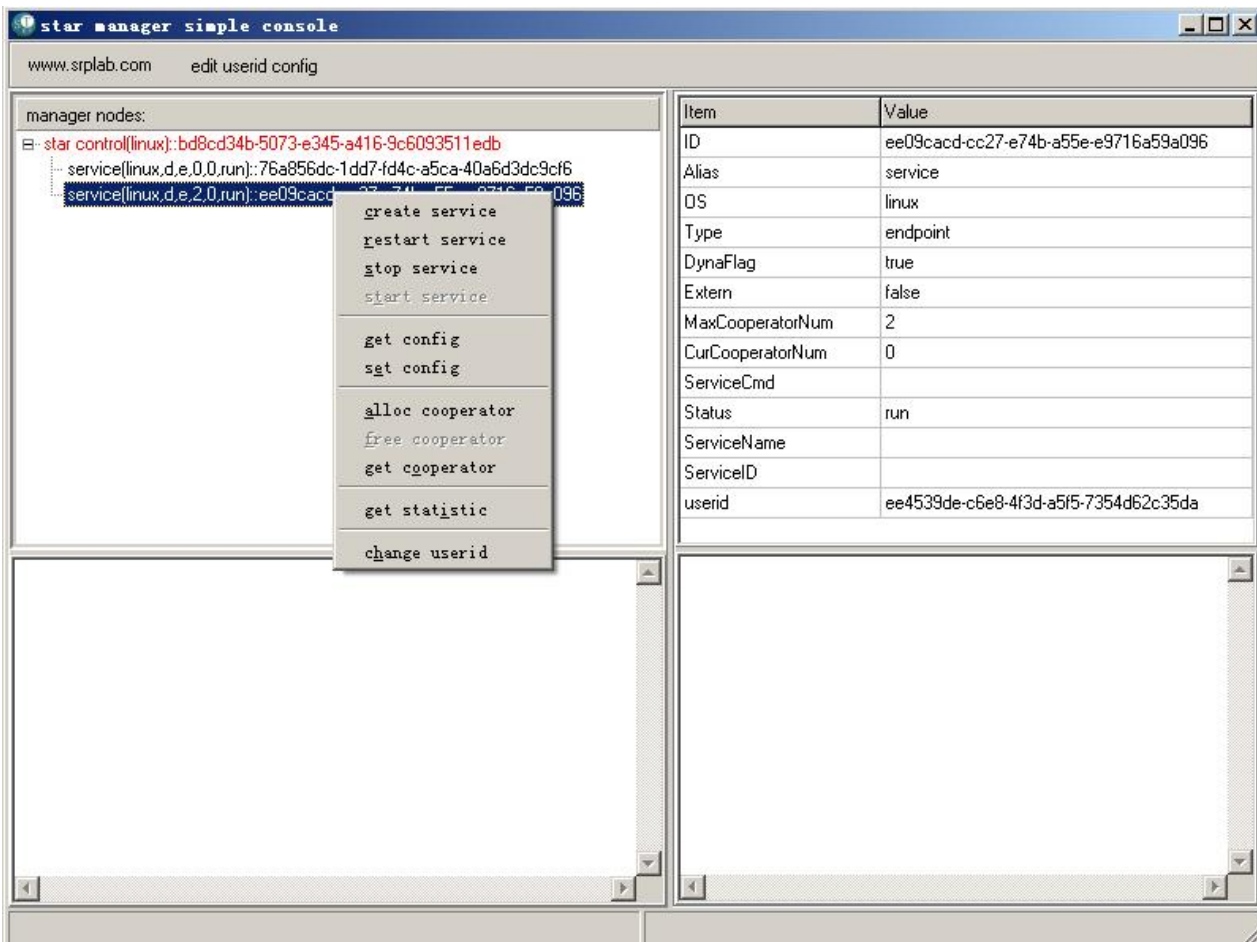
Information in period of running, will be written in file serverapp\ starctl.log.

### 7.3 simple manager node :starmconsole

starcore presents a simple manager node : starmconsole.



