Date: November 2022

FACE Profile for UAF

Version 1.0

OMG Document Number: formal/22-12-03

Specification URL: https://www.omg.org/spec/FACE/1.0

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.

For a complete list of trademarks, see: https://www.omg.org/legal/tm_list.htm. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

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

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

OMG's Issue Reporting Procedure

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

Table of Contents

Preface

About the Object Management Group

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 Meta-model); and industry-specific standards for dozens of vertical markets.
More information on the OMG is available at *https://www.omg.org/*.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Formal Specifications are available from this URL: *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/IEC standards. Please consult: https://www.iso.org

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to: https://www.omg.org

1. Scope

This specification defines a profile to express The Open Group® Future Airborne Capability Environment (FACE™)[1] Technical Standard, Edition 3.0 and associated Meta-Object Facility (MOF) data architecture metamodel in terms of the Object Management Group's (OMG) Unified Architecture Framework (UAF).

1.1 FACE Profile for UAF Background

The FACE Profile for UAF v1.0 specification defines a profile to express The Open Group® FACE™ (Future Airborne Capability Environment) Technical Standard, Edition 3.0 and associated Meta-Object Facility (MOF) data architecture metamodel in terms of the Object Management Group's (OMG) Unified Architecture Framework (UAF). Where there is no corresponding concept for a FACE metamodel element in the UAF standard, this specification reverts to the UML metamodel as its basis.

The FACE Technical Standard is a software open architecture specification that "defines the software computing environment intended for the development of portable software components, including requirements for architectural segments and key interfaces."[2] The focus of the FACE Technical Standard is the support of real-time and safety critical software beginning with avionics. The FACE Technical Standard has been used in military and commercial avionics as well as in other industrial control systems such as power control and communications systems.

"UAF defines ways of representing an enterprise architecture that enables stakeholders to focus on specific areas of interest in the enterprise while retaining sight of the big picture … to meet the specific business, operational and systems-of-systems integration needs of commercial and industrial enterprises as well as the U.S. Department of Defense (DoD), the UK Ministry of Defence (MOD), the North Atlantic Treaty Organization (NATO) and other defense organizations."[3]

The UAF standard provides the larger scope to describe the environments into which the FACE-described components fit. The Open Group FACE Technical Standard defines the elements, attributes and associations for the FACE Data Architecture and includes descriptions of software components to be included in larger system-of-systems architectures. The UAF standard provides a mechanism for defining the larger context in which the FACE components reside. UAF provides representations for defense and non-defense architectures that can be used to effectively combine FACE software components and other systems components into cohesive systems architectures.

Together, the FACE Profile for UAF will enable platform and enterprise level acquisition analysis, software security and cybersecurity analysis, and rapid capability development and deployment. The definition of this profile is the first step. The implementation and realization of this profile in software and systems engineering tools, followed by organizational utilization of these tools and standards will be necessary "to ensure that systems that are being developed are interoperable and meet the overarching capabilities that they were intended to achieve."[4]

1.2 Intended Users

The profile enables the modeling of FACE components, data descriptions, data exchanges, integration elements, and traceability mechanisms in specification of system-of-systems airframe architectures. It is intended to be used in project and system planning as well as to inform acquisition and integration efforts. This specification is intended to be used by tools implementors, computer scientists, data scientists, software engineers, systems engineers, and software systems engineers. For the best application of this profile, users should have some familiarity or background with UAF and the FACE approach as well as UML and OCL.

---

[1] FACE™ is a trademark of The Open Group®.

[2] FACE FAQs | The Open Group. (2020). Opengroup.org. Retrieved 14 February 2020, from https://www.opengroup.org/content/future-airborne-capability-environment-face/faqs

[3] "Unified Architecture Framework® (UAF®) | Object Management Group. (2020). omg.org. Retrieved 14 February 2020, from https://www.omg.org/uaf/index.htm"

[4] FACE FAQs | The Open Group. (2020). Opengroup.org. Retrieved 14 February 2020, from https://www.opengroup.org/content/future-airborne-capability-environment-face/faqs

1.3 Related Documents

The specification includes a metamodel and description as separate documents. Other supporting and descriptive resources are provided as separate documents. The table below provides a listing of these documents.

Table 1-1 – Table of Related Documents

Zip archive containing references that can be used in the construction of the FACE Architectural Segment illustrations.
Informational files (Eclipse exports & Excel spreadsheet) describing the “unified” UML profile created from existing FACE Consortium member UML Profile contributions. The “unified” UML profile was the starting point for this specification.

2. Conformance

The FACE Profile for UAF is dependent upon UAF. It defines constraints that pair together with application of UAFP stereotypes. There are three levels of conformance designated for the FACE Profile for UAF. The requirements for a tool to be considered as conformant with the FACE Profile for UAF at each level of conformance are detailed below.

2.1 Level A Conformance

Level A is the lowest level of conformance. Level A Conformance provides the basic profile and constraints that are based on the FACE metamodel, along with enhanced export/import that includes both the FACE and UAF model elements. This is the minimum implementation that can meet the conformance requirements of this standard.

Table 2-1: Level A Conformance Points

Implementation of profile stereotypes	All stereotypes, classes, attributes, associations, and package structures must exist and be conformant with this specification.
XMI data exchange	Provide XMI import and export (.xmi) of the user model and profile, including both UAF and FACE elements.
Fidelity of XMI exchange	Be able to import and export FACE Profile for UAF models with 100% fidelity (i.e., no loss or transforms).
Basic constraints only	Application of only “Constraint” constraints (no requirement for FACE Conformance/OCL Constraints).
FACE Element Aggregation Tables	Provide a mechanism to generate the specified tabular views that aggregate FACE constructs.

2.2 Level AA Conformance

Level AA Conformance is a mid-range level of conformance. AA Conformance includes all Level A conformance points and adds .face file format export and import (round-tripping) in support of external checks for FACE model conformance. Level AA Conformance provides the minimum support needed by the users of FACE data architecture models in order to use the authored information in a FACE integration effort.

Table 2-2: Level AA Conformance Points

Level A Conformance	Level A conformance criteria met.
.face file XML data exchange	Provide import and export of FACE elements in the FACE XML (.face) format as specified in the XMI Specification (document c4i/20-02-02) delivered with this document.
Fidelity of .face file exchange	Be able to import and export FACE elements to and from FACE XML models (.face files) with 100% fidelity (i.e., no loss or transforms).

2.3 Level AAA Conformance

Level AAA Conformance is the highest level of conformance. AAA Conformance supports the rapid development of FACE architecture, data models, and software development through application of the FACE/OCL Constraints during the architecture modeling process. By applying these constraints during the model authoring process, the user is spared export of the data model for conformance testing.

Table 2-3: Level AAA Conformance Points

Level AA Conformance	Level AA conformance criteria met (includes Level A).
Basic PLUS FACE Conformance/OCL Constraints	All constraints must exist and be conformant with this specification.
FACE Conformance Checks in tool	FACE Conformance checking in tool using FACE Conformance/OCL Constraints.

3. 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.

3.1 OMG Documents (Normative References)

Meta Object Facility (MOF), v2.5.1, October 2016, https://www.omg.org/spec/MOF/

Unified Modeling Language (UML), v2.5.1, December 2017, https://www.omg.org/spec/UML

Object Constraint Language (OCL), v2.4, February 2014, https://www.omg.org/spec/OCL

System Modeling Language (SysML), v1.6, December 2019, https://www.omg.org/spec/SysML

Diagram Definition (DD), v1.1, August 2015, https://www.omg.org/spec/DD

Unified Architecture Framework (UAF), v1.1, April 2020, https://www.omg.org/spec/UAF/

Interface Definition Language (IDL), v4.2, March 2018, https://www.omg.org/spec/IDL/

UML Profile for MARTE, v1.2, April 2019, https://www.omg.org/spec/MARTE/

3.2 Other Normative References

- FACE Technical Standard, Edition 3.0
    - Open Group FACE™ Consortium, The Open Group FACE™ (Future Airborne Capability Environment) Technical Standard, Edition 3.0, 15 November 2017, accessed 26 August 2019, <https://www.opengroup.org/library/c17c>
    - The written FACE Technical Standard remains the normative standard FACE Architecture, and most importantly, conformance. The profile presented in this standard follows the metamodel in section J of the FACE Technical Standard, Edition 3.0 for all elements other than the Conceptual/Logical/Platform CharacteristicPathNode, ParticipantPathNode, and PathNode elements. Those metamodel elements are represented in the stereotypes FACE_ConceptualParticipant, FACE_LogicalParticipant, and FACE_PlatformParticipant as strings in the stereotypes' "path" tagged values. The path strings for these stereotypes use the notation described in Section 3.6.4.1.1.3 of the Technical Standard for Future Airborne Capability Environment (FACE™), Edition 2.1. The two notations (elements and string) are interchangeable using a translation algorithm. XMI exchange mechanisms between models using the FACE Profile for UAF and the FACE XMI (face) file are required to translate between the two notations.
- FACE Technical Standard, Edition 2.1
    - Open Group FACE™ Consortium, The Open Group FACE™ (Future Airborne Capability Environment) Technical Standard, Edition 2.1, 24 June 2014, accessed 26 August 2019, <https://www.opengroup.org/library/c145>
    - The expression of FACE Path data in this specification refers to the path notation in the FACE 2.1 Technical Standard. The FACE Technical Standard, Edition 2.1 is referenced in this standard solely for the purpose of simplifying the expression of FACE Path elements. The profile presented in this standard follows the metamodel in section J of the FACE Technical Standard, Edition 3.0 for all elements other than the Conceptual/Logical/Platform CharacteristicPathNode, ParticipantPathNode, and PathNode elements. Those metamodel elements are represented in the stereotypes FACE_ConceptualParticipant, FACE_LogicalParticipant, and FACE_PlatformParticipant as strings in the stereotypes' "path" tagged values. The path strings for these stereotypes use the notation described in Section 3.6.4.1.1.3 of the Technical Standard for Future Airborne Capability Environment (FACE™), Edition 2.1. The two notations (elements and string) are interchangeable using a translation algorithm. XMI exchange mechanisms between models using the FACE Profile for UAF and the FACE XMI (face) file are required to translate between the two notations.

- FACE Metamodel, emof 2.0 machine-readable format
    - URL: https://www.opengroup.org/face/3.0/Normative%20FACE%20metamodel%203.0.emof
    - This is the metamodel as it appears in the FACE Technical Standard.

3.3 Informative References

- FACE Metamodel, emof 2.5.1 machine-readable format
    - URL: https://www.opengroup.org/face/3.0/Informative%20FACE%20metamodel%203.0%20EMOF%202.5.1.emof
    - The FACE metamodel file from the specification, updated to the newer EMOF 2.5.1 format. As this file is not a formal part of the FACE Technical Specification, it is informative rather than normative
- FACE 3rd Party Tools
    - These tools located at URL: https://web.archive.org/web/20170915185145/http://www.isis.vanderbilt.edu/FACE
    - FACE Edition 2.1 EA Data Model Profile and Plugins NAVAIR Public Release 2015-746
    - FACE Edition 2.1 Rhapsody Data Model Profile and Plugins NAVAIR Public Release 2015-746
    - FACE Conformance Test Suites
- FACE Consortium Conformance Publications & Tools
    - These publications and tools located at URL: https://www.opengroup.org/face/conformance-publications-and-tools
    - FACE™ Conformance Verification Matrix, Edition 3.0
- FACE New Users Resources
    - These resources available at URL: https://www.opengroup.org/face/#NewUser
    - Basic Avionics Lightweight Source Archetype (BALSA), a working software example of applications aligned to the FACE Technical Standard executing in a FACE Reference Architecture (includes data model in .face format)

4. Terms and Definitions

Table 4-1: Symbols in the Specification

ARINC	Avionics Application Standard Software Interface
DAL	Design Assurance Level
FACE	Future Airborne Capability Environment
GCM	General Component Model
IOSS	Input/Output Services Segment
MARTE	Modeling and Analysis of Real-Time and Embedded systems
OCL	Object Constraint Language
OSS	Operating System Segment
PCS	Portable Components Segment
PSSS	Platform-Specific Services Segment
TSS	Transport Services Segment
UAF	Unified Architecture Framework

5. Symbols

No new symbols have been required to create this specification.

6. Additional Information

6.1 Changes to Adopted OMG Specifications [optional]

This specification requires no changes to previously adopted OMG specifications.

6.2 Acknowledgments

The following companies submitted this specification:

- The MITRE Corporation
- No Magic Inc. a Dassault Systemes Company
- SimVentions, Inc.

The following companies contributed to this specification:

- CRL Technologies
- Lockheed Martin
- Sparx Systems

6.3 Scope of this Specification

This specification covers the entire scope of the FACE Technical Standard, Edition 3.0 metamodel and includes references to the FACE Technical Standard, Edition 3.0 OCL.

Rationale for complete FACE Metamodel:

- Coupling between Views (mandatory requirements) and the remaining FACE Data Model elements – cannot determine the size or type of data being exchanged between components without the underlying data model.
- Traceability Model (mandatory requirements) is dependent on FACE Data Model elements.
- Existing 3rd-party UML-based profiles and import/export plugins that extend to the entire scope of the FACE metamodel.
- Existing 3rd-party Import/Export plugins provide exchange between tool-authored data architecture and “gold standard” FACE XMI format.

6.4 How to Read this Specification

The rest of this document contains the technical content of this specification. As background for this specification, readers are encouraged to first read the FACE Technical Standard that is the basis for the elements contained herein (The Open Group FACE™ (Future Airborne Capability Environment) Technical Standard, Edition 3.0). That specification includes the FACE Data Architecture specification (Sections 3.9 and Appendix J.2), Object Constraint Language (OCL) rules (Appendix J.6), and EMOF metamodel (Appendix J.4 and J.5) that govern the FACE artifact, and from which all elements of this specification have been derived. After that, the UAF 1.1 specification provides needed background for understanding the UAF concepts and acts as a reference when considering the mapping of the FACE standard to the UAF and UML standards. The UAF and FACE standards provide the basic constructs used to define the FACE Profile for UAF.

6.4.1 Content Notes for this Specification

In the interest of avoiding potential inconsistencies between this specification and the FACE Technical Standard, Edition 3.0, this specification adds no information about FACE elements that is not present in that standard. This specification refrains from providing descriptions of FACE metamodel elements, associations, attributes, and enumeration elements that are not provided in the FACE Technical Standard. As such, these unspecified descriptions for metamodel attributes, relationships, and enumerated values within this specification will appear ‘blank’ where those descriptions would normally appear.

In the interest of clarity and of avoiding any possible name collisions with other profiles, all stereotypes and enumerations defined in this specification are prefixed with "FACE_". Where appropriate, this prefix has also been applied to the descriptions for stereotypes that correspond to elements in the FACE metamodel. The content of those descriptions otherwise remains unchanged from the corresponding descriptions in the FACE metamodel.

This specification introduces some abstract elements not found in the FACE metamodel. The additional abstract elements are provided in support of XMI data interchange with the FACE XMI Schema and/or application of constraints. They can be considered optional if not otherwise needed for conformant implementation of the profile.

This specification introduces some concrete elements not found in the FACE metamodel. The additional concrete elements are separated from the FACE Architecture element package and exist to supplement the FACE metamodel with elements that recognize the larger context of a UAF system-of-systems. The supplemental elements either represent FACE segments that are not explicitly represented in the FACE metamodel or provide connection between FACE Components and other components of a system-of-systems.

6.4.2 Representing Additional Properties and Constraints on Stereotypes

The FACE Profile for UAF follows the enhanced standard notation used in the UAF Standard to represent metaconstraints graphically. The FACE Profile has extended the metaconstraint notation to express application of stereotyped Associations and stereotyped Generalizations. The enhanced standard notation has been used both in this and in the UAF profile diagrams to improve readability of the profile specifications and overcome limitations of being unable to visualize constraints diagrammatically in UML.

The enhanced notation dependencies (metaconstraint, stereotyped relationship, stereotyped association, stereotyped generalization) appear in the FACE Profile for UAF specification diagrams for visualization purposes only. The representation in the standard varies by dependency stereotype:

- metaconstraint is represented in the standard is as a UML constraint, specified in structured English. These constraints are implementable in a tool, by OCL for example.
- A stereotyped relationship is represented in the standard by a correspondingly named stereotype with metatype Dependency. These dependencies are implemented using the corresponding stereotype and the constraints associated with them in the standard.
- A stereotyped association is represented in the standard by a correspondingly named stereotype with metatype Association. These associations are implemented using the metatypes and constraints associated with them in this standard.
- A stereotyped generalization is represented in the standard by a correspondingly named stereotype with metatype Generalization. These generalizations are implemented using the metatypes and constraints associated with them in this standard.

A simple UML profile defines the enhanced notation.

The following sub clauses detail the enhanced notation profile definition within the FACE Profile for UAF.

6.4.2.1 FACE Conformance/OCL Constraints

The FACE Conformance/OCL Constraints represented in this standard are representations of the FACE OCL Constraints listed in the FACE Technical Standard, Edition 3.0, section J.6. These constraints are not represented by any graphical notation in diagrams appearing in this standard but are included to provide additional information about the constraints needed for full conformance to the standard. The FACE Conformance/OCL Constraints descriptions have been taken from the FACE Technical Standard, Edition 3.0, with minor modifications to indicate the intent of the constraint (e.g., "is" changed to "must be"). For the full Object Constraint Language (OCL) expansions of the FACE Conformance/OCL Constraints, see the appropriate subsections of the FACE Technical Standard, Edition 3.0, Section J.6.

6.4.2.2 Metaconstraint Dependency

«metaconstraint» is a stereotype that extends the Dependency metaclass. It is used to specify constrained elements within the profile.

6.4.2.2.1 Definition of the Metaconstraint Dependency Stereotype

metaconstraint

isAbstract: No
Extension: Dependency

Description

The metaconstraint stereotype has been created for the purpose of expressing the FACE Profile for UAF specification and is not part of the profile itself. It is applied to dependencies between stereotypes to visually model constraints on the stereotypes' underlying UML metatypes. The umlRole Tag relates to a role on an existing meta-level association between metatypes (for example ownedAttribute on the association between Class and Property). This allows us to model the equivalent of redefines/subsets which has been used in the M3, for example, without creating unnecessary and unwanted associations between the stereotypes (as subset and redefine can only be used through inheritance and we're using extends). It has nothing to do with creating aggregation between stereotypes (i.e., tagged values). Tagged values are shown as stereotype properties in the profile.

Figure 6-1: metaconstraint Dependency (specification stereotype)

Attributes

umlRole :	String []	UML Role (property) of the source of the Dependency that is to be typed by the target of the Dependency.

6.4.2.2.2 Example Usage of the Metaconstraint Dependency

A sample of the «metaconstraint» dependency is a diagram for a stereotype extending the Association metaclass.

FACE_AbstractView is a FACE Profile for UAF stereotype that extends Association. The constraint on this stereotype is that its memberEnd[1].type end must be stereotyped by either a FACE_ConceptualView or a FACE_LogicalView (both of which are abstract) and its memberEnd[0].type must stereotyped by a FACE_AbstractConnection. But as it is not possible to show this constraint graphically, the diagram does not communicate the needed information without additional notation. We use the "metaconstraint" dependency to visualize the constraint. In order to fully describe the usage of the stereotype in the FACE Profile, the diagram also shows the application of the association stereotype «FACE_AbstractView» to create associations between the memberEnd[0].type and memberEnd[1].type stereotype elements (the «stereotyped association» Dependency is discussed in a subsequent section). Additional constraints on the Association properties are shown as «metaconstraint» dependencies FACE_AbstractView has with itself.

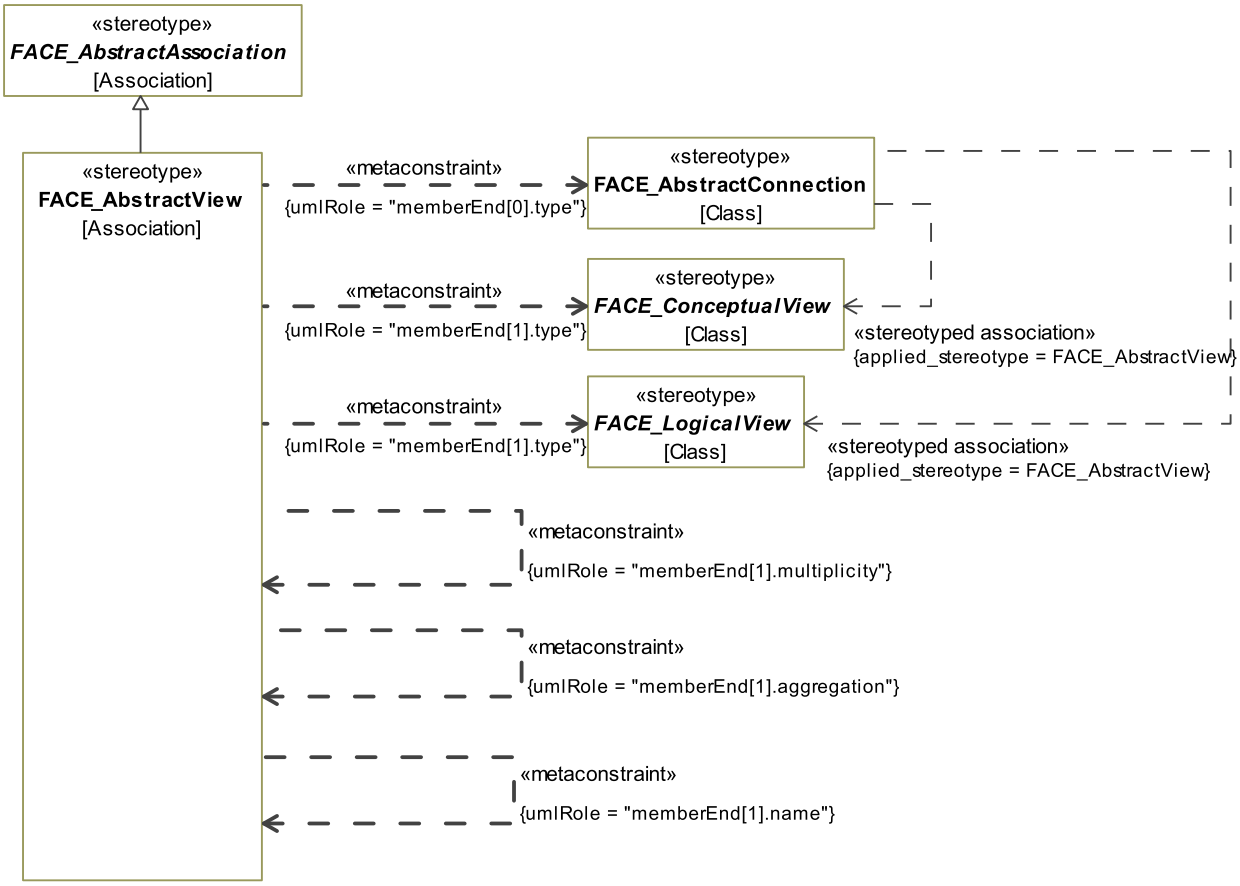

Figure 6-2: Use of «metaconstraint» dependency

6.4.2.3 Stereotyped Relationship Dependency

There are stereotypes in the profile specification that have Metaclass Dependency. While the constraints described for these stereotypes express the allowed sources and targets of these dependencies, when showing a diagram representing another stereotype in this profile it is also helpful to see how elements typed by that stereotype could be related to other elements using these dependencies. The stereotyped relationship dependency is a mechanism to graphically represent the application of stereotyped dependencies between elements of the FACE Profile for UAF and other elements.

6.4.2.3.1 Definition of the Stereotyped Relationship Dependency Stereotype

stereotyped relationship

Package: stereotyped dependencies
isAbstract: No
Extension: Dependency

Description

The «stereotyped relationship» stereotype has been created for the purpose of expressing the FACE Profile for UAF specification and is not part of the profile itself. It is applied to dependencies between stereotypes to visually model UML relationships that the profile explicitly dictates to be possible between model elements to which the stereotypes have been applied. The applied stereotype tag names the FACE profile stereotype for the dependency that is being expressed.

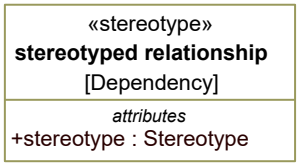

Figure 6-3: stereotyped relationship Dependency (specification stereotype)

Attributes

stereotype :	Stereotype []	The stereotype that applies to the Dependency depicted by the relationship. The "type" of Dependency that is being expressed by the depicted relationship.

6.4.2.3.2 Example Usage of the Stereotyped Relationship Dependency

For example, Figure 6-4 shows a partial definition of the FACE_Implements stereotype. While metaconstraints indicate a client and a supplier for the dependency, the «stereotyped relationship» dependency on the right side of the diagram indicates that an element stereotyped by FACE_AbstractUoP can have a FACE_Implements relationship with an element stereotyped by OperationalPerformer. In this specification, the «FACE_Implements» dependency indicating a potential relationship to OperationalPerformer is also reflected in the diagram describing FACE_AbstractUoP.

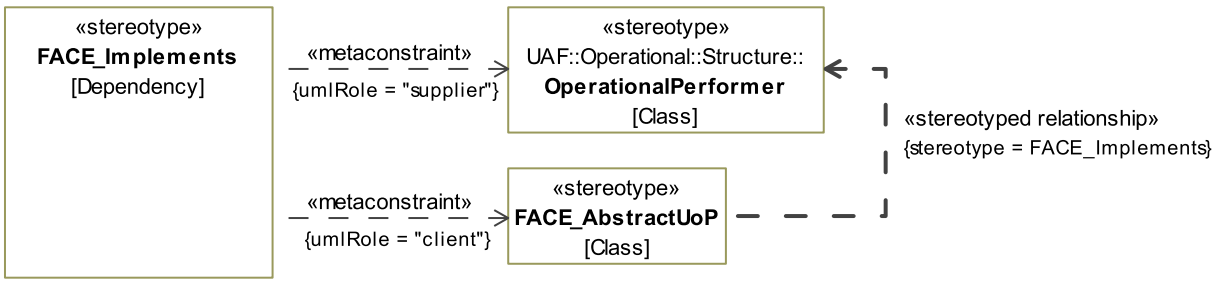

Figure 6-4: Use of «stereotyped relationship» dependency

6.4.2.4 Stereotyped Association Dependency

There are several stereotypes in the profile specification that have the Metaclass Association. There is no mechanism in the specification of the Association to express cardinality and aggregation, especially if the cardinality changes with the source/target pair. As a result, the FACE Profile for UAF introduces a notation to describe Associations that are part of the FACE Specification that have features that are normally properties of a UML Association. This information is represented using «stereotyped association» dependencies.

6.4.2.4.1 Definition of the Stereotyped Association Dependency Stereotype

stereotyped association

isAbstract: No
Extension: Dependency

Description

The stereotyped association stereotype has been created for the purpose of expressing the FACE Profile for UAF specification and is not part of the profile itself. It is applied to dependencies between stereotypes to visually model Associations that the profile explicitly dictates to be possible between model elements to which the stereotypes have been

applied. The applied stereotype tag names the FACE profile stereotype for the association that is being expressed. The stereotype referenced by the applied stereotype tag further describes the nature of the Association. The stereotyped association Dependency has nothing to do with creating aggregation between stereotypes (i.e., tagged values). Tagged values are shown as stereotype properties in the profile.

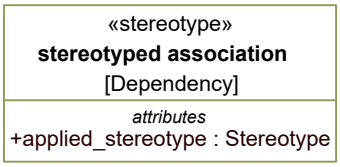

Figure 6-5: stereotyped association Dependency (specification stereotype)

Attributes

applied_stereotype :	Stereotype []	The stereotype that applies to the Association depicted by the Dependency. The "type" of Association that is being expressed by the Dependency.

6.4.2.4.2 Example Usage of the Stereotyped Association Dependency

For example, Figure 6-6 shows a subset of the definition of the FACE_Realize Association. It shows two of the metaconstraints that define the endpoints for the association, and on the right shows the application of the FACE_Realize association between FACE_UnitOfPortability and FACE_AbstractUoP. This indicates that an element stereotyped by FACE_UnitOfPortability may have a FACE_Realize association with an element stereotyped by FACE_AbstractUoP.

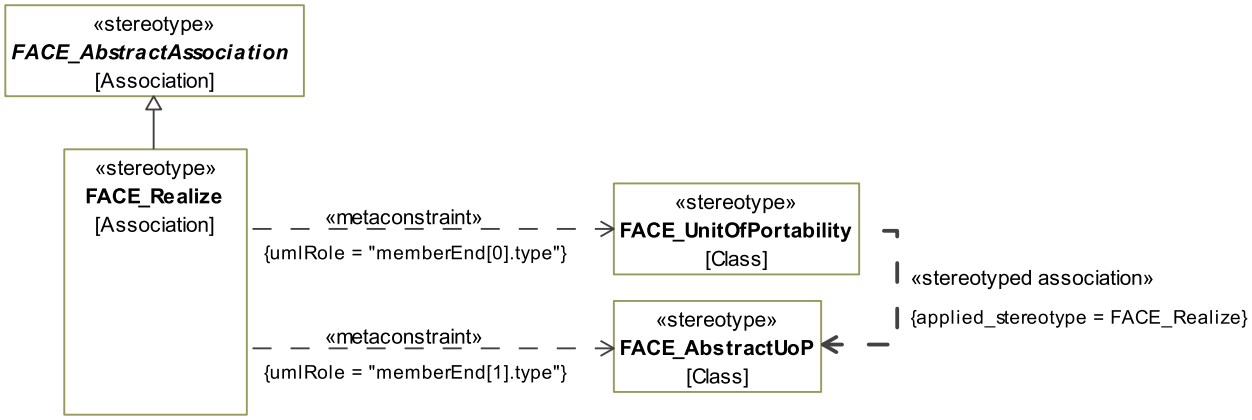

Figure 6-6: Use of «stereotyped association» dependency

6.4.2.5 Stereotyped Generalization Dependency Stereotype

The FACE Profile for UAF defines specific Generalization relationships in its data model. These generalizations are the only generalizations between elements stereotyped by FACE profile stereotypes that are meaningful to the FACE framework. The «stereotyped generalization» dependency in this specification is a means of graphical depiction for these generalization-specialization relationships.

6.4.2.5.1 Definition of the Stereotyped Generalization Dependency

stereotyped generalization

isAbstract: No
Extension: Dependency

Description

The stereotyped generalization stereotype has been created for the purpose of expressing the FACE Profile for UAF specification and is not part of the profile itself. It is applied to dependencies between stereotypes to visually model specialized Generalizations that the FACE Profile defines as applicable between model elements. The "stereotype" Tag indicates the FACE Profile stereotype that is applicable to the Generalization. The stereotype referenced by the "stereotype" tag further describes the nature of the Generalization. The stereotyped generalization Dependency has nothing to do with creating aggregation between stereotypes (i.e., tagged values). Tagged values are shown as stereotype properties in the profile.

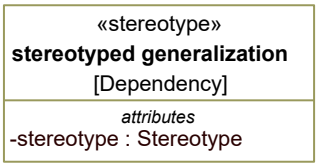

Figure 6-7: stereotyped generalization Dependency (specification stereotype)

Attributes

stereotype :	Stereotype []	The name of the stereotype that applies to the Generalization depicted by the Dependency. The "type" of Generalization that is being expressed by the Dependency.

6.4.2.5.2 Example Usage of the Stereotyped Generalization Dependency

Figure 6-8 shows the stereotyped generalization EntityBasis and its application between the FACE_ConceptualEntity and FACE_BasisEntity stereotypes. The diagram indicates that:

- FACE_ConceptualEntity is the source endpoint in a FACE_EntityBasis Generalization.
- FACE_BasisEntity is the target endpoint in a FACE_EntityBasis Generalization.
- FACE_ConceptualEntity may specialize FACE_BasisEntity using the FACE_EntityBasis Generalization.

The constraints for the EntityBasis stereotype are part of its specification section.

