somFree Compiler and Emitter Framework

Оглавление

[I. User's Guide 9](#_Toc112278602)

[1. Introduction 9](#_Toc112278603)

[2. Changes 10](#_Toc112278604)

[II. Programmer's Guide 10](#_Toc112278605)

[1. Introduction 10](#_Toc112278606)

[2. Structure of SOM Compiler and Emitter Framwork 10](#_Toc112278607)

[3. Interaction of SOM Compiler components 13](#_Toc112278608)

[4. Template faculty 13](#_Toc112278609)

[5. Generic Emitter 15](#_Toc112278610)

[6. DEF Emitter 16](#_Toc112278611)

[7.LNK Emitter 16](#_Toc112278612)

[8. CSC, PSC, SC Emitters 16](#_Toc112278613)

[9. IDL, PDL Emitters 16](#_Toc112278614)

[10. Developing new emitter 16](#_Toc112278615)

[11. SOM Interface Definition Language 16](#_Toc112278616)

[12. SOM Object Interface Definition Language 16](#_Toc112278617)

[13. CORBA C Language mapping 17](#_Toc112278618)

[14. SOM C Language mapping 17](#_Toc112278619)

[III. Programmer's reference 17](#_Toc112278620)

[1. SOM Runtime C library 17](#_Toc112278621)

[2. SOM Compiler library 17](#_Toc112278622)

[somtfexists, somtfexistsSL function 17](#_Toc112278623)

[somtsearchFile, somtsearchFileSL function 18](#_Toc112278624)

[somttraverseParents, somttraverseParentsSL function 18](#_Toc112278625)

[somtloadSL function 18](#_Toc112278626)

[somtfindBaseEp, somtfindBaseEpSL function 18](#_Toc112278627)

[somtgetType, somtgetTypeSL function 18](#_Toc112278628)

[somtokfopen, somtokfopenSL function 18](#_Toc112278629)

[somtokrename, somtokrenameSL function 18](#_Toc112278630)

[somtopenEmitFile, somtopenEmitFileSL function 18](#_Toc112278631)

[somtisDbcs, somtisDbcsSL function 18](#_Toc112278632)

[somtremoveExt, somtremoveExtSL function 19](#_Toc112278633)

[somtaddExt, somtaddExtSL function 19](#_Toc112278634)

[somtarrayToPtr, somtarrayToPtrSL function 19](#_Toc112278635)

[somtattNormalise, somtattNormaliseSL function 19](#_Toc112278636)

[somtbasename, somtbasenameSL function 19](#_Toc112278637)

[somtctos, somtctosSL function 19](#_Toc112278638)

[somtdbcsPostincr, somtdbcsPostincrSL function 19](#_Toc112278639)

[somtdbcsPreincr, somtdbcsPreincrSL function 19](#_Toc112278640)

[somtdbcsStrchr, somtdbcsStrchrSL function 19](#_Toc112278641)

[somtdbcsStrrchr, somtdbcsStrrchrsL function 19](#_Toc112278642)

[somtdbcsStrstr, somtdbcsStrstrSL function 19](#_Toc112278643)

[somteptotype, somteptotypeSL function 20](#_Toc112278644)

[somtgetDesc, somtgetDescSL function 20](#_Toc112278645)

[somtgetVersion, somtgetVersionSL function 20](#_Toc112278646)

[somtgetgatt, somtgetgattSL function 20](#_Toc112278647)

[somtnextword, somtnextwordSL function 20](#_Toc112278648)

[somtnormaliseDesc, somtnormaliseDescSL function 20](#_Toc112278649)

[somtsatos, somtsatosSL function 20](#_Toc112278650)

[somtsearchFile, somtsearchFileSL function 20](#_Toc112278651)

[somtskipws, somtskipwsSL function 20](#_Toc112278652)

[somtstringFmt, somtstringFmtSL function 20](#_Toc112278653)

[somttype, somttypeSL function 20](#_Toc112278654)

[somtuniqFmt, somtuniqFmtSL function 20](#_Toc112278655)

[somtargFlag, somtargFlagSL function 21](#_Toc112278656)

[somtattjoin, somtattjoinSL function 21](#_Toc112278657)

[somtdbcsLastChar, somtdbcsLastCharSL function 21](#_Toc112278658)

[somtdbcsScan, somtdbcsScanSL function 21](#_Toc112278659)

[somtdiskFull, somtdiskFullSL function 21](#_Toc112278660)

[somtfclose, somtfcloseSL function 21](#_Toc112278661)

[somtisparent, somtisparentSL function 21](#_Toc112278662)

[somtmget, somtmgetSL function 21](#_Toc112278663)

[somtmopen, somtmopenSL function 21](#_Toc112278664)

[somtmprintf, somtmprintfSL function 21](#_Toc112278665)

[somtokremove, somtokremoveSL function 21](#_Toc112278666)

[somtunload, somtunloadSL function 21](#_Toc112278667)

[somtwriteaccess, somtwriteaccessSL function 22](#_Toc112278668)

[somtsmalloc, somtsmallocSL function 22](#_Toc112278669)

[somtaddGAtt, somtaddGAttSL function 22](#_Toc112278670)

[somtcalcFileName, somtcalcFileNameSL function 22](#_Toc112278671)

[somtcleanFilesFatal, somtcleanFilesFatalSL function 22](#_Toc112278672)

[somtemitTypes, somtemitTypesSL function 22](#_Toc112278673)

[somterror, somterrorSL function 22](#_Toc112278674)

[somtfatal, somtfatalSL function 22](#_Toc112278675)

[somtinternal, somtinternalSL function 22](#_Toc112278676)

[somtmclose, somtmcloseSL function 22](#_Toc112278677)

[somtmsg, somtmsgSL function 22](#_Toc112278678)

[somtreadDescFile, somtreadDescFileSL function 22](#_Toc112278679)

[somtsetDefaultDesc, somtsetDefaultDescSL function 22](#_Toc112278680)

[somtsetEmitSignals, somtsetEmitSignalsSL function 23](#_Toc112278681)

[somtsetTypeDefn, somtsetTypeDefnSL function 23](#_Toc112278682)

[somtshowVersion, somtshowVersionSL function 23](#_Toc112278683)

[somtsmfree, somtsmfreeSL function 23](#_Toc112278684)

[somtunsetEmitSignals, somtunsetEmitSignalsSL function 23](#_Toc112278685)

[somtwarn, somtwarnSL function 23](#_Toc112278686)

[somtuppercase, somtuppercaseSL function 23](#_Toc112278687)

[somtlowercase, somtlowercaseSL function 23](#_Toc112278688)

[somtdbcsuppercase, somtdbcsuppercaseSL function 23](#_Toc112278689)

[somtdbcslowercase, somtdbcslowercaseSL function 23](#_Toc112278690)

[somtresetEmitSignals, somtresetEmitSignalsSL function 23](#_Toc112278691)

[somtsizeofEntry, somtsizeofEntrySL function 23](#_Toc112278692)

[somtepname, somtepnameSL function 24](#_Toc112278693)

[somtgenSeqName, somtgenSeqNameSL function 24](#_Toc112278694)

[somtmrifatal, somtmrifatalSL function 24](#_Toc112278695)

[somtmriinternal, somtmriinternalSL function 24](#_Toc112278696)

[somtmrierror, somtmrierrorSL function 24](#_Toc112278697)

[somtmrimsg, somtmrimsgSL function 24](#_Toc112278698)

[somtmriwarn, somtmriwarnSL function 24](#_Toc112278699)

[somtsetInternalMessages, somtsetInternalMessagesSL function 25](#_Toc112278700)

[somtisvoid, somtisvoidSL function 25](#_Toc112278701)

[somtreturnsStruct, somtreturnsStructSL function 25](#_Toc112278702)

[somtreturnsPtr, somtreturnsPtrSL function 25](#_Toc112278703)

[somtsimpleName, somtsimpleNameSL function 25](#_Toc112278704)

[somtqualifyNames, somtqualifyNamesSL function 25](#_Toc112278705)

[somtfindBaseEpNonPtr, somtfindBaseEpNonPtrSL function 25](#_Toc112278706)

[somtprocessTraps, somtprocessTrapsSL function 25](#_Toc112278707)

[somtallocMlist, somtallocMlistSL function 25](#_Toc112278708)

[somtmlistend, somtmlistendSL function 25](#_Toc112278709)

[somtisMutRef, somtisMutRefSL function 25](#_Toc112278710)