Input manager directory and port, and start starmconsole. Its GUI is as follows :

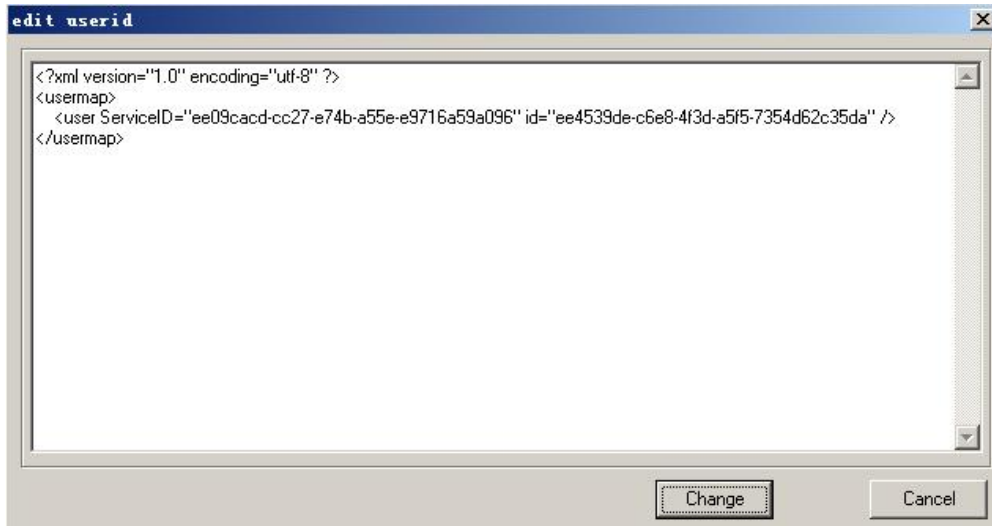


may load service files : lua/python/dll/so.

may modify config, alloc cooperaotr.

### May alloc dynamic process to user, and set user ID

1. Click edit user config,in the dialog, input xml directly.



2. Click change userid, to make the change to take effect.

#### 7.4 Simple user node :starmuser

Run by command, used by end user to set, start, stop the service process.

starmuser UserID -m ManagerServerUrl / -c ControlServerUrl [-a OutputUrl] [-o OutputPort] [--start cmd]/[--stop]/[--clear]/[--config] /[--local]

UserID: the ID of the service process allocated to user by manager node.

ManagerServerUrl/ControlServerUrl: must input one of them. If the user knows url of the control node, it maybe directly used. Or else, uses manager node Url, which will returns url of the control node.

OutputUrl, may be omitted, or else is the address to receive starcore information. If omitted, address will be captured from request by the control node.

OutputPort, may be omitted,. It should be used with OutputUrl. It is port of the server to receive syslog message.

If the ip is mapped by NAT, then the port should be configed properly. Make sure that the port can be connected from external.

--start: Start virtual process

--start: cmd Start virtual proces and load service. For example: --start [http://127.0.0.1/remotecall\\_lua.srb](http://127.0.0.1/remotecall_lua.srb)

--stop: Stop process

--clear: clear service file.

--config: get config information

--local. If you use webserver or ftp on local computer. The control node will capture the IP address from the request. If the ip is not public, and which is mapped by NAT. The address and port should be config in NAT properly form extern be able to connect. This parameter is valid when --start exists.



## 8 *Starcore packing and publishing*

### 8.1 *starcore packing*

Using starsrvpack, you can pack application and publish it on web site.

Examples in [directoryexamples\service.publish](#)

#### 8.1.1 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
```

### 8. 1. 2 Packing applications developed with c/c++

[examples\service.publish\websevice.c](#)

Applications developped with c/c++, should be compiled into sharelibs.

Take websevice as example:

The share lib should exports two function which prototype is defined in vsopenapi.h.

