Starting Help

C++ Add-In will guarantee to maximize the productivity and quality because it helps a tool to adapt C++ code generation and reverse engineering easily and quickly after it transforms UML model to C++ source and C++ source to UML model.

Index of C++ Add-In Help

- 1. <u>C++ Add-In Overview</u>
- 2. <u>C++ Profile</u>
- 3. <u>C++ Framework</u>
- 4. Option Configurations
- 5. <u>C++ Reverse Engineering</u>
- 6. <u>C++ Code Generation</u>
- 7. <u>FAQ</u>

C++ Add-In Overview

Top Previous Next

This chapter contains a general overview of C++ Add-In: functions and configuration procedures.

- <u>C++ Add-In Functions</u>
- <u>C++ Add-In Configurations</u>

C++ Add-In Functions

C++ Add-In provides the following functions.

C++ Profile

C++ profile is provided to allow visual modeling of C++ concepts such as package, class, interface, imports, modifier and so on in StarUML(tm). Include the C++ profile when you start your StarUML(tm) project, in order to apply the features of C++ language in your software modeling.

C++ Model Framework

C++ Add-In provides Microsoft MFC 6.0 in the Model Framework format.

C++ Reverse Engineering

C++ Add-In provides the reverse engineering function that generates StarUML(tm) models by analyzing C++ codes.

C++ Code Generation

C++ Add-In provides the forward engineering function that generates C++ codes by analyzing StarUML(tm) models.

C++ Add-In Configurations

Once C++ Add-In is installed, it is enabled for use in StarUML(tm) by default. Installed Add-Ins can be enabled or disabled through Add-In Manager in StarUML(tm). If an Add-In is disabled, no main or popup menu items related to it are displayed, and no StarUML(tm) events are relayed to it.

Procedure for Enabling C++ Add-In:

- 1. Select the **[Tools]->[Add-In Manager...]** menu in StarUML(tm).
- 2. At the Add-In Manager dialog box, check the "C++ Add-In" checkbox in the Add-In list.

Name	Version	
🗹 🚓 C++ Add-In	1.0.1.68	
🗹 🧬 C# Add-In	1.0.1.70	
🗹 🏝 Java Add-In	1.0.1.94	
🗹 🚫 Pattern Add-In	1.0.1.63	_
🗹 冬 Rose Add-In	1.0.1.64	
🗹 🍪 Default Extension Pack	1.0.1.314	
 		

3. Click the **[OK]** button to close the dialog box.

Procedure for Disabling C++ Add-In:

- 1. Select the **[Tools] -> [Add-In Manager...]** menu in StarUML(tm).
- 2. At the Add-In Manager dialog box, uncheck the "C++ Add-In" checkbox in the Add-In list.
- 3. Click the **[OK]** button to close the dialog box.

C++ Profile

This chapter describes C++ profile: configuration procedures and definitions.

- <u>Including C++ Profile</u>
- Excluding C++ Profile
- <u>C++ Profile Definition List</u>

Including C++ Profile

C++ profile must be included in the project in order to utilize the Stereotypes, TagDefinitions, and DataTypes defined in C++ profile.

Procedure for Including C++ Profile :

- 1. Select the **[Model]->[Profiles...]** menu.
- 2. At the Profile Manager window, select "C++ Profile" from the "Available profiles" list on the left.

Profile		×
Available profiles:	Included grofiles: Include > In	
Description:		
C++ Language Profile	Close	Help

- 3. Click the **[Include]** button or hit Alt-I to move "C++ Profile" to the "Included profiles" list.
- 4. Click the **[Close]** button to close the Profile Manager window.
- 5. C++ profile is included in the current project.

Note

Opening a project with C++ profile on another StarUML(tm) system that

• does not have C++ profile installed may result in loss of extension information of model elements (Stereotype, Tag definition, etc).

Excluding C++ Profile

C++ profile can be excluded from the current project. Once C++ profile is excluded, Stereotypes, TagDefinitions and DataTypes defined in the profile cannot be used in the project.

