Shared Data Model and Notation (SDMN)

Version 1.0 Beta 2

OMG Document Number: dtc/2024-02-05
Normative reference: https://www.omg.org/spec/SDMN/1.0/Beta2
Machine readable file(s): https://www.omg.org/SDMN/20240210

Normative:
https://www.omg.org/spec/SDMN/20240210/SDMN.xsd
https://www.omg.org/spec/SDMN/20240210/SDMN.xmi
https://www.omg.org/spec/SDMN/20240210/SDMN.mdzip

Informative:
https://www.omg.org/spec/SDMN/20240210/SDMN-examples.zip

Commented [SW1]: This convenience document provides the changes (deletions and additions) to the SDMN specification based on the resolutions for issues raised for the SDMN FTF. For this document, all the issues that were being approved for all FTF Ballots. Thus, this represents the final specification for the FTF. A comment is attached to each change in the document. The comment identifies the type of change (e.g., a figure update) and the raised issue and its resolution sub-task. Thus, the issues will be identified as so (e.g., SDMN-2/SDMN-51). By searching through the document for a particular issue (e.g., SDMN-2), you can find all the changes to the specification based on the resolution for that issue.
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, 9C Medway Road PMB 274, Milford, MA, 01757 U.S.A.

TRADEMARKS

CORBA®, CORBA logos®, FIBO®, Financial Industry Business Ontology®, FINANCIAL INSTRUMENT GLOBAL IDENTIFIER®, IDI®, IMM®, Model Driven Architecture®, MDA®, Object Management Group®, OMG®, OMG Logo®, SoaML®, SOAML®, SysML®, UAF®, Unified Modeling Language®, UML®, UML Cube Logo®, VSIPL®, and XMI® are registered trademarks of the Object Management Group, Inc.
For a complete list of trademarks, see: https://www.omg.org/legal/tm_list.htm. 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 Specifications, Report a Bug/Issue.
Table of Contents

1 Scope ................................................................................................................. 8
2 Conformance ..................................................................................................... 8
2.1 General ........................................................................................................... 8
2.2 Shared Data Modeling Conformance ............................................................. 8
2.3 Visual Conformance ....................................................................................... 8
3 References ........................................................................................................ 9
3.1 Normative References ................................................................................... 9
3.2 Non-normative References ............................................................................ 10
4 Terms and Definitions ....................................................................................... 10
5 Symbols ............................................................................................................. 10
6 Additional Information ..................................................................................... 10
6.1 Conventions................................................................................................... 10
6.2 Typographical and Linguistic Conventions and Style .................................... 11
6.3 Display of Metamodel Diagrams .................................................................... 11
6.4 Use of Text, Color, Size, and Lines in a Diagram ............................................ 13
6.5 Abbreviations .............................................................................................. 13
6.6 Structure of this Document .......................................................................... 13
6.7 Acknowledgements ....................................................................................... 14
7 Overview .......................................................................................................... 14
7.1 What Constitutes a BPM+ Model? ................................................................. 15
7.2 Why a Shared Data Model? .......................................................................... 15
7.2.1 Use Case: Hello Patient ........................................................................... 16
7.3 The Purpose and Use of a Shared Data Model .............................................. 22
8 Specification Core Elements ........................................................................... 23
9 SDMN Metamodel .......................................................................................... 25
9.1 SharedDataModel ......................................................................................... 28
10 SDMN Model Elements ............................................................................... 34
10.1 DataItems ................................................................................................... 34
10.1.1 DataItem .................................................................................................. 36
10.1.2 Pre-Assigning Values for DataItems ....................................................... 44
10.2 ItemDefinitions .......................................................................................... 45
10.2.1 ItemDefinition ....................................................................................... 47
10.3 Connectors ................................................................................................... 61
10.3.1 Connector .............................................................................................. 61
10.3.2 CompositionConnector .......................................................................... 63
10.3.3 ContainmentConnector .......................................................................... 65
10.3.4 DataAssociation .................................................................................... 67
10.3.5 ReferenceConnector .............................................................................. 70
10.4 ModelArtifacts ............................................................................................ 72
11 Mapping to BPM+ Models ............................................................................ 78
11.1 Element Terminology Mapping to BPM+ Element Terminology............... 79
11.2 BPMN .......................................................................................................... 79
11.3 CMMN ........................................................................................................ 81
11.4 DMN ............................................................................................................ 82

Commented [SW3]: The content tables were updated to reflect changes in the document structure.
12 SDMN Examples.................................................................................................84
12.1 Hello Patient.......................................................................................................84
13 Exchange Formats................................................................................................87
13.1 Interchanging Incomplete Models.................................................................87
13.2 XSD..................................................................................................................87
13.2.1 Document Structure.......................................................................................87
13.2.2 References within the SDMN XSD...............................................................88
14 SDMN Diagram Interchange (SDMN DI).........................................................88
14.1 Scope...................................................................................................................88
14.2 Diagram Definition and Interchange...............................................................89
14.3 SDMN Diagram Interchange Meta-Model.......................................................89
14.3.1 How to read this Chapter.............................................................................89
14.3.2 Overview.......................................................................................................89
14.3.3 Measurement Unit.......................................................................................90
14.3.4 Elements.......................................................................................................90
14.3.4.1 SDMN DI................................................................................................90
14.3.5 Notation.........................................................................................................96
14.3.5.1 Labels.......................................................................................................96
14.3.5.2 SDMNShape Resolution...........................................................................97
14.3.5.3 SDMNEdge Resolution............................................................................98
Preface

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 Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

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

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Specifications are available from the OMG website at:

https://www.omg.org/spec

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
9C Medway Road
PMB 274
Milford, MA 01757
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 https://www.iso.org

Issues

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 https://www.omg.org, under Documents, Report a Bug/Issue.
1 Scope

A Shared Data Model is a repository collection of DataItems and ItemDefinitions to be used (referenced) by the other BPM-Plus (BPM+) data elements:

- BPMN Data Objects, CMMN Case File Items, DMN Data Inputs, etc.
- The DataItems and ItemDefinitions can be created once and maintained in a single location and can then maintenance can be distributed across multiple models
  - This eliminates the manual synchronization burden of working with the BPM+ models without a Shared Data Model
- A Shared Data Model is a model because there are relationships between the DataItems and ItemDefinitions (e.g., parent-child)
  - A diagram can be included to visualize the DataItems and their relationships or ItemDefinitions and their relationships
- Note that a Shared Data Model is not an executable model
  - It is a library for the executable BPM+ models

The primary goal of SDMN is to provide a set of structural elements that are common to other Object Management Group (OMG) specifications [BPMN Knowledge Package Model and Notation (BKPMN), Pedigree and Provenance Model and Notation (PPMN), and SDMN have been structured to be dependent on the elements defined in Specification Common Elements (see the SCE specification for more information [OMG doc number bmi-2021-12-09])]. Other Business Modeling and Integration (BMI) Task Force and Healthcare Domain Task Force (HDTF) specifications may also utilize the elements of SCE as they are updated in the future.

2 Conformance

2.1 General

Software can claim compliance or conformance with SDMN 1.0 if and only if the software fully matches the applicable compliance points as stated in the specification. In addition, the structural elements provided by Specification Common Elements (SCE 1.0 [OMG doc number bmi-2021-12-09]) are also required in a compliant or conformant software solution. Software developed only partially matching the applicable compliance points can claim only that the software was based on this specification but cannot claim compliance or conformance with this specification.

2.2 Shared Data Modeling Conformance

The implementation claiming conformance to the Shared Data Modeling Conformance SHALL comply with all of the requirements set forth in Clauses 8, 9, and 10; and it should be conformant with the Visual Notation Conformance in Clause 14. Conformant implementations SHALL fully support and interpret the exchange format specified in Clause 13.

This compliance point is intended to be used by SDMN modeling tools.

2.3 Visual Conformance

An implementation that creates and displays SDMN models SHALL conform to the specifications and restrictions with respect to diagrammatic relationships between graphical elements, as described in Clause 14. A key element of SDMN is the choice of shapes and icons used for the graphical elements identified in this specification. The intent is to create a standard visual language that all Shared Data modelers will recognize and understand. An implementation that creates and displays SDMN
models SHALL use the graphical elements, shapes, markers and decorators illustrated in this specification.

There is flexibility in the size, color, line style, and text positions of the defined graphical elements, except where otherwise specified. In particular:

- **SDMN** elements MAY have labels (e.g., its name and/or other attributes) placed inside the shape, or above or below the shape, in any direction or location, depending on the preference of the modeler or modeling tool vendor.
- The fills that are used for the graphical elements MAY be white or clear. The notation MAY be extended to use other fill colors to suit the purpose of the modeler or tool (e.g., to highlight the value of an object attribute).
- Graphical elements, shapes, and decorators MAY be of any size that suits the purposes of the modeler or modeling tool with the condition that the additional graphical elements SHALL NOT conflict with any current BPM+ Standard defined graphical element.
- The lines that are used to draw the graphical elements MAY be black.
- The notation MAY be extended to use other line colors to suit the purpose of the modeler or tool (e.g., to highlight the value of an object attribute).
- The notation MAY be extended to use other line styles to suit the purpose of the modeler or tool (e.g., to highlight the value of an object attribute) with the condition that the line style SHALL NOT conflict with any current BPM+ Standard defined line style.

The following extensions to a **SDMN** model are permitted:

- New decorators or indicators MAY be added to the specified graphical elements. These decorators or indicators could be used to highlight a specific attribute of a **SDMN** element or to represent a new subtype of the corresponding concept with the condition that the additional graphical elements SHALL NOT conflict with any current BPM+ Standard defined decorator or indicator.
- A new shape representing a kind of **DataItem** or **ItemDefinition** MAY be added to a model with the condition that the shape SHALL NOT conflict with the shape specified for any other BPM+ Standard element or decorator.
- Graphical elements MAY be colored, and the coloring MAY have specified semantics that extend the information conveyed by the element as specified in this standard.
- The line style of a graphical element MAY be changed, but that change SHALL NOT conflict with any other line style REQUIRED by this specification or the other BPM+ Standards.
- An extension SHALL NOT change the specified shape of a defined graphical element or decorator. (e.g., changing a square into a triangle, or changing rounded corners into squared corners, etc.).

This compliance point is intended to be used by entry-level **SDMN** tools.

### References

#### 3.1 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

- Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, IETF RFC 2119, March 1997
  
  [RFC2119]: http://www.ietf.org/rfc/rfc2119.txt

- Business Process and Model Notation (BPMN™): [BPMN](https://www.omg.org/bpmn/)

- Case Management Model and Model Notation (CMMN™): [CMMN](https://www.omg.org/spec/CMMN/)

- Diagram Definition (DD™): [DD](https://www.omg.org/spec/DD/)

Commented [SW9]: This text was bolded for the resolution of Issue SCE-69/SCE-106. Editorial changes. Make "BPM+" bold. This will be done throughout the document but WILL NOT be marked with a comment each time.

Commented [SW10]: This text was updated for the resolution of Issue SDMN-69/SDMN-106. Editorial Issues.
3.2 Non-normative References

The following normative documents does not contain any non-normative references provisions which, through reference in this text, constitute exemplars or influencers of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.


4 Terms and Definitions

The table below presents a glossary for this specification:

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case</td>
<td>A CMMN element that is a proceeding that involves actions taken regarding a subject in a particular situation to achieve a desired outcome.</td>
</tr>
<tr>
<td>DataItem</td>
<td>A SDMN DataItem represents a common definition and structure for the data handling elements of the other BPM+™ models.</td>
</tr>
<tr>
<td>DataState</td>
<td>DataItems can optionally reference a DataState element, which is the state of the data contained in the DataItem. The definition of these DataStates, e.g., possible values and any specific semantic are out of scope of this specification. Therefore, SDMN adopters can use the DataState element and the SDMN extensibility capabilities to define their DataStates.</td>
</tr>
<tr>
<td>Decision</td>
<td>A DMN element that is the act of determining an output value (the chosen option), from a number of input values, using logic defining how the output is determined from the inputs.</td>
</tr>
<tr>
<td>ItemDefinition</td>
<td>Defines the detailed structure, which can be simple or complex, of a DataItem.</td>
</tr>
<tr>
<td>Process</td>
<td>A BPMN element that describes a sequence or flow of Activities in an organization with the objective of carrying out work. The ProcessRef element provides a link to a Process in a BPMN document.</td>
</tr>
</tbody>
</table>

5 Symbols

There are no symbols defined in this specification.

6 Additional Information

6.1 Conventions

The section introduces the conventions used in this document. This includes (text) notational conventions and
notations for schema components. Also included are designated namespace definitions.

### 6.2 Typographical and Linguistic Conventions and Style

This document incorporates the following conventions:

- The keywords “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document are to be interpreted as described in RFC-2119.
- A **term** is a word or phrase that has a special meaning. When a term is defined, the term name is highlighted in **bold** typeface.
- A reference to another definition, section, or specification is highlighted with underlined typeface and provides a link to the relevant location in this specification.
- A reference to a graphical element is highlighted with a bold, capitalized word (e.g., `ProcessRef`).
- A reference to a non-graphical element or SDMN concept is highlighted by being italicized and (e.g., `Documentation`).
- A reference to an attribute or model association will be presented with the **Courier New** font (e.g., `Expression`).
- Non-normative examples are set off in boxes and accompanied by a brief explanation.
- XML and pseudo code is highlighted with **Courier New** typeface. Different font colors MAY be used to highlight the different components of the XML code.
- The cardinality of any content part is specified using the following operators:
  - `[1]` — exactly once
  - `[0..1]` — 0 or 1
  - `[0..*]` — 0 or more
  - `[1..*]` — 1 or more
- Attributes separated by `|` and grouped within `{` and `}` — alternative values
- `<value>` — default value
- `<type>` — the type of the attribute

### 6.3 Display of Metamodel Diagrams

The metamodel presented in these sections utilizes the patterns and mechanisms that are used for the current BPM+ specifications. BPM+ specifications rarely display the entire metamodel of a technical specification in a single diagram. The entire metamodel would be very large, complicated, and hard to follow. Typically, a specification will present sub-sets of the overall metamodel as they apply to specific topics. For example, in the BPMN specification there are metamodel diagrams that show the elements relating to activities or data elements. This document will follow that pattern and present sub-sets of a larger metamodel.

The metamodel diagrams are Unified Modeling Language (UML) structure diagrams. In addition to the metamodel, OMG specifications provide XML schemas which map to the metamodels. In general, it is through XML documents that BPM+ models are stored and exchanged.

Further, some of the metamodel elements are references to elements from other specifications. To clarify the owner of the metamodel element, there is a parenthesized text that identifies the model owner of that element. In addition, colors are used to support the text identification of the owner-language of that element. The colors are used as an aid to distinguish the languages but does not represent a normative aspect of the metamodels nor do they add any semantic information about the metamodels.

The table below presents examples of elements used throughout the metamodel diagrams within this specification:
Table 2. SDMN Metamodel Color-Coding

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Example Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCE Structural Class</td>
<td>Metamodel elements from the SCE 1.0 specification [OMG doc number bmi-2021-12-09] are shown in SDMN metamodel diagrams when SDMN elements are dependent on a SCE element. These elements include the owner of the language (SCE) in parentheses below the element name and these elements are color-coded lavender (see figure to the right).</td>
<td><img src="SCE_Core" alt="SDMN_element" /></td>
</tr>
<tr>
<td>SDMN General Class</td>
<td>These elements include the owner of the language (SDMN) in parentheses below the element name and these elements are color-coded purple and the border line color is purple (see figure to the right). These make up the majority of metamodel elements shown in this specification.</td>
<td><img src="SDMN" alt="SituationalDataPackage" /></td>
</tr>
<tr>
<td>SDMN General Class (focus of diagram)</td>
<td>These elements have the same naming and color, but the border line color is dark blue instead of light brown (see figure to the right). They are highlighted as the focus of the particular metamodel diagram. This is an informative depiction that does not add any semantic information about the particular metamodel diagram.</td>
<td><img src="SDMN_Data" alt="DataItem" /></td>
</tr>
<tr>
<td>DMN General Class</td>
<td>Metamodel elements from the DMN 1.3 specification are shown in SDMN metamodel diagrams when SDMN elements are dependent on a DMN element. These elements include the owner of the language (DMN) in parentheses below the element name and these elements are color-coded yellowish (see figure to the right).</td>
<td><img src="DMN-3" alt="ItemDefinition" /></td>
</tr>
<tr>
<td>External Class</td>
<td>Classes from specifications that are not specifically part of the BPM+ stack of standards can be included in metamodel diagrams and display the owner of the language in parentheses below the element name and these elements are color-coded light-gray. (see figure to the right).</td>
<td><img src="SCE_Di" alt="Shape" /></td>
</tr>
<tr>
<td>SDMN Class Instance</td>
<td>These elements elements include the owner of the language (SDMN) in parentheses below the element name and these elements are color-coded light-purple to identify SDMN class instances from the SDMN Library (see figure to the right).</td>
<td><img src="SDMN_library.ItemKinds" alt="DataType - ItemKind" /></td>
</tr>
<tr>
<td>SCE Class Instance</td>
<td>These elements include the owner of the language (SCE) in parentheses below the element name and these elements are color-coded light-violet to identify SCE class instances from the SCE Library (see figure to the right).</td>
<td><img src="SCELibrary.RelationshipKinds" alt="Composition - RelationshipKind" /></td>
</tr>
<tr>
<td>Enumerations</td>
<td>(see figure to the right).</td>
<td><img src="enumeration" alt="&lt;enumeration&gt; MultiplicityTypes" /></td>
</tr>
</tbody>
</table>

Commented [SW15]: This text and figure were updated for the resolution of Issue SDMN-129/SDMN-130. Editorial Issues. Clean up references to DMN.
6.4 Use of Text, Color, Size, and Lines in a Diagram

- Diagram elements MAY have labels (e.g., its name and/or other attributes) placed inside the shape, or above or below the shape, in any direction or location, depending on the preference of the modeler or modeling tool vendor.
- The fills that are used for the graphical elements MAY be white or clear.
  - The notation MAY be extended to use other fill colors to suit the purpose of the modeler or tool (e.g., to highlight the value of an object attribute).
- Diagram elements and markers MAY be of any size that suits the purposes of the modeler or modeling tool.
- The lines that are used to draw the graphical elements MAY be black.
  - The notation MAY be extended to use other line colors to suit the purpose of the modeler or tool (e.g., to highlight the value of an object attribute).
  - The notation MAY be extended to use other line styles to suit the purpose of the modeler or tool (e.g., to highlight the value of an object attribute) with the condition that the line style SHALL NOT conflict with any current defined line style of the diagram.

