**Date**: ~~March 2008~~August 2009

# Model Driven Message Interoperability (MDMI), Beta ~~1~~2

*OMG Adopted Specification*

**Previously known as Conversion Models for Payment Messages (~~CM4PM~~MDMI)**

OMG Document Number: dtc/2008-08-01
Standard document URL: http://www.omg.org/spec/MDMI/1.0/PDF Associated File*: http://www.omg.org/spec/MDMI/20070901

** original file: finance/07-09-03

This OMG document replaces the submission document (finance/2007-09-02, Alpha). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome, and should be directed to *issues@omg.org* by August 25, 2008.

You may view the pending issues for this specification from the OMG revision issues web page *http://www.omg.org/issues/.*

The FTF Recommendation and Report for this specification will be published on October 15, 2008. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

## USE OF SPECIFICATION — TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

## LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

## PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

## GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means — graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems — without permission of the copyright owner.

## DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED

BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

## RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 140 Kendrick Street, Needham, MA 02494, U.S.A.

## TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management GroupTM, OMGTM , Unified Modeling LanguageTM, Model Driven Architecture LogoTM, Model Driven Architecture DiagramTM, CORBA logosTM, XMI LogoTM, CWMTM, CWM LogoTM, IIOPTM , MOFTM , OMG Interface Definition Language (IDL)TM , and OMG SysMLTM are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

## COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

# OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.org*, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.htm).

# Table of Contents

# Preface

## About the Object Management Group

### OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling LanguageTM); CORBA® (Common Object Request Broker Architecture); CWMTM (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at *http://www.omg.org/.*

## OMG Specifications

As noted, OMG specifications address middleware, modeling, and vertical domain frameworks. A catalog of all OMG Specifications is available from the OMG website at:

*http://www.omg.org/technology/documents/spec_catalog.htm*

Specifications within the Catalog are organized by the following categories:

### OMG Modeling Specifications

- UML
- MOF
- XMI
- CWM
- Profile specifications

### OMG Middleware Specifications

- CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM)

**Platform Specific Model and Interface Specifications**

- CORBAservices

- CORBAfacilities
- OMG Domain specifications
- OMG Embedded Intelligence specifications
- OMG Security specifications.

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

## Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

**Helvetica/Arial - 10 pt. Bold:** OMG Interface Definition Language (OMG IDL) and syntax elements.

**`Courier - 10 pt. Bold:`** Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

**Note—:** Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

## Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/technology/agreement.htm.*

# 1 Scope

Complete financial transactions often involve multiple steps that require the transmission of information across financial systems in multiple enterprises. Each step of a transaction usually relies on the transmission of information via standardized messages. Some examples of standardized message formats utilized in financial services are MDDL, FIX, FpML, IFX, TWIST, SWIFT messages, Visa messages, RosettaNet, OAGi, ACORD, and CIDX. Each of these standards provides a particular type of functionality within the financial service industry. For example, FIX deals with front-office transactions in the securities sector, while a certain group of SWIFT messages will deal with back-office security transactions, such as clearing and settling, in the same sector. Each set of financial message standards is usually supported by a separate industry standards body, e.g., SWIFT for SWIFT messages, Visa for Visa Messages, the FIX Protocol committee for the FIX standard, etc. The messages created by these groups have evolved over many years with little or no coordination between groups.

To get true Straight Though Processing (STP), information must be correctly interpreted and processed by each involved financial system at each step of the financial transaction. This implies – amongst other things -- that information must be accurately moved from one system to the next. This may require moving information from one message format to another~~. For example, data in~~, e.g., from a FIX ~~messages to data in~~pre-trade message into a SWIFT ~~messages.~~settlement message. In addition, a financial institution will often have ~~their~~its own internal data elements used either in internal data stores or in internal messages. These internal data elements must also be appropriately mapped to and from the industry standard messages if information is to be transmitted from one institution to another. Currently, the mapping of financial data from one format to another is not standardized. The mappings are usually done in an ad hoc procedural manner. The complicated and complex maze of existing formats and hard-coded transformations has created an environment where every introduction of new message formats, and even changes to older messages, is very expensive. The goal of the ~~Conversion Models for Payment Message (CM4PM)~~current standard (MDMI) is to provide a declarative, model-driven mechanism to perform message data transformation ~~-~~ – not only to handle the movement of data between different message formats, but also to ~~create a~~support versioning by providing a mechanism ~~so that~~ to map information between a new ~~version of a message can be mapped to~~ and an older version~~.~~ of the same message. Thus, the ~~Conversion Models for Payment Message~~current standard can help reduce the barriers that prevent the introduction of new versions of messages and thereby greatly reduce the cost of change~~.~~.

The Finance Domain Task Force wishes to emphasize that this specification is intended for use by the financial services community, and has been developed with its specific needs and requirements in mind. While it can certainly be envisioned that the concepts, models, and mechanisms described in this specification can be applied or adapted to other application domains, it is not the intent of this document to cover other than the financial services domain.

# 2 Conformance

To be compliant with the specification, an implementation would need to be able to create ~~and consume, as applicable,~~ the artifacts that are shown in ~~figures 1 and 2 of~~ the model specification ~~("overview of proposed runtime specification"), and~~ (OMG document # 2009-09-09); to utilize expression languages that are ~~elaborated upon in detail in this specification. In particular~~consistent with the constraints described in section "8.1.4 MessageGroup – Detailed"; to utilize MDMIDatatypes that are consistent with the description and constraints in section 8.4, and to utilize a central dictionary that provides a function delivering a unique identifier as described in section 8.1.5. In addition, an implementation needs to support a ~~design-time activity with all of the steps involved in generating the conversion maps, and the~~ runtime application ~~of~~, as described in figures 7.1 and 7.2, (See section 7.1 Informal Overview of artifacts) that can consume the generated maps ~~to~~ and match unique identifiers to provide a transformation of a Semantic Element from a source ~~physical~~ message ~~in order~~ to ~~create~~ a target ~~physical~~ message. ~~The runtime aspects of the implementation form the normative part of this specification.~~

# 3 Normative References

~~**NOTE:** If applicable, this needs to be completed.~~

This specification references ISO 20022. A complete reference for ISO 20022 can be found at www.ISO20022.com.

# 4 Terms and Definitions

**~~Autonomy~~**

~~Autonomy is defined when each domain authority retains or delegates management and share-ability control over the resources it offers for sharing.~~

**~~Business analyst~~**

~~An actor who is familiar with the area of interest of a message set, for example payments, and who can design semantic maps.~~

**Business Element**

A Business Element is ~~a refinement of a Semantic Element that represents a~~ the smallest semantic unit in an external dictionary. For example, in ISO20022 Business Elements are the attributes of Business Component (or their related Message) classes and represent a "business concept~~. A Business Element is associated with a Domain Data Dictionary.~~".

**~~Business rule~~**

~~A Business Rule is associated with a Semantic Element and describes constraints on the value of any instance of that Semantic Element.~~

**Composition**

A configuration of related entities that ~~result~~results in a new entity at a different level of abstraction~~. That~~ that is, a composition is a grouping of two or more entities that can be referred to as a single entity at a different level of abstraction from its component entities~~.~~.

**Conversion Rule**

A ~~feature of a semantic map that describes a~~ rule that is to be applied to ~~a conversion between~~convert a value ~~in~~of a source ~~message element and~~Semantic Element into a value ~~in~~of a target business element or a target ~~message element.~~Semantic Element

**Datatype**

A prescription of the form of the data that has no specific ~~business or~~ message format related semantic content, for example an address, a date, etc.

**~~Domain~~**

~~A domain is a particular form of group in which the particular aspect of the behavior of the objects in the group is controlled by the same authority.~~

**~~Domain Data~~ Federated Dictionary**

~~That section of a Domain Repository that contains the reusable items, specifically business elements, qualified business elements, qualifiers.~~

**~~Domain Repository~~**

~~An ISO20022 compatible repository that focuses on a domain such as Banking, Securities, eCommerce, etc.~~

**Entity**

Any abstract or concrete thing.

**Environment**

All those entities that are not part of the modeled entities.

**Federated dictionary**

A federated dictionary is a collection of multiple domains that shares definitions while retaining autonomy over the resources.

**Federate**

To Federate is the action of federating a configuration of components.

**Group**

A Group is a set of objects or models grouped together for structural reasons or because the behavior of the objects or models have common features.

**Keyword List**

A Keyword List is a list of name-value pairs, which may include a reference to a formal taxonomy.

**Leaf Format**

The attribute of the Leaf Syntax Translator that contains a description of the physical format of the associated Message Element in its Physical Message.

**Leaf Format Expression Language**

An attribute in a Leaf Syntax Translator that identifies the language used to define a format

**Location**

The attribute of the Leaf Syntax Translator that contains a description of the location of its associated Node in its Physical Message.

**Location Expression Language**

An attribute in a Leaf Syntax Translator that identifies the language used to define a Leaf Location.

**Leaf Syntax Translator**

A class whose properties provide the necessary to extract or insert the value of a Message Element Instance to or from a Physical Message.

**Message Element**

A Message Element is a refinement of a Semantic Element that represents a smallest semantic concept in a message format

**Message Element Relationship Model**

A model of the directed associations between a Message Element and one or more Message Elements, in the same Message Element set.

**Message Composite**

A Message Composite represents a composition of Message Composites, Simple Message Composites or Message Elements.

**Message format**

A message format definesA collection of physical Data Dictionaries, whereby each data dictionary contains Business and Semantic Elements that are relevant to a particular domain of the financial industry and whereby the collection of all Business and Semantic Elements represents a single logical data dictionary for the financial industry.

**Message Format**

Definition of the syntax and semantics of a class of messages. Can be defined in many ways including paper documentation

**MXxx**

Message **Group**

A message group consists of all allowable Message Models that are categorized together in a setformat developed according to reflect a domain of interest usually created by a standards body, e.g. all SWIFT 15022 messages, all Visa TCxx messages, etc.

**Message Element Set**

A set of Message Elements, Message Composites and Simple Message Composites and Message Element Relationship Model that represent the semantics contained in a message format.

**Message Model**

A Message Model models a message format and is composed of a Message Element Set and a Message Syntax Model.

**Message Syntax Model**

A Message Syntax Model is associated with a Message Element set and models the syntax of a message format.

**Message Syntax Translator**

The combination of a defined process and a set of specifications that, when executed, can insert or extract data from a message.

**MSxx**

SWIFT message sets that utilize a standard XML format based on ISO 20022 specification.

**MTxx**

SWFT message sets that are based onMessage format developed according to the SWIFT defined EDI formatsspecification, including the ISO 15022 messages.

### Near Synonym

A Semantic Element that can be ~~mapped~~derived using prescribed mapping rules ~~to~~from a set of other Semantic Elements ~~**Node ID**~~, thus lying within a clearly bounded semantic distance from those Semantic Elements.

~~A Node ID is a unique identifier for each node in a syntactic model, e.g., the path from the root to that node. All Nodes in a Syntactic Model have a Node id.~~

**Physical Message Instance**

An instance of a message that is used to transmit information from a source to a target application ~~. **Qualified Business Element**~~

~~A UML class representing a business element that has been qualified, i.e., specialized to reflect a business context. E.g., a given unqualified business element = client account; qualified business element = *primary* client account, where primary is a qualifier~~

~~**Qualifier**~~

~~Prescriptive predicate defined in an approved structured set that can be used to refine a Semantic Element.~~

~~**Runtime Application**~~

~~A set of processes that can perform transformations by utilizing instances of artifacts such as those defined for the CM4PM standard.~~

**Semantic Element**

An ~~object that represents a concept whose properties are datatype and value. It may also have optional properties that include a description and a Keyword List.~~entity in a message format that represents a "smallest" business concept specific to that message format. The easiest way to describe is by analogy. If the information in a message were used to define a denormalzed table in a database table, then the Semantic Elements would represent the columns of that table.

**Semantic Element Set**

A set of SemanticElements, MessageComposites and Simple MessageComposites and SemanticElementRelationships that represent the semantics contained in a message format.

**Semantic Map**

A map that describes the relationship between a ~~Message~~Semantic Element in a ~~Message~~Semantic Element Set and a Business Element in a Domain Data Dictionary or between a ~~Message~~Semantic Element in one Message Model and a ~~Message~~Semantic Element in another Message Model.

~~**Simple Message Composite**~~

~~A Simple Message Composite is a composite of one or more Message Elements.~~

**Synonym**

A Semantic Element that can be mapped to another ~~semantic element~~Semantic Element by simple equivalence, i.e., A=B.

**TCxx**

~~The set of Visa message formats~~Message format developed according to the VISA EDI specifications for retail banking ~~transactions.~~applications.

~~**Technical analyst**~~

~~An actor that has the ability to create Message Models.~~

~~**Type**~~

A conceptualization of a property of one or more entities. Type is a predicate.

**UNIFI**

The name associated with the ISO 20022 message sets associated with the financial service domain

**Wire format**

The physical syntax of a Physical Message as it is transmitted "over the wire" e.g., XML, Edifact, flat-file record, etc.

# 5 Additional Information

## 5.1 Acknowledgements

The following companies submitted and/or supported this specification:

FireStar Software, Inc.

IBM Corporation

Informatica Corporation

IP Commerce

Visa International, Inc.

Barry Steer acted as a consultant to Visa International, Inc. and was a co-author of this work. Barry's contact information is: SteerConsulting, bsteer@earthlink.net, 650-375-0424.

Adaptive

The authors wish to acknowledge the contributions of Gabriel Oancea, David Frankel, and Christian Nentwich for their work in refining the standard. We would like to acknowledge Kris Ketels, Pete Rivett,Said Tabet and Frank Vandamme in reviewingfor thrit careful and constructive review the materials. The authors also would like to make a special acknowledgement for Pete Rivett for very careful review and suggestions for both the documents and the specification and for Sridhar Iyengar for his patience and guidance.

# 6 Overview

Given the lack of a financial industry-mapping standard, data is usually mapped directly from one message format to another. It is a well-known principle in the field of system architecture that as the number of interfaces in a "system" increases linearly the cost of maintaining point-to-point mappings increases geometrically. In addition, since many of these mappings are done locally and procedurally, errors are easily introduced. All financial organizations face this situation. ~~It is estimated~~ Certainly, financial organizations ~~currently~~ spend ~~some 20-40%~~a good deal of their software development budget on developing new interfaces and mappings or extending existing ones. In addition, it is very hard to introduce any changes into existing message formats or introduce new formats because of the tremendous cost of changing applications that process the older message formats.

The goal of the ~~CM4PM~~MDMI standard is to provide a standard framework and methodology for the financial services industry, which will alleviate the mapping problem.

This standard will:

- Reduce significantly the cost and time needed to define conversion rules to map data from one message format to another~~;~~.
- Handle versioning issues as particular message ~~standards change;~~formats evolve over time.
- Allow the expedited adoption of new standards ~~--~~ as mapping the new standard to the existing standard will allow applications to continue to use the legacy standards thus greatly reducing the introduction cost of new standards.
- Improve the interoperability and STP in end-to-end financial transactions that are based on multiple message formats.

The ~~CM4PM~~MDMI standard's framework is based on two concepts:

- First, removing any syntax associated with a message format, revealing the set of core "~~message elements~~Semantic Elements" contained in that message format. A ~~message element~~Semantic Element is the smallest semantic unit defined in a message format.
- Second, specifying a semantic map of those ~~message elements~~Semantic Elements to an industry accepted data dictionary made up of "~~business elements.~~"Business Elements." A ~~business element~~Business Element is the smallest ~~semantic element~~Semantic Element that is an entry in the dictionary and representing ~~a~~ business concept for the industry sector.

The easiest way to recognize ~~message elements~~Semantic Elements or business elements is that they cannot be constructed from other ~~message elements~~Semantic Elements or business ~~element~~elements, respectively, i.e., they are represented by a class, whose ~~only~~ primary property is a general data type~~. (See section 4.4)~~.

Providing semantic maps to a central data dictionary creates a "hub and spoke" approach to mapping as each standards body need only develop maps to the standard data dictionary. A mapping then will have two steps, utilizing a map from a source to the data dictionary and then utilizing a map from the data dictionary to the target. Thus, the mapping process is reduced from being geometric to being linear with the number of message formats.

## 6.1 ~~UNIFI~~Relationship to ISO 20022

To be effective, there needs to be an industry-wide consensus on the semantic content of the business elements in the data dictionary and there needs to be an organization that will take on responsibility for maintaining its integrity. In the financial services industry the responsible organization is TC68 and its working groups as outlined in part 2 of the ISO 20022 standard ~~under the rubric "UNIversal Financial Industry message scheme" or UNIFI.~~.

~~Currently UNIFI working groups are redesigning the framework of its data dictionary. The CM4PM standard is designed to accommodate this new framework. They have embraced the concept of a data dictionary made up of a set of business elements, where a business element is the smallest semantic element that defines a business concept in a particular domain of Financial Services.~~

~~For example:~~

In the ISO 20022 Data Dictionary, Message Elements, which are properties of Message Components (where Message Components, in turn, are related to Business Components), are the equivalent of the Business Elements as defined in this specification. Thus, Semantic Elements can be mapped to the Message Elements in ISO_20022.

