

XMI Toolkit for Rhapsody®

Mapping Rules Overview



Notices

© Copyright IBM Corporation 1997, 2009.

US Government Users Restricted Rights—Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send written license inquiries to:

IBM Director of Licensing IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send written inquiries to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome
Minato-ku Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions. Therefore, this statement may not apply to you. ii This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for Rational Software
IBM Corporation
1 Rogers Street
Cambridge, Massachusetts 02142 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 1997, 2009.

IBM, the IBM logo, ibm.com, Rhapsody, and Statemate are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.html.

1. INTRODUCTION

This document outlines the rules governing the mapping of Rhapsody models to UML Models using XMI 1.3 and XMI 2.1. It is mainly targeted for users working in the field of model driven engineering.

2. CONVENTIONS

In UML 2.1, tagged values do not exist anymore. They are replaced by a combination of Stereotypes and Properties applicable to stereotyped elements.

In this document, we use the following notations:

MetaType <<Stereotype>>	Denotes an instance of the metatype “MetaType” to which we apply the stereotype “Stereotype”
(<<Stereotype>>) Metatype name[=value]	For XMI 2.1, indicates that the property with the name “name”, derived from the application of the stereotype “Stereotype” on the current element is created, with an optional value of “value”.
<Metatype>*	Indicates a of elements of the metatype
<Metatype1 Metatype2>	Indicates an alternative between metatype1 and metatype2

All stereotypes used for representing Rhapsody specific modeling concepts are grouped in a profile named **“RhapsodyProfile”**

All Stereotypes used for representing Rhapsody properties are in the profile “RhpProperties”

3. RULES

The rules are grouped by diagram type so that users using a subset of the diagram types available in UML can concentrate on the rules they are more specifically interested in.

3.1. Object Model Diagram

Rhapsody Model Element	UML 1.3	UML 2.1
IRPPProject	Model	Model
IRPPackage	Package	Package + <<RhpTopLevelClass>> Classifier TopLevelClass ¹
IRPClass 1. <IRPDDiagram> behavioralDiagrams 2. <IRPClass>* nestedClassifiers 3. <IPRType>* types 4. <IRPGeneralization>* generalizations	Class 1. <Behavior> 2. <ModelElement>* ownedElement (subset) 3. <ModelElement>* ownedElement (subset) 4. <Generalization>* generalization	Class 1. <Behavior> classifierBehavior 2. <Classifier>* nestedClassifiers 3. <Classifier>* nestedClassifiers 4. <Generalization InterfaceRealization>* generalization interfaceRealization
IRPActor	Actor	Class <<RhpActor>> ²
Interface (IRPClass<<interface>>)	Interface	Interface
IRPModelElement 1. description 2. <Tags>* 3. <Properties>*	ModelElement 1. TaggedValue description 2. <TaggedValue>* 3. <TaggedValue>*	NamedElement 1. (<<RhpModelElement>>) String Description 2. <Property>* 3. <Property>* ³
IRPType(Struct)	DataType	DataType <<RhpStruct>>
IRPType(Union)	DataType	DataType <<RhpUnion>>
IRPType(Typedef) 1. typeDefBaseType 2. isTypedefReference 3. typedefMultiplicity	DataType 1. Dependency to typeDefBaseType 2. TaggedValue isReference	DataType <<RhpTypeDef>> 1. <Property> baseType ⁴ 2. (<<RhpAttribute>>) Boolean isReference = true 3. baseType.lowerValue,

¹ This class allows to add classifier features to the UML Package

² In Rhapsody, an Actor can have the same features as a Class. In order to map these features to UML 2.1, the Rhapsody Actor is mapped to a Class with a stereotype <<RhpActor>>.

³ Tags not exist in UML 2.1 which suggests using Stereotype with properties. For same reasons, rhapsody.Properties are mapped to properties of applied stereotypes. All Stereotypes of Rhapsody Properties are nested in a Package "RhpProperties"

4. isTypedefOrdered 5. isTypedefConstant	3. taggedValue multiplicity 4. taggedValue isOrdered 5. taggedValue isConstant	baseType.upperValue 4. baseType.isOrdered 5. baseType.isReadOnly
IRPType(Language) 1. TaggedValue declaration	DataType 1. TaggedValue declaration	DataType <<RhpLanguage>>
IRPType(Enumeration) <ul style="list-style-type: none"><IRPEnumerationLiteral>* enumerationLiterals	Enumeration <ul style="list-style-type: none"><EnumerationLiteral>* literal	Enumeration <ul style="list-style-type: none"><EnumerationLiteral>* ownedLiteral
IRPEnumerationLiteral <ul style="list-style-type: none">NameValue	EnumerationLiteral <ul style="list-style-type: none">NameN/A	EnumerationLiteral <ul style="list-style-type: none">Name<LiteralString> specification