[somtfreeMlist, somtfreeMlistSL function 25](#_Toc112278711)

[somtdupMlist, somtdupMlistSL function 25](#_Toc112278712)

[somtfreeWorld, somtfreeWorldSL function 25](#_Toc112278713)

[somtinitMalloc, somtinitMallocSL function 25](#_Toc112278714)

[somtInitialiseEmitlib. somtInitialiseEmitlibSL function 26](#_Toc112278715)

[somtInitialiseSmmeta, somtInitialiseSmmetaSL function 26](#_Toc112278716)

[somtInitialiseCreatetc, somtInitialiseCreatetcSL function 26](#_Toc112278717)

[somtInitialiseSmtypes, somtInitialiseSmtypesSL function 26](#_Toc112278718)

[somtInitialiseSomc, somtInitialiseSomcSL function 26](#_Toc112278719)

[somtInitialiseSmsmall, somtInitialiseSmsmallSL function 26](#_Toc112278720)

[somtattMap, somtattMapSL function 26](#_Toc112278721)

[somtexit, somtexitSL function 26](#_Toc112278722)

[somtdymain, somtdymainSL function 26](#_Toc112278723)

[somtaddHeader, somtaddHeaderSL function 26](#_Toc112278724)

[somtnthArg, somtnthArgSL function 26](#_Toc112278725)

[somtemitModule, somtemitModuleSL function 26](#_Toc112278726)

[somtallocDataList, somtallocDataListSL function 26](#_Toc112278727)

[somtallocMethodList, somtallocMethodListSL function 26](#_Toc112278728)

[somtclsfilename, somtclsfilenameSL function 27](#_Toc112278729)

[somtclsname, somtclsnameSL function 27](#_Toc112278730)

[somtfindMethodName, somtfindMethodNameSL function 27](#_Toc112278731)

[somtfullPrototype, somtfullPrototypeSL function 27](#_Toc112278732)

[somtfullTypedef, somtfullTypedefSL function 27](#_Toc112278733)

[somtgetNonRepeatedParent, somtgetNonRepeatedParentSL function 27](#_Toc112278734)

[somtgetatt, somtgetattSL function 27](#_Toc112278735)

[somtgetdatt, somtgetdattSL function 27](#_Toc112278736)

[somtgetAbistyle, somtgetAbistyleSL function 27](#_Toc112278737)

[somtimplicit, somtimplicitSL function 27](#_Toc112278738)

[somtimplicitArgs, somtimplicitArgsSL function 27](#_Toc112278739)

[somtincludeOnce, somtincludeOnceSL function 27](#_Toc112278740)

[somtpclsfilename, somtpclsfilenameSL function 28](#_Toc112278741)

[somtpclsname, somtpclsnameSL function 28](#_Toc112278742)

[somtprefixedPrototype, somtprefixedPrototypeSL function 28](#_Toc112278743)

[somtreplaceDataName, somtreplaceDataNameSL function 28](#_Toc112278744)

[somtrmSelf, somtrmSelfSL function 28](#_Toc112278745)

[somtshortArgList, somtshortArgListSL function 28](#_Toc112278746)

[somtimplicitMeta, somtimplicitMetaSL function 28](#_Toc112278747)

[somtlistAttribute, somtlistAttributeSL function 28](#_Toc112278748)

[somtnewMethodsCount, somtnewMethodsCountSL function 28](#_Toc112278749)

[somtprivateMethodsCount, somtprivateMethodsCountSL function 28](#_Toc112278750)

[somtaddHeader, somtaddHeaderSL function 28](#_Toc112278751)

[somtcleanFiles, somtcleanFilesSL function 28](#_Toc112278752)

[somtdeclareIdlVarargs, somtdeclareIdlVarargsSL function 28](#_Toc112278753)

[somtdymain. somtdymainSL function 28](#_Toc112278754)

[somtemitModuleTypes, somtemitModuleTypesSL function 28](#_Toc112278755)

[somtemitPassthru, somtemitPassthruSL function 29](#_Toc112278756)

[somtfreeDataList, somtfreeDataListSL function 29](#_Toc112278757)

[somtfreeMethodList, somtfreeMethodListSL function 29](#_Toc112278758)

[somtfullComment, somtfullCommentSL function 29](#_Toc112278759)

[somthandleDiskFull, somthandleDiskFullSL function 29](#_Toc112278760)

[somtinitialiseMeta, somtinitialiseMetaSL function 29](#_Toc112278761)

[somtoidlComment, somtoidlCommentSL function 29](#_Toc112278762)

[somtscmsg, somtscmsgSL function 30](#_Toc112278763)

[somtshortDefine, somtshortDefineSL function 30](#_Toc112278764)

[somtuninitialiseMeta, somtuninitialiseMetaSL function 30](#_Toc112278765)

[somtobseleteHeaderFile, somtobseleteHeaderFileSL function 30](#_Toc112278766)

[somtwidenType, somtwidenTypeSL function 30](#_Toc112278767)

[somtgenAttStubs, somtgenAttStubsSL function 30](#_Toc112278768)

[somtstrictidl, somtstrictidlSL function 30](#_Toc112278769)

[somtcreateTypeCodes, somtcreateTypeCodesSL function 30](#_Toc112278770)

[somtemitTcConstant, somtemitTcConstantSL function 30](#_Toc112278771)

[somtemitPredefinedTcConstants, somtemitPredefinedTcConstantsSL function 30](#_Toc112278772)

[somtAncestorClass, somtAncestorClassSL function 30](#_Toc112278773)

[somttcAlignment, somttcAlignmentSL function 30](#_Toc112278774)

[somttcSize, somttcSizeSL function 30](#_Toc112278775)

[somttcKind, somttcKindSL function 31](#_Toc112278776)

[somttcSeqFromListString, somttcSeqFromListStringSL function 31](#_Toc112278777)

[somtGetReintroducedMethods, somtGetReintroducedMethodsSL function 31](#_Toc112278778)

[Symbol table support functions 31](#_Toc112278779)

[somtallocBuf, somtallocBufSL function 31](#_Toc112278780)

[somtuniqString, somtuniqStringSL function 31](#_Toc112278781)

[somtkeyword, somtkeywordSL function 31](#_Toc112278782)

[somtaddEntry, somtaddEntrySL function 31](#_Toc112278783)

[somtgetEntry, somtgetEntrySL function 31](#_Toc112278784)

[somtstabFirst, somtstabFirstSL function 32](#_Toc112278785)

[somtstabNext, somtstabNextSL function 32](#_Toc112278786)

[somtstabFirstName, somtstabFirstNameSL function 32](#_Toc112278787)

[somtstabNextName, somtstabNextNameSL function 32](#_Toc112278788)

[somtcreateMemBuf, somtcreateMemBufSL function 32](#_Toc112278789)

[somtcreateStab, somtcreateStabSL function 32](#_Toc112278790)

[somticstrcmp, somticstrcmpSL function 33](#_Toc112278791)

[somtaddEntryBuf, somtaddEntryBufSL function 33](#_Toc112278792)

[somtfreeStab, somtfreeStabSL function 33](#_Toc112278793)

[3. SOM Emitter Framework 33](#_Toc112278794)

[SOMTAttributeEntryC Class 33](#_Toc112278795)

[somtGetFirst<Item> Methods 33](#_Toc112278796)

[somtGetNext<Item> Methods 33](#_Toc112278797)

[SOMTBaseClassEntryC Class 33](#_Toc112278798)

[SOMTClassEntryC Class 33](#_Toc112278799)

[somtFilterNew Method 33](#_Toc112278800)

[somtFilterOverridden Method 33](#_Toc112278801)

[somtGetFirst<Item> Methods 33](#_Toc112278802)

[somtGetNext<Item> Methods 34](#_Toc112278803)

[somtGetReleaseNameList Method 34](#_Toc112278804)

[SOMTCommonEntryC Class 34](#_Toc112278805)

[somtGetFirstArrayDimension Method 34](#_Toc112278806)

[somtGetNextArrayDimension Method 34](#_Toc112278807)

[somtIsArray Method 34](#_Toc112278808)

[somtIsPointer Method 34](#_Toc112278809)

[SOMTConstEntryC Class 34](#_Toc112278810)

[SOMTDataEntryC Class 34](#_Toc112278811)

[SOMTEmitC Class 34](#_Toc112278812)

[somtAll Method 34](#_Toc112278813)

[somtEmit<Section> Methods 34](#_Toc112278814)

[somtEmitFullPassthru Method 34](#_Toc112278815)