Examples of Message Elements:

- The amount in a client's retail bank account
- The name of a bank branch
- The name of the sender of a wire transfer

~~These common business elements can be derived from independent modeling of financial processes, from the reverse engineering of existing messages or from the specific request for inclusion of a new element that responds to a new market requirement. A user should be able to use the set of business elements as Lego blocks, to build new syntax-independent message formats models, which can then be transformed into syntax specific message formats according to any desired syntax. Applications that are "aware" of the UNIFI business elements would then be able to process these new messages without further coding. Mapping from UNIFI data dictionary and thus any of these new messages to data defined in existing messages would be accomplished based on maps developed using the CM4PM standard.~~

## ~~6.2 CM4PM, UNIFI and~~ 6.2Different Ways to Use the Current Standard

### 6.2.1 Moving Data ~~Dictionaries~~

~~UNIFI includes a number of working groups that define various industry-wide messaging standards and maintains the UNIFI repository. The WG4 working group of ISO 20022 is responsible for creating the framework and criteria for what elements should be entered into the UNIFI data dictionary. It is currently defining a next generation reference framework for the data dictionary. A key aspect of the new framework will be to make sure that there is not redundancy or ambiguity among the business elements in the dictionary.~~

~~The CM4PM standard can play an important part in creating this robust UNIFI data dictionary. Given a business element candidate, if there is redundancy, it should be possible to map the submitted business element to existing business elements in the dictionary using the semantic mapping component of the CM4PM standard (see section 4.5.) If a mapping is not possible, the business element has enough semantic "distance" from the other business elements to be entered as a new business element in the UNIFI dictionary.~~

~~(The combination of the semantic mapping in the CM4PM standard and the new reference framework for the UNIFI Data dictionary defines a methodology that can be generally applied to the creation of robust data dictionaries for message groups. This is an extra benefit of the CM4PM standard.)~~

## ~~6.3 Use of the Conversion Models for Payment~~from One Message ~~Standards~~ ~~6.3.1 Moving data from one message to another~~to Another

The primary focus of the ~~CM4PM~~MDMI standard is moving some information from a source message in a message format that has been defined by one standards body to a target message in a message format independently defined by another standards body utilizing an industry defined central data dictionary.

For example:

One message format may define a "client address" field while another message format may have separate fields for "client street," "client city," "client state," etc.

One message format may define a bank ID number as a BIC number while another message format may define a bank ID as an ABA routing number.

~~Two~~The key is that the fields in each message are mapped to the same central dictionary element. There are two important benefits of mapping to a central data dictionaries such as the ~~UNOIFI data dictionary are:~~ISO 20022 Repository:

1. ~~Only~~ The central dictionary creates a hub and spoke architecture for transformations. Therefore, only a linear set of transformation must be created among ~~a~~ different message format groups~~.~~ instead of the $n^2$ mappings required for bilateral transformations. For example, by using a central data dictionary for payments, only ~~four~~six maps need to be created to map payment information among SWIFT MT messages, SWIFT MX messages, FIX messages, Visa TC messages, RosettaNet messages and ~~FRB~~ ACH messages, whereas ~~six direct~~15 bilateral conversion maps would be needed~~.~~.
2. ~~• More importantly, given~~Given that a standards body or enterprise takes responsibility for creating standard conversion maps to a central dictionary, it need only be expert in its own message formats and the well-defined semantics of the central data dictionary, rather than needing to understand the semantics and syntax of many other message groups if the ~~direct mapping~~bilateral element method is employed.

### 6.~~3.~~2.2 Versioning

A second costly problem in the financial services space is versioning. The market continually requires changes in message formats. Given the legacy of existing software, even a small change in a message format can be prohibitively costly to implement. Thus, required changes are often implemented very slowly and, in the worse case, not implemented at all. By providing ~~CM4PM~~MDMI maps ~~of~~between new versions and older versions, new message formats can be introduced without requiring that existing message formats be abandoned or that legacy applications be re-coded, as long as the legacy applications do not utilize the new information in the new version.

### ~~6.3.3 Direct Mapping~~

### ~~CM4PM~~6.2.3 Moving Data from an Internal Enterprise Message Format to an External Standard

Another important value of MDMI is moving information from an enterprise's internal message or data formats to an external message standard. It is important to note that a record definition in a database schema can be considered to be a "message format" and maps can be generated that transform data from that internal database to an external standard. Currently large staffs are devoted to creating bilateral maps between their internal standard and the external standard. Whenever either message format changes, these maps must be changed. With MDMI maps, the Semantic Elements in is internal message formats are mapped to a central dictionary, such as the ISO 20022 Data Dictionary. Given that a standards body, such as SWIFT, distributes new MDMI maps to account for the change in their standard, then the internal enterprise maps do not have to be changed. This will result in very significant savings.

### 6.2.4 Bilateral mapping

MDMI can be used to model and define conversion maps directly between two message formats. In this case, the semantic mapping is between the ~~message elements~~Semantic Elements in a source message format and the ~~message elements~~Semantic Elements in a target message format. (The ConversionRules, which define the relationship between Semantic Elements must be as complicated as required to accomplish a mapping whereas conversion rules mapped to a central dictionary will have a restricted set of operators.)

## 6.~~4~~ 3 Basic ~~approach of the Conversion Models~~Approach for ~~Payment Message Standards~~the Use of This Standard

~~First, the~~ The artifacts defined for ~~the Conversion Models for Payment Message standards are influenced by viewing conversions as a mapping data~~this standard are designed to map data (i.e., sets of Semantic Elements) from one message format to another rather than the wholesale conversion of a complete message in one message format to another message format. With this focus, each data field conversion needs to be atomic, containing all the meta-data necessary to move

the data in the field to a target field (or fields) with as little reference to additional meta on -data such as a complete model of the message format..

The Conversion Models for Payment Message standards standard is a declarative standard based on a MOF compliant UML model that definedefines the artifacts necessary to define a standardized conversion. These artifacts represent a two-stage process:, as described below.

### 6.43.1 Stage 1

The first stage artifacts utilize a Message Syntax Model to create a syntax-neutral set of Message Semantic Element classes. Message Semantic Elements are the smallest Semantic Elements semantic entities contained in thata message typeformat, for which further parsing would lose semantic meaning leaving only generic data-type values.

### 6.43.2 Stage 2

The second stage provides semantic mapping. to a central dictionary. It does this by specifying To and From Conversion Rules for source MessageSemantic Elements either to target MessageSemantic Elements in another message format or to Business Elements in a the central data dictionary such as UNIFIthe ISO 20022 Repository.

In many cases, this mapping will amount to a simple isomorphic mapping; in other cases, simple transformations will be required, such as splitting or concatenating Message Elementsdefining an arithmetic expression, doing a table lookup, or splitting or concatenating a string. Separate transform may need to be defined for the mapping 1) from a source Semantic Element to a Business Element as compared to 2) from a Business Element to a Semantic Element.

For example:

- ~~• "Name" •~~ Mapping "Primary Client Identifier" element in the source message ~~maps~~ to the two elements ~~"GivenName"~~, "Primary Client Name" and "~~FamilyName"~~Primary Client BIC", in the ~~target.~~dictionary
- Mapping "Primary Account Beginning Balance" and "Primary Account Ending Balance" in the ~~source~~dictionary to "Primary Account Beginning Balance" and "Primary Account Debited Amount" in the target~~.~~

**Note:** There may be no simple or reasonable Conversion Rule ~~from~~between a source ~~Message~~Semantic Element ~~to~~and a ~~target~~ Business Element in an industry data dictionary ~~like UNIFI.~~, such as the ISO 20022 repository. This indicates that the ~~Message~~Semantic Element represents a concept not yet included in the industry data dictionary. In ~~that~~this case, a submission should be made to the governing body to enhance the industry data dictionary, rather than include a complex or convoluted mapping.)

## 6.~~5 4~~ Future Benefits of the ~~Conversion Models for Payment Message~~ Standard

There are a number of ~~future~~ extensions to the ~~Conversion Models for Payment Message~~MDMI standard that ~~can add~~ should enhance the value of the standard.

### 6.~~5.1 Use of Semantic Mapping for Structuring~~4.1 Dealing With (Near) Synonyms

A key feature of the ~~Conversion Models for Payment Message~~MDMI standard is the semantic mapping ~~is~~ that it is carried out between the ~~financial institutions Message~~ Semantic Elements and ~~the~~ Business Elements contained in an industry data dictionary, such as ~~UNIFI through a~~ the ISO 20022 Data Dictionary. These conversions should be restrictive to a small set of direct conversion rules~~.~~, e.g., only allowing arithmetic and logical expressions and limiting external functions to table lookups.

In effect, establishing such a set of rules ~~defines~~can be used to define the semantic proximity between the ~~Message Element and the~~ Business Elements~~.~~ (or in the case of ISO 20022 the Message Elements) in a data dictionary. This semantic proximity can be characterized as defining synonyms and "near synonyms." ~~The same mechanism~~ This is accomplished because terms that can be ~~used to provide a well-structured industry data dictionary. Given a~~mapped to the dictionary~~, only~~ using the conversion rules must be synonyms or "near synonyms." Only terms that are not synonyms or near synonyms of other Business Elements would be allowed in the basic dictionary itself. The synonyms and near synonyms with their mappings could be kept in an auxiliary catalogue. The allowable rules established for the Conversion Rules in effect define the minimum semantic distance that is allowed for dictionary entries, resulting in a "measurable" well-structured dictionary. The future work would involve defining appropriate sets of conversion rules and understanding their implication on the dictionary structure.

### 6.~~5~~4.2 Mapping between Data Dictionaries

The current standard is focused on supporting ~~message element~~Semantic Element conversions among one or more message standards within Financial Services by mapping ~~message elements~~Semantic Elements to one large, central Data dictionary. However, the ~~Conversion Models for Payment Message semantic~~MDMI Semantic mappings could be applied to create maps between data dictionaries. Thus, the ~~CM4PM~~MDMI standard could be used to effectively support federated dictionaries. This, in turn, will allow content aware standards groups to manage dictionaries for specific ~~subsections~~subsections of the financial services industry, as opposed to one group being responsible for a large data dictionary. A federated set of dictionaries ~~will~~might be more effective to maintain~~.~~.

### 6.~~5~~4.3 Handling ~~lossless conversion~~Lossless Conversion

An important need in messaging is dealing with the loss of information when performing ~~Message element~~Semantic Element conversions. While this problem can never be completely solved improvements in lossless conversions will be a great benefit. The proposed artifacts for the ~~CM4PM~~MDMI standard can provide a strong basic framework for creating lossless conversions, e.g., syntax incompatibilities can be traced and accommodated; auxiliary storage for lost information can be created with additional ~~Message~~Semantic Elements, etc~~.~~.

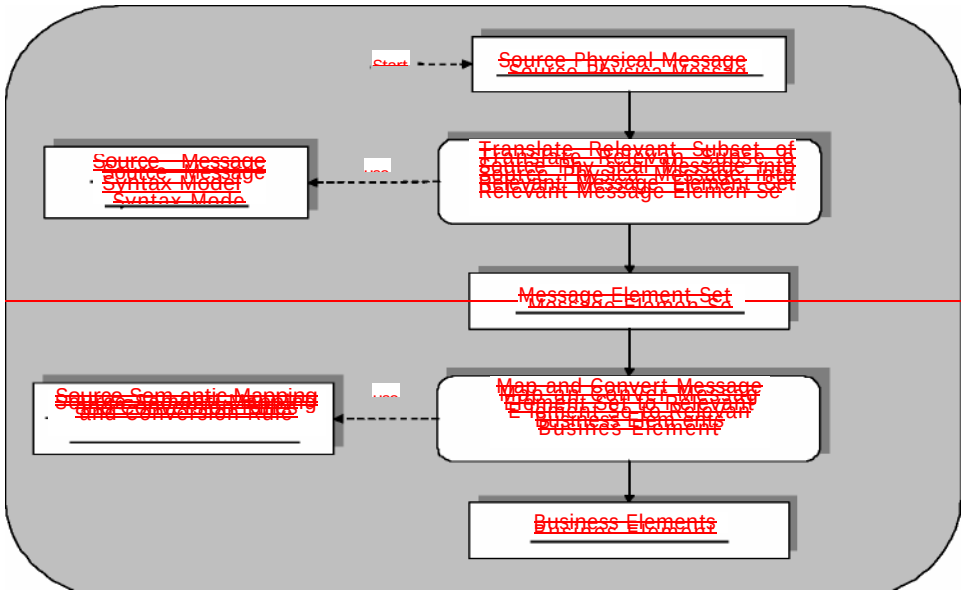
# 7 ~~UML Semantics~~ Use of MDMI Artifacts Overview

The focus of the ~~Conversion Models for Payment Message~~MDMI standards is to create a template for machine-readable maps that standardize the conversion of data from a source message instance based on one message format to data in a target message instance based on another message format. This may involve the movement of as little as one data element or it may involve the conversion of a complete ~~physical~~ message. The standard can be used to ~~convert~~map data for message formats within a Message Group or across Message Groups.

## 7.1 Informal Overview of ~~artifacts~~Artifacts

Before presenting the artifacts in the ~~CM4PM~~MDMI standard, an overview and example of the use of the key artifacts in performing a conversion may be helpful.

Figure 7.1 and Figure 7.2 present an implementation of a conversion utilizing the key artifacts in the ~~Conversion Models for Payment Message Standards.~~MDMI Standard. The rectangles in the diagram represent these artifacts. In addition, it should be understood that the Business Elements in Figure 7.1 are the same Business Elements as in Figure 7.2 and ~~bundled together in some technologically appropriate way~~that these Business Elements are defined in a central dictionary.



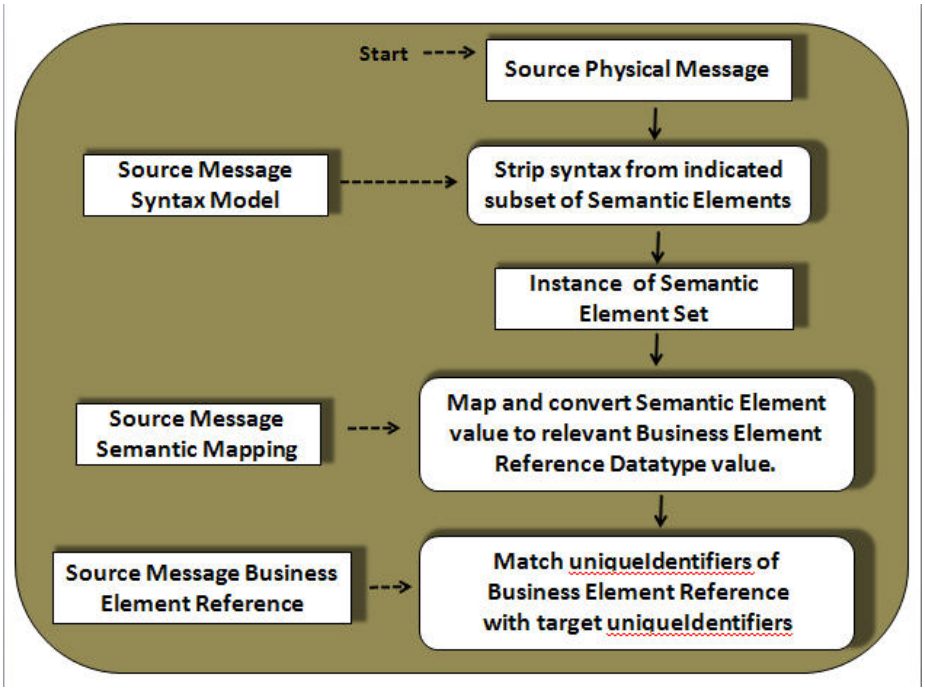



**Figure 7.1 - Overview of proposed run-time conversion methodology from Source ~~to UNIFI~~**

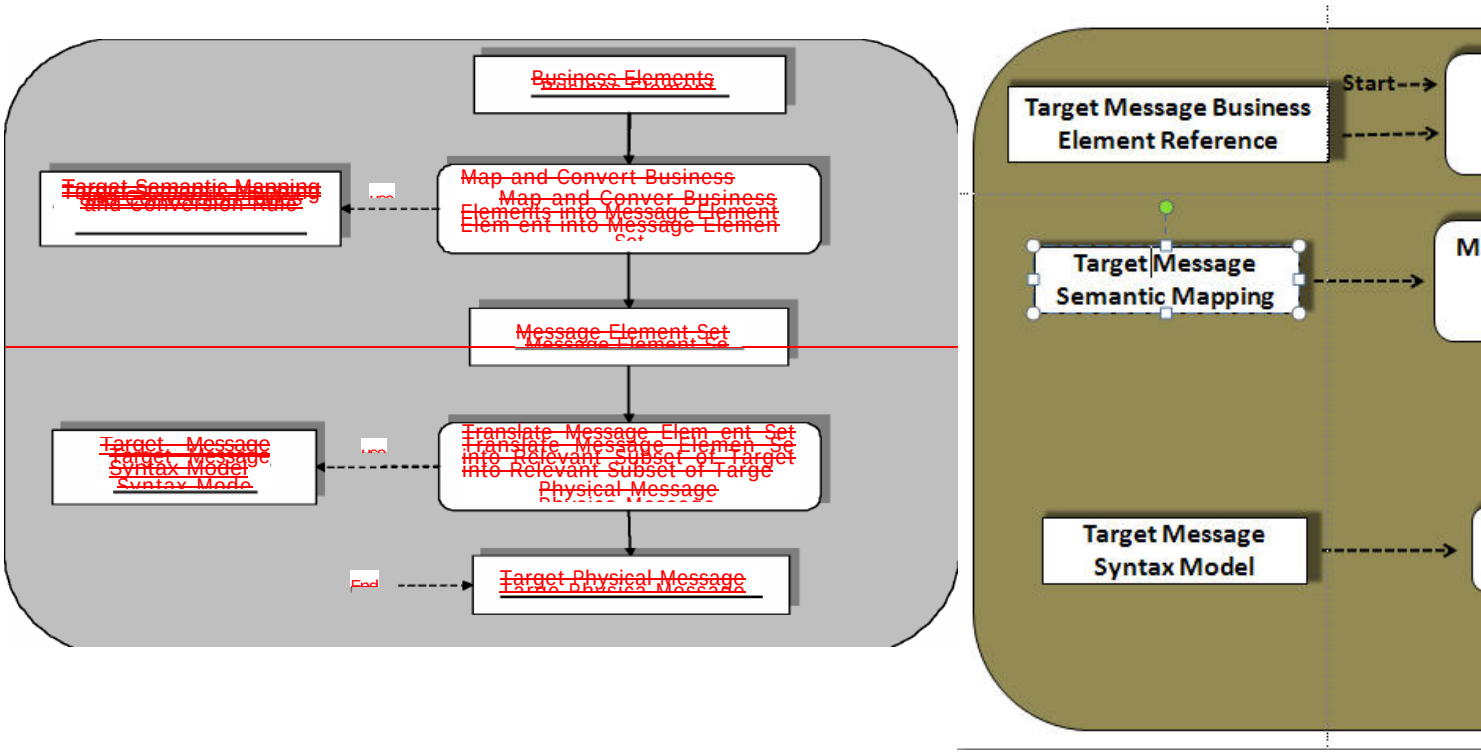



**Figure 7.2 - Overview of proposed run-time conversion ~~methodology UNIFI~~ to Target**

The following step descriptions annotate this conversion example.