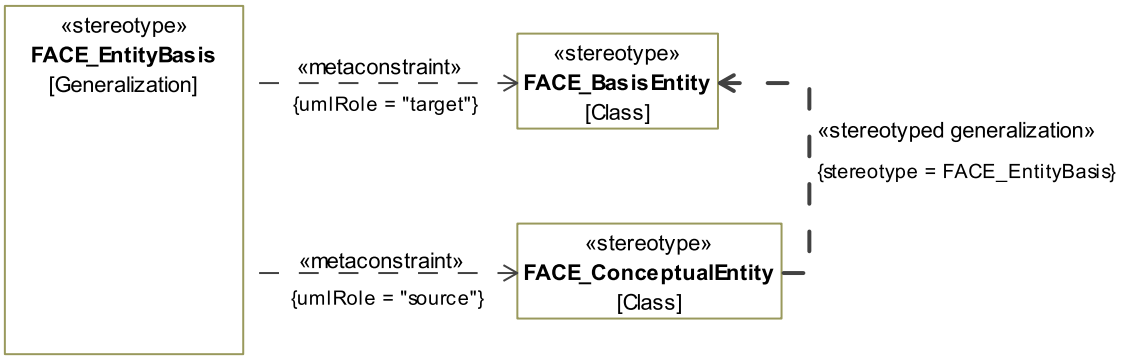

Figure 6-8: Use of «stereotyped generalization» dependency

7. Profile for UAF

Although FACE Profile for UAF implementations must use the UAF Profile in order to indicate implementation of UAF elements through dependencies, the FACE Profile for UAF itself imports only the UML metamodel.

The FACE-UAF Profile is the top-level profile root. The package structure of the FACE Profile is based on the FACE package structure in the FACE metamodel, as defined in the FACE Technical Standard, Edition 3.0.

7.1 FACE_UAF_Profile

The FACE_UAF_ Profile package contains the FACE Profile as derived from existing FACE Consortium member UML Profile contributions and mappings from FACE elements to UAF elements. The underlying UML profile represents a unification of multiple too-specific UML profiles written to describe the FACE 3.0 metamodel. After establishing the FACE UML stereotypes, the UAF stereotypes that best matched the FACE metamodel were applied using stereotyped Dependency relationships. The package organization of the FACE Profile for UAF mimics the FACE metamodel packages.

FACE_AbstractAssociation

Package: FACE_UAF_Profile
isAbstract: Yes
Extension: Association

Description

The FACE_AbstractAssociation stereotype exists to characterize the constraints that apply to all FACE Association stereotypes. These constraints and characteristics hold true unless overridden in a subclassed stereotype. By default, all FACE Stereotypes of metaclass Association are binary Associations (2 endpoints) with no aggregation from the "target" endpoint (memberEnd[1].aggregation = none) and default to directional (navigable only from memberEnd[0] to memberEnd[1]) . The default constraints can be overridden by constraints specified in specializations of this stereotype. Directionality and other association memberEnd properties vary and are specified with the Association stereotypes to which they apply.

This stereotype exists only for specification of constraints that apply to the specialized FACE Profile stereotypes. It is optional in the implementation of this specification.

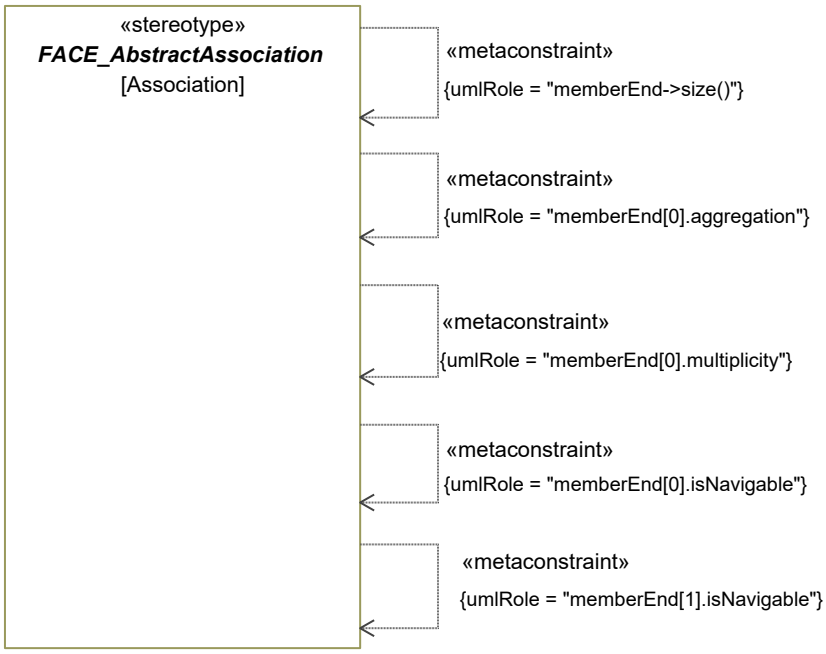

Figure 7-1: abstract FACE_AbstractAssociation

Constraints

[1] FACE_AbstractAssociation.memberEnd->size()	2
[2] FACE_AbstractAssociation.memberEnd[0].aggregation	none
[3] FACE_AbstractAssociation.memberEnd[0].isNavigable	false
[4] FACE_AbstractAssociation.memberEnd[0].multiplicity	1
[5] FACE_AbstractAssociation.memberEnd[1].isNavigable	true

FACE_ArchitectureModel

Package: FACE_UAF_Profile
isAbstract: No
Generalization: FACE_Element
Extension: Package

Description

A FACE_ArchitectureModel is a container for FACE_DataModels, FACE_UoPModels, FACE_IntegrationModels, and FACE_TraceabilityModels.

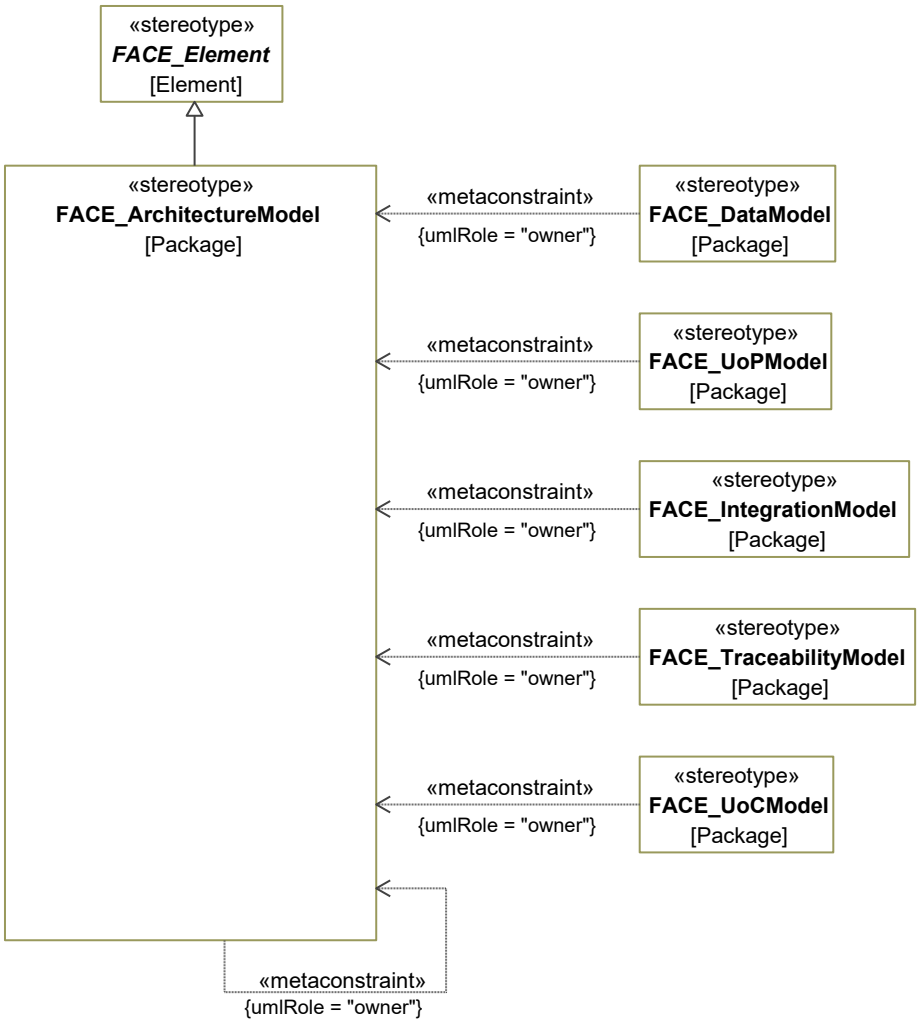

Figure 7-2: FACE_ArchitectureModel

Constraints

[1] FACE_ArchitectureModel.owner	This element may only be contained in (owned by) packages or architectures that are not stereotyped by a FACE stereotype

FACE Conformance/OCL Constraints

[1] FACE_ArchitectureModel.hasUniqueName	In the context of the entire FACE Architectural Model, the name of the element must be unique using case-insensitive tests.

FACE_Element

Package: FACE_UAF_Profile
isAbstract: Yes
Generalization: FACE_ModelElement

Description

A FACE_Element is the root type for defining all described elements in the FACE_ArchitectureModel. The description attribute captures a description for the element.

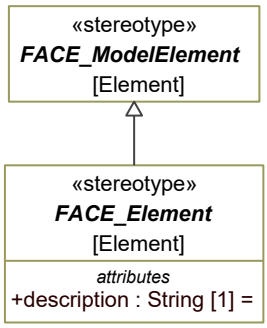

Figure 7-3: abstract FACE_Element

Attributes

description : String [1]

FACE_ModelElement

Package: FACE_UAF_Profile
isAbstract: Yes
Extension: Element

Description

An abstract stereotype used to represent the unique identity of constructed FACE model elements. Ensures that all FACE elements are identified by a GUID that is stable across all representations of the model, regardless of tool. Applied directly to FACE elements that are not specified to have a description in the FACE metamodel.

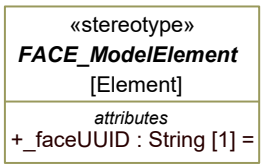

Figure 7-4: abstract FACE_ModelElement

Attributes

_faceUUID : String [1]	The FACE unique identifier for the element. FACE UUIDs are stable across all imports and exports of the FACE model regardless of tool and are maintained as part of the .face file. FACE UUIDs are generated as GUIDs for new (no previous FACE UUID) FACEModelElements upon export of a new or updated FACE architecture.

FACE Conformance/OCL Constraints

[1] FACE_ModelElement.hasUniqueName	Every Element in a FACE_ArchitectureModel must have a unique name.
[2] FACE_ModelElement.nameIsValidIdentifier	The name of a FACE_ModelElement must be a valid identifier.

7.1.1 FACE_UAF_Profile::FACE Data Architecture

FACE_DataModel

Package: FACE Data Architecture
isAbstract: No
Generalization: FACE_Element
Extension: Package

Description

A FACE_DataModel is a container for FACE_ConceptualDataModels, FACE_LogicalDataModels, and FACE_PlatformDataModels.

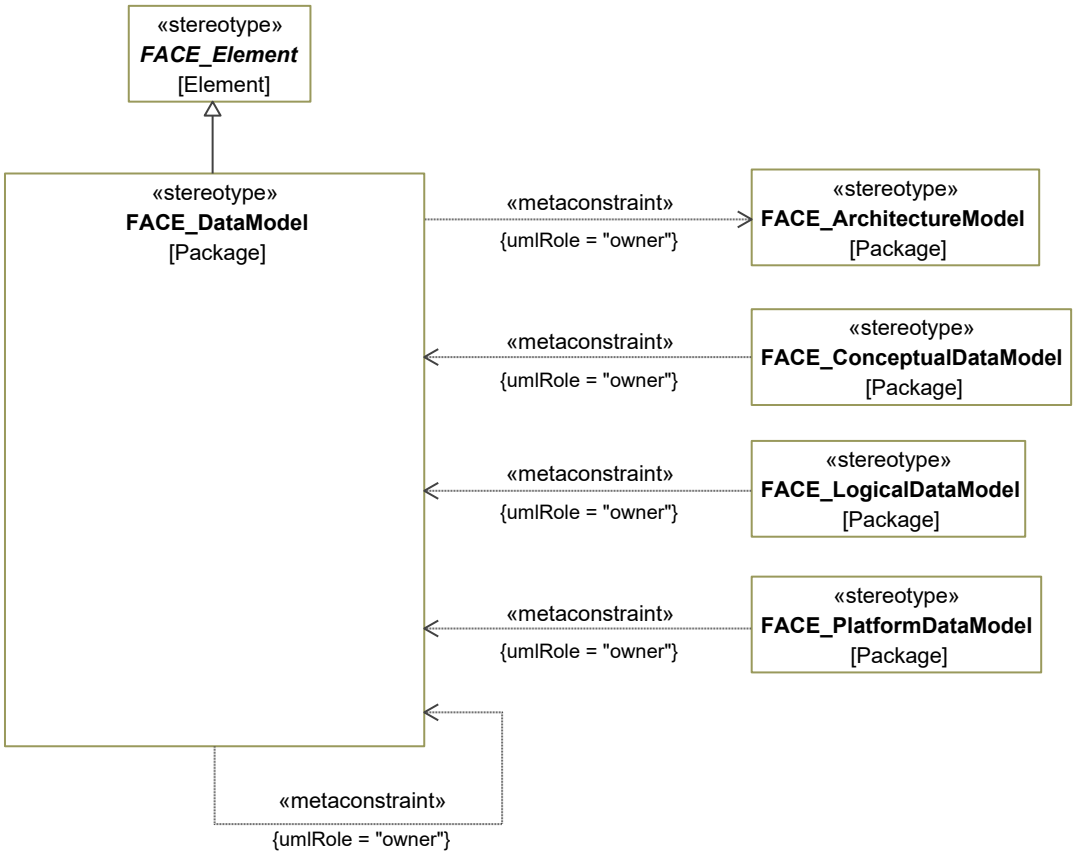

Figure 7-5: FACE_DataModel

Constraints

[1] FACE_DataModel.contains	The contained elements must by stereotyped one of the following:

	«FACE_DataModel» «FACE_ConceptualDataModel» «FACE_LogicalDataModel» «FACE_PlatformDataModel»
[2] FACE_DataModel.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_ArchitectureModel» «FACE_DataModel»

FACE Conformance/OCL Constraints

[1] FACE_DataModel.hasUniqueName	Each FACE Data Model Element must have a unique name.

FACE_DataModelElement

Package: FACE Data Architecture
isAbstract: Yes
Generalization: FACE_Element

Description

A FACE_DatamodelElement is the root type for defining the elements of the FACE Data Model Language.

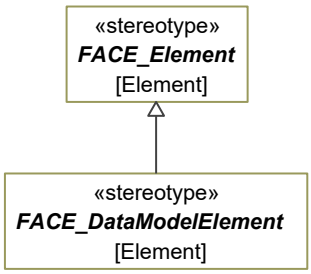

Figure 7-6: abstract FACE_DataModelElement

FACE_ElementTrace

Package: FACE Data Architecture
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to identify traceable FACE elements to the FACE_TraceabilityModel.

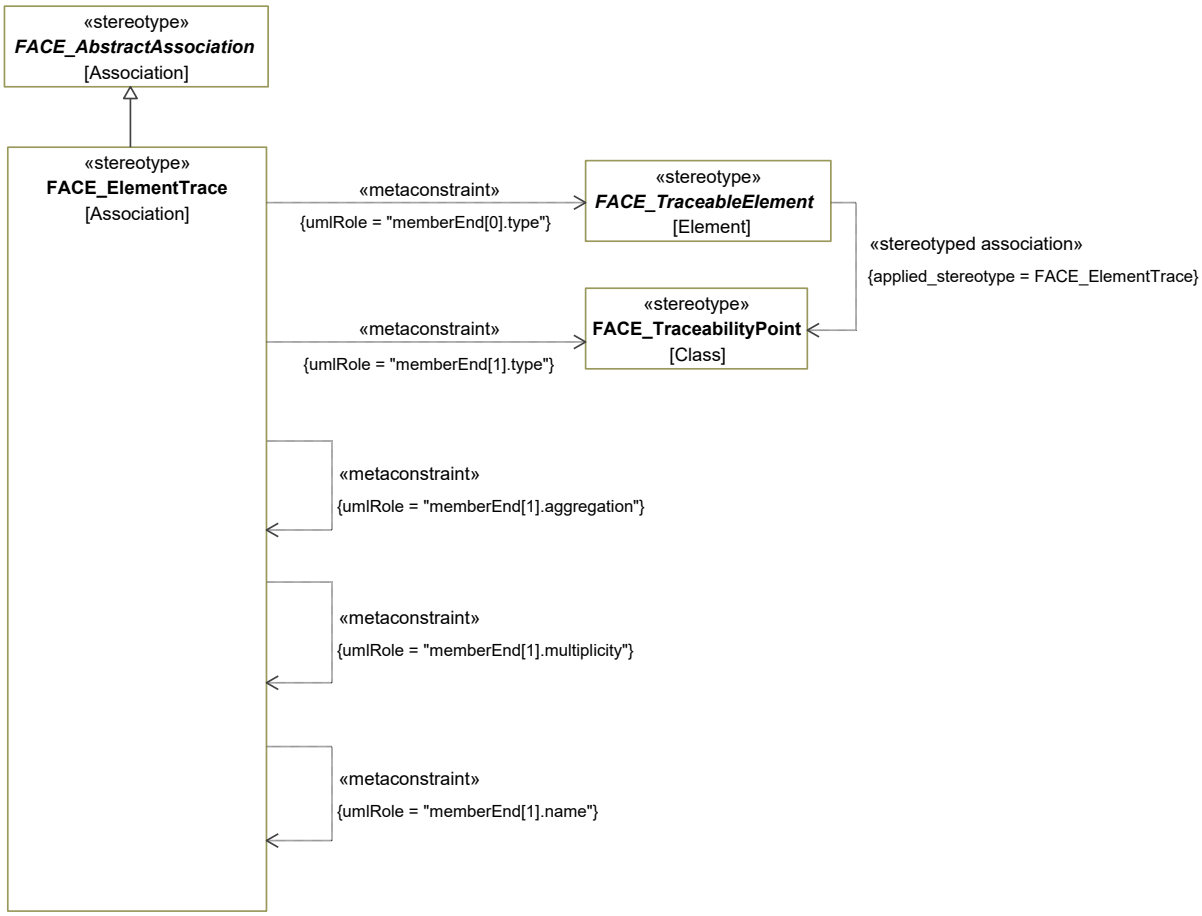

Figure 7-7: FACE_ElementTrace

Constraints

[1] FACE_ElementTrace.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by a specialization of «FACE_TraceableElement».
[2] FACE_ElementTrace.memberEnd[1].aggregation	composite
[3] FACE_ElementTrace.memberEnd[1].multiplicity	0..*
[4] FACE_ElementTrace.memberEnd[1].name	"traceabilityPoint"
[5] FACE_ElementTrace.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_TraceabilityPoint».

FACE_EndPoint

Package: FACE Data Architecture
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to associate the FACE Components (FACE_UnitOfPortability, FACE_AbstractUoP) with FACE_Connections. In addition to aggregation and multiplicity specifications on memberEnd[1], this association differs from the default FACE_AbstractAssociation in that it is bi-directionally navigable.

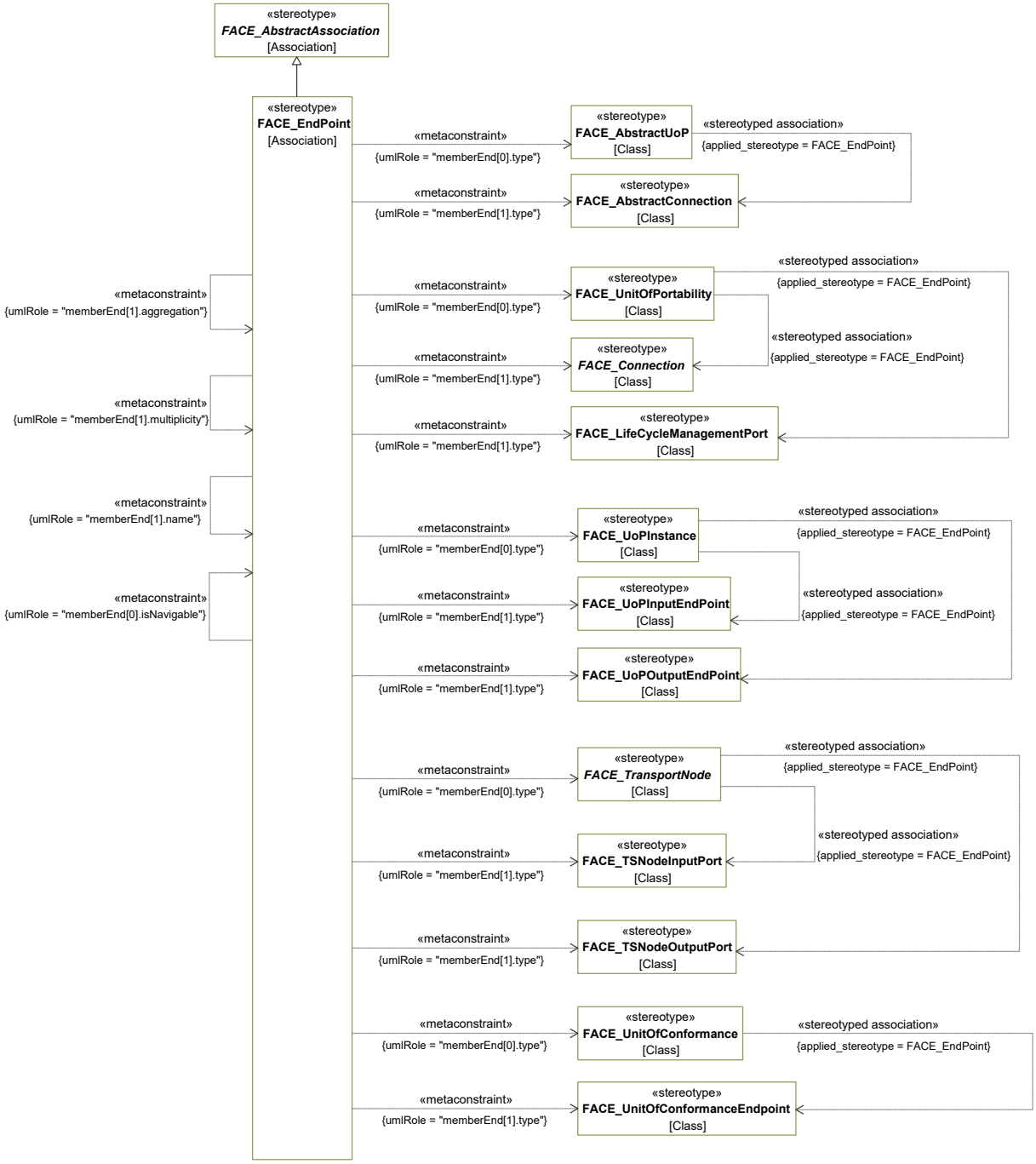

Figure 7-8: FACE_EndPoint

Constraints

[1] FACE_EndPoint.memberEnd[0].isNavigable	true
[2] FACE_EndPoint.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by one of the following: «FACE_UnitofPortability» «FACE_AbstractUoP» «FACE_UoPInstance» A specialization of «FACE_TransportNode» «FACE_UnitOfConformance»
[3] FACE_EndPoint.memberEnd[1].aggregation	composite
[4] FACE_EndPoint.memberEnd[1].multiplicity	MemberEnd[1].multiplicity depends on the stereotypes of the values connected by the association:

memberEnd[0].type	memberEnd[1].type	memberEnd[1].multiplicity
«FACE_AbstractUoP»	«FACE_AbstractConnection»	0..*
«FACE_UnitOfPortability»	specialization of «FACE_Connection»	1..*
«FACE_UnitOfPortability»	«FACE_LifeCycleManagementPort»	0..2
specialization of «FACE_TransportNode»	«FACE_TSNodeInputPort»	0..*
specialization of «FACE_TransportNode»	«FACE_TSNodeOutputPort»	0..1
«FACE_UoPInstance»	«FACE_UoPInputEndPoint»	0..*
«FACE_UnitOfConformance»	«FACE_UnitOfConformanceEndpoint»	0..*

[5] FACE_EndPoint.memberEnd[1].name	MemberEnd[1].name depends on the stereotypes of the values connected by the association:

memberEnd[0].type	memberEnd[1].type	memberEnd[1].name
«FACE_AbstractUoP»	«FACE_AbstractConnection»	connection
«FACE_UnitOfPortability”	specialization of «FACE_Connection”	connection
«FACE_UnitOfPortability”	«FACE_LifeCycleManagementPort”	lcmPort
specialization of «FACE_TransportNode”	«FACE_TSNodeInputPort”	inPort
specialization of «FACE_TransportNode”	«FACE_TSNodeOutputPort”	outPort

«FACE_UoPInstance”	«FACE_UoPInputEndPoint ”	input
«FACE_UoPInstance”	«FACE_UoPOutputEndPoi nt”	output

[6] FACE_EndPoint.memberEnd[1].ty pe	Based on the EndPoint.memberEnd[0].type's stereotype: = «FACE_UnitOfPortability», the memberEnd[1].type metaproperty must be stereotyped by one of the following: A specialization of «FACE_Connection» «FACE_LifeCycleManagementPort» = «FACE_AbstractUoP», the memberEnd[1].type metaproperty must be stereotyped by «FACE_AbstractConnection» = «FACE_UoPInstance», the memberEnd[1].type metaproperty must be stereotyped by one of the following: «FACE_UoPInputEndPoint» «FACE_UoPOutputEndPoint» = A specialization of «FACE_TransportNode», the memberEnd[1].type metaproperty must be stereotyped by one of the following: «FACE_TSNodeInputPort» «FACE_TSNodeOutputPort» = «FACE_UnitOfConformance», the memberEnd[1].type metaproperty must be stereotyped by «FACE_UnitOfConformanceEndpoint»

FACE_IntegrationModel

Package: FACE Data Architecture
isAbstract: No
Generalization: FACE_Element
Extension: Package

Description

A FACE_IntegrationModel is a container for FACE_IntegrationElements.

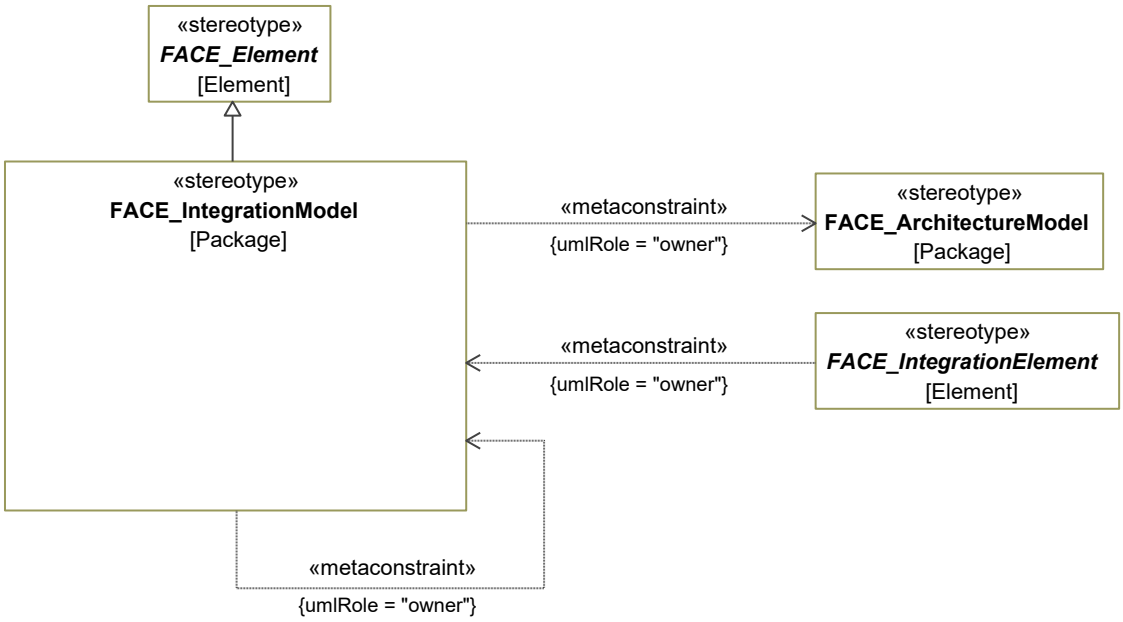

Figure 7-9: FACE_IntegrationModel

Constraints

[1] FACE_IntegrationModel.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_ArchitectureModel» «FACE_IntegrationModel»

FACE_MessageType

Package: FACE Data Architecture
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to identify the FACE_PlatformView that specifies the data to be exchanged through a FACE_Endpoint.

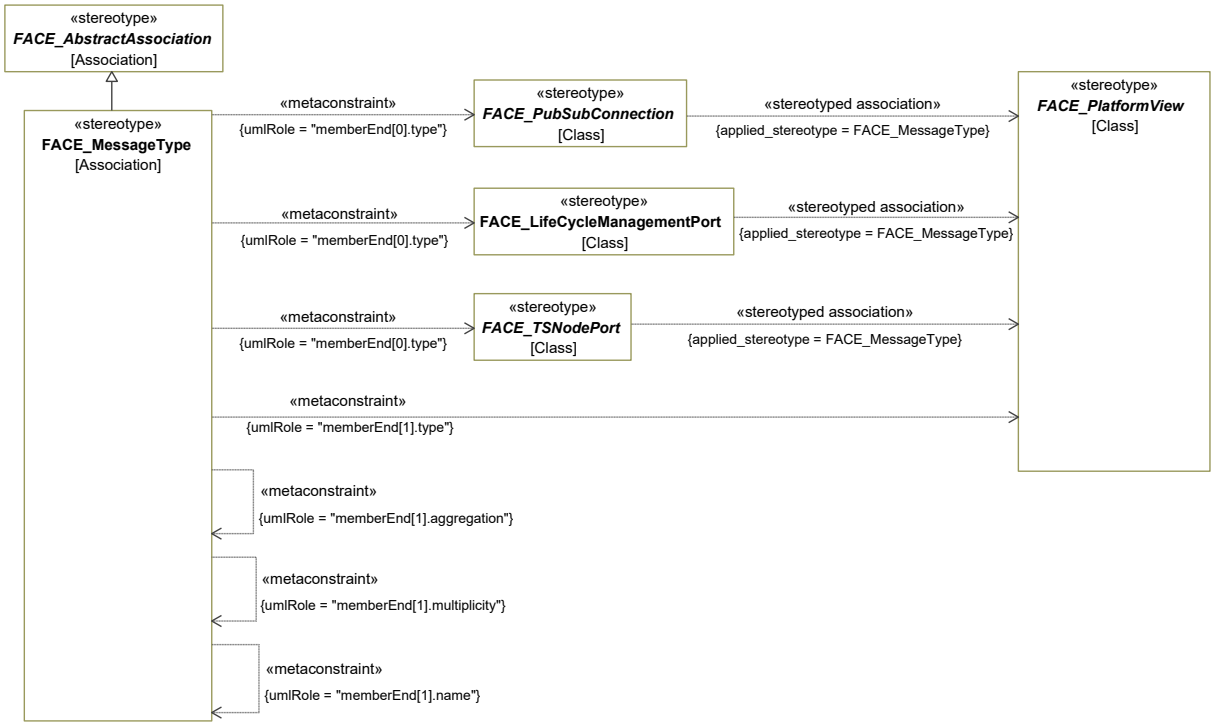

Figure 7-10: FACE_MessageType

Constraints

[1] FACE_MessageType.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by one of the following: Specialization of «FACE_PubSubConnection» «FACE_LifeCycleManagementPort» Specialization of «FACE_TSNodePort»
[2] FACE_MessageType.memberEnd[1].aggregation	none
[3] FACE_MessageType.memberEnd[1].multiplicity	1
[4] FACE_MessageType.memberEnd[1].name	Based on the stereotype of the memberEnd[0].type metaproperty: = Specialization of «FACE_PubSubConnection», memberEnd[1].name is "messageType" = «FACE_LifeCycleManagementPort», memberEnd[1].name is "lcmMessageType" = Specialization of «FACE_TSNodePort», memberEnd[1].name is "view"
[5] FACE_MessageType.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_PlatformView».

FACE_Realize

Package: FACE Data Architecture
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to indicate a FACE element realization of another FACE element.

