Programmer's Guide

Introduction

somFree compiler is a tool to convert various interface definition languages to another one or language bindings. somFree compiler frontend is a sc or somc command which control workflow. Because somFree compiler and Emitter Framework modeled after IBM SOM Compiler from here SOM Compiler term will be used. Most of somFree Compiler and Emitter Framework and SOM Compiler and Emitter Framework are same and binary compatible at the documented level. Internal structures of somFree and IBM versions are different.

Structure of SOM Compiler and Emitter Framwork

SOM Compiler at file level consist of:

- SOM Compiler frontend
  - sc [Linux]
  - sc.exe [OS/2, Windows]
  - somc.exe [Windows]
- IDL SOM Pre-processor
  - somcpp [Linux]
  - somcpp.exe [OS/2, Windows]
- IDL SOM Compiler
  - somipc [Linux]
  - somipc.exe [OS/2, Windows]
- OIDL SOM Pre-processor
  - spp [Linux]
  - spp.exe [OS/2, Windows]
- OIDL SOM Compiler
  - somopc [Linux]
  - somopc.exe [OS/2, Windows]
- SOM Compiler Library
  - somc.so [Linux]
  - somc.dll [OS/2, Windows]
- SOM Emitter Framework
  - some.so [Linux]
  - some.dll [OS/2, Windows]
- Emitters
  - emit*.so [Linux]
  - emit*.dll [OS/2, Windows]
- Public IDL files generator
  - pdl [Linux]
  - pdl.exe [OS/2, Windows]

Currently SOM Compiler provides following emitters:

- IDL - IDL Emitter
- CSC - OIDL Emitter
- SC - OIDL public emitter
- GEN - Generic Emitter
- IR - Interface Repository Emitter
- H - C Binding public header files
- C - C Binding implementation template file
- IH - C Binding implementation header files
- XH - C++ Binding public header files
- XIH - C++ Binding implementation header files
- DEF - DEF Module Definition file
- LNK - LNK Module Linking file
- HC
- IMOD - SOM Module initialization emitter
- MODS - List of class modifiers
- PDL - Private IDL emitter
- PH
- PSC - OIDL private emitter
- UC
- UXC
- XPH
- XTM
- PAS - Pascal client library for use of SOM
- IPAS - Pascal implementation library to write SOM classes.

Some of Emitters uses Templates such as:

- cpp.efw
- ctm.efw
- gen_c.efc
- gen_c.efs
- gen_c.efw
- gen_cpp.efw
- gen_def.efw
- gen_emit.efc
- gen_emit.efs
- gen_emit.efw
- gen_emit.efx
- gen_idl.efw
- gen_make.efc
- gen_make.efs
- gen_make.efw
- gen_make.efx
- gen_mk32.efc
- gen_mk32.efs
- gen_mk32.efw
- gen_mk32.efx
- gen_mknt.efs
- gen_mknt.efw
- gen_mknt.efx
- gen_nid.efw

- gen_temp.efw
- imod.efw

Interaction of SOM Compiler components

Emitter Framework is a set of classes and SOM Compiler tool. Emitter Framework is used to produce various file formats from the SOM Interface Definition Language files. Emitter Framework classes consist of Emitter classes and Entry classes. Classes can be shadowed. This means a programmer can replace original classes with his own classes. So the SOM Compiler can be highly customized. The only things hard-coded (and closed source) are the IDL file reader and abstract graph builder.

Before starting description of Emitter Framework let's talk about SOM Compiler. We already talked briefly about SOM Compiler. But for emitters we need to know internals of SOM Compiler much better.

Let's start from visible parts of SOM Compiler that requires for its work the following files:

- sc.exe, somc.dll and somc.msg - Main part of compiler.
- somcpp.exe - SOM Preprocessor
- somipc.exe - Goals not known. Seems just execute different emitters
- emit*.dll - Emitters
- *.efw - Emitter templates

`sc.exe` is general part of compiler. Let's try to investigate some internals of sc.exe. First of all we can switch on verbose output and look on it:

```
Running shell command:
somcpp -D__OS2__  -I. -IC:\os2tk45\h -IC:\os2tk45\idl -
IC:\os2tk45\som\include \
      -D__SOMIDL_VERSION_1__ -D__SOMIDL__ -C somobj.idl >
C:\var\temp\0a500000.CTN
somipc -mppfile=C:\var\temp\0a500000.CTN -v -e emith -e emitih -e emitctm -e
emitc \
      -o somobj somobj.idl
Loading  emith.
"SOMObject"
Unloading  emith.
Loading  emitih.
"SOMObject"
Unloading  emitih.
Loading  emitctm.
"SOMObject"
Unloading  emitctm.
Loading  emitc.
"SOMObject"
Unloading  emitc.
Removed "C:\var\temp\0a500000.CTN".
```

Not so many info, but some information here. If we look at SMINCLUDE environment variable:

```
SMINCLUDE=.;C:\os2tk45\h;C:\os2tk45\idl;C:\os2tk45\som\include;
```

then we will see all paths in -I option.