3.2. Features

Rhapsody Model Element	UML 1.3	UML 2.1
IRPOperation 1. <RPPParameter>* arguments 2. Boolean constant 3. Boolean isStatic 4. Boolean isAbstract 5. IRPClassifier returnType 6. IRPFlowchart flowchart 7. isCtor=true 8. isDtor=true	Operation 1. <Parameter>* parameter 2. taggedValue isConstant="true" if true 3. String ownerScope (static instance) 4. Boolean isAbstract 5. <Parameter>* parameter kind= return 6. Statemachine behavior 7. TaggedValue initializer 8. N/A	Operation 1. <Parameter>* ownedParameter 2. Boolean isQuery 3. Boolean isStatic 4. Boolean isAbstract 5. <Parameter>* ownedParameter Direction = return ⁵ 6. <Behavior> method 7. <<RhpConstructor>> + CreationEvent ⁶ 8. <<RhpDestructor>> + DestructionEvent
IRPEvent 1. <Attribute>* arguments 2. <IRPEvent> eventBase	Reception	SignalEvent + Signal 1. <Property>* ownedAttribute 2. <Generalization> generalization
IRPEventReception	Reception	

⁴ From Rhapsody to UML2.1, an attribute of the DataType, is created with the name baseType. This attribute has all features needed for expressing a TypeDef

⁵ Return value of an Operation is a parameter with the directionKind "return".

⁶ This kind of events can be used in Sequence Diagram to express the creation or destruction of an instance.

IRPAttribute 1. String Multiplicity 2. Initial value 3. Boolean Constant 4. Boolean IsStatic 5. String visibility (public, protected, private) 6. Boolean isReference	Attribute 1. TaggedValue Multiplicity 2. Expression Attribute.initialValue 3. taggedValue isConstant="true" if true 4. String ownerScope (static instance) 5. String visibility 6. taggedValue isReference="true" if true	Property 1. lowerValue, upperValue 2. Expression Property.defaultValue 3. Boolean isReadOnly 4. Boolean isStatic 5. String visibility 6. (<<RhpAttribute>>) Boolean isReference=true
IRPRelation 1. String multiplicity 2. String qualifier 3. isNavigable 4. Enumeration relationType (Aggregation, Composition, Association) 5. otherClass	AssociationEnd 1. Multiplicity exported as Multiplicity instance 2. <Attribute>* qualifier 3. Boolean isNavigable 4. AggregationKind aggregation (Aggregate, Composite, None) 5. Classifier type	Property <ul style="list-style-type: none"> Association⁷ (if not an AssociationClass) 1. Property is an instance of MultiplicityElement 2. <Property>* qualifier 3. if true : owner=ofClass if false : owner=association(ownedEnds) ⁸ 4. AggregationKind aggregation (Shared, Composite, None) 5. type
IRPAssociationClass 1. IRPRelation End1 2. IRPRelation End2	AssociationClass. 1., 2. <AssociationEnd>* connection	AssociationClass 1., 2. <Property>* ownedEnd

⁷ Properties are linked into association.memberEnds

⁸ In UML 2.1, a non navigable end is owned by the association

3.3. Generalization

Rhapsody Model Element	UML 1.3	UML 2.1
IRPGeneralization	Generalization	Generalization InterfaceRealization a) If (general Classifier not an Interface) and (derived Classifier not an Interface) <Generalization>* generalization else if (generalClassifier is an Interface) and (derivedClassifier is an Interface) <Generalization>* generalization else if (general Classifier is an Interface) and (derivedClassifier is an Interface) <InterfaceRealization>* interfaceRealization ⁹ b) If general Classifier GC is a template and derived Classifier DC is also template then the template signature of DC redefines the one of GC