Procedure for Excluding C++ Profile :

- 1. Select the **[Model]->[Profiles...]** menu.
- 2. At the Profile Manager window, select "C++ Profile" from the "Included profiles" list on the right.

Profile		×
Available profiles:	Included grofiles: Include > In	
Description:		
C++ Language Profile	Close	Help

- 3. Click the **[Exclude]** button or hit Alt-E to remove "C++ Profile" from the "Included profiles" list.
- 4. Click the **[Close]** button to close the Profile Manager window.
- 5. C++ profile is excluded from the current project.

Note

• Re-including C++ profile after excluding it does not restore the previously edited tag definitions of the model elements.

Profile Definition List

Stereotype

C++ profile contains definitions for the following stereotypes.

Sterotype	Target Element	Description
< <cppsourcefile>></cppsourcefile>	Component	Source file with C++ code
< <cppheaderfile>></cppheaderfile>	Component	C++ header file
< <cppstruct>></cppstruct>	Class	Indicates struct, that is a compound type similar to class.
< <cppunion>></cppunion>	Class	Indicates union, that can substitute value for only one member at a time.
< <cppoperator>></cppoperator>	Operation	Indicates operator function that overrides operator.
< <cppmacro>></cppmacro>	Operation	Used for expressing a macro that defines a part of member definition such as class as a UML operation element.
< <cppusing>></cppusing>	Dependency	Used for referencing other namespace with the using syntax.
< <cppfriend>></cppfriend>	Dependency, Operation	Used for expressing friend of class as dependency element or expressing friend function.
< <cpptypedef>></cpptypedef>	Class	Used for expressing typedef, which defines the type, as class element.
< <cppsynonym>></cppsynonym>	Dependency	Indicates the dependency between the typedef class and the original type when defining a type with a typedef of a different name.
< <cppdelegate>></cppdelegate>	Class	Indicates delegate function defined in .NET managed C++.

TagDefinition

C++ profile contains definitions for the following tag definitions.

TagDefinition	Туре	Target Element	Description

CppVirtual		Generalization, Realization, Operation	The CppVirtual tag definition in Generalization or Realization indicates that the parent class is a virtual base class. If the CppVirtual tag definition is applied to an operation, it indicates that the C++ member function is a virtual function that can be redefined by the derived class.
CppStatic	Boolean	AssociationEnd	Indicates that the C++ data member matching with AssociationEnd is a static data member that shares only one copy for all instances of the class.
CppMutable	Boolean	Attribute, AssociationEnd	Indicates that the C++ data member matching with Attribute or AssociationEnd is a mutable data member that can be changed by the const member function.
CppVolatile	Boolean	Attribute, AssociationEnd	Indicates that the C++ data member matching with Attribute or AssociationEnd is a volatile variable.
CppCollection	String	Attribute, AssociationEnd	Defines the C++ Collection type name. The CppConst tag definition used in Operation element
			indicates that the C++

CppConst	Boolean	Operation, Parameter	function is a read-only function. If the CppConst tag definition is used in Parameter element, it indicates that the parameter cannot be changed within the function.
CppInline	Boolean	Operation	Indicates that the operation element is an inline function that compiles the function body by inserting it in the place where the function is called.
CppDimension	Integer	Attribute, AssociationEnd, Operation, Parameter	Indicates the array dimension of the declared object.
CppBitField	Integer	Attribute	Indicates the bit field of the data member.
CppEnumLiteralValue	String	EnumerationLiteral	Records the value of each item for C++ enumeration (enum) objects.
CppPointer	String	Attribute, Parameter	Indicates the pointer symbols (*, &, etc.) for Attributes and Parameters.
CppFunctionTemplate	String	Operation	Records the template declaration for template function.
CppThrow	String	Operation	Records the expression in the throw clause for function.
CppTypedefPointer	String	Class	Indicates the pointer symbols (*, &, etc.) for typedef declaration.
CppTypedefDefinition	String	Class	Records the type declaration for typedef

