

Author	Clemens Maas	RXF Migration Guide How to migrate to a V5 RXF
Document Version	1.2	
Document Status	Released	

Change History

Version	Date	Author	Description
1.2	April 2008	C.Maas	Added Additional Sources and Libraries
1.1	April 2008	W. van der Heiden	Added Report on Model
1.0	April 2008	C. Maas	First version

Table of Contents

1	INTRODUCTION	4
2	MIGRATION	5
2.1	Preparations	5
2.2	Compiler options	6
2.3	Startfile	7
2.4	Linker options	7
2.5	Rhapsody	8
2.6	Additional Sources and Libraries	10
3	DEPLOYER	11
	REFERENCES	12

1 Introduction

The purpose of this document is to help you in migrating an existing Rhapsody project based on a V3 or V4 product by Willert Software Tools to the new V5 Framework.

The new V5 product has been setup in such way, that it allows you to install and use multiple versions next to each other in a safe manner, including a new V5 next to a V4 or V3 product.

In V5, the Embedded UML Studio relies on an external build environment, namely the IDE which comes with your toolchain, where all generated files will be deployed or copied to. In earlier versions, the build process was carried out inside Rhapsody via a generated Makefile; in your Rhapsody model you therefore had to specify compiler options, startup file and linker details. This was all done via properties, and in V5 most of these details are configured entirely in your IDE.

The advantage of this approach is that all tool dependant options can be used directly. Most of the time the user is already using the IDE to program his application and is comfortable with using all these options.

Most IDE's deliver special dialogs to setup CPU specifics; they can be used then without having to go to difficulties to integrate the results in Rhapsody.

When support questions concerning the result of the compiler come up, most compiler support departments demand a working IDE project before they supply support. With the new development procedure this is automatically generated.

As a result, you must carefully migrate an existing project to the new environment. Please use the examples in [GettingStarted.pdf](#) [1] first so you get a feeling about how the new V5 product works. You can safely install it on your PC without breaking existing projects or Rhapsody, your previous version and projects will be preserved.

2 Migration

2.1 Preparations

Of course the WST V5 product must be installed on your computer.

Although it may seem tempting, do not combine this migration with other migrations like installing a new Rhapsody version or a new compiler version.

A good preparation for a conversion is to make a print of all overridden properties. In Rhapsody, open your model, then select *Tools | Report on Model*. In the dialog box deselect all options except *Overridden Properties*. Then click *OK*. This will generate a report of all overridden properties in your model in the file `RhapsodyRep.rtf` in the directory of your Rhapsody project.

Use *Code | Re Generate* and *Code | Rebuild* for your model in the old environment to make sure that everything works. This will prevent the search for environment errors that are not real environment errors.

When this is done:

- create a new project in your IDE that will be used to compile and debug your existing Rhapsody application. If your current environment already supports an IDE, you may use that IDE file to create the new project. Keep this project open.
- create a backup of your Rhapsody project or add a Tag in your Configuration Management. Open your existing UML model and create a new component there where you will make the migration changes.

2.2 Compiler options

The compiler options must be set in your IDE. Very few compiler options are set in your UML model; these merely deal with differences between *Debug* and *Release* buildset.

Compiler options set in your UML model must have the form `-Dxxxxx`, because they are translated into `#define`'s in a generated include file during the Rhapsody build process. This prevents you from defining them again in your IDE and this approach is also toolchain independent.

The compiler options you do need to set in your IDE are related to code generation (for example memory model, if any) and include paths.

Because you most probably will be using the RXF as a library, it is important that you use the very same compiler options in your IDE project for building the RXF libraries and your applications.

To migrate an existing UML model with respect to compiler options to the new version, please

- select the *Features* of the component(s) in your model and open the *Stereotype* combobox to enable `RXFComponent` and optionally `StaticComponent` in the list of stereotypes used in your component(s).
- select your configurations like *Debug* and *Release* to verify the compiler options. Open the *Features* dialog, select the *Properties* tab and use the combobox to *View All* properties, and scroll down to `C.CG::<your product>` and open those to see the properties set for your product.
- carefully inspect the contents of *CppCompileSwitches*. If you have overridden compiler options in your model, you have to take all options and set them in the IDE project. Then unoverride the property in your model. The resulting options must look like:

```
-DWST_RXF_V5 -DWST_RIC_VERSION=$(RICVERSION) -DWST_PORTS_DISABLED
-DOM_NO_RCS_ID -D_16BIT $(OMCPPCompileCommandSet)
```

Any non `-Dxxxxx` option will disturb the compilation process. Options other than shown above must be moved to your IDE projects: your application project and the CreateRXFLibrary project!

