Starting Help

Since C# Add-In transforms UML model to C# source code or C# source code to UML model, you can achieve high productivity and quality improvement as adopting C# code generation and reverse engineering to the tool easily and quickly.

Index of C# Add-In Help

- 1. <u>C# Add-In Overview</u>
- 2. <u>C# Profile</u>
- 3. <u>.NET BCL 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, using, 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 the C# language in your software modeling.

.NET BCL Model Framework

C# Add-In provides the ECMA-334 Standard Library 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.

Note

• The scope of application for C# Add-In is C# codes based on ECMA-334 (C# Language Specification v1.2).

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 menu and 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).

At the Add-In Manager dialog box, check the "C# Add-In" checkbox in 2.

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	_
🗹 🗇 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).

At the Add-In Manager dialog box, uncheck the "C# Add-In" checkbox in 2. 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.

- Including C# Profile
- 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.

At the Profile Manager window, select "C# Profile" from the "Available 2. profiles" list on the left.

Available profiles:	Java Profile	Include > < Exclude	Included grofile:	s: UML Standard Profile	
Description: C# Language Profile				Close	Help

Click the **[Include]** button or hit Alt-I to move "C# Profile" to the 3.

"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, Tagged Value, 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 :

2.

1. Select the [Model] -> [Profile...] menu.

At the Profile Manager window, select "C# Profile" from the "Included

Profile			×
Available profiles:	Frofile EJB Profile	Include 2 profiles:	
Description:			
C# Language Profile		Close	Help

profiles" list on the right.

Click the [Exclude] button or hit Alt-E to remove "C# Profile" from the 3.

"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 tagged values of the model elements.

C# Profile Definition List

Stereotype

C# profile contains definitions for the following stereotypes.

Sterotype	Target Element	Description
< <csharpsourcefile>></csharpsourcefile>	Component	Source file with C# code
< <dotnetassembly>></dotnetassembly>	Component	File with compiled C# source code
< <csharpdelegate>></csharpdelegate>	Class	C# Delegate indicator
< <csharpstruct>></csharpstruct>	Class	C# Struct indicator
< <csharpevent>></csharpevent>	Operation	C# Event indicator
< <csharpproperty>></csharpproperty>	Operation	C# Property indicator
< <csharpoperator>></csharpoperator>	Operation	C# Operator indicator
< <csharpindexer>></csharpindexer>	Operation	C# Indexer indicator

TagDefinition

C# profile contains definitions for the following tag definitions.

TagDefinition	Туре	Target Element	Description
CSharpOverride	Boolean	Operation	An instance method declared by using an override modifier is called "redefinition method". A redefinition method redefines an inherited virtual method that has the same signature.
CSharpVirtual	Boolean	Operation	If a declaration contains a virtual limiter, the method is called "virtual method". If there is no virtual modifier,

			the method is called non-virtual method.
CSharpExplicit	Boolean	Operation	The explicit keyword is used for declaring explicit user definition format conversion operators.
CSharpExtern	Boolean	Operation	Using the extern limiter in the method declaration indicates that the method is manipulated externally.
CSharpSetter	Boolean	Operation	Property accessors
CSharpGetter		Operation	include execution lines related to importing properties (reading or calculating) and configuring properties (writing). An access declaration may include the get accessor, the set accessor, or both.
CSharpAdd	Boolean	Operation	These are
CSharpRemove	Boolean	Operation	declarations of accessors that are used for adding or removing event processors in the client code. Accessor functions are add and remove .

CSharpConstructorInitializer	String	Operation	This defines other instance generators that will be called before executing the lines in the instance generator.
CSharpImplicit	Boolean	Operation	The implicit keyword is used for declaring implicit user definition format conversion operators
CSharpUnsafe	Boolean	Class, Operation, Attribute, AssociationEnd	The unsafe keyword indicates unsafe contexts.
CSharpNew	Boolean	Class, Operation	The new modifier is used for explicitly hiding the members inherited from the basic classes.
CSharpStatic	Boolean	Class, Operation, Attribute	The static modifier is used for declaring not specific objects but static members that belong to specific formats.
CSharpConst	Boolean	Attribute	The const keyword is used for editing declarations of data members or local variables.
CSharpVolatile	Boolean	Attribute, AssociationEnd	The volatile keyword indicates that the data member can be edited in the program by threads run by the operating

			system, hardware, or both at the same time.
CSharpInternal	Boolean	Class, Operation, Enumeration	The internal keyword is an access modifier for formats and format members. Internal members can be accessed within the same assembly file only.
CSharpBasetype	String	Enumeration	Each enumeration format includes an integer type format called internal format of the enumeration format. This internal format must be able to express all enumeration values defined in the enumeration.
CShapEnumLiteralValue	String	EnumerationLiteral	Each enumeration member contains a constant value. The format of this value is the internal format of the enumeration.
CSharpDimension	Integer	Parameter, Attribute, AssociationEnd	This is the array dimension of the object declared.