6.5 Abbreviations

The table below presents a list of acronyms, and their definition, that are used in this specification:

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BHMN</td>
<td>BPM+ Harmonization Model and Notation</td>
</tr>
<tr>
<td>BKPMN</td>
<td>BPM+ Knowledge Package Model and Notation</td>
</tr>
<tr>
<td>BPM+</td>
<td>Business Process Management Plus</td>
</tr>
<tr>
<td>BPMN</td>
<td>Business Process Model and Notation</td>
</tr>
<tr>
<td>CMMN</td>
<td>Case Management Model and Notation</td>
</tr>
<tr>
<td>DC</td>
<td>Diagram Commons</td>
</tr>
<tr>
<td>DD</td>
<td>Diagram Definition</td>
</tr>
<tr>
<td>DI</td>
<td>Diagram Interchange</td>
</tr>
<tr>
<td>DMN</td>
<td>Decision Model and Notation</td>
</tr>
<tr>
<td>MDEME</td>
<td>Model Driven Message Interoperability</td>
</tr>
<tr>
<td>MGF</td>
<td>Meta Object Facility</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>PPMN</td>
<td>Provenance and Pedigree Model and Notation</td>
</tr>
<tr>
<td>RFC</td>
<td>Remote Function Call</td>
</tr>
<tr>
<td>SCE</td>
<td>Specification Common Elements</td>
</tr>
<tr>
<td>SDMNDI</td>
<td>Shared Data Model and Notation Diagram Interchange</td>
</tr>
<tr>
<td>SDMN</td>
<td>Shared Data Model and Notation</td>
</tr>
<tr>
<td>SysML</td>
<td>Systems Modeling Language</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>XMI</td>
<td>XML Metadata Interchange</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>

6.6 Structure of this Document

This document provides a brief introduction to SDMN and its purpose (see the section entitled “Overview” for the current edition). The introduction is followed by normative clauses that define the elements of the specification and their properties and associations (see the sections entitled...
6.7 Acknowledgements

The following companies submitted version 1.0 of this specification:

- Auxilium Technology Group, LLC
- BPM Advantage Consulting, Inc.

The following companies supported this specification:

- Adaptive
- agnos.ai UK Ltd
- Airbus Group
- BookZurman, Inc.
- Camunda Services GmbH
- Department of Veterans Affairs
- FICO
- Mayo Clinic
- MDIX, Inc.
- Red Hat
- Thematix Partners, LLC
- Trisotech
- University of Utah
- XZYOS, LLC

Special Acknowledgements

The following persons were members of the core teams that contributed to the content of this specification (in alphabetical order):


7 Overview

The focus of this document is to define the content and structure of [the Shared Data Model and Notation (SDMN)].

A "Shared Data Model" (SDM) is a library collection of data elements and item definitions that supports a set of BPM-Plus (BPM+) BPM-Models (see directly below) that are used together to address a particular business modeling topic. In particular, a Shared Data Model will provide a single source for the definition of all shared data elements that are used across those correlated BPM-Plus models. Thus, the SDM provides a shared, scoped, and focused view that supports mutual interfaces between the models, as well as external data.
7.1 What Constitutes a BPM+ Model?

Three OMG standards – Business Process Model and Notation (BPMN); Case Management Model and Notation (CMMN); and Decision Model and Notation (DMN) – are often used together to model real-world business situations since they provide (for the most part) a good separation of concerns for Process, Case, and Decision. Thus, the three languages are often spoken about and written about in this context. The origins of the BPM+ acronym was to reduce the burden of referring to all three specifications in speech and in print. A single acronym to refer to the three languages is just simpler.

The idea of BPM+ has since expanded to be a business modeling language stack that will gain new standards as they are developed. The standards that fit into that stack will be languages that address additional areas of concerns and can interact with, in one way or another, with at least one of the other BPM+ languages. SDMN is a modeling standard designed to fit into the BPM+ stack. In this context, a Shared Data Model is considered the “fourth pillar” of a BPM+ Knowledge Package. The other three pillars being the BPM+ standards for Process, Case, and Decision. Additionally, new standards are being developed to fit in the BPM+ stack. These include BPM+ Knowledge Package Model and Notation (BKPMN) and Pedigree and Provenance Model and Notation (PPMN).

7.2 Why a Shared Data Model?

Based on experience with the current set of BPM+ standards – BPMN, CMMN, and DMN – the need of a centralized library collection of DataItems and ItemDefinitions was identified (see the use case described in Clause 14.1 as an illustration of the drivers of this need). For example, using BPM+ models to address a large topic, such as the behaviors of a healthcare clinical guideline (e.g., for hypertension or kidney disease) may result in dozens of individual Process, Case, and Decision models. Specific data elements are frequently used by multiple models across the three classes of BPM+ model types (Process, Case, and Decision). To continue the hypertension example, a data element for “blood pressure” may be used within a Process, Case, and/or Decision. To ensure consistency and accuracy across the models of these large topics, the detailed structures (names and types) of the data elements should be synchronized across all the models that use them.

Since the development of the models of these large topics are lengthy and iterative, the detailed structures of the shared data elements are likely to change over time. Experience has shown that synchronizing the changes to data elements across multiple models, multiple times, is a burdensome maintenance requirement.

Thus, a need for a central library collection for the data elements of a BPM+ Knowledge Package Models was identified. This library collection would serve as a central source for the development of data elements that would be referenced by the other BPM+ models. This library should reflect the structure and capabilities of the current BPM+ models data elements. The library should also include a diagram and modeling environment that is consistent with the data representations of the current BPM+ modeling environments to ease the modeling experience as a modeler moves between the respective modeling tools.

In addition to the three new BPM+ standardsSDMN, there is the sixth standard Specification Core Elements (SCE), that provides a set of common modeling language elements, such as root element and basic packaging capabilities. Instead of defining these basic, non-language specific elements within each of the new languages, BPMN, PPNM, and SDMN are built upon the structures provided by SCE. Other BPM+ languages can also use SCE.

The following figure illustrates the relationships between the old and new BPM+ standards.
7.2.1 Use Case: Hello Patient

The BPM+ Health community has been defining Shareable Clinical Pathways by using the current BPM+ standards to define formal and executable versions of current clinical guidelines (e.g., for hypertension, chronic kidney disease, etc.). Current clinical guidelines are usually found in printed or PDF documents and they contain vague and often confusing semantics leading to a great variability in how the guidelines are understood and performed.

This section describes a simple use case that was developed by the BPM+ Health community. At that time there was no concept of a BPM+ Knowledge Package or of a Shared Data Model. The work on this and other use cases was instrumental in identifying the need and requirements for a BPM+ Knowledge Package and a Shared Data Model.
Organizing BPM+ Models (A BPM+ Knowledge Package)

The use case defined the Processes, Cases, and Decision Services that are involved in managing a visit to a doctor's office. Note that these models were intended to be illustrative rather than an official, comprehensive healthcare guideline.

The following table lists the major BPM+ model elements that made up the use case.

Table 1. List of BPM+ Models for the Hello Patient Use Case

<table>
<thead>
<tr>
<th>Cases</th>
<th>Decision Services</th>
<th>Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Hello Patient</td>
<td>1. What is Treatment Plan?</td>
<td>1. Manage Hello Patient Triggers</td>
</tr>
<tr>
<td>2. Perform Examination</td>
<td>2. What is Patient's BMI Category?</td>
<td>2. Evaluate Applicability</td>
</tr>
<tr>
<td></td>
<td>5. Physical Required?</td>
<td>5. Take Vital Signs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. Check out Patient</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7. Update Appointment Information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. Ask Screening Questions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9. Manage Counseling Referral</td>
</tr>
</tbody>
</table>

A larger use case for “Antenatal Care” was developed and contained more models than listed above. For that use case there were 9 Cases, 15 Decision Services, and 28 Processes.

Reviewing one of the models listed in the table above does not provide the overall scope and context of the set of models in the use case. While it may be possible to trace through the connections between the BPM+ models, that tracing still does provide the proper context.

This lack of perspective resulted in a new type of diagram included with the use case. It is referred to as a Knowledge Diagram in this specification. The diagram provides graphical representations of the BPM+ models and draws connectors to represent how the models can be traced through their connections. The following figure displays the Knowledge Diagram for the Hello Patient use case. Note that all the items listed in Table 1 above, have diagram elements associated with them. There are different notations for Processes, Cases, and Decision Services.
Figure 2 — Example of a BPM+ Knowledge Diagram

This diagram is just an example, and the exact notation is not a requirement for this specification, but there are requirements as to the type of elements, and how they are connected, listed below.

Note that the development of the Knowledge Diagram for the use cases was an indication that something else was needed to fully document the contexts of a set of BPM+ models created for a specific topic. This and other factors led to the requirements for a BPM+ Knowledge Package.

Organizing BPM+ Data Elements (A Shared Data Model)

Several elements in BPM+ Models are intended to store or convey data required for the execution of those Models. BPMN has Data Objects, Data Inputs, Data Outputs, Data Stores, and Properties. CMMN has Case File Items. DMN has Information Items that are used for Data Inputs and Decisions. The Hello Patient use case employed many of these types of data elements within its BPM+ models. The following table lists those data elements used within the set of BPM+ models for the Hello Patient use case.
Table 4. List of Data Elements used by the BPM+ Models in the Hello Patient Use Case

<table>
<thead>
<tr>
<th>Cases</th>
<th>Decision Services</th>
<th>Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Encounter</td>
<td>4. BMI Category</td>
<td>4. BMI Category</td>
</tr>
<tr>
<td>5. Exam Data</td>
<td>5. Demographics</td>
<td>5. Demographics</td>
</tr>
<tr>
<td>15. Weight Counseling Referral</td>
<td>15. Treatment Choice</td>
<td>15. Referral</td>
</tr>
<tr>
<td>17. Weight Counseling Referral Choice</td>
<td>17. Treatment Choice</td>
<td>17. Treatment Choice</td>
</tr>
</tbody>
</table>

Note that the data elements listed in bold in the table are those that appear in all three types of BPM+ models. The other data elements appear in at least two of the model types.

The set of data elements listed in the above table reflect those data elements that are necessary for only the context of this use case (Hello Patient). They do not represent all the data elements that a doctor’s office may require for all of its operations – let alone all the data elements required for the healthcare domain. The use case only specified the data elements that are shared across the models for its particular situation. Hence, we refer to sets of data elements used in this way as “Shared Data”.

Since the use case employed all three different types of BPM+ models (Process, Case, and Decision Service), the common data elements of the use case are shared and distributed across the three types of models. While there are some technical differences between how data is structured and used across the BPM+ specifications, at the logical level, they all play the same role within the respective languages. This is evident when a specific conceptual data element (e.g., “Vital Signs and Measures”) can be included in all three BPM+ modeling languages (see figure below). That is, the same data element (and its values during runtime) can be passed from a CMMN Case to a BPMN Process and then be used in a DMN Decision.

![Shared Data Model and Notation (SDMN), v1.0 – beta]
Currently, the same data element has to be defined separately in the tools dedicated to each modeling language. There are no standard mechanisms for sharing data elements across the three types of BPM+ models.

If there are a lot of data elements that are shared between the models of a set of BPM+ Knowledge Package models, the development and maintenance burden for synchronizing the properties of the data elements will be problematic. All of the Hello Patient use date elements were used in at least two types of models. Each time any of the data elements were modified, which can happen multiple times during the BPM+ Knowledge Package development cycle, there would be one or more modifications in the other types of BPM+ models. It would be up to the modeler to ensure that the modifications were made and were consistent.

This maintenance burden was the driver for defining a Shared Data Model, which would be a library collection of data elements that would readily be available for synchronization with the other BPM+ models. That is, the DataItems and ItemDefinitions of the Shared Data Model should share the same characteristics as the data elements of the three BPM+ model data elements. Further, the modeling experience should be very similar across all four models to ease burdens on the modeler.

The Shared Data Model would provide an environment where data elements can be defined and modified in a single location and the changes could be distributed to the other BPM+ models without additional work and vigilance by the modeler. Modeling tools that implement SDMN should provide a diagramming capability that is consistent with how current BPM+ modeling tools represent their data elements. Specifically, the notation for BPM and CMMN data elements are consistent and should be used as the basis for a SDMN diagram. The following figure provides an example of how a SDMN Data Item Diagram could look.
Commented [SW26]: This figure was updated as a resolution for SDMN-83/SDMN-85. Change in containment connector notation.

Commented [SW27]: This figure was replaced as a resolution for issue SDMN-29/SDMN-64.

Commented [SW28]: This figure was updated as a resolution for issue SDMN-29/SDMN-64.
A Shared Data Model would then become another component of a BPM+ Knowledge Package (as shown in Figure 2 above) set of models.

7.3 The Purpose and Use of a Shared Data Model in a BPM+ Knowledge Package

A Shared Data Model serves multiple purposes with a set of BPM+ Knowledge Package models. First, it provides a library collection of DataItems and ItemDefinitions that serve as the source for the data elements of the BPMN, CMMN, and DMN models within the BPM+ Knowledge Package, including:

- BPMN Data Objects, Data Inputs, Data Outputs, and Messages
- CMMN Case File Items
- DMN Data Inputs and Decision Outputs
A Shared Data Model may also serve as a source for BPMN Data Object initialization at the start of a Process. See the section “Pre-Assigning Values for DataItems,” below, for more information.

8 Specification Core Elements

The SDMN specification utilizes (is dependent on) structural elements defined in the Specification Core Elements (SCE) metamodel. This metamodel is defined in a separate specification [OMG doc number bmi-2021-12-09See the SCE specification] and contains a set of basic metamodel classes that are common to BKPMN, PPMM, and SDMN – and potentially other OMG specifications. Details about the elements of the SCE are maintained in a separate document.

As can be seen in the below, SCE defines elements that can be used by any modeling specification – that is, the elements are not specific to any particular area of concern, such as data, process, decision, etc. For example, the SCE Documentation element can be used (and is used) in any modeling specification since it is important to allow modelers to provide documentation about a semantic element they include in a model.

Because SCE defines these elements, SDMN does not have to duplicate them in this specification. SDMN can just create metamodel bindings to the elements in SCE. Thus, throughout this specification, SCE elements will be seen in metamodel diagrams and SDMN elements will be shown as being specializations of those SCE elements. The SCE and SDMN metamodel elements will be identified as described in Section 6.3.

The SCE high-level metamodel defines the basic infrastructure elements of a BPM+ model (see figure below).
Commented [SW31]: This figure was updated by the resolution of issue SDMN-93/SDMN-94. Updating SDMN based on structural changes to SCE.
The SDMN core metamodel defines the basic infrastructure elements of a Shared Data Model. As mentioned in the previous section, SDMN is dependent on SCE [OMG doc number bmi-2021-12-09 see the SCE specification]. This dependency is manifested in multiple SDMN metamodel relationships. For example, the SDMNModel.SharedDataModel.PackageSDMNModelPackage element directly specializes the SCE SCEModelPackage element, thus inheriting all the properties and associations of that element.

The following figure shows the organization of the SDMN metamodel packages.

**Figure 5 — Figure 4**  The Specification Core Elements (SCE) Base Metamodel

9 SDMN Metamodel

The SDMN core metamodel defines the basic infrastructure elements of a Shared Data Model. As mentioned in the previous section, SDMN is dependent on SCE [OMG doc number bmi-2021-12-09 see the SCE specification]. This dependency is manifested in multiple SDMN metamodel relationships. For example, the SDMNModel.SharedDataModel.PackageSDMNModelPackage element directly specializes the SCE SCEModelPackage element, thus inheriting all the properties and associations of that element.

The following figure shows the organization of the SDMN metamodel packages.

Commented [SW32]: This text was updated for the resolution of issue SDMN-69/SDMN-106. Editorial issues.

Commented [SW33]: This text was updated by the resolution of issue SDMN-93/SDMN-94. Updating SDMN based on structural changes to SCE.
Commented [SW34]: This figure was updated for the resolution of Issue SDMN-27/SDMN-86. The SDMN Library was removed.

Commented [SW35]: This figure was updated for the resolution of Issue SDMN-33/SDMN-107. Directly re-use SCEDI.

Commented [SW36]: This figure was also updated for the resolution of Issue SDMN-113/SDMN-114. Merge SDMNIModel and SharedDataModel.
Further, most of the other SDMN elements directly specialize the SCE BaseElement or SCERootElement. These relationships can be seen in the metamodel diagrams in this chapter as well as being identified in the “Generalizations” subsections for the relevant SDMN classes defined in this chapter.

The following figure shows the SDMN core elements metamodel.
Figure 7 — The SDMN Core Metamodel

9.1 Packaging

SDMN extends three of the five main packaging elements of SCE [OMG doc number bmi-2021-12-09]. SCEModelPackage, SCEModel, and SCEDefinitions. See the next three sections.

9.1.1 SDMNM|SharedDataModel|PackageSDMNModelPackage

The SDMNM|SharedDataModel|PackageSDMNModelPackage class is the outermost containing object for all SDMN elements. It defines the scope of visibility and the namespace for all contained elements. The interchange of SDMN files will always be through one or more SDMNM|SharedDataModel|PackageSDMNModelPackage elements. Specifically, an XML file for a SharedDataModel Shared Data Model usually would be appended with a “.sdmn” label.

The ItemDefinition element is directly contained in a SharedDataModel. Other SDMN elements, such as DataItem and Connector, are also included in a SharedDataModel since they subclass the SCE RootElement, which is contained in the SCE Model element. And thus, a SharedDataModel will contain any element that is based on SCE RootElement.

The SDMNM|ModelPackage contains two main elements: SDMNMModel and SDMNMDefinition.

The following figure shows the SDMNM|ModelPackage|Packages|SDMNMModelPackage metamodel.
Generalizations

The `SDMNModelPackage` element inherits the attributes and/or associations of:

- `SCEModelPackage` (see the SCE-SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties

The following table presents the additional attributes and/or associations for `SDMNModelPackage`:

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>expressionLanguage</code></td>
<td>This attribute identifies the formal expression language used in expressions within the elements of this <code>SDMNModelPackage</code>. The value is a URI that identifies the language. The language SHALL be specified in a URI format.</td>
</tr>
<tr>
<td><code>model</code></td>
<td>This the <code>SDMNModel</code> sub-package contained within a <code>SDMNModelPackage</code>. This is a subset of the contained Package association of the <code>SCEModelPackage</code>.</td>
</tr>
</tbody>
</table>
9.1.2 SDMNModel

The SDMNModel element contains the modeling content of an SDMN model (as opposed to the diagram interchange content for the modeling content). The two subpackages for SDMNModel are SDMNDefinitions and SDMNVocabulary.