The constant `WST_PORTS_DISABLED` is used to disable ports: if you need them, be aware of any restrictions, see the online Help [2], section *Introduction | Restrictions*.

- select the *Properties* tab and use the combobox to *View All* properties, and scroll down to `C.CG::<your V3/V4 product>` and open those to see the properties set for the old product.
- determine which compiler options must be moved into your IDE projects.

2.3 Startfile

The start file must be specified in your IDE.

To migrate an existing UML model with respect to specified start file to the new version, please

- select the *Properties* tab and use the combobox to *View All* properties, and scroll down to `C.CG::<your V3/V4 product>` and open those to see the properties set for the old product.
- determine which start file was specified, if any. If nothing was specified, the start file was taken from the directory `<your Rhapsody>\Share\MakeTmpl`
- specify the start file in your IDE project where you build your application.

2.4 Linker options

To migrate an existing UML model with respect to specified linker options to the new version, please

- select the *Properties* tab and use the combobox to *View All* properties, and scroll down to `C.CG::<your V3/V4 product>` and open those to see the properties set for the old product.
- inspect the properties `LinkSwitches`, `LinkDebug` and `LinkRelease` etc (depending on memory model or available buildsets). Specify the linker options in your IDE project where you build your application.
- sometimes a linker configuration file was taken from the directory `<your Rhapsody>\Share\MakeTmpl` - you can copy it to your IDE project and add it there.

2.5 Rhapsody

2.5.1 Properties

In the V5 products, there are only a few Willert Software Tools specific properties. These are related to the RXF being used: based on the OO RTX or based on an RTOS. Next to that, we have added a profile to set multiple properties in a single sweep, so you no longer need to verify or set as many properties as in V3 or V4 products.

At large there are four kinds of properties you might set yourselves:

- `CPPCompileSwitches` in relation to the *Debug* or *Release* `BuildCommandSet`.

For the *Debug* buildset, the options `-D_DEBUG` `-DHIGHWATERMARKS` are added to `CPPCompileSwitches`, while for a *Release* buildset the option `-DNDEBUG` is added. The compiler options are translated into `#define`'s in the file `RxfConstants.h` which is generated when you select *Code | Build* in Rhapsody.

- properties for your application which deal with sizes of queues etc.

These are specific for the RXF being used, and their values are translated into `#define`'s in the file `RxfDimensions.h` which is generated when you select *Code | Build* in Rhapsody. The properties are described in the online Help [2], section *Usage | High water marks*.

If your RXF is based on an RTOS, the RTOS configuration now is according to the documentation of your RTOS provider: we no longer use dedicated properties. Please refer to the online Help [2], section *Usage | RTOS Resource Management* and section *Bridges | RTOS* for further details.

- properties which deal with the dynamic or static memory allocation behaviour of your application. These properties can be set via the stereotype `StaticComponent` which is part of the profile which comes with your product.
- properties which deal with Memory Section Management, which is implemented via toolchain independent macros. For Rhapsody generated code, we typically use the properties `SpecificationProlog`, `SpecificationEpilog`, `ImplementationProlog` and `ImplementationEpilog`. These properties are set via the stereotype `RXFComponent` which is part of the profile which comes with your product.

If you are using these Prolog- and Epilog properties already for some reason, you must either merge these or remove them from the profile which comes with your product.

To migrate an existing UML model with respect to properties to the new version, please

- verify existing properties - do this at a Rhapsody GUI level using the *Features* dialog.
- verify the default values of the properties for example in the *GettingStarted* example, and determine which compiler options you must set in your IDE.

2.5.2 Profiles

In V5 products we have added a profile. A profile is the best way to override properties which are not restricted to an environment: if you set a property X in a property file whereas X is not defined inside an environment, the setting of property X will be used in any model you open in Rhapsody as long as Rhapsody has read that property file. Obviously, this is an undesired effect and therefore we introduced profiles in V5. Profiles also let you easily configure a UML model; instead of tweaking property per property, you use or do not use stereotypes which come with a profile.