3.4. Templates and template instantiations

Rhapsody Model Element	UML 1.3	UML 2.1
IRPOperation IRPClassifier 1. <RPTemplateParameter>* templateParameters 2. RPTemplateInstantiation ti a. <RPTemplateInstantiationParameters>* templateInstantiationParameters	Operation Classifier 1. <TemplateParameter>* templateParameter 2. Binding a. <Parameter>* argument	Operation Classifier 1. TemplateSignature RedefinableTemplateSignature a. <ClassifierTemplateParameter>* ownedParameter 2. TemplateBinding a. <TemplateParameterSubstitution>* templateParameterSubstitution

3.5. Annotations

Rhapsody Model Element	UML 1.3	UML 2.1
IRPComment 1. String name 2. String specification	Comment 1. name ¹⁰ 2. String body	Comment 1. (<<RhpComment>>) String name ¹¹ 2. String body Expression specification

⁹ In rhapsody, Interfaces are created by applying a Sterotype “Interface” on a Class. A Class generalizing this interface will realize it. In UML 2.1 this kind of relationship is expressed by InterfaceRealization

¹⁰ In UML 1.3, Comment derives from ModelElement, which has a name

¹¹ In UML 2.1, Comment has no name

3. RPModelElement Owner 4. <RPModelElement>* anchoredElements	BooleanExpression body 3. Dependency with “commentOwner” tagValue to owner 4. <ModelElement>* annotatedElement	3. ownedComment association 4. <Element>* annotatedElement
IRPRequirement 1. requirementID 2. specification	<<requirement>> comment	Constraint 1. (<<RhpRequirement>>) String requirementID 2. specification
IRPConstraint 1. RPModelElement Owner 2. specification	Constraint 1. <<constraintOwner>> dependency to Owner	Constraint 1. PackagedElement nestedPackage 2. specification

3.6. Statechart

Rhapsody Model Element	UML 1.3	UML 2.1
IRPStateMachine <ul style="list-style-type: none"> IRPState rootState 	StateMachine <ul style="list-style-type: none"> State top 	StateMachine or ProtocolStateMachine ¹² <ul style="list-style-type: none"> <Region>* region
IRPConnector <ol style="list-style-type: none"> isConditionConnector = true isDiagramConnector = true isForkConnector = true isJoinConnector = true isJunctionConnector = true isHistoryConnector = true isStubConnector isTerminationConnector = true 	PseudoState <ol style="list-style-type: none"> kind = "branch" S1=>DC=>S2 = S1=>S2¹³ kind = "fork" kind = "join" S1=>JC=>S2 = S1=>S2¹⁴ kind = "deepHistory" S1=>SC=>S2 = S1=>S2¹⁵ kind = "final" 	PseudoState <ol style="list-style-type: none"> kind = Choice kind = Junction kind = Fork kind = Join kind = Junction kind = DeepHistory ConnectionPointReference / PseudoState pair¹⁶ kind = Terminate
IRPState <ol style="list-style-type: none"> String stateType <ul style="list-style-type: none"> stateType = "Or" <Vertex>* subStateVertices stateType="And" <State>* subStates stateType="LocalTermination" stateType="EventState" String entryAction String exitAction <IRPTransition>* incomings/outgoings Statechart nestedStatechart 	StateVertex <ol style="list-style-type: none"> N/A <ul style="list-style-type: none"> SimpleState CompositeState <StateVertex>* subVertex FinalState CallState ActionSequence entry ActionSequence exit <Transition>* transitions¹⁷ SubactivityState submachine 	Vertex <ol style="list-style-type: none"> N/A <ul style="list-style-type: none"> State region State <region>* regions¹⁸ FinalState <<RhpSendAction>> State <ul style="list-style-type: none"> (<<RhpSendAction>>) Element target (<<RhpRequirement>>) Element event Activity entry Activity exit

¹² If the owner is an instance of Interface.

¹³ Transitively resolved

¹⁴ transitively resolved

¹⁵ transitively resolved

¹⁶ If a Rhapsody enter/exit point EEP' of a submachineState S has a matching Rhapsody enter/exit point EEP'' inside the submachine SM of S then EEP' is mapped to a UML ConnectionPointReference and EEP'' is mapped to a UML enter/exit point PseudoState