### 7.1.1 Step 1 - Remove the Syntax

The first step of a conversion is to convert the targeted data in a physical message instance (e.g., a SWIFT ~~MT102~~MT103, a Visa TC05, etc.) from its existing format to a syntax-neutral format. The conversion involves the extraction of data from the existing Message using a syntax translation process. This process utilizes the ~~Conversion Models for Payment Message Standards~~MDMI Standard artifact, "Message Syntax Model." The Message Syntax Model provides a syntactic description that contains the necessary information to extract or insert any particular data item from/to a physical message instance.

A data item in a message is defined as the smallest semantic unit in a message for which further parsing would lose semantic meaning leaving only generic ~~Datatype values. Normally the smallest semantic unit is a "field."~~datatype values. For example, in a SWIFT MT102 there is a field representing a Settlement Date. If further parsing was done, the value left would simply be a date and indistinguishable, in a business semantic context, from any other date. Therefore, Settlement Date is a data item that is a smallest semantic unit. The data item "Settlement Date" has a datatype of date.

~~For example, in a SWIFT MT102 there is a field representing a Settlement Date. If further parsing was done, the value left would simply be a Date and indistinguishable, in a business semantic context, from any other date. Therefore the Settlement Date is~~ Normally the smallest semantic unit ~~whose Data type is date.~~

in a message is a field but in many overloaded message formats, a semantic unit can be a sub-field or a combination of fields. In existing message formats, ~~there are~~ many "fields" ~~that~~ have been subdivided into numerous semantic units. For example, a field may contain ~~both~~ a list of "Primary Account ID"s separated by commas. In that case, each

"primary account ~~balance" followed by a "balance currency code." In this case there would be two smallest semantic units~~ID" is a separate data item even though they appear in ~~the~~ one field.

When the data is ~~extracted from the physical~~stripped of its specific message format syntax, its value ~~is placed into~~ will be represented by an instance of the artifact "~~Message~~Semantic Element~~.~~"". There will be a ~~Message~~Semantic Element class defined for every semantic unit contained in ~~that~~ a message's message format. All of these the ~~Message~~Semantic Element classes ~~relate to~~are contained in the ~~artifact "Message~~"Semantic Element Set" by composition.

### 7.1.2 Step 2 - Mapping a Source ~~Message~~Semantic Element to ~~Business Element~~a Target Semantic Element through the use of a Unique Identifier acquired from a central dictionary

The second step for the conversion ~~uses~~leverages a ~~mapping process to convert the data. The data can be either directly moved into a Message Element Instance in the Message Element Set of another message format or, preferably, moved into an instance of a Business Element defined in the industry defined data~~central dictionary~~. Business~~ to define the relationship between a Source Semantic Element and one or more Target Semantic Elements.

The Source and Target Semantic Elements are ~~the smallest semantic units contained in an industry data~~associated with a central dictionary Business Element through a Business Element Reference class. That association may be a simple isomorphic mapping or it may involve a more complex map utilizing various artifacts in the MDMI specification such as ~~the ISO20022 UNIFI, which cover the business concepts~~a computed Semantic Element or a Conversion rule. Each element in the central dictionary has to provide a unique identifier for ~~that industry sector. (The~~ its Business Elements ~~described~~. That unique identifier will be stored in ~~this document are duplicates of~~ the Business Element ~~class definitions~~references that are associated with Semantic Elements. The appropriate Unique Identifiers will have been stored in the MDMI map for all Semantic Elements in ~~ISO 20022v2 compliant data dictionaries.)~~the both the Source and Target message formats.

~~In the case where~~An MDMI runtime application can locate a complete definition of a transformation by lining up the Source and Target maps by for the Semantic Elements that have matching Unique Identifiers.

However knowing the direct mapping instructions is often not enough information to insert a value into a Target message, as the validity of that insertion often depends on other Semantic Elements in a message. For example, it may be invalid to store a "Primary Account Balance amount" if there is no value for a "Primary Account ID." Therefore, the ~~mapping is done to an industry data dictionary, the mapping processor will utilize the artifact "Conversion Rule" to map data from the source message Element to a Business~~maps for each Semantic Element~~. In many cases, this mapping will amount to a simple isomorphic mapping. In this case the Message~~ include a set of Semantic Element ~~and Business~~Relationships that describe the relationship of a particular Semantic Element ~~are synonyms. In~~with all other ~~cases, simple transformations will be required, such as splitting or concatenating the value in the Message~~Semantic Elements in the message. A runtime application uses the Semantic Element~~. For example, a "Client Name" element may be parsed into two elements "Client GivenName," and "Client FamilyName." These straightforward semantic mappings define "near synonyms" between a Message Element and a set of Business Elements.~~

~~(If there is no simple or reasonable transformation to/from a Message Element to a Business Element, this probably indicates that a submission should be made to enhance a industry data dictionary such as UNIFI, as it reveals that a business concept is not adequately represented in that dictionary. )~~

### ~~7.1.3 Step 3 - Reversing the process~~

~~Given a value stored in a Business Element Instance, the above two steps are reversed, using the same artifacts with one modification. The mapping processor uses the "MessageElementRelationship"~~ Relationships in its ~~mapping. A MessageElementRelationship class describes dependencies between the target Message Element and other Message Element in the target message's MessageElementSet. The message processor uses the MessageElementRelationships~~ target mapping to make sure that no constraints are violated and that the inserted value is valid in relationship to other elements in the Message. ~~For example, it may be invalid to store an "account balance amount" if there is no "account ID."~~

### ~~7.1.4 Direct Bilateral Mapping~~

~~The CM4PM standard can be used to directly covert data from one message format to another message format. In this case the ConversionRules, which define the relationship between a MessageElement in the Source MessageElementSet and a MessageElement in the target Message Element Set must be as complicated as required by the mapping processor to accomplish a mapping.~~

### ~~7.1.5 Message Aggregates~~

~~A MessageSyntaxModel and its associated MessageElementSet comprise a MessageModel, which is complete, machine readable specification for a message format. In turn a set of MessageModels that are meant to cover a specific sector of Financial Service are associated with a MessageGroup. A MessageGroup follow the natural grouping of message formats, usually as defined by an industry standards body. For example, Visa has a set of retail payment messages referred to as TCxx messages, which would make up a MessageGroup. There will be a MessageModel, for every message format in that retail payment set of messages, for example a MessageModel for the TC05 message format.~~

# 8 UML Semantics - Normative Definition

The following is the formal Meta-Object Facility (MOF) model of the Conversion Models for Payment Messages Standards. It is first presented as a set of annotated views followed by the presentation of all the "elements" brought together in a single view.

## 8.1 MessageModels, MessageGroup, ~~MessagePackage~~ MDMIDictionaryReference

### 8.1.1 Overview

This view presents the MessageModel, the MessageGroup~~, the MessagePackage~~ and the ~~MessageModel.~~MDMIDictionaryReference. A MessageModel is a formal representation of a message format. A MessageGroup is composed of a set of Message Models that are usually grouped together because they focus on a particular messaging domain. ~~A MessagePackage is a formal structuring of the metadata for a particular MessageGroup, to be used by the conversion processors~~ For example, the set of SWIFT MTxx payment messages, the set of SWIFT MXxx fund messages, the set of Visa TCxx retail payment messages. An MDMIDictionaryReference provides a reference to the central dictionary to which the Semantic Elements for all MessageModels in the MessageGroup will be mapped.

~~For example:~~

- ~~The set of SWIFT MTxx payment messages~~
- ~~The set of SWIFT MXxx fund messages~~
- ~~The set of Visa TCxx retail payment messages~~

### 8.1.2 Abstract Syntax

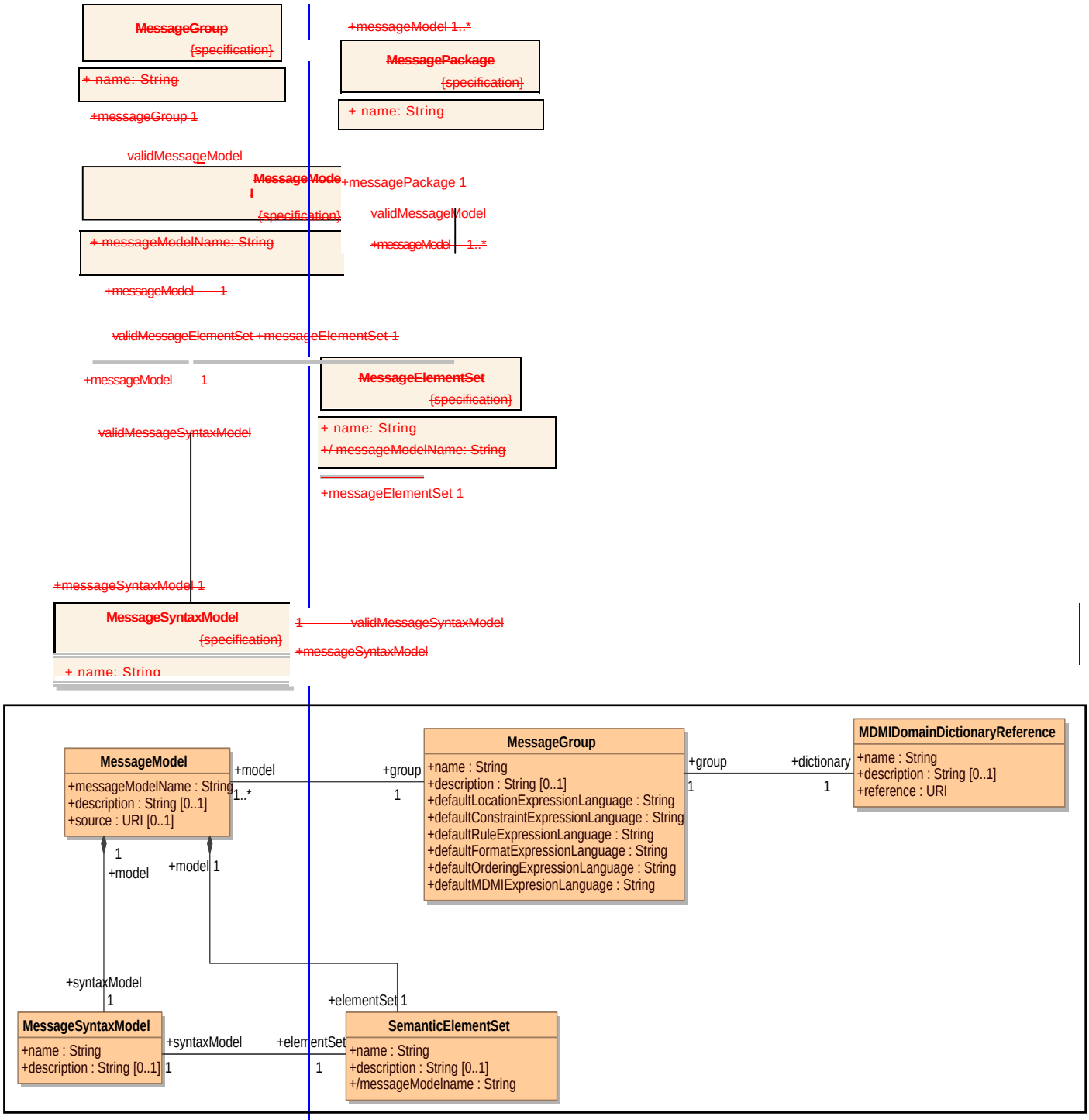



**Figure 8.1 - Message Model, MessageGroup, MDMIDictionaryReference**

### 8.1.3 MessageModel -- Detailed Semantics

A MessageModel has one propertydescription:

1. The property is "messageModelName." The messageModelName property is a string that is usually similar to the

nameThe MessageModel is the parent class that contains the MDMI model of a message format. The database schema of a record in a table can also be considered a message format as well as most XML documents.

MessageModel properties:

1. A "messageModelName" property, of type String, names the model of the message format it is modeling.being modeled. For example, the value of a messageModelName for a MT103 MessageModel could undoubtedly be "MT103."
2. This viewAn optional "description" property, of type String, contains a description of the message model.
3. A "source" is a property, of type URI, which contains a reference to the definition of the message format whose elements are being mapped. This reference can take many forms, for example, the reference might be to a machine-readable definition, such as the location of the message definition in the ISO 20022 repository, or it might reference a paper document.

MessageModel has four associations:

1. A MessageModel has a MessageSyntaxModel by composition.
2. A MessageModel has a MessageElementSetSemanticElementSet by composition. Together the MessageModel's composed MessageSyntaxModel and the MessageElementSet uniquely define a message format.
3. A MessageModel is associated with a MessageGroup.

A Message model is associated, by composition, with a MessagePackage.

### 8.1.4 MessageGroup - Detailed ~~semantics~~Semantics

~~A~~ MessageGroup ~~has one property~~description:

~~1.~~ The ~~property is "name." The value of name is the name of the~~ MessageGroup~~. A~~ class contains a set of message models that are considered in the same grouping, e.g., SWIFT MX messages, SWIFT 15022 messages, FIX security messages, etc. The MessageGroup ~~has one~~class is useful for setting various defaults for closely related message formats.

The MessageGroup properties:

1. The property "name" of type String, names the MessageGroup.
2. The optional property "description", of type String, provides a description of MessageGroup.
3. The property "defaultLocationExpressionLanguage" of type String identifies the location language to be used as a default for specifying location for all the messages in the MessageGroup. The value must be recognized by a runtime transformation application. The location of any field or sub-field in a message must be expressible in the chosen locationExpressionLanguage. For example, a location language for an XML message format would be "XPath 2.0".
4. The property "defaultConstraintExpressionLanguage" of type String identifies the constraint language to be used as a default for specifying the constraints in the Choice class for all the messages in the MessageGroup. The constraintExpressionLanguage must be able to reference nodes. An appropriate language, which has been used in an example implementation, is NRL 1.0.
5. The property "defaultRuleExpressionLanguage" of type String identifies the rule language to be used as a default for specifying rules in all classes with the property "rule" for all the messages in the MessageGroup. This rule language must be able to access the values of any SemanticElement and thus it must be able to access the fields in complex datatypes. An appropriate language, which has been used in an example implementation, is NRL 1.0[C46].
6. The property "defaultFormatExpressionLanguage", of type String, identifies the format language to be used as a default for specifying formats in the LeafSyntaxTranslator class for all the messages in the MessageGroup. The formatExpressionLanguage must be able to describe fields as well as sub-fields, in particular the proper termination character for a field or sub-field. Appropriate languages, which have been used in an example implementation, are the SWIFT 150022 regular expression format language and XSD format attributes.
7. The property "defaultOrderingExpressionLanguage", of type String, identifies the ordering language to be used as a default for specifying the ordering of multiple instances of Semantic Elements in which the Boolean property "mutipleInstances" is "True". The ordering language should provide expressions evaluate to both cardinal and ordinal positioning. For example, NRL is a language that can be used to specify ordering.
8. The property "defaultMDMIExpressionLanguage", of type String, identifies the computational language to be used as a default for specifying the computational expression in computed Semantic Elements that are of type MDMIExpression. For example, NRL, with its declarative and action language, can be used as a MDMI Expression Language.

MessageGroup associations:

An association~~.~~

1. ~~1. A MessageGroup has~~ with one or more MessageModels, which comprise the MessageGroup ~~. There are various reasons to group MessageModels~~

### ~~8.1.5 MessagePackage - Detailed Semantics~~

~~A MessagePackage has one property~~

~~1. The property is "name." The value of name is the name of the MessagePackage.~~

~~A MessagePackage has one association.~~

~~1. A MessagePackage has, by composition, one or more MessageModels, which comprise the MessagePackage.~~

2. ~~8.2~~ An association with zero or more DataRules that are utilized by the Message models within the group.
3. An association with the MDMIDictionaryReference that identifies the central dictionary utilized by the group

### 8.1.5 MDMIDomainDictionaryReference

MDMIDomainDictionaryReference description:

The MDMIDomainDictionaryReference class provides a reference to the central dictionary that contains the Business Elements to which the Semantic Elements in the MessageModels in the MessageGroup are mapped. This class is purely informational as the URI reference to the dictionary does not have be machine-readable. The dictionary could reside on paper, for example. However, there must be a function or method associated with the dictionary that will provide: 1) a uniqueIdentifier for all Business Elements, and 2) a reference to a datatype that is compatible with the set of MDMIDatatype.