[somtFileSymbols Method 34](#_Toc112278816)

[somtGenerateSections Method 34](#_Toc112278817)

[somtGetFirstGlobalDefinition Method 34](#_Toc112278818)

[somtGetGlobalModifierValue Method 34](#_Toc112278819)

[somtGetNextGlobalDefinition Method 34](#_Toc112278820)

[somtImplemented Method 35](#_Toc112278821)

[somtInherited Method 35](#_Toc112278822)

[somtNew Method 35](#_Toc112278823)

[somtNewNoProc Method 35](#_Toc112278824)

[somtNewProc Method 35](#_Toc112278825)

[somtOpenSymbolsFile Method 35](#_Toc112278826)

[somtOverridden Method 35](#_Toc112278827)

[somtScan<Section> Methods 35](#_Toc112278828)

[somtSetPredefinedSymbols Method 35](#_Toc112278829)

[somtVA Method 35](#_Toc112278830)

[SOMTEntryC Class 35](#_Toc112278831)

[somtFormatModifier Method 35](#_Toc112278832)

[somtGetFirstModifier Method 35](#_Toc112278833)

[somtGetModifierList Method 35](#_Toc112278834)

[somtGetModifierValue Method 35](#_Toc112278835)

[somtGetNextModifier Method 35](#_Toc112278836)

[somtSetSymbolsOnEntry Method 35](#_Toc112278837)

[SOMTEnumEntryC Class 35](#_Toc112278838)

[somtGetFirstEnumName Method 36](#_Toc112278839)

[somtGetNextEnumName Method 36](#_Toc112278840)

[SOMTEnumNameEntryC Class 36](#_Toc112278841)

[SOMTMetaClassEntryC Class 36](#_Toc112278842)

[SOMTMethodEntryC Class 36](#_Toc112278843)

[somtGetFirst<Item> Methods 36](#_Toc112278844)

[somtGetFullCParamList Method 36](#_Toc112278845)

[somtGetFullParamNameList Method 36](#_Toc112278846)

[somtGetIDLParamList Method 36](#_Toc112278847)

[somtGetNext<Item> Methods 36](#_Toc112278848)

[somtGetNthParameter Method 36](#_Toc112278849)

[somtGetShortCParamList Method 36](#_Toc112278850)

[somtGetShortParamNameList Method 36](#_Toc112278851)

[SOMTModuleEntryC Class 36](#_Toc112278852)

[somtGetFirst<Item> Methods 36](#_Toc112278853)

[somtGetNext<Item> Methods 36](#_Toc112278854)

[SOMTParameterEntryC Class 36](#_Toc112278855)

[SOMTPassthruEntryC Class 36](#_Toc112278856)

[somtIsBeforePassthru Method 37](#_Toc112278857)

[SOMTSequenceEntryC Class 37](#_Toc112278858)

[SOMTStringEntryC Class 37](#_Toc112278859)

[SOMTStructEntryC Class 37](#_Toc112278860)

[somtGetFirstMember Method 37](#_Toc112278861)

[somtGetNextMember Method 37](#_Toc112278862)

[SOMTTemplateOutputC Class 37](#_Toc112278863)

[somtAddSectionDefinitions Method 37](#_Toc112278864)

[somtCommentStyle attribute 37](#_Toc112278865)

[somtLineLength attribute 37](#_Toc112278866)

[somtCommentNewline attribute 37](#_Toc112278867)

[somtCheckSymbol Method 37](#_Toc112278868)

[somtExpandSymbol Method 38](#_Toc112278869)

[somtGetSymbol Method 38](#_Toc112278870)

[somto Method 38](#_Toc112278871)

[somtOutputComment Method 39](#_Toc112278872)

[somtOutputSection Method 39](#_Toc112278873)

[somtReadSectionDefinitions Method 39](#_Toc112278874)

[somtSetOutputFile Method 39](#_Toc112278875)

[somtSetSymbol Method 39](#_Toc112278876)

[somtSetSymbolCopyBoth Method 39](#_Toc112278877)

[somtSetSymbolCopyName Method 40](#_Toc112278878)

[somtSetSymbolCopyValue Method 40](#_Toc112278879)

[SOMTTypedefEntryC Class 40](#_Toc112278880)

[somtGetFirstDeclarator Method 40](#_Toc112278881)

[somtGetNextDeclarator Method 40](#_Toc112278882)

[SOMTUnionEntryC Class 40](#_Toc112278883)

[somtGetFirstCaseEntry Method 40](#_Toc112278884)

[somtGetNextCaseEntry Method 40](#_Toc112278885)

[SOMTUserDefinedTypeEntryC Class 40](#_Toc112278886)

[SOMStringTableC Class 40](#_Toc112278887)

[somstTargetCapacity attribute 40](#_Toc112278888)

[somstAssociationsCount attribute 40](#_Toc112278889)

[somstAssociate method 41](#_Toc112278890)

[somstAssociateCopyKey method 41](#_Toc112278891)

[somstAssociateCopyValue method 41](#_Toc112278892)

[somstAssociateCopyBoth method 41](#_Toc112278893)

[somstGetAssociation method 41](#_Toc112278894)

[somstClearAssociation method 41](#_Toc112278895)

[somstGetIthKey method 41](#_Toc112278896)

[somstGetIthValue method 41](#_Toc112278897)

[somtStrDup function 42](#_Toc112278898)

[somtEntryTypeName function 42](#_Toc112278899)

[somtShowEntry function 42](#_Toc112278900)

[somtStrCat function 42](#_Toc112278901)

[somtMakeIncludeStr function 42](#_Toc112278902)

[somtNewSymbol function 42](#_Toc112278903)

[somtGetFileStem function 42](#_Toc112278904)

[somtGetObjectWrapper function 42](#_Toc112278905)

[IV. Appendixes 44](#_Toc112278906)

[1. Appendix 1. SOM ABI 44](#_Toc112278907)

[Список литературы 44](#_Toc112278908)

# I. User's Guide

## 1. Introduction

somFree Compiler and Emitter Framework is a free open source binary compatible reimplementation of IBM SOM Compiler and Emitter Framework. It is tries to be as compatible as possible on API and ABI level.

## 2. Changes

Most of internal structures now also present as in old IBM SOM 2.1 NT Toolkit.

New emitters:

* LNK - Open Watcom WLink support.
* DUMP - displays structures, available to emitter.
* PAS - Pascal client support.
* IPAS - Pascal implementation classes support.

SOM Compiler library now mostly documented.

SOMLINK style functions for most of SOM Compiler library added.

Emitters now IBM SOM 2.1 and IBM SOM 3.0 compatible without recompilation.

somFree Compiler supports IBM SOM 2.1, IBM SOM 3.0, and somFree 1.0 emitters.

somtShowEntry function outputs more info.

Undocumented SOMTTypes now documented.

SOMIPC now supports IDL 4.2 specification.

CORBA C Language Mapping Specification 1.0 now supported by default instead of SOM C Language mapping.

# II. Programmer's Guide

## 1. 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.

## 2. 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

## 3. Interaction of SOM Compiler components

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.

## 4. 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:

Привести к виду таблицы

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

## 5. 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.

## 6. 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.

## 7.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.

## 8. 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.

## 9. 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.

## 10. 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.

## 11. SOM Interface Definition Language

Latest IBM SOM 3.0 supports CORBA IDL mostly at level of CORBA 1.1. somFree supports CORBA IDL 4.2 with all extensions found in SOM IDL.

**Include Directives** (optional)

**Type and Constant Declarations** (optional)

**Exception Declarations** (optional)

**Interface Declarations** (optional)

**Module** declaration (optional)

## 12. SOM Object Interface Definition Language

SOM Object Interface Definition Language is a pre-IDL object definition language used before IBM SOM 2.1. Since IBM SOM 2.1 uses CORBA IDL as defined in OMG CORBA 1.1. SOM Object Interface Definition Language (OIDL) is a simple definition language and not recommended to use. SOM Compiler support is only for compatibility with old source code. OIDL support implementation mostly based on [1] and various OIDL source files found on the Web.

OIDL consist of sections set:

* Include section (optional)
* Class section (required)
* Release order section (optional)
* Parent class section (required)
* Metaclass section (optional)
* Data section (optional)
* Methods section (optional)

## 13. 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].

## 14. 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.

# III. Programmer's reference

## 1. SOM Runtime C library

SOM Runtime C library **somwm35i** is a subset of C runtime library functions found to be used by IBM SOM 3.0 for NT emitters. SOM Runtime C library provided only for support of IBM SOM 3.0 for NT emitters. This is not full featured C library but compatibility layer and must not be used for development. Functions utilize IBM Optlink calling convention. This library required only under Windows NT systems.