		4. <Transition>* transitions ¹⁹ 5. StateMachine submachine
IRPTransition 1. isDefaultTransition = 0 2. IRPAction itsAction 3. IRPGuard itsGuard 4. IRPTrigger itsTrigger 5. IRPStateVertex itsSource 6. IRPStateVertex itsTarget	Transition 1. N/A 2. Action effect 3. Guard guard 4. Event trigger 5. StateVertex source 6. StateVertex target	Transition 1. N/A 2. Activity effect 3. Constraint guard 4. Trigger trigger 5. Vertex source 6. Vertex target
IRPTransition • isDefaultTransition = 1	PseudoState • Kind = “initial” Transition (From InitialNode to target of transition)	PseudoState • Kind = “initial” Transition (From InitialNode to target of transition)
IRPAction String body	Action UninterpretedAction ActionExpression script	OpaqueBehavior String body
IRPTrigger(isOperation) • InterfaceItem operation	CallEvent	Trigger • Event event ²⁰
IRPTrigger(isTimeout) • body ²¹	TimeEvent	Trigger • TimeEvent event ◦ LiteralString when
IRPTrigger(isEvent)	Signal	Trigger • SignalEvent event

3.7. Activity Diagram²²

Rhapsody Model Element	UML 1.3	UML 2.1
IRPFlowchart	ActivityGraph	Activity
1. Owner instanceof	1. Context = Operation	1. Context = Class owing the operation that

¹⁷ Owned by the statemachine they are defined in.

¹⁸ Each region has a top level state representing the matching rhapsody component state

¹⁹ Owned by the region they are defined in.

²⁰ Actual metatype is one of (CreationEvent|DestructionEvent|SendOperationEvent event|SignalEvent) depending on whether the operation is a constructor, destructor, regular operation or triggered operation.

²¹ Timeouts use the format “tm(time)”

²² In Rhapsody, a Flowchart can own several Flowcharts through subactivity, and several defaultTransitions through Block. But in UML 2.1, an Activity Diagram doesn't allow this kind of hierarchy, just partitions through ActivityPartition

IRPOperation 2. Owner instance of IRPClass	2. Context = Class owning the activity diagram	owns the activity 2. Context = Class owning the activity diagram
IRPSwimlane	Partition	ActivityPartition
IRPState(isRoot) ²³		InitialNode
IRPState(LocalTermination) ²⁴ <ul style="list-style-type: none"> entryAction exitAction 		ActivityFinalNode <ul style="list-style-type: none"> (<<RhpFinalNode>>)String entryAction (<<RhpFinalNode>>)String exitAction
IRPState 1. stateType="Action" <ul style="list-style-type: none"> String entryAction 2. stateType="Block" 3. stateType="SubActivity" ²⁵ 4. stateType="ReferenceActivity" <ul style="list-style-type: none"> Flowchart referenceToActivity 5. stateType="LocalTermination" <ul style="list-style-type: none"> entryAction exitAction 	StateVertex 1. ActionState 2. CompositeState <ul style="list-style-type: none"> <StateVertex> subVertex 3. SubActivityState 4. SimpleState 5. FinalState	ActivityNode 1. OpaqueAction <ul style="list-style-type: none"> <String> body 2. ActivityPartition <ul style="list-style-type: none"> MergeNode entryState (containedNode) MergeNode exitState (containedNode)²⁶ 3. Only the block is mapped (see Block mapping). All transitions from/to this element are changed from/to the block 4. OpaqueAction <ul style="list-style-type: none"> (<<RhpReferenceActivity>>)String referenceToActivity 5. OpaqueAction/ActivityFinalAction ²⁷ <ul style="list-style-type: none"> (<<RhpFinalNode>>)String entryAction (<<RhpFinalNode>>)String exitAction
IRPObjectNode <ul style="list-style-type: none"> ModelElement represents 	ObjectFlowState <ul style="list-style-type: none"> Classifier type 	ActivityParameterNode <ul style="list-style-type: none"> Type type
IRPTransition <ul style="list-style-type: none"> (Target OR Source) 	Transition N/A	ActivityEdge <ul style="list-style-type: none"> ObjectFlow

²³ If his owner is the full flowChart²⁴ If his owner is the full flowChart²⁵ A subactivity contains a Block²⁶ In UML 2.1, an ActivityPartition is not a State. We can't have transitions from and to this Element. In order to map rhapsody transitions, we create two nodes, an entry node and an exit node where transitions can link to.²⁷ An ActivityFinalAction expresses the final action of all the Activity. A LocalTermination can not be map to an ActivityFinalAction if the state is in a Block or a SubActivity