Considering -D is same as for CPP we can see three symbols defined: * OS2 * SOMIDL_VERSION_1 * SOMIDL somobj.idl it is file we emitted and CTN file is output from preprocessor. The only unknown switch is -C. After small playing we can see it means “leave comments”.

So, we can try to replace somcpp with some preprocessor. In [http://www.osfree.org osFree] project we tried to use [http://mcpp.sourceforge.net MCPP] preprocessor. Results is well.

sc.exe reads SMINCLUDE variable and puts its content to -I options of somcpp.exe and redirect output to temporary file.

Ok. Now we can try to detect what is somipc.exe. If we try to execute it with command line pointed above, then we will see: Loading emith. “SOMObject” Unloading emith. Loading emitih. “SOMObject” Unloading emitih. Loading emitctm. “SOMObject” Unloading emitctm. Loading emitc. “SOMObject” Unloading emitc.

Heh. Actually, somipc.exe is a real SOM Compiler. Not sc.exe. sc.exe only prepares the input file for the compiler and handles command line and environment variables.

After some playing we can see, somipc returns 0 if all ok and -1 if error.

So, somipc.exe parses preprocessed IDL file and builds Abstract graph. From Abstract graph Object Graph are build. After this somipc.exe calls one by one all emit*.dll files according to -e switches. emit*.dll are set of DLLs with SOM classes.

Drawing here!!

SOM Compiler IDL SOM Preprocessor IDL SOM Compiler Emitter Template

```
        OIDL SOM Preprocessor    OIDL SOM Compiler
```

Разрисовать по аналогии с со структурой, что в патентах и документации по SOM, но с учетом наличия OIDL и SOMC.

SOM Compiler sc or somc is a frontend which controls basic workflow. Depending on source file extension it call or IDL or OIDL pre-processor and, after preprocessing, IDL or OIDL compiler. IDL or OIDL compiler builds abstract syntax graph using Entry structure. Entry structure contains information about entry type, pointer to object wrapper and all information about object specific attributes.

Note! Entry structure is not documented and differs in somFree and IBM SOM versions.

IDL or OIDL calls required emitters with root Entry structure on emitter entry. Emitter requests root object wrapper and, using or not using template faculty, process all graph using Object Syntax Graph. Object Syntax Graph generates required Entry objects on demand.

Emitter is a subclass of '`SOMTEmitC`' class. Emitter used to produce output file using template file from object graph of the SOM Interface Definition Language file. Physically emitter represented as DLL with name EMIT<identificator>.DLL. For C headers emith.dll emitter DLL is used. For C++ headers emitxh.dll emitter DLL is used. Emitter DLL contains only one entry with ordinal 1 and name '`emit`'.

```
SOMEXTERN FILE * SOMLINK emit(char *file, Entry * cls, Stab * stab);
```

'emit' function creates emitter object (from emitter class, which based on 'SOMTEmitC') and calls 'somtGenerateSections' method.

Usually an emitter file can be generated using 'newemit.cmd' script (can be found at Hobbes in SOMObjects toolkit).

```
newemit -C <className> <file_stem>
```

To emitter passed Entry object which is root of Object Graph. The root object can be an interface or module class. Processing of such classes slightly different.

Last part of Emitter Framework is template files. Template files allow you to make some control of emitting process. Templates are usual text files with extension *.efw. Here you can modify output for your wish. Not all emitters support templates.

So, now you have some imagination about that Emitter Framework is and how it works.