List of emulated function and variables.

\_CRT\_init, \_CRT\_term, \_abort\_in\_progress, \_exception\_dllinit, \_matherr, fclose, \_fprintfieee, strlen, \_sprintfieee, strcmp, strstr, \_ctype, feof, fgetc, fgets, fputs, fread, fseek, fwrite, memmove, memset, remove, rename, rewind, strchr, strcpy, strlen, strncmp, strncpy, strrchr, strtok, tolower, memcpy, strcat, getenv, \_printfieee, \_sscanfieee, exit, stderr, \_putenv, \_terminate, \_PrintErrMsg, \_SysFindFirst, \_SysFindNext, \_SysFindClose, malloc, free, strdup, strpbrk

## 2. SOM Compiler library

SOM Compiler library **somc** is a set of helper functions for compiler tasks. Used by IBM SOM emitters. Library provided solely to provide support of IBM emitters. Must not be used to write new code.

### somtfexists, somtfexistsSL function

SOMEXTERN BOOL somtfexists(char \*file);

SOMEXTERN BOOL SOMLINK somtfexistsSL(char \*file);

Check is file exists in paths.

Note: somtfexists version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtsearchFile, somtsearchFileSL function

SOMEXTERN char \* somtsearchFile(char \*file, char \*fullpath, char \*env);

SOMEXTERN char \* SOMLINK somtsearchFileSL(char \*file, char \*fullpath, char \*env);

Search path using file and env dirs and return full path if exists.

Note: somtsearchFile version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somttraverseParents, somttraverseParentsSL function

Note: somttraverseParents version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtloadSL function

SOMEXTERN EmitFn SOMLINK somtloadSL(char \*fileName, char \*functionName, void \*\*modHandle);

Load emitter <fileName> and return pointer <EmitFn> to emit or emitSL function <functionName> and return handle <modHandle> of loaded module.

This function switches **somc** to IBM SOM 3.0 ABI if emitSL function found or to IBM SOM 2.1 ABI if emit function found.

### somtfindBaseEp, somtfindBaseEpSL function

Note: somtfindBaseEp version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtgetType, somtgetTypeSL function

Note: somtGetType version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtokfopen, somtokfopenSL function

SOMEXTERN FILE \* somtokfopen(char \*path, char \*mode);

SOMEXTERN FILE \* SOMLINK somtokfopenSL(char \*path, char \*mode);

Same as C fopen function.

Note: somtokfopen version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtokrename, somtokrenameSL function

Note: somtokrename version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtopenEmitFile, somtopenEmitFileSL function

Note: somtopenEmitFile version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtisDbcs, somtisDbcsSL function

Note: somtisDbcs version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtremoveExt, somtremoveExtSL function

SOMEXTERN boolean somtremoveExt(char \*name, char \*ext, char \*buf);

SOMEXTERN boolean SOMLINK somtremoveExt(char \*name, char \*ext, char \*buf);

Remove extension from<name> and return to <buf>

Note: somtremoveExt version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtaddExt, somtaddExtSL function

SOMEXTERN char \* somtaddExt(char \*name, char \*ext, char \*buf);

SOMEXTERN char \* SOMLINK somtaddExtSL(char \*name, char \*ext, char \*buf);

Add <ext> extension to <name> filestem and return result in <buf>

Note: somtaddExt version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtarrayToPtr, somtarrayToPtrSL function

Note: somtarrayToPtr version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtattNormalise, somtattNormaliseSL function

### somtbasename, somtbasenameSL function

SOMEXTERN char \* somtbasenameSL(char \*path);

SOMEXTERN char \* SOMLINK somtbasenameSL(char \*path);

Return filename without path.

Note: somtbasename version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtctos, somtctosSL function

### somtdbcsPostincr, somtdbcsPostincrSL function

### somtdbcsPreincr, somtdbcsPreincrSL function

### somtdbcsStrchr, somtdbcsStrchrSL function

### somtdbcsStrrchr, somtdbcsStrrchrsL function

### somtdbcsStrstr, somtdbcsStrstrSL function

### somteptotype, somteptotypeSL function

### somtgetDesc, somtgetDescSL function

### somtgetVersion, somtgetVersionSL function

### somtgetgatt, somtgetgattSL function

### somtnextword, somtnextwordSL function

### somtnormaliseDesc, somtnormaliseDescSL function

### somtsatos, somtsatosSL function

### somtsearchFile, somtsearchFileSL function

Note: somtsearchFile version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtskipws, somtskipwsSL function

### somtstringFmt, somtstringFmtSL function

SOMEXTERN char \* somtstringFmtSL(char \*fmt, ...)

SOMEXTERN char \* SOMLINK somtstringFmtSL(char \*fmt, ...)

Allocate buffer for string, format it using <fmt> and return pointer to buffer.

Note: somtstringFmt version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somttype, somttypeSL function

SOMEXTERN char \* somttype(SOMTType type);

SOMEXTERN char \* SOMLINK somttypeSL(SOMTType type);

Return string representation of type of Entry structure except special case SOMTEmitterBeginE and SOMTEmitterEndE types.

Note: somttype version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

Warning: Depricated. Use somtEntryTypeName instead.

### somtuniqFmt, somtuniqFmtSL function

SOMEXTERN char \* somtuniqFmt(MemBuf \*membuf, char \*fmt, ...)

SOMEXTERN char \* SOMLINKL somtuniqFmtSL(MemBuf \*membuf, char \*fmt, ...)

Return unique formatted string.

Note: somtuniqFmt version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtargFlag, somtargFlagSL function

### somtattjoin, somtattjoinSL function

### somtdbcsLastChar, somtdbcsLastCharSL function

### somtdbcsScan, somtdbcsScanSL function

### somtdiskFull, somtdiskFullSL function

### somtfclose, somtfcloseSL function

SOMEXTERN int somtfclose(FILE \*fp);

SOMEXTERN int SOMLINK somtfcloseSL(FILE \*fp);

Same as C fclose function.

Note: somtfclose version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtisparent, somtisparentSL function

### somtmget, somtmgetSL function

### somtmopen, somtmopenSL function

### somtmprintf, somtmprintfSL function

### somtokremove, somtokremoveSL function

SOMEXTERN int somtokremove(char \*file);

SOMEXTERN int SOMLINK somtokremoveSL(char \*file);

Alias of C remove function.

Note: somtokremove version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtunload, somtunloadSL function

### somtwriteaccess, somtwriteaccessSL function

### somtsmalloc, somtsmallocSL function

SOMEXTERN void \* somtsmalloc(size\_t nbytes, BYTE clear);

SOMEXTERN void \* SOMLINK somtsmallocSL(size\_t nbytes, BYTE clear);

Allocate <nbytes> of memory and fill it by zeroes if <clear> flag is set.

Note: somtsmalloc version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtaddGAtt, somtaddGAttSL function

### somtcalcFileName, somtcalcFileNameSL function

### somtcleanFilesFatal, somtcleanFilesFatalSL function

SOMEXTERN void somtcleanFilesFatal(int status);

SOMEXTERN void SOMLINK somtcleanFilesFatalSL(int status);

Delete temporary files (if emitted file opened) and exit.

Note: somtcleanFilesFatal version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtemitTypes, somtemitTypesSL function

### somterror, somterrorSL function

### somtfatal, somtfatalSL function

### somtinternal, somtinternalSL function

### somtmclose, somtmcloseSL function

### somtmsg, somtmsgSL function

### somtreadDescFile, somtreadDescFileSL function

### somtsetDefaultDesc, somtsetDefaultDescSL function

### somtsetEmitSignals, somtsetEmitSignalsSL function

### somtsetTypeDefn, somtsetTypeDefnSL function

### somtshowVersion, somtshowVersionSL function

### somtsmfree, somtsmfreeSL function

### somtunsetEmitSignals, somtunsetEmitSignalsSL function

### somtwarn, somtwarnSL function

### somtuppercase, somtuppercaseSL function

SOMEXTERN char \* somtuppercase(char \*s, char \*buf);

SOMEXTERN char \* SOMLINK somtuppercaseSL(char \*s, char \*buf);

Convert <s> to upper case and return to <buf>.