Each product now comes with a profile which:

- sets the Environment: the name of composer plus version number.
This allows you to install and use multiple versions next to each other in a safe manner.
- defines the stereotype `RXFComponent`.
The stereotype `RXFComponent` should be selected for all components generating code for the Realtime eXecution Framework. Its tags are used by the Deployer.
- defines the stereotype `StaticComponent`.
The stereotype `StaticComponent` can be used to switch to static behaviour regarding memory allocation. The standard C library functions like `malloc()` and `free()` are mapped onto memory functions in the Framework and the macro `NO_MALLOC` is defined to prevent use of the heap.

Please refer to the online Help [2], section *Usage | Profiles* for further information.

To migrate an existing UML model with respect to profiles to the new version, please

- verify that no properties exist in your model which were set in a V3 or V4 version which are not related to the `WSTxx` or `WSTxxlt` environment.
These are: `C_CG::Framework::HeaderFile (oxf/WST.h)` and `CG::Event::BaseNumberOfInstances`.
- add the profile which comes with your product to your UML model as a reference
- select the stereotypes `RXFComponent` and `StaticComponent` in that profile and use the *Features* dialog to see which properties are set, and verify if a conflict would arise.
- select the *Features* of the component(s) in your model and open the *Stereotype* combobox to enable `RXFComponent` and optionally `StaticComponent` in the list of stereotypes used in your component(s).

2.6 Additional Sources and Libraries

Rhapsody allows you to specify additional sources and libraries as well as include files. The include files are needed to compile the generated sources, so specifying these remains unchanged - but you must specify the proper Include path option in your IDE.

The *Additional Sources* and *Libraries* must be specified in your IDE instead of in Rhapsody:



Because the build process is no longer within Rhapsody, your IDE project must compile the additional sources and libraries. This has an advantage too: you can use different sources and/or libraries per buildset *Debug* and *Release*, which was not possible in earlier versions.

3 Deployer

The Embedded UML Studio relies on an external build environment where all generated files will be deployed or copied to. The Deployer will not only copy generated files to an EW project, but also insert the names of all files in the IDE project file.

We have defined the Bridge for an IDE in such a way, that the model does not contain the location of your IDE project on your PC but the Deployer will maintain that information instead. This enables one to have the same project used by different persons at different locations and the installation location of the toolchain on various PCs can differ.

For the Deployer to work correctly, one basically needs the name of the IDE project file plus the relative paths inside an IDE project to be maintained in Rhapsody. The absolute paths are maintained by the Deployer on the PC. In the UML model we use tags to store the relative paths and the name of the IDE project, so what you need is:

- the name of an IDE project and relative paths inside that project are specified in tags
- the IDE project must exist and have the proper compiler options, startup file and linker settings.

To migrate an existing UML model with respect to the Deployer to the new version, please

- create an IDE project for your application. Make sure you are using exact the same compiler options and include paths as in the project `CreateRXFLibrary` which created the RXF libraries.
- select the *Features* of the component(s) in your model and open the *Stereotype* combobox to verify that `RXFComponent` is enabled.
- in the *Features* dialog, select the tab *Tags*.

When no tags are specified for a component, tags are not visible in the model browser. You must select the *Features* of the component to specify them. After that, they show up.

Please specify the following:

- for the *toolchainProjectFile* tag, insert the exact name of your IDE project (not the workspace it may belong to: we must add file names to the project file).
- for the *rootTitle* tag, you can use the name of the IDE project
- for the *rootDescription* tag, insert text like "Path to the <your project> IDE project"
- for the tags *relativeGeneratedFilePath* and *relativeRxfPath* you can use a dot (.) to start with; these are paths inside the IDE project. Using a dot results in all files in a single directory.

References

- [1] RXFGettingStarted.pdf – RXF Getting Started.
- [2] Rhapsody\Share\WST_RXF_V5\\Doc\Help\RXF.chm or
Rhapsody\Share\WST_RXF_V5\\Doc\Help\index.htm
Online documentation for your product.