```
VS_BOOL StarCoreService_Init(class ClassOfStarCore *StarCore);
```

Init function, is called when sharelib is loaded.

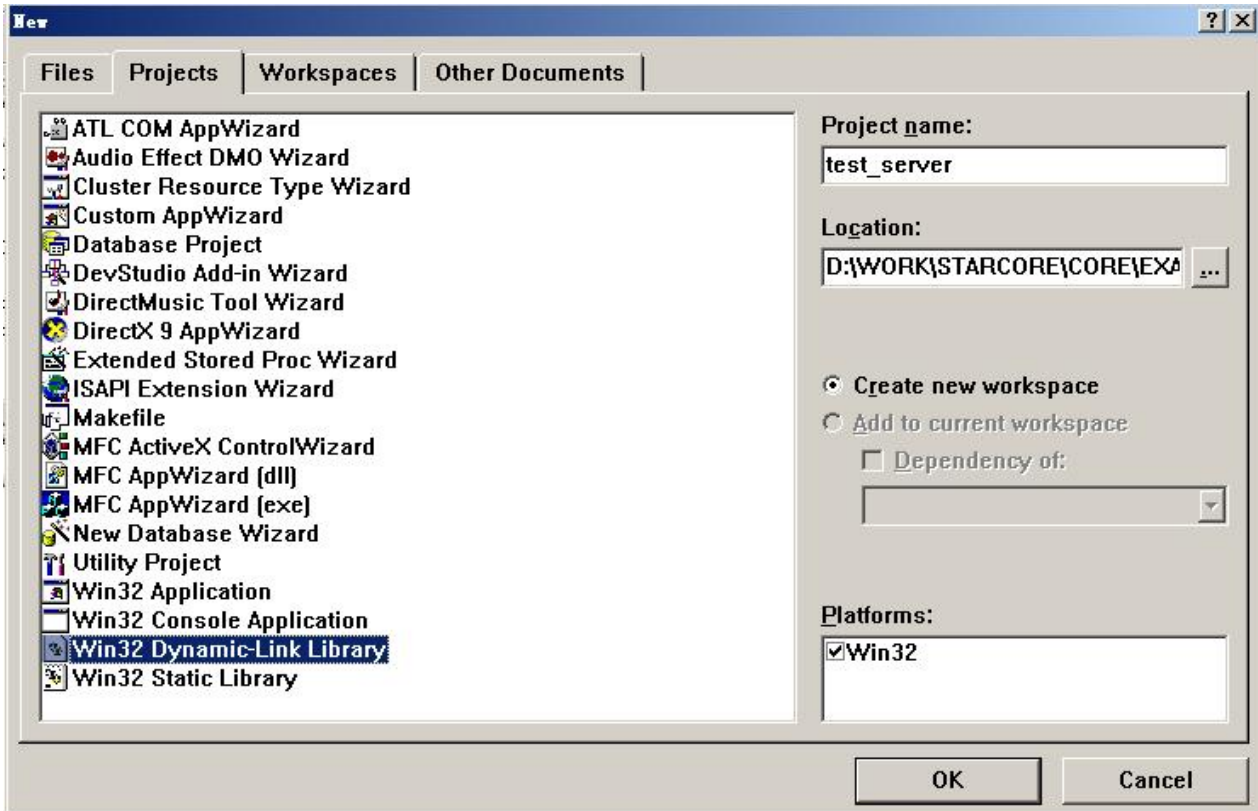
Using SRPControlInterface = StarCore ->GetControlInterface() ->Dup(); to get control interface, and then to get other interface.

```
void StarCoreService_Term(class ClassOfStarCore *StarCore);
```

Term function, called when sharelib is unloaded.

#### 8. 1. 2. 1 Win32

##### 8.1.2.1.1 Create project(VC6)



#### 8.1.2.1.2 Create and edit source file

Modify as follows. You can compare it with the previous example in 6.2.1.4.

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

//-----
/*
VS_HANDLE hDllInstance;
VS_Core_RegisterCallBackInfoProc RegisterCallBackInfoProc;
VS_Core_InitProc VSInitProc;
VS_Core_TermProc VSTermProc;
VS_Core_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);
    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();
}

```

### 8.1.2.2 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}
LDLAGS    := ${RELEASE_LDLFLAGS}
endif

ifeq (YES, ${PROFILE})
    CFLAGS := ${CFLAGS} -pg -O3
    CXXFLAGS := ${CXXFLAGS} -pg -O3
    LDLAGS := ${LDLAGS} -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_CXXOBS := $(TEST_SERVER_CXXSRCS:%.cpp=%.o)

#####
CXXOBS := ${TEST_SERVER_CXXOBS}
COBS :=

EXEC_TEST_SERVER_OBS := ${TEST_SERVER_CXXOBS}

#####
# Targets of the build
#####
OBS_PATH = .

EXEC_TEST_SERVER := ${OBS_PATH}/test_server.so

all: ${EXEC_TEST_SERVER}

#####
# Output
#####

${EXEC_TEST_SERVER}: ${EXEC_TEST_SERVER_OBS}
    ${LD} -shared -o $@ ${LDLAGS} ${EXEC_TEST_SERVER_OBS} ${LIBS} ${EXTRA_LIBS}