		Parameter arrays are parameters declared by using the params
CSharpParameter	Boolean Parameter	modifier. If the parameter array is included in the format parameter
		list, it must be located at the end of the parameter list and it
		has to be in 1-dimension array format.

DataTypes

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

- bool
- decimal
- sbyte
- byte
- short
- ushort
- int
- uint

- long
- ulong
- char
- float
- double
- object
- string
- void

.NET BCL Framework

The .NET BCL (Base Class Libraries) Framework is the C# development platform model included in C# 2 Standard Edition. In StarUML(tm), a Model Framework refers to a software model that expresses a Class Library or application frameworks such as MFC, VCL, and JFC. Using a framework with the project greatly increases the convenience of the user for modeling software that depends on a certain class library or application framework.

• <u>Using the .NET BCL Framework</u>

Using the .NET BCL Framework

Procedure for Importing the .NET BCL Framework :

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

Import Framew	vork				×
Erameworks list:				000	8-8- 8-8- 8-8-
.net	<u></u>	<u>&</u>			
,NET Base Class Libraries 1.0	Java 2 Enterprise 1.4	Java 2 Standard 1.3			
Description: .NET Base Clas	s Libraries(BCL)	1.0 Framework		 	
		ОК	Cancel	Help	

The Select Element dialog box will appear to determine in which element

3. the .NET BCL 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 Use Case Model Design Model Implementation Model Deployment Model	
::Design Model OK Cancel He	elp

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]**, **[Code Style]**, and **[File Header]** sub-categories.

Option		×			
Option <u>c</u> ategory	Option <u>i</u> tem				
invironment	General				
C++	Generate codes even when the				
C#	Code Generation				
Code Generation	Generate 1 file each for Class,	✓			
Reverse Engineering	Generate unnamed Association				
🛄 Java	Generate C# Doc				
	Generate empty C# Doc				
	🗆 Code Style				
	Insert tabs as spaces				
	Tab width	4			
	Place the opening curly brace in				
	🗆 File Header	_			
J	J [
Description: Code Generation					
Contains options for generating C# co	ode 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 C#
even when there is	False	profile is not loaded in the project (not
no profile		recommended).

Code Generation

Option Item	Default	Description
Generate 1 file each for Class, Struct, Interface, and Enum		Specifies whether to generate 1 file each for 1 member (Class, Struct, Interface, and Enum).
Generate unnamed		Generates unnamed AssociationEnd as a reference type field for C# member. In this case,

AssociationEnd	the field identifier is indicated as "UnspecifiedType" with a number at the end.	
Generate C# Doc	False	Specifies whether to generate modeling element documentation as C# Doc.
Generate empty C# Doc	False	If checked, C# Doc is generated even when the modeling element documentation does not have any values (empty).

Code Style

Option Item	Default	Description
Insert tabs as	False	Uses space instead of tab for indentation.
spaces		Specifies the number of appear to be used when
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
File Header Comments		Adds the comments in the beginning of the source file. (Default) // // Generated by StarUML(tm) C# Add-In // // @@ Project : @p // @@ File Name : @f // @@ Date : @d

	//@@ Author : @a
	//
	//

Reverse Engineering Option Configurations

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 category	Option item			
Environment	Model Generation		•	
🗀 C++	Generate public member	×		
C#	Generate internal member	✓		
Code Generation	Generate protected member	✓		
Reverse Engineering	Generate private member	✓		
🛄 Java	Omit initial value for fields			
	Generate C# Doc as document.			
	Generate fields as Association			
	🗆 Diagram		_	
	Generate Overview diagram	✓		
	Fit the diagram area to the gen			
	Diagram name	Overview of %s	-	
Description: Reverse Engineering				
Contains options for reverse engineering C# code into model information.				
Reset to default values Revert to	the last value	OK Cancel		

Model Generation

Default	Description
	Specifies whether to generate class and interface
	members with public visibility.
Iriie	Specifies whether to generate class and interface
	members with internal visibility.
Truo	Specifies whether to generate class and interface
IIue	members with protected visibility.
Tranc	Specifies whether to generate class and interface
	members with private visibility.
Falco	Does not include the field initial value in the
гаізе	Attribute model information.
	True True True

Generate C# Doc as documentation	False	it as documentation information. C# Doc of each method is analyzed for tag information and entered as documentation information for each parameter of the operation model.
Generate fields as Association	False	Analyzes the field information in the source code to establish association relationships with the respective field type models. If unchecked, field information is generated as attribute for the respective class model.