			declaration.
CppDeclspec	String	Class, Interface, Attribute, AssociationEnd, Operation	Records the modifier list ofdeclspec keyword for the tag definition grammar used in Microsoft Visual C++.
CppGc	Boolean	Class, Interface, Attribute, AssociationEnd	Indicates the <u>gc</u> keyword defined in Microsoft's Managed Extension for C++.
CppNogc	Boolean	Class, Interface, Attribute, AssociationEnd	Indicates thenogc keyword defined in Microsoft's Managed Extension for C++.
CppValue	Boolean	Class	Indicates thevalue keyword defined in Microsoft's Managed Extension for C++.
CppEvent	IROOIDON	Interface, Attribute, Operation	Indicates theevent keyword that is used to signify events in C++, COM, and Managed Extension for C++.
CppW64	Boolean	Attribute	Indicates that thew64 keyword is being used for Microsoft Visual C++.
CppBased	String	Attribute	Tag definition that supports Based Addressing, which is used in Microsoft Visual C++.
CppCdecl	Boolean	Operation	Indicates thatcdecl keyword, which is one of the function calling methods in Microsoft Visual C++, is being used.

L			1
CppStdcall	Boolean	Operation	Indicates thatstdcall keyword, which is one of the function calling methods in Microsoft Visual C++, is being used.
CppFastcall	Boolean	Operation	Indicates thatfastcall keyword, which is one of the function calling methods in Microsoft Visual C++, is being used.
CppInline	Boolean	Operation	Indicates the inline function that uses the inline keyword in Microsoft Visual C++.
CppForceInline	Boolean	Operation	Indicates the inline function that uses the forceinline keyword in Microsoft Visual C++.
CppProperty	Boolean	Operation	Indicates theproperty defined in Microsoft's Managed Extension for C++.

DataTypes

C++ profile contains definitions for the following basic C++ data types.

- int
- unsigned int
- bool

- char
- unsigned char
- short
- unsigned short
- long
- long long
- unsigned long
- void
- float
- double
- long double
- wchar_t
- __int8
- __int16
- _____int32
- _____int64
- _____m64

C++ Framework

Top Previous Next

C++ Add-In provides MFC 6.0 Framework, which is a model conversion of the MFC class library included in Microsoft Visual C++ 6.0. When modeling software that uses MFC in the C++ language platform, using MFC 6.0 Framework greatly increases productivity and convenience as the classes defined in MFC can be used directly, without having to create them from scratch.

• Using MFC 6.0 Framework

Using MFC 6.0 Framework

Procedure for Importing MFC 6.0 Framework:

- 1. Select the **[File] -> [Import] -> [Framework...]** menu.
- 2. At the **[Import Framework]** dialog, select MFC 6.0 Framework from the list and click the **[OK]** button.

Import Framev				X 000
.net	<u>&</u>	<u>&</u>	MFC	
.NET Base Class Librar	Java 2 Enterprise 1.4		Microsoft Foundation Classes 6.0	
Description: Microsoft Four	ndation Classes (N	1FC) 6.0 Framew	vork	
		ОК	Cancel	lelp

The Select Element dialog box will appear, to determine in which element 3. MFC 6.0 Framework will be located. Select an element (package, model, subsystem or project) to contain the framework and then click the **[OK]** button.

Select a package where the framework will be imported to	×
Untitled Scenarios Logical View Development View Process View Physical View	
::Logical View	
OK Cancel <u>H</u> el	P

4. The framework is included in the selected element.

Note

Importing a framework does not store the framework elements in the

- project. Since framework units are referenced by the project, the framework unit files must be present when opening the project.
- To delete the imported frameworks, remove the respective framework units.

Option Configurations

Top Previous Next

This chapter discusses procedures for configuring the C++ Add-In environment and describes the option items in detail.

- <u>Code Generation Option Configuration</u>
- <u>Reverse Engineering Option Configuration</u>

Code Generation Option Configuration Top Previous Next

Code Generation Option (**[Tools]** -> **[Options...]**) is the group of option items for code generation by C++ Add-In. This category includes the **[General]**, **[Code Generation]**, **[Editing]**, and **[File Header]** sub-categories.

