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®, IIOP®, 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.
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 ............................................................. 10
   6.3 Display of Metamodel Diagrams .............................................................................................. 11
   6.4 Use of Text, Color, Size, and Lines in a Diagram ...................................................................... 12
   6.5 Abbreviations .......................................................................................................................... 13
   6.6 Structure of this Document ...................................................................................................... 13
   6.7 Acknowledgements .................................................................................................................. 13
7. Overview ......................................................................................................................................... 14
   7.1 What Constitutes a BPM+ Model? ............................................................................................ 14
   7.2 Why a Shared Data Model? ...................................................................................................... 14
     7.2.1 Use Case: Hello Patient ................................................................................................... 15
   7.3 The Purpose and Use of a Shared Data Model .......................................................................... 18
8. Specification Core Elements .......................................................................................................... 19
9. SDMN Metamodel ......................................................................................................................... 19
   9.1 SharedDataModel .................................................................................................................... 20
10. SDMN Model Elements .............................................................................................................. 22
    10.1 DataItems .............................................................................................................................. 22
    10.1.1 DataItem ........................................................................................................................... 23
    10.1.2 Pre-Assigning Values for DataItems ............................................................................... 26
    10.2 Item Definitions ...................................................................................................................... 27
    10.2.1 ItemDefinition .................................................................................................................. 28
7.3 Connectors .................................................................................................................................... 30
    10.3.1 Connector .......................................................................................................................... 30
    10.3.2 CompositionConnector ..................................................................................................... 31
    10.3.3 ContainmentConnector ...................................................................................................... 32
    10.3.4 DataAssociation ............................................................................................................... 33
    10.3.5 ReferenceConnector ......................................................................................................... 35
    10.4 Model Artifacts ....................................................................................................................... 36
11. Mapping to BPM+ Models ........................................................................................................... 37
    11.1 Element Terminology Mapping to BPM+ Element Terminology ........................................ 37
    11.2 BPMN ..................................................................................................................................... 38
    11.3 CMMN ..................................................................................................................................... 39
    11.4 DMN ........................................................................................................................................ 40
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>SDMN Examples</td>
<td>12–41</td>
</tr>
<tr>
<td>12.1</td>
<td>Hello Patient</td>
<td>12–41</td>
</tr>
<tr>
<td>13</td>
<td>Exchange Formats</td>
<td>13–44</td>
</tr>
<tr>
<td>13.1</td>
<td>Interchanging Incomplete Models</td>
<td>13–44</td>
</tr>
<tr>
<td>13.2</td>
<td>XSD</td>
<td>13.2–44</td>
</tr>
<tr>
<td>13.2.1</td>
<td>Document Structure</td>
<td>13.2.1–44</td>
</tr>
<tr>
<td>13.2.2</td>
<td>References within the SDMN XSD</td>
<td>13.2.2–45</td>
</tr>
<tr>
<td>14</td>
<td>SDMN Diagram Interchange (SDMN DI)</td>
<td>14–45</td>
</tr>
<tr>
<td>14.1</td>
<td>Scope</td>
<td>14.1–45</td>
</tr>
<tr>
<td>14.2</td>
<td>Diagram Definition and Interchange</td>
<td>14.2–45</td>
</tr>
<tr>
<td>14.3</td>
<td>SDMN Diagram Interchange Meta-Model</td>
<td>14.3–46</td>
</tr>
<tr>
<td>14.3.1</td>
<td>How to read this Chapter</td>
<td>14.3.1–46</td>
</tr>
<tr>
<td>14.3.2</td>
<td>Overview</td>
<td>14.3.2–46</td>
</tr>
<tr>
<td>14.3.3</td>
<td>Measurement Unit</td>
<td>14.3.3–46</td>
</tr>
<tr>
<td>14.3.4</td>
<td>Elements</td>
<td>14.3.4–46</td>
</tr>
<tr>
<td>14.3.4.1</td>
<td>SDMN DI</td>
<td>14.3.4.1–46</td>
</tr>
<tr>
<td>14.3.5</td>
<td>Notation</td>
<td>14.3.5–46</td>
</tr>
<tr>
<td>14.3.5.1</td>
<td>Labels</td>
<td>14.3.5.1–47</td>
</tr>
<tr>
<td>14.3.5.2</td>
<td>SDMNShape Resolution</td>
<td>14.3.5.2–47</td>
</tr>
<tr>
<td>14.3.5.3</td>
<td>SDMNEdge Resolution</td>
<td>14.3.5.3–48</td>
</tr>
</tbody>
</table>
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 http://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
1 Scope