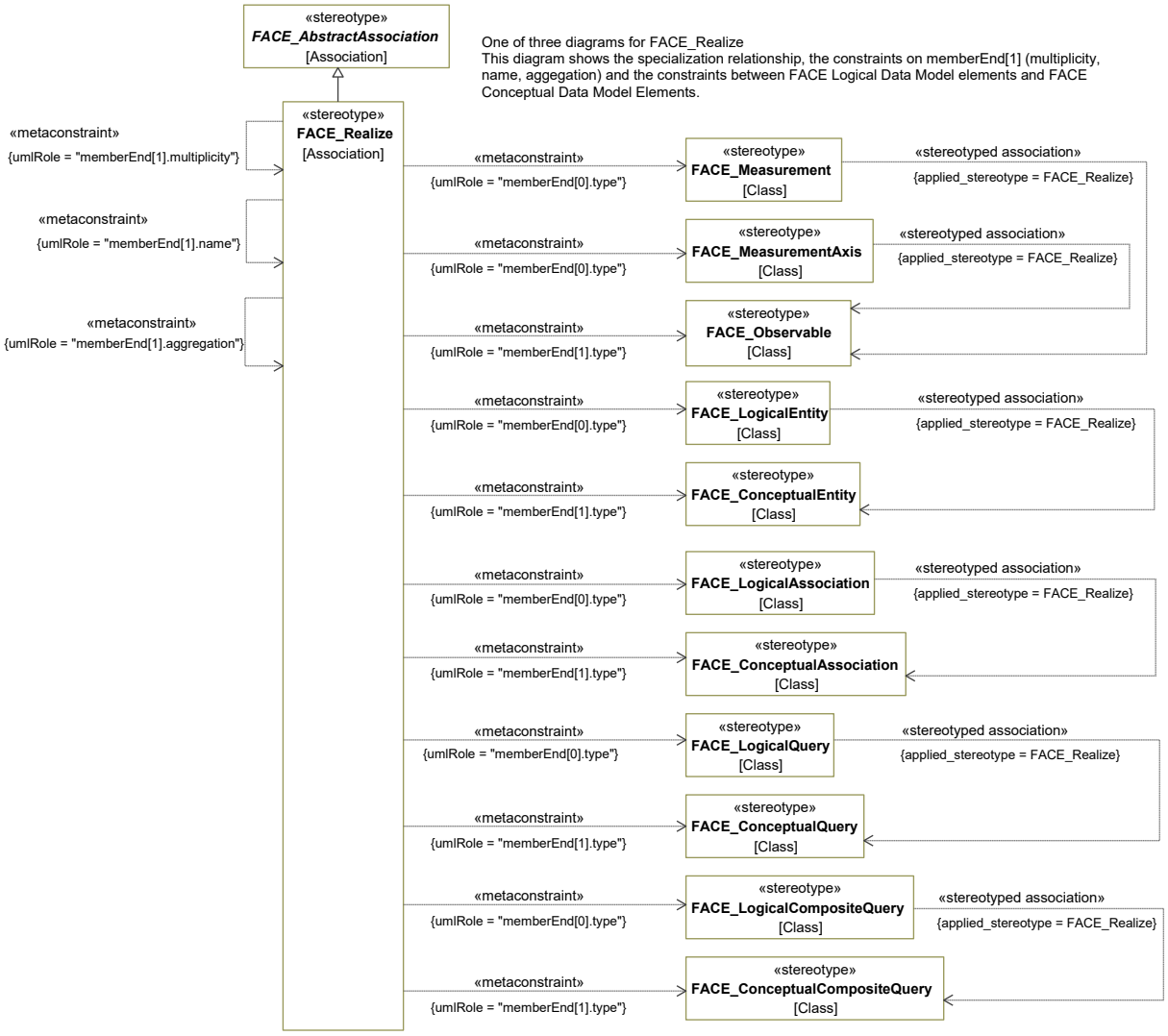

Figure 7-11: FACE_Realize , diagram 1 of 3

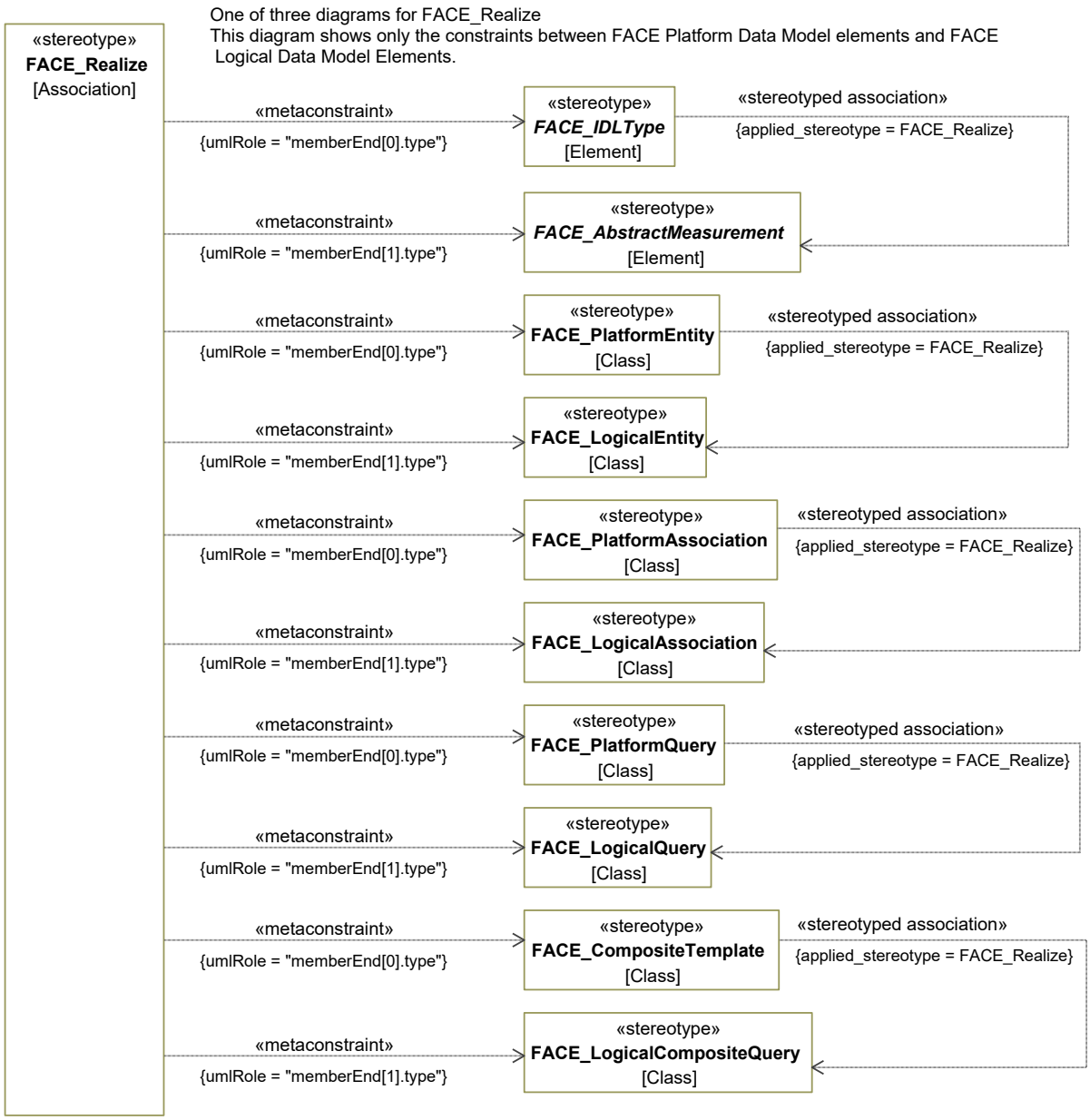

Figure 7-12: FACE_Realize , diagram 2 of 3

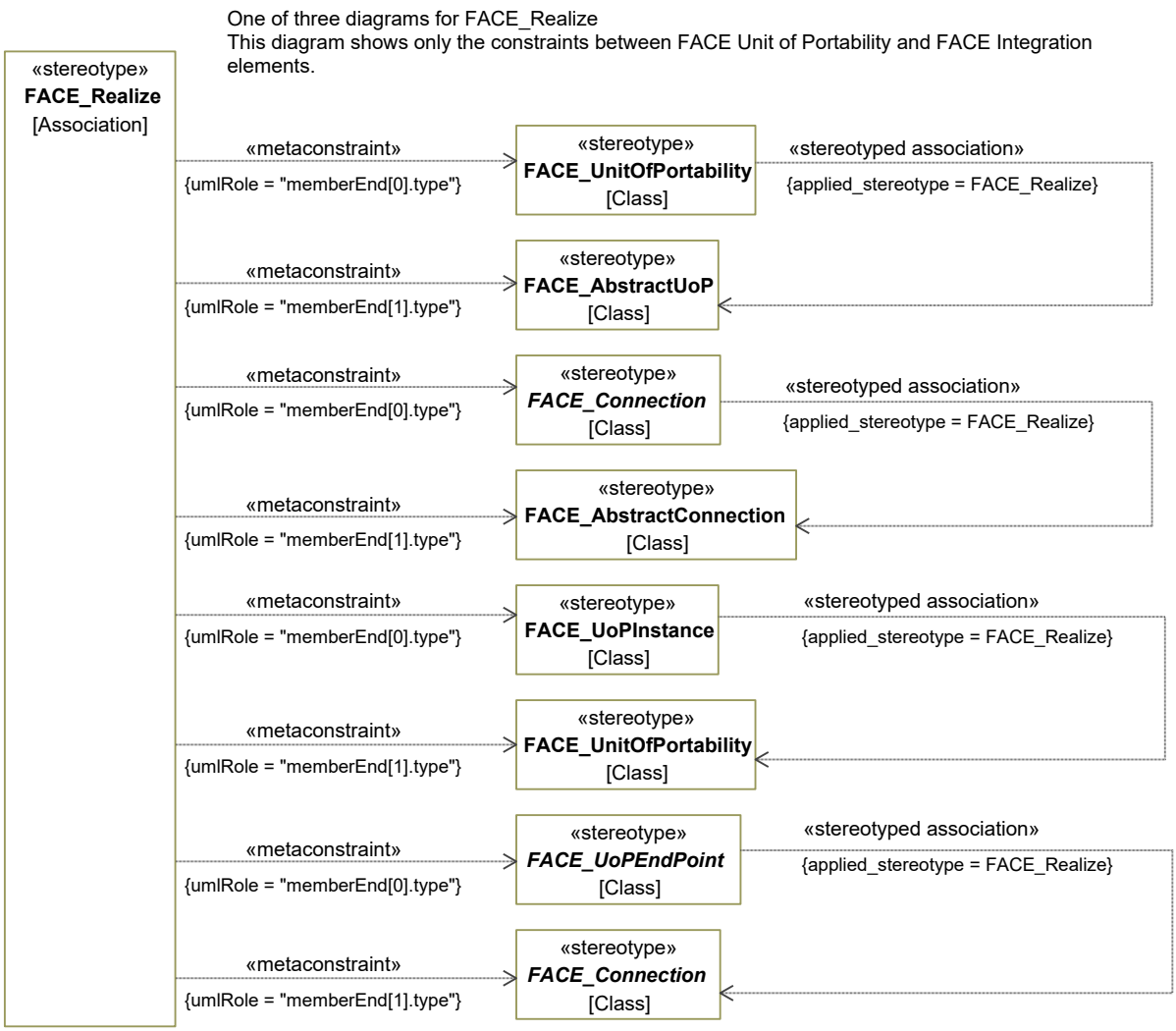

Figure 7-13: FACE_Realize, , diagram 3 of 3

Constraints

[1] FACE_Realize.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by a one of the following stereotypes: «FACE_Measurement» «FACE_MeasurementAxis» «FACE_LogicalEntity» «FACE_LogicalAssociation» «FACE_LogicalQuery» «FACE_LogicalCompositeQuery» Specializations of «FACE_IDLType» «FACE_PlatformAssociation» «FACE_PlatformEntity» «FACE_PlatformQuery» «FACE_CompositeTemplate» «FACE_UnitOfPortability» Specializations of «FACE_Connection»

	«FACE_UoPInstance» Specializations of «FACE_UoPEndPoint»
[2] FACE_Realize.memberEnd[1].aggregation	none
[3] FACE_Realize.memberEnd[1].multiplicity	Based on the stereotype of the memberEnd[0].type metaproperty: = «FACE_CompositeTemplate», «FACE_PlatformQuery», «FACE_UnitOfPortability», or specialization of «FACE_Connection», memberEnd[1].multiplicity is 0..1 = specialization of «FACE_IDLType», «FACE_LogicalAssociation», «FACE_LogicalCompositeQuery», «FACE_LogicalComposition», «FACE_LogicalEntity», «FACE_LogicalQuery», «FACE_Measurement», «FACE_MeasurementAxis», «FACE_PlatformAssociation», «FACE_PlatformEntity», specialization of «FACE_UoPEndPoint», or «FACE_UoPInstance», memberEnd[1].multiplicity is 1
[4] FACE_Realize.memberEnd[1].name	"realize"
[5] FACE_Realize.memberEnd[1].type	Based on the Realize.memberEnd[0].type value's stereotype: = «FACE_Measurement», the memberEnd[1].type metaproperty must be stereotyped by «FACE_Observable» = «FACE_MeasurementAxis», the memberEnd[1].type metaproperty must be stereotyped by «FACE_Observable» = «FACE_LogicalEntity», the memberEnd[1].type metaproperty must be stereotyped by «FACE_ConceptualEntity» = «FACE_LogicalAssociation», the memberEnd[1].type metaproperty must be stereotyped by «FACE_ConceptualAssociation» = «FACE_LogicalQuery», the memberEnd[1].type metaproperty must be stereotyped by «FACE_ConceptualQuery» = «FACE_LogicalCompositeQuery», the memberEnd[1].type metaproperty must be stereotyped by «FACE_ConceptualCompositeQuery» = A specialization of «FACE_IDLType», the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_AbstractMeasurement» = «FACE_PlatformAssociation», the memberEnd[1].type metaproperty must be stereotyped by «FACE_LogicalAssociation» = «FACE_PlatformEntity», the memberEnd[1].type metaproperty must be stereotyped by «FACE_LogicalEntity» = «FACE_PlatformQuery», the memberEnd[1].type metaproperty must be stereotyped by «FACE_LogicalQuery» = «FACE_CompositeTemplate», the memberEnd[1].type metaproperty must be stereotyped by «FACE_LogicalCompositeQuery» = «FACE_UnitofPortability», the memberEnd[1].type metaproperty must be stereotyped by «FACE_AbstractUoP» = A specialization of «FACE_Connection», the memberEnd[1].type metaproperty must be stereotyped by «FACE_AbstractConnection» = «FACE_UoPInstance», the memberEnd[1].type metaproperty must be stereotyped by «FACE_UnitOfPortability» = A specialization of «FACE_UoPEndPoint», the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_Connection»

FACE_TraceabilityModel

Package: FACE Data Architecture
isAbstract: No
Generalization: FACE_Element
Extension: Package

Description

A FACE_TraceabilityModel is a container for FACE_TraceabilityElements.

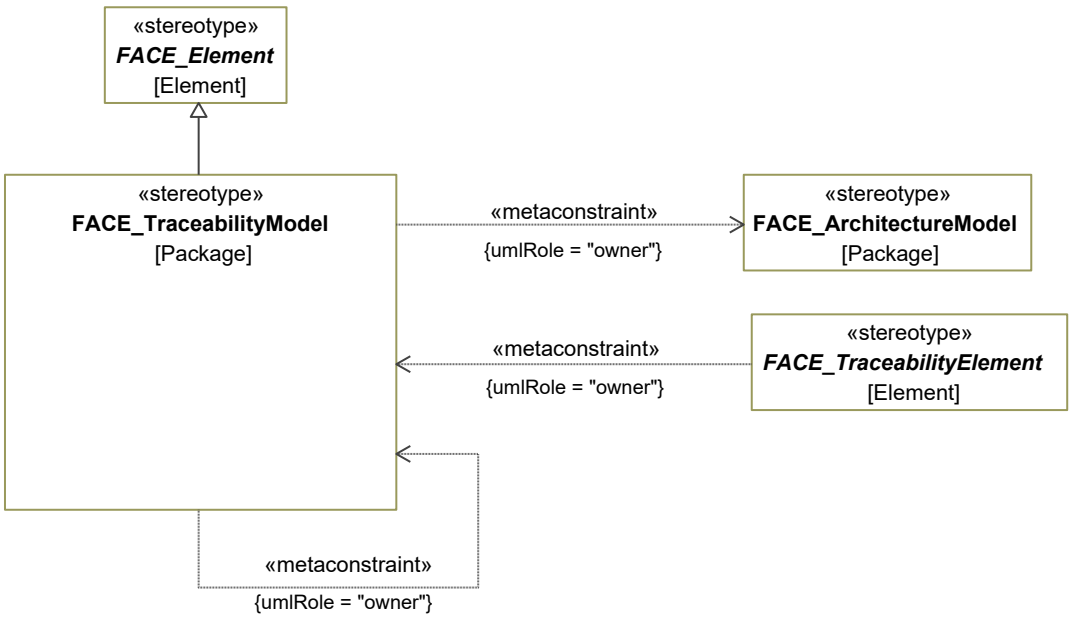

Figure 7-14: FACE_TraceabilityModel

Constraints

[1] FACE_TraceabilityModel.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_ArchitectureModel» «FACE_TraceabilityModel»

FACE_UoPModel

Package: FACE Data Architecture
isAbstract: No
Generalization: FACE_Element
Extension: Package

Description

A FACE_UoPModel is a container for FACE_UoPElements.

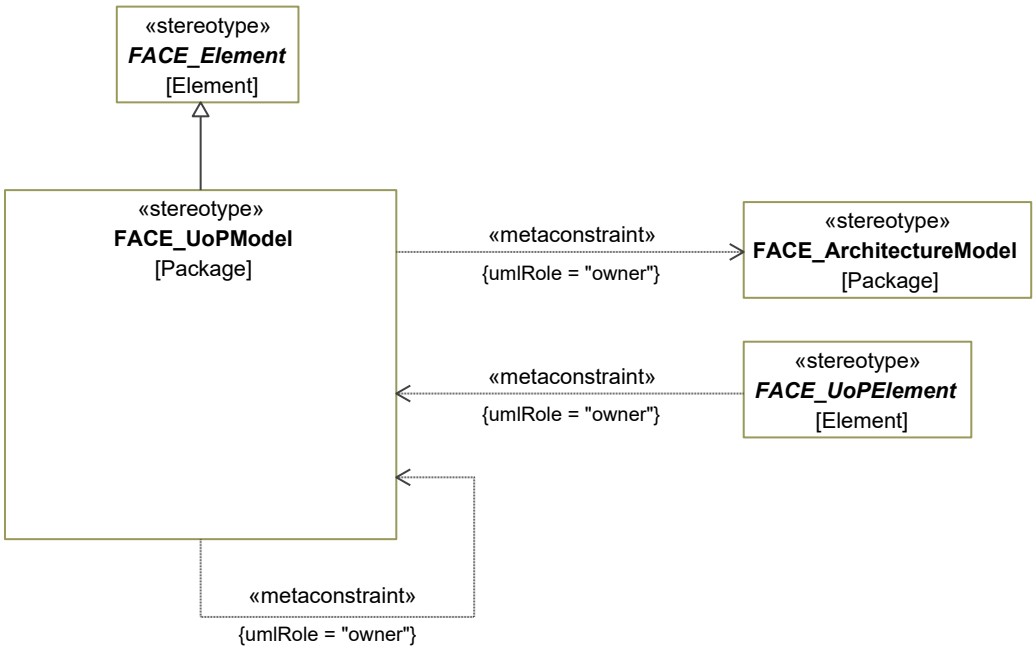

Figure 7-15: FACE_UoPModel

Constraints

[1] FACE_UoPModel.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_ArchitectureModel» «FACE_UoPModel»

7.1.1.1 FACE_UAF_Profile::FACE Data Architecture::FACE Data Model

FACE_AssociatedParticipant

Package: FACE Data Model
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to identify the FACE participant entities in FACE element associations.

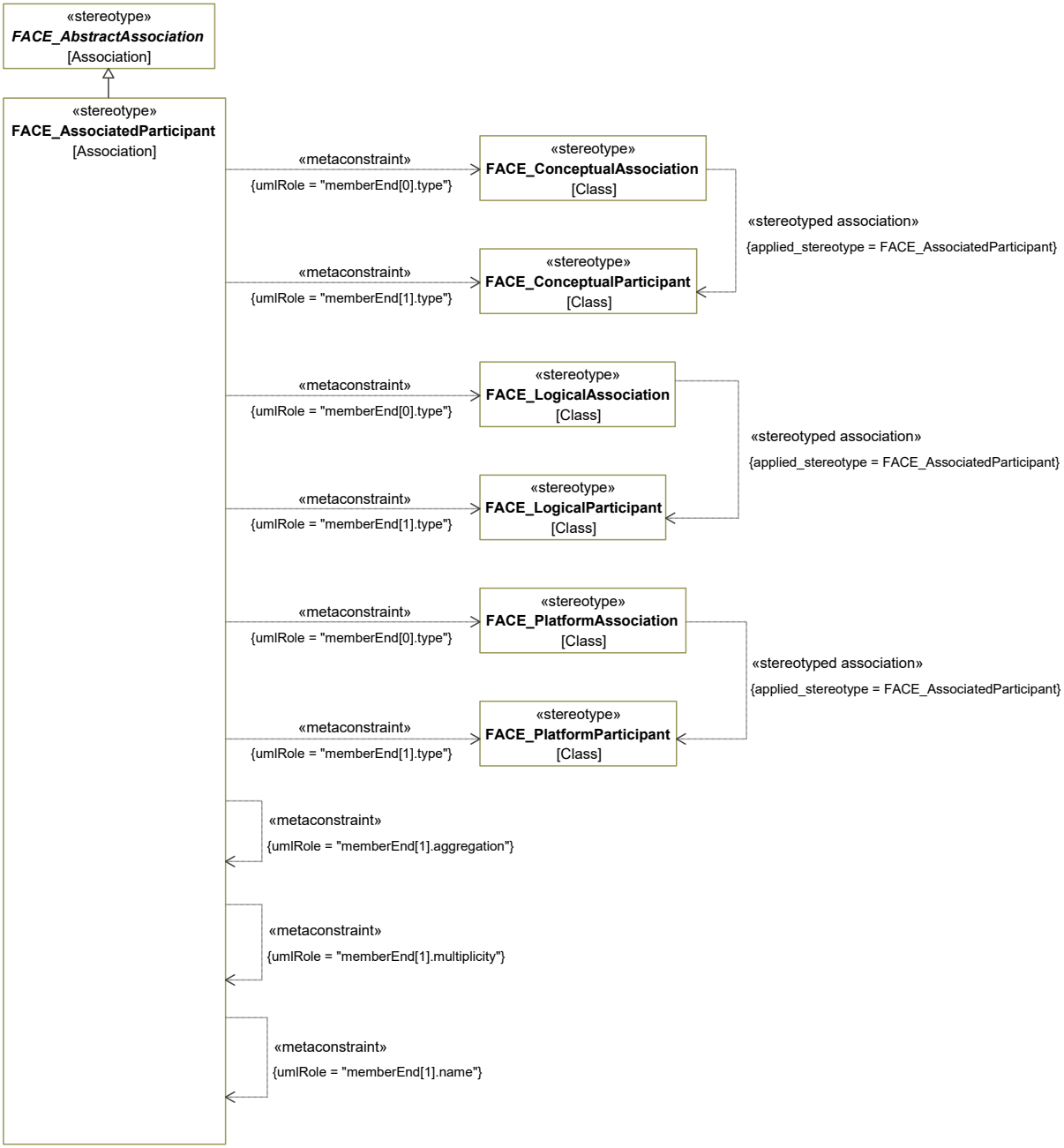

Figure 7-16: FACE_AssociatedParticipant

Constraints

[1] FACE_AssociatedParticipant.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by one of the following: «FACE_ConceptualAssociation» «FACE_LogicalAssociation» «FACE_PlatformAssociation»
[2] FACE_AssociatedParticipant.memberEnd[1].aggregation	composite

[3] FACE_AssociatedParticipant.memberEnd[1].multiplicity	0..*
[4] FACE_AssociatedParticipant.memberEnd[1].name	"participant"
[5] FACE_AssociatedParticipant.memberEnd[1].type	Based on the FACE_AssociatedParticipant.memberEnd[0].type value's stereotype: = «FACE_ConceptualAssociation», the memberEnd[1].type metaproperty must be stereotyped by «FACE_ConceptualParticipant» = «FACE_LogicalAssociation», thememberEnd[1].type metaproperty must be stereotyped by «FACE_LogicalParticipant» = «FACE_PlatformAssociation», thememberEnd[1].type metaproperty must be stereotyped by «FACE_PlatformParticipant»

FACE_ConceptualDataModel

Package: FACE Data Model
isAbstract: No
Generalization: FACE_DataModelElement
Extension: Package

Description

A FACE_ConceptualDataModel is a container for FACE_ConceptualElements.

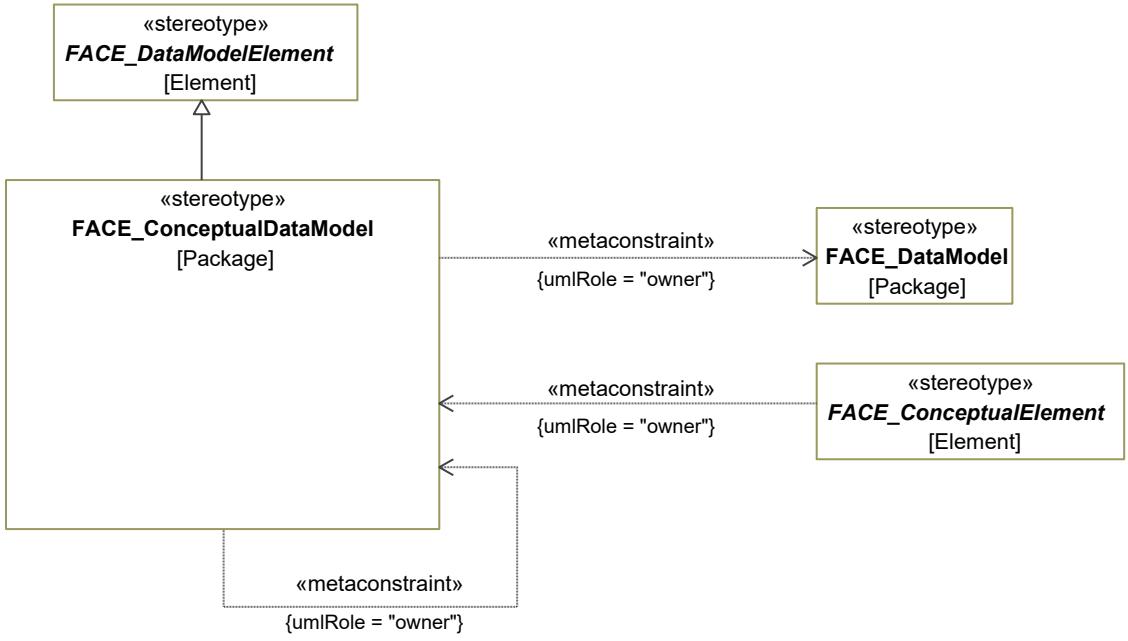

Figure 7-17: FACE_ConceptualDataModel

Constraints

[1] FACE_ConceptualDataModel.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_DataModel» «FACE_ConceptualDataModel»

FACE_LogicalDataModel

Package: FACE Data Model
isAbstract: No
Generalization: FACE_DataModelElement
Extension: Package

Description

A FACE_LogicalDataModel is a container for FACE_LogicalElements (Logical Data Model elements).

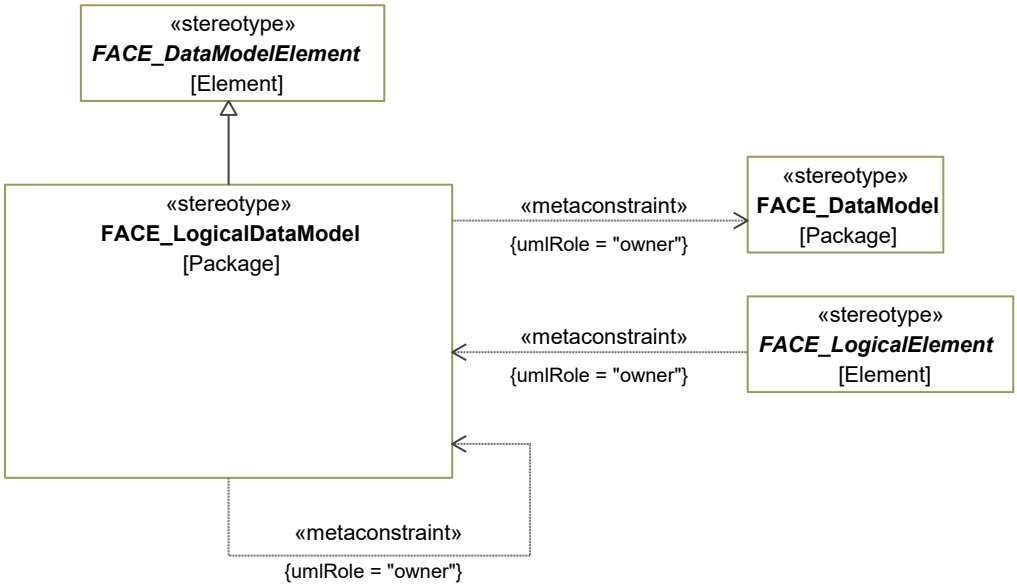

Figure 7-18: FACE_LogicalDataModel

Constraints

[1] FACE_LogicalDataModel.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_DataModel» «FACE_LogicalDataModel»

FACE_PlatformDataModel

Package: FACE Data Model
isAbstract: No
Generalization: FACE_DataModelElement

Extension: Package

Description

A FACE_PlatformDataModel is a container for FACE_PlatformElements (platform Data Model Elements).

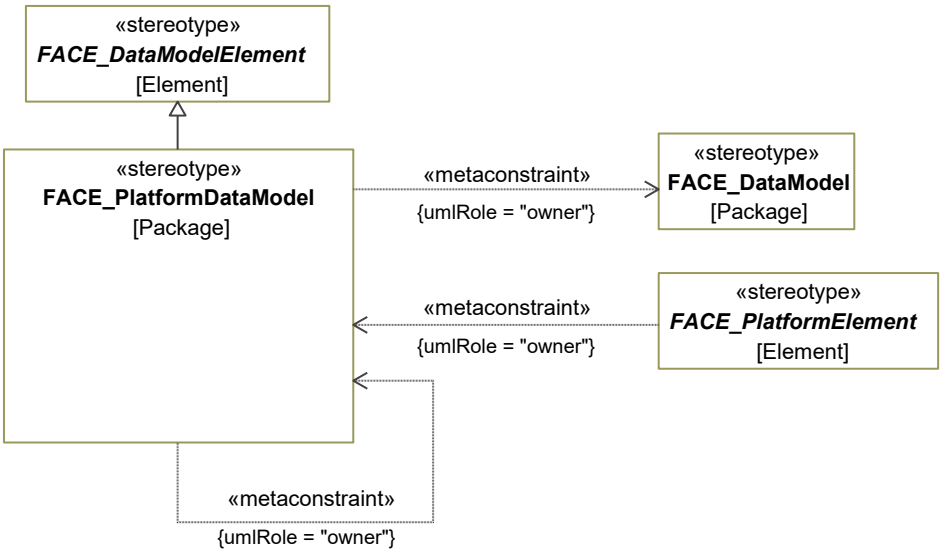

Figure 7-19: FACE_PlatformDataModel

Constraints

[1] FACE_PlatformDataModel.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_DataModel» «FACE_PlatformDataModel»

FACE_SpecializationOwner

Package: FACE Data Model
isAbstract: Yes
Extension: Class

Description

Abstract type to group all FACE stereotypes that can own a «Specialize» generalization. Enables application of constraints uniformly within specialized elements.

This stereotype exists only for specification of constraints that apply to the specialized FACE Profile stereotypes. It is optional in the implementation of this specification.