Diagram

Option Item	Default	Description	
Generate Overview diagram		Specifies whether to generate the Overview diagram for the generated model. If unchecked, the following diagram and view options are ignored.	
Fit the diagram area to the generated view area	False Enlarges the Overview diagram to fit the generated view area.		
Diagram name	Overview of %s	Specifies the Overview diagram name. The w 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 area 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

This chapter discusses the procedures for using C# reverse engineering and the concepts of reverse engineering.

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

Procedure for Reverse Engineering :

In StarUML(tm), select the **[Tools]** -> **[C#]** -> **[Reverse Engineering...]** 1.

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.

C# Reverse Engineering				X
Select Source Code Select source code for reverse	engineering.			
Directory:		C# file in the current di	rectory:	
🗹 Desktop		Name		Size Type
		Application.cs		1 KB CS File
∰		Document.cs		1 KB CS File
Recycle Bin		ImageProxy.cs		1 KB CS File
Source file for reverse engineering:		Add	<u>R</u> emove	Add Alj
File name	Path			
		< <u>B</u> ack	Next >	⊆ancel

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	
Design Model	
Implementation Model	
1	
< <u>B</u> ack <u>N</u> ext >	Cancel

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

Reverse	Enginee	1000
REVEISE		
		_

Model Generation Generate C# doc to documentation Omit the initial value of the field Generate the following visibility only public protected private	Reference Field Creation Generate the field to the Attribute Generate the field 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 Suppress the Operation compartment Suppress th	Hide the Operation signature Generate Generalization and Realization views only < Back Run(R) Cancel

X

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

urce Files:		(8
ile Name	Path	Status
BookBean.cs	C:₩CS₩BookApplication	Complete
BookDAO.cs	C:₩CS₩BookApplication	Complete
BookMgr.cs	C:₩CS₩BookApplication	Complete
IBook.cs	C:₩CS₩BookApplication	Complete
CategoryBean.cs	C:₩CS₩CategoryApplication	Complete
CategoryDAO.cs	C: WCSWCategoryApplication	Complete
CategoryMgr.cs	C:₩CS₩CategoryApplication	Complete
ICategory.cs	C:₩CS₩CategoryApplication	Complete
# reverse engineering has Refer the occurred event to		

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?
	<u>Y</u> es <u>N</u> o

Reverse Engineering Option Configuration

Reverse Engineering Option Setup 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 Generate C# doc to documentation Omit the initial value of the field Generate the following visibility only Generate the following visibility only for public for internal for protected for private	Reference Field Creation Generate the field to the Attribute Generate the field 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 Suppress the Operation compartment	Hide the Operation signature Generate Generalization and Realization views only < <u>Back</u> Run(<u>R</u>) <u>Cancel</u>

Model Generation

Model Generation includes various options for model generation.

Item	Description
doc to model	Specifies whether to generate C# Document as StarUML(tm) model documentation.
Omit field initial values	Specifies whether to omit the initial values for C# fields.
public	Specifies whether to generate elements with public access

	modifiers.
package	Specifies whether to generate elements with package access modifiers.
protected	Specifies whether to generate elements with protected access modifiers.
private	Specifies whether to generate elements with private access modifiers.

Reference Field Creation

Reference Field Creation specifies generation methods for reference fields when generating models.

Item	Description
Generate the field to the Attributes	Specifies whether to generate C# fields as StarUML(tm) model attributes.
Generate the field to the Associations	Specifies whether to generate C# fields as StarUML(tm) model associations.

Diagram

Diagram specifies diagram generation and the default generation names.

Item	Description
Create Overview diagram	Specifies whether to generate Overview diagram when generating model.
The name of Diagram	Specifies names for Overview diagram generation. The string %s is automatically replaced by the package name.

View

View specifies view-related options after model generation.