instance of ObjectFlow <ul style="list-style-type: none"> • else <ol style="list-style-type: none"> 1. IRPAction itsAction 2. IRPGuard itsGuard 3. IRPTrigger itsTrigger 4. IRPStateVertex itsSource 5. IRPStateVertex itsTarget 	<ol style="list-style-type: none"> 1. Action effect 2. Guard guard 3. Event trigger 4. StateVertex source 5. StateVertex target 	<ul style="list-style-type: none"> • ControlFlow <ol style="list-style-type: none"> 1. (<<RhpActivityEdge>>)String action <OpaqueBehavior> 2. <LiteralString> guard 3. (<<RhpActivityEdge>>) String trigger 4. ActivityNode source 5. ActivityNode target
IRPTransition isDefaultTransition = 1	PseudoState Kind = “initial” Transition (From InitialNode to target of transition)	Like another Transition
IRPConnector <ol style="list-style-type: none"> 1. isDiagramConnector = true 2. isForkConnector = true 3. isJoinConnector = true 4. isJunctionConnector = true 5. isTerminationConnector = true 	Pseudostate <ol style="list-style-type: none"> 1. S1=>DC=>S2 = S1=>S2 2. kind = “fork” 3. kind = “join” 4. S1=>JC=>S2 = S1=>S2 5. kind = “final” 	ControlNode <ol style="list-style-type: none"> 1. MergeNode 2. ForkNode 3. JoinNode 4. MergeNode

3.8. Collaboration/Interaction diagrams

Rhapsody Model Element	UML 1.3	UML 2.1
IRPCollaboration <ol style="list-style-type: none"> 1. <RPAssociationRole>* associations 2. <RPClassifierRole>* classifier 3. <RPMMessage>* message 4. <RPMMessagePoint>* messagePoints <ol style="list-style-type: none"> a. From a message b. From interactionOccurrence 	Collaboration <ol style="list-style-type: none"> 1. <AssociationRole>* ownedElement 2. Collection<ClassifierRole>* ownedElement 3. Interaction <ul style="list-style-type: none"> <Interaction>* message 4. Interaction <ul style="list-style-type: none"> <Interaction>* message²⁸ 5. N/A 	Interaction <<RhpCollaborationDiagram>> <<RhpSequenceDiagram>> ²⁹ <ol style="list-style-type: none"> 1. <Connector> 2. <Lifeline> lifeline 3. <Message> message 4. Collection <ol style="list-style-type: none"> a. MessageOccurrenceSpecification b. InteractionUse c. CombinedFragment 5. ExecutionSpecification

²⁸ With a tagged value for keeping track of messages order.

²⁹ It depends of the owner of the Collaboration

c. From interactionOperator		
5. ExecutionOccurrence		
<p>IRPMessage</p> <ol style="list-style-type: none"> IRPAssociationRole communicationConnection IRPClassifierRole source IRPClassifierRole target String sequenceNumber String timerValue <String>* actualParameterList String messageType <ol style="list-style-type: none"> Constructor Destructor Operation Trigger Triggered <ul style="list-style-type: none"> IRPInterfaceItem formalInterfaceItem Event <ul style="list-style-type: none"> IRPInterfaceItem formalInterfaceItem <Other> 	<p>Message</p> <ol style="list-style-type: none"> AssociationRole CommunicationConnection Classifier sender Classifier receiver TaggedValue "SequenceNumber" TaggedValue"TimerValue" <Argument>* actualArgument Action action <ol style="list-style-type: none"> CreateAction TerminateAction CallAction <ul style="list-style-type: none"> Operation operation SendAction <ul style="list-style-type: none"> Signal signal UninterpretedAction 	<p>Message<<RhpMessage>></p> <ol style="list-style-type: none"> Connector connector MessageOccurrenceSpecification messageEnd MessageOccurrenceSpecification messageEnd (<<RhpMessage>>)String SequenceNumber (<<RhpMessage>>)String TimerValue N/A <MessageSort>messageSort <ol style="list-style-type: none"> MessageSort.CreateMessage MessageSort.DeleteMessage Else : MessageSort.SynchCall
<p>IRPAssociationRole</p> <ul style="list-style-type: none"> <RPRelation>* formalRelations <RPClassifierRole>* classifierRoles 	<p>AssociationRole</p> <ul style="list-style-type: none"> Association base <ul style="list-style-type: none"> <AssociationEndRole>* connection N/A 	<p>Connector</p> <ul style="list-style-type: none"> Association type <ul style="list-style-type: none"> <ConnectorEnd>* end ConnectorEnd end <ul style="list-style-type: none"> End.role
<p>IRPClassifierRole</p> <ol style="list-style-type: none"> IRPClassifier formalClassifier 	<p>ClassifierRole</p> <ol style="list-style-type: none"> Classifier base String Name 	<p>Lifeline</p> <ol style="list-style-type: none"> Property Role type=formalClassifier