A Shared Data Model is a 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 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)
  - Diagrams can be included to visualize the **DataItems** and their relationships or **ItemDefinitions** and their relationships

The primary goal of **SDMN** is to provide a set of structural elements that are common to other Object Management Group (OMG) specifications. **SDMN** has been structured to be dependent on the elements defined in Specification Common Elements (see the SCE specification for more information). 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**) 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.

### 3 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


3.2 Non-normative References

The following normative documents do not contain any non-normative references.

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>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>
<thead>
<tr>
<th>Table 2. SDMN Metamodel Color-Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>SCE Structural Class</td>
</tr>
<tr>
<td>SDMN General Class</td>
</tr>
</tbody>
</table>
### SDMN General Class (focus of diagram)
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.

### DMN General Class
Metamodel elements from the DMN 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 light-yellowish (see figure to the right).

### External Class
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).

### SDMN Class Instance
These 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).

### SCE Class Instance
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).

### Enumerations
(see figure to the right).

### 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).
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>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>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>SCE</td>
<td>Specification Common Elements</td>
</tr>
<tr>
<td>DMN</td>
<td>Shared Data Model and Notation</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”). The introduction is followed by normative clauses that define the elements of the specification and their properties and associations (see the sections entitled “SDMN Metamodel” (Clause 9); “SDMN Model Elements” (Clause 10); “Mapping to BPM+ Models” (Clause 11); and “SDMN Diagram Interchange” (Clause 14)).

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
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 collection of data elements and item definitions that supports a set of BPM-Plus (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 the data elements that are used across those correlated BPM+ models. Thus, the SDM provides a shared, scoped, and focused view that supports mutual interfaces between the models, as well as external data sources.

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 BPM+. 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.

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 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 data collection for the data elements of BPM+ Models was identified. This collection would serve as a central source for the development of data elements that would be referenced by the other BPM+ models. This collection 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 SDMN, there is the 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 SDMN is 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.

![Figure 1 - Overview of SDMN in the Context of BPM+ Standards](image)

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 Shared Data Model. The work on this and other use cases was instrumental in identifying the need and requirements for a Shared Data Model.

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</td>
<td>17. Weight Counseling Referral</td>
<td>18. Vital Signs and Measurements</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.

![Figure 2 - Illustration of How Data Elements are Shared Across BPM+ Models](image-url)
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+ 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 frequently, 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 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 BPMN 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.
7.3 The Purpose and Use of a Shared Data Model

A Shared Data Model serves multiple purposes with a set of BPM+ models. First, it provides a collection of DataItems and ItemDefinitions that serve as the source for the data elements of the BPMN, CMMN, and DMN models, 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.
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 [See the SCE specification] and contains a set of basic metamodel classes that are common to 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).

---

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 [see the SCE specification]. This dependency is manifested in multiple SDMN metamodel relationships. For example, the SharedDataModel element directly specializes the SCE SCEModel element, thus inheriting all the properties and associations of that element.

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

![SDMN Main Packages](image_url)

**Figure 5 - SDMN Main Packages**

Further, most of the other SDMN elements directly specialize the SCE BaseElement or SCE RootElement. 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.

### 9.1 SharedDataModel

The SharedDataModel 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 SharedDataModels. Specifically, an XML file for a SharedDataModel 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 following figure shows the SharedDataModel metamodel.
**Generalizations**

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

- `SCEModel` (see the SCE Specification for more information).

**Properties**

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

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>itemDefinition</strong> : ItemDefinition [0..*]</td>
<td>This is a list of the ItemDefinitions that are included in the SharedDataModel. See the section entitled &quot;Item Definitions,&quot; below, for more information about ItemDefinitions.</td>
</tr>
<tr>
<td><strong>diagrams</strong> : SCE::Diagrams [0..1]</td>
<td>This attribute contains the Diagram Interchange information contained within this SharedDataModel. See the section entitled “SDMN Diagram Interchange” for more information.</td>
</tr>
</tbody>
</table>
10 SDMN Model Elements

This chapter defines DataItem and its related elements and ItemDefinition and its related elements.