Note: somtuppercase version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtlowercase, somtlowercaseSL function

SOMEXTERN char \* somtlowercase(char \*s, char \*buf);

SOMEXTERN char \* SOMLINK somtlowercase(char \*s, char \*buf)

Convert <s> to lower case and return to <buf>.

Note: somtlowercase version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtdbcsuppercase, somtdbcsuppercaseSL function

### somtdbcslowercase, somtdbcslowercaseSL function

### somtresetEmitSignals, somtresetEmitSignalsSL function

### somtsizeofEntry, somtsizeofEntrySL function

SOMEXTERN size\_t somtsizeofEntry(SOMTTypes type);

SOMEXTERN size\_t SOMLINK somtsizeofEntrySL(SOMTTypes type);

Return size of Entry structure for <type> of entry;

Note: somtsizeofEntry version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

Mapping of type to Entry.u type and nameis following:

| Entry type | union struct | union name |
| --- | --- | --- |
| SOMTClassE | Class | c |
| SOMTMetaE | Meta | mt |
| SOMTBaseE | Parent | p |
| SOMTPassthruE | Passthru | pt |
| SOMTNewMethodE | Method\_OR\_Data | m |
| SOMTOverrideMethodE | Method\_OR\_Data | m |
| SOMTOverriddenMethodE | Method\_OR\_Data | m |
| SOMTDataE | Method\_OR\_Data | m |
| SOMTArgumentE | Method\_OR\_Data | m |
| SOMTTypedefBE | Method\_OR\_Data | m |
| SOMTVoidPtrBE | Method\_OR\_Data | m |
| SOMTStructE | Struct | struc |
| SOMTTyDclE | Typedef | ty |
| SOMTTypedefE | Typedef | ty |
| SOMTUnionE | Union | un |
| SOMTUnionSE | Union | un |
| SOMTEnumE | Enumerator | enumerator |
| SOMTConstE | Const | con |
| SOMTAttE | Att | att |
| SOMTSequenceE | Sequence | seq |
| SOMTSequenceTDE | Sequence | seq |
| SOMTStringE | String | str |
| SOMTEnumBE | EnumName | enumN |
| SOMTModuleE | Module | mod |

### somtepname, somtepnameSL function

Note: somtgetenname version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtgenSeqName, somtgenSeqNameSL function

### somtmrifatal, somtmrifatalSL function

### somtmriinternal, somtmriinternalSL function

### somtmrierror, somtmrierrorSL function

### somtmrimsg, somtmrimsgSL function

### somtmriwarn, somtmriwarnSL function

### somtsetInternalMessages, somtsetInternalMessagesSL function

### somtisvoid, somtisvoidSL function

SOMEXTERN boolean somtisvoidSL(Entry \*type, char \*defn)

SOMEXTERN BOOL SOMLINK somtisvoidSL(Entry \*type, char \*defn)

Return TRUE if type->type is SOMTVoidBE it defn equal to "void", "VOID", "PMVOID".

Note: somtisvoid version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtreturnsStruct, somtreturnsStructSL function

### somtreturnsPtr, somtreturnsPtrSL function

### somtsimpleName, somtsimpleNameSL function

### somtqualifyNames, somtqualifyNamesSL function

### somtfindBaseEpNonPtr, somtfindBaseEpNonPtrSL function

### somtprocessTraps, somtprocessTrapsSL function

### somtallocMlist, somtallocMlistSL function

### somtmlistend, somtmlistendSL function

### somtisMutRef, somtisMutRefSL function

### somtfreeMlist, somtfreeMlistSL function

### somtdupMlist, somtdupMlistSL function

### somtfreeWorld, somtfreeWorldSL function

### somtinitMalloc, somtinitMallocSL function

SOMEXTERN void somtinitMalloc(BOOL dynamic)

SOMEXTERN void SOMLINK somtinitMallocSL(BOOL dynamic)

Initialize memory allocation/free functions.

Note: <dynamic> flag ignored in somFree version.

Note: somtinitMalloc version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtInitialiseEmitlib. somtInitialiseEmitlibSL function

### somtInitialiseSmmeta, somtInitialiseSmmetaSL function

### somtInitialiseCreatetc, somtInitialiseCreatetcSL function

### somtInitialiseSmtypes, somtInitialiseSmtypesSL function

### somtInitialiseSomc, somtInitialiseSomcSL function

### somtInitialiseSmsmall, somtInitialiseSmsmallSL function

### somtattMap, somtattMapSL function

### somtexit, somtexitSL function

### somtdymain, somtdymainSL function

### somtaddHeader, somtaddHeaderSL function

### somtnthArg, somtnthArgSL function

### somtemitModule, somtemitModuleSL function

SOMEXTERN FILE \* somtemitModule(char \*file, Entry \*cls, char \*ext);

SOMEXTERN FILE \* SOMLINK somtemitModuleSL(char \*file, Entry \*cls, char \*ext);

Same as somtopenEmitFile.

Note: somtemitModule version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtallocDataList, somtallocDataListSL function

### somtallocMethodList, somtallocMethodListSL function

### somtclsfilename, somtclsfilenameSL function

### somtclsname, somtclsnameSL function

SOMEXTERN char \* somtclsname(Entry \* cls);

SOMEXTERN char \* SOMLINK somtclsnameSL(Entry \* cls);

Return name of class <cls>.

Note: somclsname version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtfindMethodName, somtfindMethodNameSL function

### somtfullPrototype, somtfullPrototypeSL function

### somtfullTypedef, somtfullTypedefSL function

### somtgetNonRepeatedParent, somtgetNonRepeatedParentSL function

### somtgetatt, somtgetattSL function

### somtgetdatt, somtgetdattSL function

### somtgetAbistyle, somtgetAbistyleSL function

SOMEXTERN enum SOMTABIStyle somtgetAbistyle( Entry \* ep );

SOMEXTERN enum SOMTABIStyle SOMLINK somtgetAbistyleSL( Entry \* ep );

Return ABI style of Entry. At the current time returns always SOMTABIStyle\_2

Note: somtgetABIStyle version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtimplicit, somtimplicitSL function

### somtimplicitArgs, somtimplicitArgsSL function

### somtincludeOnce, somtincludeOnceSL function

SOMEXTERN char \* somtincludeOnceSL(Entry \*cls, char \*ext, char \*buf);

SOMEXTERN char \* SOMLINK somtincludeOnceSL(Entry \*cls, char \*ext, char \*buf);

Return token to <buf> for once include checks using name of class <cls> and extension <ext> in form SOM\_classname\_ext.

Note: somtincludeOnce version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtpclsfilename, somtpclsfilenameSL function

### somtpclsname, somtpclsnameSL function

### somtprefixedPrototype, somtprefixedPrototypeSL function

### somtreplaceDataName, somtreplaceDataNameSL function

### somtrmSelf, somtrmSelfSL function

### somtshortArgList, somtshortArgListSL function

### somtimplicitMeta, somtimplicitMetaSL function

### somtlistAttribute, somtlistAttributeSL function

### somtnewMethodsCount, somtnewMethodsCountSL function

### somtprivateMethodsCount, somtprivateMethodsCountSL function

### somtaddHeader, somtaddHeaderSL function

### somtcleanFiles, somtcleanFilesSL function

SOMEXTERN void somtcleanFiles(int status);

SOMEXTERN void SOMLINK somtcleanFilesSL(int status);

Delete temporary files and exit.

Note: somtcleanFiles version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtdeclareIdlVarargs, somtdeclareIdlVarargsSL function

### somtdymain. somtdymainSL function

### somtemitModuleTypes, somtemitModuleTypesSL function

### somtemitPassthru, somtemitPassthruSL function

### somtfreeDataList, somtfreeDataListSL function

### somtfreeMethodList, somtfreeMethodListSL function

### somtfullComment, somtfullCommentSL function

SOMEXTERN void somtfullCommentSL(FILE \* fp, char \*fmt,...);

SOMEXTERN void SOMLINK somtfullCommentSL(FILE \* fp, char \*fmt,...);

Outout formatted string <fmt> to emitted file as comment using C-style comment via somtoidlComment function;

Note: somtfullComment version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somthandleDiskFull, somthandleDiskFullSL function

### somtinitialiseMeta, somtinitialiseMetaSL function

### somtoidlComment, somtoidlCommentSL function

SOMEXTERN void somtoidlComment(FILE \* fp, int min, int max, char style, char \*comment);

SOMEXTERN void SOMLINK somtoidlCommentSL(FILE \* fp, int min, int max, char style, char \*comment);

Output oidl-<style> <comment> to file <fp> from colon <min> up to colon <max>.