MDMIDomainDictionaryReference properties:

1. A "name" property, of type string, that provides a name for the referenced central dictionary.
2. An optional "description" property, of type String, that provides a description of the referenced central dictionary.
3. A "reference" property, of the type URI, that provides a reference to the central dictionary, such as a URL.

MDMIDomainDictionaryReference associations:

1. MDMIDomainDictionaryReference has a one-to-one association with MesssageGroup to indicate the central dictionary that will be used for the maps in MessageModels in the MessageGroup.
2. MDMIDomainDictionaryReference has a one-to-many relationship to the MDMIBusinessElementReference class so that a reference to the parent dictionary, to which a Business Element belongs, is easily found.

## 8.2 MessageSyntaxModel, Node, ~~Set, SetChoice~~Bag, Choice, LeafSyntaxTranslator

### 8.2.1 Overview

The MessageSyntaxModel and related classes ~~provides~~ provide syntax information that will ~~allow a Message Syntax Translator~~enable a process to either extract or insert a data value into or from an instance of a message. It does this by providing a description of the location and format of every ~~semantic unit~~Semantic Element in the message format.

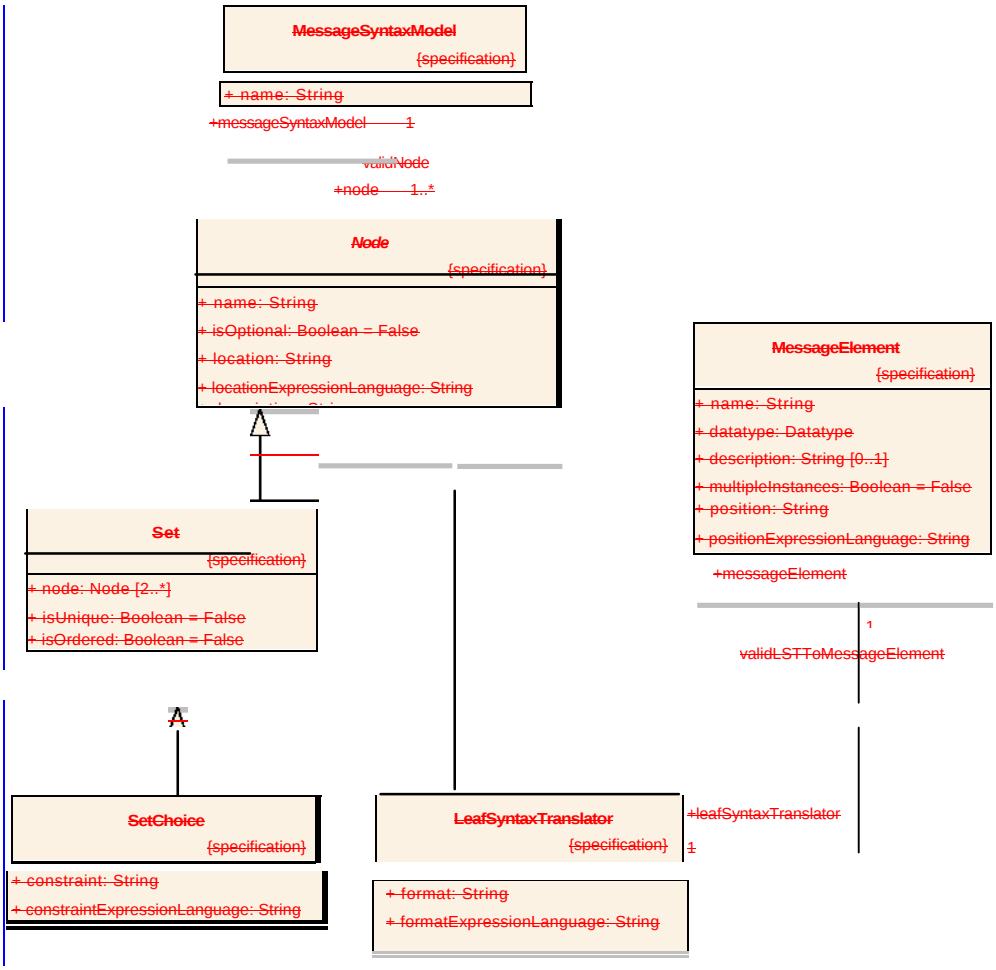
The MessageSyntaxModel class ~~acts as~~is the root of ~~a~~ the syntax tree. The syntax tree provides a map for navigating a message format. ~~Each leaf~~ The leafs of the tree~~, i.e., a LeafSyntaxTranslator, is associated with a Message Element (the semantic units in the message format). The~~ are LeafSyntaxTranslator nodes. The LeafSyntaxTranslator has location and format properties, which ~~contains the~~ contain information that defines how to move a data item from/to ~~the value property~~an instance of a ~~Message~~message and associate the data item with a Semantic Element ~~Instance to/from the location and format for that value in a physical message.~~. The ~~CM4PM~~MDMI standard does not require a specific language ~~for the properties in the LeafSyntaxTranslator, which describe a location or a format. Instead, properties are defined that reference the expression language that will be used~~ to describe ~~location and format.~~a location or a format for

the properties in the LeafSyntaxTranslator. Instead, language properties are included that provide a reference the expression language that will be used to describe location and format. This flexibility was chosen given the variety of different types of message formats ~~–~~ for example: XML, EDIFACT, Object models, etc., ~~which must be accommodated~~and the legacy languages already out there to express location and format.

The other classes associated with the MessageSyntaxModel are used to construct the branches of the syntax tree. They are:

- *Node* ~~–~~ an abstract class that represents the branches and leaf nodes of the syntax tree
- ~~Set –~~Bag – a branch Node that identifies a set of Nodes that are aggregated in a message format
- ~~SetChoice –~~Choice – a branch Node that defines rules to identify the conditions for which ~~elements~~values in ~~a set~~its children nodes should appear in a physical message instance.

### 8.2.2 Abstract Syntax



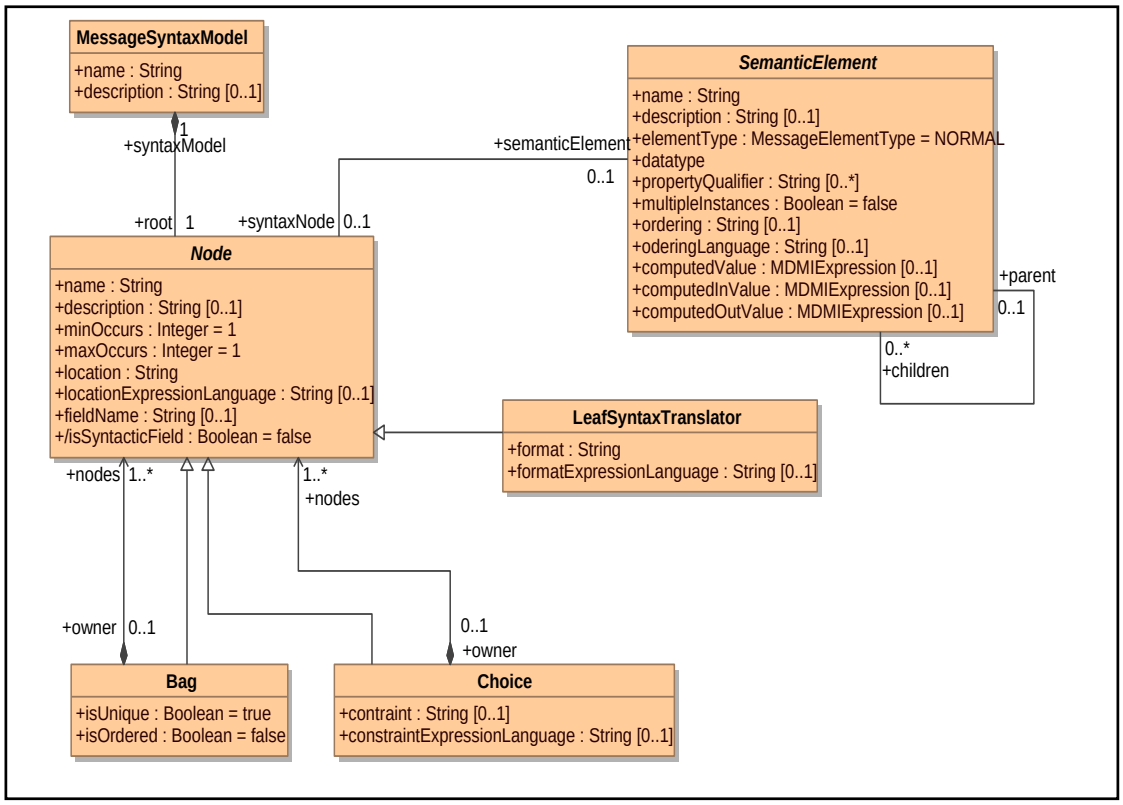



**Figure 8.2 - Message Syntax Model**

### 8.2.3 MessageSyntaxModel ~~--~~ Detailed Semantics

~~A~~ MessageSyntaxModel ~~class has one property~~description:

> ~~1.~~ The MessageSyntaxModel contains a syntax tree that describes how each Semantic Element can be either inserted into or extracted from a message based on that message's message format.

MessageSyntaxModel properties:

1. A "name" property, ~~whose value~~ of type String, is the name of the MessageSyntaxModel~~—~~. This name will often be similar to the MessageModel name, e.g., "MT103 Syntax Tree."

~~This view of the MessageSyntaxModel has one association:~~

2. ~~1. One-to-many Nodes can be associated with a~~ The optional property "description" of type String provides a description of MessageGroup.

MessageSyntaxModel associations:

1. An associations with one-to-many Nodes as it is the parent class of the syntax tree.
2. An association with its parent MesssageModel
3. An association with its sibling SemanticElement Set

### 8.2.4 Node - Detailed Semantics

~~A Node class has five properties:~~

~~1. A~~*Node* description:

The *Node* class is an abstract class that is inherited by all nodes in the syntax tree. It primarily contains location information so that any field or data item in a message can be located.

*Node* properties:

1. The "name" property, ~~whose value is~~ of type String, provides a name for the Node~~—~~. This name can be useful to label a section or element in a message format. The name property is important because it should provide ~~a~~an addressable reference to ~~a node for the~~ the node, which can be used in an expression~~ languages used in the syntax tree~~.

2. ~~An "~~The optional~~"~~ "description" property, ~~whose value is a Boolean true or false -- If the optional~~ of type String, describes the *Node*'s purpose.
3. The "minOccurs" property ~~is true, then the node can optionally appear in a physical message. If the property is false, then the node~~, of type Integer, has a value of 0..1. The value of "0" indicates that the *Node* is optional whereas the value "1" indicates that the *Node* is required ~~in the physical message~~.
4. An optional "maxOccurs" property, of type Integer, puts an upper limit on the number of instances allowed for the node.
5. A "location" property ~~whose value~~, of type String, describes the location of the Node in the physical message. The location is often in reference to, or anchored by, the URI that defines the location of the physical message instance.
6. A "locationExpressionLanguage" property ~~whose value~~, of type String, defines a reference to the expression language used in the location property. The ~~language used~~ locationExpressionLanguage must ~~have a reference. The language may be a formally defined language, such as XPath, or a simple local language, such as a "byte-count and length."~~satisfy the same constraints described for the "defaultLocationExpressionLanguage in section 8.1.5.
7. ~~A "description~~An optional "fieldname" property, ~~whose value is a string containing a description of the node. This view of~~ of type String, provides the field name of a simple datatype that is part of a complex MDMIDatatype. The data item, whose location is indicated by the Node ~~shows two~~, has the datatype associated with the "fieldname".
8. A derived property "isSyntaticField", of type Boolean, indicates, if the property's value is "True", that this node corresponds to a data item that is part of an MDMIComplexDatatype. "isSyntaxField" will be "True" if the optional "fieldname" is present.

*Node* class generalizations:

Three classes ~~that~~ inherit from the *Node* ~~Set~~abstract class: Bag, Choice and LeafSyntaxTranslator.

*Node* class associations

1. Node has a many-to-one association with the Bag class as a Bag can have Node children.
2. Node has a many-to-one association with the Choice class as a Choice can have Node children.
3. Node has a one-to-one relationship with a SemanticElement. This is the key association that links a SemanticElement to its syntax.

### 8.2.5 ~~Set~~ Bag - Detailed Semantics

~~A Set class has three additional~~Bag description:

The Bag class represents a set of syntax nodes. The nodes in a Bag can be a unique set or a bag, and the nodes can be ordered or unordered.

Bag properties:

~~A "node" property, whose value is an association of two-to-many Nodes -- The location and locationExpressionLanguage properties of a Set must indicate the mechanism for identifying the beginning of the Set, the end of the Set, and the separator between Nodes in the message format.~~

1. ~~An~~ The "isUnique" ~~Boolean~~ property ~~whose~~, of type Boolean, indicates, if its value is ~~true if~~ "True", that the bag is a set ~~is~~ composed of unique items ~~and false if the set~~. If its value is "False", the bag of nodes can contain duplicates~~.~~.
2. ~~An~~The "isOrdered" ~~Boolean~~ property ~~whose~~, of type Boolean indicates, if its value is ~~true if~~ "True" that the nodes in the bag must be in an ordered sequence ~~and false if the set~~. If the value is "False", the nodes in the bag

can be unordered. This ~~view of the Set shows~~property is useful for parsing a message. The actual ordering of SemanticElements is handled 1) using the "location" property in the *Node* class ~~that inherits from Set – SetChoice~~and 2) using the "ordering" property in the SemanticElement class.

Bag associations:

1. The Bag class has a one-to-many association with some other classes that inherits from Node. Thus, it becomes a branch in the syntax tree. Since it must have at least one association with another class by composition, it cannot be a leaf of the syntax tree.

### 8.2.6 ~~SetChoice~~ Choice - Detailed Semantics

~~The SetChoice class has two additional properties beyond the properties inherited from Node and Set.~~

Choice description:

The Choice class contains the conditions that can identify the subset of its children nodes that will be present in a particular message instance. The subset is determined by a constraint expression.

Choice properties:

1. A "constraint" property whose value is an expression that can be used to determine which of the set of nodes should be in a physical message instance.
2. ~~A~~An optional "constraintExpressionLanguage~~" property whose value references~~ ", of type String .that is a reference to the language used in the ~~SetChoice~~ "constraint" property. The constraintExpressionLanguage must be able to reference ~~nodes~~any node in the syntax tree.

Choice associations:

1. The Choice class has a one-to-many association with some other class that inherits from *Node*. Thus, it becomes a branch in the syntax tree. Since it must have at least one association with another class by composition, it cannot be a leaf of the syntax tree.

### 8.2.7 LeafSyntaxTranslator

~~The~~ LeafSyntaxTranslator ~~class has two~~description:

The LeafSyntaxTranslator class is represents a leaf of the syntax tree. There is a LeafSyntaxTranslator corresponding to every field, sub-field or data item in the message format. The LeafSyntaxTranslator inherits location information from the *Node* and has additional properties ~~in addition to those it inherits from Node~~that describe the format of the data item with which it is associated.

LeafSyntaxTranslator properties:

1. The "format" property, ~~whose value describes~~of type String, provides the specific format of a field or subfield in the ~~semantic unit in a physical~~ message ~~instance that is associated with a Message Element.~~format.

2. The "formatExpressionLanguage" property, ~~whose value~~of type String, is a reference to the expression language used in the format property~~ – There are a number of good choices for the formatExpressionLanguage values. –~~. For example, SWIFT has a defined regular expression language for the format of ~~its semantic units.~~fields in MT messages. The formatExpressionLanguage must be able to reference and fully describe the format of data item. An example would be being able to specify the proper termination character for list of fields that occur within a string. While the ~~CM4PM~~MDMI standard does not require a specific formatExpressionLanguage, if no formatExpressionLanguage exists for a particular message format, the ~~CM4PM~~MDMI standard is recommending the use of a small subset of DFDL as a general solution.

## 8.3 ~~MessageElementSet, MessageElement,~~ SemanticElementSet, SemanticElement SimpleMessageComposite, MessageComposite, Keyword

### 8.3.1 Overview

The ~~MessageElementSet~~SemanticElementSet contains a set of ~~Message~~Semantic Element classes. Each ~~MessageElement~~SemanticElement represents a smallest semantic unit in a message format. The ~~MessageElementSet~~SemanticElementSet and the MessageSyntaxModel, which are the two entities that comprise a Message model, can provide a complete specification of a message format. ~~This is because~~ If all the ~~semantic units~~Semantic Elements in a message are stored in the ~~MessageElementSet~~SemanticElementSet and instructions on how to insert or extract each of those elements are contained in the MessageSyntaxModel~~.~~, then a complete model of a message format will be created. However, one of the advantages of MDMI is subsets of a message format can also be mapped. For example, given a specification such as RosettaNet and a goal of executing a payment, only the payment data-items that are to be moved into a SWIFT payment message need to be mapped.

The ~~MessageElementSet~~SemanticElementSet represents the "flattening" or the "linearization" of a message format. This flattening is important, since a primary goal of ~~CM4PM~~MDMI is to expedite the insertion or extraction of as little as one semantic unit of a message. For processing efficiency, it is very important that the information needed to convert one item from/to a message does not require complete information about the structure of the entire message format.

The primary constituents of the ~~MessageElementSet~~SemanticElementSet are ~~Message~~Semantic Elements. A couple of additional classes are provided primarily for the ease of the designer, but they do not play a major role in the conversion process. These are SimpleMessageComposites and MessageComposites. These classes are conveniences for bundling ~~MessageElements~~SemanticElements in the design process.