Template faculty

Emitters uses template faculty to produce output file. Template file has structure divided by sections. Each section begins from section name ended by colon. Each emitter can use its own section names. Refer to corresponding emitter and Entry classes description for section names information. Here is template file example:

```
:copyrightS
This is example template
:templateS
/* Template output example */
<className>
```

Core of Template faculty is a Key-Value strings collection represented by SOMStringTableC class. All substitutable to template values stored in SOMStringTableC class instance. On template file process, First of all SOMTEmitC method somtSetPredefinedSymbols sets section names symbols. By default it is following sections:

prologSN	prologS
baseIncludesPrologSN	baseIncludesPrologS
baseIncludesSN	baseIncludesS
baseIncludesEpilogSN	baseIncludesEpilogS
metaIncludeSN	metaIncludeS
classSN	classS
metaSN	metaS
basePrologSN	basePrologS
baseSN	baseS
baseEpilogSN	baseEpilogS
constantPrologSN	constantPrologS

constantSN	constantS
constantEpilogSN	constantEpilogS
typedefPrologSN	typedefPrologS
typedefSN	typedefS
typedefEpilogSN	typedefEpilogS
structPrologSN	structPrologS
structSN	structS
structEpilogSN	structEpilogS
unionPrologSN	unionPrologS
unionSN	unionS
unionEpilogSN	unionEpilogS
enumPrologSN	enumPrologS
enumSN	enumS
enumEpilogSN	enumEpilogS
attributePrologSN	attributePrologS
attributeSN	attributeS
attributeEpilogSN	attributeEpilogS
interfacePrologSN	interfacePrologS
interfaceSN	interfaceS
interfaceEpilogSN	interfaceEpilogS
modulePrologSN	modulePrologS
moduleSN	moduleS
moduleEpilogSN	moduleEpilogS
passthruPrologSN	passthruPrologS
passthruSN	passthruS
passthruEpilogSN	passthruEpilogS
releaseSN	releaseS
dataPrologSN	dataPrologS
dataSN	dataS
dataEpilogSN	dataEpilogS
methodsPrologSN	methodsPrologS
methodsSN	methodsS
overrideMethodsSN	overrideMethodsS
overriddenMethodsSN	overriddenMethodsS
inheritedMethodsSN	inheritedMethodsS
methodsEpilogSN	methodsEpilogS
epilogSN	epilogS

Generic Emitter

Generic emitter is a generic template based emitter. It uses simplest template with only one section “template”. Main goal of Generic Emitter is to produce Generic framework emitter files. It is used by newemit tool to produce full set of files required to build new emitter. Добавить описание символов шаблона и описание, какой шаблон за что отвечает.

DEF Emitter

DEF emitter used to generate definition file for DLL creation using MS LINK. somFree version of emitter uses template file to generate DEF file. Original IBM SOM DEF Emitter uses hard coded generation. Добавить описание символов шаблона.

LNK Emitter

LNK emitter used to generate linking file for DLL creation using Watcom WLINK. somFree version of emitter uses template file to generate LNK file. Original IBM SOM DEF Emitter doesn't have such emitter. Добавить описание символов шаблона.

CSC, PSC, SC Emitters

CSC emitter used to generate OIDL class definition file (CSC) used in IBM SOM 1.0. somFree version of emitter uses template file to generate CSC file. Original IBM SOM CSC Emitter uses hard coded generation. Добавить описание символов шаблона.

IDL, PDL Emitters

IDL emitter used to generate IDL class definition file used in IBM SOM 2.0 and higher. somFree version of emitter uses template file to generate IDL file. Original IBM SOM IDL Emitter uses hard coded generation. Добавить описание символов шаблона.

Developing new emitter

somFree Emitter Framework provides templates and libraries for developing emitters compatible with both IBM SOM 2.1 and IBM SOM 3.0 compilers. Because of different ABI (refer Appendix 1 for more information) somFree emitters automatically configures for corresponding API.

CORBA C Language mapping

somFree Compiler support CORBA C Language Mapping Specification 1.0 [1]. CORBA C Language mapping slightly differ from SOM C Language mapping, used by original IBM SOM 2.1. CORBA C Language mapping is default for somFree Compiler. This chapter provides short description of mapping. For full description refer to [1].

SOM C Language mapping

SOM C Language mapping is a IBM SOM mapping variant. For some reason (most probably because variable arguments support) IBM SOM not exactly implements C Language Mapping Specification.

Tools Reference
MKMSGF MSGEXTRT MSGBIND BIND JWASM UNI IPFC
somFree Compiler and Emitter framework
User's Guide Programmer's Guide Programmer's Reference

2024/10/09 03:43 · prokushev · 0 Comments