10.1 DataItems

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. If the internal content of the DataItem is known, an XML Schema, describing the DataItem, may be imported.

DataItems can be organized into arbitrary hierarchies either by containment or by composition.

The data structure these elements hold is specified using an associated ItemDefinition. A DataItem 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 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.

![DataItem Metamodel](image)

Figure 7 - The DataItem Metamodel
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 SharedDataModel.

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 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 DataItem is set to anything other than folder, which has a different notation as shown below.

![Figure 8 - A DataItem Object](image)

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

![Figure 9 - A DataItem Object](image)

Generalizations

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

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

Properties

The following table presents the additional attributes and/or associations for DataItem:
Table 6. DataItem Attributes and/or Associations

<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>isCollection : Boolean [0..1] = false</td>
<td>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.</td>
</tr>
<tr>
<td>itemKind : String [0..1] default : Information</td>
<td>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.</td>
</tr>
<tr>
<td>metaDefinitionRef : ItemDefinition [0..1]</td>
<td>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.</td>
</tr>
<tr>
<td>multiplicityKind: String [0..1] default: ExactlyOne</td>
<td>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.</td>
</tr>
<tr>
<td>preAssignment : Assignment [0..1]</td>
<td>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.</td>
</tr>
<tr>
<td>typeDefinitionRef : ItemDefinition [0..1]</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 1. ItemKind Values

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conceptual</strong></td>
<td>The type of the <strong>DataItem</strong> 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 <strong>Conceptual DataItem</strong>, they are used to document aspects of the modeled behaviors.</td>
</tr>
<tr>
<td><strong>Information</strong></td>
<td>The type of the <strong>DataItem</strong> that fully utilizes the structural data capabilities inherent to <strong>ItemDefinition</strong>. If the <strong>DataItem</strong> has a <code>typedefinitionRef</code> (to an <strong>ItemDefinition</strong>), then the value of <code>itemKind</code> <strong>MUST</strong> be <strong>Information</strong>.</td>
</tr>
<tr>
<td><strong>Document</strong></td>
<td>This represents a Data Object or Case File Item that is a type of <strong>Document</strong>. In <strong>BPMN</strong>, the document could be physical (e.g., printed) or electronic. In <strong>CMMN</strong>, 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 <strong>CMMN</strong> Case File Item that is a <strong>Folder</strong>. A <strong>Folder</strong> can contain other <strong>Folders</strong> or <strong>Documents</strong>. Neither <strong>BPMN</strong> nor <strong>DMN</strong> have the concept of a <strong>Folder</strong> as a data element. Thus, <strong>DataItems</strong> based on a <strong>Folder</strong> would not map to <strong>BPMN</strong> or <strong>DMN</strong> data elements. The <strong>Folder</strong> 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 <strong>ItemKind</strong> is represents objects in a <strong>BPMN</strong> Process that are physical objects, such as printed documents or manufactured items. These types of <strong>DataItems</strong> are not currently relevant to <strong>CMMN</strong> or <strong>DMN</strong>.</td>
</tr>
<tr>
<td><strong>Relationship</strong></td>
<td>The <strong>ItemKind</strong> 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 <strong>ItemKind</strong> is represents a <strong>UML</strong> Class in a Class Diagram.</td>
</tr>
<tr>
<td><strong>Unknown</strong></td>
<td>The <strong>ItemKind</strong> 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 <strong>ItemKind</strong> 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 <strong>ItemKind</strong> is represents a <strong>WSDL</strong> Message.</td>
</tr>
<tr>
<td><strong>XSDComplexType</strong></td>
<td>For <strong>ItemKinds</strong> of this type, the (SCE) <strong>Import</strong> class <strong>SHOULD</strong> be used to import an XML Schema definition into the <strong>Shared Data Model</strong>. The <strong>ItemKind</strong> 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>
</tbody>
</table>
**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

### 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.

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>ExactlyOne</td>
<td>There is one copy of this DataItem.</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>

### 10.1.2 Pre-Assigning Values for DataItems

There are situations in the development of a collection of BPM+ 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 with 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 RootElement** (see the SCE Specification for more information).