A SimpleMessageComposite is an "aggregation" that only contains ~~MessageElements.~~SemanticElements. It is important, as this first level of aggregation is a very common design mechanism.

~~For example:~~

~~An "account owner address" in a message format may be defined to be composed of separate fields (or semantic units) for "account owner street," "account owner city," "account owner state," etc. By definition, each one of these individual fields will be a separate MessageElement, e.g., there will be a MessageElement for AccountOwnerStreet, a MessageElement for AccountOwnerCity, and a MessageElement for AccountOwnerState and so on. However, at design time it may be more convenient to deal with the aggregation of the fields and have a SimpleMessageComposite that represents an AccountOwnerAddress.~~

A MessageComposite is an aggregation that contains ~~MessageElements~~SemanticElements, SimpleMessageComposites and MessageComposites. It is possible therefore to create exceedingly complicated MessageComposite structures. However, these structuring mechanisms should be used with considerable caution. Such complicated structures are far away from the desired linearization or flattening of semantic units, which is a core design principle of the ~~CM4PM~~MDMI standard.

An important property of ~~MessageElements~~SemanticElements merits further discussion. This is the ~~attribute~~property "multipleInstances." MultipleInstances indicates that ~~the semantic unit represented by that MessageElement~~instances of a particular SemanticElement can appear multiple times in a physical message instance~~. In its simplest form, a MessageElement, which has a value of true for its multipleInstance property,~~, usually in the form of repeating fields or a list. In effect, the SemanticElement is a vector and not a singular value. As expected, the fact that SemanticElements can ~~simply represent~~ be an array ~~or list — for example, a list of codes, a list~~ of ~~numeric entries, and so on. However, a MessageElement marked with "multipleInstances" can also be a component of a more complex structure. Thus in the above example of AccountOwnerAddress, if the message format allowed multiple account owner addresses, each MessageElement in~~values increases the ~~address, such as "AccountOwnerStreet" would have its multipleInstances property~~

~~as true. It would be up to~~ complexity of the ~~"location" property in the LeafSyntaxTranslator that is associated "AccountOwnerStreet" to know where each of the instances of AccountOwnerStreet should be placed so they go into or can be retrieved from the right address structure in a message. (Even though AccountOwnerAddress may be a SimpleMessageComposite, at run time each Message Element of that composite is handled independently, i.e., the SimpleMessageComposite is only a design time convenience.)~~model

**8.3.**

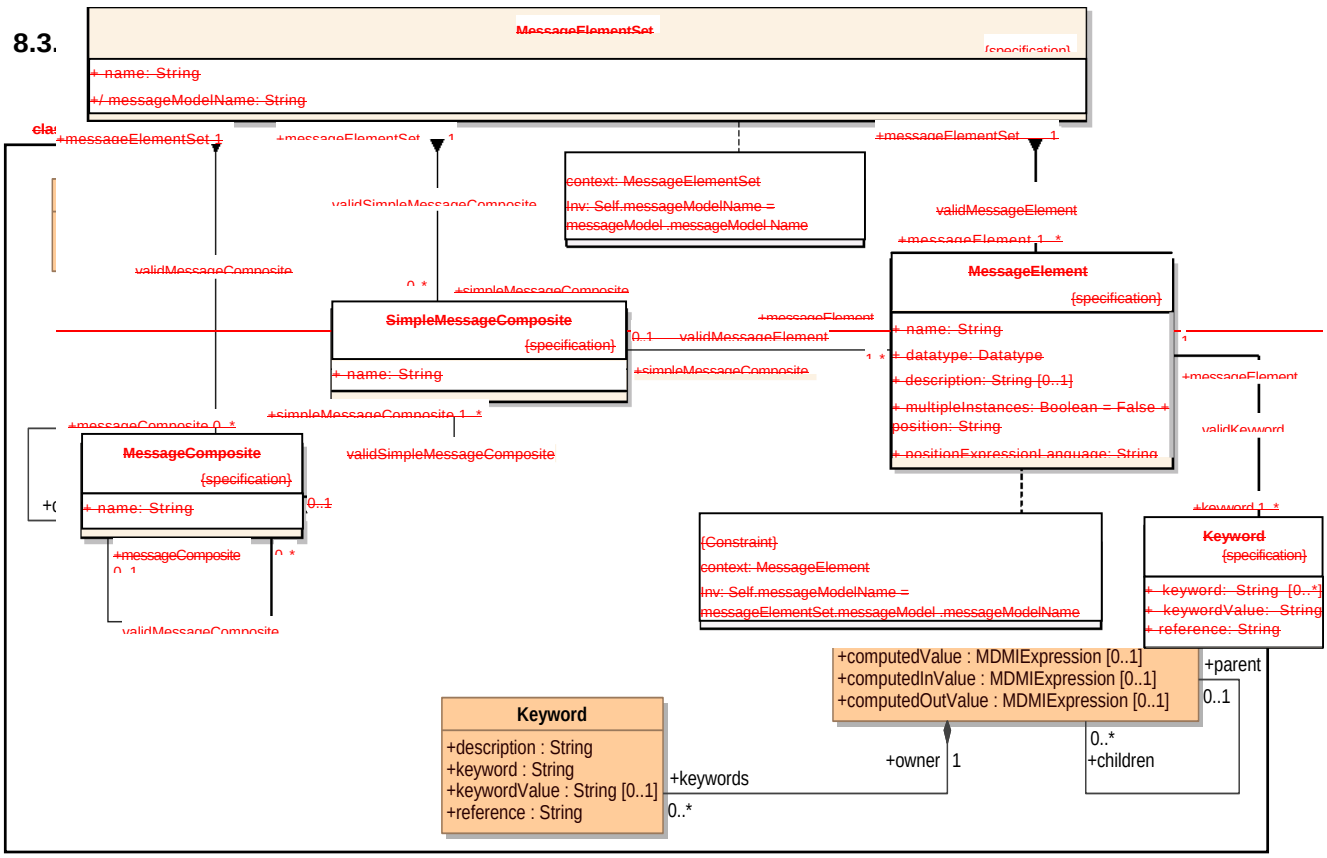


**Figure 8.3 - ~~Message Element Set~~ SemanticElementSet and associated classes**

### 8.3.3 ~~MessageElementSet --~~ SemanticElementSet - Detailed Semantics

~~The MessageElement~~SemanticElement Set description:

The SemanticElement Set contains the smallest Semantic Elements contained in a message format. The set only holds Semantic Elements. All of the message-specific syntax of selected elements from a particular message format has ~~two~~been removed.

SemanticElementSet properties:

A "name" property ~~whose value is~~, of type String, contains the name of the ~~MessageElementSet.~~

~~A "MessageModelName" property, whose value is constrained to be the same as the name property in the MessageModel that contains the MessageElementSet.~~

~~This view of the MessageElementSet shows three associations:~~

1. ~~The MessageElement Set will contain, by composition, one-to-many MessageElements. A MessageElement is the primary entity in the MessageElementSet~~SemanticElementSet.

The ~~MessageElementSet can contain, by composition, zero-to-many SimpleMessageComposites. A SimpleMessageComposite, is a convenient mechanism for grouping MessageElements.~~

~~The MessageElementSet can contain, by composition, zero-to-many MessageComposites. A MessageComposite is a convenient mechanism for grouping MessageComposites, SimpleMessageComposite, and MessageElements.~~

### ~~8.3.4 MessageElement - Detailed Semantics~~

~~The MessageElement class has seven properties:~~

~~1. A "name" property, whose value is the name of the MessageElement.~~

~~A "datatype" property, whose value is associated with a Datatype.~~

2. ~~An~~ optional "description~~"~~" property, ~~whose value is~~of type String, provides a ~~string that describes~~description of the ~~MessageElement~~SemanticElement Set.
3. The derived "MessageModelName" property, of type string, contains the name of the MessageModel to which theSemanticElementSet belongs. This derived property is included for implementation convenience.

SemanticElementSet associations:

1. The SemanticElementSet has a one-to-many association by composition to SemanticElements.
2. The SemanticElementSet has a zero-to-many association with SimpleMessageComposites. A SimpleMessageComposite is a convenient mechanism for grouping SemanticElements.
3. The SemanticElementSet has a one-to-one relationship to its parent MessageModel.
4. The SemanticElementSet has a one-to-one relationship to its sibling, the MessageSyntaxModel.

### 8.3.4 SemanticElement - Detailed Semantics

SemanticElement description:

The SemanticElement class is the core of the MDMI message map. SemanticElements represent the smallest semantic units in a message format, stripped of any complicating syntax considerations. Each SemanticElement is unique in the context of its message format, i.e., it must have an individual semantic meaning. As example, "address" cannot be a SemanticElement; "address" is a datatype that can be repeated in many message fields. "Primary Debtor Address" is a SemanticElement as it refers to a particular unique address in a message format.

The SemanticElement properties:

1. A "name" property, of type String, contains the name of the SemanticElement.
2. The optional "description" property, of type String, contains a description of the SemanticElement.
3. An "elementType" property, of the enumerated type MessageElementType, can have three values each of which defines the type of Semantic Element.
    - NORMAL – a "NORMAL" Semantic Element is equivalent to the current definition of a SemanticElement, i.e., a Semantic Element, contained in a message format, which is to be mapped to a central dictionary.
    - LOCAL – a "LOCAL Semantic Element contains some technical information that is needed to correctly map NORMAL Semantic Elements, e.g., it may contain an index that is used to provide the ordering for a child Semantic Element that has multiple instances.
    - COMPUTED – a "COMPUTED" Semantic Element is to be mapped to the central dictionary but contains a value that is not directly contained in a message. Instead, a "COMPUTED" Semantic Element's value is computed using.
4. A "datatype" property, of type MDMIDatatype, defines the simple or complex datatype of the Semantic Element.
5. A zero-to-many "propertyQualifier" property, of type String, is a list of keywords that contains reference keywords of interest that are associated with the message format, such as a "tag" associated with a SemanticElement.
6. A "multipleInstances" property, of type Boolean ~~property, whose value~~, which if true indicates that instances of this ~~MessageElement~~SemanticElement can be repeated in a physical message ~~instance~~ as a list or array. ~~The~~

multipleInstances property, if true, also can indicate that this MessageElement can be repeated as part of a more complex structure defined in that message's message format.

7. A "positionAn "ordering" property, whose valueof type String, contains an expression that describes how the position or index of an instance of the MessageElement in the set of multipleSemantic Element instances are ordered, if that MessageElement'sthe SemanticElement's multipleInstances property is not false."True".
8. A "positionExpressionLanguageAn optional "orderingExpressionLanguage" property, whose valueof type String, that is a reference to the expression language used for the value of the "positionordering" property. The flexibility of a "positionExpressionLanguage" is needed as the position property needs to be richer than an integer index. For example, it might include key words such as "first," "last," or "next." The position attribute might also be a calculated variable as opposed to an integer constant. The ordering language must be able to describe ordinal and cardinal positioning as well as expressions that when evaluated will provide an index. As an example, a language that can be used is NRL 1.0.
9. A "MessageModelName" derivedcomputedValue" property, whose of type MDMIexpression, contains an expression that computes the value is constrained to be the same asfor the "name" SemanticElement. The expression can refer to the value of other SemanticElements. This property in the MessageModel in whichis most often used for SemanticElements of the type LOCAL.
10. A "computedInValue" property, of type MDMIexpression, contains an expression to compute a value for the SemanticElement when it is containeda target, based on the values of one or more BusinessElements and SemanticElements. The value when it is a source is directly mapped.
11. This view of MessageElement has three A "computedOutValue" property, of MDMIexpression, contains an expression to computes value for a SemanticElement, when it is a source, based on the values of one or more SemanticElements. The value when it is a target is directly mapped.

The SemanticElement associations:

1. A one-to-one association with a keyword list that is used to identify the MesssageElement for searches and can be associated with a formal ontology.many association with any children through a parent association. This allows the SemanticElementSet to include container Semantic Elements, which are identified by "parent". Explicit container Semantic Elements allow the hierarchical structure of a message format to be maintained in the SemanticElementSet. In the case where a container SemanticElement has no message-based properties itself, that container should be of type Computed with a simple index as the computed value.

A one-to-many association, by composition, with the MessageElementSet.

A one-to-many association with a SimpleMessageComposite.

### 8.3.5 Keyword - Detailed Semantics

The Keyword class has three properties.

A "keyword" property and a "keywordValue" property, whose value is a list of keyword value pairs used to categorize this MessageElement.

An optional reference that identifies the origin set for the keywords, for example a formal ontology.

### 8.3.6 SimpleMessageComposite - Detailed Semantics

The SimpleMessageComposite has one property:

1. A "name" property, whose value is the name of the SimpleMessageComposite.

This view of the SimpleMessageComposite shows two associations:

2. A zero-to-many association~~, by composition, with a MessageElementSet~~ to the SemanticElementRelationship class. The SemanticElementRelationship provide the valid context for each SemanticElement.

A ~~zero-or-~~one ~~association with MessageElements.~~

### 8.3.7 MessageComposite - Detailed Semantics

~~The MessageComposite has one property:~~

~~1. A "name" property whose value is the name of the MessageComposite. This view of the MessageComposite shows three associations:~~

~~A zero-to-manyassociation, by composition, with a MessageElementSet.~~

~~A zero-or-one association with SimpleMessageComposites.~~

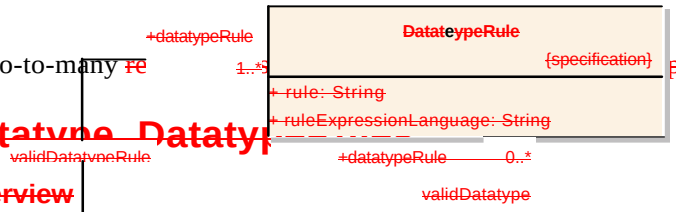
MessageComposite description:

The MessageComposite class inherits from the SimpleMessageComposite class, allowing the construction of a complex object tree. MessageComposite are an informative artifact that can be useful when there is a desire to associate SemanticElements with a complex object model.

MessageComposite associations:

A zero to many association with other MessageComposites that are the children of .the MessageComposite, thus providing a mechanism to specify a tree of MessageComposites.

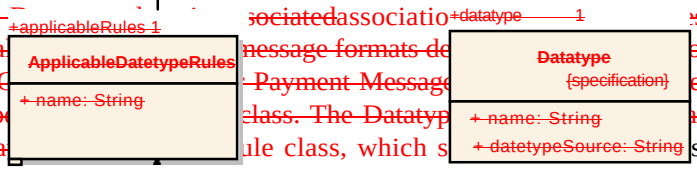
.




A zero-to-many ~~re~~ ~~posites.~~

## ~~8.4 Datatype, Datatyp~~

### ~~8.4.1 Overview~~




~~The~~ ~~sociated~~association ~~s allowed for a particular message format. They are usua~~ ~~nessage formats de~~ ~~oup. These standard Datatypes are not considered part of the C~~ ~~Payment Message~~ ~~ence for the data type definition source is included as a prop~~ ~~class. The Datatyp~~ ~~ata type of the value in a MessageElement Instance. A Data~~ ~~ule class, which s~~ s ~~associated with it. These rules are defined by the class, DatatypeRules. A set of the rules will~~ that apply to ~~each MessageElement, that set is defined by the in the class, ApplicableDataRules. DataRules are used on extraction for~~

3. ~~+applicableRules 1 +~~the datatype ~~1~~ of the SemanticElement.

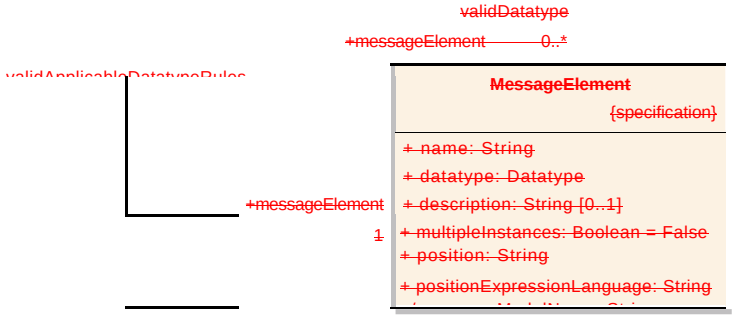



**Figure 8.4 - Datatype Rules, DataType**

### 8.4.3 Datatype - Detailed Semantics

A Datatype has two properties:

A "name" property whose value is the standard name of the data type it represents.

A "DatatypeSource" property whose value is a reference to a source that defines the named data type.

This view of Datatype shows two associations:

4. A Datatype is associated with from zero-to-many MessageElements, as the value in zero to many MessageElementsassociation with a keyword list, which can be used to identify the SemanticElement for searches and which can be associated with a particular Datatype. For example, the Datatype named "integer" will be associated with many MessageElements. (There is of course, the constraint that the Datatype property in a MessageElement be associated with one and only one Datatype class.)formal ontology.

A Datatype can be associated with zero to many DatatypeRules, which can be executed to make sure that a value associated with that Datatype is valid.

### 8.4.4 DatatypeRules -- Detailed Semantics

DatatypeRules have two properties:

5. A "rule" property A zero-to-many association with a SemanticElementBusinessRule, which provides for a specific set of rules that holds should apply to the value of the SemanticElement.
6. A one-to-many association with the ToBusinessElement class that describes the conversion of the value of the SemanticElement to conform to the reference value of the business element referenced by the MDMIBusinessElementReference class.
7. A one-to-many association with the ToSemanticElement Semantic class that describes the conversion of the reference value of the business element referenced by the MDMIBusinessElementReference class to the value of the SemanticElement.