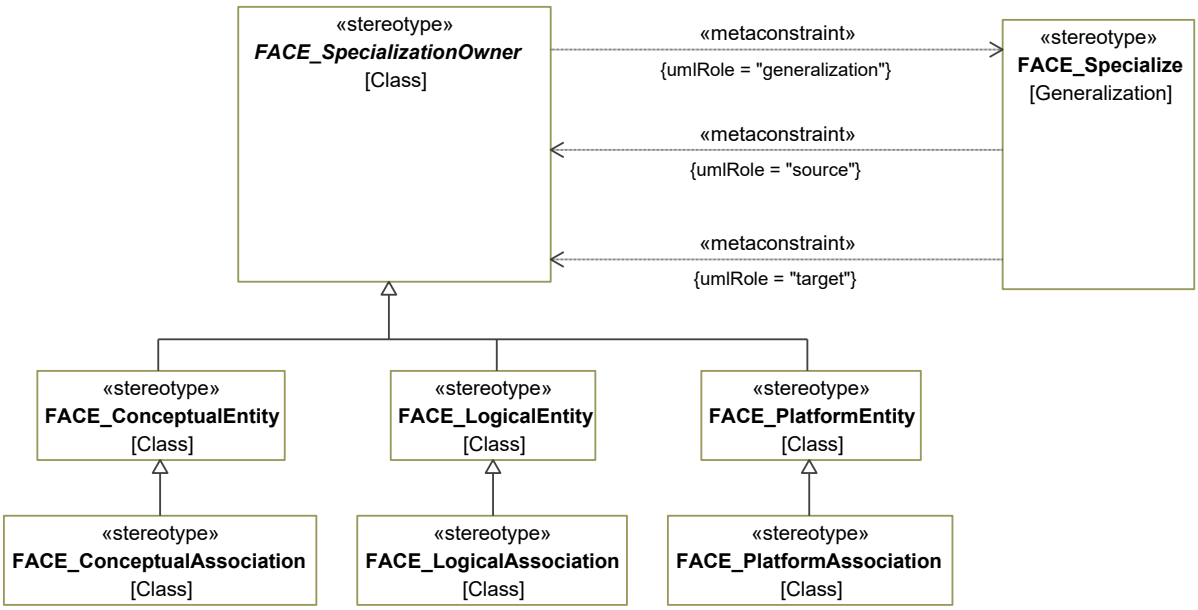

Figure 7-20: abstract FACE_SpecializationOwner

Constraints

[1] FACE_SpecializationOwner.generalization	The generalization collection may contain no more than one «FACE_Specialize» generalization.

FACE_Specialize

Package: FACE Data Model
isAbstract: No
Extension: Generalization

Description

Used to indicate a FACE element Specialization of another FACE element.

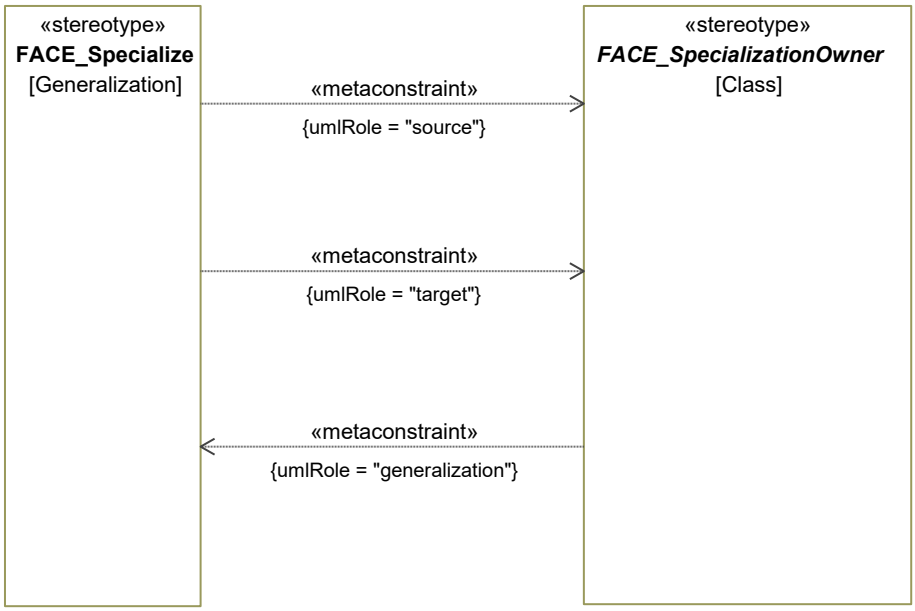

Figure 7-21: FACE_Specialize

Constraints

[1] FACE_Specialize.source	The value for the source metaproperty must be stereotyped by a specialization of «FACE_SpecializationOwner».
[2] FACE_Specialize.target	Based on the Specialize.source value's stereotype: = «FACE_ConceptualEntity», the target metaproperty must be stereotyped by «FACE_ConceptualEntity» = «FACE_ConceptualAssociation», the target metaproperty must be stereotyped by one of the following: «FACE_ConceptualEntity» «FACE_ConceptualAssociation» = «FACE_LogicalEntity», the target metaproperty must be stereotyped by «FACE_LogicalEntity» = «FACE_LogicalAssociation», the target metaproperty must be stereotyped by one of the following: «FACE_LogicalEntity» «FACE_LogicalAssociation» = «FACE_PlatformEntity», the target metaproperty must be stereotyped by «FACE_PlatformEntity» = «FACE_PlatformAssociation», the target metaproperty must be stereotyped by one of the following: «FACE_PlatformEntity» «FACE_PlatformAssociation»

7.1.1.1.1 FACE_UAF_Profile::FACE Data Architecture::FACE Data Model::ConceptualDataModel

FACE_BasisElement

Package: ConceptualDataModel
isAbstract: Yes

Generalization: FACE_ConceptualComposableElement

Description

A conceptual FACE_BasisElement is a conceptual data type that is independent of any specific data representation.

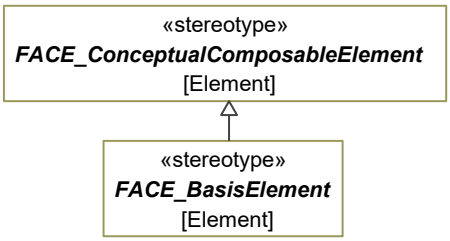

Figure 7-22: abstract FACE_BasisElement

FACE_BasisEntity

Package: ConceptualDataModel
isAbstract: No
Generalization: FACE_ConceptualElement
Extension: Class

Description

A FACE_BasisEntity represents a unique domain concept and establishes a basis from which FACE_ConceptualEntities can be specialized.

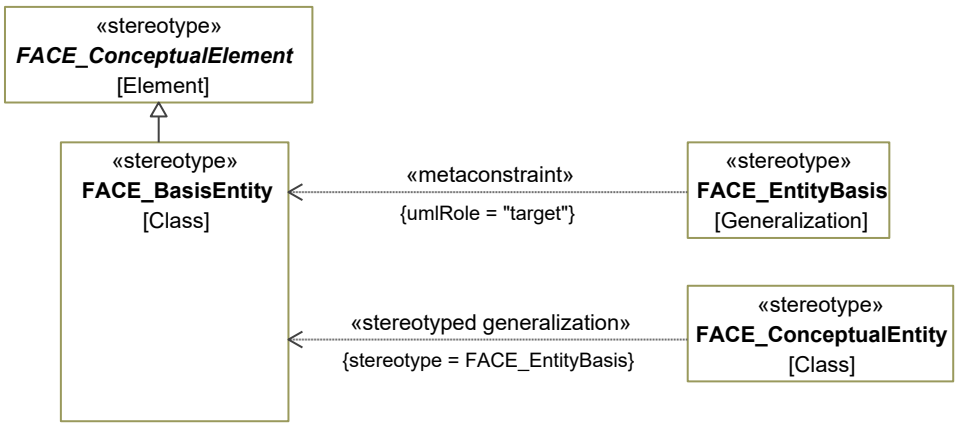

Figure 7-23: FACE_BasisEntity

FACE_ConceptualAssociation

Package: ConceptualDataModel
isAbstract: No
Generalization: FACE_ConceptualEntity

Description

A FACE_ConceptualAssociation represents a relationship between two or more FACE_ConceptualEntities. In addition, there may be one or more conceptual Composable Elements that characterize the relationship. FACE_ConceptualAssociations are FACE_ConceptualEntities that may also participate in other FACE_ConceptualAssociations.

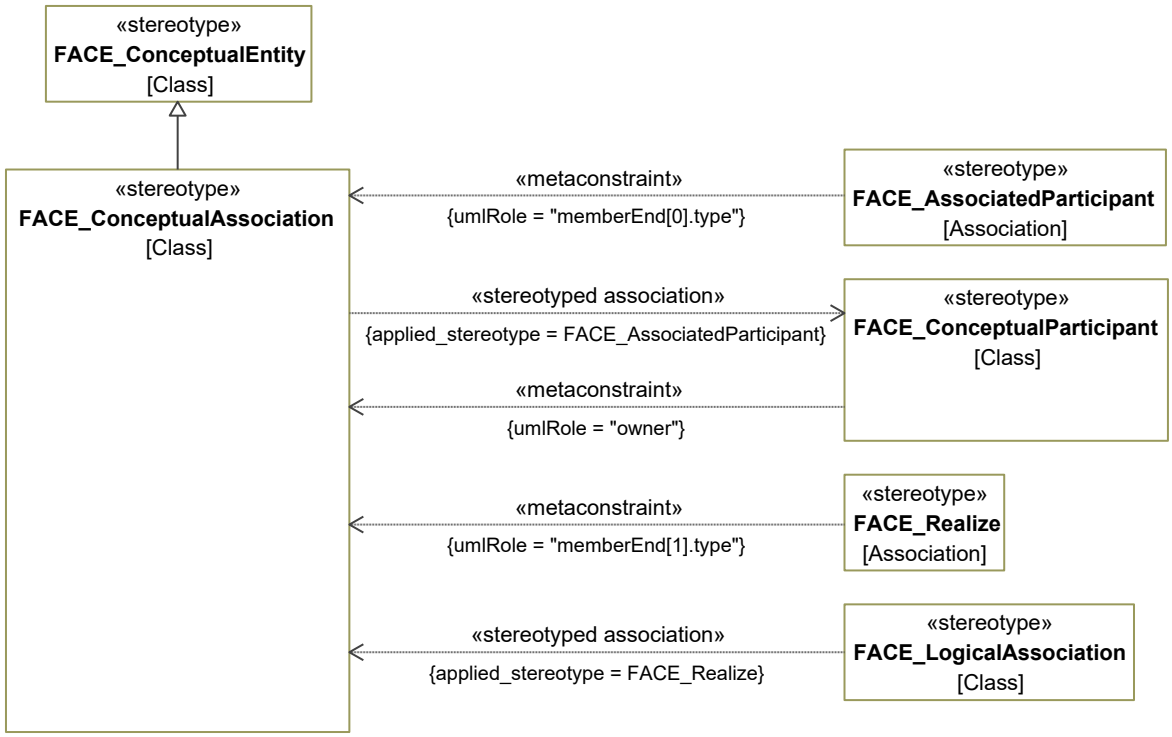

Figure 7-24: FACE_ConceptualAssociation

FACE Conformance/OCL Constraints

[1] FACE_ConceptualAssociation.hasAtLeastTwoParticipants	A FACE_Association must have at least two Participants.

FACE_ConceptualCharacteristic

Package: ConceptualDataModel
isAbstract: Yes
Generalization: FACE_ModelElement

Description

A FACE_ConceptualCharacteristic is a defining feature of a FACE_ConceptualEntity. The rolename attribute defines the name of the FACE_ConceptualCharacteristic within the scope of the FACE_ConceptualEntity. The lowerBound and upperBound attributes define the multiplicity of the composed Characteristic. An upperBound multiplicity of -1 represents an unbounded sequence.

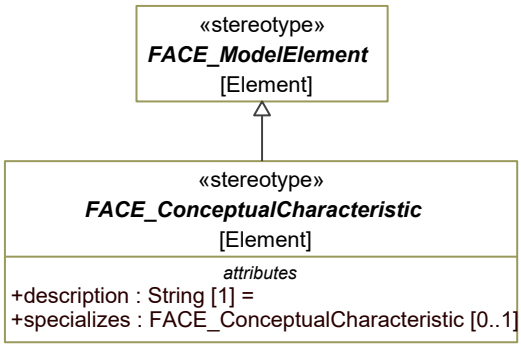

Figure 7-25: abstract FACE_ConceptualCharacteristic

Attributes

description : String [1]

specializes : FACE_ConceptualCharacteristic [0..1]

FACE Conformance/OCL Constraints

[1] FACE_ConceptualCharacteristic.lowerBoundValid	A FACE_ConceptualCharacteristic's lowerBound must be greater than or equal to zero.
[2] FACE_ConceptualCharacteristic.lowerBound_LTE_UpperBound	A FACE_ConceptualCharacteristic's lowerBound must be less than or equal to its upperBound, unless its upperBound is -1.
[3] FACE_ConceptualCharacteristic.rolenameIsValidIdentifier	The rolename of a FACE_ConceptualCharacteristic must be a valid identifier.
[4] FACE_ConceptualCharacteristic.specializeCharacteristicOnce	A FACE_ConceptualCharacteristic must be specialized once in a generalization hierarchy.
[5] FACE_ConceptualCharacteristic.upperBoundValid	A FACE_ConceptualCharacteristic's upperBound must be equal to -1 or greater than 1.

FACE_ConceptualComposableElement

Package: ConceptualDataModel
isAbstract: Yes
Generalization: FACE_ConceptualElement

Description

A FACE_ConceptualComposableElement is a FACE_ConceptualElement that is allowed to participate in a Composition relationship. In other words, these are the conceptual Elements that may be a characteristic of a FACE_ConceptualEntity.

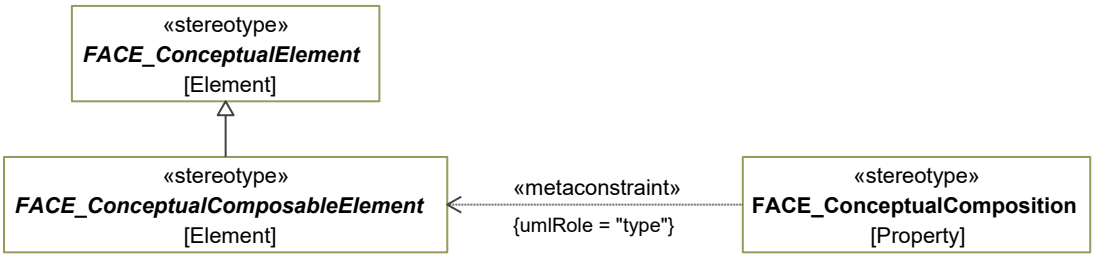

Figure 7-26: abstract FACE_ConceptualComposableElement

FACE_ConceptualCompositeQuery

Package: ConceptualDataModel
isAbstract: No
Generalization: FACE_ConceptualView
Extension: Class

Description

A FACE_ConceptualCompositeQuery is a collection of two or more FACE_ConceptualQueries. The isUnion attribute specifies whether the composed FACE_ConceptualQueries are intended to be represented as cases in a FACE_IDL union or as members of a FACE_IDL struct.

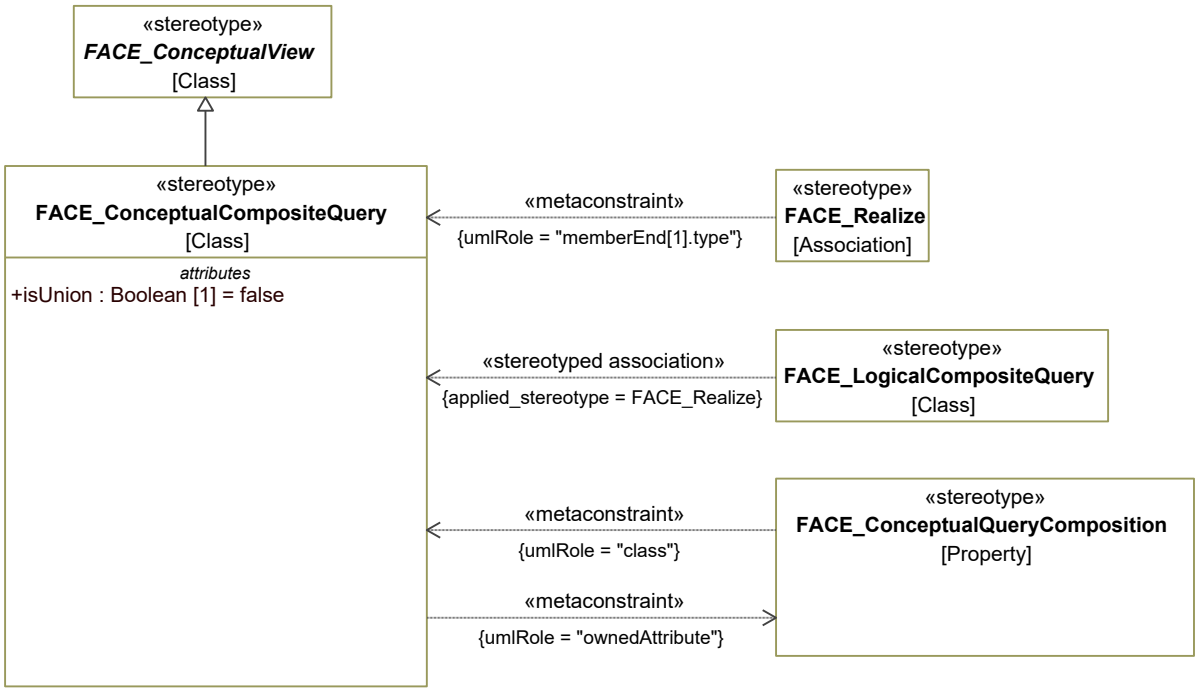

Figure 7-27: FACE_ConceptualCompositeQuery

Attributes

isUnion : Boolean [1]

Constraints

[1] FACE_ConceptualCompositeQuery.ownedAttribute	The values for the ownedAttribute metaproperty must meet the following criteria: - must be ordered list - must be stereotyped «FACE_ConceptualCompositeQuery» or its specializations - must contain 2 or more elements

FACE Conformance/OCL Constraints

[1] FACE_ConceptualCompositeQuery.compositionsHaveUniqueRolenames	A FACE_ConceptualQueryComposition's rolename must be unique within a FACE_ConceptualCompositeQuery.
[2] FACE_ConceptualCompositeQuery.noCyclesInConstruction	A FACE_ConceptualCompositeQuery may not compose itself.
[3] FACE_ConceptualCompositeQuery.viewComposedOnce	A FACE_ConceptualCompositeQuery may not compose the same FACE_ConceptualView more than once.

FACE_ConceptualComposition

Package: ConceptualDataModel
isAbstract: No
Generalization: FACE_ConceptualCharacteristic
Extension: Property

Description

A FACE_ConceptualComposition is the mechanism that allows FACE_ConceptualEntity to be constructed from other FACE_ConceptualComposableElements. The type of a FACE_ConceptualComposition is the FACE_ConceptualComposableElement being used to construct the FACE_ConceptualEntity.

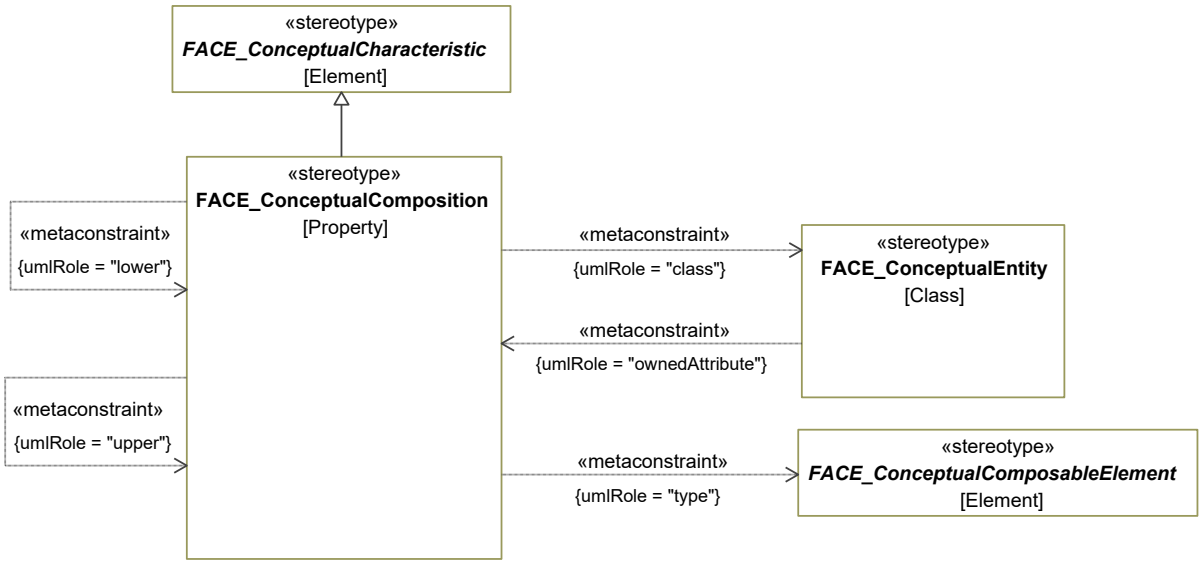

Figure 7-28: FACE_ConceptualComposition

Constraints

[1] FACE_ConceptualComposition.class	Value for the class metaproperty must be stereotyped «FACE_ConceptualEntity» or its specializations.
[2] FACE_ConceptualComposition.lower	The value for the lower (lower bound of multiplicity) metaproperty must be an integer greater than or equal to -1.
[3] FACE_ConceptualComposition.type	Value for the type metaproperty must be stereotyped «FACE_ConceptualComposableElement» or its specializations.
[4] FACE_ConceptualComposition.upper	The value for the upper (upper bound of multiplicity) metaproperty must be an integer greater than or equal to -1

FACE Conformance/OCL Constraints

[1] FACE_ConceptualComposition.multiplicityConsistentWithSpecialization	If a FACE_ConceptualComposition specializes, its multiplicity must be at least as restrictive as the FACE_ConceptualComposition it specializes.
[2] FACE_ConceptualComposition.specializationDistinct	If a FACE_ConceptualComposition specializes, its type or multiplicity must be different from the FACE_ConceptualComposition it specializes.
[3] FACE_ConceptualComposition.typeConsistentWithSpecialization	If a FACE_ConceptualComposition specializes, it specializes a FACE_ConceptualComposition. If FACE_ConceptualComposition "A" specializes

	FACE_ConceptualComposition "B", then A's type must be B's type or a specialization of B's type.

FACE_ConceptualElement

Package: ConceptualDataModel
isAbstract: Yes
Generalization: FACE_DataModelElement

Description

A FACE_ConceptualElement is the root type for defining the conceptual elements of the FACE Data Model Language.

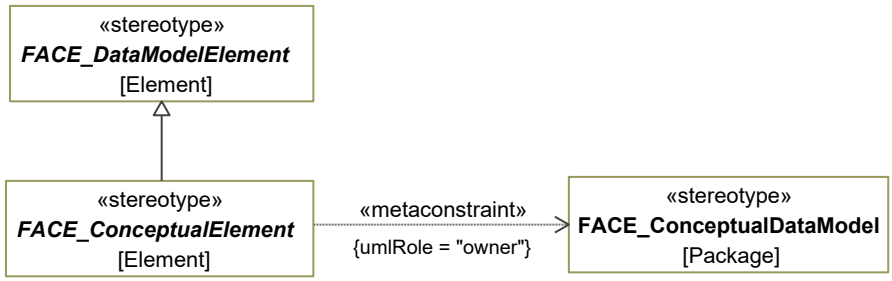

Figure 7-29: abstract FACE_ConceptualElement

Constraints

[1] FACE_ConceptualElement.owner	Elements that are stereotyped by specializations of this abstract stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_ConceptualDataModel»

FACE Conformance/OCL Constraints

[1] FACE_ConceptualElement.hasUniqueName	Each FACE ConceptualElement must have a unique name.

FACE_ConceptualEntity

Package: ConceptualDataModel
isAbstract: No
Generalization: FACE_ConceptualComposableElement, FACE_TraceableElement, FACE_SpecializationOwner
Extension: Class

Description

A FACE_ConceptualEntity represents a domain concept in terms of its FACE_Observables and other composed FACE_ConceptualEntities. Since a FACE_ConceptualEntity is built only from FACE_ConceptualComposableElements, it is independent of any specific data representation, units, or reference frame.

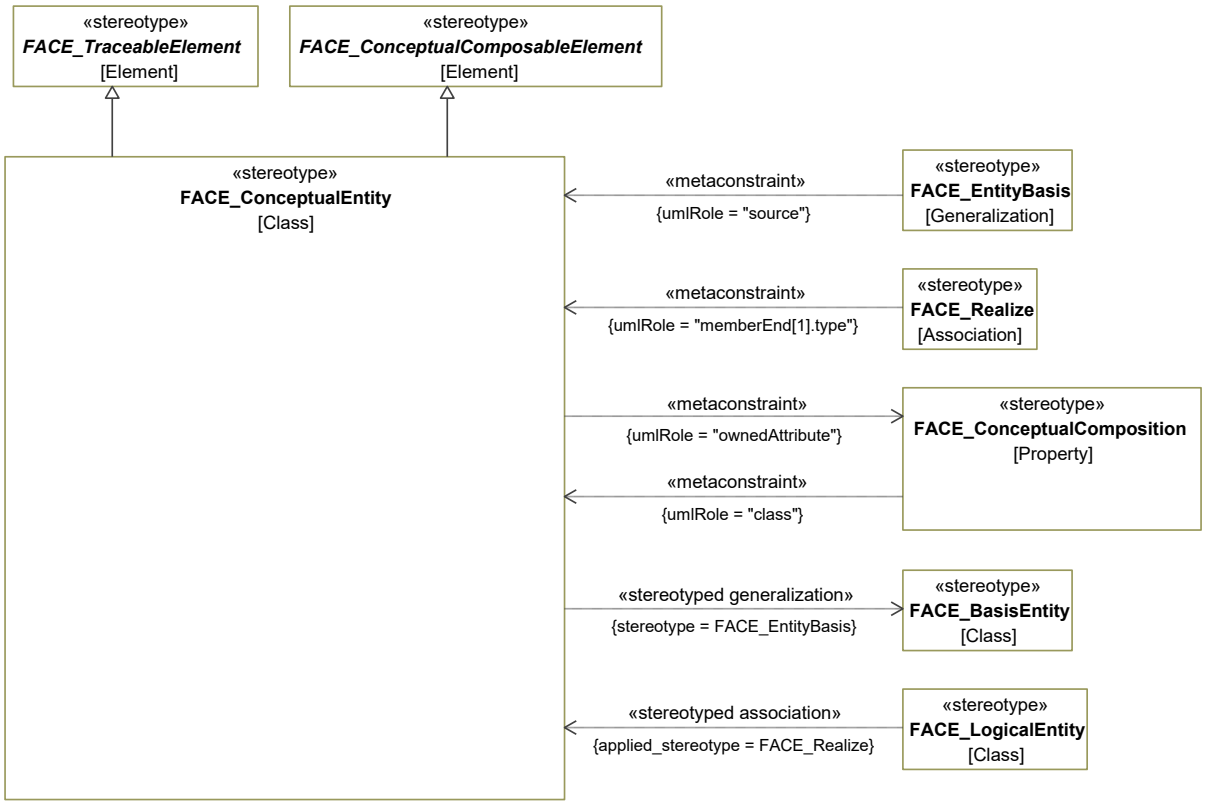

Figure 7-30: FACE_ConceptualEntity

Constraints

[1] FACE_ConceptualEntity.ownedAttribute	The value for the ownedAttribute metaproperty must be stereotyped «FACE_ConceptualComposition» or its specializations

FACE Conformance/OCL Constraints

[1] FACE_ConceptualEntity.characteristicsHaveUniqueRolenames	A Characteristic's rolename must be unique within a FACE_ConceptualEntity.
[2] FACE_ConceptualEntity.entityIsUnique	A FACE_ConceptualEntity must be unique in a Conceptual Data Model. (An Entity must be unique if the set of its Characteristics is different from other FACE_ConceptualEntities' in terms of type, lowerBound, upperBound, and path (for Participants)). NOTE: If a FACE_ConceptualEntity is part of a specialization cycle, its uniqueness must be undefined. So, if a FACE_ConceptualEntity must be part of a specialization cycle, it will not fail entityIsUnique, but will fail noCyclesInSpecialization.
[3] FACE_ConceptualEntity.hasAtLeastOneLocalCharacteristic	A FACE_ConceptualEntity must have at least one Characteristic defined locally (not through generalization).

[4] FACE_ConceptualEntity.noCyclesInSpecialization	A FACE_ConceptualEntity must not be a specialization of itself.
[5] FACE_ConceptualEntity.nonEmptyDescription	The following FACE elements must have a non-empty description: - FACE_Observable - FACE_Unit - FACE_Landmark - FACE_ReferencePoint - FACE_MeasurementSystem - FACE_MeasurementSystemAxis - FACE_CoordinateSystem - FACE_CoordinateSystemAxis - FACE_MeasurementSystemConversion - FACE_Boolean - FACE_Character - FACE_Numeric - FACE_Integer - FACE_Natural - FACE_NonNegativeReal - FACE_Real - FACE_String
[6] FACE_ConceptualEntity.observableComposedOnce	A FACE_ConceptualEntity may not compose the same FACE_Observable more than once.
[7] FACE_ConceptualEntity.specializingCharacteristicsConsistent	If FACE_ConceptualEntity A' specializes FACE_ConceptualEntity A, all characteristics in A' specialize nothing, specialize characteristics from A, or specialize characteristics from a FACE_ConceptualEntity that must be a generalization of A. (If A' does not specialize, none of its characteristics specialize.)

FACE_ConceptualParticipant

Package: ConceptualDataModel
isAbstract: No
Generalization: FACE_ConceptualCharacteristic
Extension: Class

Description

A FACE_ConceptualParticipant is the mechanism that allows a FACE_ConceptualAssociation to be constructed between two or more FACE_ConceptualEntities. The type of a conceptual Participant is the FACE_ConceptualEntity being used to construct the FACE_ConceptualAssociation. The sourceLowerBound and sourceUpperBound attributes define the multiplicity of the FACE_ConceptualAssociation relative to the Participant. A sourceUpperBound multiplicity of -1 represents an unbounded sequence. The path attribute of the Participant describes the chain of entity characteristics to traverse to reach the subject of the association beginning with the entity referenced by the type attribute.

The strings provided in the "path" tagged value are a representation of the full set of Conceptual CharacteristicPathNode, ParticipantPathNode, and PathNode elements in the path attribute as specified in the Technical Standard for Future Airborne Capability Environment (FACE™), Edition 3.0. The notation used for path string is described in Section 3.6.4.1.1.3 of the Technical Standard for Future Airborne Capability Environment (FACE™), Edition 2.1. The two notations (elements and

string) are interchangeable using a translation algorithm. XMI exchange mechanisms between models using the FACE Profile for UAF and the FACE XMI (face) file are required to translate between the two notations.

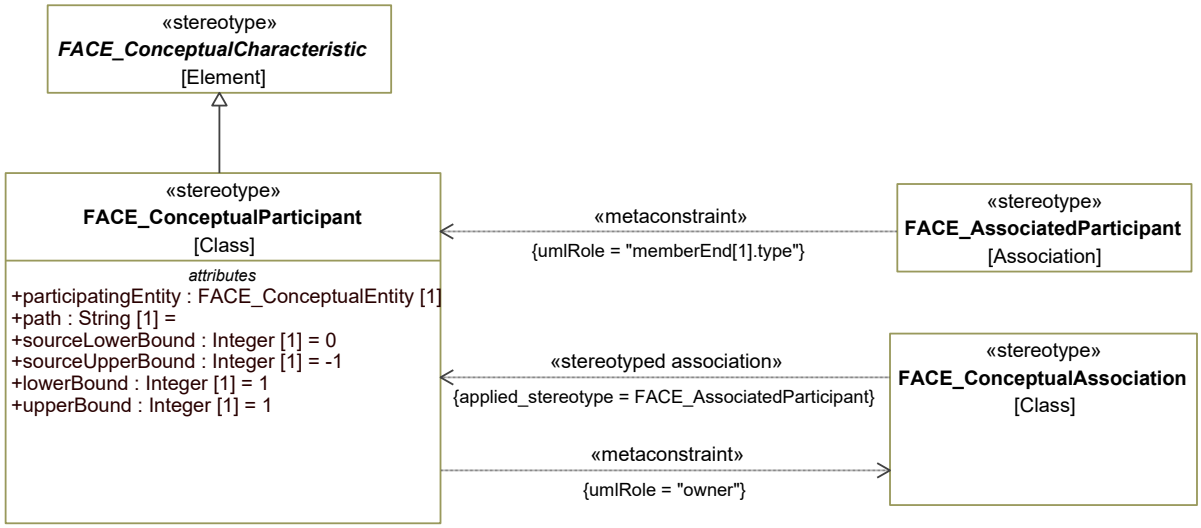

Figure 7-31: FACE_ConceptualParticipant

Attributes

lowerBound : Integer [1]

participatingEntity : FACE_ConceptualEntity [1]

path : String [1]

sourceLowerBound : Integer [1]

sourceUpperBound : Integer [1]

upperBound : Integer [1]

Constraints

[1] FACE_ConceptualParticipant.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_ConceptualAssociation»

FACE Conformance/OCL Constraints

[1] FACE_ConceptualParticipant.multiplicityConsistentWithSpecialization	If a FACE_ConceptualParticipant specializes, its multiplicity must be at least as restrictive as the FACE_ConceptualParticipant it specializes.
[2] FACE_ConceptualParticipant.pathNodeResolvable	If a FACE_ConceptualParticipant has a path sequence, the first PathNode in the sequence must be resolvable from the type of the FACE_ConceptualParticipant.