Item	Description
Suppress the Attribute	Suppresses the attribute area of class models.
compartment	
Suppress the	

Operation compartment	Suppresses the operation area of the class models.
Hide operation signature	Specifies whether to display all signatures for operation elements.
Generate generalization and realization views only	Specifies whether to generate generalization and realization views only for the models generated.

C# Code Generation

Top Previous Next

This chapter discusses the procedures for using C# code generation and the concepts of forward engineering.

- <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#] -> [Generate code...]** menu.
- 2. At the **[Select Package Starting Location]** page in the **[Generate code...]** dialog box, select a package and click **[Next]**.

C# Code Generation		×
Select Starting Package Location		
Select the starting package for C# code generation.		
Select the <u>P</u> ackage:		
🗆 🔁 Application Model		*
🗆 💼 Modeling Elements		
🖃 🚞 UML Model Elements		
Model_Management		
🖻 💼 Foundation		
Data_Types		
Core		
표 🛅 Behavioral_Elements		
- 🔁 UML View Elements		
ViewCore Elements		
Core Elements		
Application Objects		
		-
Extension Elements		
	Next >	ancel

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

_	10 m 10 m	
IC#	Code	Generation

C# Code Generation			×
Select the code generation element(s) Select elements to generate by C# code.			
Code Generation <u>E</u> lement:			
🕑 🔽 Data_Types			*
E 🗹 🛅 Core			
UMLGeneralizableElement			
UMLNamespace			
🗹 📃 UMLAssociationEnd			
🖂 🗹 📃 UMLInterface			
🗹 📃 UMLRelationship			
UMLAssociation			-
Select <u>All</u> Deselect All			
	< <u>B</u> ack	<u>N</u> ext >	Cancel

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

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

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

C# Code Generation	×
Option Setup Configure options for code generation.	
Generation Options Generate one file for each element Generate unnamed AssociationEnd Generate documentation as C# Doc Generate empty C# Doc	Code Style Insert tab as space Tab width: 4 Place opening curly brace "{" in the new line
File Header Comment: // // // Generated by Agora Plastic(tm) C# Add-In // // @@ Project : @p // @@ Prile Name : @f // @@ Date : @d // @@ Author : @a	Description: @p : Title @d : Date @d : Date @c : Company @a : Author @r : Copyright @f : File name @e : Element name @@ : Character@
	< <u>B</u> ack <u>N</u> ext > <u>C</u> ancel

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

Location	Status	(54 /
		-
	•	
	•	
Application Model: Modeling Elements (UML Mod	Complete	
	::Application Model::Modeling Elements::UML Mod ::Application Model::Modeling Elements::UML Mod	::Application Model::Modeling Elements::UML Mod Complete ::Application Model::Modeling Elements::UML Mod Complete

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 Setup Screen

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

C# Code Generation	×
Option Setup Configure options for code generation.	
Generation Options Generate one file for each element Generate unnamed AssociationEnd Generate documentation as C# Doc Generate empty C# Doc	Code Style Insert tab as space Tab width: 4 Place opening curly brace "{" in the new line
File Header Comment: // // // Generated by Agora Plastic(tm) C# Add-In // // @@ Project : @p // @@ Project : @p // @@ Date : @d // @@ Author : @a	Description:
	< Back Next > Cancel

Generation Options

Generation options are model-related options for code generation.

Item	Description
Generate one file for each element	Specifies whether to generate one file for each element.
	Specifies whether to generate code for unnamed AssociationEnds.
Generate documentation as C# Doc	Specifies whether to generate StarUML(tm) model documentation as C# Doc.

Generate empty	Specifies whether to generate empty StarUML(tm)
C# Doc	documentation.

Code Style Options

Code Style options are text-related options for code generation.

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

File Header Comments

File Header Comments defines the comments for each file head

Item	Description
	Contains the comments to be inserted in the beginning of the
File header	source file. As described in the "header comments
comments	description" section, the '@' symbol and alphanumeric
	characters can be used to insert specific values here.

FAQ

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

- 1. <u>What are the C# Language versions supported by C# Add-In?</u>
- 2. <u>I get a "C# Profile is not loaded" warning window when generating code.</u> <u>What does this mean?</u>

1. What are the C# Language versions supported by C# Add-In?

C# Add-In supports C# Language Specification version 1.2, which is the most widely used version. C# Add-In also supports the ECMA-334 standard.

2. I get a "C# Profile is not loaded" warning window when generating code. What does this mean?

C# Profile must be included in order to use the C# code generation function. Including <u>C# Profile</u>