### 8.3.5 Keyword - Detailed Semantics

Keyword description:

The keyword class contains either a keyword or a keyword/value pair. The set of Keywords can be used to profile a SemanticElement, to provide a mechanism to search for a SemanticElement, and to associate a SemanticElement with an external ontology or taxonomy.

Keyword properties:

1. The optional "description" property, of type string, describes the Keyword and/or the set of Keyword associated with a SemanticElement.
2. A "keyword" property, of type String, used to describe or profile a SemanticElement.
3. An optional "keywordValue", of type string, that is associated with the keyword creating a keyword/value pair.
4. An optional reference, of type String, identifies the origin set for the keywords, for example a formal ontology.

Keyword associations:

1. An optional many-to-one association with the SemanticElement it is describing.

### 8.3.6 SimpleMessageComposite - Detailed Semantics

SimpleMessageComposite description:

SimpleMessageComposite represent aggregations of SemanticElements. SimpleMessageComposite is an informative artifact that can be useful when a group of SemanticElements are associated with a class in an object model. Usually the attributes of an object will be equivalent to a SemanticElement and the object itself equivalent to a SimpleMessageComposite.

SimpleMessageComposite properties:

1. A “name” property, of type String, names the SimpleMessageComposite.
2. An optional “description” property, of type String, describes SimpleMessageComposite.

SimpleMessageComposite generalization:

MessageComposite inherits from SimpleMessageComposite.

SimpleMessageComposite associations:

1. A zero-to-many association with a SemanticElementSet by composition.
2. A (zero or one)-to-many association with SemanticElements.

### 8.3.7 MessageComposite -- Detailed Semantics

MessageComposite description:

The MessageComposite class inherits from the SimpleMessageComposite class, allowing the construction of a complex object tree. MessageComposite are an informative artifact that can be useful when there is a desire to associate SemanticElements with a complex object model.

MessageComposite properties:

1. The optional “owner” property” of type MessageComposite identifies the parent of a MessageComposite or SimpleMessageComposite.
2. The zero-to-many “composites” property of type MessageComposite identifies children[C77] MessageComposites of this MessageComposite.

MessageComposite associations:

Through inheritance, MessageComposite will have the same associations as SimpleMessageComposite.

## 8.4 MDMIDatatype, DataRules

### 8.4.1 Overview

The MDMIDatatype references a datatype used in the model. These MDMIDatatypes are not considered part of the MDMI standard. While the specification does not deal with datatypes directly, some restrictions on MDMIDatatype definitions are necessary for syntactic modeling and to ensure that a runtime engine will do proper transformations. These restrictions include: 1) that the simple datatypes be from a known standard, such as the XML simple datatypes. 2) that complex datatypes are ultimately composed of simple datatypes and that every simple datatypes has an identified “fieldname”. Associated with any value can be DataRules that describe constraints for that datatype, e.g., a zip code value must be in a table of legal zip codes. DataRules must be written in an appropriate Rule Expression Language that can access the components of a complex MDMIDatatype using “fieldnames”.[C79]

### 8.4.2 An example of Complex Datatype

A Semantic Element can be composed of complex datatypes that actually span a number of fields (or sub-fields) in a message format. Each such field, by itself, does not have a specific semantic meaning in the message but is rather a syntactic artifact that when combined with other fields represent a complete datatype. For example, an address is can be composed of many fields and is a complex datatype. The Syntax Model must be able to associate each component of a complex datatype with a field in the message.

An example of a modeled MDMI complex datatype is shown in figure 8.4.2 and is posted as OMG document # 2009-09-10. This complex datatype model is composed of classes, where the classes themselves can be complex datatypes or a class with a single valued simple datatype. Ultimately, all complex datatypes resolve to a set of simple datatypes, which correspond to fields (or subfields) in a message format. Therefore, to accommodate Semantic Elements that are complex datatypes, a "fieldname" attribute is a property of the *Node* abstract class, which holds the name of the simple datatype class. For computational efficiency, a derived attribute is also added that says this node instance contains a syntactic element that is part of a complex datatype.

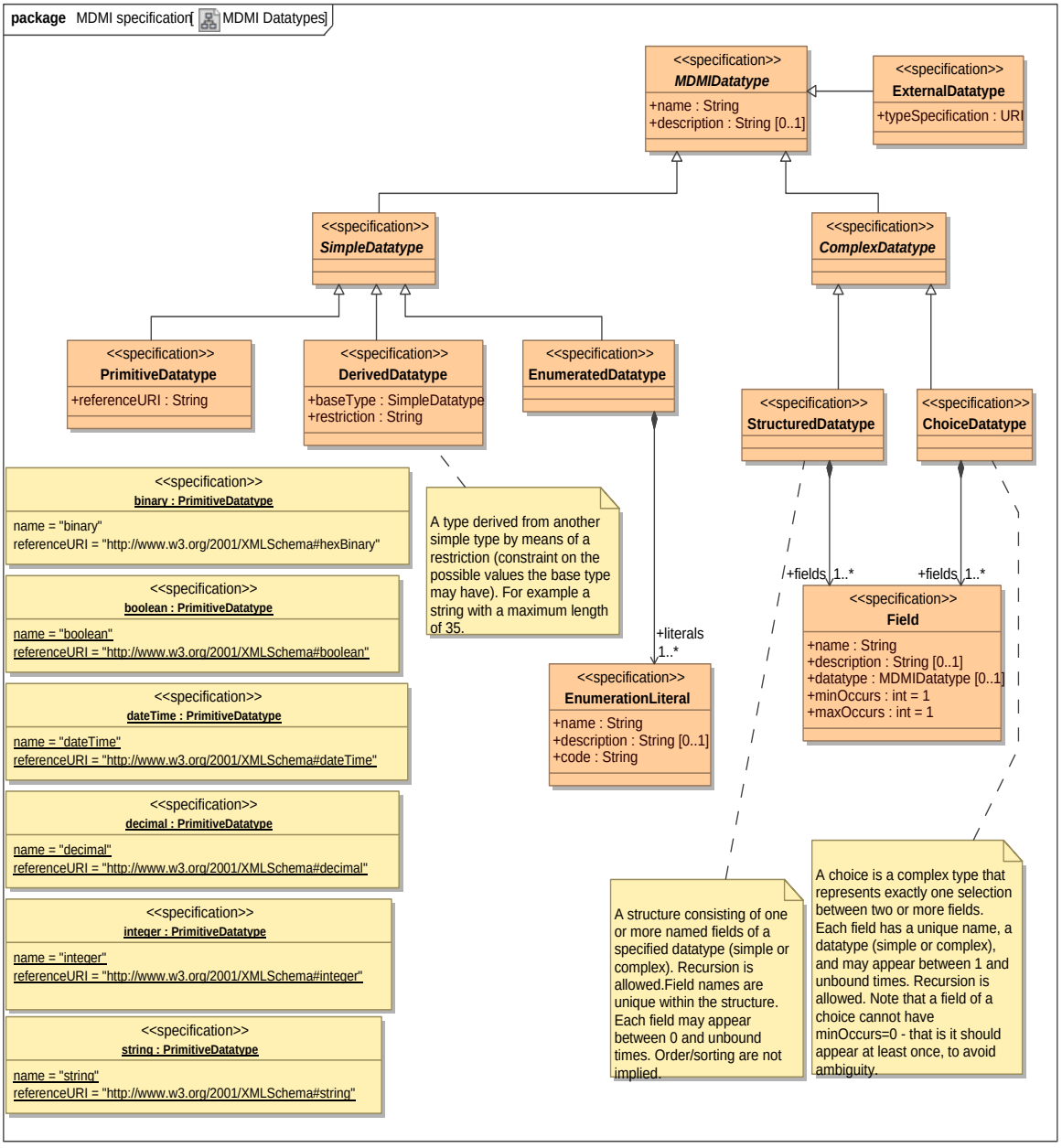



**Figure 8.4.2 Complex Datatype**

### 8.4.3 MDMIDatatype, DataRules – Abstract Syntax

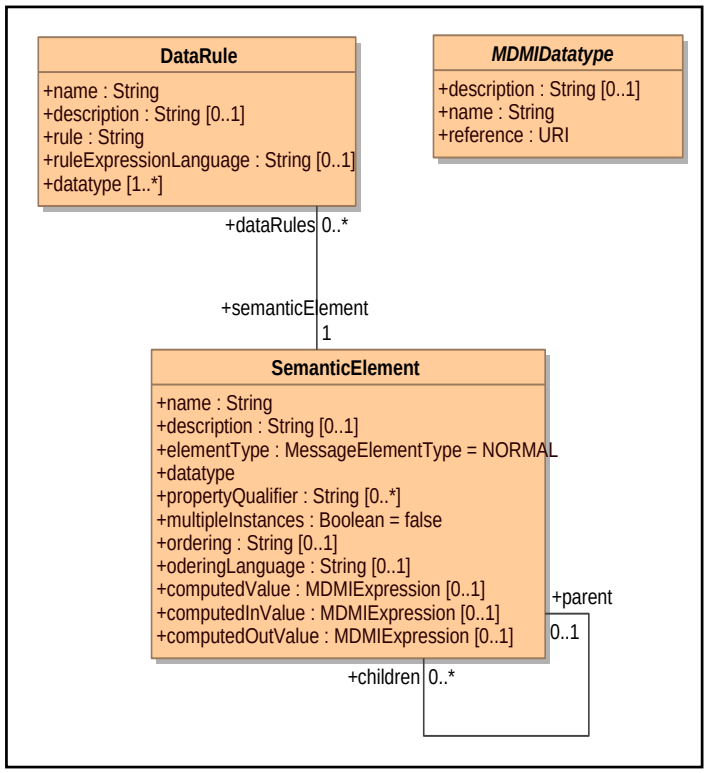



**Figure 8.4.2 – MDMIDatatypes, DataRule**

### MDMIDatatype – Detailed Semantics

MDMIDatatype description:

The MDMIDatatype class contains a reference to a conformant datatype, i.e., one that can be processed by the DataRule language. This class is used as a property type.

MDMIDatatype properties:

1. A “name” property, of type string, names of the MDMIDatatype
2. An optional “description” property, of type string, describes the MDMIDatatype
3. A “reference” property, of type URI, contains a reference to the MDMIDatatype definition

### 8.4.5 DataRules - Detailed Semantics

DataRule description:

The DataRule class contains a rule that is a constraint on the MDMIDatatype that are used in the MessageGroup, to ensure that values extracted or inserted are valid[C80].

DataRules properties:

1. A “name” property of type String whose value is the name of the DataRule.
2. An optional “description” property, of type String, contains a description of the DataRule.

3. A "rule" property, of type String, contains an expression for a rule or constraint associated with an ~~associated Datatype~~MDMIDatatype either for the entire MessageGroup or for the particular use of an MDMIDatatype in .a SemanticElement class.
4. A "ruleExpressionLanguage~~" that has a reference to~~", of type String, references the language in which the "rule" property is expressed~~—~~. The standard does not require any particular rule language, ~~so this property is needed. For example, two "ruleExpressionLanguage" values would be a reference to OCL and or a reference to a Java Expression Language~~but the language has to allow access to fields represented by simple datatype classes within a complex datatype.
5. ~~This view DatatypeRules has two~~A "datatype" property, of type MDMIDatatype and multiplicity of one-to-many, explicitly identifies the MDMIDatatypes that are referenced in a DataRule's "rule". The "datatype" references the complete structure of an MDMIDatatype, so that its structure and simple datatype fields are known. The "datatype" property is used to assist in the parsing and runtime processing of complex data[C82].

DataRules associations:

1. Zero-to-many ~~DatatypeRules~~DataRules can be associated with a ~~Datatype~~MessageGroup.

Zero-to-many ~~DatatypeRules~~DataRules can be associated with ~~an ApplicableDatatypeRules~~a SemanticElement class~~.~~

### ~~8.4.5 ApplicableDatatypeRules - Detailed Semantics~~

~~ApplicableDatatypeRules has one property:~~

~~1. A "name" property whose value is the name of the ApplicableDatatypeRules.~~

~~ApplicableDatatypeRules have two associations:~~

~~A one-to-one association, by composition, with the MessageElement for which the identified DatatypeRules apply.~~

2. ~~A one-to-many association to the DatatypeRules that are to apply to this particular MessageElement~~.

## 8.5 MDMIBusinessElementReference, Conversion Rule, ~~UnqualifiedBusinessElement~~ ToSemanticElement, To BusinessElement, MDMIBusinessElementRule

### 8.5.1 Overview

The classes in this view describe the ~~semantic mapping between a MessageElement and the UnqualifiedBusinessElement~~
~~that will be a data dictionary entry in an ISO 20022v2 compatible industry repository, for example UNIFI v2. The core of the relationship is a ConversionRule class, which contains the expressions that describe the mapping between the two~~mapping between a SemanticElement and an MDMIBusinessElementReference. An MDMIBusinessElementReference class references a Business Element in a dictionary. No assumption is made about the format of the business element in the central dictionary. Because the format of the dictionary is not known and can even be a reference to documentation, an MDMIBusinessElementRules class is included in the specification so that rules and constraints concerning the business element can be specified.

~~semantic entities. The mappings~~Given the BusinessElementReference, a conversion between it and a SemanticElement can be made. This conversion may not be symmetric so a mapping must be defined for each direction - ~~MessageElement to UnqualifiedBusinessElement and UnqualifiedBusinessElement to MessageElement.~~SemanticElement to MDMIBusinessElement and MDMIBusinessElement to SemanticElement. (Mappings for both directions must be defined, one way mappings are not allowed in the standard.) These mappings

are specified in a ToSemanticElement class and a ToBusinessElement class. Both of these classes inherit from a ConversionRule abstract class that defines how conversion rules are to be specified.

A key feature of ~~this~~the conversion is the restrictions that are implied in the ConversionRules ruleExpressionLanguage. These restrictions define the allowed semantic distance for which mapping can be done. In effect, they define the domain of "near-synonyms" that are allowed in a mapping. For example, a set of allowed conversion rules may include, simple arithmetic expressions, aggregation of a set of elements, the removal or inclusion of qualifiers~~., etc~~, etc. If a SemanticElement cannot be mapped it implies that is not in the dictionary and should be added to the dictionary.

~~If a bilateral mapping is being defined, then a MessageElement will be associated with MessageElements in the MessageElementSet for a different MessageModel. This association is not explicitly shown in this view.~~

### 8.5.2 Abstract Syntax

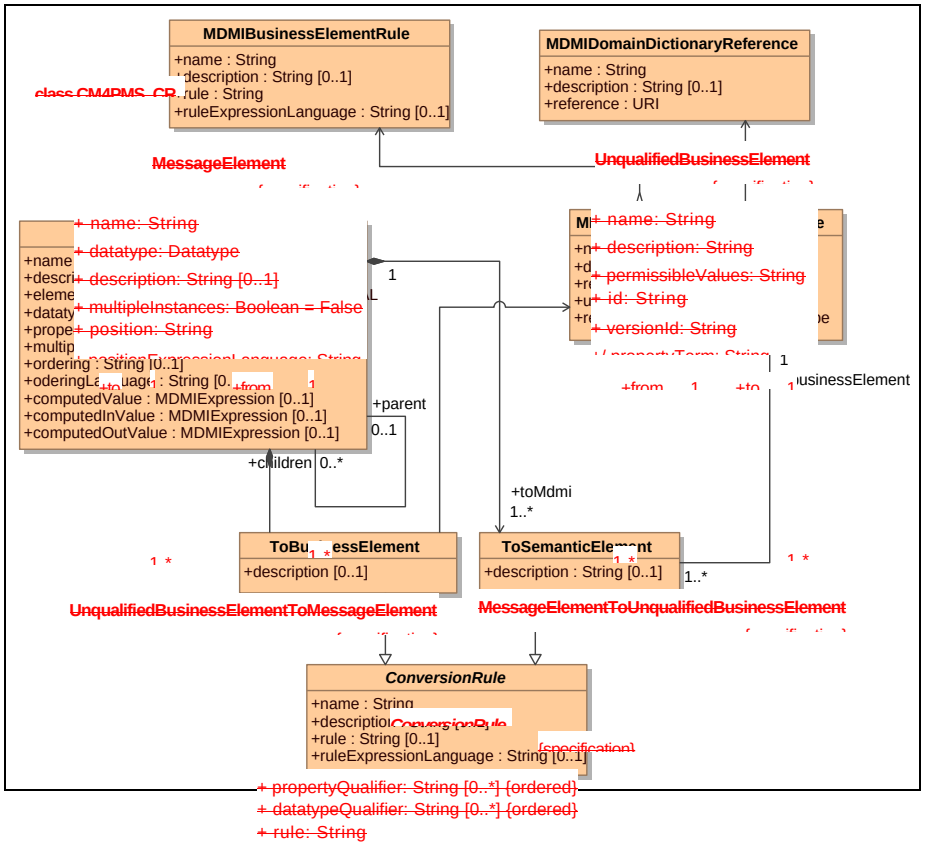


**Figure 8.5 - ~~Conversion Rule~~MDMIBusinessReference and ConversionRule**

### 8.5.3 ~~UnqualifiedBusinessElement --~~ MDMIBusinessElementReference - Detailed Semantics

~~The UnqualifiedBusinessElement is defined in the framing document for ISO 20022. The CM4PM standard will link to any ISO2022 compliant data dictionary. UnqualifiedBusinessElements must conform to their definition in that standard.~~

MDMIBusinessElementReference description:

The MDMIBusinessElementReference is a class that references a business element in a dictionary. No assumption is made about the format of the business element in the central dictionary. Therefore, the reference can only be informational. However a function must be available that, given the reference, will return a uniqueIdentifier and a reference MDMIDatatype.