An SDMNModel is contained within an SDMNModelPackage.

The following figure shows the SDMNModel metamodel:

Commented [SW50]: This text was updated for the resolution of Issue SDMN-33/SDMN-107. directly re-use SCEDI

Commented [SW51]: This text was updated by the resolution of Issue SDMN-93/SDMN-94. Updating SDMN based on structural changes to SCE.

Commented [SW52]: This text was updated for the resolution of Issue SDMN-27/SDMN-86

Commented [SW53]: This figure was updated for the resolution of Issue SDMN-27/SDMN-86. The SDMN Vocabulary Class was removed.
Generalizations
The SDMNModel element inherits the attributes and/or associations of:
- SCEModel (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties
The following table presents the additional attributes and/or associations for SDMNModel:

Table 2 — SDMNModel Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitions: SDMNDefinitions [0..*]</td>
<td>This is a list of the SDMNDefinitions that are included in the SDMNModel.</td>
</tr>
</tbody>
</table>

9.1.3 SDMNDefinitions
Most of the SDMN modeling elements are contained within the SDMNDefinitions sub-package. This includes the two key elements DataItem and ItemDefinition. It is contained within an SDMNModel.

The following figure shows the SDMNDefinitions metamodel.
**Generalizations**

The SDMN/Definitions element inherits the attributes and/or associations of:

- SCE/Definitions (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

**Properties**

The following table presents the additional attributes and/or associations for SDMN/Definitions:

**Table 3. SDMN/Definitions Attributes and/or Associations**

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataItem : DataItem [0..*]</td>
<td>This is a list of the DataItems that are included in the SDMN/Definitions. See the section entitled “DataItems,” below, for more information about DataItems.</td>
</tr>
<tr>
<td>dataState : DataState [0..*]</td>
<td>This is a list of the potential DataStates that can be associated with a DataItem.</td>
</tr>
<tr>
<td>itemFormat : ItemFormat [0..*]</td>
<td>A list of potential ItemFormats for DataItems. This will apply mainly to electronic documents (such as .pdf).</td>
</tr>
</tbody>
</table>

Commented [SW55]: DataState removed from table for Issue SDMN-3/SDMN-52

Commented [SW56]: ItemFormat removed from table for Issue SDMN-62/SDMN-63
9.2 **SDMN Vocabularies**

Vocabularies (lists of terms) can be added to an SDMNModel. SDMN Vocabularies are sets of terms that can be defined by an external ontology. The terms link to formal definitions for the model elements that are created by the modeling language. The SemanticReference element is used to name the term and provide a link to the definitions. These terms/definitions can then be associated with the appropriate model elements. SDMN Vocabularies are contained within an SDMNModel.

The figure below displays the SDMN Vocabulary metamodel (including the two predefined instances for SDMN Vocabularies):

![SDMN Vocabulary Metamodel](image)

**Figure 11 – The SDMN Vocabulary Metamodel**

### 9.2.1 **SDMN Vocabulary**

An SDMN Vocabulary is a list of terms, through the SemanticReference element, that can be used to relate to model elements to the external definition or meaning. The terms themselves do not represent the definitions or meanings but provide links to an external source. Multiple SDMN Vocabularies can be defined. They are contained in an SDMN Model.

Further, SDMN Vocabularies can be used for creating a user-defined list of enumerated values for use within a SDMN (as opposed to a fixed enumeration list). It is up to the SDMN modeling tool to organize the SDMN Vocabularies into the appropriate enumerated lists. Since the SemanticReference element has a name and the
links to external definitions are optional, the list (the “enumeration” SDMN Vocabularies) can be created before the specific external definitions are established.

SDMN has two pre-defined SDMN Vocabularies for the enumerated terms for the ItemKind element (see the section entitled “ItemKind” for more information) and the MultiplicityKind (see the section entitled “MultiplicityKind” for more information).

Generalizations
The SDMN vocabulary element inherits the attributes and/or associations of
- SCE Vocabulary (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties
The SDMN vocabulary element does not have any additional attributes and/or associations.

10 SDMN Model Elements
This chapter defines DataItem and its related elements and ItemDefinition and its related elements.

10.1 DataItems
Several elements in BPM+ Models are intended to store or convey data required for the execution of those Models. BPMN has Data Objects, Data Inputs, Data Outputs, Data Stores, and Properties. CMMN has Case File Items. DMN has Information Items that are used for Data Inputs and Decisions. While there are some technical differences between how data is structured and used across the BPM+ specifications, at the logical level, they all play the same role within the respective languages. This is evident when a specific conceptual data element (e.g., “Invoice”) can be included in all three BPM+ modeling languages. That is, the same data element can be passed from a CMMN Case to a BPMN Process and then used in a DMN Decision. Currently, the same logical data element has to be defined separately in each modeling language. If there are a lot of data elements that are shared between the models of a BPM+ Knowledge Package, the development and maintenance burden for synchronizing the properties of the data elements will problematic. This is the driver for defining a DataItem.

Thus, a SDMN DataItem represents a common definition and structure for the data handling elements of the other BPM+ models.

A DataItem may represent a piece of information of any nature, ranging from unstructured to structured, and from simple to complex, which information can be defined based on any information modeling “language.” A DataItem can be anything from a folder or document stored with CMIS, an entire folder hierarchy referring or containing other DataItems, or simply an XML document with a given structure. The structure, as well as the “language” (or format) to define the structure, is defined by the associated ItemDefinition (see below). This may include the definition of properties (“metadata”) of a DataItem, which is only applied to CMMN Case File Items. If the internal content of the DataItem is known, an XML Schema, describing the DataItem, may be imported.

To support CMMN CaseFileItems, DataItems can be organized into arbitrary hierarchies either by containment or by composition.

For containment hierarchies the attributes children and parent are used whereas for reference hierarchies the associations targetRefs and sourceRef are used. For example, a folder hierarchy can be implemented by using a CaseFileItemDefinition definitionType of CMISFolder, and using children and parent CaseFileItems as the folder structure. The resulting hierarchy can include metadata for each folder represented by the properties as defined by the associated CaseFileItemDefinition. Case file items can be used to represent arbitrary content. For example, documents can be implemented by using CaseFileItemDefinition definitionType of CMISDocument. There is no need to know the internals of those content objects, but if the internals of the object are known, the XML Schema can be defined by the Import class (see 5.1.3) of the CaseFileItemDefinition. The document or content object can include metadata as well, as represented by the properties as defined by the associated
Several elements in BPMN are subject to store or convey items during process execution. These elements are referenced generally as “item-aware elements.” This is similar to the variable construct common to many languages. As with variables, these elements have an ItemDefinition.

The data structure these elements hold is specified using an associated ItemDefinition. A DataItem ItemAwareElement MAY be underspecified, meaning that the structure attribute of its ItemDefinition is optional if the modeler does not wish to define the structure of the associated data. The elements in the BPMN specification defined as item-aware elements are: Data Objects, Data Object References, Data Stores, Properties, DataInputs and DataOutputs.

The following figure shows the metamodel elements related to the DataItem element.

Commented [SW61]: This text was updated for the resolution of Issue SDMN-69/SDMN-106. Editorial changes. Removing extraneous text.

Commented [SW62]: This text was updated for the resolution of Issue SDMN-69/SDMN-106. Editorial changes.

Commented [SW63]: This figure was updated by the resolution of Issue SDMN-93/SDMN-94. Updating SDMN based on structural changes to SCE.

Commented [SW64]: This figure was updated by the following issues:
SDMN-3/SDMN-52 - removal of Data State
SDMN-4/SDMN-53 - removal of Location
SDMN-62/SDMN-63 - removal of ItemFormat
10.1.1 DataItem

A SDMN DataItem represents a common definition and structure for the data handling elements of the other BPM+ models (as described above). It is contained within a [Core/Definitions/SharedDataModel] notation.

**Notation**

The following statements define the notation for a DataItem:

- A DataItem is a shape that SHALL be a document shape with folded upper right corner and drawn with a single line (see below).

The use of text, color, size, and lines for a [Reference Connector/DataItem] SHALL follow the rules defined in the section entitled “Use of Text, Color, Size, and Lines in a Diagram” above.

The DataItem shape is a document with folded upper right corner (see figure below). This is the default notation for a DataItem and occurs when the itemKind property of the ItemDefinition assigned to the DataItem is set to anything other than folder, which has a different notation as shown below.

**Figure 13 – Figure 7 – The DataItem Metamodel**

Commented [SW65]: This figure was also updated for the resolution of Issue SDMN-113/SDMN-114. Merge SDMModel and SharedDataModel.

Commented [SW66]: This figure was updated as a resolution for Issue SDMN-70/SDMN-71. DataItem is now a subclass of SCEElement.

Commented [SW67]: This figure was also updated for the resolution of Issue SDMN-56/SDMN-87. Update DMN Dependencies

Commented [SW68]: This figure was also updated for the resolution of Issue SDMN-8/SDMN-66. The MultiplicityKind class was removed and replaced by an attribute, the isCollection attribute was added.

Commented [SW69]: This figure was also updated for the resolution of Issue SDMN-130/SDMN-132. Move itemKind to Dataitem.

Commented [SW70]: This figure was also updated for the resolution of Issue SDMN-133/SDMN-134. remove multiplicityKind from ItemDefinition

Commented [SW71]: This text was updated by the resolution of Issue SDMN-93/SDMN-94. Updating SDMN based on structural changes to SCE.

Commented [SW72]: This text was updated for the resolution of Issue SDMN-69/SDMN-106. Editorial Issues.

Commented [SW73]: This text was updated for the resolution of Issue SDMN-130/SDMN-132. Move itemKind to Dataitem.
Figure 14 - Figure 8 - A DataItem Object

The DataItem shape is a folder when the itemKind property of the ItemDefinition assigned to the DataItem is set to folder. (see figure below).

Figure 15 - Figure 9 - A DataItem Object

Generalizations

The DataItem element inherits the attributes and/or associations of:

- SCE.RootSCEElement[ElementType][see the SCE Specification for more information][OMG doc number bmi-2021-12-09]

Properties

The following table presents the additional attributes and/or associations for DataItem:

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataItemRef: QName [0..1]</td>
<td>A reference to an external DataItem that is imported into this Shared Data Model. The DataItem and its details can only be viewed in this model. Any changes to the original SHALL be carried out in the source Shared Data Model. A DataItem can have only one of dataItemRef or ItemDefinitionRef as a set attribute. Neither of them is required, though. If a dataItemRef is defined, then the graphical notation for the DataItem will include a locked icon.</td>
</tr>
<tr>
<td>dataStateRef: DataState [0..*]</td>
<td>A DataItem can have multiple DataStates, which represent significant states in its lifecycle. The DataStates of a DataItem may show up as Milestones within a CMMN Case.</td>
</tr>
<tr>
<td>formatRef: ItemFormat [0..1]</td>
<td>A list of potential ItemFormats for the DataItem. This will apply mainly to electronic documents (such as pdf).</td>
</tr>
<tr>
<td>locationRef: Location [0..*]</td>
<td>A list of potential Locations for the DataItem.</td>
</tr>
</tbody>
</table>

Commented [SW74]: This text was updated for the resolution of Issue SDMN-130/SDMN-132. Move itemKind to DataItem.

Commented [SW75]: This text was updated for the resolution of Issue SDMN-130/SDMN-132. Move itemKind to DataItem.

Commented [SW76]: This figure was updated as a resolution for Issue SDMN-29/SDMN-64.

Commented [SW77]: This text was updated by the resolution of Issue SDMN-91/SDMN-94. Updating SDMN based on structural changes to SCE.

Commented [SW78]: This text was updated as a resolution for Issue SDMN-70/SDMN-71.

Commented [SW79]: This text was updated for the resolution of Issue SDMN-69/SDMN-106. Editorial Issues.

Commented [SW80]: dataStateRef removed from table for Issue SDMN-3/SDMN-52.

Commented [SW81]: formatRef removed from table for Issue SDMN-62/SDMN-63.

Commented [SW82]: locationRef removed from table for Issue SDMN-4/SDMN-53.
**isCollection**: Boolean [0..1] = false

Defines if the `DataItem` represents a collection of elements. It is not needed when an `itemDefinition` is referenced. If an `itemDefinition` is referenced, then this attribute MUST have the same value as the `isCollection` attribute of the referenced `itemDefinition`. The default value for this attribute is false.

**ItemKind**: String [0..1] default: Information

This defines the nature of the `DataItem`. Possible values are physical, information, conceptual, and others (see the table entitled “ItemKind Values”). The default value is Information.

**metaDefinitionRef**: `ItemDefinition` [0..1]

A reference to an `itemDefinition` that defines the `Properties` of the `DataItem`. The `itemComponents` of the `ItemDefinition` structure map to the `Properties` of a `CMMN Case File Item`. Each of the `itemComponents` SHALL be a simple type.

**multiplicityKind**: `MultiplicityKind` String [0..1] default: ExactlyOne

This attribute sets the multiplicity of the `DataItem`. The default is ExactlyOne. See the table entitled “MultiplicityKind Values” below, for the entire set of values. This attribute SHALL have the same value as the `multiplicity` attribute of the associated `ItemDefinition`, if there is one.

If the multiplicity is set to `ZeroOrMore`, or `OneOrMore`, then the graphical notation for the `DataItem` will include a Collection (multi-instance) icon.

**preAssignment**: `Assignment` [0..1]

Specifies an optional pre-assignment DMN Expression. The expression will provide values for one or more of the simple type `itemComponents` of the `ItemDefinition` set for the `DataItem`.

**typeDefinitionRef**: `ItemDefinition` [0..1]

A reference to an `itemDefinition` that defines the detailed structure, which can be simple or complex, of the `DataItem`. A `DataItem` can have only one of `dataItemRef`, or `typeDefinitionRef` as a set attribute. None of them are required, though.

---

### ItemKinds

The possible values of the `ItemKind` attribute support the BPMN, CMMN, and possible future BPM+ specifications. DMN does not include an `ItemKind` attribute and thus it should be ignored by DMN models. See Sections 11.2 and 11.3 for mappings of the values presented below to the BPMN and CMMN specifications.

The following table presents a description for the possible values for `ItemKind`:

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual</td>
<td>The type of the <code>DataItem</code> that doesn’t represent data or physical items, but represents concepts in the minds of users that are important for tasks or decisions. For example, a preference for a particular type of procedure will influence a doctor’s decision. While actual computations cannot be made with Conceptual <code>DataItem</code>, they are used to document aspects of the modeled behavior.</td>
</tr>
<tr>
<td>DataType</td>
<td>The type of the DataItem that fully utilizes the structural data capabilities inherent to ItemDefinition. If the DataItem has a TypeDefinitionRef (to an ItemDefinition), then the value of ItemKind MUST be Information.</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Document</td>
<td>This represents a Data Object or Case File Item that is a type of Document. In BPMN, the document could be physical (e.g., printed) or electronic. In CMMN, it would represent a document in a Document Management System and is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/CMISDocument">http://www.omg.org/spec/CMMN/DefinitionType/CMISDocument</a></td>
</tr>
<tr>
<td>Folder</td>
<td>This represents a CMMN Case File Item that is a Folder. A Folder can contain other Folders or Documents. Neither BPMN nor DMN have the concept of Folder as a data element. Thus, DataItems based that is a Folder would not map to BPMN or DMN data elements. The Folder is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/CMISFolder">http://www.omg.org/spec/CMMN/DefinitionType/CMISFolder</a></td>
</tr>
<tr>
<td>Physical</td>
<td>The ItemKind is represents objects in a BPMN Process that are physical objects, such as printed documents or manufactured items. These types of DataItems are not currently relevant to CMMN or DMN.</td>
</tr>
<tr>
<td>Relationship</td>
<td>The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/CMISRelationship">http://www.omg.org/spec/CMMN/DefinitionType/CMISRelationship</a></td>
</tr>
<tr>
<td>UMLClass</td>
<td>The ItemKind is represents a UML Class in a Class Diagram.</td>
</tr>
<tr>
<td>Unknown</td>
<td>The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/Unknown">http://www.omg.org/spec/CMMN/DefinitionType/Unknown</a></td>
</tr>
<tr>
<td>Unspecified</td>
<td>The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/Unspecified">http://www.omg.org/spec/CMMN/DefinitionType/Unspecified</a></td>
</tr>
<tr>
<td>WSDLMessage</td>
<td>The ItemKind is represents a WSDL Message.</td>
</tr>
<tr>
<td>XSDComplexType</td>
<td>For ItemKinds of this type, the (SCE) Import class SHOULD be used to import an XML Schema definition into the Shared Data Model. The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/XSDComplexType">http://www.omg.org/spec/CMMN/DefinitionType/XSDComplexType</a></td>
</tr>
<tr>
<td>XSDElement</td>
<td>For ItemKinds of this type, the (SCE) Import class SHOULD be used to import an XML Schema definition into the Shared Data Model. The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/XSDElement">http://www.omg.org/spec/CMMN/DefinitionType/XSDElement</a></td>
</tr>
<tr>
<td>XSDSimpleType</td>
<td>For ItemKinds of this type, the (SCE) Import class SHOULD be used to import an XML Schema definition into the Shared Data Model. The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/XSDSimpleType">http://www.omg.org/spec/CMMN/DefinitionType/XSDSimpleType</a></td>
</tr>
</tbody>
</table>

Commented [SW88]: This text was updated for the resolution of Issue SDMN-129/SDMN-130. Editorial Issues. The child and parent attributes mentioned if Folder do not exist.

Commented [SW90]: This section was moved ItemDefinition for the resolution of Issue SDMN-130/SDMN-132. Move ItemKind to DataItem. Additional text changes were made as necessary, such as replacing ItemDefinition with DataItem.
**MultiplicityKinds**

The MultiplicityKind attribute is included in SDMN to support the DefinitionType attribute of CMMN CaseFileItemDefinition Elements. The values listed in the table below are the same values as defined by CMMN. The MultiplicityKind attribute does not map to any attribute of BPMN and DMN and thus, should be ignored by those types of models. BPMN and DMN both use an isCollection attribute to determine multiplicity.