[3] FACE_ConceptualParticipant.rolenameDefined	A FACE_ConceptualParticipant must have a rolename, either projected from a characteristic or defined directly on the FACE_ConceptualParticipant.
[4] FACE_ConceptualParticipant.specializationDistinct	If a FACE_ConceptualParticipant specializes, its type, PathNode sequence, or multiplicity must be different from the FACE_ConceptualParticipant it specializes.
[5] FACE_ConceptualParticipant.typeConsistentWithSpecialization	If a FACE_ConceptualParticipant specializes, it specializes a FACE_ConceptualParticipant. If FACE_ConceptualParticipant "A" specializes FACE_ConceptualParticipant "B", then A's type must be the same or a specialization of B's type, and A's PathNode sequence is "equal to" or "specializes" B's PathNode sequence (see "pathIsEqual" and "pathIsSpecializationOf" helper methods).

FACE_ConceptualQuery

Package: ConceptualDataModel
isAbstract: No
Generalization: FACE_ConceptualView
Extension: Class

Description

A FACE_ConceptualQuery is a specification that defines the content of FACE_ConceptualView as a set of FACE_ConceptualCharacteristics projected from a selected set of related FACE_ConceptualEntities. The specification attribute captures the specification of a Query as defined by the Query grammar in Appendix J.3.

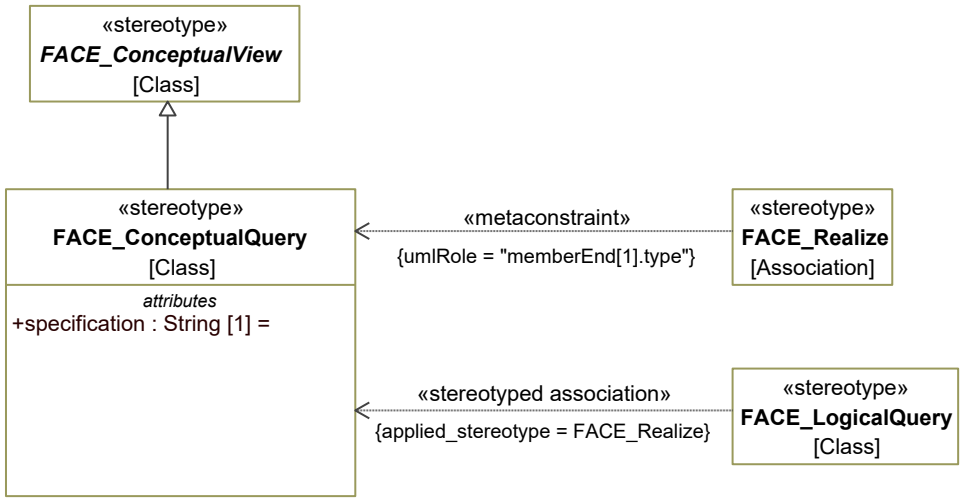

Figure 7-32: FACE_ConceptualQuery

Attributes

specification : String [1]

FACE_ConceptualQueryComposition

Package: ConceptualDataModel
isAbstract: No
Generalization: FACE_ModelElement
Extension: Property

Description

A FACE_ConceptualQueryComposition is the mechanism that allows a FACE_ConceptualCompositeQuery to be constructed from FACE_ConceptualQueries and other FACE_ConceptualCompositeQueries. The rolename attribute defines the name of the composed FACE_ConceptualView within the scope of the composing FACE_ConceptualCompositeQuery. The type of a FACE_ConceptualQueryComposition is the FACE_ConceptualView being used to construct the FACE_ConceptualCompositeQuery.

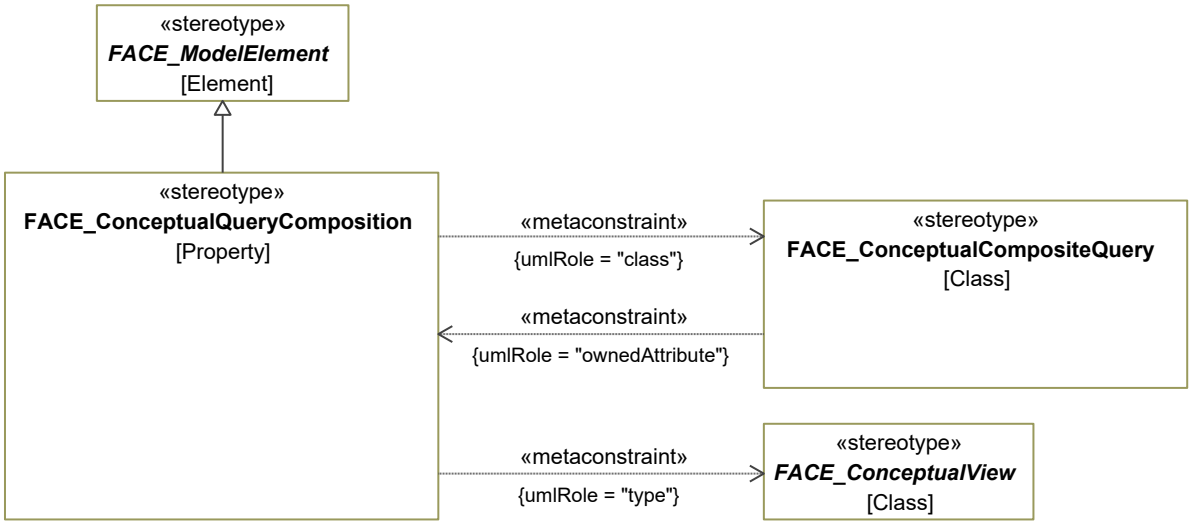

Figure 7-33: FACE_ConceptualQueryComposition

Constraints

[1] FACE_ConceptualQueryComposition.class	Value for the class metaproperty must be stereotyped «FACE_ConceptualCompositeQuery».
[2] FACE_ConceptualQueryComposition.type	Value for the type metaproperty must be stereotyped «FACE_ConceptualView» or its specializations.

FACE Conformance/OCL Constraints

[1] FACE_ConceptualQueryComposition.rolenameIsValidIdentifier	The rolename of a FACE_ConceptualQueryComposition must be a valid identifier.

FACE_ConceptualView

Package: ConceptualDataModel
isAbstract: Yes
Generalization: FACE_ConceptualElement, FACE_TraceableElement
Extension: Class

Description

A FACE_ConceptualView is a FACE_ConceptualQuery or a FACE_ConceptualCompositeQuery.

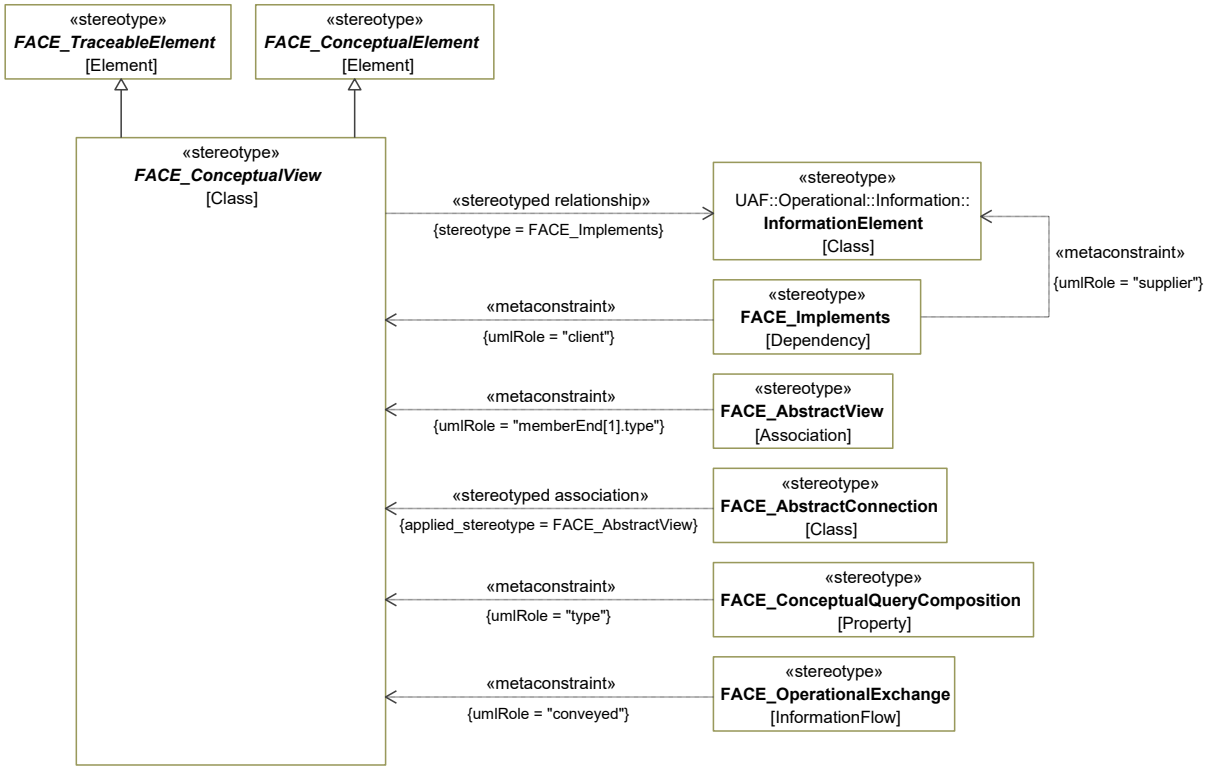

Figure 7-34: abstract FACE_ConceptualView

FACE_Domain

Package: ConceptualDataModel
isAbstract: No
Generalization: FACE_ConceptualElement
Extension: Class

Description

A FACE_Domain represents a space defined by a set of FACE_BasisEntities relating to well understood concepts by practitioners within the domain.

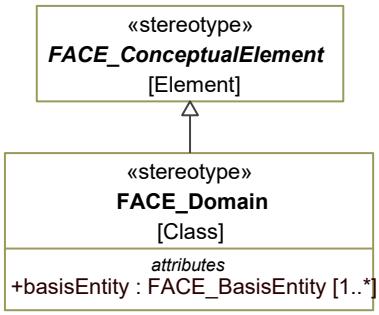

Figure 7-35: FACE_Domain

Attributes

basisEntity : FACE_BasisEntity [1..*]

FACE_EntityBasis

Package: ConceptualDataModel
isAbstract: No
Extension: Generalization

Description

Used to indicate a specialization between FACE_ConceptualEntity types and FACE_BasisEntities.

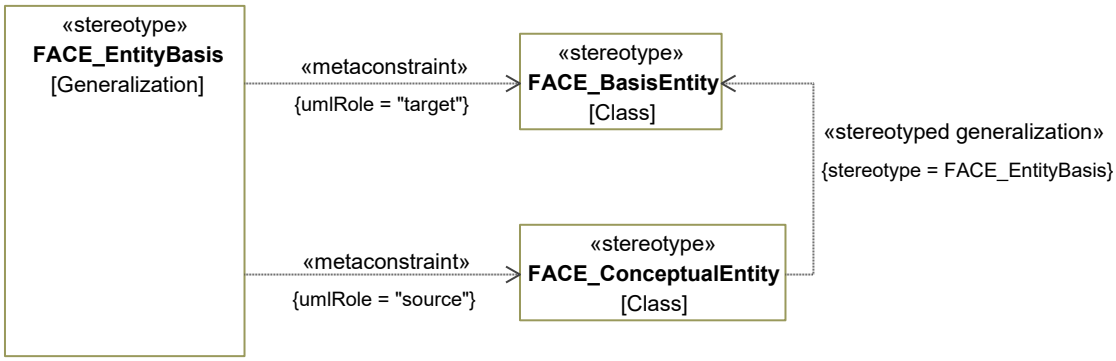

Figure 7-36: FACE_EntityBasis

Constraints

[1] FACE_EntityBasis.source	The value for the source metaproperty must be stereotyped by «FACE_ConceptualEntity» or a specialization of «FACE_ConceptualEntity».
[2] FACE_EntityBasis.target	The value for the target metaproperty must be stereotyped by «FACE_BasisEntity».

FACE_Observable

Package: ConceptualDataModel
isAbstract: No
Generalization: FACE_BasisElement

Extension: Class

Description

A FACE_Observable is something that can be observed but not further characterized and is typically quantified through measurements of the physical world. An observable is independent of any specific data representation, units, or reference frame. For example, length may be thought of as an observable in that it can be measured, but at the conceptual level the nature of the measurement is not specified.

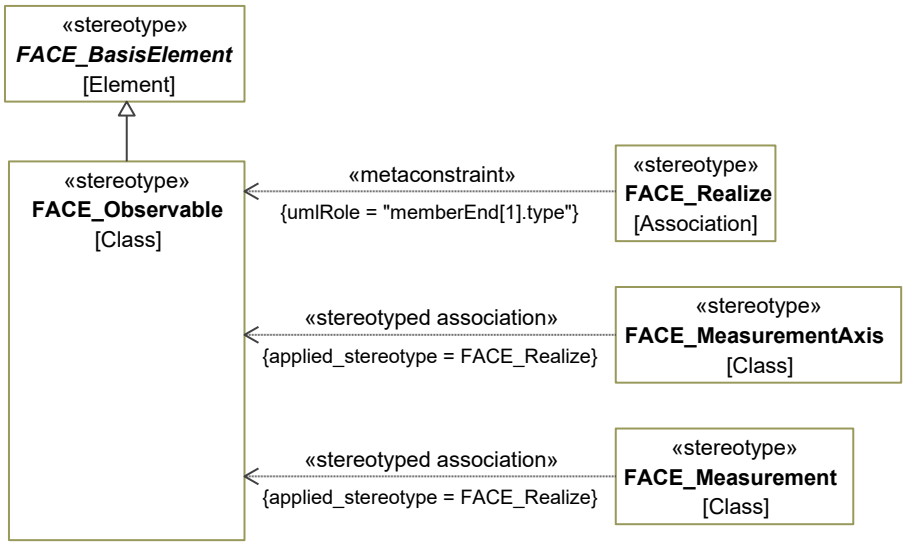

Figure 7-37: FACE_Observable

FACE Conformance/OCL Constraints

[1] FACE_Observable.nonEmptyDescription FACE_Observable must have a non-empty description.

7.1.1.1.2 FACE_UAF_Profile::FACE Data Architecture::FACE Data Model::LogicalDataModel

FACE_AbstractMeasurement

Package: LogicalDataModel
isAbstract: Yes
Extension: Element

Description

A FACE_AbstractMeasurement is a FACE_Measurement, FACE_MeasurementAxis, or a FACE_ValueTypeUnit.

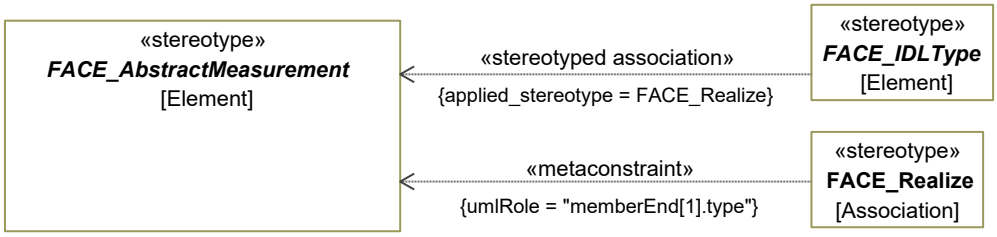

Figure 7-38: abstract FACE_AbstractMeasurement

FACE_AbstractMeasurementSystem

Package: LogicalDataModel
isAbstract: Yes
Generalization: FACE_LogicalElement
Extension: Class

Description

A FACE_AbstractMeasurementSystem is an abstract parent for FACE_StandardMeasurementSystems and FACE_MeasurementSystems. It is used for structural simplicity in the metamodel.

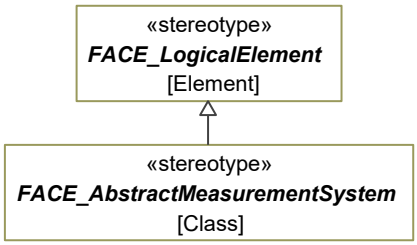

Figure 7-39: abstract FACE_AbstractMeasurementSystem

FACE_AffineConversion

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_Conversion

Description

A FACE_AffineConversion is a relationship between two FACE_ConvertibleElements in the form mx+b.

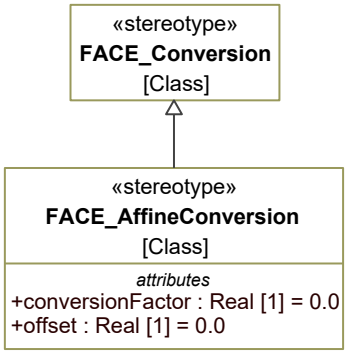

Figure 7-40: FACE_AffineConversion

Attributes

conversionFactor : Real [1]

offset : Real [1]

FACE_AppliedConstraint

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to identify constraints that apply to FACE_MeasurementSystem elements.

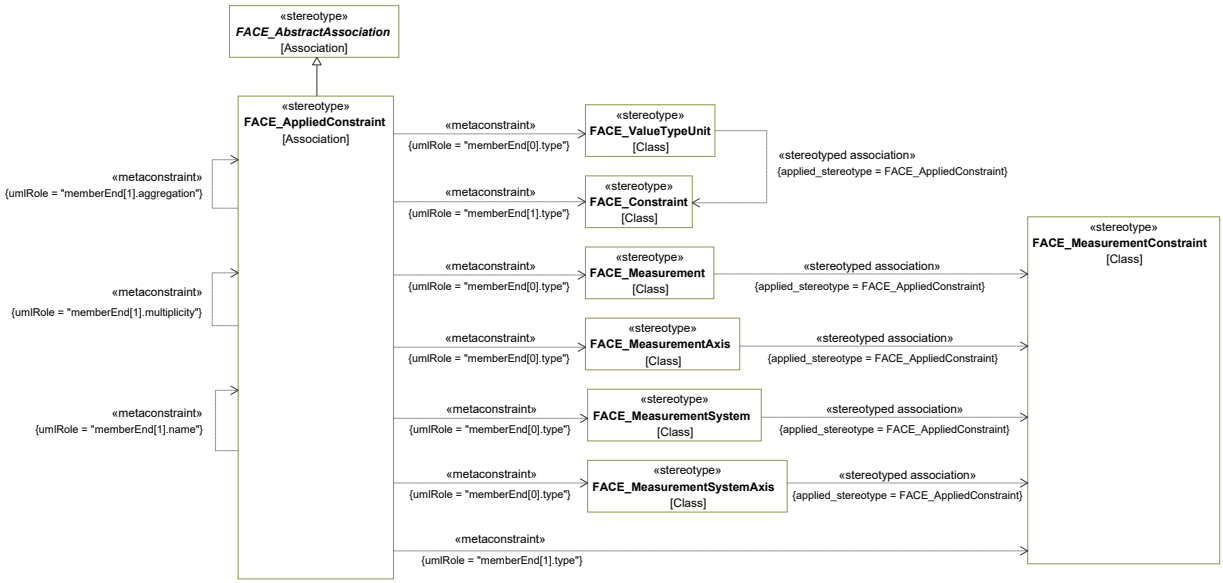

Figure 7-41: FACE_AppliedConstraint

Constraints

[1] FACE_AppliedConstraint.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by a one of the following stereotypes: «FACE_ValueTypeUnit» «FACE_Measurement» «FACE_MeasurementAxis» «FACE_MeasurementSystem» «FACE_MeasurementSystemAxis»
[2] FACE_AppliedConstraint.memberEnd[1].aggregation	composite
[3] FACE_AppliedConstraint.memberEnd[1].multiplicity	0..*
[4] FACE_AppliedConstraint.memberEnd[1].name	"constraint"
[5] FACE_AppliedConstraint.memberEnd[1].type	Based on the FACE_AppliedConstraint.memberEnd[0].type value's stereotype: = «FACE_ValueTypeUnit», the memberEnd[1].type metaproperty must be stereotyped by «FACE_Constraint» = «FACE_Measurement», «FACE_MeasurementAxis», «FACE_MeasurementSystem», or «FACE_MeasurementSystemAxis», the memberEnd[1].type metaproperty must be stereotyped by «FACE_MeasurementConstraint»

FACE_AppliedValueTypeUnit

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to associate FACE_Measurement and FACE_MeasurementSystem Axes with the logical descriptions of the data types that characterize them.

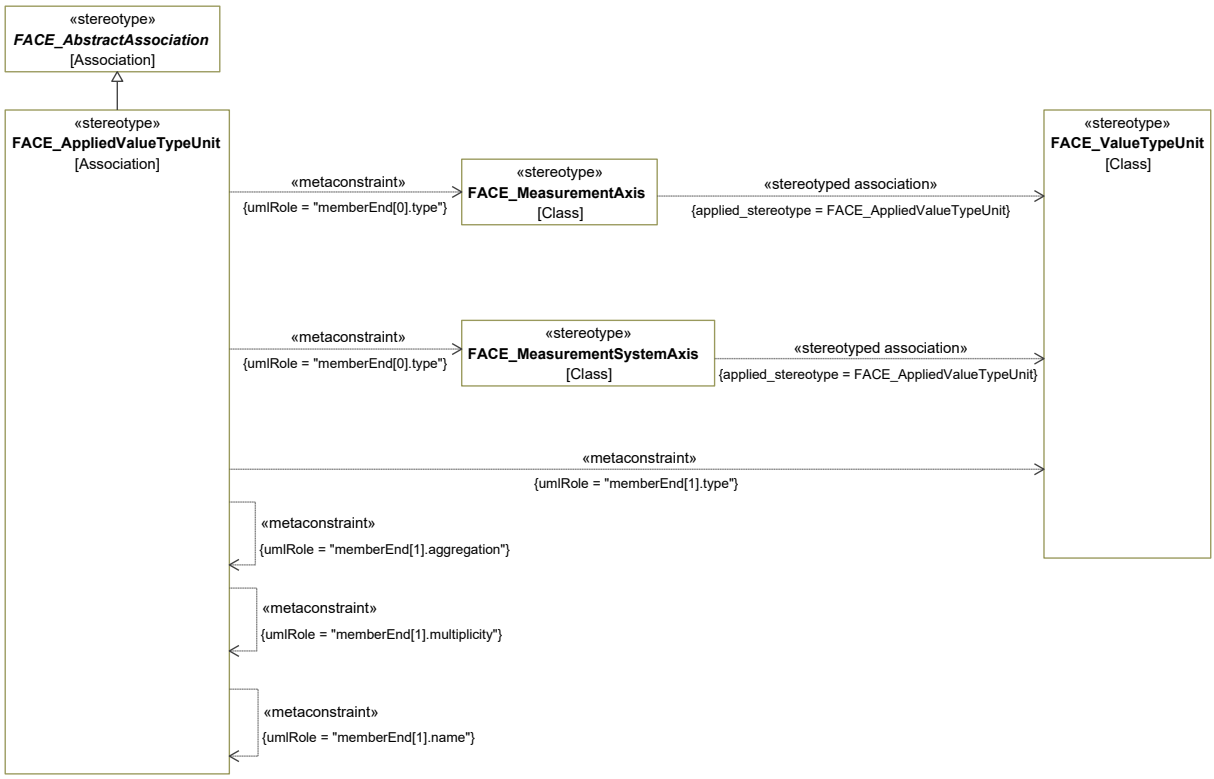

Figure 7-42: FACE_AppliedValueTypeUnit

Constraints

[1] FACE_AppliedValueTypeUnit.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by one of the following: «FACE_MeasurementAxis» «FACE_MeasurementSystemAxis»
[2] FACE_AppliedValueTypeUnit.memberEnd[1].aggregation	none
[3] FACE_AppliedValueTypeUnit.memberEnd[1].multiplicity	Based on the stereotype of the memberEnd[0].type metaproperty: = Specialization of «FACE_MeasurementAxis», memberEnd[1].multiplicity is 0..* = Specialization of «FACE_MeasurementSystemAxis», memberEnd[1].multiplicity is 1..*
[4] FACE_AppliedValueTypeUnit.memberEnd[1].name	Based on the stereotype of the memberEnd[0].type metaproperty: = Specialization of «FACE_MeasurementAxis», memberEnd[1].name is "valueTypeUnit" = Specialization of «FACE_MeasurementSystemAxis», memberEnd[1].name is "defaultValueTypeUnit"
[5] FACE_AppliedValueTypeUnit.memberEnd[1].type	The value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_ValueTypeUnit».

FACE_Axis

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to associate FACE_Measurements, FACE_MeasurementSystems, and FACE_CoordinateSystems to the axes that characterize them.

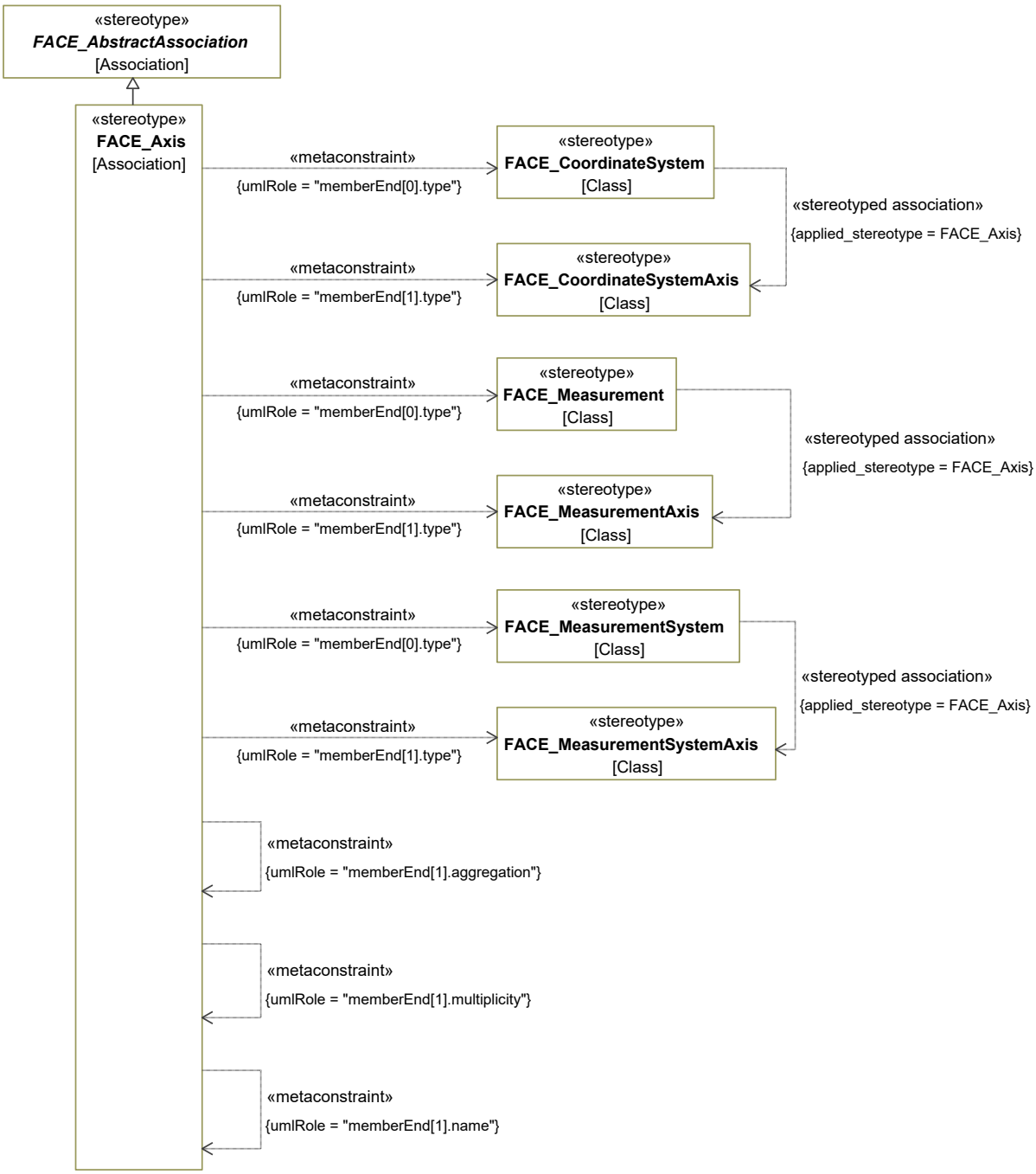

Figure 7-43: FACE_Axis

Constraints

[1] FACE_Axis.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by one of the following: «FACE_CoordinateSystem» «FACE_Measurement» «FACE_MeasurementSystem»

[2] FACE_Axis.memberEnd[1].aggregation	none
[3] FACE_Axis.memberEnd[1].multiplicity	Based on the stereotype of the memberEnd[0].type metaproperty: = «FACE_CoordinateSystem», memberEnd[1].multiplicity is 1..* = «FACE_Measurement», memberEnd[1].multiplicity is 0..* = «FACE_MeasurementSystem», memberEnd[1].multiplicity is 0..1
[4] FACE_Axis.memberEnd[1].name	Based on the stereotype of the memberEnd[1].type metaproperty: = «FACE_CoordinateSystemAxis», memberEnd[1].name is "coordinateSystemAxis" = «FACE_MeasurementAxis», memberEnd[1].name is "measurementAxis" = «FACE_MeasurementSystemAxis», memberEnd[1].name is "measurementSystemAxis"
[5] FACE_Axis.memberEnd[1].type	Based on the FACE_Axis.source value's stereotype: = «FACE_CoordinateSystem», the memberEnd[1].type metaproperty must be stereotyped by «FACE_CoordinateSystemAxis» = «FACE_Measurement», the memberEnd[1].type metaproperty must be stereotyped by «FACE_MeasurementAxis» = «FACE_MeasurementSystem», the memberEnd[1].type metaproperty must be stereotyped by «FACE_MeasurementSystemAxis»

FACE_Constraint

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_Element
Extension: Class

Description

A FACE_Constraint limits the set of possible values for the FACE_ValueType of a FACE_MeasurementSystem or FACE_Measurement.

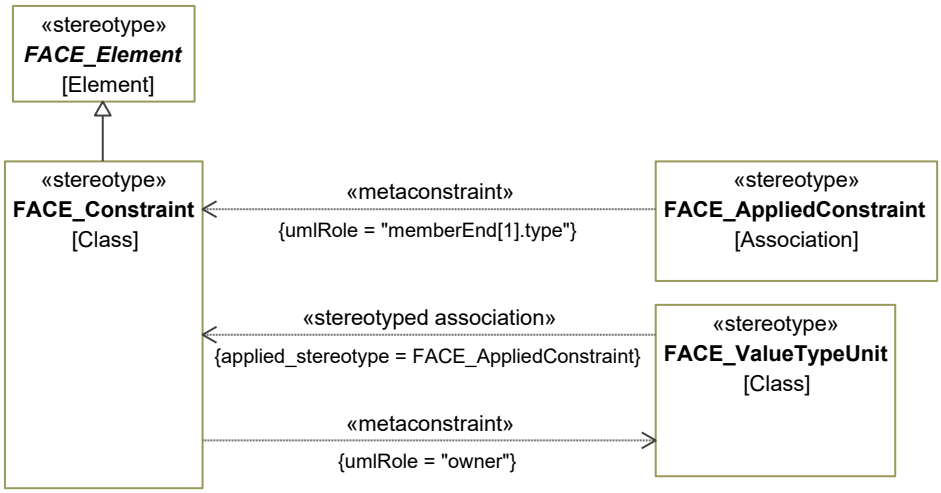

Figure 7-44: FACE_Constraint

Constraints

[1] FACE_Constraint.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_ValueTypeUnit»

FACE_Conversion

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalElement
Extension: Class

Description

A FACE_Conversion is a relationship between two FACE_ConvertibleElements that describes how to transform measured quantities between two FACE_Units.