**Properties**

The following table presents the additional attributes and/or associations for Assignment:
### Table 8. 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.

```json
{
   "Medication" : {
      "Id" : "medication1Medication" ,
      "Code" : {
         "System" : "http://www.nlm.nih.gov/research/umls/rxnorm" ,
         "Code" : "23456" ,
         "Text" : "Medication"
      },
      "Form" : {
         "Coding" : {
            "System" : "http://snomed.info/sct" ,
            "Code" : "385055001" ,
            "Display" : "Tablet dose form"
         },
         "Text" : "Tablet dose form"
      },
      "Ingredient" : {
         "Substance" : {
            "Id" : "medication1Substance" ,
            "Code" : {
               "Coding" : {
                  "System" : "http://www.nlm.nih.gov/research/umls/rxnorm" ,
                  "Code" : "789012" ,
                  "Text" : "Substance"
               }
            },
            "Strength" : {
               "Numerator" : {
                  "Value" : "25" ,
                  "Unit" : "mg"
               },
               "Denominator" : {
                  "Value" : "1" ,
                  "Unit" : "{tablets}" ,
                  "Text" : "Tablet"
               }
            }
         }
      }
   }
}
```

#### 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.
10.2.1 ItemDefinition

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

![Figure 11 - An ItemDefinition Object with no ItemComponent](image)

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.
**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 - A 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 specification for more information).

---

**Properties**

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

<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemComponent name</td>
<td></td>
</tr>
<tr>
<td>typeRef</td>
<td></td>
</tr>
<tr>
<td>allowedValues</td>
<td></td>
</tr>
</tbody>
</table>

---
Table 9. ItemDefinition Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conceptReference : URI [0..1]</td>
<td>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 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.</td>
</tr>
<tr>
<td>itemDefinitionRef : QName [0..1]</td>
<td>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, BPMN 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.</td>
</tr>
</tbody>
</table>

10.3 Connectors

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.

Figure 15 - The Connectors Metamodel
Generalizations

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

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

Properties

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

Table 10. Connector Attributes and/or Associations

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

10.3.2 CompositionConnector

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

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 a larger “health record” DataItem, will help focus the model for the context that is being addressed at that point.

CompositionConnectors are contained in a SharedDataModel.

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

- If the source DataItem is deleted, then the target DataItem will also be deleted.
- If a source DataItem is moved or set within a container, then the target DataItem will also be moved.
- If a target DataItem within a source DataItem is updated (e.g., through a change in the value of a property), the source DataItem will also be updated. I.e., the source DataItem is aware of changes to target DataItems.

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:

Figure 16 - A Composition Connector
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:

- `SCE RoleElement` (see the SCE Specification for more information).

Properties

The `Composition Connector` element does not have any additional attributes and/or associations.

10.3.3 ContainmentConnector

A `ContainmentConnector` is used to define a relationship between `DataItems`. This relationship will specify that one `DataItem` is contained within another `DataItem`. Container `DataItems` must have their `ItemKind` set to either folder or physical. They are contained in a `SharedDataModel`.

The runtime consequences of creating 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 `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`.

Notation

The following statements define the notation for a `ContainmentConnector`:

- A `ContainmentConnector` 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.
  - 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 `ContainmentConnector`:

![Figure 17 - A Containment Connector](image_url)

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

**Properties**

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

### 10.3.4 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.

![Metamodel Diagram](image)

**Figure 18 - The DataAssociation Metamodel**

**Notation**

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.
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 CompositionConnector 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 19 - 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 20 - 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 RootElement (see the SCE Specification for more information).

**Properties**

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

<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>
10.3.5 ReferenceConnector

An ReferenceRelationship is used to define a relationship between DataItems. This relationship will specify that one DataItem is references another DataItem. 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 source DataItem but the referenced DataItem exists on its own and can be referenced by other DataItems.

They are contained in a SharedDataModel.

The runtime consequences of creating a ReferenceConnector 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.

Notation

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 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 ReferenceConnector:

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 (see the SCE Specification for more information).

Properties

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

![Diagram of model artifacts](image)

Figure 22 - The Use of SCE Artifacts in SDMN

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**

The table below displays the graphical elements of SCE’s Model Artifacts:
Table 12. Shared Data Model Graphical Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Notation</th>
</tr>
</thead>
</table>
| Association        | 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.  
An Association is a line that is drawn with a dotted single line. An angle 30° arrowhead may optionally be added to either end of the line. |          |
| Group              | 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.