10.1.2 DataItemRelationship

An DataItemRelationship is used to define a relationship between DataItems. This relationship will specify that one DataItem is connected in some way to another DataItem—there is some type of relationship. This relationship is included to support the source and target reference associations that exists between CMMN CaseFileItems. For example, the CaseFileItems might be created at the same time (although independently) or they are often sent together during the performance of a CMMN Case, etc. In a sense, it is a non-graphical way of grouping DataItem.

They are contained in the SDMN/Definitions package.

The following table presents a description for the possible values for MultiplicityKind:

<table>
<thead>
<tr>
<th>MultiplicityKind</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>null</strong></td>
<td>Not applicable</td>
</tr>
<tr>
<td><strong>1,0</strong></td>
<td>Represents an exact multiplicity of 1</td>
</tr>
<tr>
<td><strong>0,1</strong></td>
<td>Represents an exact multiplicity of 0 or 1</td>
</tr>
<tr>
<td><strong>0,*,1</strong></td>
<td>Represents a minimum multiplicity of 0 and a maximum multiplicity of 1</td>
</tr>
<tr>
<td><strong>0,*,n</strong></td>
<td>Represents a minimum multiplicity of 0 and a maximum multiplicity of n</td>
</tr>
<tr>
<td>*<strong>,1</strong></td>
<td>Represents a minimum multiplicity of 1 and a maximum multiplicity of n</td>
</tr>
<tr>
<td>*<strong>,n</strong></td>
<td>Represents a minimum multiplicity of 1 and a maximum multiplicity of n</td>
</tr>
</tbody>
</table>

Generalizations

The DataItemRelationship element inherits the attributes and/or associations of:
- ElementRelationshipType (see the SCE Specification for more information [OMG doc number bmi-2021-12-01]).

Properties

The following table presents a description for the possible values for MultiplicityKind:

<table>
<thead>
<tr>
<th>MultiplicityKind</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>null</strong></td>
<td>Not applicable</td>
</tr>
<tr>
<td><strong>1,0</strong></td>
<td>Represents an exact multiplicity of 1</td>
</tr>
<tr>
<td><strong>0,1</strong></td>
<td>Represents an exact multiplicity of 0 or 1</td>
</tr>
<tr>
<td><strong>0,*,1</strong></td>
<td>Represents a minimum multiplicity of 0 and a maximum multiplicity of 1</td>
</tr>
<tr>
<td><strong>0,*,n</strong></td>
<td>Represents a minimum multiplicity of 0 and a maximum multiplicity of n</td>
</tr>
<tr>
<td>*<strong>,1</strong></td>
<td>Represents a minimum multiplicity of 1 and a maximum multiplicity of n</td>
</tr>
<tr>
<td>*<strong>,n</strong></td>
<td>Represents a minimum multiplicity of 1 and a maximum multiplicity of n</td>
</tr>
</tbody>
</table>

Commented [SW91]: Section 10.1.2 “DataItemRelationship” with its subsections, figure, and table have been removed as a resolution for Issue SDMN-1/SDMN-34
### Table 8. MultiplicityKind Values

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExactlyOne</td>
<td>There is one copy of this DataItem. The source DataItem, for reference hierarchies of a DataItem, refers to the source of the DataItem. If DataItem b is a target of DataItem a, then source of DataItem b is a. This redefines the sourceRef association inherited from SCE:ElementRelationshipType.</td>
</tr>
<tr>
<td>OneOrMore</td>
<td>There is at least one copy of this DataItem, but there may be more.</td>
</tr>
<tr>
<td>Unknown</td>
<td>The multiplicity is not known for this DataItem.</td>
</tr>
<tr>
<td>Unspecified</td>
<td>The multiplicity is not specified for this DataItem.</td>
</tr>
<tr>
<td>ZeroOrMore</td>
<td>There may be no copies of this DataItem or there may be multiple copies.</td>
</tr>
<tr>
<td>ZeroOrOne</td>
<td>There may be no copies of this DataItem or there may be one copy.</td>
</tr>
</tbody>
</table>

---

**DataState**

DataItems can optionally reference a DataState element, which is the state of the data contained in the DataItem. The definition of these DataStates, e.g., possible values and any specific semantic, are out of scope of this specification. Therefore, SDMN adopters can use the DataState element and the SDMN extensibility capabilities to define their DataStates.

**Generalizations**

The DataState element inherits the attributes and/or associations of:

- ElementType (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

**Properties**

The DataState element does not have any additional attributes and/or associations.

---

**10.1.3 ItemFormat**

Represents the format of a DataItem. It can be something as simple as “mime types” or the specification of a format documented in a formal format registry. ItemFormats are contained within a SDMN Definitions and can be referenced by DataItems.

**Generalizations**

The ItemFormat element inherits the attributes and/or associations of:

- ElementType (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).
Properties

The following table presents the additional attributes and/or associations for ItemFormat:

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formatDefinitionRef</td>
<td>URI [0..*] The identifier of the format within the specified format registry. For example, &quot;dicom&quot; if the registry is that of W3C mime types. This is not the usual &quot;id&quot; found commonly in this specification. This is a &quot;stringified&quot; (if necessary) unique id in the context of the .formatRegistry.</td>
</tr>
</tbody>
</table>

10.1.4 Locations

The Locations package contains elements related to physical or virtual locations. Organizations may deem the locations at which a DataItem may exist to be of significance. Locations are often tracked in the context of pedigree and provenance.

10.1.4.1 Location

Location is an abstract class where its concrete specializations identify a particular place or position. Locations are contained within a SDMN/Definitions and can be referenced by DataItems.

The following figure shows the metamodel elements related to the Location element.

![Figure 17 - The Location Metamodel](image)

Generalizations

The Location element inherits the attributes and/or associations of:
- ElementType (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties

The following table presents the additional attributes and/or associations for Location:

...
Table 6. Location Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>A description of the Location</td>
</tr>
</tbody>
</table>

10.1.4.2 GeospatialExtent
A location that is a volume in the world such as a container or a room.

Generalizations
The GeospatialExtent element inherits the attributes and/or associations of:
- Location (see the section entitled “Location” for more information).
  Further, the Location element inherits the attributes and/or associations of:
  - ElementType (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties
The GeospatialExtent element does not have any additional attributes and/or associations.

10.1.4.3 NetworkAddress
The address of an element or node on a network.

Generalizations
The NetworkAddress element inherits the attributes and/or associations of:
- Location (see the section entitled “Location” for more information).
  Further, the Location element inherits the attributes and/or associations of:
  - ElementType (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties
The NetworkAddress element does not have any additional attributes and/or associations.

10.1.4.4 PhysicalAddress
A physical location in the real world.

Generalizations
The PhysicalAddress element inherits the attributes and/or associations of:
- Location (see the section entitled “Location” for more information).
  Further, the Location element inherits the attributes and/or associations of:
  - ElementType (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties
The PhysicalAddress element does not have any additional attributes and/or associations.
10.1.4.5 SpaceTime

A Location at a particular point in time.

Generalizations

The SpaceTime element inherits the attributes and/or associations of:

- Location (see the section entitled “Location” for more information).

Further, the Location element inherits the attributes and/or associations of:

- ElementType (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties

The following table presents the additional attributes and/or associations for SpaceTime:

Table 7. SpaceTime Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endTime: Date [1]</td>
<td>The ending time of the SpaceTime.</td>
</tr>
</tbody>
</table>

10.1.5 Pre-Assigning Values for DataItems

There are situations in the development of a collection of BPM+ Knowledge Package models when the values of some of the DataItem properties are known. For example, in a healthcare scenario, certain medications are recommended for a particular condition. Each medication will have a representative DataItem in the Shared Data Model that will share the same ItemDefinition. The ItemDefinition will define the properties that are needed for prescribing the medication, such as medication name, codes, dosages, etc.

Assignment

An Assignment is contained within a DataItem or a DataAssociation.

Generalizations

The Assignment element inherits the attributes and/or associations of:

- SCE:SCERootElementType (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties

The following table presents the additional attributes and/or associations for Assignment:

Table 9. Assignment Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value: Expression [1]</td>
<td>The DMN Expression that evaluates the Assignment.</td>
</tr>
</tbody>
</table>
Pre-Assignment Example

The following example is a FEEL expression that preassigns values for a “medication” DataItem.

```
{
    "Metoprolol Tartrate 25": {
        Id: "metoprololTartrate25Medication",
        Code: {
            Code: "866426"
        },
        Text: "Metoprolol Tartrate 25 MG"
    },
    Form: {
        Coding: {
            System: "http://snomed.info/sct",
            Code: "385055001",
            Display: "Tablet dose form"
        },
        Text: "Tablet dose form"
    },
    Ingredient: {
        Substance: {
            Id: "metoprololTartrate25Substance",
            Code: {
                Coding: {
                    Code: "6918"
                },
                Text: "Metoprolol"
            }
        }
    }
}
```

10.2 Item Definitions

The ItemDefinition element is the mechanism for providing the data structure of DataItems. The following figure shows the metamodel elements related to the ItemDefinition element.
Commented [SW100]: This figure was updated by the resolution of Issue SDMN-93/SDMN-94. Updating SDMN based on structural changes to SCE.

Commented [SW101]: This figure was updated for the resolution of Issue SDMN-7/SDMN-65. ItemKind changed to a string attribute.

Commented [SW102]: This figure was also updated for the resolution of Issue SDMN-56/SDMN-87. Update DMN Dependencies

Commented [SW103]: This figure was also updated for the resolution of Issue SDMN-8/SDMN-66. MultiplicityKind changed to a string attribute.

Commented [SW104]: This figure was also updated for the resolution of Issue SDMN-113/SDMN-114. Merge SDMNMModel and SharedDataModel.
10.2.1 ItemDefinition

The ItemDefinition element is the mechanism for providing the data structure of DataItems. It is contained within a SDMN:Definitions:SharedDataModel.

Notation

The following statements define the notation for an ItemDefinition:

- A ItemDefinition is a shape that SHALL be a boxes with two or more sections.
  - The top section will display the name of the ItemDefinition.
  - The bottom section(s) will display a type or a list of types.

The use of text, color, size, and lines for an ItemDefinition SHALL follow the rules defined in the section entitled “Use of Text, Color, Size, and Lines in a Diagram” above.

If the ItemDefinition is a simple type (e.g., has no ItemComponent), the ItemDefinition will be shown as in the following figure:

Label

<table>
<thead>
<tr>
<th>typeRef</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowedValues</td>
</tr>
</tbody>
</table>

Figure 11 - An ItemDefinition Object with no ItemComponent

If the ItemDefinition is a complex type (e.g., has one or more ItemComponent), the ItemDefinition will be shown as in the following figure. Additional paired sections will be added to the bottom of the shape for each ItemComponent that makes up the ItemDefinition.

Commented [SW105]: This figure was also updated for the resolution of Issue SDMN-130/SDMN-132. Move ItemKind to DataItem.

Commented [SW106]: This figure was also updated for the resolution of Issue SDMN-130/SDMN-132. Move ItemKind to DataItem.

Commented [SW107]: This text was updated by the resolution of Issue SDMN-93/SDMN-94. Updating SDMN based on structural changes to SCE.
Figure 12 - An ItemDefinition Object with one or more ItemComponents

ItemDefinitions can be connected together through composition and reference relationships. The following figure displays an example of a composition indicator. It shares the line style of the CompositionConnector (see below) since it has the same basic semantic meaning.

Figure 13 - An ItemDefinition Composition Indicator

The connection rules for a composition indicator are as follows:

- The source of a CompositionConnector SHALL be an ItemDefinition.
- The source end of the CompositionConnector SHALL be attached to the right boundary of the type section of the sub-element for the source ItemDefinition.
- The target of a CompositionConnector SHALL be an ItemDefinition.
- The target end of the CompositionConnector SHALL be attached to the top-level name section for the target ItemDefinition.

The following figure displays an example of a reference indicator. It shares the line style of the ReferenceConnector (see below) since it has the same basic semantic meaning.

Figure 14 - An ItemDefinition Reference Indicator

The connection rules for a reference indicator are as follows:

- The source of a ReferenceConnector SHALL be an ItemDefinition.
- The source end of the ReferenceConnector SHALL be attached to the right boundary of the type section of the sub-element for the source ItemDefinition.
- The target of a ReferenceConnector SHALL be an ItemDefinition.
- The target end of the ReferenceConnector SHALL be attached to the top-level name section for the target ItemDefinition.

Generalizations

The ItemDefinition element inherits the attributes and/or associations of:

- DMN ItemDefinition (see the DMN 1.4 or later 1.X specification for more information [OMG doc number formal-2023-0320211-018-v01])

Properties

The following table presents the additional attributes and/or associations for ItemDefinition:
The specific context of the BPM+ elements may result in different terminology or sub-sets of data representation elements within the normative domain models. To reduce any confusion due to terminology or data representation, the capability of linking model elements to the appropriate external sources of truth for their domain is provided (i.e., a conceptReference). Other SDMN elements receive this attribute by inheriting it from SCE.SCE.BaseElement. However, since ItemDefinition derives from DMN, which does not have the attribute, it is added here. It is expected that the value of the URI will be persistent.

A reference to an external ItemDefinition that is imported into this Shared Data Model. The ItemDefinition and its details can only be viewed in this model. Any changes to the original SHALL be carried out in the source Shared Data Model. Other types of structures are not allowed for the SDMN. However, BPM Data Objects and CMMN Case File Items have the capability of references other types of structures. These other types of structures would not be a part of the SDMN Shared Data Model.

This defines the nature of the DataItem. Possible values are physical, information, conceptual, and others (see the section entitled "ItemKind."). The default value is information. If the ItemDefinition has ItemComponents or ItemComponentRefs, then the ItemKind for each of these sub-ItemDefinitions SHALL match the top-level ItemDefinition.

This sets the multiplicity of the ItemDefinition. The default is ExactlyOne. This attribute SHALL have the same value as the multiplicity attribute of the associated DataItem. This attribute redefines the isCollection attribute of the DMN ItemDefinition.

A ItemDefinition can include multiple SCE.SemanticReference elements. This attribute was added because ItemDefinition is based on the DMN.ItemDefinition, which is not based on the SCE specification and thus, does not have a built-in SemanticReference as part of its definition. See the section entitled "Semantic Reference" in the SCE specification for more information.

The type of the ItemDefinition that doesn’t represent data of physical items, but represents concepts in the minds of users that are important for tasks or decisions. For example, a preference for a particular type of procedure will influence a doctor’s decision. While actual computations cannot be made with Conceptual ItemDefinitions, they are used to document aspects of the modeled behavior.
<table>
<thead>
<tr>
<th><strong>DataType</strong></th>
<th>The type of the ItemDefinition that fully utilizes the structural data capabilities inherent to ItemDefinition. Using FEEL, these will define simple types or data structures.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Document</strong></td>
<td>This represents a Data Object or Case File Item that is a type of Document. In BPMN, the document could be physical (e.g., printed) or electronic. In CMMN, it would represent a document in a Document Management System and is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/CMISDocument">http://www.omg.org/spec/CMMN/DefinitionType/CMISDocument</a></td>
</tr>
<tr>
<td><strong>Folder</strong></td>
<td>This represents a CMMN Case File Item that is a Folder. A Folder can contain other Folders or Documents. These relationships are set through the Child and Parent attributes of the DataItem. Neither BPMN nor DMN have the concept of Folder as a data element. Thus, DataItems based on a Folder ItemDefinition would not map to BPMN or DMN data elements. The Folder is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/CMISFolder">http://www.omg.org/spec/CMMN/DefinitionType/CMISFolder</a></td>
</tr>
<tr>
<td><strong>Physical</strong></td>
<td>The ItemKind is represents objects in a BPMN Process that are physical objects, such as printed documents or manufactured items. These types of DataItems are not currently relevant to CMMN or DMN.</td>
</tr>
<tr>
<td><strong>Relationship</strong></td>
<td>The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/CMISRelationship">http://www.omg.org/spec/CMMN/DefinitionType/CMISRelationship</a></td>
</tr>
<tr>
<td><strong>UMLClass</strong></td>
<td>The ItemKind is represents a UML Class in a Class Diagram.</td>
</tr>
<tr>
<td><strong>Unknown</strong></td>
<td>The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/Unknown">http://www.omg.org/spec/CMMN/DefinitionType/Unknown</a></td>
</tr>
<tr>
<td><strong>Unspecified</strong></td>
<td>The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/Unspecified">http://www.omg.org/spec/CMMN/DefinitionType/Unspecified</a></td>
</tr>
<tr>
<td><strong>WSDLMessage</strong></td>
<td>The ItemKind is represents a WSDL Message.</td>
</tr>
<tr>
<td><strong>XSDComplexType</strong></td>
<td>For ItemKinds of this type, the (SCE) Import class SHOULD be used to import an XML Schema definition into the Shared Data Model. The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/XSDComplexType">http://www.omg.org/spec/CMMN/DefinitionType/XSDComplexType</a></td>
</tr>
<tr>
<td><strong>XSDElement</strong></td>
<td>For ItemKinds of this type, the (SCE) Import class SHOULD be used to import an XML Schema definition into the Shared Data Model. The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/XSDElement">http://www.omg.org/spec/CMMN/DefinitionType/XSDElement</a></td>
</tr>
<tr>
<td><strong>XSDSimpleType</strong></td>
<td>For ItemKinds of this type, the (SCE) Import class SHOULD be used to import an XML Schema definition into the Shared Data Model. The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/XSDSimpleType">http://www.omg.org/spec/CMMN/DefinitionType/XSDSimpleType</a></td>
</tr>
</tbody>
</table>
### 10.2.2 ItemKind

This class is a type of SemanticReference that serves as the terms for an SDMN Vocabulary that is used to specify the kind of multiplicity that exists for an ItemDefinition. Instead of being defined as a fixed enumerated list, the kinds can be defined through a class (ItemKind) and instances of that class (as shown below). The instances defined in the SDMN Library SHALL be included in any SDMN implementation. However, the implementation can allow additional instances of this class if required for a particular modeling situation (see the section entitled “MultiplicityKinds” for more information). Some of the literals for ItemKind are based on the CMMN CaseFileItemDefinition literals for DefinitionType.

In practice, when a modeler creates a model with an ItemDefinition, the ItemKind will be instantiated by one of the 13 instances in the Library. The ItemKind identifies the different natures that ItemDefinitions, and thus the DataItems that reference the ItemDefinitions, may represent.

The following figure shows the metamodel elements related to the ItemKind element (which includes the standard set of instances provided by the SDMN Library).