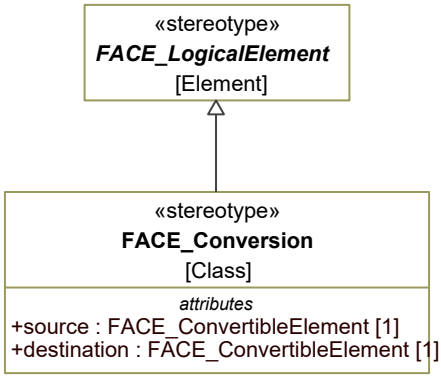

Figure 7-45: FACE_Conversion

Attributes

destination : FACE_ConvertibleElement [1]

source : FACE_ConvertibleElement [1]

FACE_ConvertibleElement

Package: LogicalDataModel
isAbstract: Yes
Generalization: FACE_LogicalElement

Description

A FACE_ConvertibleElement is a FACE_Unit.

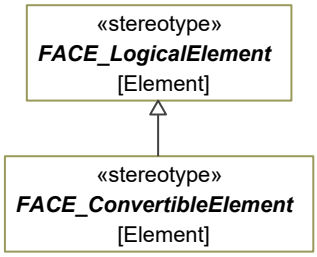

Figure 7-46: abstract FACE_ConvertibleElement

FACE_CoordinateSystem

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalElement
Extension: Class

Description

A FACE_CoordinateSystem is a system which uses one or more coordinates to uniquely determine the position of a point in an N-dimensional space. The coordinate system is comprised of multiple FACE_CoordinateSystemAxis which completely span the space. Coordinates are quantified relative to the FACE_CoordinateSystemAxis. It is not required that the dimensions be ordered or continuous.

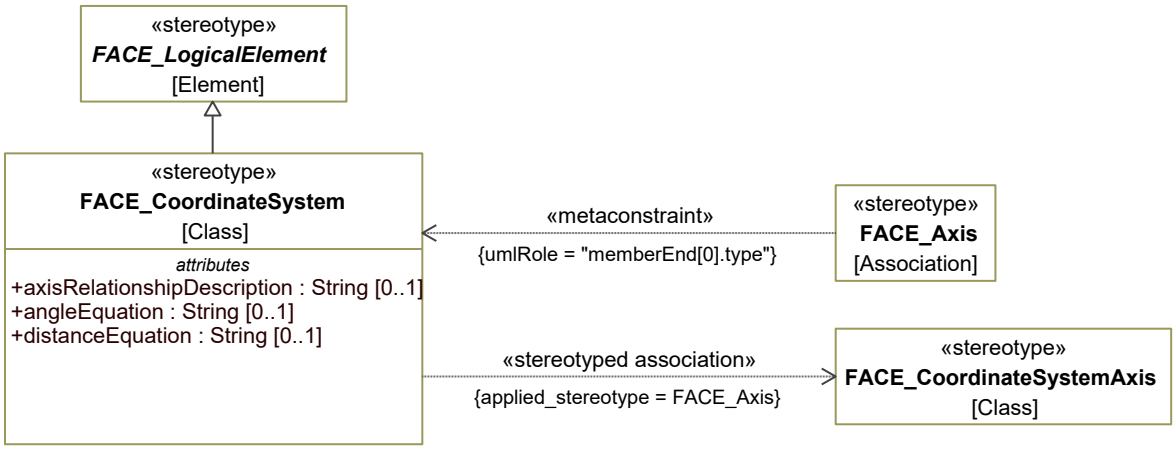

Figure 7-47: FACE_CoordinateSystem

Attributes

angleEquation : String [0..1]

axisRelationshipDescription : String [0..1]

distanceEquation : String [0..1]

FACE Conformance/OCL Constraints

[1] FACE_CoordinateSystem.nonEmptyDescription FACE_CoordinateSystem must have a non-empty description.

FACE_CoordinateSystemAxis

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalElement
Extension: Class

Description

A FACE_CoordinateSystemAxis represents a dimension within a FACE_CoordinateSystem.

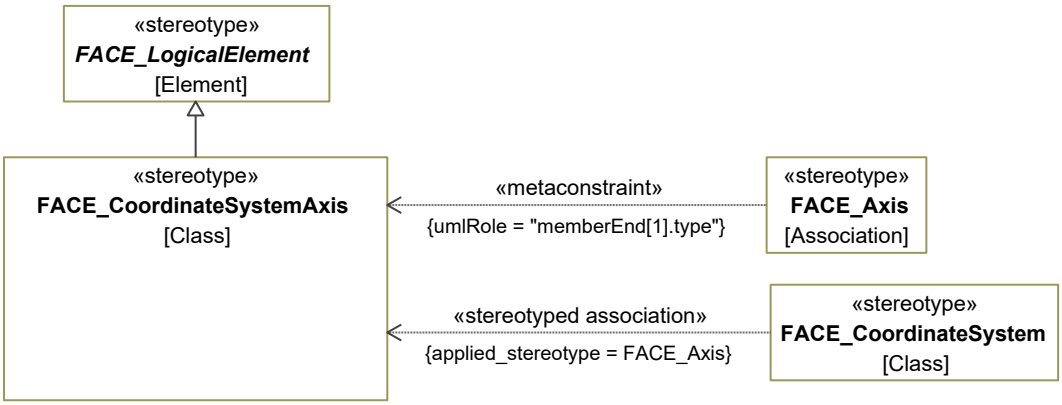

Figure 7-48: FACE_CoordinateSystemAxis

FACE Conformance/OCL Constraints

[1] FACE_CoordinateSystemAxis.nonEmptyDescription	FACE_CoordinateSystemAxis must have a non-empty description.

FACE_DefinedReferencePoint

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to identify the reference point that characterizes a FACE_MeasurementSystem.

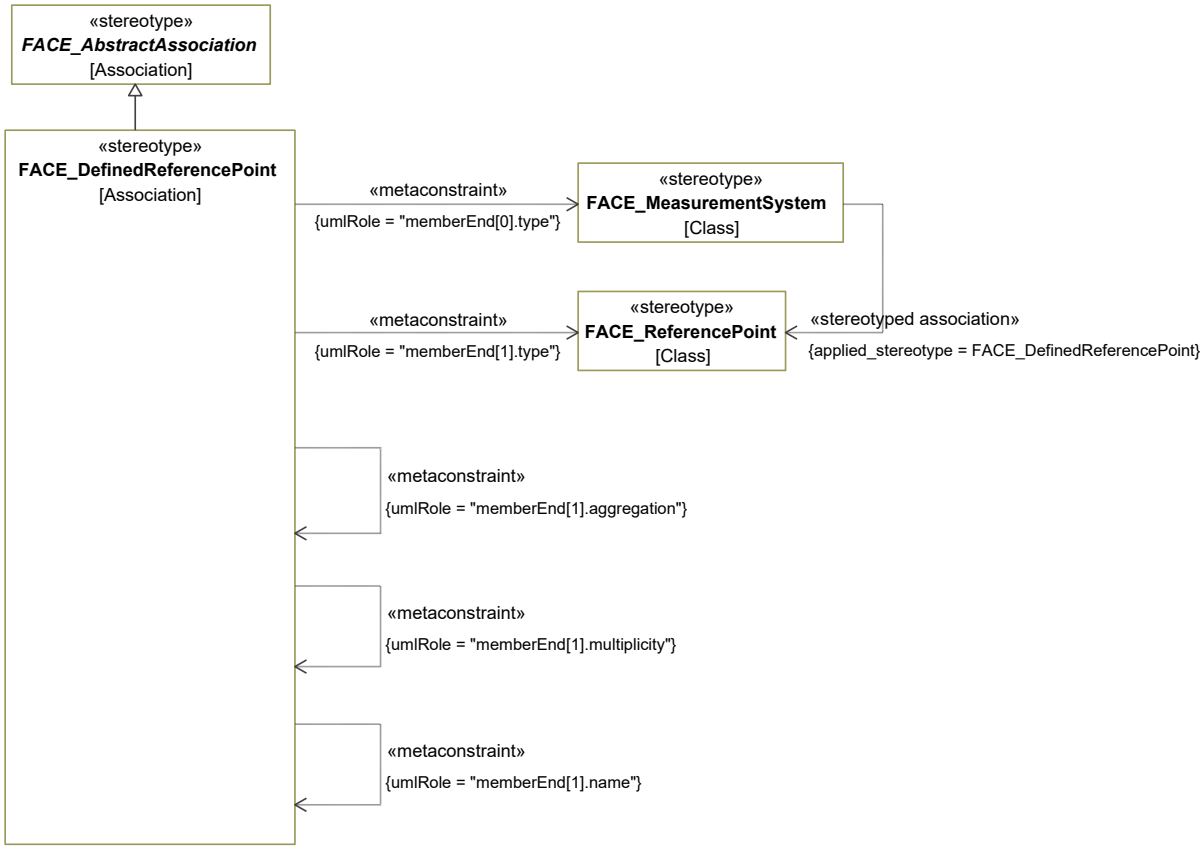

Figure 7-49: FACE_DefinedReferencePoint

Constraints

[1] FACE_DefinedReferencePoint.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_MeasurementSystem».
[2] FACE_DefinedReferencePoint.memberEnd[1].aggregation	composite
[3] FACE_DefinedReferencePoint.memberEnd[1].multiplicity	0..*
[4] FACE_DefinedReferencePoint.memberEnd[1].name	"referencePoint"
[5] FACE_DefinedReferencePoint.memberEnd[1].type	The value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_ReferencePoint».

FACE_EnumerationConstraint

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_Constraint

Description

A FACE_EnumerationConstraint identifies a subset of enumerated values (EnumerationLabel) considered valid for a FACE_Enumerated value type of a FACE_MeasurementAxis.

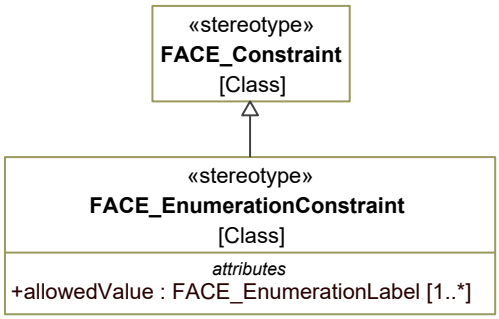

Figure 7-50: FACE_EnumerationConstraint

Attributes

allowedValue : FACE_EnumerationLabel [1..*]

FACE_EnumerationLabel

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_Element
Extension: Property

Description

A FACE_EnumerationLabel defines a named member of a FACE_Enumerated value set.

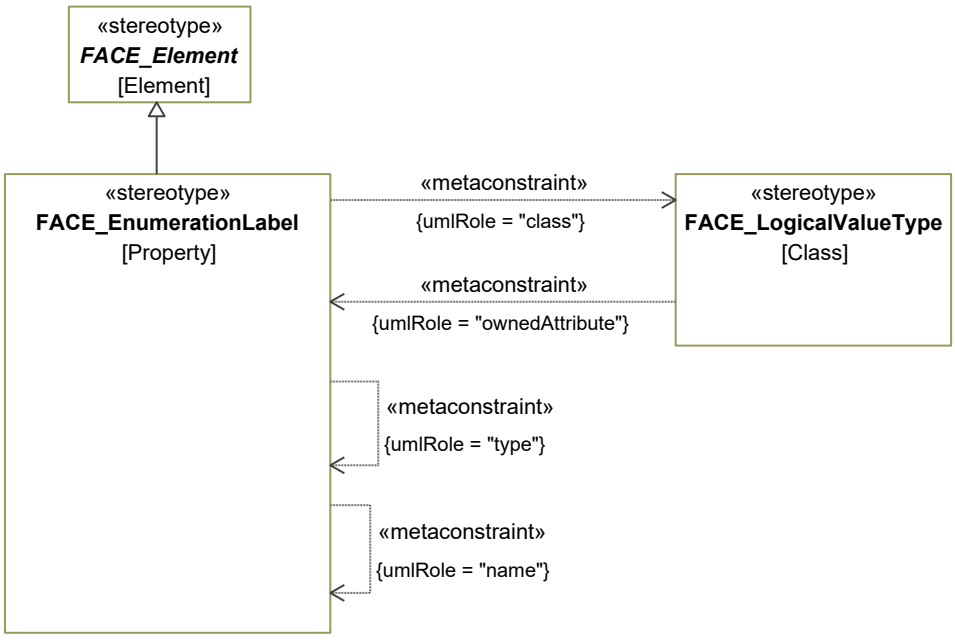

Figure 7-51: FACE_EnumerationLabel

Constraints

[1] FACE_EnumerationLabel.class	Value for the class metaproperty must be stereotyped «FACE_LogicalValueType»
[2] FACE_EnumerationLabel.name	Value for the name metaproperty must not be null and must be unique within the owning class.
[3] FACE_EnumerationLabel.type	Value for the type metaproperty must be null. (The name metaproperty is the only valid information.)

FACE Conformance/OCL Constraints

[1] FACE_EnumerationLabel.nameIsNotReservedWord	A FACE_EnumerationLabel's name may not be an IDL reserved word.

FACE_FixedLengthStringConstraint

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_StringConstraint

Description

A FACE_FixedLengthStringConstraint specifies a defined set of meaningful values for a String as with of a specific fixed length. The length attribute defines the fixed length, an integer value greater than 0.

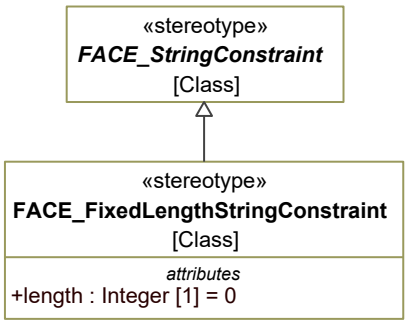

Figure 7-52: FACE_FixedLengthStringConstraint

Attributes

length : Integer [1]

FACE Conformance/OCL Constraints

[1] FACE_FixedLengthStringConstraint.nonNegativeLength	A FACE_FixedLengthStringConstraint's length must be greater than zero.

FACE_IntegerConstraint

Package: LogicalDataModel
isAbstract: Yes

Generalization: FACE_Constraint

Description

A FACE_IntegerConstraint specifies a defined set of meaningful values for an Integer or Natural.

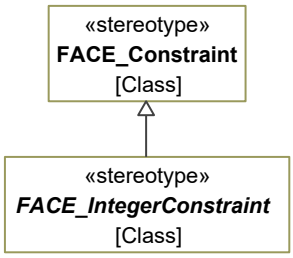

Figure 7-53: abstract FACE_IntegerConstraint

FACE_IntegerRangeConstraint

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_IntegerConstraint

Description

A FACE_IntegerRangeConstraint specifies a defined range of meaningful values for an Integer or Natural. The upperBound is greater than or equal to the lowerBound. The defined range is inclusive of the upperBound and lowerBound.

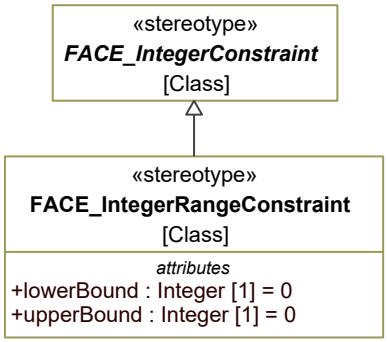

Figure 7-54: FACE_IntegerRangeConstraint

Attributes

lowerBound : Integer [1]

upperBound : Integer [1]

FACE_Landmark

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalElement
Extension: Class

Description

A FACE_Landmark represents a described point which relates a FACE_ReferencePoint to a well-known location.

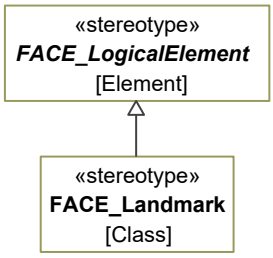

Figure 7-55: FACE_Landmark

FACE Conformance/OCL Constraints

[1] FACE_Landmark.nonEmptyDescription FACE_Landmark must have a non-empty description.

FACE_LogicalAssociation

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalEntity

Description

A FACE_LogicalAssociation represents a relationship between two or more FACE_LogicalEntities. In addition, there may be one or more FACE_LogicalComposableElements that characterize the relationship. FACE_LogicalAssociations are FACE_LogicalEntities that may also participate in other FACE_LogicalAssociations.

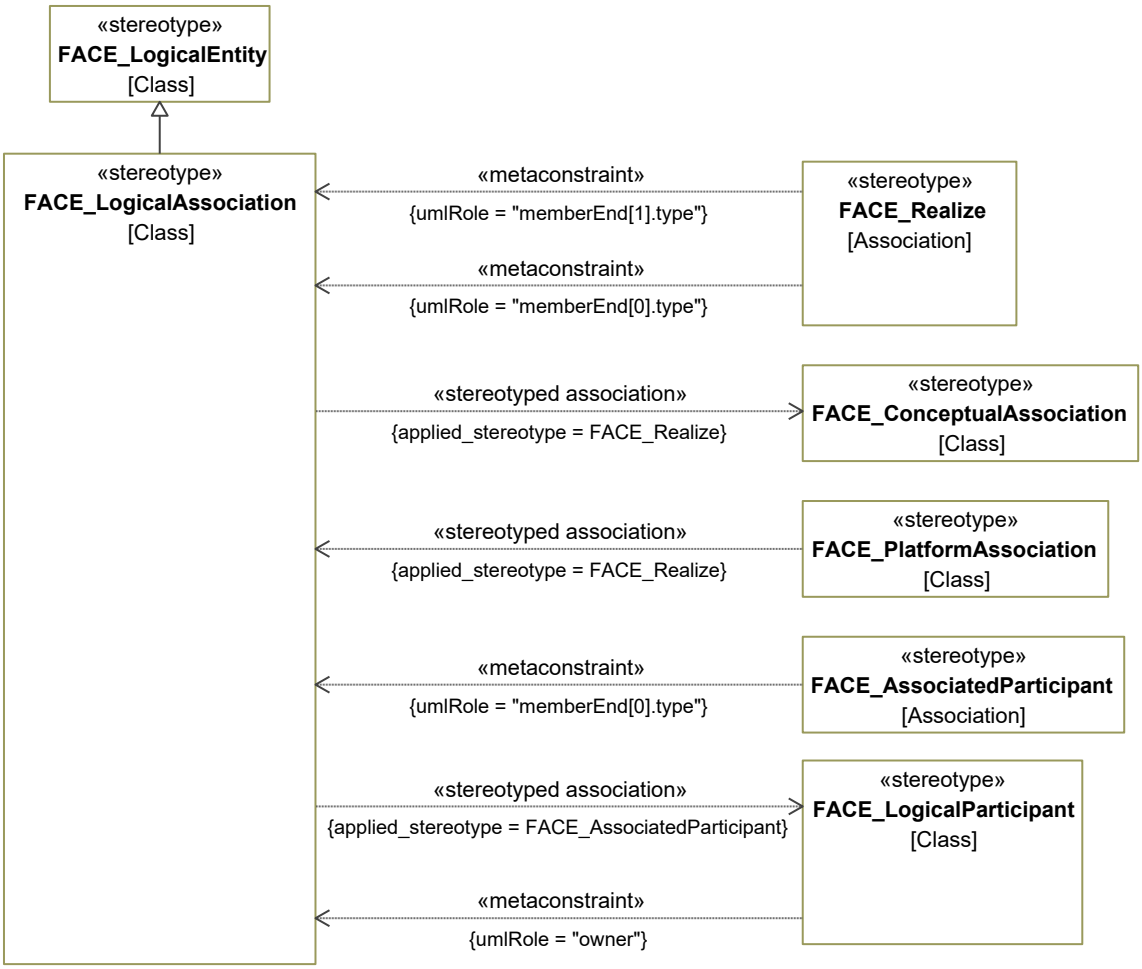

Figure 7-56: FACE_LogicalAssociation

FACE Conformance/OCL Constraints

[1] FACE_LogicalAssociation.participantsConsistentWithRealization	FACE_LogicalParticipants in a FACE_LogicalAssociation must realize FACE_ConceptualParticipants in the FACE_LogicalAssociation that the FACE_LogicalAssociation realizes.
[2] FACE_LogicalAssociation.participantsRealizeUniquely	FACE_LogicalParticipants in a FACE_LogicalAssociation must realize unique FACE_ConceptualParticipants.

FACE_LogicalCharacteristic

Package: LogicalDataModel
isAbstract: Yes
Generalization: FACE_ModelElement

Description

A FACE_LogicalCharacteristic is a defining feature of a FACE_LogicalEntity. The rolename attribute defines the name of the FACE_LogicalCharacteristic within the scope of the FACE_LogicalEntity. The lowerBound and upperBound attributes define the multiplicity of the composed Characteristic. An upperBound multiplicity of -1 represents an unbounded sequence.

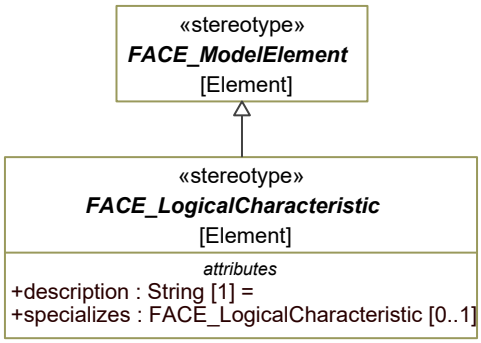

Figure 7-57: abstract FACE_LogicalCharacteristic

Attributes

description : String [1]

specializes : FACE_LogicalCharacteristic [0..1]

FACE Conformance/OCL Constraints

[1] FACE_LogicalCharacteristic.lowerBound_LTE_UpperBound	A FACE_LogicalCharacteristic's lowerBound must be less than or equal to its upperBound, unless its upperBound is -1.
[2] FACE_LogicalCharacteristic.rolenameIsValidIdentifier	The rolename of a FACE_LogicalCharacteristic must be a valid identifier.
[3] FACE_LogicalCharacteristic.specializationConsistentWithRealization	If a FACE_LogicalCharacteristic specializes, its specialization must be consistent with its realization's specialization.
[4] FACE_LogicalCharacteristic.upperBoundValid	A FACE_LogicalCharacteristic's upperBound must be equal to -1 or greater than 1.

FACE_LogicalComposableElement

Package: LogicalDataModel
isAbstract: Yes
Generalization: FACE_LogicalElement

Description

A FACE_LogicalComposableElement is a FACE_LogicalElement that is allowed to participate in a FACE_Composition relationship. In other words, these are the FACE_LogicalElements that may be a characteristic of a FACE_LogicalEntity.

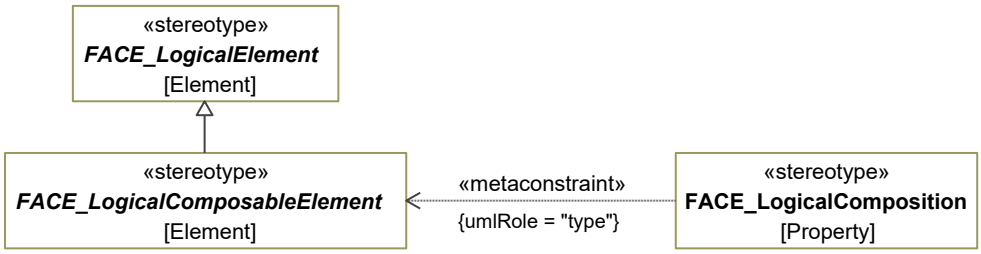

Figure 7-58: abstract FACE_LogicalComposableElement

FACE_LogicalCompositeQuery

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalView
Extension: Class

Description

A FACE_LogicalCompositeQuery is a collection of two or more FACE_LogicalQueries. The isUnion attribute specifies whether the composed FACE_LogicalQueries are intended to be represented as cases in a FACE_IDL union or as members of a FACE_IDL struct.

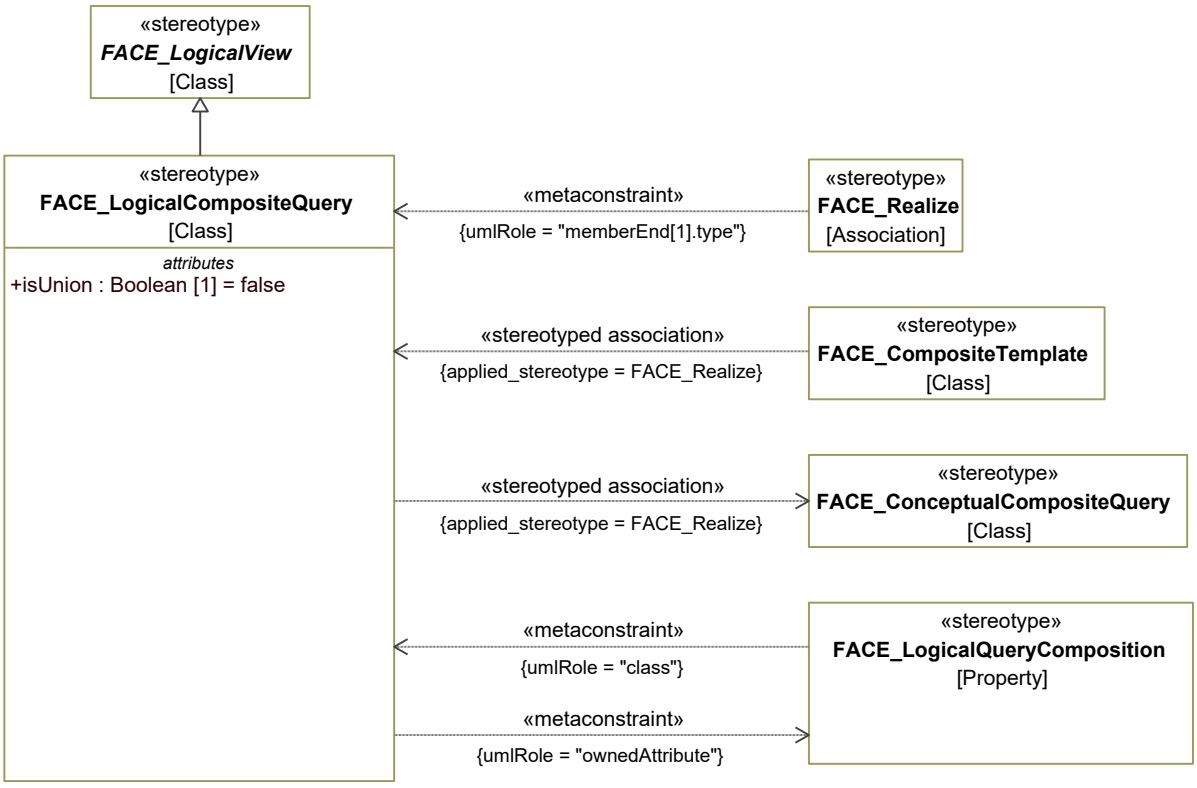

Figure 7-59: FACE_LogicalCompositeQuery

Attributes

isUnion : Boolean [1]

Constraints

[1] FACE_LogicalCompositeQuery.ownedAttribute	The values for the ownedAttribute metaproperty must meet the following criteria: - must be ordered list - must be stereotyped «FACE_LogicalQueryComposition» or its specializations - must contain 2 or more elements

FACE Conformance/OCL Constraints

[1] FACE_LogicalCompositeQuery.compositionsConsistentWithRealization	FACE_LogicalQueryCompositions in a FACE_LogicalCompositeQuery must realize FACE_ConceptualQueryCompositions in the FACE_ConceptualCompositeQuery that the FACE_LogicalCompositeQuery realizes.
[2] FACE_LogicalCompositeQuery.compositionsHaveUniqueRolenames	A FACE_LogicalQueryComposition's rolename must be unique within a FACE_LogicalCompositeQuery.
[3] FACE_LogicalCompositeQuery.noCyclesInConstruction	A FACE_LogicalCompositeQuery must not compose itself.
[4] FACE_LogicalCompositeQuery.realizationUnionConsistent	A FACE_LogicalCompositeQuery that realizes must have the same "isUnion" property as the FACE_LogicalCompositeQuery it realizes.
[5] FACE_LogicalCompositeQuery.realizedCompositionsHaveDifferentTypes	A FACE_LogicalCompositeQuery must not contain two FACE_LogicalQueryCompositions that realize the same FACE_ConceptuallQueryComposition.
[6] FACE_LogicalCompositeQuery.viewComposedOnce	A FACE_LogicalCompositeQuery must not compose the same FACE_LogicalView more than once.

FACE_LogicalComposition

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalCharacteristic
Extension: Property

Description

A FACE_LogicalComposition is the mechanism that allows FACE_LogicalEntities to be constructed from other FACE_LogicalComposableElements. The type of a FACE_LogicalComposition is the FACE_LogicalComposableElement being used to construct the FACE_LogicalEntity. The lowerBound and upperBound define the multiplicity of the composed FACE_LogicalEntity. An upperBound multiplicity of -1 represents an unbounded sequence.

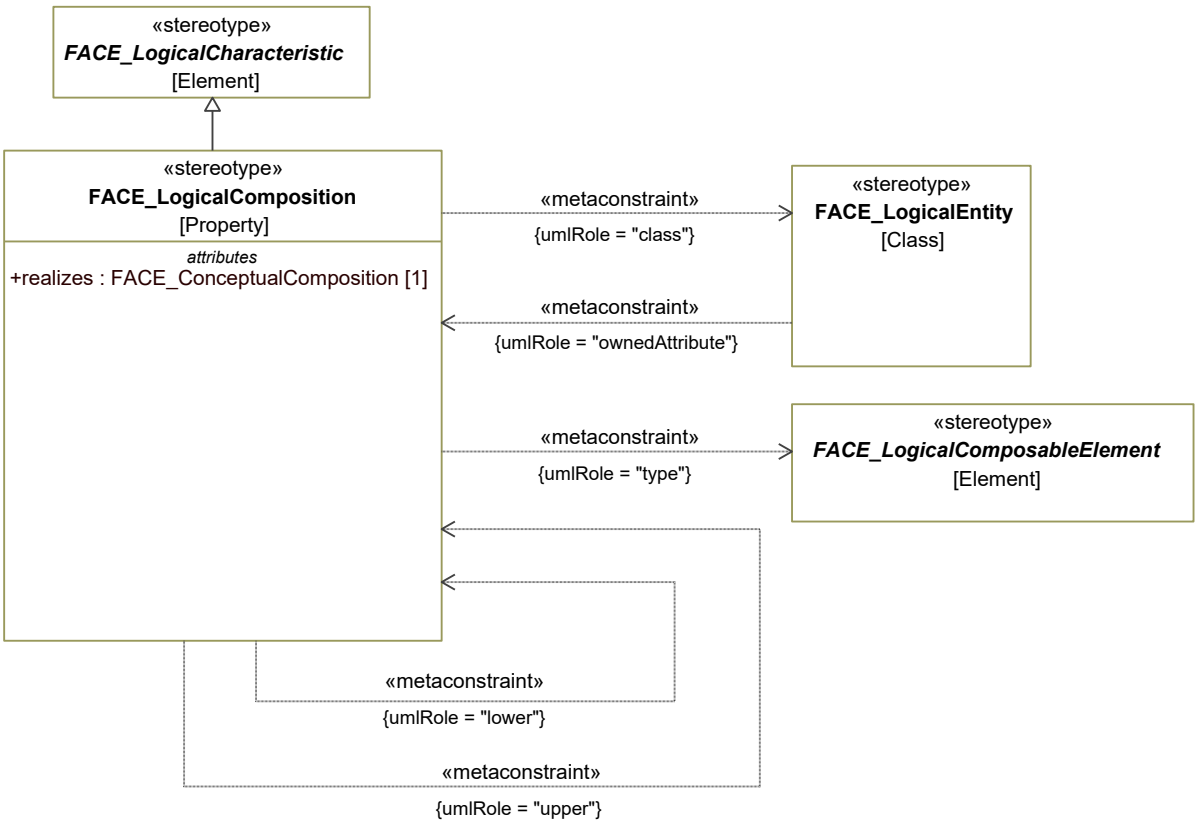

Figure 7-60: FACE_LogicalComposition

Attributes

realizes: FACE_ConceptualComposition [1]

Constraints

[1] FACE_LogicalComposition.class	Value for the class metaproperty must be stereotyped «FACE_LogicalEntity» or its specializations.
[2] FACE_LogicalComposition.lower	The value for the lower (lower bound of multiplicity) metaproperty must be an integer greater than or equal to -1.
[3] FACE_LogicalComposition.type	Value for the type metaproperty must be stereotyped «FACE_LogicalComposableElement» or its specializations.
[4] FACE_LogicalComposition.upper	The value for the upper (upper bound of multiplicity) metaproperty must be an integer greater than or equal to -1

FACE Conformance/OCL Constraints

[1] FACE_LogicalComposition.multiplicityConsistentWithRealization	A FACE_LogicalComposition's multiplicity must be at least as restrictive as the FACE_ConceptualComposition it realizes
[2] FACE_LogicalComposition.multiplicityConsistentWithSpecialization	A FACE_LogicalComposition's multiplicity must be at least as restrictive as the FACE_LogicalComposition of which it is a specialization.
[3] FACE_LogicalComposition.typeConsistentWithRealization	A FACE_LogicalComposition's type must be consistent with its realization's type.