#####
# common rules
#####

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

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

clean:
    -rm -f ${CXXOBS} ${COBS} ${EXEC_TEST_SERVER}

```

### 8.1.2.3 Packing and testing

For binary module of C/C++, is different on win32 and linux. Therefore should set startup file separately. 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 -s win32

test:

starapp -e webservice\_c.srb

Upload webservice\_c.srb to web site, publishing for win32 and linux is finished

## 8.2 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 lib 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 download on demand.

For dynamic file, if pack to single file, they are same as static files.

`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 Binbufinterface.

How application to get data from package? The method is list below.

### 8. 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 be used 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>
```

#### 8. 2. 1. 1 C

##### 8.2.1.1.1 Win32

##### 8.2.1.1.1.1 Create project(VC6)

skip

##### 8.2.1.1.1.2 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
iParam, 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;
}

```

#### 8.2.1.1.1.3 Compile and run

starsrvpack testsingle\_pack.srprj -s win32 -i

Run:

testsingle

#### 8.2.1.1.2 linux

Write Makefile

### 8. 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.

Call interface function **GetStaticDataEx** with token set to file name

Using service SRPFSEngine, may simplify the procedure.

packing 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"/>
  </static>
</srpproject>

```

```

    <file name="s2.txt"/>
  </static>
  <dyna>
    <file name="d1.txt"/>
    <path name="bbb">
      <file name="d2.txt"/>
    </path>
  </dyna>
</srpproject>

```

Includes two dynamic files : d1.txt and bbb/d2.txt

The following code load 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()

```

#### 8.2.2.1.1.1 Run

starsrvpack testdir\_pack.srprj

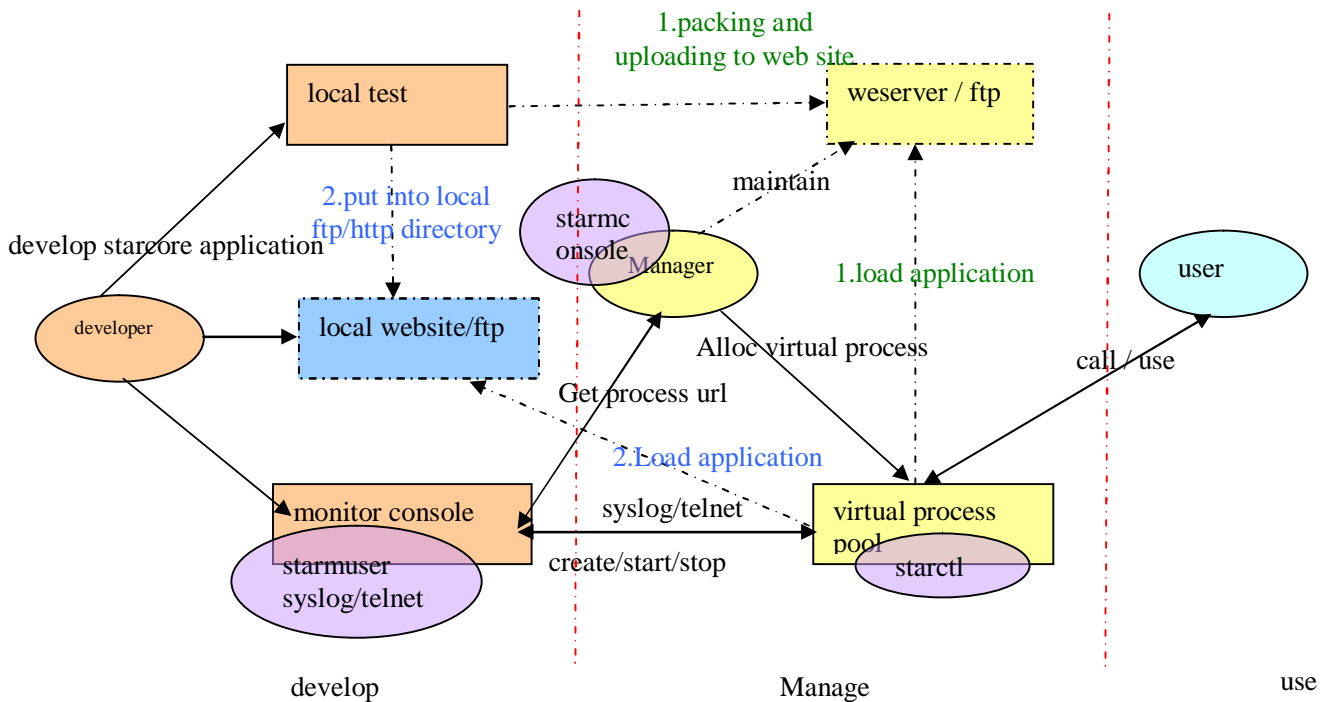
Copy directory testdir\_service to website

Run:

```
starapp -e "http://XXX/testdir_service"
```

### 8.3 Publish

publishing process is as follows:



Steps:

1. Programming and testing  
Programmer develops applications based on starcore, and tests it at local
2. packing and uploading
  - a. Packing starcore application
  - b. If the manager provides web server, then the application can be uploaded to the server. Get the external link, such as: <http://XXX/XX> or <ftp://XXX/XX>, test :  
`starapp -e "http(ftp)://XXX/XX/XXX"`
  - c. If manager does not provide server, then you can use sever on local machine. Test  
`starapp -e "http\(ftp\)://127.0.0.1/XXX/XXX".`  
 Programer should get the public address to test it on public network.

## 9 How to register

For static service(can be installed on local), client side, Activex control, applications, you need not register.

Their function has no restriction.

### 9.1 Difference between two versions

Share version presents basic and popular functions. Examples in the document, can be run with share version.

Register version provides some special features, such as object event registration, atomic object operation, service description language support, conversion between service and xml. Details of the difference is list below:

function	Explain
<i>SetSysEvent</i>	<i>Object system event process</i>
<i>RegSysEventFunction</i>	<i>Object system event process for c/c++ and scripts.</i>
<i>Active</i>	<i>Active object</i>
<i>ActiveClient</i>	<i>Active object</i>
<i>CreateNameScript</i>	<i>Obejct name script, which is lua, may be syncto client.</i>
<i>CreateNameScriptEx</i>	
<i>SetPrivateValue</i>	<i>Object private variable</i>
<i>SaveToBuf</i>	<i>Object serialization</i>
<i>SaveToFile</i>	
<i>Redirect</i>	<i>Service redirec</i>
<i>SetClientQos</i>	<i>Set client Qos</i>
<i>GetClientInfo</i>	<i>Get client info</i>
<i>RegClientOpFunction</i>	<i>Register client operation callback</i>
<i>XmlToSysRootItem</i>	<i>Service parse and unparse</i>
<i>XmlToObject</i>	
<i>ServiceToXml</i>	
<i>SysRootItemToXml</i>	
<i>ObjectToXml</i>	
<i>CreateAtomicStruct</i>	<i>Atomic functions</i>
<i>CreateAtomicObject</i>	
<i>CreateAtomicAttachAttribute</i>	
<i>CreateAtomicAttribute</i>	
<i>CreateAtomicFuncRetAttribute</i>	
<i>CreateAtomicFuncParaAttribute</i>	
<i>CreateAtomicStructAttribute</i>	
<i>SetAtomicAttributeLength</i>	
<i>SetAtomicAttributeStruct</i>	
<i>SetAtomicAttributeCombobox</i>	
<i>SetAtomicAttributeSyncFlag</i>	

<i>CreateAtomicScript</i>	
<i>CreateAtomicFunction</i>	
<i>CreateAtomicOvlFunction</i>	
<i>CreateAtomicLuaFunction</i>	
<i>CreateAtomicFunctionEx</i>	
<i>CreateAtomicInEvent</i>	
<i>CreateAtomicOutEvent</i>	
<i>SetLog</i>	<i>Objec toperation logs. used for high avaiablity</i>
<i>SetLogFile</i>	
<i>GetLogFile</i>	
<i>ClearLog</i>	
<i>ApplyLog</i>	
<i>SetNameBoolValue</i>	<i>Object name value</i>
<i>SetNameIntValue</i>	
<i>SetNameFloatValue</i>	
<i>SetNameBinValue</i>	
<i>SetNameStrValue</i>	
<i>SetNameTimeValue</i>	

## 9.2 Buy and Register

There are three kinds of license. Details please visit srplab web site.

### 9.2.1 single computer license

Run: starregister, get local machine code, and visit <http://www.srplab.com/products.htm> to get register code.

Run: starregister {register code}

### 9.2.2 Single service license

Determine service name and service ID, then visit <http://www.srplab.com/products.htm> to get register code.

After service with the name and ID is created, then uses function SetRegisterCode to pass the register code to cle.

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
```

```

false);
    starcore._PreAuthorize("test","997362D4-E961-4b7b-8472-4985958C79F6","XXXXXXXXXXXXXXXXXXXX",
    StarServiceClass Service=starcore._InitSimple1("test","997362D4-E961-4b7b-8472-4985958C79F6","123",0,0);
    starcore._SetRegisterCode("XXXXXXXXXXXXXXXXXXXX",false);
    ....
    starcore._ModuleExit();
}
}

```

### 9.2.3 Single computer service license

Determine machine number,service name and ID , then visit <http://www.srplab.com/products.htm> to get register code.