![Diagram of ItemKind Metamodel](image)

**Figure 18 – The ItemKind Metamodel**

**Generalizations**

The ItemKind element inherits the attributes and/or associations of...
Properties
The ItemKind element does not have any additional attributes and/or associations.

Standard Terms Vocabulary
The following table presents a description for the included instances for ItemKind.

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual</td>
<td>The type of the ItemDefinition that doesn’t represent data or physical items, but represents concepts in the minds of users that are important for tasks or decisions. For example, a preference for a particular type of procedure will influence a doctor’s decision. While actual computations cannot be made with Conceptual ItemDefinitions, they are used to document aspects of the modeled behaviors.</td>
</tr>
<tr>
<td>DataType</td>
<td>The type of the ItemDefinition that fully utilizes the structural data capabilities inherent to ItemDefinition. Using FEEL, these will define simple types or data structures.</td>
</tr>
<tr>
<td>Document</td>
<td>This represents a Data Object or Case File Item that is a type of Document. In BPMN, the document could be physical (e.g., printed) or electronic. In CMMN, it would represent a document in a Document Management System and is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/CMISDocument">http://www.omg.org/spec/CMMN/DefinitionType/CMISDocument</a></td>
</tr>
<tr>
<td>Folder</td>
<td>This represents a CMMN Case File Item that is a Folder. A Folder can contain other Folders or Documents. These relationships are set through the Child and Parent attributes of the DataItem. Neither BPMN nor DMN have the concept of Folders as a data element. Thus, DataItems based on a Folder ItemDefinition would not map to BPMN or DMN data elements. The Folder is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/CMISFolder">http://www.omg.org/spec/CMMN/DefinitionType/CMISFolder</a></td>
</tr>
<tr>
<td>Physical</td>
<td>The ItemKind is used to represent objects in a BPMN Process that are physical objects, such as printed documents or manufactured items. These types of DataItems are not currently relevant to CMMN or DMN.</td>
</tr>
<tr>
<td>Relationship</td>
<td>The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/CMISRelationship">http://www.omg.org/spec/CMMN/DefinitionType/CMISRelationship</a></td>
</tr>
<tr>
<td>UML.Class</td>
<td>The ItemKind represents a UML Class in a Class Diagram.</td>
</tr>
<tr>
<td>Unknown</td>
<td>The ItemKind is defined through the following URI: <a href="http://www.omg.org/spec/CMMN/DefinitionType/Unknown">http://www.omg.org/spec/CMMN/DefinitionType/Unknown</a></td>
</tr>
</tbody>
</table>
### MultiplicityKind

This class is a type of SemanticReference that serves as the **term** for an SDMN Vocabulary that is used to specific the kind of multiplicity that exists for an ItemDefinition and a DataItem. Instead of being defined a fixed enumerated list, the kinds can be defined through a class (MultiplicityKind) and instances of that class (as shown below). The instances defined in the SDMN Library SHALL be included in any SDMN implementation. However, the implementation can allow additional instances of this class if required for a particular modeling situation (see the section entitled "MultiplictyKinds" for more information).

In practice, when a modeler creates a model with an ItemDefinition and a DataItem, the MultiplicityKind will be instantiated by one of the six instances in the Library. This set of instances is based on the CMMN CaseFileItem multiplicity setting. These kinds can be mapped to the BPMN Collection setting and the DMN Collection setting. See the section below for the mappings.

The following figure shows the metamodel elements related to the MultiplicityKind element (which includes the standard set of instances provided by the SDMN Library).

| Unspecified       | The ItemKind is defined through the following URI:  
|-------------------|---------------------------------------------------|
| WSDLMessage       | http://www.omg.org/spec/CMMN/DefinitionType/Unspecified  
| WSDIComplexType   | The ItemKind is a represents a WSDL Message.  
| XSDElement        | The ItemKind is represents a WSDL Message.  
| XSDSimpleType     | The ItemKind is a represents a WSDL Message.  

#### 10.2.3 MultiplicityKind

<table>
<thead>
<tr>
<th>MultiplicityKind</th>
<th>Description</th>
</tr>
</thead>
</table>
| Unspecified      | The ItemKind is defined through the following URI:  
| WSDLMessage      | http://www.omg.org/spec/CMMN/DefinitionType/Unspecified  
| WSDIComplexType  | The ItemKind is a represents a WSDL Message.  
| XSDElement       | The ItemKind is a represents a WSDL Message.  
| XSDSimpleType    | The ItemKind is a represents a WSDL Message.  

In practice, when a modeler creates a model with an ItemDefinition and a DataItem, the MultiplicityKind will be instantiated by one of the six instances in the Library. This set of instances is based on the CMMN CaseFileItem multiplicity setting. These kinds can be mapped to the BPMN Collection setting and the DMN Collection setting. See the section below for the mappings.

The following figure shows the metamodel elements related to the MultiplicityKind element (which includes the standard set of instances provided by the SDMN Library).
Figure 19 - The MultiplicityKind Metamodel

**Generalizations**

The MultiplicityKind element inherits the attributes and/or associations of:

- SemanticReference (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

**Properties**

The MultiplicityKind element does not have any additional attributes and/or associations.

**Standard Terms Vocabulary**

The following table presents a description for the included instances for MultiplicityKind.
Table 9. MultiplicityKind Literals

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExactlyOne</td>
<td>There is one copy of this ItemDefinition or DataItem.</td>
</tr>
<tr>
<td>OneOrMore</td>
<td>There is at least one copy of this ItemDefinition or DataItem, but</td>
</tr>
<tr>
<td></td>
<td>there may be more.</td>
</tr>
<tr>
<td>Unknown</td>
<td>The multiplicity is not known for this ItemDefinition or DataItem.</td>
</tr>
<tr>
<td>Unspecified</td>
<td>The multiplicity is not specified for this ItemDefinition or DataItem.</td>
</tr>
<tr>
<td>ZeroOrMore</td>
<td>There may be no copies of this ItemDefinition or DataItem or there</td>
</tr>
<tr>
<td></td>
<td>may be multiple copies.</td>
</tr>
<tr>
<td>ZeroOrOne</td>
<td>There may be no copies of this ItemDefinition or DataItem or there</td>
</tr>
<tr>
<td></td>
<td>may be one copy.</td>
</tr>
</tbody>
</table>

11 SDMN Models

The main purpose of SDMN is to allow modelers to create DataItem models, through a diagramming tool, to define the elements that are required for other BPM+ models, such as BPMN. This chapter defines the elements for constructing such models.

11.1 SharedDataModel

SharedDataModel is the abstract element that provides the foundation for the concrete SDMN models. Currently, there is only one concrete model, DataItemModel (see below). In future versions of SDMN, additional models can be added. It is a type of SDMNDefinitions and is contained in an SDMNDefinitions.

The figure below displays an example of a SharedDataModel for the Hello Patient use case.
Figure 20 – Example of the “Hello Patient” DataItem Model

The following figure shows the metamodel elements related to the SharedDataModel element.
Generalizations

The SharedDataModel element inherits the attributes and/or associations of:

- SCEDefinitions (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties

The following table presents the additional attributes and/or associations for SharedDataModel.
Table 10. SharedDataModel Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connector: Connector [0..*]</td>
<td>This is a list of the Connectors (Composition, Containment, Reference, and Data Association) that are included in the SharedDataModel. See the section entitled &quot;Connectors,&quot; below, for more information about Connectors.</td>
</tr>
<tr>
<td>dataItemRef: DataItem [0..*]</td>
<td>This is a list of the DataItems that are in the SharedDataModel.</td>
</tr>
<tr>
<td>ItemDefinitionRef: ItemDefinition [0..*]</td>
<td>This is a list of the Item Definitions that are in the SharedDataModel.</td>
</tr>
</tbody>
</table>

11.2 DataItemModel

The DataItemModel is mechanism for creating SDMx models. Through a modeling tool, which provides the specified notation, these models can be created to support other BPM+ languages, particularly in the context of a BPM+ Knowledge Package. It is contained in an SDMN Definitions.

The figure below displays an example of a DataItemModel for the Hello Patient use case.

Commented [SW121]: This figure was updated as a resolution for SDMN-83/SDMN-85
Generalizations
The `DataItemModel` element inherits the attributes and/or associations of:
- SDMNModel (see the section entitled “SDMNModel” for more information).
Further, the SDMNModel element inherits the attributes and/or associations of:
- SCEModel (see the SCE Specification for more information [OMG doc number bmi-2021-12-09]).

Properties
The following table presents the additional attribute and/or associations for `DataItemModel`.

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dataItemRef</code> DataItem [0..*]</td>
<td>This is a list of the DataItems that are in the DataItemModel.</td>
</tr>
</tbody>
</table>
### 11.2.1 Graphical Elements

The table below displays the graphical elements of a SharedDataModelDataItem Model.

**Table 12. Shared Data Model Graphical Elements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataItem</td>
<td>This is a DataItem that is set to a any type, except folder, through the itemKind property of the ItemDefinition assigned to the DataItem. A document shape with folded upper right corner is default notation for a DataItem (see figure to the right).</td>
<td></td>
</tr>
<tr>
<td>Parent DataItem (folder)</td>
<td>This is a DataItem that is set to a folder type through the itemKind property of the ItemDefinition assigned to the DataItem. For this variation of DataItem, its notation is set to a folder shape (see figure to the right).</td>
<td></td>
</tr>
<tr>
<td>DataItem (Collection)</td>
<td>Note that this marker can be used in combination with any of the following four markers shown in this table.</td>
<td></td>
</tr>
<tr>
<td>ItemDefinition (simple type)</td>
<td>DataItem (with Hidden Relationships—children) This is an ItemDefinition that is defined as a simple type.</td>
<td></td>
</tr>
<tr>
<td>ItemDefinition (complex type)</td>
<td>Locked DataItem</td>
<td></td>
</tr>
<tr>
<td>DataItem with Semantic Reference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Locked DataItem</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ItemDefinition (complex type)</td>
<td>DataItem with pre-assigned data This is an ItemDefinition that is defined as a complex type. The lines added within the DataItem shape indicate that some of the DataItem properties have been set with pre-assigned values through a LiteralExpression.</td>
<td></td>
</tr>
<tr>
<td>Composition Connector</td>
<td>This is a Connector that represents a composition relationship between two DataItems.</td>
<td></td>
</tr>
</tbody>
</table>
### 11.4.10.3 Connectors

#### 11.4.10.3.1 Connector

The Connector element is the abstract class that provides the common properties for the four concrete types of connectors (listed in the next four sections). It is contained in a SharedDataModel.

The following figure shows the metamodel elements related to the Connectors element.
Generalizations

The Connector element inherits the attributes and/or associations of:

- SCE:RootElement:ElementRelationshipType (see the SCE Specification for more information).

Further, the ElementRelationshipType element inherits the attributes and/or associations of:

- ElementType (see the SCE Specification for more information).
Properties

The following table presents the additional attributes and/or associations for Connector:

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceRef : DataItem [1]</td>
<td>The DataItem that the Connector is connecting from.</td>
</tr>
<tr>
<td>targetRef : DataItem [1]</td>
<td>The DataItem that the Connector is connecting to.</td>
</tr>
</tbody>
</table>

11.4.2 CompositionConnector

A CompositionConnector is used to define a relationship between DataItems. It represents a part-of relationship between two DataItems. That is, this relationship will specify that one DataItem (the target) is contained within part of another DataItem (the source). A Data Item of kind folder cannot be part of a composition relationship.

This relationship supports the composition of DataItems.

An example for DataItem composition can be found in healthcare scenarios: e.g., a patient record DataItem can be very complex and often only a portion of that DataItem is required for a DMN Decision. For example, creating a separate DataItem just for “demographics”, which is part of (contained by) a larger “health record” DataItem, will help focus the model for the context that is being modeled addressed at that point. This will help modelers and readers of the models to have a better understanding of the behaviors.

CompositionConnectors are contained in a SharedDataModel.

The runtime consequences of creating a RelationshipKind of Composition through a CompositionConnector between two DataItems include:

- If a container the source DataItem is deleted, then the target DataItem DataItems within the container will also be deleted.
- If a source DataItem container is moved or set within another a container, then the target DataItem DataItems within the container will also be moved.
- If element within a folder-type DataItem within a source DataItem container is updated (e.g., through a change in the value of a property), the container will not be updated. I.e., the container is not aware of changes to existing contained DataItems.
- If a target DataItem within a non-folder-type source DataItem container is updated (e.g., through a change in the value of a property), the container source DataItem will also be updated. I.e., the container source DataItem is aware of changes to contained target DataItems.

The following figure shows the metamodel elements related to the CompositionConnector element.
The CompositionConnector element does not have any additional attributes and/or associations.

Notation

The following statements define the notation for a CompositionConnector:

- A CompositionConnector is a line that SHALL be drawn with a single line (see below) with a filled diamond start and an angle 45° arrowhead end.

The use of text, color, size, and lines for a CompositionConnector SHALL follow the rules defined in the section entitled “Use of Text, Color, Size, and Lines in a Diagram” above.
The following figure displays an example of a **CompositionConnector**:

![Composition Connector](image)

### Connection Rules

The following statements define connection rules for a **CompositionConnector**:

- The source of a **CompositionConnector** SHALL be a **DataItem**.
- The target of a **CompositionConnector** SHALL be a **DataItem**.

### Generalizations

The **CompositionConnector** element inherits the attributes and/or associations of:

- **Connector** (see the section entitled "Connector" for more information).

Further, the **Connector** element inherits the attributes and/or associations of:

- **SCEElement**
- **RootElement**
- **ElementRelationshipType** (see the **SCE** Specification for more information).

### Constraints

A **CompositionConnector** is a type of **SCE ElementRelationshipType**, but is distinguished with this constraint:

- The instance for the element's **RelationshipKind** SHALL be named "Composition".

### Properties

The **CompositionConnector** element does not have any additional attributes and/or associations.

#### 4.4.310.3.3 ContainmentConnector

A **ContainmentConnector** is used to define a relationship between **DataItems**. This relationship will specify that one **DataItem** is contained within another **DataItem**. This relationship supports the containment of **DataItems**, including the parent-child association that exists between CMMN CaseFileItems. **Container DataItems** must have their **ItemKind** set to either folder or physical.

They are contained in a **SharedDataModel**.

The runtime consequences of creating a **RelationshipKind** of **Containment** through a **ContainmentConnector** between two **DataItems** include:

- If a container is deleted, then the **DataItems** within the container will also be deleted.
- If a container is moved or set within another container, then the **DataItems** within the container will also be moved.
- If an element within a **folder-type** DataItem container is updated (e.g., through a change in the value of a property), the container will not be updated. I.e., the container is not aware of changes to existing contained **DataItems**.
- If **DataItem** within a non-folder-type DataItem container is updated (e.g., through a change in the value of a property), the container will also be updated. I.e., the container is aware of changes to contained **DataItems**.

The following figure shows the metamodel elements related to the **ContainmentConnector** element.
The following statements define the notation for a Containment Connector:

- A Containment Connector is a line that SHALL be drawn with a single line (see below) with a cross-filled circle start and an angle 45° arrowhead end without an arrowhead end.

The use of text, color, size, and lines for a Containment Connector SHALL follow the rules defined in the section entitled “Use of Text, Color, Size, and Lines in a Diagram” above.

The following figure displays an example of a Containment Connector:
Connection Rules

The following statements define connection rules for a ContainmentConnector:

- The source of a ContainmentConnector SHALL be a DataItem.
  - The source DataItem MUST be assigned the ItemKind folder or physical.
- The target of a ContainmentConnector SHALL be a DataItem.

Generalizations

The ContainmentConnector element inherits the attributes and/or associations of:

- Connector (see the section entitled “Connector” for more information).

Further, the Connector element inherits the attributes and/or associations of:

- SCE_SCERootElement[ElementRelationshipType]; see the SCE Specification for more information [OMG doc number bmi-2021-12-09].

Constraints

A ContainmentConnector is a type of SCE ElementRelationshipType, but is distinguished with this constraint:

- The instance for the element’s RelationshipKind SHALL be named “Containment”.

Properties

The ContainmentConnector element does not have any additional attributes and/or associations.

11.4.4 DataAssociation

The DataAssociation class is a Connector and used to model how data is mapped between two DataItems. The source of the association is mapped to the target. The ItemDefinition from the sourceRef and targetRef MUST have the same ItemDefinition or the DataAssociation MUST have a transformation Expression that transforms the source ItemDefinition into the target ItemDefinition. It is contained within a SharedDataModel.

The following figure shows the metamodel elements related to the DataAssociation element.
The following statements define the notation for a **DataAssociation**:

- A **DataAssociation** is a line that SHALL be drawn with a dotted single line (see below) with an angle 45° arrowhead end.
- Note that the line style of the Data Association is the same as the SCE Model Artifact, **Association**. This graphical overlap was included in the BPMN 2.0 specification and SDMN was designed to be consistent with other BPM+ specifications. Thus, the same graphical overlap is being applied.

**Commented [SW151]**: This figure was updated by the resolution of Issue SDMN-93/SDMN-94. Updating SDMN based on structural changes to SCE.

**Commented [SW152]**: This figure was updated as a resolution for Issue SDMN-2/SDMN-51.

**Commented [SW153]**: This figure was also updated for the resolution of Issue SDMN-56/SDMN-87. Update DMN Dependencies.

**Commented [SW154]**: This figure was also updated for the resolution of Issue SDMN-113/SDMN-114. Merge SDMMModel and SharedDataModel.

**Commented [SW155]**: This figure was also updated for the resolution of Issue SDMN-130/SDMN-132. Move ItemKind to DataItem. Also removed DataItem attribute details since they are not needed here.
If a line that looks like an Association or a DataAssociation is connected between two DataItems, then the connector is assumed to be a DataAssociation (see the section entitled “Data Association Connection Rules,” below). If the source or target of the line is not a DataItem, then the connector is assumed to be an Association.

The use of text, color, size, and lines for a CompositionConnectorDataAssociation SHALL follow the rules defined in the section entitled “Use of Text, Color, Size, and Lines in a Diagram” above.

The arrowhead of the connector is attached to the DataItem that is the target of the data mapping.

The following figure displays an example of a DataAssociation:

![Figure 29: A Data Association](image)

**Connection Rules**

The following statements define connection rules for a DataAssociation connector:

- The source of a DataAssociation SHALL be a DataItem.
- The target of a DataAssociation SHALL be a DataItem.