MDMIBusinessElementReference properties:

1. The “name” property, of type String, names the MDMIBusinessElementReference
2. The optional “description” property, of type String, describes the MDMIBusinessElementReference
3. The “reference” property, of type URI, identifies the location of the BusinessElement in a central dictionary. (URIs are very general addresses, i.e., the URI could even point to a line in a page in a document therefore the “reference’ property is informational.)
4. The “uniqueIdentifier”, of type String, provides a unique identifier for all MDMIBusinessElementReference instances that reference the same business element in the central dictionary. There must be a function associated with the central dictionary that provides this identifier. Runtime transformation engines recognize the matching source and target mappings for a Semantic Element because they will each have the same “uniqueIdentifier”.
5. The “referenceDatatype” property, of type MDMIDatatype, provides a reference datatype for each business element in the central dictionary. There must be a function associated with the central dictionary that will deliver the “referenceDatatype”. Maps to/from this reference datatype to the “datatype” in the SemanticElement should be provided as a ConversionRule.

MDMIBusinessElementReference associations:

1. MDMIBusinessElementReference has a one-to-many association with the ToSemantic class.
2. MDMIBusinessElementReference has a one-to-many association with the ToBusinessElement class.
3. MDMIBusinessElementReference has a (zero or one)-to-many association with the MDMIBusinessElementRule class.
4. MDMIBusinessElementReference has a many-to-one relationship with the MDMIDomainDictionaryReference class.

### 8.5.4 ConversionRule -- Detailed Semantics

~~The~~ ConversionRule ~~class~~ description:

ConversionRule is an abstract class that ~~has four~~defines a rule used to convert values.

ConversionRule properties:

1. A “~~propertyqualifier” property whose value is a string of qualifiers that specialize the MessageElement in relationship to its associated UnqualifiedBusinessElement~~name” property, of type String, names the ConversionRule.
2. ~~A “datatype~~An optional “description” property ~~whose value contains the list~~, of ~~qualifiers associated with the UnqualifiedBusinessElement's datatype~~type String, describes the ConversionRule.
3. A “rule” property ~~whose value describes the required mapping between the MessageElement and the UnqualifiedBusinessElement.~~, of type String, holds an expression for converting one value to another.
4. A “ruleExpressionLanguage” property ~~whose value~~, of type String, is a reference to the expression language used to define the rule. The scope of the language allowed in conversions should be limited so that only very

straightforward transformations are possible. This is because these ConversionRules can be used to define the semantic distance between business elements in a central dictionary by identifying “near synonyms”. It is important that the “near synonyms” do not turn out to be far synonyms.

~~The~~ ConversionRule ~~Class~~generalizations:

The abstract ConversionRule class is inherited by two classes~~ the UnqualifiedBusinessElementToMessageElement class and the MessageElementToUnqualifiedBusinessElement class.~~, the “ToBusinessElement” and the “ToSemanticElement”.

### 8.5.5 ~~UnqualifiedBusinessElementToMessageElement~~ ToSemanticElement - Detailed Semantics

ToSemanticElement description:

The ~~UnqualifiedBusinessElementToMessageElement~~ToSemanticElement associates an ~~UnqualifiedBusinessElement~~MDMIBusinessElementReference to a ~~MessageElement~~SemanticElement, describing the directed conversion rule for converting the reference value of ~~the UnqualifiedBusinessElement~~a Business Element to the value in ~~the MessageElement. An UnqualifiedBusinessElement~~a SemanticElement. MDMIBusinessElementReferences may be related to more than one ~~MessageElement~~SemanticElement but will have a separate ~~UnqualifiedBusinessElementToMessageElement~~ToSemanticElement class with individual rules for each relationship.

ToSemanticElement properties:

1. The ~~UnqualifiedBusinessElementToMessageElement, which inherits from ConversionRule, has two~~optional “description” property, of type String, describes the ToSemanticElement.

ToSemanticElement associations:

1. ~~An~~ A many-to-one association with ~~one UnqualifiedBusinessElement from which a value is to be extracted~~an MDMIBusinessElementReference.
2. ~~An~~ A many-to-one association with ~~one MessageElement for which a value is to be inserted~~a SemanticElement.

### 8.5.6 ~~MessageElementToUnqualifiedBusinessElement~~ ToBusinessElement

~~The MessageElementToUnqualifiedBusinessElement~~ToBusinessElement description:

The ToBusinessElement associates an ~~UnqualifiedBusinessElement~~MDMIBusinessElementReference with a ~~MessageElement~~SemanticElement, describing the directed conversion rule for converting the value of the ~~Message Element~~SemanticElement to the reference value of the ~~UnqualifiedBusinessElement.~~referenced business element. A ~~Message Element~~ SemanticElement may be related to more than one ~~UnqualifiedBusinessElement~~MDMIBusinessElementReference but will have a separate ~~MessageElementToUnqualifiedBusinessElement~~ToBusinessElement class with individual rules for each relationship.

ToBusinessElement properties:

1. The ~~MessageElementToUnqualifiedBusinessElement class, which inherits from ConversionRule, has two~~optional “description” property, of type String, describes the ToBusinessElement.

ToBusinessElement associations:

1. ~~An~~ A many-to-one association with ~~one UnqualifiedBusinessElement to which a value is to be inserted~~an MDMIBusinessElementReference.
2. ~~An~~ A many-to-one association with ~~one MessageElement~~a SemanticElement.

### 8.5.7 MDMIBusinessElementRule

MDMIBusinessElementRule description:

> Given that the MDMI standard does not provide a specification for ~~which a value~~a the hub dictionary and allows mapping to any appropriate dictionary, such as the ISO 20022 Data Dictionary, then some business rules may have to be specified within a map to make sure that the mapping is correct. Instances of the MDMIBusinessElementRule maintain these rules.

MDMIBusinessElementRule properties:

1. A “name” property, of type String, contains a name of the rule.
2. An optional “description” property, of type String, provides a description of the rule.
3. A “rule” property, of type String, is ~~to~~ an expression defining the rule that applies to an associated MDMIBusinessElementReference.
4. An optional “ruleExpressionLanguage”, of type String, provides a reference to the language used in the “rule” property. This language must be ~~extracted~~able to describe the context in which the rule applies. The language should be able to reference the value of any Semantic Element instance and it should allow external function calls. If this property is not specified the default ruleExpressionLanguage will be used.

MDMIBusinessElementRule associations:

1. The MDMIBusinessElementRule has a many-to-one association with an MDMIBusinessElementReference.

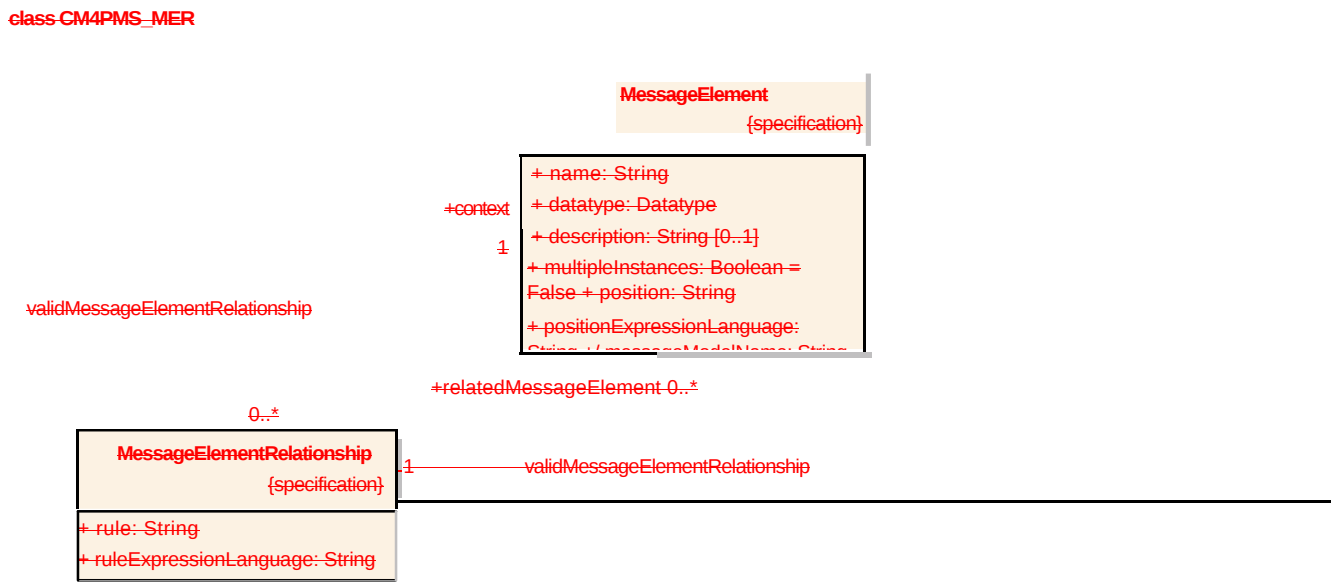
## 8.6 ~~MessageElementRelationship - Detailed Semantics~~ SemanticElementRelationship

### 8.6.1 Overview

The ~~MessageElementRelationship~~SemanticElementRelationship classes define all the allowed contexts for ~~MessageElement~~SemanticElement in a message format. For example, a ~~MessageElement~~SemanticElement that is “ClientAccountBalance” may not be valid in a message instance unless there is also a value in the ~~MessageElement~~SemanticElement “ClientAccountID.” The ~~MessageElementRelationship~~SemanticElementRelationship class would define this ~~rule.~~

relationship. On the other hand, “ClientAccountID” may exist without a value for “ClientAccountBalance,” in which case there will be no constraint defined forSemanticElementRelationship associating “ClientAccountID” in relationship towith “ClientAccountBalance.”

### 8.6.2 Abstract Syntax



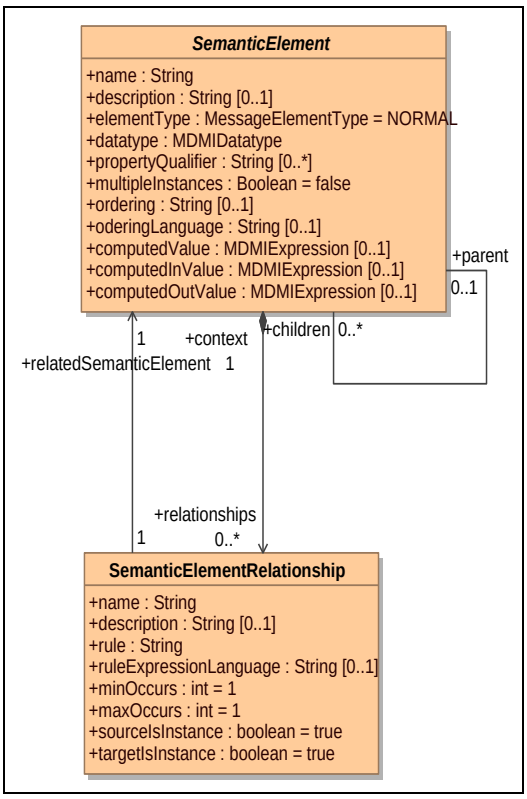



**Figure 8.6 - ~~MessageElementRelationship~~SemanticElementRelationship**

### 8.6.3 ~~MessageElementRelationship –~~ SemanticElementRelationship - Detailed Semantics

~~The MessageElementRelationship has two~~SemanticElementRelationship description:

> The SemanticElementRelationship class is a key artifact in the MDMI standard. It provides all the context and dependency relationships for each SemanticElement. SemanticElementRelationship make it possible to extract and insert SemanticElement values in a valid manner.

SemanticElementRelationship properties:

1. A "~~rule~~name" property, ~~whose value expresses a validity constraint~~of type String, assigns a name to the rule.
2. An optional "description" property, of type String, provides a description of the rule
3. A "rule" property, of type String, defines a relationship between a ~~MessageElement~~source SemanticElement and other ~~MessageElements~~SemanticElements in the ~~MessageElementSet~~SemanticElementSet.
4. A "ruleExpressionLanguage" property, ~~whose value~~of type String, that contains a reference to the expression language used in the "rule" property. This rule language must be able to access the values of any SemanticElement and to do that it must be able to access the fields in complex datatypes.
5. "minOccurs" property, of type integer, indicates how many instances of the target at a minimum must be involved in the relationship.
6. A "maxOccurs" property of type integer, which says how many instances, at most can be involved in the relationship.
7. A "sourceIsInstance" property of type Boolean. When the sourceIsInstance is true, the defined relationship is for each Instance of the source SemanticElement. (The association with the "source" Semantic Element is labeled "relatedSemanticElement. The relatedSemanticElement owns the relationship by composition. This source is the

SemanticElement whose context is being modeled) When the sourceIsInstance is false, the defined relationship is for the source SemanticElement class as a whole

8. A "targetIsInstance" property of type Boolean. When the targetIsInstance is true, the defined relationship is for each Instance of the target SemanticElement. (The association with the set of one-to-many "targets" is labeled "context. (Thus, a SemanticElementRelationship describes a relationship between a source and the other SemanticElements, which are then targets.) When the targetIsInstance is false, the defined relationship is for the SemanticElement class as a whole

SemanticElementRelationship associations:

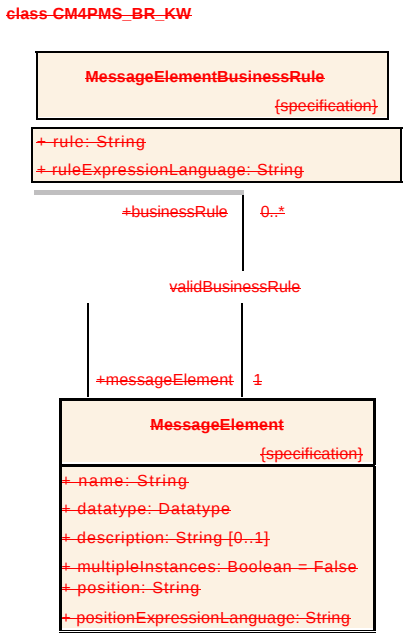
1. The SemanticElementRelationship has a (zero or many)-to-one association with its source SemanticElement.
2. The SemanticElementRelationship has one to-one association with a target SemanticElement.

## 8.7 ~~MessageElementBusinessRule~~ SemanticElementBusinessRule

### 8.7.1 Overview

The ~~MessageElementBusinessRule~~SemanticElementBusinessRule class contains a rule that is to be applied to a specific ~~MessageElement~~SemanticElement in the context of the MessageModel that contains the ~~MessageElement~~SemanticElement.

### 8.7.2 Abstract Syntax



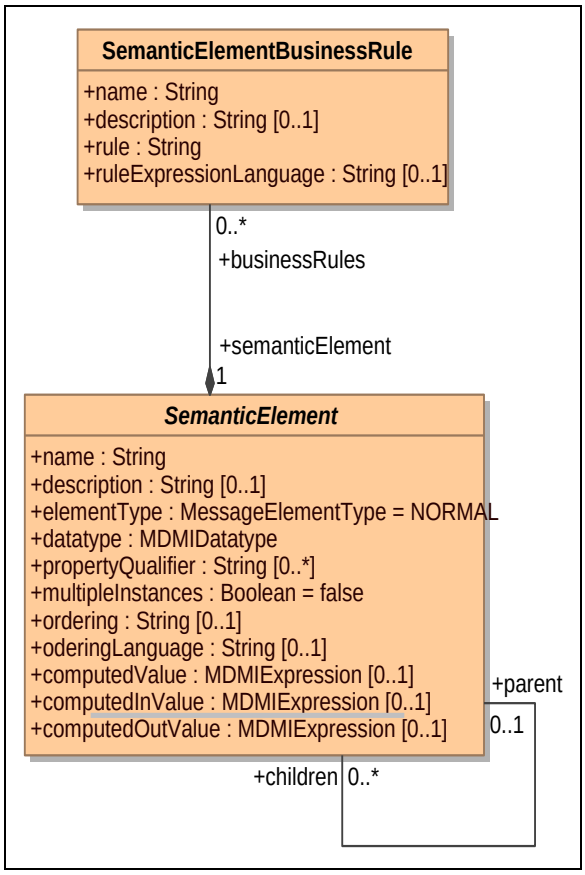



**Figure 8.7 - ~~Business~~SemanticElementBusiness Rule**

### 8.7.3 ~~Business.~~ SemanticElementBusinessRule - Detailed Semantics ~~- Detailed Semantics~~

~~The BusinessRule class contains two properties:~~

~~A "rule~~SemanticElementBusinessRule description:

> The SemanticElementBusinessRule holds a rule that is to be applied to a SemanticElement to make sure that the SemanticElement is valid. SemanticElementBusinessRule usually do not refer to other SematicElements in a message. They are meant to provide rules that reflect an external context, e.g., a "Primary AcountID" SemanticElement must be from an EU bank, etc.

SemanticElementBusinessRule properties:

1. A "name" property, ~~whose value is a Boolean expressions~~ of type String, assigns a name to the rule.
2. An optional "description" property, of type String, provides a description of the rule.
3. A "rule" property, of type String, is an expression defining a business rule or constraint.
4. A "ruleExpressionLanguage" property, ~~whose value~~of type String, is a reference to the expression language used in the "rule" property.