Note: Seems IBM SOM ignores <max> value.

Style is one of following:

* '/' - each line started from "//#";
* '#' - each line started from "#";
* 'c' - C-style comment started from '/\*' and ended with '\*/'. each line started from "\*";
* 's' -each line started from "--";
* 'd' - each line started from ";";
* '+' - each line started from "//";

Other values forced to 'c' style.

<comment> can contains at offset 0 0x01 signature indicating comment style. If style is zero the used style from comment position 1. Two first symbols of comment are ignored if style signature is present.

Note: somtoidlComment version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtscmsg, somtscmsgSL function

### somtshortDefine, somtshortDefineSL function

### somtuninitialiseMeta, somtuninitialiseMetaSL function

### somtobseleteHeaderFile, somtobseleteHeaderFileSL function

SOMEXTERN FILE \* somtobseleteHeaderFile(char \*file, Entry \*cls, char \*ext, char \*newext);

SOMEXTERN FILE \* SOMLINK somtobseleteHeaderFileSL(char \*file, Entry \*cls, char \*ext, char \*newext);

Open emit file and write info about obsolete header. Return file pointer.

Note: somtoboleteHeaderFile version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtwidenType, somtwidenTypeSL function

### somtgenAttStubs, somtgenAttStubsSL function

### somtstrictidl, somtstrictidlSL function

SOMEXTERN void somtstrictidl(FILE \*fp);

SOMEXTERN void SOMLINK somtstrictidlSL(FILE \*fp);

Output definition of SOM\_STRICT\_IDL macro if somadd variable is TRUE;

Note: somtstrictidl version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

### somtcreateTypeCodes, somtcreateTypeCodesSL function

### somtemitTcConstant, somtemitTcConstantSL function

### somtemitPredefinedTcConstants, somtemitPredefinedTcConstantsSL function

### somtAncestorClass, somtAncestorClassSL function

### somttcAlignment, somttcAlignmentSL function

### somttcSize, somttcSizeSL function

### somttcKind, somttcKindSL function

### somttcSeqFromListString, somttcSeqFromListStringSL function

### somtGetReintroducedMethods, somtGetReintroducedMethodsSL function

### Symbol table support functions

#### somtallocBuf, somtallocBufSL function

Note: somtallocBuf version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtuniqString, somtuniqStringSL function

SOMEXTERN char \* somtuniqString(MemBuf \*membuf, char \*s);

SOMEXTERN char \* SOMLINK somtuniqStringSL(MemBuf \*membuf, char \*s);

Check is string unique and return NULL if not, or string itself if unique;

Note: somtuniqString version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtkeyword, somtkeywordSL function

SOMEXTERN long somtkeyword(KeytabEntry \*keytab, char \*kword, long keytabsize);

SOMEXTERN long SOMLINK somtkeywordSL(KeytabEntry \*keytab, char \*kword, long keytabsize);

Return token for keyword <kword> from keytaable <keytab> of <keytabsize> size.

Note: somtkeyword version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtaddEntry, somtaddEntrySL function

SOMEXTERN void \* somtaddEntry(Stab \*stab, char \*name, void \*ep);

SOMEXTERN void \* SOMLINK somtaddEntrySL(Stab \*stab, char \*name, void \*ep);

Add entry <ep> with name <name> to symbol table <stab>. Buffer for entry allocated by function.

Note: somtaddEntry version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtgetEntry, somtgetEntrySL function

SOMEXTERN void \* somtgetEntry(Stab \*stab, char \*name);

SOMEXTERN void \* SOMLINK somtgetEntrySL(Stab \*stab, char \*name);

Return pointer to entry structure with name equal to <name> from symbol table <stab>

Note: somtgetEntry version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtstabFirst, somtstabFirstSL function

SOMEXTERN void \* somtstabFirst(Stab \*stab, Sep \*\*sepp);

SOMEXTERN void \* SOMLINK somtstabFirstSL(Stab \*stab, Sep \*\*sepp);

Return first entry from symbol table <stab> and, optionally, returns sep entry in <sepp>.

Note: somtstabFirst version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtstabNext, somtstabNextSL function

SOMEXTERN void \* somtstabNext(Stab \*stab, Sep \*\*sepp);

SOMEXTERN void \* SOMLINK somtstabNextSL(Stab \*stab, Sep \*\*sepp);

Return next after last search entry from symbol table <stab> and, optionally, returns sep entry in <sepp>.

Note: somtstabNext version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtstabFirstName, somtstabFirstNameSL function

SOMEXTERN void \* somtstabFirstName(Stab \*stab, char \*name, Sep \*\*sepp);

SOMEXTERN void \* SOMLINK somtstabFirstNameSL(Stab \*stab, char \*name, Sep \*\*sepp);

Return first entry with <name> from symbol table <stab> and, optionally, returns sep entry in <sepp>.

Note: somtstabFirstName version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtstabNextName, somtstabNextNameSL function

SOMEXTERN void \* somtstabNextName(Stab \*stab, Sep \*\*sepp);

SOMEXTERN void \* SOMLINK somtstabNextNameSL(Stab \*stab, Sep \*\*sepp);

Return next after last search entry from symbol table <stab> and, optionally, returns sep entry in <sepp>.

Note: somtstabNextName version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtcreateMemBuf, somtcreateMemBufSL function

Note: somtcreateMemBuf version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtcreateStab, somtcreateStabSL function

SOMEXTERN void somtcreateStab(Stab \*stab, long stabsize, long entrysize);

SOMEXTERN void SOMLINK somtcreateStabSL(Stab \*stab, long stabsize, long entrysize);

Initialize symbol table structure <stab> using hash index size <stabsize> and entry size <entrysize>.

Note: somtcreateStab version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somticstrcmp, somticstrcmpSL function

SOMEXTERN int somticstrcmp(char \*s, char \*t)

SOMEXTERN int SOMLINK somticstrcmpSL(char \*s, char \*t);

Alias of C stricmp.

Note: somticstrcmp version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtaddEntryBuf, somtaddEntryBufSL function

SOMEXTERN void \* somtaddEntryBuf(Stab \*stab, char \*name, void \*ep, void \*buf, size\_t len);

SOMEXTERN void \* SOMLINK somtaddEntryBufSL(Stab \*stab, char \*name, void \*ep, void \*buf, size\_t len);

Add entry <ep> with name <name> to symbol table <stab> to buffer <buf> with size <len>

Note: somtaddEntryBuf version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

#### somtfreeStab, somtfreeStabSL function

Note: somtfreeStab version uses default compiler calling convention. For IBM SOM 3.0 for NT it is Optlink.

## 3. SOM Emitter Framework

### SOMTAttributeEntryC Class

### somtGetFirst<Item> Methods

### somtGetNext<Item> Methods

### SOMTBaseClassEntryC Class

### SOMTClassEntryC Class

### somtFilterNew Method

### somtFilterOverridden Method

### somtGetFirst<Item> Methods

### somtGetNext<Item> Methods

### somtGetReleaseNameList Method

### SOMTCommonEntryC Class

### somtGetFirstArrayDimension Method

### somtGetNextArrayDimension Method

### somtIsArray Method

### somtIsPointer Method

### SOMTConstEntryC Class

### SOMTDataEntryC Class

### SOMTEmitC Class

### somtAll Method

### somtEmit<Section> Methods

### somtEmitFullPassthru Method

### somtFileSymbols Method

### somtGenerateSections Method

### somtGetFirstGlobalDefinition Method

### somtGetGlobalModifierValue Method

### somtGetNextGlobalDefinition Method

### somtImplemented Method

### somtInherited Method

### somtNew Method

### somtNewNoProc Method

### somtNewProc Method

### somtOpenSymbolsFile Method

### somtOverridden Method

### somtScan<Section> Methods

### somtSetPredefinedSymbols Method

### somtVA Method

### SOMTEntryC Class

### somtFormatModifier Method

### somtGetFirstModifier Method

### somtGetModifierList Method

### somtGetModifierValue Method

### somtGetNextModifier Method

### somtSetSymbolsOnEntry Method

### SOMTEnumEntryC Class

### somtGetFirstEnumName Method

### somtGetNextEnumName Method

### SOMTEnumNameEntryC Class

### SOMTMetaClassEntryC Class

### SOMTMethodEntryC Class

### somtGetFirst<Item> Methods

### somtGetFullCParamList Method

### somtGetFullParamNameList Method

### somtGetIDLParamList Method

### somtGetNext<Item> Methods

### somtGetNthParameter Method

### somtGetShortCParamList Method

### somtGetShortParamNameList Method

### SOMTModuleEntryC Class

### somtGetFirst<Item> Methods

### somtGetNext<Item> Methods

### SOMTParameterEntryC Class

### SOMTPassthruEntryC Class

### somtIsBeforePassthru Method

### SOMTSequenceEntryC Class

### SOMTStringEntryC Class

### SOMTStructEntryC Class

### somtGetFirstMember Method

### somtGetNextMember Method

### SOMTTemplateOutputC Class

#### somtAddSectionDefinitions Method

void somtAddSectionDefinitions(in string defString);

Add section definitions from <defString> buffer to Symbol table.