![Figure 30: Example of DataAssociations between two DataItems](image)

**Generalizations**

The DataAssociation element inherits the attributes and/or associations of:

- Connector (see the section entitled “Connector” for more information).

Further, the Connector element inherits the attributes and/or associations of:

- SCE, SCERootElement, [ElementRelationshipType](https://www.omg.org/spec/SCE/2021-12-09/

**Constraints**

A DataAssociation is a type of SCE, ElementRelationshipType, but is distinguished with this constraint:

- The instance for the element’s RelationshipKind SHALL be named “Correlation”.

**Properties**

The following table presents the additional attributes and/or associations for DataAssociation:

| Commented [SW156]: This figure was updated for issue SCE-33/SCE-107. Editorial issues. Depicted markers are no longer in use. |
| Commented [SW157]: This text was updated by the resolution of issue SDMN-93/SDMN-94. Updating SDMN based on structural changes to SCE. |
| Commented [SW158]: This text was updated as a resolution for issue SDMN-2/SDMN-51. |
| Commented [SW159]: This text was updated for the resolution of issue SDMN-68/SDMN-106. Editorial issues. |
| Commented [SW160]: The Constraints section was removed as a resolution for issue SDMN-2/SDMN-51. |
Table 12: DataAssociation Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>assignment : Assignment [0..*]</td>
<td>Specifies one or more data elements Assignments. By using an Assignment, single data structure elements can be assigned from the source structure to the target structure.</td>
</tr>
<tr>
<td>transformation : LiteralExpression [0..1]</td>
<td>Specifies an optional transformation Expression. The actual scope of accessible data for that Expression is defined by the source and target of the specific DataAssociation types.</td>
</tr>
</tbody>
</table>

### 11.4.510.3.5 ReferenceConnector

An ReferenceRelationship is used to define a relationship between DataItems. This relationship will specify that one DataItem is referenced connected in some way to another DataItem; there is some type of relationship. This mechanism defines the technical structure of the DataItem. The referenced structure, shown as a separate DataItem in the diagram does extend the structure within the relationship as included to support the source DataItem but the referenced DataItem and target reference associations that exists on its own and can be found in CMMN CaseFileItems. For example, the CaseFileItems might be referenced by other created at the same time (although independently) or they are often sent together during the performance of a CMMN Case, etc. In a sense, it is a non-graphical way of grouping DataItems. They are contained in a SharedDataModel.

The runtime consequences of creating a RelationshipKind of ReferenceConnector through a CompositionConnector between two DataItems include:

- If a DataItem that is referenced by another DataItem (either as a source or target) is deleted, then the referenced DataItems will not be deleted.
- If a DataItem is moved or set within another container, then the DataItems referenced by that DataItem will not be moved.
- If DataItem is updated (e.g., through a change in the value of a property), the DataItems referenced by that DataItem will not be updated. I.e., a DataItem is not aware of changes to any referenced DataItems.

The following figure shows the metamodel elements related to the ReferenceConnector element.

Commented [SW161]: This text was also updated for the the resolution of Issue SDMN-129/SDMN-130. Editorial issues. Remove Text left over from the development of SDMN-95’s resolution.

Commented [SW162]: This text was updated as a resolution for Issue SDMN-2/SDMN-51.
The following statements define the notation for a ReferenceConnector:

- A ReferenceConnector is a line that SHALL be drawn with a long dashed single line (see below) with an angle 45° arrowhead end.

The use of text, color, size, and lines for a ReferenceConnector shall follow the rules defined in the section entitled “Use of Text, Color, Size, and Lines in a Diagram.”

- Use of Text, Color, Size, and Lines in a Diagram

Commented [SW163]: This figure was updated as a resolution for Issue SDMN-2/SDMN-51.

Commented [SW164]: This figure was removed for the resolution of Issue SCE-69/SCE-106. Editorial changes. Removing redundant figures.
The following figure displays an example of a ReferenceConnector:

Figure 32—Figure 21 - A Reference Connector

Connection Rules
The following statements define connection rules for a ReferenceConnector:
- The source of a ReferenceConnector SHALL be a DataItem.
- The target of a ReferenceConnector SHALL be a DataItem.

Generalizations
The ReferenceConnector element inherits the attributes and/or associations of:
- Connector (see the section entitled “Connector” for more information).

Further, the Connector element inherits the attributes and/or associations of:
- [SCE RootElement ElementRelationshipType see the SCE Specification for more information][OMG doc number bmi-2021-12-09].

Constraints
A ReferenceConnector is a type of SCE ElementRelationshipType, but is distinguished with this constraint:
- The instance for the element’s RelationshipKind SHALL be named “Reference”.

Properties
The ReferenceConnector element does not have any additional attributes and/or associations.

11.510.4 Model Artifacts
SDMN provides modelers with the capability of showing additional information about a Shared Data Model that is not directly related to the model elements through the capability provided by the ModelArtifact elements that are defined in the SCE specification. SDMN utilizes the three standard SCE ModelArtifacts: Associations, Groups, and TextAnnotations.

SDMN does not extend the capabilities of these ModelArtifacts but uses them as-is from the SCE specification. The following figure shows how the SCE ModelArtifact is included within SDMN. ModelArtifacts are contained within a SDMNModelSharedDataModel.

Commented [SW165]: This text was updated as a resolution for Issue SDMN-2/SDMN-51.
Commented [SW166]: This text was updated as a resolution for Issue SDMN-2/SDMN-51.
Commented [SW167]: This text was updated for the resolution of Issue SDMN-98/SDMN-106. Editorial Issues.
Commented [SW168]: The Constraints section was removed as a resolution for Issue SDMN-2/SDMN-51.
Commented [SW169]: This figure was updated by the resolution of Issue SDMN-93/SDMN-94. Updating SDMN based on structural changes to SCE.
Commented [SW170]: This figure was also updated for the resolution of Issue SDMN-113/SDMN-114. Merge SDMMModel and SharedDataModel.
A modeler or modeling tool MAY extend a SDMN Model and add new types of ModelArtifacts. Any new ModelArtifacts SHALL follow the connection rules (listed below). Associations can be used to link ModelArtifacts to other Model elements.

### Notation

Full details of ModelArtifacts are available in the section entitled “ElementType”, above, but the notation of the elements is provided here for convenience.

The table below displays the graphical elements of SCE’s Model Artifacts:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association</td>
<td>An Association is used to associate Model Artifacts (often Text Annotations) or model elements to other model elements. The connection only specifies that there is some relationship between the two elements, but no model semantics are implied.</td>
<td><img src="image" alt="Association Notation" /></td>
</tr>
<tr>
<td></td>
<td>An Association is line that is drawn with a dotted single line. An angle 30° arrowhead may optionally be added to either end of the line.</td>
<td><img src="image" alt="Association Notation" /></td>
</tr>
</tbody>
</table>
The **Group** object is a Model Artifact that provides a visual mechanism to group elements of a Model informally. **Groups** are often used to highlight certain sections of a Model without adding additional semantics. The highlighted (grouped) section of the Model can be separated for reporting and analysis purposes. A **Group** is a rounded corner rectangle that is drawn with a solid dashed and dotted line (see figure to the right).

**Text Annotation**

**Text Annotations** are a mechanism for a modeler to provide additional information for the reader of a SDMN Model. An **Association** may be used to connect user-defined text (a **Text Annotation**) with a Model element. A **Text Annotation** is an open rectangle that is drawn with a solid single line (see figure to the right).

**Model Artifact Connection Rules**

The following statements define connection rules for a **ModelArtifact**:

- A **ModelArtifact** **SHALL NOT** be a target for a **CompositionConnector**, a **ContainmentConnector**, a **DataAssociation**, or a **ReferenceConnector**.
- A **ModelArtifact** **SHALL NOT** be a source for a **CompositionConnector**, a **ContainmentConnector**, a **DataAssociation**, or a **ReferenceConnector**.

**12—SDMN Library**

A Library is included in SDMN to provide standard instances that should be implemented by tools supporting SDMN. Currently, SDMN defines the instances for two sub-packages named **ItemKinds** and **MultiplicityKinds** (See next two sections).

**12.1—ItemKinds**

The following figure presents the instances for the **ItemKind** element that are terms for the instance (**ItemKinds**) of the **SDMNVocabulary** element.
Some of the literals for ItemKind are based on the CMMN CaseFileItemDefinition literals for DefinitionType.

The following table presents a description for the included instances for ItemKind.

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual</td>
<td>The type of the ItemDefinition that doesn’t represent data or physical items, but represents concepts in the minds of users that are important for tasks or decisions. For example, a preference for a particular type of procedure will influence a doctor’s decision. While actual computations cannot be made with Conceptual ItemDefinitions, they are used to document aspects of the modeled behaviors.</td>
</tr>
<tr>
<td>DataType</td>
<td>The type of the ItemDefinition that fully utilizes the structural data capabilities inherent to ItemDefinition. Using FEEL, these will define simple types or data structures.</td>
</tr>
</tbody>
</table>
| Document  | This represents a Data Object or Case File Item that is a type of document. In BPMN, the document could be physical (e.g., printed) or electronic. In CMMN, it would represent a document in a Document Management System and is defined through the following URI: 
  http://www.omg.org/spec/CMMN/DefinitionType/CMISDocument |
| Folder    | This represents a CMMN Case File Item that is a Folder. A Folder can contain other Folders or Documents. These relationships are set through the Child and Parent attributes of the DataItem. Neither BPMN nor DMN have the concept of Folder as a data element. Thus, DataItems based on a Folder ItemDefinition would not map to BPMN or DMN data elements. The Folder is defined through the following URL:  
  http://www.omg.org/spec/CMMN/DefinitionType/CMISFolder |
### Physical
The `ItemKind` is represents objects in a BPMN Process that are physical objects, such as printed documents or manufactured items. These types of DataItems are not currently relevant to CMMN or DMN.

### Relationship
The `ItemKind` is defined through the following URI:
http://www.omg.org/spec/CMMN/DefinitionType/CMISRelationship

### UMLClass
The `ItemKind` is represents a UML Class in a Class Diagram.

### Unknown
The `ItemKind` is defined through the following URI:
http://www.omg.org/spec/CMMN/DefinitionType/Unknown

### Unspecified
The `ItemKind` is defined through the following URI:
http://www.omg.org/spec/CMMN/DefinitionType/Unspecified

### WSDLMessage
The `ItemKind` is represents a WSDL Message.

### XSDComplexType
For `ItemKinds` of this type, the (SCE) Import class SHOULD be used to import an XML Schema definition into the Shared Data Model. The `ItemKind` is defined through the following URI:
http://www.omg.org/spec/CMMN/DefinitionType/XSDComplexType

### XSDElement
For `ItemKinds` of this type, the (SCE) Import class SHOULD be used to import an XML Schema definition into the Shared Data Model. The `ItemKind` is defined through the following URI:
http://www.omg.org/spec/CMMN/DefinitionType/XSDElement

### XSDSimpleType
For `ItemKinds` of this type, the (SCE) Import class SHOULD be used to import an XML Schema definition into the Shared Data Model. The `ItemKind` is defined through the following URI:
http://www.omg.org/spec/CMMN/DefinitionType/XSDSimpleType

### Multiplicity Kinds
The following figure presents the instances for the `MultiplicityKind` element that are terms for the instance (`MultiplicityKinds`) of the SDMN Vocabulary element.
Figure 35 – The MultiplicityKinds Instance Model

The following table presents a description for the included instances for MultiplicityKind.

Table 14 – MultiplicityKind Literals

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExactlyOne</td>
<td>There is one copy of this ItemDefinition or DataItem.</td>
</tr>
<tr>
<td>OneOrMore</td>
<td>There is at least one copy of this ItemDefinition or DataItem, but there may be more.</td>
</tr>
<tr>
<td>Unknown</td>
<td>The multiplicity is not know for this ItemDefinition or DataItem.</td>
</tr>
<tr>
<td>Unspecified</td>
<td>The multiplicity is not specified for this ItemDefinition or DataItem.</td>
</tr>
<tr>
<td>ZeroOrMore</td>
<td>There may be no copies of this ItemDefinition or DataItem or there may be multiple copies.</td>
</tr>
<tr>
<td>ZeroOrOne</td>
<td>There may be no copies of this ItemDefinition or DataItem or there may be one copy.</td>
</tr>
</tbody>
</table>

**1311 Mapping to BPM+ Models**

The elements of SDMN, especially DataItems and ItemDefinitions, are intended for use by BPMN, CMMN, and DMN models. There are differences between the way data is used and defined across these three types of models. Thus, if SDMN is going to support all of them, then the SDMN must in fact define data elements as a super-set of the capabilities of the other three models.

The following sub-sections define any mappings required for SDMN data elements to be imported into the other BPM+ models.
### 43.111.1 Element Terminology Mapping to BPM+ Element Terminology

The following table defines the mapping for terms across the BPM+ languages and SDMN:

<table>
<thead>
<tr>
<th>Table 14. Mapping to BPMN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SDMN Element</strong></td>
</tr>
<tr>
<td>Containment Connector</td>
</tr>
<tr>
<td>Composition Connector</td>
</tr>
<tr>
<td>Data Association</td>
</tr>
<tr>
<td>DataItem</td>
</tr>
<tr>
<td>Data State</td>
</tr>
<tr>
<td>ItemDefinition</td>
</tr>
<tr>
<td>isCollection</td>
</tr>
<tr>
<td>ItemKind</td>
</tr>
<tr>
<td>Multiplicity</td>
</tr>
<tr>
<td>Pre-Assignment</td>
</tr>
<tr>
<td>Reference Connector</td>
</tr>
<tr>
<td>SDMNModel</td>
</tr>
</tbody>
</table>

Commented [SW174]: Row for DataState removed as a resolution for Issue SDMN-3/SDMN-52

Commented [SW175]: This text was updated for the resolution of Issue SDMN-129/SDMN-130. Multiplicity is not used by BPMN. isCollection is used instead by BPMN and DMN.

Commented [SW176]: This text was updated for the resolution of Issue SDMN-130/SDMN-132. Move ItemKind to DataItem. DataType was renamed to Information (as in BPMN)

### 43.211.2 BPMN

This section provides mapping from SDMN to BPMN.

**ItemKind Mapping**

The following table defines the mapping from SDMN ItemKind instances to BPMN itemKind literals:

<table>
<thead>
<tr>
<th>Table 15. ItemKind Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SDMN ItemKinds</strong></td>
</tr>
<tr>
<td>Conceptual</td>
</tr>
<tr>
<td>DataType</td>
</tr>
<tr>
<td>Document</td>
</tr>
<tr>
<td>Folder</td>
</tr>
</tbody>
</table>

Commented [SW176]: This text was updated for the resolution of Issue SDMN-130/SDMN-132. Move ItemKind to DataItem. DataType was renamed to Information (as in BPMN)
Physical Physical
Relationship Information
UMLClass Information
Unknown Information
Unspecified Information
WSDLMessage Information
XSDComplexType Information
XSDElement Information
XSDSimpleType Information

**MultiplicityKind Mapping**

The multiplicity attribute for the SDMN ItemDefinition element and the DataItem element is consistent with the CMMN multiplicity attribute for a Case File Item. However, the attribute is not consistent with the isCollection attribute for the ItemDefinition element of both BPMN. But the multiplicity values can be mapped to the Boolean isCollection.

The following table defines the mapping from SDMN MultiplicityKind instances to the BPMN Collection property:

<table>
<thead>
<tr>
<th>SDMN MultiplicityKinds</th>
<th>BPMN Collection Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZeroOrOne</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>An actual value of zero would not be valid for BPMN data elements. Thus, it is recommended to avoid this setting for SDMN DataItems that are used in BPMN models.</td>
</tr>
<tr>
<td>ZeroOrMore</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>An actual value of zero would not be valid for BPMN data elements. Thus, it is recommended to avoid this setting for SDMN DataItems that are used in BPMN models.</td>
</tr>
<tr>
<td>ExactlyOne</td>
<td>True</td>
</tr>
<tr>
<td>OneOrMore</td>
<td>True</td>
</tr>
<tr>
<td>Unspecified</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>The setting implies that the actual value could be zero, which would not be valid for BPMN data elements. Thus, it is recommended to avoid this setting for SDMN DataItems that are used in BPMN models.</td>
</tr>
<tr>
<td>Unknown</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>The setting implies that the actual value could be zero, which would not be valid for BPMN data elements. Thus, it is recommended to avoid this setting for SDMN DataItems that are used in BPMN models.</td>
</tr>
</tbody>
</table>

**Commented [SW177]: This section and table were removed for the resolution of issue SDMN-129/SDMN-130. Multiplicity is not used by BPMN (isCollection is used instead and the mapping shown above)**
The following table defines the mapping from SDMN elements and attributes to BPMN:

**Table 16. Mapping to BPMN**

<table>
<thead>
<tr>
<th>SDMN Element/Attribute</th>
<th>BPMN Element/Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataItem (not ItemKind Folder)</td>
<td>Data Object</td>
</tr>
<tr>
<td>DataItem (ItemKind Folder)</td>
<td>N/A</td>
</tr>
<tr>
<td>Item Definition</td>
<td>Item Definition</td>
</tr>
<tr>
<td>Item Definition</td>
<td>If the ItemDefinition for a DataItem is of type definitionRef, then the ItemDefinition will mapped to a BPMN ItemDefinition for an Data Object.</td>
</tr>
<tr>
<td>Item Definition</td>
<td>If the ItemDefinition for a DataItem is of type metaDefinitionRef, then the contents of the ItemDefinition will be ignored. There is no equivalent in BPMN. A separate ItemDefinition will not be created.</td>
</tr>
<tr>
<td>DataItem/Location</td>
<td>N/A</td>
</tr>
<tr>
<td>DataItem/ItemFormat</td>
<td>N/A</td>
</tr>
<tr>
<td>DataState</td>
<td>DataState</td>
</tr>
<tr>
<td>DataItem/preAssignment</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Although implementations of BPMN could establish an activity at the beginning of the process that fills the output Data Object with the values listed in the pre-assignment.</td>
</tr>
</tbody>
</table>

**43.311.3 CMMN**

This section provides mapping from SDMN to CMMN.