Option		×			
Option category	Option <u>i</u> tem				
Environment	General				
C++	Generate codes even when the				
Code Generation	🗆 File Generation				
Reverse Engineering	Header file extension name	.h			
C#	Implementation file extension n	,cpp			
🛄 Java	Header file subdirectory				
	Implementation file subdirector				
	Filename format	Same as Model			
	Generate namespace as subdire				
	Code Generation				
	Generate package as namespa				
	Use Microsoft Visual C++ gram				
	J				
Description: Code Generation					
Contains options for generating C++	code from modeling elements.				
Reset to default values Revert to	the last value	OK Cancel			

General

Option Item	Default	Description
Generate codes		Specifies whether to generate codes even when
even when there is	False	C++ profile is not loaded in the project (not
no profile		recommended).

File Generation

Option Item	Default	Description
Header file extension	.h	Specifies the extension name for C++ header
name	•11	files (e.gh).
Implementation file		Specifies the extension name for C++
extension name	.cpp	implementation files (e.gcpp).
		Specifies the subdirectory for saving the C++

Header file subdirectory		header file generated. If unspecified, the file is saved in the current directory.
Implementation file subdirectory		Specifies the subdirectory for saving the C++ implementation file generated. If unspecified, the file is saved in the current directory.
Filename format	Same as the model name	Specifies the format for the C++ source file name.
Generate namespace as subdirectory	False	Generates namespace as subdirectory.

Code Generation

Option Item	Default	Description
Generate package as namespace	False	Generates package elements as namespaces.
Use Microsoft Visual C++ grammar	raise	Generates code using Microsoft Visual C++ grammar.
.NET Managed C++ support	Faise	Generates code using .NET Managed C++ grammar.
Member order	Public member first	Determines the code generation order for the members defined in classes and interfaces.

Code Style

Option Item	Default	Description
Insert tabs as spaces	False	Uses space instead of tab for indentation.
Tab width	4	Specifies the number of spaces to be used when inserting tabs as spaces.
Place the opening curly brace in the new line	False	Places the opening curly brace "{" in the new line.

File Header

Option Item	Default	Description
		Adds the comments in the beginning of the

		source file.
		(Default)
File Header Comments	See description	// // Generated by StarUML(tm) C++ Add-In // // @@ Project : @p // @@ File Name : @f // @@ Date : @d // @@ Author : @a //
Default Include text for header file		Contains the #include text that is included in all header files generated.
Default Include text for implementation file		Contains the #include text that is included in all implementation files generated.

Reverse Engineering Option Configuration

Reverse Engineering Option Configuration (**[Tools] -> [Options...]**) is the group of reverse engineering option items for C++ Add-In. This category includes the **[Model Generation]**, **[Diagram]** and **[View]** sub-categories.

Option		×
Option <u>c</u> ategory	Option <u>i</u> tem	
Divironment	Model Generation	
C++	Generate public member	✓
Code Generation	Generate protected member	
Reverse Engineering	Generate private member	
C#	Omit initial value for data memb	
Java 🛄	Generate data member as Assc	
	🗆 Diagram	
	Generate Overview diagram	
	Diagram name	Overview of %s
	🗆 View	
	Suppress the Attribute compart	
	Suppress the Operation compar	
1	J	
Description: Reverse Engineering		
Contains options for reverse enginee	ering C++ code into model information.	
Reset to default values Revert t	o the last value	OK Cancel

Model Generation

Option Item	Default	Description
Generate public member	True	Specifies whether to generate class and interface members with public visibility.
Generate package member	True	Specifies whether to generate class and interface members with package visibility.
Generate private member	True	Specifies whether to generate class and interface members with private visibility.
Omit initial value for data members	False	Does not include the initial data member value in the Attribute model information.
Generate data		Analyzes the field information in the source code to establish association relationships with the

members as	False	respective data member type models. If
Association		unchecked, data member information is generated
		as attribute for the respective class model.