11 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.

11.1 Element Terminology Mapping to BPM+ Element Terminology

The following table defines the mapping for terms across the BPM+ languages and SDMN:
### Table 13. Mapping to BPMN

<table>
<thead>
<tr>
<th>SDMN Element</th>
<th>BPMN Element</th>
<th>CMMN Element</th>
<th>DMN Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containment Connector</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Composition Connector</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Data Association</td>
<td>Data Association</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>DataItem</td>
<td>Item Aware Element (Data Object, Data Object Reference, Data Input, Data Output, Data Store, Data Store Reference, Property)</td>
<td>Case File Item</td>
<td>Information Item (Data Input, Decision Output)</td>
</tr>
<tr>
<td>ItemDefinition</td>
<td>ItemDefinition</td>
<td>CaseFileItemDefinition</td>
<td>ItemDefinition</td>
</tr>
<tr>
<td>ItemKind</td>
<td>ItemKind</td>
<td>definitionType</td>
<td>N/A</td>
</tr>
<tr>
<td>isCollection</td>
<td>isCollection</td>
<td>N/A</td>
<td>isCollection</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>N/A</td>
<td>multiplicity</td>
<td>N/A</td>
</tr>
<tr>
<td>Pre-Assignment</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Reference Connector</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>SharedDataModel</td>
<td>Definitions</td>
<td>Definitions</td>
<td>Definitions</td>
</tr>
</tbody>
</table>

#### 11.2 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 14. ItemKind Mapping

<table>
<thead>
<tr>
<th>SDMN ItemKinds</th>
<th>BPMN itemKind Literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual</td>
<td>Information (this is to allow a formal documentation of the characteristics of the Conceptual DataItem.)</td>
</tr>
<tr>
<td>Information</td>
<td>Information</td>
</tr>
<tr>
<td>Document</td>
<td>Information</td>
</tr>
<tr>
<td>Folder</td>
<td>Information</td>
</tr>
<tr>
<td>Physical</td>
<td>Physical</td>
</tr>
<tr>
<td>Relationship</td>
<td>Information</td>
</tr>
<tr>
<td>UMLClass</td>
<td>Information</td>
</tr>
<tr>
<td>Unknown</td>
<td>Information</td>
</tr>
<tr>
<td>Unspecified</td>
<td>Information</td>
</tr>
</tbody>
</table>
Element Mapping

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

**Table 15. 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></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></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/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>

11.3 CMMN

This section provides mapping from SDMN to CMMN.

**ItemKind Mapping**

The following table defines the mapping from SDMN ItemKind instances to the CMMN definitionType property:

**Table 16. 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>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>SDMN Element/Attribute</td>
<td>CMMN Element/Attribute</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<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>Item Definition</td>
<td>If the ItemDefinition for a DataItem is of type</td>
</tr>
<tr>
<td></td>
<td>definitionRef, then the ItemDefinition will mapped to a</td>
</tr>
<tr>
<td></td>
<td>CMMN ItemDefinition for an CaseFileItem.</td>
</tr>
<tr>
<td></td>
<td>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/preAssignment</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Element Mapping**

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

**Table 17. Mapping to CMMN**

<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>
</tbody>
</table>

**11.4 DMN**

This section provides mapping from SDMN to DMN.

**Element Mapping**

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

**Table 18. 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>
</tbody>
</table>
Item Definition

<table>
<thead>
<tr>
<th>Item Definition</th>
<th>Item Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the <code>ItemDefinition</code> for a <code>InformationItem</code> is of type <code>definitionRef</code>, then the <code>ItemDefinition</code> will be mapped to a DMN <code>ItemDefinition</code> for an <code>InformationItem</code>.</td>
<td></td>
</tr>
<tr>
<td>If the <code>ItemDefinition</code> for a <code>DataItem</code> is of type <code>metaDefinitionRef</code>, then the contents of the <code>ItemDefinition</code> will be ignored. There is no equivalent in DMN. A separate <code>ItemDefinition</code> will not be created.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DataItem/ItemKind</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataItem/preAssignment</td>
<td>N/A</td>
</tr>
</tbody>
</table>

12  SDMN Examples

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 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.
Figure 23 - An Example of How a SDMN DataItem Diagram

The following table lists the data elements used by the BPM+ models.
Table 19. 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 Choice</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30. Weight Counseling Request Choice 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 **type** of the element and any defined **allowedValues** of that type (see “tEthnicity” 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 “tPatient_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 “tPatient_Health_Record” and “tPatient_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 “tPatient_Health_Record”). That is why the type section is empty since the contents of the type 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

13 Exchange Formats

13.1 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.