#### somtCommentStyle attribute

attribute somtCommentStyleT somtCommentStyle;

Set style of output comment. Supported styles are:

* somtDashesE: "--" at the start of each line
* somtCPPE: C++ style, "//" at the start of each line
* somtCSimpleE: simple C style, each line wrapped in / \* and \* /
* somtCBlockE: block C style, block style, ie leading / \* then a \* on each line and then a final \* /
* somtPSimpleE: simple Pascal style, each line wrapped in (\* and \*)
* somtPBlockE: block Pascal style, block style, ie leading (\* then a \* on each line and then a final \*)
* somtPBorlandE: block Borland Pascal style, block style, ie leading { and then a final }

#### somtLineLength attribute

attribute long somtLineLength;

Line length limit. At least on list item will be output.

#### somtCommentNewline attribute

attribute boolean somtCommentNewline;

Output comment block from new line flag.

#### somtCheckSymbol Method

boolean somtCheckSymbol(in string name);

Return TRUE id symbol <name> exists in Symbol Table.

#### somtExpandSymbol Method

string somtExpandSymbol(in string s, in string buf);

#### somtGetSymbol Method

string somtGetSymbol(in string name);

Return symbol value for <name> from Symbol table.

#### somto Method

void somto(in string tmplt);

Outputs a template, <tmplt>, after substitution for any symbols that occur in it. Five substitutions are supported: simple, list, comment, tab, and conditional.

Substitutable items in the template are bracketed with angle brackets. (Backslash can be used to escape an angle bracket.)

Simple substitutions just replace a symbol with its value. If the symbol has no value in this template object then the symbol is replaced error string but no error is raised.

List substitution assumes that the symbol has a value in output template list form. This is a newline separated string of values. The list substitution specification consists of four parts, a prefix, a symbol, a separator, and a list indicator. prefixes and separators can only be composed of blanks, comma, colons, and semi-colons. The list indicator is "..." (three periods). For example, the list substitution specification "<, name, ...> has a prefix of ", ", a symbol of "name" and a separator of ", ". The prefix will be used whenever there is at least one item in the list and the separator will be used between any two list items. After the first items of a list is placed each additional item is evaluated to see if it would begin after the line length limit (set by \_set\_somtLineLength), if it would then a new line is begun and the value is placed directly under the first item.

Comment substitution assumes that the symbol has a value in output template list form. A comment specification consists of a comment indicator followed by a symbol name. The comment indicator is "--". Eg, <-- classComment> is a valid comment substitution specification. The lines of the comment are output according to the current comment style (see <somtCommentStyle>) and aligned with the starting column of the comment specification.

Tab substitution is specified by <@dd> where "dd" is a valid positive integer. Blanks will be inserted into the output stream if necessary to position the next character of output at the column indicated by "dd".

Conditional substitution is specified by putting a question mark, "?", in column one of the template line. The line will not be output at all unless at least one valid, non-blank, symbol substitution occurs on the line.

Note: Due design error in IBM SOM 3.0 this method can't be fully replaced. You can do some preprocessing of <templ> and call parent method. This is due direct usage of FILE structure in somto method. This means you can't write to file using standard C file functions because FILE structure is a compiler depended. But you don't know which compiler was used for. Header files contains compiler-independed file functions (somtok\*), but no any of this functions, except two ones, exported in SOM DLLs. So, if you want to fully replace this method then you need also replace lot of other methods and functions of Emitter Framework and SOM Compiler library. For IBM SOM 2.1 all seems to be ok, but you must use somtok\* functions from SOMC.DLL, not standard C runtime for file operations.

#### somtOutputComment Method

void somtOutputComment(in string comment);

Outputs comment using comment style settings.

Note: Due design error in IBM SOM 3.0 this method can't be fully replaced. You can do some preprocessing of <comment> and call parent or somto method. This is due direct usage of FILE structure in somto method. This means you can't write to file using standard C file functions because FILE structure is a compiler depended. But you don't know which compiler was used for. Header files contains compiler-independed file functions (somtok\*), but no any of this functions, except two ones, exported in SOM DLLs. So, if you want to fully replace this method then you need also replace lot of other methods and functions of Emitter Framework and SOM Compiler library. For IBM SOM 2.1 all seems to be ok, but you must use somtok\* functions from SOMC.DLL, not standard C runtime for file operations.

#### somtOutputSection Method

void somtOutputSection(in string sectionName);

Same as somto method, but template read from Symbol table with key equal to sectionName. Uses somto method for actual output.

#### somtReadSectionDefinitions Method

void somtReadSectionDefinitions(inout FILE fp);

This method reads sections from template file and stores them in Symbol table. fp is a value returned by somtOpenSymbolsFile method of SOMTEmitC class.

Note: Due design error in IBM SOM 3.0 this method can't be replaced. This is due unknown structure of FILE type. This means you can't read file using standard C file functions because FILE structure is a compiler depended. But you don't know which compiler was used for. Header files contains compiler-independed file functions (somtok\*), but no any of this functions, except two ones, exported in SOM DLLs. So, if you want to fully replace this method then you need also replace lot of other methods and functions of Emitter Framework and SOM Compiler library. For IBM SOM 2.1 all seems to be ok, but you must use somtok\* functions from SOMC.DLL, not standard C runtime for file operations.

#### somtSetOutputFile Method

void somtSetOutputFile(inout FILE fp);

Pass FILE structure to object to use for file I/O. fp is a value returned by somtOpenEmitFile or somtOpenEmitFileSL.

Note: FILE structure must be same as in other I/O methods and functions.

#### somtSetSymbol Method

void somtSetSymbol(in string name, in string value);

Set symbol name in Symbol table to value. name and value must be allocated using SOMMaloc function. It will be deallocated using SOMFree on object destroying.

#### somtSetSymbolCopyBoth Method

void somtSetSymbolCopyBoth(in string name, in string value);

Same as somtSetSymbol but name and value will be copied to internally allocated buffer.

#### somtSetSymbolCopyName Method

void somtSetSymbolCopyName(in string name, in string value);

Same as somtSetSymbol but name will be copied to internally allocated buffer.

#### somtSetSymbolCopyValue Method

void somtSetSymbolCopyValue(in string name, in string value);

Same as somtSetSymbol but value will be copied to internally allocated buffer.

### SOMTTypedefEntryC Class

### somtGetFirstDeclarator Method

### somtGetNextDeclarator Method

### SOMTUnionEntryC Class

#### somtGetFirstCaseEntry Method

#### somtGetNextCaseEntry Method

### SOMTUserDefinedTypeEntryC Class

### SOMStringTableC Class

interface SOMStringTableC : SOMObject

Объектами класса SOMStringTableC являются символьные таблицы, которые отображают строки на строки (ключ-значение, ассоциативные массивы). Любой экземпляр класса может хранить неограниченное число элементов. При увеличении количества строк время поиска строки увеличивается. В отличие от IBM SOM в данной реализации не используются хэш-таблицы.

#### somstTargetCapacity attribute

attribute unsigned long somstTargetCapacity;

Емкость ассоциативного массива. Значение не влияет на работу и сохранено для совместимости. В IBM SOM данный атрибут определял размер хэш-таблицы. Данный атрибут должен выставляться до вызова любого из методов данного класса

#### somstAssociationsCount attribute

readonly attribute unsigned long somstAssociationsCount;

Текущее число ассоциаций в массиве

#### somstAssociate method

short somstAssociate(in string key, in string value);

Устанавливает связь <key> и <value>. Возвращает 0, если связь не может быть установлена (<key> нулевой или недостаточно памяти); -1 - ассоциация успешна выполнена, но <key> уже имел значение до вызова метода, 1 - ассоциация успешно выполнена и <key> не существовал. Замечание: массив сохраняет ссылки на <key> и <value>, передаваемые в аргументах. Копия значений <key> и <value> не создается. При уничтожении объекта память, занимаемая <key> и <value> освобождается с помощью SOMFree, т.е. память под <key> и <value> должна быть выделена с помощью SOMMalloc и аналогичных функций. Замечание: При замене <value> при имеющемся <key> старое <value> заменаяется, память не освобождается

#### somstAssociateCopyKey method

short somstAssociateCopyKey(in string key, in string value);

То же, что и <somstAssociate>, но массив содержит копии значений <key>. Значение <key> копируется в выделяемую с помощью SOMMalloc память.