Diagram

Option Item	Default	Description
Generate Overview diagram	True	Specifies whether to generate the Overview diagram for the generated model. If unchecked, the following diagram and view options are ignored.
Diagram name		Specifies the Overview diagram name. The package name can be included in the diagram name by using %s (e.g. Overview of %s).

View

Option Item	Default	Description
Suppress the Attribute compartment of Class	False	Suppresses the Attribute compartment of the Class View when generating the Overview diagram.
Suppress the Operation compartment of Class	False	Suppresses the Operation compartment of the Class View when generating the Overview diagram.
Hide operation signature	False	Hides the operation signature when generating the Overview diagram.
Generate Generalization and Realization views only for relations	False	Generates generalization and realization views only for relations when generating the Overview diagram. When used appropriately with other view options, this option is very useful for drawing the inheritance relations of overall classes and interfaces within the package.

C++ Reverse Engineering

Top Previous Next

C++ reverse engineering analyzes C++ sourse files and converts them into UML models. This is useful for source inspection, system structure analysis and re-designing.

- <u>C++ Reverse Engineering</u>
- <u>Reverse Engineering Option Configuration</u>

Procedure for Reverse Engineering:

In StarUML(tm), select the **[Tools] -> [C++] -> [Code Reverse**

1.

Engineering...] menu.

At the **[Select Source Code]** page in the **[C++ Reverse Engineering]**

2. dialog box, select a source and click **[Add]**. Click **[Next]** once you have completed adding the target sources for reverse engineering.

++ Reverse Engineering		×
Select Source Code Select source code for reverse engineering.		
Directory:	C++ source file in the current directory:	
🚮 Desktop	Name	Size Type
🕀 🖄 My Documents	Application.h	1 KB H File
🕀 🖳 My Computer	Document.h	1 KB H File
My Network Places Recycle Bin Solution Internet Explorer	🗃 ImageProxy.h	1 KB H File
	•	Þ
Source file for reverse engineering:	Add Remove	Add Alj
File name Path		
	< Back Next >	Cancel

3. At the **[Select the Package to Contain Result]** page, select a package to contain the output results from the package tree and click **[Next]**.

C++ Reverse Engineering	×
Select the Package to Contain Result	
Select the package to contain result of reverse engineering.	
Project Structure:	
 Untitled Use Case Model Design Model Implementation Model Deployment Model 	
< <u>B</u> ack <u>N</u> ext > <u>C</u> ano	el

4. At the **[Option Setup]** page, select the reverse engineering options and click **[Run]**. Reverse engineering will start now.

C++ Reverse Engineering	×
Option Setup Configure options for reverse engineering.	
Model Generation Comit the initial value of data member Generate the following visibility only visibility only	Reference Data member Creation Create the Data member to the Attribute Create the Data member to the Association Diagram Create Overview diagram The name of a diagram : e.g. Overview of %s Overview of %s
	Hide the Operation signature Generate Generalization and Realization views only < <u>Back</u> <u>N</u> ext > <u>Cancel</u>

5. The **[Reverse Engineering]** page will show the reverse engineering progress status and return reverse engineering failure or success results.

ource Files:		(1-	4/:
File Name	Path	Status	L,
🔵 BarShader.h	E:₩Test₩CPPSrc₩eMule0.30d-Sources₩src	Complete	
🔵 BtnST.h	E:₩Test₩CPPSrc₩eMule0.30d-Sources₩src	Complete	
🔵 CatDialog.h	E:₩Test₩CPPSrc₩eMule0.30d-Sources₩src	Complete	
CBase64Coding.hpp	E:₩Test₩CPPSrc₩eMule0.30d-Sources₩src	Complete	
ChatSelector.h	E:₩Test₩CPPSrc₩eMule0.30d-Sources₩src	Complete	
ChatWnd.h	E:₩Test₩CPPSrc₩eMule0.30d-Sources₩src	Complete	
ClientCredits.h	E:₩Test₩CPPSrc₩eMule0.30d-Sources₩src	Complete	
ClientDetailDialog.h	E:₩Test₩CPPSrc₩eMule0.30d-Sources₩src	Complete	
C++ reverse engineering has Refer the occurred event to I	been completed successfully. nformation window.		