FACE_LogicalElement

Package: LogicalDataModel
isAbstract: Yes
Generalization: FACE_DataModelElement

Description

A FACE_LogicalElement is the root type for defining the Logical Data Model elements of the FACE Data Model Language.

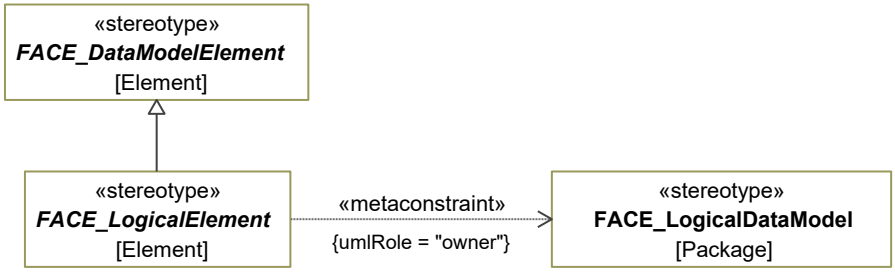

Figure 7-61: abstract FACE_LogicalElement

Constraints

[1] FACE_LogicalElement.owner	Elements that are stereotyped by specializations of this abstract stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_LogicalDataModel»

FACE Conformance/OCL Constraints

[1] FACE_LogicalElement.hasUniqueName	Every FACE_LogicalElement, with the exception of FACE_Constraint, must have a unique name.

FACE_LogicalEntity

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalComposableElement, FACE_TraceableElement, FACE_SpecializationOwner
Extension: Class

Description

A FACE_LogicalEntity "realizes" a FACE_ConceptualEntity in terms of FACE_Measurements and other FACE_LogicalEntities. Since a FACE_LogicalEntity is built from logical FACE_Measurements, it is independent of any specific platform data representation. A FACE_LogicalEntity's composition hierarchy is consistent with the composition hierarchy of the FACE_ConceptualEntity that it realizes. The FACE_LogicalEntity's composed Entities realize one to one the FACE_ConceptualEntity's composed Entities; the FACE_LogicalEntity's composed FACE_Measurements realize many to one the FACE_ConceptualEntity's composed FACE_Observables.

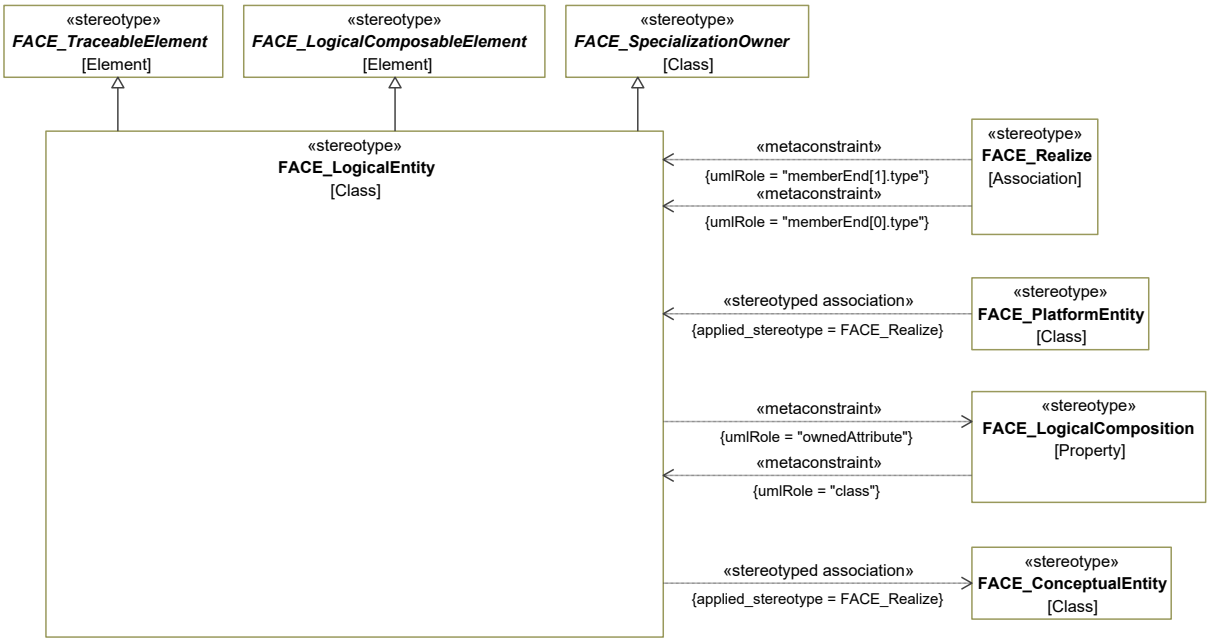

Figure 7-62: FACE_LogicalEntity

Constraints

[1] FACE_LogicalEntity.ownedAttribute	The value for the ownedAttribute metaproperty must be stereotyped «FACE_LogicalComposition» or its specializations

FACE Conformance/OCL Constraints

[1] FACE_LogicalEntity.characteristicsHaveUniqueRolenames	A FACE_LogicalCharacteristic's rolename must be unique within a FACE_LogicalEntity.
[2] FACE_LogicalEntity.compositionsConsistentWithRealization	FACE_LogicalCompositions in a FACE_LogicalEntity must realize FACE_ConceptualCompositions in the conceptual FACE_ConceptualEntity that the FACE_LogicalEntity realizes.
[3] FACE_LogicalEntity.realizedCompositionsHaveDifferentTypes	A FACE_LogicalEntity may not contain two FACE_LogicalCompositions that realize the same FACE_ConceptualComposition unless their types are different FACE_Measurements and their multiplicities are equal.

[4] FACE_LogicalEntity.specializationConsistentWithRealization	If a FACE_LogicalEntity specializes, its specialization must be consistent with its realization's specialization.

FACE_LogicalParticipant

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalCharacteristic
Extension: Class

Description

A FACE_LogicalParticipant is the mechanism that allows a FACE_LogicalAssociation to be constructed between two or more FACE_LogicalEntities. The type of a FACE_LogicalParticipant is the FACE_LogicalEntity being used to construct the FACE_LogicalAssociation. The sourceLowerBound and sourceUpperBound attributes define the multiplicity of the FACE_LogicalAssociation relative to the FACE_LogicalParticipant. A sourceUpperBound multiplicity of -1 represents an unbounded sequence. The path attribute of the FACE_LogicalParticipant describes the chain of entity characteristics to traverse to reach the subject of the association beginning with the entity referenced by the type attribute.

The strings provided in the "path" tagged value are a representation of the full set of FACE Logical CharacteristicPathNode, ParticipantPathNode, and PathNode elements in the path attribute as specified in the Technical Standard for Future Airborne Capability Environment (FACE™), Edition 3.0. The notation used for path string is described in Section 3.6.4.1.1.3 of the Technical Standard for Future Airborne Capability Environment (FACE™), Edition 2.1. The two notations (elements and string) are interchangeable using a translation algorithm. XMI exchange mechanisms between models using the FACE Profile for UAF and the FACE XMI (face) file are required to translate between the two notations.

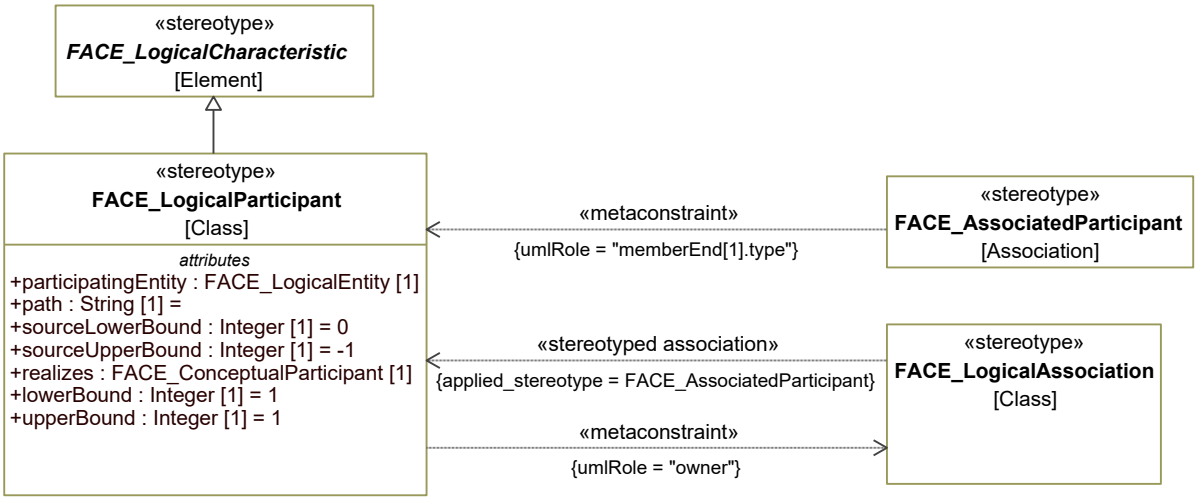

Figure 7-63: FACE_LogicalParticipant

Attributes

lowerBound : Integer [1]

participatingEntity : FACE_LogicalEntity [1]

path : String [1]

realizes : FACE_ConceptualParticipant [1]

sourceLowerBound : Integer [1]

sourceUpperBound : Integer [1]

upperBound : Integer [1]

Constraints

[1] FACE_LogicalParticipant.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_LogicalAssociation»

FACE Conformance/OCL Constraints

[1] FACE_LogicalParticipant.multiplicityConsistentWithRealization	A FACE_LogicalParticipant's multiplicity must be at least as restrictive as the FACE_ConceptualParticipant it realizes.
[2] FACE_LogicalParticipant.multiplicityConsistentWithSpecialization	A FACE_LogicalParticipant's multiplicity must be at least as restrictive as the FACE_LogicalParticipant it specializes.
[3] FACE_LogicalParticipant.rolenameDefined	A FACE_LogicalParticipant must have a rolename, either projected from a characteristic or defined directly on the FACE_LogicalParticipant.
[4] FACE_LogicalParticipant.typeConsistentWithRealization	If FACE_LogicalParticipant "A" realizes FACE_ConceptualParticipant "B", then A's type must realize B's type, and A's PathNode sequence must "realize" B's PathNode sequence. (A PathNode sequence "A" "realizes" a sequence "B" if the projected element of each PathNode in A realizes the projected element of the corresponding PathNode in B.)

FACE_LogicalQuery

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalView
Extension: Class

Description

A FACE_LogicalQuery is a specification that defines the content of FACE_LogicalView as a set of FACE_LogicalCharacteristics projected from a selected set of related FACE_LogicalEntities. The specification attribute captures the specification of a Query as defined by the Query grammar.

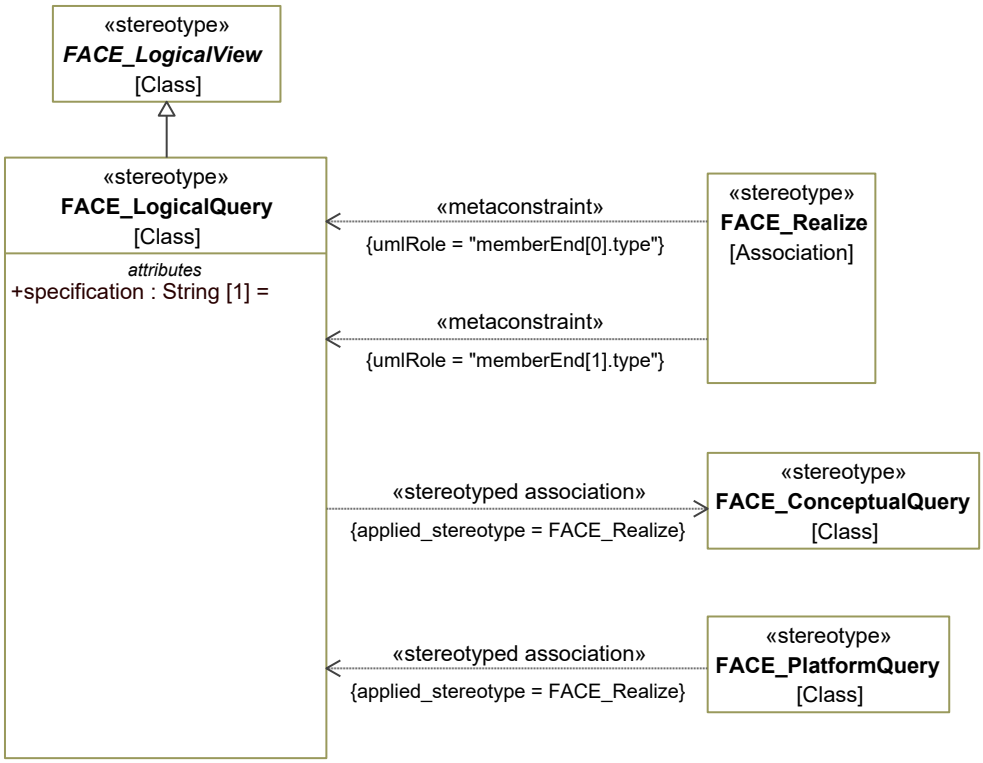

Figure 7-64: FACE_LogicalQuery

Attributes

specification : String [1]

FACE_LogicalQueryComposition

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_ModelElement
Extension: Property

Description

A FACE_LogicalQueryComposition is the mechanism that allows a FACE_LogicalCompositeQuery to be constructed from FACE_LogicalQueries and other FACE_LogicalCompositeQueries. The rolename attribute defines the name of the composed FACE_LogicalView within the scope of the composing FACE_LogicalCompositeQuery. The type of a FACE_LogicalQueryComposition is the FACE_LogicalView being used to construct the FACE_LogicalCompositeQuery.

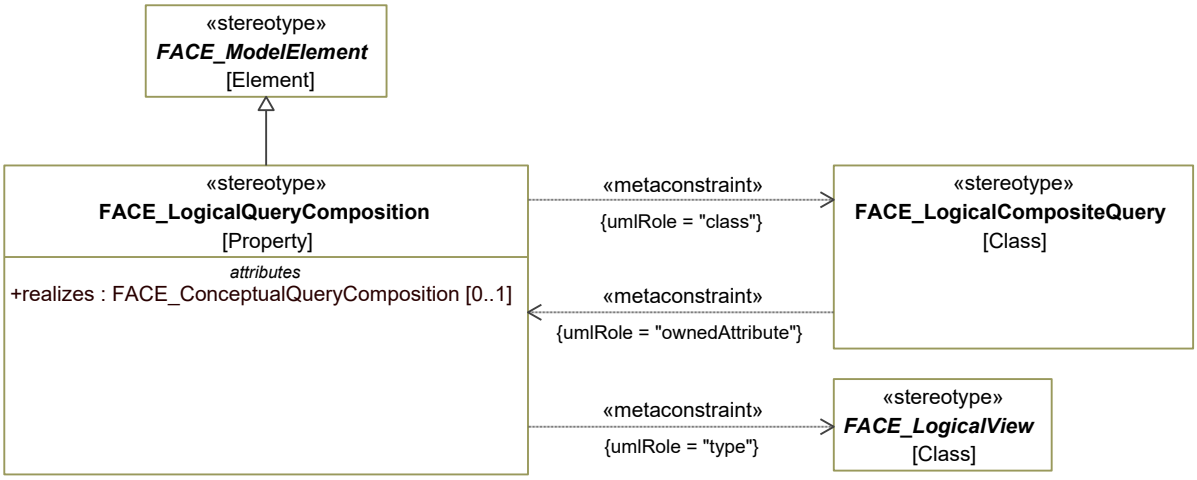

Figure 7-65: FACE_LogicalQueryComposition

Attributes

realizes : FACE_ConceptualQueryComposition [0..1]

Constraints

[1] FACE_LogicalQueryComposition.class	Value for the class metaproperty must be stereotyped «FACE_LogicalCompositeQuery».
[2] FACE_LogicalQueryComposition.type	Value for the type metaproperty must be stereotyped «FACE_LogicalView» or its specializations.

FACE Conformance/OCL Constraints

[1] FACE_LogicalQueryComposition.rolenameIsValidIdentifier	The rolename of a FACE_LogicalQueryComposition must be a valid identifier.
[2] FACE_LogicalQueryComposition.typeConsistentWithRealization	If FACE_LogicalQueryComposition "A" realizes FACE_ConceptualQueryComposition "B", then A's type must realize B's type.

FACE_LogicalValueType

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalElement
Extension: Class

Description

A ValueType specifies the logical representation of a MeasurementSystem or Measurement. Integer, Real, and String are examples of logical ValueTypes. This element is the representation for all of the logical data type elements listed in the FACE Technical Standard, Edition 3.0.

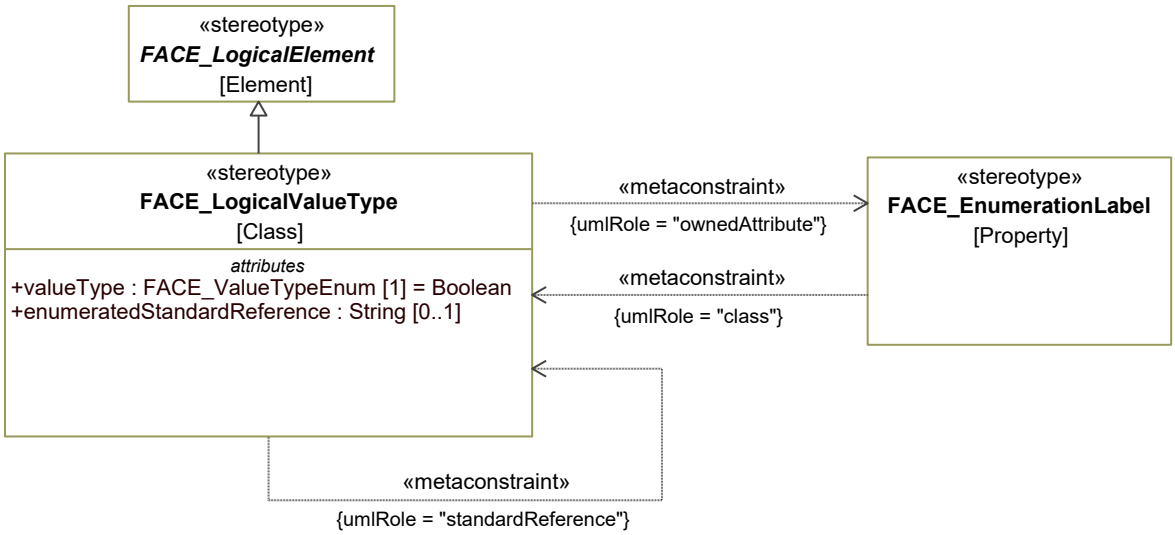

Figure 7-66: FACE_LogicalValueType

Attributes

enumeratedStandardReference : String [0..1]

valueType : FACE_ValueTypeEnum [1]

Constraints

[1] FACE_LogicalValueType.ownedAttribute	If the valueType is NOT Enumerated, no ownedAttributes are allowed. If the valueType is Enumerated, all ownedAttributes must be stereotyped by «FACE_EnumerationLabel».
[2] FACE_LogicalValueType.standardReference	standardReference may only have a value if valueType = Enumerated

FACE Conformance/OCL Constraints

[1] FACE_LogicalValueType.enumerationLabelNameUnique	If the value type is FACE_Enumeration, all contained FACE_EnumerationLabels must have unique names.
[2] FACE_LogicalValueType.enumNameIsNotReservedWord	If the value type is Enumerated, ensure that the Enumerated's name is not an IDL reserved word.
[3] FACE_LogicalValueType.nameOfValueTypeMatchesNameOfMetaclass	A FACE_LogicalValueType must be named the same as its metatype. (e.g., a String must be named "String")
[4] FACE_LogicalValueType.nonEmptyDescription	FACE_LogicalValueType must have a non-empty description.

FACE_LogicalView

Package: LogicalDataModel
isAbstract: Yes
Generalization: FACE_LogicalElement, FACE_TraceableElement
Extension: Class

Description

A FACE_LogicalView is a FACE_LogicalQuery or a FACE_LogicalCompositeQuery.

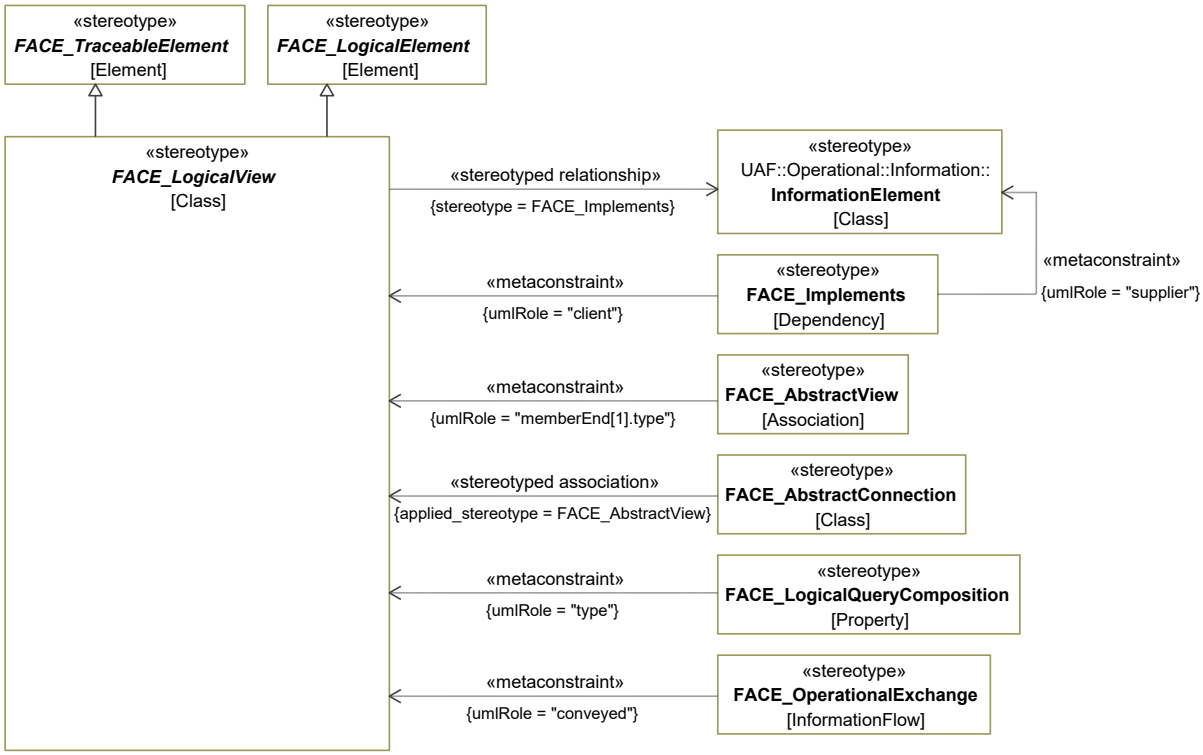

Figure 7-67: abstract FACE_LogicalView

FACE_Measurement

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_AbstractMeasurement, FACE_LogicalComposableElement
Extension: Class

Description

A FACE_Measurement realizes a FACE_Observable as a set of quantities that can be recorded for each of the axis of a FACE_MeasurementSystem. A FACE_Measurement contains the specific implementation details optionally including an override of the default Unit for each axis as well as the constraints over that space for which the FACE_MeasurementSystem is valid.

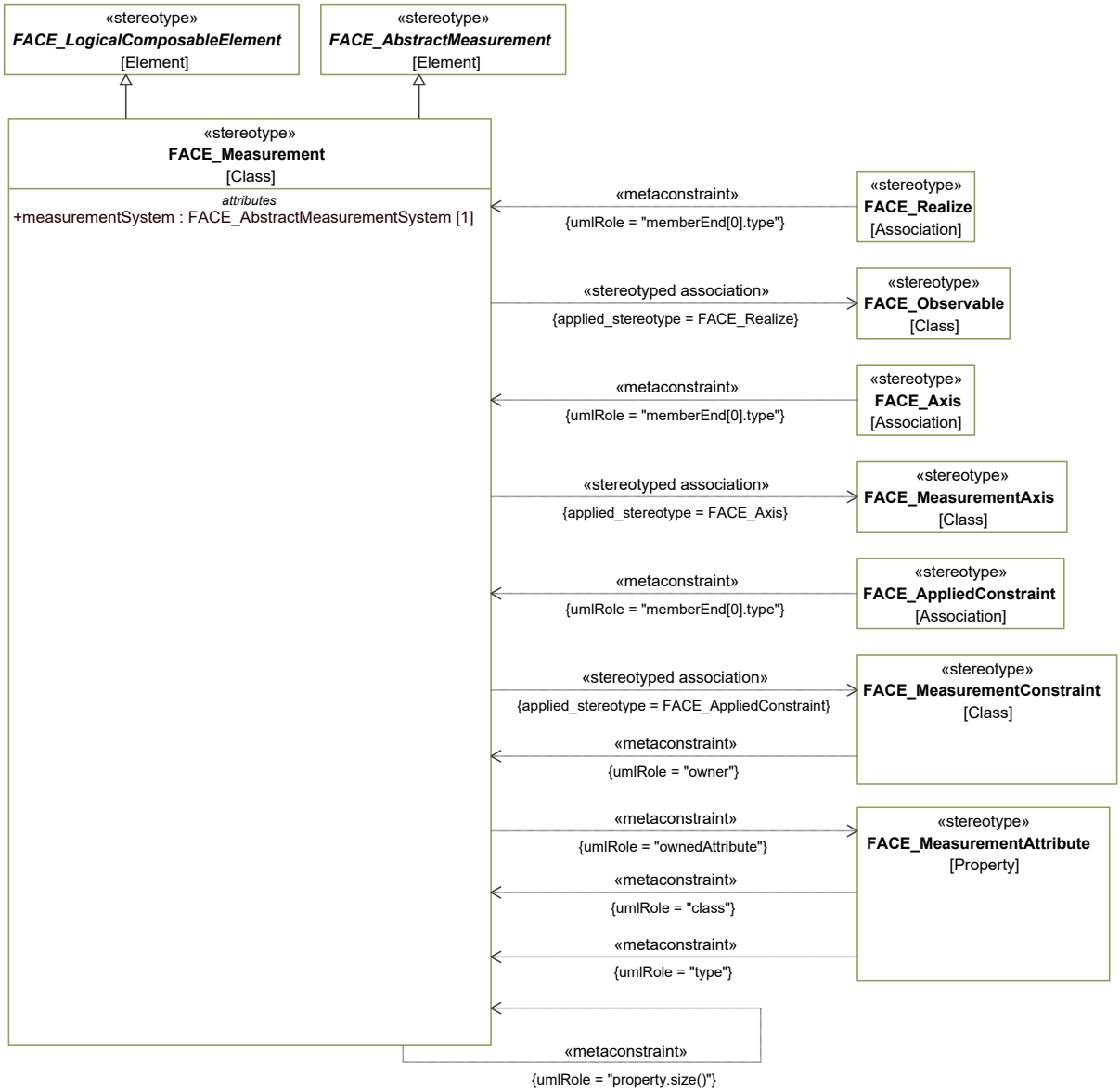

Figure 7-68: FACE_Measurement

Attributes

measurementSystem : FACE_AbstractMeasurementSystem [1]

Constraints

[1] FACE_Measurement.ownedAttribute	The values for the ownedAttribute metaproperty must meet the following criteria: - referenced elements must be stereotyped «FACE_MeasurementAttribute» or its specializations - must contain 2 or more elements

FACE Conformance/OCL Constraints

[1] FACE_Measurement.enumeratedMeasurementUsesEnumeratedMeasurementSystem	A Measurement that uses an Enumerated ValueType in any of its axes must be based on the 'AbstractDiscreteSetMeasurementSystem' MeasurementSystem.
[2] FACE_Measurement.measurementAttributesHaveUniqueRolenames	A FACE_Measurement's attributes must have unique rolenames.
[3] FACE_Measurement.measurementConsistentWithMeasurementSystem	If a FACE_Measurement "A" is based on FACE_MeasurementSystem "B", then A and B must have the same number of axes, and every FACE_MeasurementAxis in A must be based on a unique FACE_MeasurementSystemAxis in B. If a FACE_Measurement is based on a FACE_StandardMeasurementSystem, then it must have no axes.
[4] FACE_Measurement.noCyclesInMeasurements	A FACE_Measurement may not use itself as a FACE_MeasurementAttribute.

FACE_MeasurementAttribute

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_ModelElement
Extension: Property

Description

A FACE_MeasurementAttribute is supplemental data associated with a FACE_Measurement.

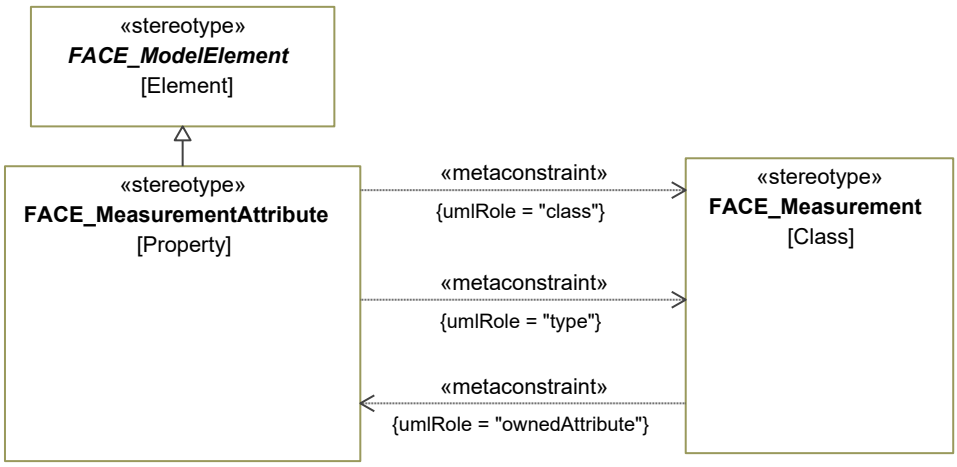

Figure 7-69: FACE_MeasurementAttribute

Constraints

[1] FACE_MeasurementAttribute.class	Value for the class metaproperty must be stereotyped «FACE_Measurement»
[2] FACE_MeasurementAttribute.type	Value for the type metaproperty must be stereotyped «FACE_Measurement»

FACE_MeasurementAxis

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_AbstractMeasurement, FACE_LogicalElement
Extension: Class

Description

A FACE_MeasurementAxis optionally establishes constraints for a FACE_MeasurementSystemAxis and may optionally override its default units and value types.