**ItemKind Mapping**

The following table defines the mapping from SDMN MultiplicityKind/ItemKind instances to the BPMN CMMN Collection-definitionType property:

**Table 17. ItemKind Literals**

<table>
<thead>
<tr>
<th>SDMN ItemKind Literals</th>
<th>CMMN ItemKind Literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual</td>
<td>Unknown</td>
</tr>
<tr>
<td>Data/Type:Information</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Document</td>
<td>Document in CMIS</td>
</tr>
<tr>
<td>Folder</td>
<td>Folder in CMIS</td>
</tr>
<tr>
<td>Physical</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Relationship</td>
<td>Relationship in CMIS</td>
</tr>
<tr>
<td>UMLClass</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Unknown</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

Commented [SW178]: Row for DataItem/Location removed as a resolution for Issue SDMN-4/SDMN-53

Commented [SW179]: Row for DataItem/ItemFormat removed as a resolution for Issue SDMN-62/SDMN-63

Commented [SW180]: Row for DataItem/ItemFormat removed as a resolution for Issue SDMN-62/SDMN-63

Commented [SW181]: Row for DataState removed as a resolution for Issue SDMN-3/SDMN-52

Commented [SW182]: This text was updated for the the resolution of Issue SDMN-129/SDMN-130. Editorial issues. fixed incorrect element and property names.

Commented [SW183]: This text was updated for the resolution of Issue SDMN-130/SDMN-132. Move ItemKind to Dataitem. Data/Type was renamed to Information (as in BPMN)
Unspecified Unspecified
WSDLMessage WSDLMessage
XSDComplexType XML Schema Complex Type
XSEDElement XML-Schema Element
XSDSimpleType XML Schema Simple Type

Element Mapping
The following table defines the mapping from SDMN elements and attributes to CMMN:

<table>
<thead>
<tr>
<th>SDMN Element/Attribute</th>
<th>CMMN Element/Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataItem (not ItemKind Folder)</td>
<td>CaseFileItem (not ItemKind Folder)</td>
</tr>
<tr>
<td>DataItem (ItemKind Folder)</td>
<td>CaseFileItem (ItemKind Folder)</td>
</tr>
<tr>
<td>Item Definition</td>
<td>CaseFileItemDefinition</td>
</tr>
<tr>
<td></td>
<td>If the ItemDefinition for a DataItem is of type definitionRef, then the ItemDefinition will mapped to a CMMN ItemDefinition for an CaseFileItem. If the ItemDefinition for a DataItem is of type metaDefinitionRef, then the simple types of the ItemDefinition will mapped to CMMN properties for an CaseFileItem.</td>
</tr>
<tr>
<td>DataItem/Location</td>
<td>N/A</td>
</tr>
<tr>
<td>DataItem/ItemFormat</td>
<td>N/A</td>
</tr>
<tr>
<td>DataState</td>
<td>N/A</td>
</tr>
<tr>
<td>DataItem/preAssignment</td>
<td>N/A</td>
</tr>
</tbody>
</table>

DMN
This section provides mapping from SDMN to DMN.

**MultiplictyKind Mapping**
The multiplicity attribute for the SDMN ItemDefinition element and the DataItem element is consistent with the CMMN multiplicity attribute for a Case File Item. However, the attribute is not consistent with the IsCollection attribute for the ItemDefinition element of both BPMN and DMN. But the multiplicity values can be mapped to the Boolean IsCollection.

The following table defines the mapping from SDMN MultiplicityKind instances to the DMN Collection property:

Commented [SW184]: Row for DataItem/Location removed as a resolution for Issue SDMN-4/SDMN-53
Commented [SW185]: Row for DataItem/ItemFormat removed as a resolution for Issue SDMN-62/SDMN-63
Commented [SW186]: Row for DataItem/ItemFormat removed as a resolution for Issue SDMN-62/SDMN-63
Commented [SW187]: Row for DataState removed as a resolution for Issue SDMN-3/SDMN-52
Commented [SW188]: This section and table were removed for the the resolution of Issue SDMN-129/SDMN-130. Multiplicity is not used by DMN (IsCollection is used instead and the mapping shown above)
Table 2: MultiplicityKind Mapping

<table>
<thead>
<tr>
<th>SDMN MultiplicityKinds</th>
<th>DMN Collection Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZeroOrOne</td>
<td>False</td>
</tr>
<tr>
<td>OneOrMore</td>
<td>True</td>
</tr>
<tr>
<td>ExactlyOne</td>
<td>False</td>
</tr>
<tr>
<td>Unspecified</td>
<td>False</td>
</tr>
</tbody>
</table>

An actual value of zero would not be valid for DMN data elements. Thus, it is recommended to avoid this setting for SDMN DataItems that are used in DMN models.

Element Mapping

The following table defines the mapping from SDMN elements and attributes to DMN:

Table 19, Mapping to DMN

<table>
<thead>
<tr>
<th>SDMN Element/Attribute</th>
<th>DMN Element/Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataItem (not ItemKind Folder)</td>
<td>InformationItem</td>
</tr>
<tr>
<td>DataItem (ItemKind Folder)</td>
<td>N/A</td>
</tr>
<tr>
<td>Item Definition</td>
<td>Item Definition</td>
</tr>
<tr>
<td></td>
<td>If the ItemDefinition for a InformationItem is of type definitionRef, then the ItemDefinition will be mapped to a DMN ItemDefinition for an InformationItem. If the ItemDefinition for a DataItem is of type metaDefinitionRef, then the contents of the ItemDefinition will be ignored. There is no equivalent in DMN. A separate ItemDefinition will not be created.</td>
</tr>
<tr>
<td>DataItem/ItemKind</td>
<td>N/A</td>
</tr>
<tr>
<td>DataItem/Location</td>
<td>N/A</td>
</tr>
<tr>
<td>DataItem/ItemFormat</td>
<td>N/A</td>
</tr>
<tr>
<td>DataState</td>
<td>N/A</td>
</tr>
<tr>
<td>DataItem/preAssignment</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Commented [SW189]: Row for DataItem/Location removed as a resolution for Issue SDMN-4/SDMN-53
Commented [SW190]: Row for DataItem/ItemFormat removed as a resolution for Issue SDMN-62/SDMN-63
Commented [SW191]: Row for DataItem/ItemFormat removed as a resolution for Issue SDMN-62/SDMN-63
Commented [SW192]: Row for DataState removed as a resolution for Issue SDMN-3/SDMN-52
## SDMN Examples

### 44.12.1 Hello Patient

The following figure provides an example of how a SDMN Data Item Diagram could look. It is based on a sample use case named “Hello Patient”, which is BPM Knowledge Package definition of a visit to a doctor’s office (which is BPM+ modeling technique for representing clinical guidelines). The DataItems in the model support the data elements of the Process, Case, and Decision models that define the behavioral components of the Knowledge Package.

Commented [SW193]: This text was updated for the resolution of Issue SDMN-69/SDMN-106. Editorial Issues.

Commented [SW194]: This figure was updated as a resolution for SDMN-83/SDMN-85

Commented [SW195]: This figure was replaced as a resolution for Issue SDMN-29/SDMN-64
Figure 36 — Figure 23 — An Example of How a SDMN DataItem Diagram

The following table lists the data elements used by the BPM+ models of the Knowledge Package.

Commented [SW196]: This figure was also updated for the resolution of Issue SDMN-135/SDMN-136. Fix SDMN diagram connector errors.

Commented [SW197]: This figure was updated as a resolution for Issue SDMN-29/SDMN-64

Commented [SW198]: This text was updated for the resolution of Issue SDMN-69/SDMN-106. Editorial issues.
### Table 20. List of Data Elements used by the BPM+ Models in the Hello Patient Use Case

<table>
<thead>
<tr>
<th>Cases</th>
<th>Decision Services</th>
<th>Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. Contributor</td>
<td>5. Demographics</td>
<td>5. BMI Category</td>
</tr>
<tr>
<td>17. Office Records</td>
<td>17. Weight Counseling Request</td>
<td>17. MRI Image</td>
</tr>
<tr>
<td>19. Patient Data</td>
<td></td>
<td>19. Office Records</td>
</tr>
<tr>
<td>20. Patient Health Record</td>
<td></td>
<td>20. Pathway Goals</td>
</tr>
<tr>
<td>22. Referrals</td>
<td></td>
<td>22. Patient Health Record</td>
</tr>
<tr>
<td>23. Scheduled Appointment</td>
<td></td>
<td>23. Patient Information</td>
</tr>
<tr>
<td>24. Treatment Plan</td>
<td></td>
<td>24. Referral</td>
</tr>
<tr>
<td>26. Weight Counseling Request</td>
<td></td>
<td>26. Treatment Plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27. Treatment Choice</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28. Vital Signs and Measurements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29. Weight Counseling Request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30. Weight Counseling Request Choice</td>
</tr>
</tbody>
</table>

Note that the data elements listed in bold in the table are those that appear in the SDMN model.

SDMN introduces a diagrammatic description of ItemDefinitions (types). ItemDefinitions are boxes with two or more sections. The top section will display the name of the ItemDefinition.

For simple type elements, a single section below the name will list the type of the element and any defined allowedValues of that type (see “Ethnicity” in the figure below).

If the element is a structure with multiple sub-elements, then each sub-element will be listed in a divided section where the of the sub-element is on the left (the name section) and the type and allowedValues are on the right (the type section – see “Patient Health Record”, which has four sub-elements, in the figure below).

If the ItemDefinition has a sub-element where the type is a sub-structure through an itemComponent, the sub-structure will be defined in a separate ItemDefinition and there will be a composition relationship line from the type section of the sub-element to the top-level name section of the sub-structure ItemDefinition (see the connection between “Patient Health Record” and “Patient Health Record Appointment” in the figure below).

The sub-structure, although shown as a separate ItemDefinition in the diagram is fully contained within (is part of) the main element and cannot be re-used by other ItemDefinitions (e.g., as for “Patient Health Record”). That is why the type section is empty since the contents of the types are displayed in a separate ItemDefinition.

If the ItemDefinition has a sub-element where the type is a referenced structure through a typeRef, the sub-
structure will be defined in a separate ItemDefinition and there will be a reference relationship line from the type section of the sub-element to the top-level name section of the referenced structure ItemDefinition (see the connection between “tPatient_Health_Record” and “tDemographics” in the figure below). The referenced structure, shown as a separate ItemDefinition in the diagram does extend the structure within the main element (e.g., as for “tPatient_Health_Record”) but the referenced ItemDefinition exists on its own and can be referenced by other ItemDefinitions. The typeRef section for the sub-element of the source ItemDefinition will also display the name of the referenced ItemDefinition.

Figure 24 - ItemDefinition Diagram Example

15.13 Exchange Formats

15.1 Interchanging Incomplete Models

In practice, it is common for models to be interchanged before they are complete. This occurs frequently when doing iterative modeling, where one user (such as a subject matter expert or business person) first defines a high-level model, and then passes it on to another user to be completed and refined.

Such “incomplete” models are ones in which all of the mandatory attributes have not yet been filled in, or the cardinality lowerbound of attributes and associations has not been satisfied.

XML allows for the interchange of such incomplete models. In SDMN, we extend this capability to interchange of XML files based on the SDMN XSD. In such XML files, implementers are expected to support this interchange by:

- Disregarding missing attributes that are marked as ‘required’ in the XSD.
- Reducing the lower bound of elements with ‘minOccurs’ greater than 0.

15.2 XSD

15.2.1 Document Structure

A domain-specific set of model elements is interchanged in one or more SDMN files. The root element of each file SHALL be <DMN xsi:schemaLocation="sharedDataModel.xsd">. The set of files SHALL be self-contained, i.e., all definitions that are used in a file SHALL be imported directly or indirectly using the
Each file SHALL declare a "targetNamespace" that may differ between multiple files of one model.

The XML namespace URI for SDMN 1.0 and backwards-compatible 1.x versions of SDMN is fixed at: https://www.omg.org/spec/SDMN/

In addition, SDMN XML files MUST use the mechanism for identifying and handling XML schema versions based on xsi:schemaLocation that is defined in the SCE specification. SDMN files MAY import non-SDMN files (such as XSDs and PMMLs) if the contained elements use external definitions.

46.2.213.2.2 References within the SDMN XSD

Many SDMN elements that may need to be referenced contain IDs and within the SDMN XSD, references to elements are expressed via these IDs. The XSD IDREF type is the traditional mechanism for referencing by IDs, however it can only reference an element within the same file. SDMN elements that inherit from SCE RootElement referencing by ID, across files, by utilizing QName. A QName consists of two parts: an optional namespace prefix and a local part. When used to reference a SDMN element, the local part is expected to be the id of the referenced SDMN element's first attribute whose value must be a valid URI reference (RFC 3986), where the path components may be absolute or relative, the reference has no query component, and the fragment consists of the value of the id of the referenced SDMN element.

For example, consider the following Decision:

```xml
<decision name="Pre Bureau Risk Category" id="prebureauriskDec01">…</decision>
```

When this Decision is referenced, e.g. by an InformationRequirement in a Decision that is defined in another file, the reference could take the following form:

```xml
<requiredDecision href="http://www.example.org/Definitions01.xml#prebureauriskDec01"/>
```

where "http://www.example.org/Definitions01.xml" is an URI reference to the XML document in which the "Pre Bureau Risk Category" Decision is defined (e.g. the value of the location URI attribute in the corresponding Import element), and "prebureauriskDec01" is the value of the id attribute for the Decision.

If the path component in the URI reference is relative, the base URI against which the relative reference is applied is determined as specified in [RFC 3986]. According to that specification, "if no base URI is embedded and the representation is not encapsulated within some other entity, then, if a URI was used to retrieve the representation, that URI shall be considered the base URI" ([RFC 3986, section 5.1.3]). That is, if the reference is not in the scope of an xml:base attribute [XBASE], a value of the base attribute that contains only a fragment and no path component references a SDMN element that is defined in the same instance of XML file as the referencing element. In the example above, assuming that the requiredDecision element is not in the scope of an xml:base attribute, the SDMN element whose id is "prebureauriskDec01" must be defined in the same XML document.

```xml
<requiredDecision href="#prebureauriskDec01"/>
```

Attribute typeRef references ItemDefinitions and built-in types by name not ID. In order to support imported types, typeRef uses the namespace-qualified name syntax [qualified],local-name], where qualifier is specified by the name attribute of the Import element for the imported type. If the referenced type is not imported, the prefix SHALL be omitted.

4614 SDMN Diagram Interchange (SDMN DI)

46.14.1 Scope

This chapter specifies the meta-model and schema for SDMN Diagram Interchange (SDMN DI). The SDMN DI is meant to facilitate the interchange of SDMN diagrams between tools rather than being used for internal diagram representation by the tools. The simplest interchange approach to ensure the unambiguous rendering of a SDMN diagram was chosen for SDMN DI. This includes the direct re-use of SCE DI elements (see the SCE specification).
without the need to create additional classes in SDMN. As such, SDMN DI does not aim to preserve or interchange any “tool smarts” between the source and target tools (e.g., layout smarts, efficient styling, etc.).

SDMN DI does not ascertain that the SDMN diagram is syntactically or semantically correct.

46.214.2 Diagram Definition and Interchange

The SDMN DI meta-model, similar to the SDMN abstract syntax meta-model, is defined as a MOF-based meta-model. As such, its instances can be serialized and interchanged using XML. SDMN-DI is also defined by an XML schema. Thus its instances can also be serialized and interchanged using XML.

Both SDMN-DI meta-model and schema is layered upon utilizes the SCE DI (see the SCE 1.0 specification), which is harmonized with the OMG Diagram Definition (DD) standard version 1.1. The referenced DD contains two main parts: the Diagram Commons (DC) and the Diagram Interchange (DI). The DC defines common types like bounds and points, while the DI provides a framework for defining domain-specific diagram models. As a domain specific DI, SDMN DI defines a few new meta-model classes that derive from the abstract classes from SCE DI and DI.

The focus of SDMN DI is the interchange of laid out shapes and edges that constitute a SDMN diagram. Each shape and edge references a particular SDMN model element. The referenced SDMN model elements are all part of the actual SDMN model. As such, SDMN DI is meant to only contain information that is neither present nor derivable, from the SDMN model whenever possible. Simply put, to render a SDMN diagram both the SDMN DI instance(s) and the referenced SDMN model are REQUIRED.

From the SDMN DI perspective, a SDMN diagram is a particular snapshot of a SDMN model at a certain point in time. Multiple SDMN diagrams can be exchanged referencing model elements from the same SDMN model. Each diagram may provide an incomplete or partial depiction of the content of the SDMN model. As described in Clause 13, a SDMN model package consists of one or more files. Each file may contain any number of SDMN diagrams. The exporting tool is free to decide how many diagrams are exported and the importing tool is free to decide if and how to present the contained diagrams to the user.

46.314.3 SDMN Diagram Interchange Meta-Model

46.314.3.1 How to read this Chapter

Clause 14.3.4.4 describes in detail the meta-model used to keep the layout and the look of SCE Diagrams (as a basis for SDMN Diagrams). Clause 14.3.4.4 presents in tables a library of the SCE element depictions and an unambiguous resolution between a referenced SDMN model element and its depiction. Clause 16.5 describes in detail the meta-model used to keep the layout and the look of SDMN diagrams. Clause 16.5.4 presents in tables a library of the SDMN element depictions and an unambiguous resolution between a referenced SDMN model element and its depiction.

46.314.3.2 Overview

The SDMN DI, which utilizes the SCE DI, is an instance of the OMG DI meta-model. The basic concept of SDMN DI, as with diagram interchange in general, is that serializing a diagram [SDMNDiagram,SCEDiagram] for interchange requires the specification of a collection of shapes [SDMNShape,SCEShape] and edges [SDMNEdge,SCEEdge].

The SDMN SCEDI classes only define the visual properties used for depiction. All other properties that are REQUIRED for the unambiguous depiction of the SDMN element are derived from the referenced SDMN model element [SDMNDiagramElement,SCEDiagramElement].

SDMN diagrams may be an incomplete or partial depiction of the content of the SDMN model. Some SDMN elements from a SDMN model may not be present in any of the diagram instances being interchanged.

SDMN DI does not directly provide for any containment concept. The SDMNDiagram,SCEDiagram is an ordered collection of mixed SDMNShape,SCEShape(s) and SDMNEdge,SCEEdge(s). The order of the SDMNShape,SCEShape(s) and SDMNEdge,SCEEdge(s) inside a SDMNDiagram,SCEDiagram determines their Z-order (i.e., what is in front of what). SDMNShape,SCEShape(s) and SDMNEdge,SCEEdge(s) that are SDMNEdge,SCEEdge(s) MUST appear after them in the SDMNDiagram,SCEDiagram. Thus, the exporting tool...
MUST order all \texttt{SCEShape}(s) and \texttt{SCEEdge}(s) such that the desired depiction can be rendered.