After service with the name and ID is created, then uses function SetRegisterCode to pass the register code to cle.

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._PreAuthorize("test","997362D4-E961-4b7b-8472-4985958C79F6", "XXXXXXXXXXXXXXXXXXXX",
true);
        StarServiceClass Service=starcore._InitSimple1("test","997362D4-E961-4b7b-8472-4985958C79F6","123",0,0);
        starcore._SetRegisterCode("XXXXXXXXXXXXXXXXXXXX",true);
        ....
        starcore._ModuleExit();
    }
}

```

### 9.2.4 starcore registry operation

In addition, starregister may be used to create PrivateKey, which command is:

starregister Key ValueName, Value, ValueName2, Value2,...

For example: starregister a b c

Will create Software/SRPLab/PrivateKey/a in registry and set attribute b = c.

Application may be use function \_GetRegStr("Software/SRPLab/PrivateKey/a","b","") to get attribute value.

## 9.3 Object status event

After object is created, the class will receive OnCreate event. Before object is destoried, its class will receive OnDestory event. Object can also be activated by command, and then deactivated. Which will create OnActivate and OnDeactivate event.

For example,

Capture child object event and display information. The class is implemented by java, and called by python. The examples will run correctly for register version.

java code:

```
import com.srplab.www.starcore.*;
import java.util.Hashtable;

class MyObjectClass extends StarObjectClass{
    public Object[] _OnCreate(Hashtable Ev)
    {
        if( _Tobool(Ev.get("_ThisObject")) == false ){
            System.out.println("Child Object "+Ev.get("_DesObject")+" is created");
        }
        return null;
    }
    public Object[] _OnDestroy(Hashtable Ev)
    {
        if( _Tobool(Ev.get("_ThisObject")) == false ){
            System.out.println("Child Object "+Ev.get("_DesObject")+" is destory");
        }
        return null;
    }
    public MyObjectClass(StarObjectClass srcobj){
        super(srcobj);
    }
}

public class SimpleTest{
    public static void main(String[] args){
        StarCoreFactory starcore=StarCoreFactory.GetFactory();
        // StarServiceClass Service=starcore._InitSimple("SimpleTestService","123",0,0);
        starcore._InitCore(true,true,false,true,"",0,"",0);
        StarSrvGroupClass SrvGroup = starcore._GetSrvGroup(0);
        SrvGroup._CreateService( "", "SimpleTestService", "123",5,0,0,0,0,"F0611A16-BFAA-4d0b-803F-807EC63BD265" );
        StarServiceClass Service = SrvGroup._GetService("root","123");

        ///---single service registration
        starcore._SetRegisterCode("XXXXXX",false);
        ///----single computer and single service registration
        ///starcore._SetRegisterCode("XXXXXXXXXX",true);

        Object[] AtomicClassArray = Service._CreateAtomicObjectSimple("TestItem","TestClass","", "");
        MyObjectClass ImgObj = new MyObjectClass(Service._AtomicToObject(Service._Toint(AtomicClassArray[0])));
    }
}
```



```
}
```

caller code:

```
import starpy
Service=starpy._InitSimple("test","123",0,0,"files/SimpleTest.class?script=java");
a = Service.TestClass._New();
b = Service.TestClass._New();
c = Service.TestClass._New();
Service._ServiceGroup._ClearService()
starpy._ModuleExit()
```

## *10 Redistributing with your products*

CLE is permitted to redistribute 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:starpy.dll Located at directory X:\srplab\libs  
php:php\_starcore.dll Located at directory X:\srplab\libs  
c#:star\_csharp.dll Located at directory X:\srplab\libs

linux:

libstarcore.so.1.0.1 Located at directory /usr/lib  
java:libstar\_java.so Located at directory /usr/lib  
python:starpy.so Located at directory /etc/srplab/libs  
php:compiled by your self.

For c++ and lua, you need only include libstarcore.dll or libstarcore.so.1.0.1.