Note

If C++ reverse engineering is executed without including C++ profile, the

• following dialog box will appear asking whether you want to include C++ profile. Select "Yes(Y)" to continue the reverse engineering process.

Plastic	X
?	To C++ reverse engineering, C++ Profile is needed. Do you want to include C++ Profile to the current project?
	Yes No

Reverse Engineering Option Configuration

Reverse Engineering Option Configuration Screen

This is the screen for configuring the options required for C++ reverse engineering.

C++ Reverse Engineering	×
Option Setup Configure options for reverse engineering	
Model Generation Image: Omit the initial value of data member Generate the following visibility only Image: Omit the initial value of data member Generate the following visibility only Image: Omit the initial value of data member Image: Omit the initial value of da	Reference Data member Creation • Create the Data member to the Attribute • Create the Data member to the Association Diagram • Create Overview diagram The name of a diagram : e.g. Overview of %s Overview of %s
View Suppress the Attribute compartment Image: Compartment Suppress the Operation compartment Image: Compartment	

Model Generation

Model Generation includes various options for model generation.

Item	Description
Omit field initial value for data member	Does not include the initial data member value in the Attribute model information.
public	Specifies whether to generate class and interface members with public visibility.
protected	Specifies whether to generate class and interface members

	with protected visibility.
Drivate	Specifies whether to generate class and interface members
private	with private visibility.

Reference Data Member Generation

Reference Data Member Generation specifies generation methods for reference fields when generating models.

Item	Description
mompor to the	Specifies whether to generate C++ data members as attribute elements.
mompor to the	Specifies whether to generate C++ data members as association elements.

Diagram

Diagram specifies diagram generation and the default generation names.

Item	Description
Create Overview diagram	Specifies whether to create Overview diagram when generating model. If not selected, all options related to diagram and views are ignored.
The name of diagram	Specifies names for Overview diagram generation. The string %s is automatically replaced by the package name (e.g. Overview of %s).

View

View specifies view-related options after model generation.

Item	Description
Suppress the	
Attribute	Suppresses the attribute compartment of class models.
compartment Suppress the	
Operation	Suppresses the operation compartment of the class models.
compartment	
Hide operation	Specifies whether to display all signatures for operation

signatures	elements.
Generate	Specifies whether to generate generalization and realization
generalization and	views only for the models generated. This option is useful
realization views	if used with other view options to express inheritance
only	relations between classes and interfaces within the package.

C++ Source Code Generation

Top Previous Next

C++ code generation generates C++ source files from StarUML(tm) models.

- <u>C++ Code Generation</u>
- <u>Code Generation Option Configuration</u>

C++ Code Generation

Procedure for Code Generation :

- 1. In StarUML(tm), select the **[Tools]->[C++]->[Code Generation...]** menu.
- 2. At the **[Select Starting Package Location]** page in the **[C++ Code Generation]** dialog box, select a package and click **[Next]**.



3. At the **[Select the code generation element]** page, select the elements and click **[Next]**.

C++ Code Generation	x
Select the code generation element(s) Select elements to generate by C++ code.	
Code Generation Element:	
✓ UMLPackage ✓ UMLSubsystem ✓ UMLElementImport ✓ UMLProject	
Select <u>All</u> Deselect All	
< <u>Back</u> <u>N</u> ext > <u>Cancel</u>	

4. At the **[Output Directory Setup]** page, select a directory to save the output sources and click **[Next]**.

C++ Code Generation			×
Output Directory Setup Specify the directory to save generated codes.			
Output Directory: Posktop My Documents My Computer My Network Places Recycle Bin Internet Explorer			
	< <u>B</u> ack	<u>N</u> ext >	Cancel

5. At the **[Option Setup]** page, select options and click **[Run]**. Reverse engineering will start now.

C++ Code Generation

Option Setup Configure options for code generation.	
Implementation file:	eader file subdirectory
Code Style Insert tab as space Tab width: 4 Place opening curly brace "{" in the new line	Code Generation Generate Package to Namespace Use Microsoft Visual C++ grammar SupportNET Managed C++ Member in order Public member first
Code Generation Options File Header Comment and Def	ault Include

×

6. The **[Code Generation]** page will show the code generation progress status and return code generation failure or success results.