16.3.3.3 Measurement Unit

As per OMG DD, all coordinates and lengths defined by SDMN \texttt{DI} are assumed to be in user units, except when specified otherwise. A user unit is a value in the user coordinate system, which initially (before any transformation is applied) aligns with the device’s coordinate system (for example, a pixel grid of a display). A user unit, therefore, represents a logical rather than physical measurement unit. Since some applications might specify a physical dimension for a diagram as well (mainly for printing purposes), a mapping from a user unit to a physical unit can be specified as a diagram’s resolution. Inch is chosen in this specification to avoid variability, but tools can easily convert from/to other preferred physical units. Resolution specifies how many user units fit within one physical unit (for example, a resolution of 300 specifies that 300 user units fit within 1 inch on the device).

16.3.4 Elements

The following sections define the elements necessary for exchanging the diagrams from an SDMN modeling tool.

16.3.4.1 SDMNDI

SDMN \texttt{DI} doesn’t have any defined elements, but uses the elements provided by SCEDI (see Figure 1 - to see how SCEDI is structured in the overall SDMN metamodel). However, it should be noted that the use of the \texttt{SCEDiagramElement} class (see the figure below) should be restricted for SDMN. That is, the \texttt{elementRef} association for \texttt{SCEDiagramElement} should be restricted to include only the concrete sub-classes of \texttt{BaseElement} that are in the SharedDataModel (i.e., SDMN model elements, such as \texttt{DataItems}, etc.).

The class SDMNDI is a container for the shared SCE:SCEShape and all the SDMNDiagrams defined in a SharedDataModel. The following figure shows the SDMNDI metamodel diagram.

![SDMNDI Metamodel Diagram](image)

Figure 37 – SDMNDI

Generalizations

The SDMNDI element does not inherit any attributes or associations of from another element.

Properties

The following table presents the additional attributes and/or associations for SDMNDI.

Table 15 – SDMNDI Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diagram : SDMNDiagram [0..*]</td>
<td>A list of SDMNDiagrams</td>
</tr>
</tbody>
</table>

Commented [SW207]: This text was updated for the resolution of Issue SDMN-33/SDMN-107. Directly utilize SCEDI.

Commented [SW208]: This text was updated for the resolution of Issue SDMN-33/SDMN-107. Directly utilize SCEDI.

Commented [SW209]: This text was added for the resolution of Issue SDMN-33/SDMN-107. Directly utilize SCEDI.
16.3.4.2 SDMNDiagram

The class SDMNDiagram specializes SCE:SCEDiagram, which specializes DI:Diagram. It is a kind of Diagram that represents a depiction of all or part of a SDMNDiagram.

SDMNDiagram is the container of SDMNDiagramElement (SDMNShape(s) and SDMNEdge(s)). SDMNDiagram cannot include other SDMNDiagrams.

A SDMNDiagram can define a SCE:SCEStyle locally and/or it can refer to a shared one defined in the SDMNDI. Properties defined in the local style override the one in the referenced shared style. That combined style (shared and local) is the default style for all the SDMNDiagramElement contained in this SDMNDiagram.

The SDMNDiagram class represents a two-dimensional surface with an origin of (0, 0) at the top left corner. This means that the x and y axes have increasing coordinates to the right and bottom. Only positive coordinates are allowed for diagram elements that are nested in a SDMNDiagram.

The following figure shows the SDMNDiagram metamodel diagram.

![SDMNDiagram Metamodel Diagram](image)

Figure 38 - SDMNDI Diagram

Generalizations

The SDMNDiagram element inherits the attributes and/or associations of:

• SCEDiagram (see the SCE specification for more information [OMG doc number bmi-2021-12-09]).

Further, the SCEDiagram element inherits the attributes and/or associations of:

• Diagram (see the SCE specification for more information).

Further, the Diagram element inherits the attributes and/or associations of:
In addition, the DiagramElement element inherits the attributes and/or associations of:

- Diagram (see the SCE specification for more information).

Further, the Diagram element inherits the attributes and/or associations of:

- DiagramElement (see the SCE specification for more information).

Properties

The following table presents the additional attributes and/or associations for SDMNDiagram:

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diagramElement: SDMNDiagramElement [0..*]</td>
<td>A list of SDMNDiagramElements (SDMNShape and SDMNEdge) that are depicted in the SDMNDiagram. This redefines the diagramElement association within the SCE:SCEDiagram element.</td>
</tr>
</tbody>
</table>

16.3.4.3 SDMNDiagramElement

The SDMNDiagramElement class is contained by the SDMNDiagram and is the base class for SDMNShape and SDMNEdge. SDMNDiagramElement inherits its styling from its parent SCE:Diagram. In addition, it can refer to one of the shared SCE:SCEStyle defined in the SMDNDI and/or it can define a local style. See section below for more details on styling.

SDMNDiagramElement MAY also contain a SCE:SCELabel when it has a visible text label. If no SCE:SCELabel is defined, the SDMNDiagramElement should be depicted without a label.

The following figure shows the SDMNDiagramElement metamodel diagram.

Commented [SW212]: This text was removed for the resolution of Issue SDMN-33/SDMN-107. Directly utilize SCEDI.
Generalizations

The SDMNDiagramElement element inherits the attributes and/or associations of:

- SCEDiagramElement (see the SCE specification for more information [OMG doc number bmi-2021-12-09]).

Further, the SCEDiagramElement element inherits the attributes and/or associations of:

- DiagramElement (see the SCE specification for more information).

In addition, the DiagramElement element inherits the attributes and/or associations of:

- DiagramElement (see the SCE specification for more information).

Properties

The SDMNDiagramElement element does not have any additional attributes and/or associations.

16.3.4.4 SDMNShape

The SDMNShape class specializes SCE::SCEShape, which specializes DI::Shape and SCE::SCEDiagramElement. It is a kind of shape that depicts a SCE::Element from the SDMNDiagram model.

SDMNShapes represent a DataItem or a SCE::DiagramArtifacts Group or a Text Annotation that is depicted on the diagram. SDMNDiagram models may add additional shapes to their diagrams.

SDMNShapes have no additional properties, but a SDMNDiagram model may extend this class to add properties that are used to further specify the appearance of some shapes that cannot be deduced from the SDMNDiagram model.

The following figure shows the SDMNShape metamodel diagram.
Figure 40 - SDMNDI Shape

**Generalizations**

The SDMNShape element inherits the attributes and/or associations of:
- SCEShape (see the SCE specification for more information [OMG doc number bmi-2021-12-09]).
  Further, the SCEShape element inherits the attributes and/or associations of:
  - SCEDiagramElement (see the SCE specification for more information).
  Further, the SCEDiagramElement element inherits the attributes and/or associations of:
  - DiagramElement (see the SCE specification for more information).
In addition, the DiagramElement element inherits the attributes and/or associations of:
  - DiagramElement (see the SCE specification for more information).
In addition, the DiagramElement element inherits the attributes and/or associations of:
  - Shape (see the SCE specification for more information).
In addition, the Shape element inherits the attributes and/or associations of:
  - Shape (see the section entitled “Shape” for more information).
In addition, the SDMNShape element inherits the attributes and/or associations of:
  - SDMNDiagramElement (see the section entitled “SDMNDiagramElement” for more information).
  Further, the SDMNDiagramElement element inherits the attributes and/or associations of:
  - SCEDiagramElement (see the section entitled “SCEDiagramElement” for more information).
  Further, the SCEDiagramElement element inherits the attributes and/or associations of:
  - DiagramElement (see the section entitled “DiagramElement” for more information).
In addition, the DiagramElement element inherits the attributes and/or associations of:
  - DiagramElement (see the section entitled “DiagramElement” for more information).
Properties

The SDMNShape element does not have any additional attributes and/or associations.

16.3.4.5 SDMNEdge

The SDMNEdge class specializes SCE:SCEEdge, which specializes DI::Edge and SCE:SCEDiagramElement. It is a kind of edge that can depict a relationship between two SDMNDiagram model elements.

SDMNEdge are used to depict Connectors or Associations in the SDMNDiagram model. Since SDMNDiagramElement might be depicted more than once, sourceElement and targetElement attributes allow to determine to which depiction a SDMNEdge is connected. When SDMNEdge has a source, its sourceModelElement SHALL refer to the SDMNDiagramElement it starts from. That SDMNDiagramElement SHALL resolved to the SCE:SCEElement that is the actual source of the Connector or Association. For Requirement, this is the required SCE:SCEElement. When it has a target, its targetModelElement SHALL refer to the SCE:SCEElement where it ends. That SDMNDiagramElement SHALL resolved to the SCE:SCEElement that is the actual target of the Connector or Association.

The following figure shows the SDMNEdge metamodel diagram.

![SDMNEdge Metamodel Diagram](image)

**Commented [SW215]:** This section was removed for the resolution of Issue SDMN-33/SDMN-107. Directly utilize SCEDI.

Generalizations

The SDMNEdge element inherits the attributes and/or associations of:

- SCEEdge (see the SCE specification for more information) [OMG doc number bmi-2021-12-09].
- Edge (see the SCE specification for more information).

Further, the SCEEdge element inherits the attributes and/or associations of:

- Edge (see the SCE specification for more information).

In addition, the Edge element inherits the attributes and/or associations of:

- Edge (see the SCE specification for more information).

In addition, the Edge element inherits the attributes and/or associations of:

- SCEDiagramElement (see the SCE specification for more information).
Further, the SCEDiagramElement element inherits the attributes and/or associations of:
- DiagramElement (see the SCE specification for more information).
In addition, the DiagramElement element inherits the attributes and/or associations of:
- SCEDiagramElement (see the SCE specification for more information).
Further, the SCEDiagramElement element inherits the attributes and/or associations of:
- SCEDiagramElement (see the SCE specification for more information).

Properties

The SDMNEdge element does not have any additional attributes and/or associations.

16.3.5.14.3.5 Notation

As a specification that contains notation, SDMN specifies the depiction for SDMN diagram elements, including SCE.DiagramArtifact elements (OMG doc number bmi-2021-12-09).

Serializing a SDMN diagram for interchange requires the specification of a collection of SDMShape, SDMNShape, and SDMNSCEEdge, and SDMNSCEEdge(s) in the SDMNSCEDiagramSDMNDiagram (see sections above). The SDMNSCEShape(s) and SDMNSCEEdge(s) attributes must be populated in such a way as to allow the unambiguous rendering of the SDMN diagram by the receiving party. More specifically, the SDMNSCEShape(s) and SDMNSCEEdge(s) MUST reference SDMN model elements. If no SCEcreateElement is referenced or if the reference is invalid, it is expected that this shape or edge should not be depicted.

When rendering a SDMN diagram, the correct depiction of a SDMNSCEShape, SDMNSShapes or SDMNSCEEdge, SDMNSEdges depends mainly on the referenced SDMN model element and its particular attributes and/ or references. The purpose of this clause is to: provide a library of the SDMN element depictions, and to provide an unambiguous resolution between the referenced SDMN model element [SCE createElement] and their depiction. Depiction resolution tables are provided below for both SDMNSCEShape, SDMNSShapes and SDMNSCEEdge, SDMNSEdges.

16.3.6.14.3.5.1 Labels

Both SDMNSCEShape, SDMNSShapes and SDMNSCEEdge, SDMNSEdges may have labels (its name attribute) placed on the shape/edge, or above or below the shape/edge, in any direction or location, depending on the preference of the modeler or modeling tool vendor.

Labels are optional for SDMNSCEShape, SDMNSShapes and SDMNSCEEdge, SDMNSEdges. When there is a label, the position of the label is specified by the bounds of the SDMNSCELabel, SDMNSLabel of the SDMNSCEShape, SDMNSShapes or SDMNSCEEdge, SDMNSEdges. Simply put, label visibility is defined by the presence of the SDMNSCELabel, SDMNSLabel element.

The bounds of the SDMNSCELabel, SDMNSLabel are optional and always relative to the containing SDMNSCEDiagram, SDMNSDiagram’s origin point. The depiction resolution tables provided below exemplify default label positions if no bounds are provided for the SDMNSCELabel, SDMNSLabel (for SDMNSCEShape, SDMNSShapes kinds and SDMNSCEEdge, SDMNSEdges kinds (see sections above)).

When the SDMNSCELabel, SDMNSLabel is contained in a SDMNSCEShape, SDMNSShapes, the text to display is the
name of the `BaseSCEElement`.

### SDMNShape Resolution

SDMNShape can be used to represent a `DataItemDataItemTextAnnotation` or an `ItemDefinition | Group`.

#### DataItems

A `DataItem` is represented in a SDMN Diagram as one of two possible shapes.

If the `DataItem`'s type is a Folder, then it is displayed with a folder shape (see table below, first row). These `DataItems` are only used within a CMMN diagram (outside of a SDMN diagram) although they will be displayed like any other CaseFileItem.

If the `DataItem`'s type is not a Folder (i.e., any other type of `DataItem`), then it is displayed with a document shape (see table below, second row). These type of `DataItems` can be used in BPMN, CMMN, and DMN diagrams. The SDMN shape for these `DataItems` match the shape used in BPMN and CMMN, but DMN uses a lozenge shape for its Data Inputs.

Note that the `DataItem` type is determined by the `DataItem`'s associated `ItemDefinition`.

The following table presents the depiction resolutions for `DataItems`:

<table>
<thead>
<tr>
<th>SDMNElement</th>
<th>SDMNShape Attributes</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DataItem</code> (Folder)</td>
<td>Shapes of <code>DataItem</code> that have <code>itemKind=Folder</code></td>
<td>![Folder Shape]</td>
</tr>
<tr>
<td><code>DataItem</code> (not Folder)</td>
<td>Shapes of <code>DataItem</code> that have <code>itemKind!=Folder</code></td>
<td>![Document Shape]</td>
</tr>
</tbody>
</table>

#### Multiplicity/Collection Decorator

The following table presents the depiction resolutions for the Multiplicity/`Collection` Decorator:

<table>
<thead>
<tr>
<th>DataItem Attribute</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Multiplicity = ZeroOrMore or OneOrMore or isCollection = true</code></td>
<td>![Multiplicities Collection]</td>
</tr>
</tbody>
</table>

#### ItemDefinitions

An `ItemDefinition` is represented in a SDMN Diagram as one of two possible shapes, depending on if the

Commented [SW217]: Typo fix for Issue SDMN-26/SDMN-57
Commented [SW218]: This text was updated for the resolution of Issue SDMN-33/SDMN-107. Directly utilize SCEDI.
Commented [SW219]: This image was updated as a resolution for Issue SDMN-29/SDMN-64
Commented [SW220]: This text was updated for the the resolution of Issue SDMN-129/SDMN-130. isCollection also applies here.
**Locked Decorator**

The following table presents the depiction resolutions for itemDefinitions with the Locked Decorator:

<table>
<thead>
<tr>
<th>SDMNElement</th>
<th>SDMNShape Attributes</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemDefinition</td>
<td>No SDMNComponent present</td>
<td><img src="#" alt="Label" /></td>
</tr>
<tr>
<td>ItemDefinition</td>
<td>With 1 or more itemComponent</td>
<td><img src="#" alt="Label" /></td>
</tr>
</tbody>
</table>

**Semantic Reference Decorator**

The following table presents the depiction resolutions for the Semantic Reference Decorator:

<table>
<thead>
<tr>
<th>DataItem Attribute</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the DataItem or any component of its associated ItemDefinition has an defined SemanticReference</td>
<td><img src="#" alt="Attribute" /></td>
</tr>
</tbody>
</table>

**Hidden Relationship Decorator**

The following table presents the depiction resolutions for the Hidden Relationship Decorator:

<table>
<thead>
<tr>
<th>DataItem Circumstance</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Circumstance" /></td>
<td><img src="#" alt="Depiction" /></td>
</tr>
</tbody>
</table>

46.3.6.314.3.5.3 **SDMNEdge Resolution**

SDMNEdge can be used to represent an Ownership Connector, Parent-Child Connector, Relationship Connector, or a Data Association.

**Commented [SW221]**: This text was updated for the resolution of Issue SDMN-33/DMN-107. Directly utilize SCEDI.
The following table presents the depiction resolutions for a **CompositionConnector**:

<table>
<thead>
<tr>
<th>SDMN Element</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompositionConnector</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Commented [SW222]:** This figure was updated for the resolution of Issue SDMN-125/SDMN-126. Removing label from the Edge.

The following table presents the depiction resolutions for a **CompositionConnector**:

<table>
<thead>
<tr>
<th>SDMN Element</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemComponent with sub-components</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Commented [SW222]:** This text and figure was updated for the resolution of Issue SDMN-121/SDMN-122. Clarify edge definition.

**Commented [SW224]:** This figure was updated for the resolution of Issue SDMN-125/SDMN-126. Removing label from the Edge.

**Commented [SW225]:** This figure was updated for the resolution of Issue SDMN-125/SDMN-126. Removing label from the Edge.

**Commented [SW226]:** This figure was updated for the resolution of Issue SDMN-125/SDMN-126. Removing label from the Edge.

**ContainmentConnector**

The following table presents the depiction resolutions for a **ContainmentConnector**:

<table>
<thead>
<tr>
<th>SDMN Element</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContainmentConnector</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Commented [SW222]:** This figure was updated for the resolution of Issue SDMN-125/SDMN-126. Removing label from the Edge.

**DataAssociation**

The following table presents the depiction resolutions for a **DataAssociation**:

<table>
<thead>
<tr>
<th>SDMN Element</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataAssociation</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Commented [SW226]:** This figure was updated for the resolution of Issue SDMN-125/SDMN-126. Removing label from the Edge.
The following table presents the depiction resolutions for a ReferenceConnector between two DataItems:

<table>
<thead>
<tr>
<th>SDMN Element</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReferenceConnector</td>
<td><img src="image1" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
</tbody>
</table>

The following table presents the depiction resolutions for a ReferenceConnector between two ItemDefinitions:

<table>
<thead>
<tr>
<th>SDMN Element</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemComponent that contains a typeRef</td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
</tbody>
</table>

Commented [SW227]: This figure was updated for the resolution of Issue SDMN-125/SDMN-126. Removing label from the Edge.

Commented [SW228]: This figure was updated for the resolution of Issue SDMN-125/SDMN-126. Removing label from the Edge.

Commented [SW229]: This text and figure was updated for the resolution of Issue SDMN-121/SDMN-122. Clarify edge definition.