~~The BusinessRule class contains one association:~~

~~1. A zero-to-many association with the MessageElement to which the BusinessRule applies.~~

## 8.8 Message Element, MessageModel and Unqualified Business Element Instances

### 8.8.1 Overview

The artifacts discussed above provide a complete description of a message format. However there also need to be artifacts that relate to a specific physical message instance so, that the instance can be located and also the values to be converted can be stored in a syntax neutral mode. There are three classes defined to hold relevant information needed for the conversion of a physical message instance:

1. A MessageInstance that identifies the location of the physical message instance;

a MessageElementInstance that holds the value on any MessageElements appearing in the physical message instance; and

an UnqualifiedBusinessElementInstance, which holds the converted value of those MessageElements.

### 8.8.2 Abstract Syntax

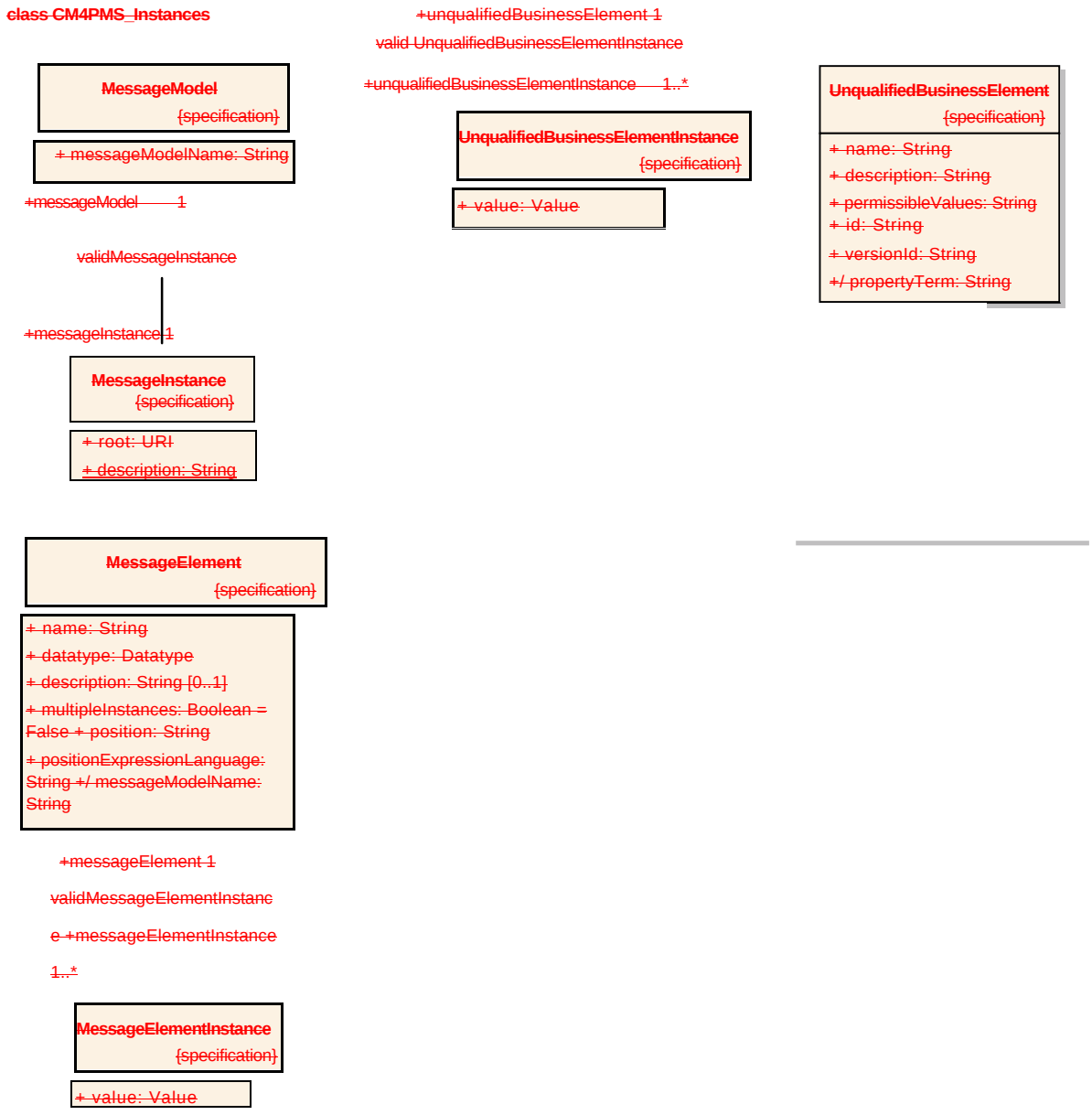



**Figure 8.8 - Message Element, Message Model and Unqualified Business Element Instances**

### 8.8.3 MessageInstances - Detailed Semantics

The MessageElement has two properties:

~~A "root" property whose value is a URI indicating the location of the message instance.~~

~~A "description" property whose value provides a description of the instance~~

~~location. The Message Element has one association:~~

~~1. A one-to-one association with the a MessageModel that describes its message format.~~

### ~~8.8.4 MessageElementInstance~~

~~The MessageElementInstance has one property~~SemanticElementBusinessRule associations:

1. A ~~"value" property whose value is the value in the physical message instance, which is associated with a particular MessageElement.~~

~~The MessageElementInstance has~~ (zero or many)-to-one association~~:~~ with the SemanticElement to which the MDMIBusinessElementRule applies.

~~1. A one-to-many association with the MessageElement to which it belongs.~~

### 8.8.5 UnqualifiedBusinessElementInstance

The UnqualifiedBusinessElementInstance has one property:

1. A "value" property whose value is the value in converted from a MessageElementInstance value using the ConversionRule.

The UnqualifiedBusinessElementInstance has one association:

1. A one-to-many association with the UnqualifiedBusinessElement that is associated with a particular MessageElement.

There is a constraint associated with instances as presented in the figure below. It is if the Message Element is not allowed to have Multiple Instances, then there can only be One MessageElementInstance associated with a MessageElement.

**class CM4PMS_Instances**

```
if
        (MessageElement.multipleInstances == False)
then
         (valueOf(MessagElement.messageElementInstance) = 1)
elseif
         (MessageElement.multipleInstances == True)
then
        (val ueOf(MessagEl ement.messageEl ementInstance) = 1 ..*)
else
        Error
end
```

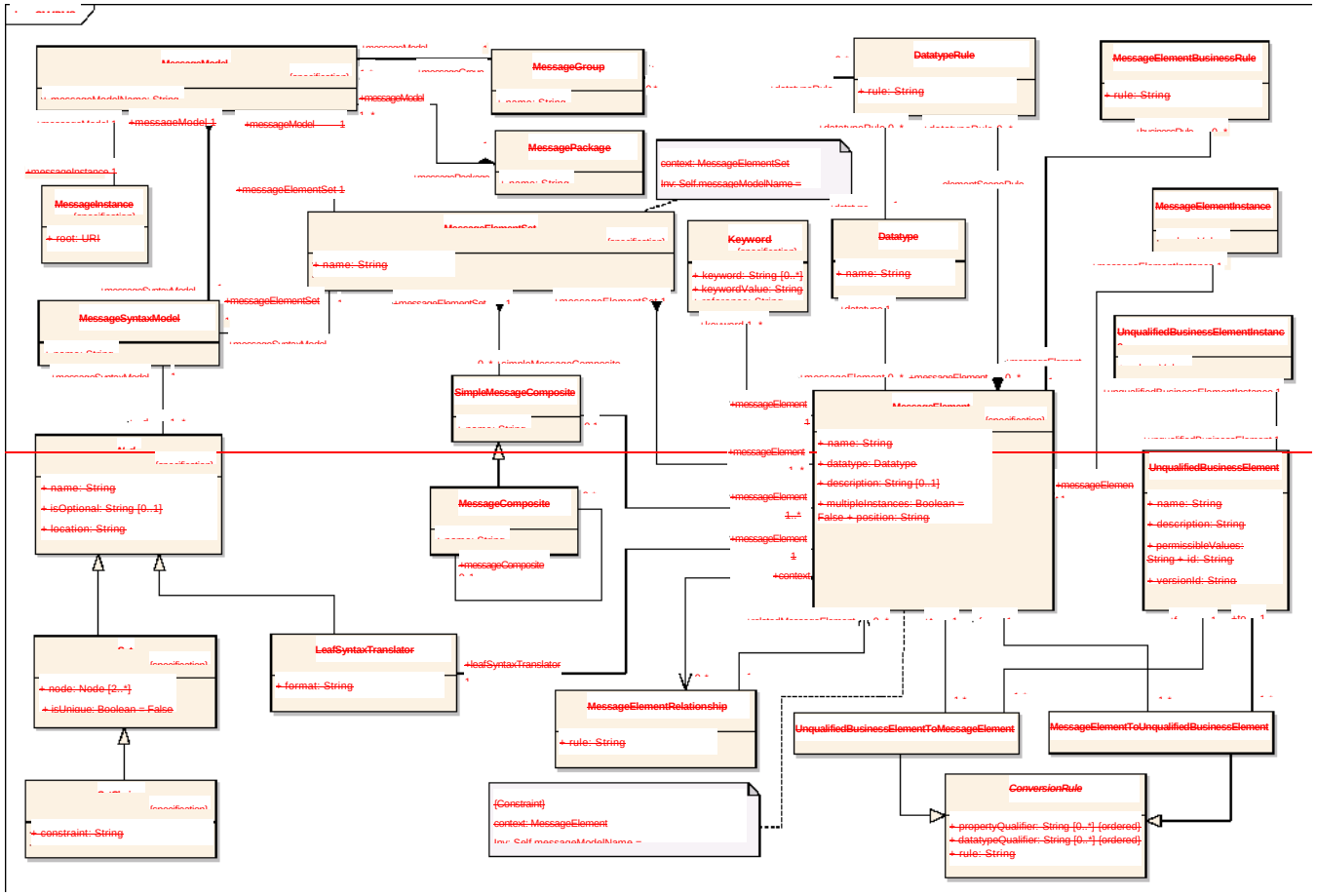
**Figure 8.9 - MessageElelementInstance Constraint**

## 8.9 8.8 Summary of Complete Metamodel

### 8.98.1 Overview

The complete metamodel is shown for completeness in Figure 8.108.

### 8.9.8.2 Abstract Syntax


MessageModel
MessageGroup
MessagePackage
MessageInstance
+ root: URI
MessageElementSet
+ name: String
MessageSyntaxModel
Keyword
+ keyword: String [0..*]
+ keywordValue: String
Datatype
+ name: String
DatatypeRule
+ rule: String
MessageElementBusinessRule
+ rule: String
MessageElementInstance
UnqualifiedBusinessElementInstanc
SimpleMessageComposite
MessageComposite
+messageComposite
+ name: String
+ isOptional: String [0..1]
+ location: String
+ name: String
+ datatype: Datatype
+ description: String [0..1]
+ multipleInstances: Boolean = False + position: String
UnqualifiedBusinessElement
+ name: String
+ description: String
+ permissibleValues: String + id: String
+ versionId: String
+ node: Node [2..*]
+ isUnique: Boolean = False
+ constraint: String
LeafSyntaxTranslator
+ format: String
MessageElementRelationship
+ rule: String
UnqualifiedBusinessElementToMessageElement
MessageElementToUnqualifiedBusinessElement
ConversionRule
+ propertyQualifier: String [0..*] {ordered}
+ datatypeQualifier: String [0..*] {ordered}
+ rule: String
{Constraint}
context: MessageElement
context: MessageElementSet
inv: Self.messageModelName =

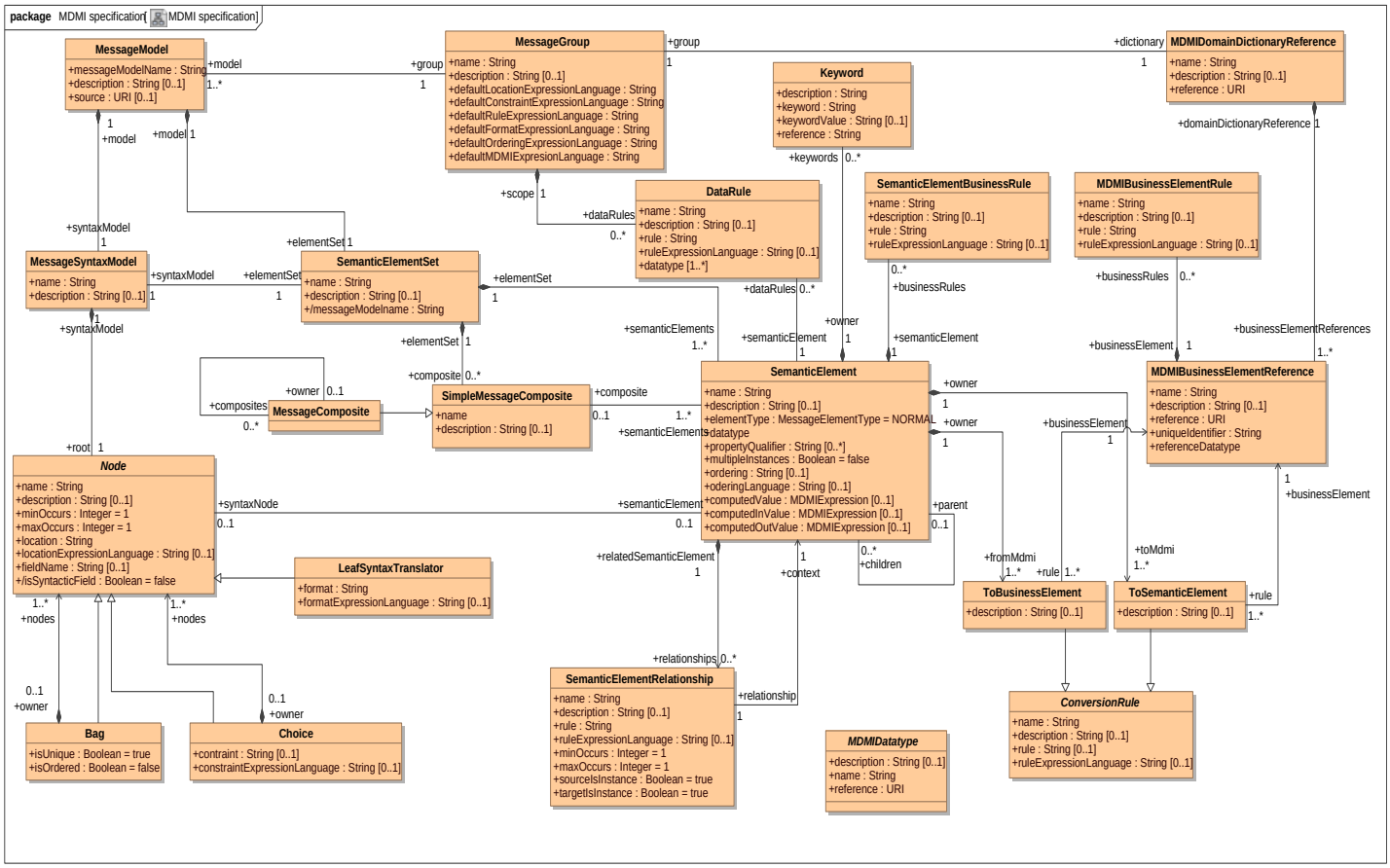



**Figure 8.~~108~~ - Summary: Complete Metamodel**

# Annex A - List of Acronyms

| Abbreviation | Notes |
|---|---|
| Chips | Clearing House Inter-Bank Payments System<br>www.chips.org |
| CLS | Continuous Linked Settlement<br>http://www.cls-services.com |
| FATF | The FATF is an inter-governmental body whose purpose is the development and promotion of national and international policies to combat money laundering and terrorist |
| FIX | Financial Information eXchange<br>http://www.fixprotocol.org |
| FpML | Financial products Markup Language is the industry-standard protocol for complex financial products.<br>http://www.fpml.org |
| I FX | Interactive Financial eXchange<br>www.ifxforum.org |
| GiovanniniM DDL | The Giovannini Group is a group of financial market participants, |
| M DDL | Market Data Definition Language www.mddl.org |
| NRL and NRL 1.0 | Natural Rule Language – Open source constraint and action language based on OCL The user guide can be found at http://nrl.sourceforge.net/userguide/userguide.htm |
| MiFI DSwift | MiFID will replace the existing Investment Services Directive (ISD), the most significant European Union legislation for investment intermediaries and financial markets since 1995.Society for Worldwide Interbank Financial Telecommunication supplies secure |
| Omgeo | Omgeo plays a core role as the orchestrator of post-trade pre-settlement trade management within the global securities industry<br>www.omgeo.com |
| Twist | Transaction Workflow Innovation Standards Team<br>www.twiststandards.org |
| RTGS | Real-Time Gross Settlement |
| RIXML | RIXML.org is a consortium of buy-side and sell-side firms that has established an open standard for investment and financial research.<br>www.rixml.org |
| | |
| Sepa | Single Euro Payments Area |
| Swift | Society for Worldwide Interbank Financial Telecommunication supplies secure messaging services.<br>http://www.swift.com |

| | |
|---|---|
| ~~Target2~~ | ~~TARGET2 is to become a system that provides extensively harmonized services via an integrated IT infrastructure, improves cost efficiency, is prepared for swift adaptation to future developments, including the enlargement of the Eurosystem.~~ ~~http://www.ecb.int/paym/target/target2/html/index.en.html~~ |
| ~~Twist~~ | ~~Transaction Workflow Innovation Standards Team~~ ~~www.twiststandards.org~~ |
| ~~Unifi~~ | ~~UNIversal Financial Industry message scheme (ISO20022) www.iso20022.org~~ |
| ~~XBRL~~ | ~~XBRL is a language for the electronic communication of business and financial data.~~ ~~http://www.xbrl.org~~ |