2. String roleType	3. N/A	
3. IRPSequenceDiagram referencedSequenceDiagram		

3.9. Components/Deployment Diagrams

Rhapsody Model Element	UML 1.3	UML 2.1
IRPComponent		Component + Artifact ³⁰ 1. <<RhpComponentArtifact>>.additionalSources 2. Applying <<executable>> 3. Applying <<library>> 4. nestedArtifacts:Artifact 5. <<RhpComponentArtifact>>.includePath 6. <<RhpComponentArtifact>>.libraries 7. ownedClassifiers:Classifiers 8. <<RhpComponentArtifact>>.path 9. <<RhpComponentArtifact>>.scopeAllElements 10. manifestation:Manifestation 11. <<RhpComponentArtifact>>.standardHeaders
IRPFile		<<file>>Artifact 1. elements : ModelElement 2. fileFragments : FileFragment 3. files : File 4. fileType=FileKind 5. path 1. manifestation:Manifestation 2. nestedArtifact:Artifact 3. nestedArtifact:Artifact 4. <<RhpFile>>.fileType 5. fileName:String = path+name

³⁰ Some components features are mapped in a nested artefact named “DefaultComponentArtifact” of the UML Component. It applies a stereotype “RhpComponentArtifact” to distinguish it from other nested files.

IRPFileFragment 1. fragmentElement : ModelElement 2. fragmentText : String 3. fragmentType : String 4. name		<<RhpFile>>Artifact (with isFragment=true) 1. manifestation:Manifestation 2. <<RhpFile>>.fragmentText 3. <<RhpFile>>.fragmentType 4. name
IRPComponentInstance 1. componentType : Component 2. node : Node		Deployment 1. deployedArtifact(=DefaultComponentArtifact) 2. location : DeploymentTarget

Sequence Diagram

Rhapsody Model Element	UML 1.3	UML 2.1
IRPInteractionOccurrence <ul style="list-style-type: none"> referenceSequenceDiagram 	N/A	InteractionUse <ul style="list-style-type: none"> refersTo
IRPExecutionSpecification	N/A	ExecutionSpecification
IRPInteractionOperator <ul style="list-style-type: none"> <InteractionOperand>* interactionOperands String interactionType 	N/A	CombinedFragment<<RhpInteractionOperator>> <ul style="list-style-type: none"> <InteractionOperand>* operand (<<RhpInteractionOperator>>)String interactionType
IRPInteractionOperand	N/A	InteractionOperand

3.10. Flows and FlowItems

Rhapsody Model Element	UML 1.3	UML 2.1
IRPFlow <ol style="list-style-type: none"> <RPModelElement>* Conveyed String Direction ModelElement End1 ModelElement End2 Port End1Port Port End2Port 	N/A	InformationFlow <ol style="list-style-type: none"> conveyed <ol style="list-style-type: none"> If instance of classifier conveyed = conveyed Else conveyed =InformationItem +Dependency<<RhpRepresented>>³¹ N/A Dependency realization³² Dependency realization.³³
IRPFlowItem <ul style="list-style-type: none"> <ModelElement>* represented 	N/A	InformationFlowItem <ul style="list-style-type: none"> Same transformation than conveyed in InformationFlow

³¹ If mapped element is not a classifier, we create a InformationItem with a dependency to this element

³² The relationship between the source and target of the flow in Rhapsody is mapped to a <<RhpFlow>> dependency between their UML 2.1 counterparts.

³³ Source and target depend on the feature direction in IRPFlow

3.11. UseCase Diagram

Rhapsody Model Element	UML 1.3	UML 2.1
IRPUseCase 1. <String>* extensionPoints 2. <SequenceDiagram>* describingDiagrams	UseCase 1. <ExtensionPoints>* extensionPoint 2. <Dependency<<Realizes>>> ³⁴	UseCase 1. <ExtensionPoints>* extensionPoint 2. Collection<(<<RhpUseCase>>)Interaction describingDiagrams