#### somstAssociateCopyValue method

short somstAssociateCopyValue(in string key, in string value);

То же, что и <somstAssociate>, но массив содержит копии значений <value>. Значение <value> копируется в выделяемую с помощью SOMMalloc память.

#### somstAssociateCopyBoth method

short somstAssociateCopyBoth(in string key, in string value);

То же, что и <somstAssociate>, но массив содержит копии значений <key> и <value>. Значения <key> и <value> копируются в выделяемую с помощью SOMMalloc память.

#### somstGetAssociation method

string somstGetAssociation(in string key);

Возвращается строка, ассоциированная с <key>, или NULL, если нет ассоциации. Массив продолжает хранить указатель на значение.

#### somstClearAssociation method

boolean somstClearAssociation(in string key);

The association for <key>, if any, is removed.1 is returned if <key> had an association, and 0 is returned if it did not.

#### somstGetIthKey method

string somstGetIthKey(in unsigned long i);

Возвращает ключевую часть <i>-й по счету ассоциации. Если нет ассоциации, то возвращает NULL. Порядок ассоциации в массиве не определен, но остается постоянным до следующей модификации.

#### somstGetIthValue method

string somstGetIthValue(in unsigned long i);

Возвращает значимую часть <i>-й по счету ассоциации. Если нет ассоциации, то возвращает NULL. Порядок ассоциации в массиве не определен, но остается постоянным до следующей модификации.

### somtStrDup function

SOMEXTERN char \* SOMLINK somtStrDup(char \*str);

Allocate memory and duplicate string str

### somtEntryTypeName function

SOMEXTERN char \* SOMLINK somtEntryTypeName(SOMTTypes type);

Return string representation of type of Entry structure except special case SOMTEmitterBeginE and SOMTEmitterEndE types.

### somtShowEntry function

SOMEXTERN void SOMLINK somtShowEntry(Entry \* ep);

Output using somPrintf information about Entry structure.

### somtStrCat function

SOMEXTERN char \* SOMLINK somtStrCat(int count,...);

Concatenate count of strings.

### somtMakeIncludeStr function

SOMEXTERN char \* SOMLINK somtMakeIncludeStr(boolean local, char \*stem, char \*suffix);

Produce include string for local (include "") or global (include <>) using file stem as file name and suffix as file extension.

### somtNewSymbol function

SOMEXTERN char \* SOMLINK somtNewSymbol(char \*prefix, char \*stem);

Allocate memory and produce string from prefix and stem.

### somtGetFileStem function

SOMEXTERN char \* SOMLINK somtGetFileStem(char \*fullName);

Allocate memory and return file stem from file name.

### somtGetObjectWrapper function

SOMEXTERN SOMTEntryC \* SOMLINK somtGetObjectWrapper(Entry \* ep);

Return SOMT\*EntryC object for ep Entry structure.

Mapping of Entry types to SOMT\*EntryC classes:

| **Entry type** | **Emitter Framework Class** |
| --- | --- |
| SOMTArgumentE | SOMTParameterEntryC |
| SOMTAttE | SOMTAttributeEntryC |
| SOMTBadEntryE | Fatal error |
| SOMTBaseE | SOMTBaseClassEntryC |
| SOMTClassE | SOMTClassEntryC |
| SOMTConstE | SOMTConstEntryC |
| SOMTDataE | SOMTDataEntryC |
| SOMTEnumBE | SOMTEnumNameEntryC |
| SOMTEnumE | SOMTEnumEntryC |
| SOMTEnumPE | SOMTEnumEntryC |
| SOMTFloatBE | SOMTEntryC |
| SOMTAnyBE | SOMTEntryC |
| SOMTGroupE | SOMTEntryC |
| SOMTCopyrightE | SOMTEntryC |
| SOMTLongBE | SOMTEntryC |
| SOMTNegativeBE | SOMTEntryC |
| SOMTOctetBE | SOMTEntryC |
| SOMTTypeCodeBE | SOMTEntryC |
| SOMTBooleanBE | SOMTEntryC |
| SOMTCaseEntryE | SOMTEntryC |
| SOMTCaseListE | SOMTEntryC |
| SOMTCaseSTME | SOMTEntryC |
| SOMTCharBE | SOMTEntryC |
| SOMTDclListE | SOMTEntryC |
| SOMTDefaultE | SOMTEntryC |
| SOMTDoubleBE | SOMTEntryC |
| SOMTEBaseE | SOMTEntryC |
| SOMTEEnumE | SOMTEntryC |
| SOMTShortBE | SOMTEntryC |
| SOMTStringBE | SOMTEntryC |
| SOMTUnsignedLongBE | SOMTEntryC |
| SOMTUnsignedShortBE | SOMTEntryC |
| SOMTVoidBE | SOMTEntryC |
| SOMTVoidPtrBE | SOMTEntryC |
| SOMTMetaE | SOMTMetaClassEntryC |
| SOMTModuleE | SOMTModuleEntryC |
| SOMTNewMethodE | SOMTMethodEntryC |
| SOMTOverriddenMethodE | SOMTMethodEntryC |
| SOMTOverrideMethodE | SOMTMethodEntryC |
| SOMTPassthruE | SOMTPassthruEntryC |
| SOMTSequenceE | SOMTSequenceEntryC |
| SOMTSequenceTDE | SOMTSequenceEntryC |
| SOMTStringE | SOMTStringEntryC |
| SOMTStructE | SOMTStructEntryC |
| SOMTStructPE | SOMTStructEntryC |
| SOMTStructSE | SOMTStructEntryC |
| SOMTTyDclE | SOMTTypedefEntryC |
| SOMTTypedefE | SOMTTypedefEntryC |
| SOMTTypedefBE | SOMTUserDefinedTypeEntryC |
| SOMTUnionE | SOMTUnionEntryC |
| SOMTUnionPE | SOMTUnionEntryC |
| SOMTUnionSE | SOMTUnionEntryC |
| SOMTEmitterBeginE | Fatal error |
| SOMTEmitterEndE | Fatal error |

# IV. Appendixes

## 1. Appendix 1. SOM ABI

Due switching from MSVC (IBM SOM 2.1) to VAC (IBM SOM 3.0) some problems was occur:

First problem is a calling convention. All non SOMLINK calls in IBM SOM 2.1 is a \_cdecl calls. But under IBM SOM 3.0 all non SOMLINK calls is a Optlink calls. Read some info here:

https://github.com/prokushev/SOM-Delphi-Wiki/blob/master/Known%20differences%20between%20SOM%202.1%20and%20SOM%203.0.md

Goal of somFree SOM Compiler and Emitter Framework is to provide a possibility to use original IBM SOM emitters as from IBM SOM 2.1 as from IBM SOM 3.0. Another goal is a development of somFree emitters, which can be used on both IBM SOM 2.1 and IBM SOM 3.0 compilers.To achieve above goals somFree provides some solutions:

1. Automatic somc.dll calling convention switching.

somFree SOMC.DLL provides automatic switching of IBM SOM 2.1 ABI and IBM SOM 3.0 ABI. Switching occurs on somtload call during loading of emitter. For IBM SOM 3.0 all emitter contains entry point emitSL, so, if loading was success, then somFree handles Optlink calling convention for all non SOMLINK calls. If no such entry (found only emit) then IBM SOM 2.1 ABI used.

2. Support both entry points (emitSL and emit) in emitters.

somFree emitters automatically switches to IBM SOM 2.1 ABI on emit call and to IBM SOM 3.0 ABI on emitSL call.

# Список литературы

|  |  |
| --- | --- |
| [1] | Object Management Group, «C Langue Mapping Specification 1.0,» [В Интернете]. Available: https://www.omg.org/spec/C/. [Дата обращения: 24 Август 2022]. |
| [2] | IBM, OS/2 2.0 Technical Library. System Object Model Guide and Reference. First Edition., 1991. |