XMI 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.

13.2 XSD

13.2.1 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 <sdmn:sharedDataModel>. 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 `<sce:import>` element.

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.

13.2.2 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 QNames. 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.

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 [qualifier].[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.

14 SDMN Diagram Interchange (SDMN DI)

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.

14.2 Diagram Definition and Interchange

The SDMN DI 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.

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.
14.3 SDMN Diagram Interchange Meta-Model

14.3.1 How to read this Chapter

Clause 14.3.4 describes in detail the meta-model used to keep the layout and the look of SCE Diagrams (as a basis for SDMN Diagrams). Clause 14.3.5 presents in tables a library of the SCE element depictions and an unambiguous resolution between a referenced SDMN model element and its depiction.

14.3.2 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 [SCEDiagram] for interchange requires the specification of a collection of shapes [SCEShape] and edges [SCEEdge].

The SCE DI 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 element [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 SCEDiagram is an ordered collection of mixed SCEShape(s) and SCEEdge(s). The order of the SCEShape(s) and SCEEdge(s) inside a SCEDiagram determines their Z-order (i.e., what is in front of what). SCEShape(s) and SCEEdge(s) that are SCEEdge(s) MUST appear after them in the SCEDiagram. Thus, the exporting tool MUST order all SCEShape(s) and SCEEdge(s) such that the desired depiction can be rendered.

14.3.3 Measurement Unit

As per OMG DD, all coordinates and lengths defined by SDMN 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).

14.3.4 Elements

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

14.3.4.1 SDMN DI

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

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 SCEShape(s) and SCEEdge(s) in the SCEDiagram (see sections above). The SCEShape(s) and SCEEdge(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 SCEShape(s) and SCEEdge(s) MUST reference SDMN model elements. If no BaseElement 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 SCEShape or SCEEdge 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 [BaseElement] and their depiction. Depiction resolution tables are provided below for both SCEShape and SCEEdge.

### 14.3.5.1 Labels

Both SCEShape and SCEEdge 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 SCEShape and SCEEdge. When there is a label, the position of the label is specified by the bounds of the SCELabel of the SCEShape or SCEEdge. Simply put, label visibility is defined by the presence of the SCELabel element.

The bounds of the SCELabel are optional and always relative to the containing SCEDiagram's origin point. The depiction resolution tables provided below exemplify default label positions if no bounds are provided for the SCELabel (for SCEShape kinds and SCEEdge kinds (see sections above)).

When the SCELabel is contained in a SCEShape, the text to display is the name of the BaseElement.

### 14.3.5.2 SDMNShape Resolution

SCEShape can be used to represent a DataItem or an ItemDefinition.

**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 will only be 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>DataItem (Folder)</td>
<td>Shapes of DataItem that have itemKind=Folder</td>
<td><img src="image" alt="Folder Shape" /></td>
</tr>
<tr>
<td>DataItem (not Folder)</td>
<td>Shapes of DataItem that have itemKind!=Folder</td>
<td><img src="image" alt="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>Multiplicity = ZeroOrMore or OneOrMore or isCollection = true</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

**ItemDefinitions**

An ItemDefinition is represented in a SDMN Diagram as one of two possible shapes, depending on if the ItemDefinition has an itemComponent defined.

The following table presents the depiction resolutions for Itemdefinitions:

<table>
<thead>
<tr>
<th>SDMN Element</th>
<th>SDMNShape Attributes</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemDefinition</td>
<td>No SDMNComponent present</td>
<td>![Image]</td>
</tr>
<tr>
<td>ItemDefinition</td>
<td>With 1 or more itemComponent</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

**14.3.5.3 SDMNEdge Resolution**

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

**CompositionConnector**

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>![Image]</td>
</tr>
</tbody>
</table>

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>![Image]</td>
</tr>
</tbody>
</table>
**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>

**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>

**ReferenceConnector**

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="image" 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="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>