++ Code Generation			J
Code Generation Generate codes.			
Code generation <u>e</u> lements:			(5 / 5
Element	Location	Status	
UMLPackage	::Application Model::Modeling Elements::UML Mod	Complete	
UMLModel	::Application Model::Modeling Elements::UML Mod	Complete	
UMLSubsystem	::Application Model::Modeling Elements::UML Mod	Complete	
UMLElementImport	::Application Model::Modeling Elements::UML Mod	Complete	
++ code generated successfully.			
		Ei	nish

Note

The following error will occur if C++ code generation is executed without

• including C++ profile. Please ensure that C++ profile is included in the project before executing code generation.



Code Generation Option Configuration Top Previous Next

Code Generation Option configuration Screen

This is the screen for configuring the options required for code generation.

C++ Code Generation	×
Option Setup	
Configure options for code generation.	
Implementation file:	enerate Namespace to subdirectory
	· · ·
Code Style	Code Generation Generate Package to Namespace
 Insert tab as space Tab width: 4	Use Microsoft Visual C++ grammar Support .NET Managed C++
	Member in order Public member first
Code Generation Options File Header Comment and Def	ault Include
	< Back Next > Cancel

Generation file Setup

"Generation file setup" defines the file name and path for source file generation.

Item	Description
Header file	Specifies the extension name for C++ header file (e.gh).
Implementation	Specifies the extension name for C++ implementation file
file	(e.gcpp).
File name	Specifies the format of the C++ source file name. Four options are available: "same as the model name", "lower case", "upper case", and "lower case with '_' ".
Header file subdirectory	Specifies the subdirectory name for the C++ header file.
1	1

	Specifies the subdirectory name for the C++ implementation file. If unspecified, the current directory is used.
Generate Namespace as subdirectory	Specifies whether to generate namespace as subdirectory.

Code Style

Code Style defines the code style for the C++ code generated.

Item	Description
Insert tab as space	Specifies whether to insert space strings instead of tabs.
Tab width	Specifies the number of spaces for a tab. Effective only if the "Insert tab as space" option is selected.
Place opening curly brace "{" in the new line	Specifies the location of the opening curly brace "{" for code generation.

Code Generation

Code generation configures the general options for source code generation.

Item	Description
Generate package	Generates package element as C++ namespace. Classes and
as namespace	interfaces included in package are declared in namespace.
Use Microsoft	
Visual C++	Generates code using Microsoft's Visual C++ grammar.
grammar	
Support .NET Managed C++	Generates code using grammar defined in .NET's Managed
	C++. Effective only when the "Use Microsoft Visual C++
	grammar" option is selected.
Member in order	Specifies the code generation order for the members
	defined in classes and interfaces. Three options are
	available: "Public member first", "Private member first",
	and "Unordered".

C++ Code Generation	×
Option Setup Configure options for code generation.	
File Header Comment:	Description:
// /// Generated by Agora Plastic(tm) C++ Add-In /// @@ Project : @p /// @@ File Name : @f /// @@ Date : @d /// @@ Author : @a //	 @p : Title @d : Date @c : Company @a : Author @r : Copyright @f : File name @e : Element name @@ : Character@
Initial header #includes: Initial implementation #include	:s:
Code Generation Options File Header Comment and Default Include	
< <u>B</u> ack	lext > <u>C</u> ancel

File Header Comments and Default Include

"File Header Comments and Default Include" defines the comments for each file header and the common include clause for all files.