3.12. Ports

Rhapsody Model Element	UML 1.3	UML 2.1
IRPClass (with ports) <RPPort>* ports	N/A	Class Interface <Port>* ownedPort ownedAttribute ³⁵
IRPPort 1. RPCClass Contract 2. Bool isBehavioral 3. Bool isReversed 4. <RPCClass>* providedInterfaces 5. <RPCClass>* requiredInterfaces	N/A	Port 1. Class <<RhpContract>> type ³⁶ 2. Boolean isBehavioral 3. (<<RhpPort>>) Boolean isReversed 4. <Interface>* provided ³⁷ 5. <Interface>* required ³⁸

3.13. Instances & Links

Rhapsody Model Element	UML 1.3	UML 2.1
IRPInstance 1. RPCClassifier otherClass 2. <RPLink>* outLinks <ul style="list-style-type: none"> Relation instantiates Instance to 	Instance 1. <Classifier>* classifier 2. <Link>* ownedLink 3. <AttributeLink>* slots AttributeLink.value.Name	InstanceSpecification<<RhpInstance>> (Part of nesting Classifier) (<<RhpInstance>>)Property specifiedPart ³⁹ <ol style="list-style-type: none"> Classifier classifier <Slot>* slot

³⁴ The suppliers are the collaborations representing the sequence diagrams

³⁵ In UML2.1 an Interface does not have the ownedPort feature, so Ports are contained through ownedAttribute feature

³⁶ The Rhapsody contract is mapped to the type of the port. If the contract is implicit, a new UML class is created to hold it.

³⁷ Provided Interfaces come from the Interfaces implemented by the contract class. For each Rhapsody provided interface, a UML interfaceRealization relationship is created between the UML Port type and the UML provided interface. The list of required Interfaces

³⁸ In Rhapsody, there is a <<Usage>> dependency created between the port contract and each of its required interface. This <<Usage>> dependency is mapped to a Usage relationship in UML and the list of required interfaces in UML 2.0 is derived from the list of Usage relationships starting from the UML port

³⁹ If the Instance is owned by a Package, it will be a part of its TopLevelClass.

3. <RPAttributeValue>*AttributeValues <ul style="list-style-type: none"> Attribute Attribute String Value 4. RPOperation instantiatedBy 5. <String>*ListOfInitializerArguments 6. ObjectAsObjectType	4. Dependency with “instantiatedBy” tagValue to constructor 5. <TaggedValue>* (ArgumentName,ArgumentValue) defined on the “instantiatedBy” dependency	<ul style="list-style-type: none"> Property definingFeature <InstanceValue>* values 3. <Slot>* slot <ul style="list-style-type: none"> Property definingFeature <LiteralString> values 4. N/A 5. N/A 6. Class<<RhImplicitType>>
IRPBlock	Instance TaggedValue “isBlock”=“true”	
IRPLink 1. RPRelation instantiates 2. RPLink other 3. RPRelation from 4. RPRelation to 5. Port fromPort 6. Port toPort 7. String end1Multiplicity 8. String end1Name 9. end2Multiplicity 10. end2Name	Link 1. Association association 2. N/A 3. <LinkEnd>* connection (member) 4. <LinkEnd>* connection (member) 5. N/A 6. N/A 7. LinkEnd multiplicity 8. LinkEnd name 9. LinkEnd multiplicity 10. LinkEnd name	Connector 1. Association type 2. N/A 3. <ConnectorEnd>* end (member) <ul style="list-style-type: none"> ConnectableElement role Property PartWithPort 4. <ConnectorEnd>* end (member) <ul style="list-style-type: none"> ConnectableElement role Property PartWithPort⁴⁰ 5. 3.b 6. 4.b 7. a ConnectorEnd is a MultiplicityElement 8. name of end in 3. 9. a ConnectorEnd is a MultiplicityElement 10. name of end in 4.

⁴⁰ partWithPort references the part (Instance) of a classifier

- If a connector end is attached to a port of the containing classifier, partWithPort will be empty.
- If a connector end references both a role and a partWithPort, then the role must be a port that is defined by the type of the partWithPort.
- The property held in self.partWithPort must not be a Port.

4. UML 2.1 RESTRICTIONS

There are some known restrictions in the UML2.1 support. They are detailed hereafter

4.1. Export restrictions

The following Rhapsody constructs are not exported

- Structure diagrams
- Deployment diagrams

4.2. Import restrictions

The following Rhapsody constructs are either not imported, or imported with slight adjustments..

Not imported

- Collaboration diagrams
- InteractionUse elements in sequence diagrams

Adjustments

- Blocks are imported as instances

See User Guide for more details about XMI Toolkit limitations.