Item	Description
File Header	Contains the comments to be inserted in the beginning of the source file. As described in the "header comments description" section, the '@' symbol and alphanumeric characters can be used to insert specific values here.
Initial header #includes	Contains the common #include clause for all header files.
Initial implementation #includes	Contains the common #include clause for all implementation files.

FAQ

The following are frequently asked questions and answers for C++ Add-In.

- 1. <u>The "C++" menu cannot be found under the **[Tools]** menu.</u>
- 2. When reverse engineering with C++ Add-In the "Analysis failure" error occurs even through the source has no C++ grammar problems.
- 3. <u>Not all of the C++ reverse engineering results are displayed in the diagram.</u>
- 4. <u>The pointer indicators in Attribute type properties are omitted in the C++</u> <u>reverse engineered models.</u>
- 5. <u>The reverse engineered diagram is too complicated to view.</u>

1. The "C++" menu cannot be found under the [Tools] menu.

There are two possible reasons why the menu does not show up. One is that C++ Add-In is not installed in StarUML(tm) or related files are damaged. To correct this, run the installation program to reinstall StarUML(tm) or select to reinstall C++ Add-In only. If you are familiar with the structure of StarUML(tm) Add-Ins, you may examine the registry or menu files to check for damage in the Add-In and correct the problem. Another possible reason for the problem is that C++ Add-In is disabled by Add-In Manager. If so, go to Add-In Manager and enable C++ Add-In. For the procedure to enable C++ Add-In, see the "1.2 C++ Add-In Configurations" section.

2. When reverse engineering with C++ Add-In the "Analysis failure" error occurs even though the source has no C++ grammar problems.

Source file analysis failure in reverse engineering occurs mostly with the codes written with macros.

While general C++ compilers perform another pre-process cycle prior to compiling the source code, C++ Add-In does not perform a complete preprocess cycle. This is because replacing all the macros requires analysis of the source files that contain the declarations of the macros. This requires separate management of the source file paths for searching, takes a longer time analyzing the macro declaration source files, and some sources written in specific development environments such as Visual C++ cannot be analyzed properly unless the development environment is installed in the system. Including macro comments in the analyzed model may also result in a complicated model view, which may make it difficult for the user to analyze the model.

Consequently, C++ Add-In flexibly modifies parts of the grammar used in analysis so that the macros used in member declaration in class are identified as macros in the model. However, since it is difficult to identify macros in any location due to the nature of the grammar, macros in unexpected locations may not be identified correctly. In such rare cases, the "analysis failure" error occurs. Nevertheless, even through some parts of source file analysis may fail, the rest of the file is analyzed correctly, returning the proper reverse engineering results.

3. Not all of the C++ reverse engineering results are displayed in the diagram.

Not all of the reverse engineering results are displayed in the diagram because the current diagram is not large enough to contain all of the generated views. Select the **[Tools]->[Options...]** menu to open the Options dialog box, and at the **[Environment]->[Diagram]** section, adjust the "Default Diagram Height" and "Default Diagram Width" to make the diagram size larger.

4. The pointer indicators in Attribute type properties are omitted in the C++ reverse engineered models.

UML does not include model properties for the C++ pointer concept. Therefore, it is not technically correct to use pointer indicators in Attribute type properties. Nevertheless, C++ Add-In uses the CppPointer tag definition for expressing pointer type and reference type. Select the Attribute and hit Ctrl + F7 to verify the CppPointer tag definition value in the tag definition editor window.

5. The reverse engineered diagram is too complicated to view.

One of the main functions of reverse engineering is to convert an existing source code into models and facilitate easier analysis. However, if all of the reverse engineered model information is expressed in diagram, it would be difficult to view and analyze the information. In order to assist the user in this aspect, C++ Add-In provides various options for reverse engineering. The user can specify the diagram format by manipulating these options. For example, selecting "Suppress the Attribute compartment", "Suppress the Operation compartment", and "Generate Generalization and Realization views only" outputs only the class inheritance structure in diagram.