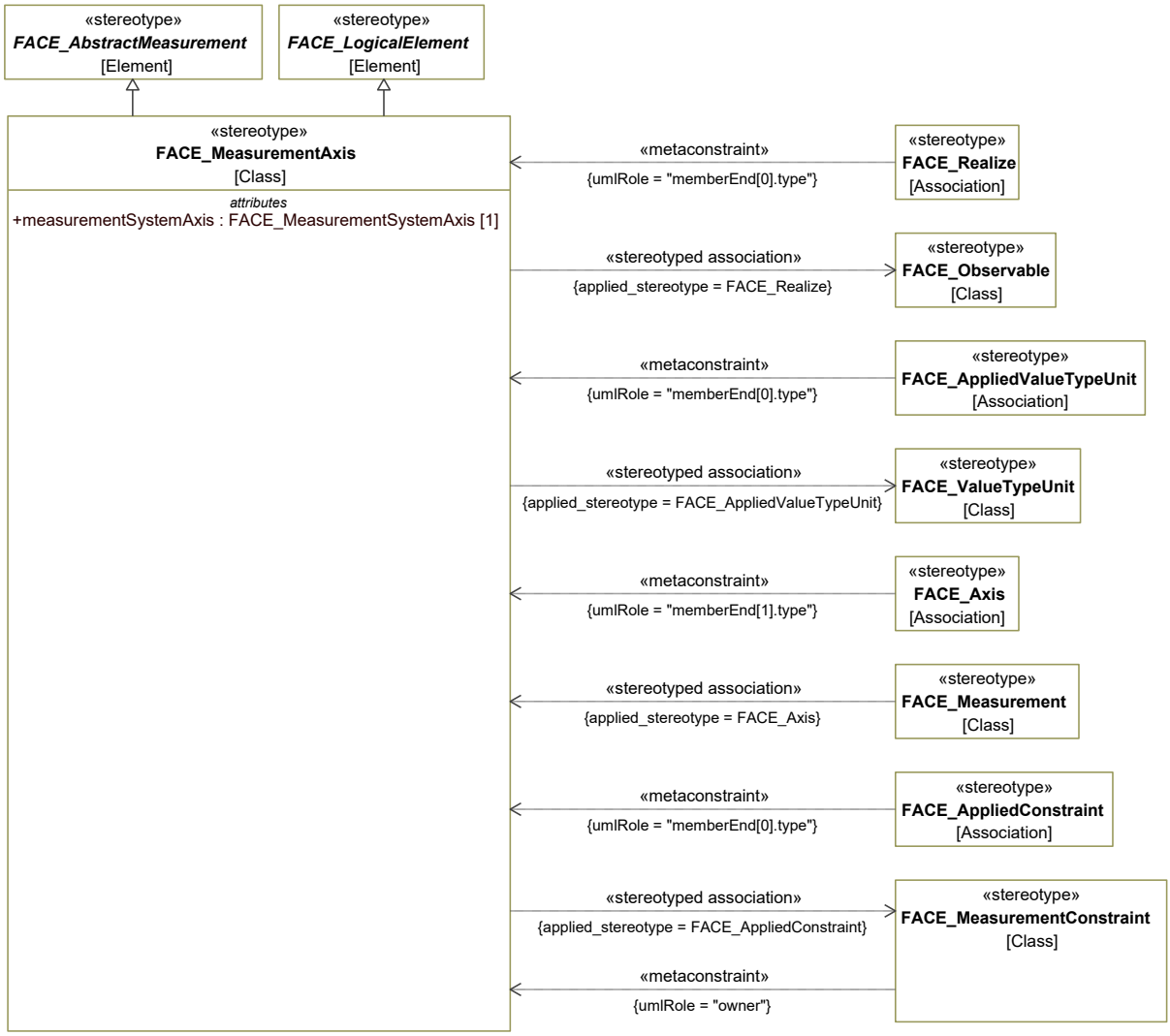

Figure 7-70: FACE_MeasurementAxis

Attributes

measurementSystemAxis : FACE_MeasurementSystemAxis [1]

FACE_MeasurementConstraint

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_ModelElement
Extension: Class

Description

A FACE_MeasurementConstraint describes the constraints over the axes of a given FACE_MeasurementSystem or FACE_Measurement or over the value types of a FACE_MeasurementSystemAxis or FACE_MeasurementAxis. The constraints are described in the constraintText attribute. The specific format of constraintText is undefined.

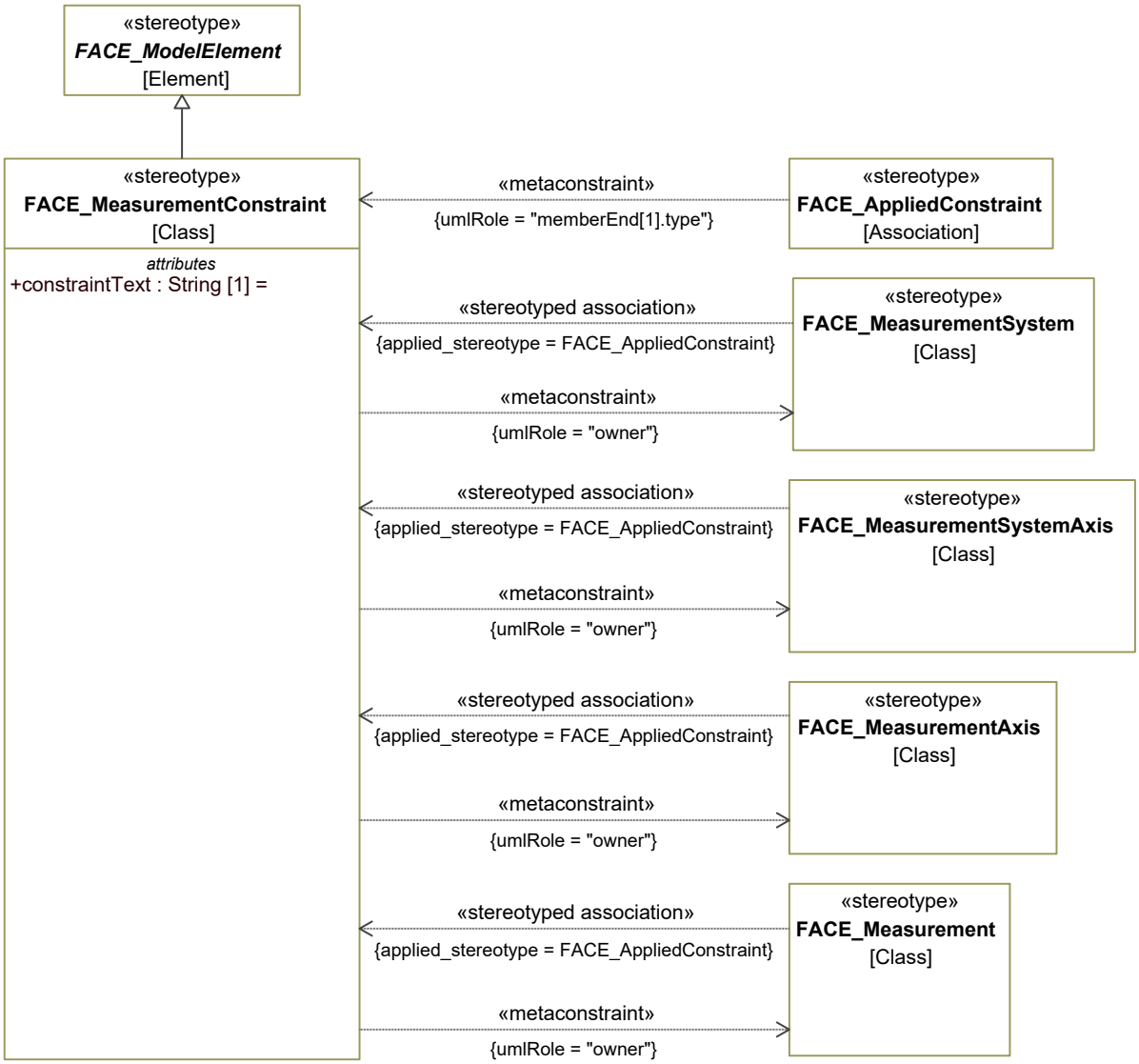

Figure 7-71: FACE_MeasurementConstraint

Attributes

constraintText : String [1]

Constraints

[1] FACE_MeasurementConstraint.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_MeasurementSystem» «FACE_MeasurementSystemAxis» «FACE_MeasurementAxis» «FACE_Measurement»

FACE_MeasurementConversion

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalElement
Extension: Class

Description

A FACE_MeasurementConversion is a relationship between two FACE_Measurements that describes how to transform measured quantities between those FACE_Measurements. The conversion is captured as a set of equations in the equation attribute. The specific format of equation is undefined. The loss introduced by the conversion equations is captured in the conversionLossDescription attribute. The specific format of conversionLossDescription is undefined.

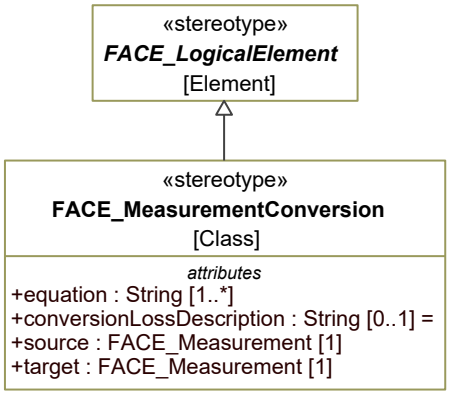

Figure 7-72: FACE_MeasurementConversion

Attributes

conversionLossDescription : String [0..1]

equation : String [1..*]

source : FACE_Measurement [1]

target : FACE_Measurement [1]

FACE_MeasurementSystem

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_AbstractMeasurementSystem

Description

A FACE_MeasurementSystem relates a FACE_CoordinateSystem to an origin and orientation for the purpose of establishing a common basis for describing points in an N-dimensional space. Defining a FACE_MeasurementSystem establishes additional properties of the FACE_CoordinateSystem including units and value types for each axis, and a set of reference points that can be used to establish an origin and indicate the direction of each axis.

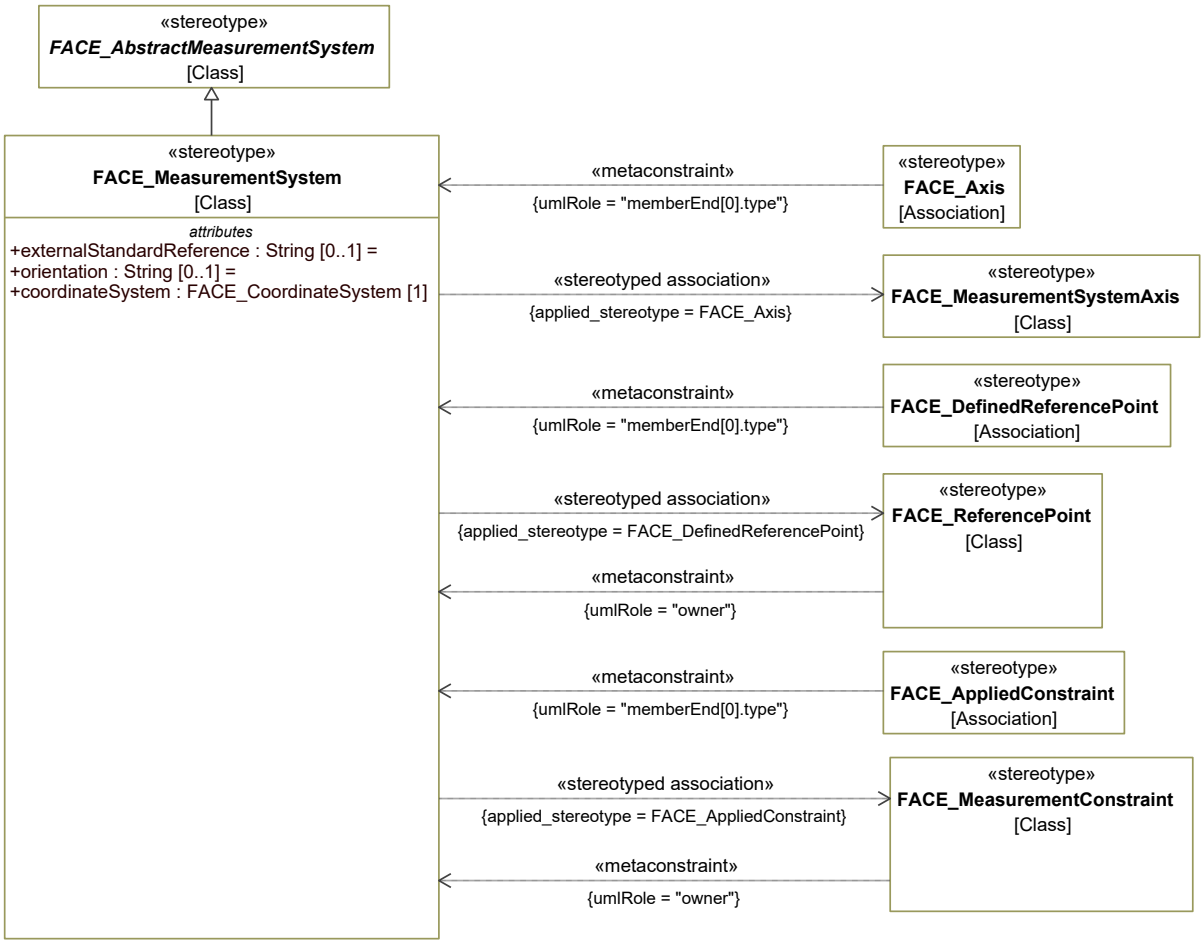

Figure 7-73: FACE_MeasurementSystem

Attributes

coordinateSystem : FACE_CoordinateSystem [1]

externalStandardReference : String [0..1]

orientation : String [0..1]

FACE Conformance/OCL Constraints

[1] FACE_MeasurementSystem.hasSufficientReferencePoints	If a FACE_MeasurementSystem has FACE_ReferencePoints, then it must have at least as many FACE_ReferencePoints as it has axes.
[2] FACE_MeasurementSystem.measurementSystemConsistentWithCoordinateSystem	If a FACE_MeasurementSystem "A" is based on FACE_CoordinateSystem "B", then A and B must have the same number of axes, and every FACE_MeasurementSystemAxis in A

	must be based on a unique FACE_CoordinateSystemAxis in B.
[3] FACE_MeasurementSystem.nonEmptyDescription	FACE_MeasurementSystem must have a non-empty description.
[4] FACE_MeasurementSystem.onlyOneEnumeratedMeasurementSystem	Enumerated FACE_LogicalValueTypes are expressed as FACE_MeasurementSystemAxis in a FACE_MeasurementSystem. The name of a FACE_MeasurementSystem expressing an Enumerated is expected to be "AbstractDiscreteSetMeasurementSystem", and this special FACE_MeasurementSystem must have only one FACE_Axis.
[5] FACE_MeasurementSystem.referencePointPartsConsistentWithAxes	A FACE_ReferencePoint in a FACE_MeasurementSystem contains FACE_ReferencePointParts. The FACE_ReferencePointParts must use the same FACE_MeasurementSystemAxes used by the owning FACE_MeasurementSystem.
[6] FACE_MeasurementSystem.referencePointPartsCoverAllAxes	In a FACE_MeasurementSystem, each FACE_ReferencePoints' parts must use the same set of VTUs as the FACE_MeasurementSystem's axes.

FACE_MeasurementSystemAxis

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalElement
Extension: Class

Description

A FACE_MeasurementSystemAxis establishes additional properties for a FACE_CoordinateSystemAxis including units and value types.

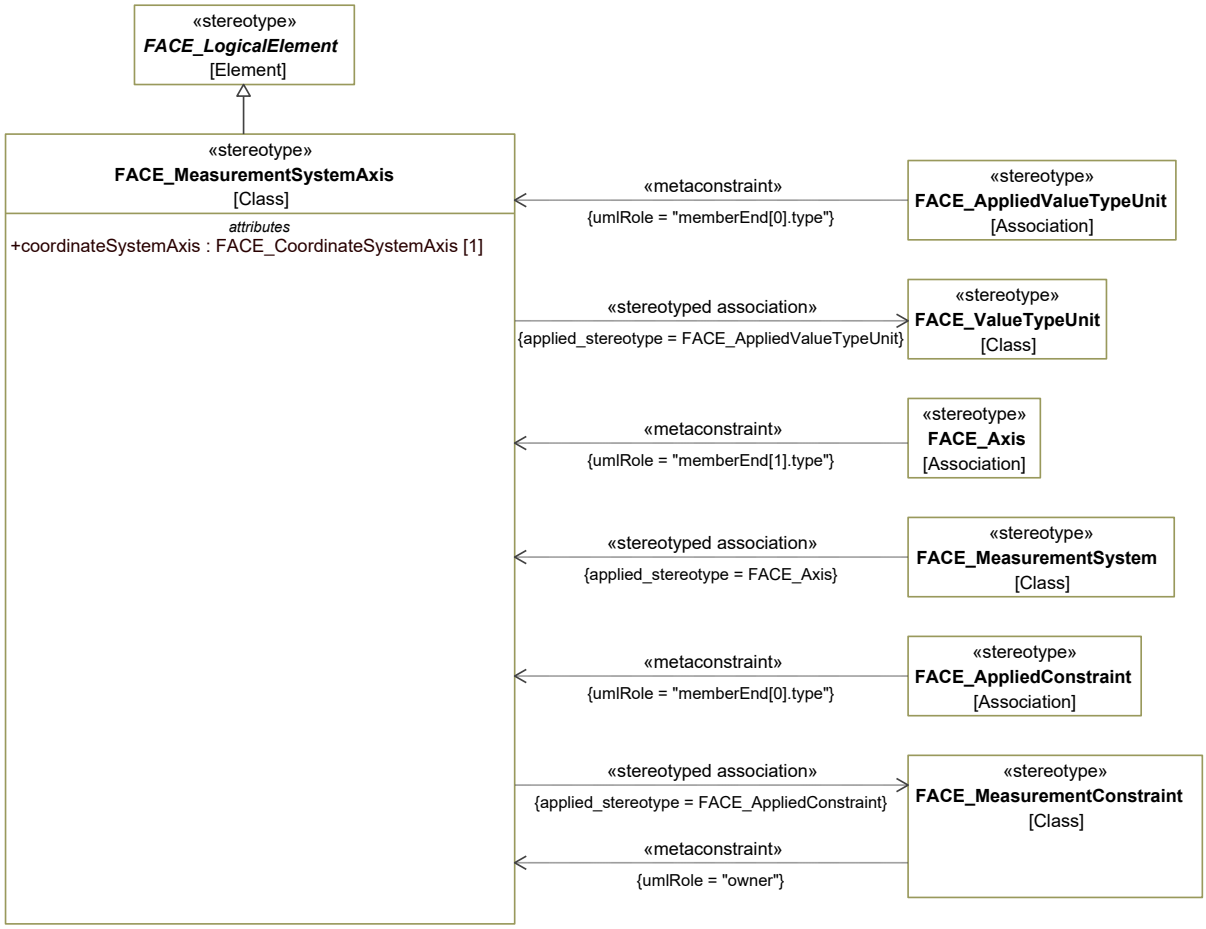

Figure 7-74: FACE_MeasurementSystemAxis

Attributes

coordinateSystemAxis : FACE_CoordinateSystemAxis [1]

FACE Conformance/OCL Constraints

[1] FACE_MeasurementSystemAxis.nonEmptyDescription	FACE_MeasurementSystem must have a non-empty description.

FACE_MeasurementSystemConversion

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_LogicalElement
Extension: Class

Description

A FACE_MeasurementSystemConversion is a relationship between two FACE_MeasurementSystems that describes how to transform measured quantities between those FACE_MeasurementSystems. The conversion is captured as a set of equations

in the equation attribute. The specific format of equation is undefined. The loss introduced by the conversion equations is captured in the conversionLossDescription attribute. The specific format of conversionLossDescription is undefined.

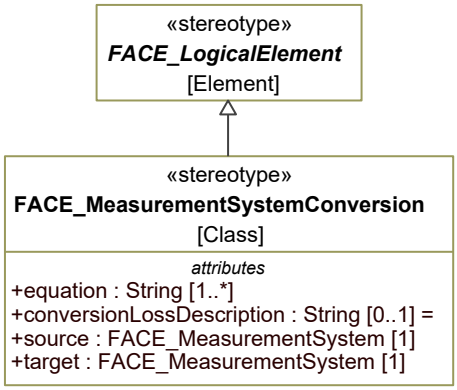

Figure 7-75: FACE_MeasurementSystemConversion

Attributes

conversionLossDescription : String [0..1]

equation : String [1..*]

source : FACE_MeasurementSystem [1]

target : FACE_MeasurementSystem [1]

FACE Conformance/OCL Constraints

[1] FACE_MeasurementSystemConversion.nonEmptyDescription	FACE_MeasurementSystemConversion must have a non-empty description.

FACE_RealConstraint

Package: LogicalDataModel
isAbstract: Yes
Generalization: FACE_Constraint

Description

A FACE_RealConstraint specifies a defined set of meaningful values for a Real or NonNegativeReal.

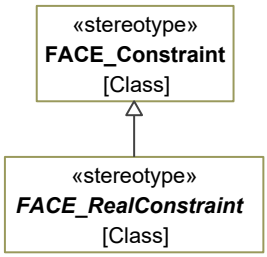

Figure 7-76: abstract FACE_RealConstraint

FACE_RealRangeConstraint

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_RealConstraint

Description

A FACE_RealRangeConstraint specifies a defined range of meaningful values for a Real or NonNegativeReal. The upperBound is greater than or equal to the lowerBound.

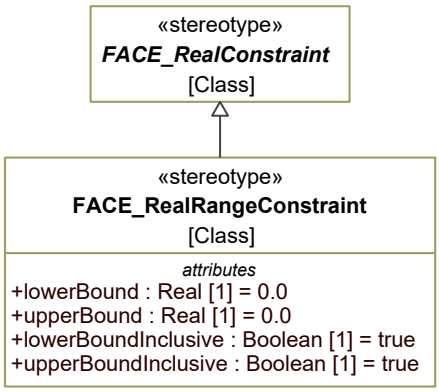

Figure 7-77: FACE_RealRangeConstraint

Attributes

lowerBound : Real [1]

lowerBoundInclusive : Boolean [1]

upperBound : Real [1]

upperBoundInclusive : Boolean [1]

FACE_ReferencePoint

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_Element
Extension: Class

Description

A FACE_ReferencePoint is an identifiable point (landmark) that can be used to provide a basis for locating and/or orienting a MeasurementSystem.

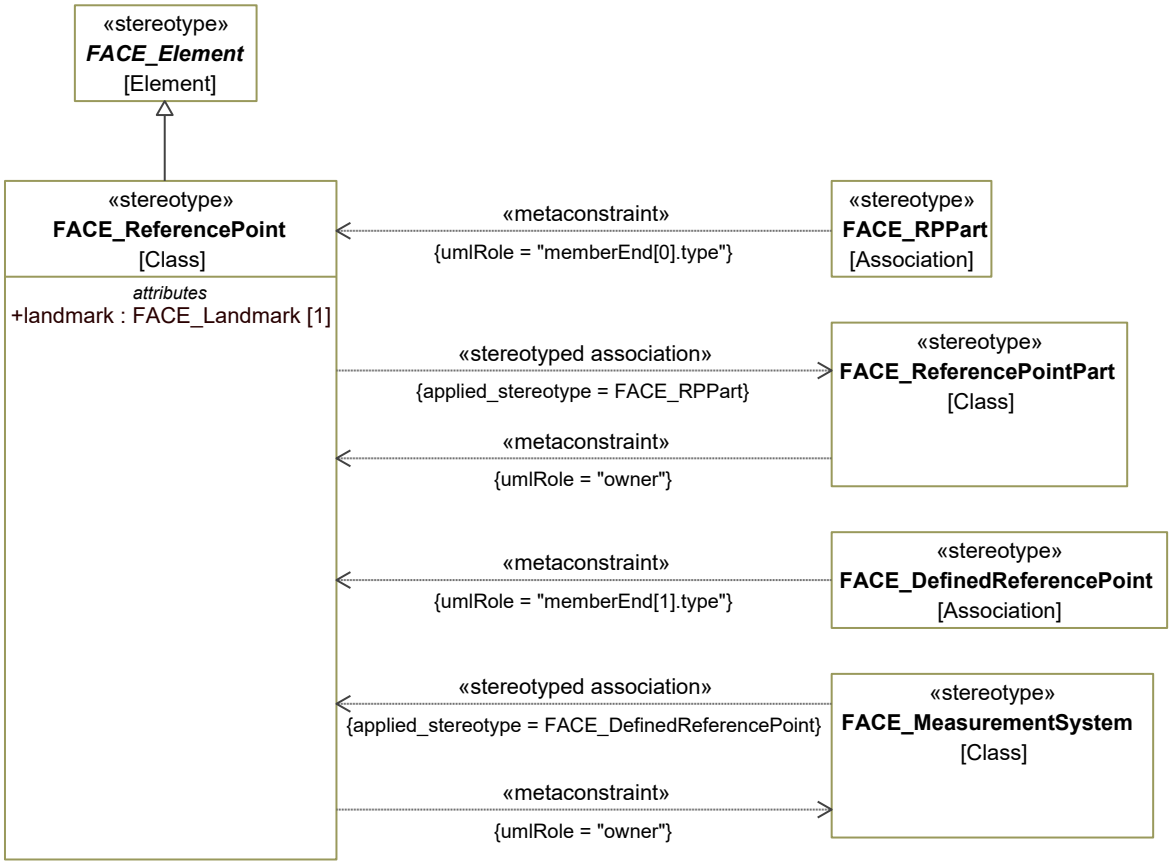

Figure 7-78: FACE_ReferencePoint

Attributes

landmark : FACE_Landmark [1]

Constraints

[1] FACE_ReferencePoint.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_MeasurementSystem»

FACE Conformance/OCL Constraints

[1] FACE_ReferencePoint.nonEmptyDescription	FACE_ReferencePoint must have a non-empty description.

FACE_ReferencePointPart

Package: LogicalDataModel
isAbstract: No

Generalization: FACE_ModelElement
Extension: Class

Description

A FACE_ReferencePointPart is a value for one FACE_ValueTypeUnit in a FACE_ValueTypeUnit set that is used to identify a specific point along an axis.

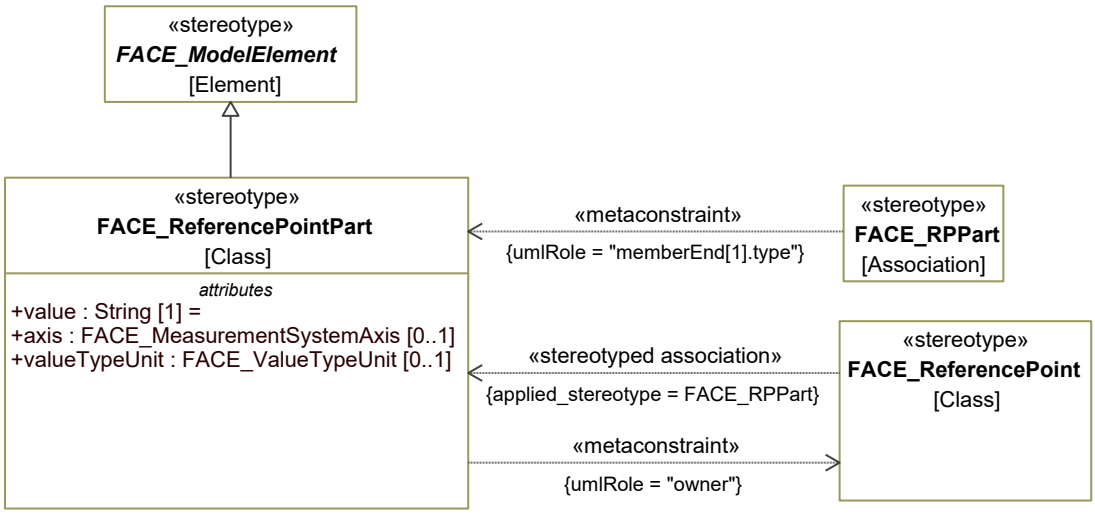

Figure 7-79: FACE_ReferencePointPart

Attributes

axis : FACE_MeasurementSystemAxis [0..1]

value : String [1]

valueTypeUnit : FACE_ValueTypeUnit [0..1]

Constraints

[1] FACE_ReferencePointPart.owner	This element may only be contained in (owned by) elements with the stereotype «FACE_ReferencePoint»

FACE Conformance/OCL Constraints

[1] FACE_ReferencePointPart.noAmbiguousVTUReference	If two FACE_ReferencePointParts in a FACE_ReferencePoint refer to the same FACE VTU, they must refer to distinct (non-null) axes.

FACE_RegularExpressionConstraint

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_StringConstraint

Description

A FACE_RegularExpressionConstraint specifies a defined set of meaningful values for a String in the form of a regular expression.

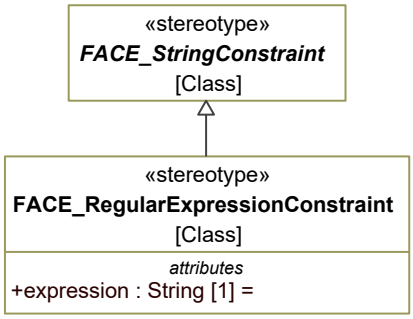

Figure 7-80: FACE_RegularExpressionConstraint

Attributes

expression : String [1]

FACE_RPPart

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to connect the parts of a FACE_ReferencePoint to the owning FACE_ReferencePoint.

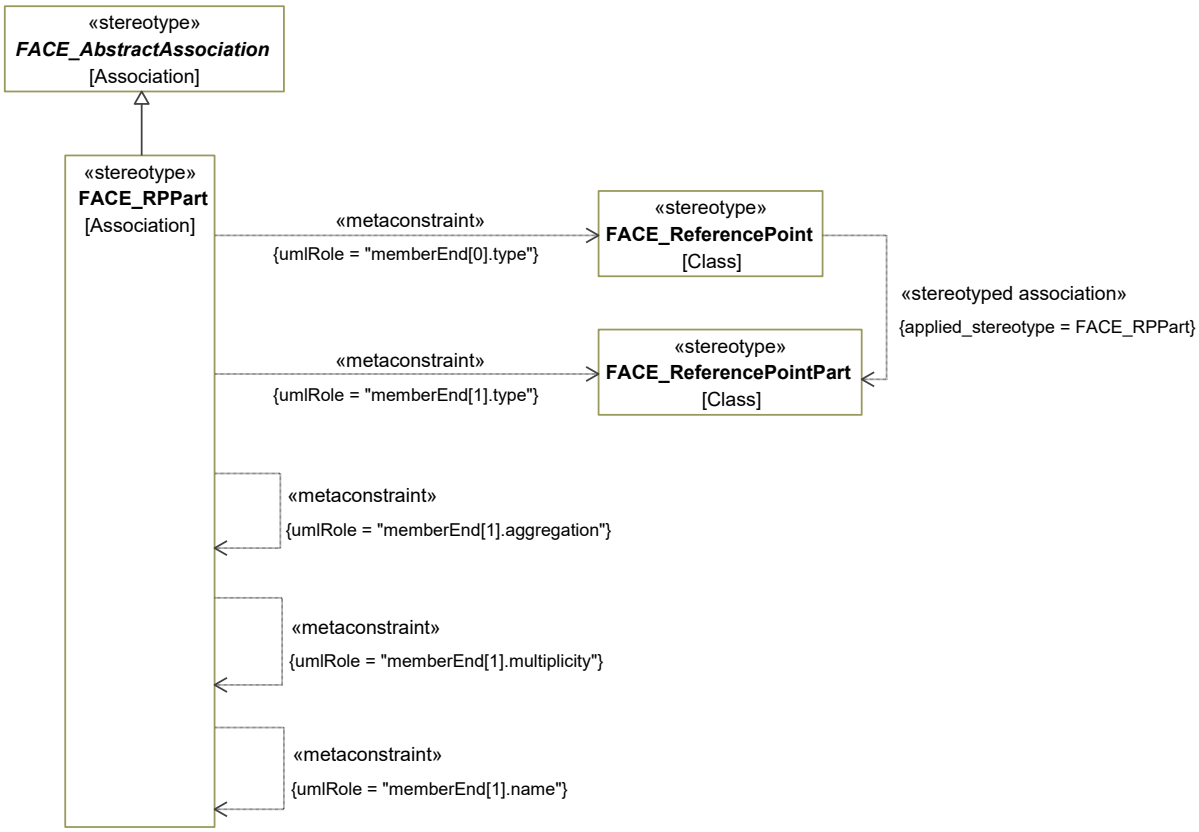

Figure 7-81: FACE_RPPart

Constraints

[1] FACE_RPPart.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_ReferencePoint».
[2] FACE_RPPart.memberEnd[1].aggregation	composite
[3] FACE_RPPart.memberEnd[1].multiplicity	1..*
[4] FACE_RPPart.memberEnd[1].name	"referencePointPart"
[5] FACE_RPPart.memberEnd[1].type	The value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_ReferencePointPart».

FACE_StandardMeasurementSystem

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_AbstractMeasurementSystem

Description

A FACE_StandardMeasurementSystem is used to represent an open, referenced FACE_MeasurementSystem without requiring the detailed modeling of the FACE_MeasurementSystem. The reference should be unambiguous and allows for full comprehension of the underlying FACE_MeasurementSystem.

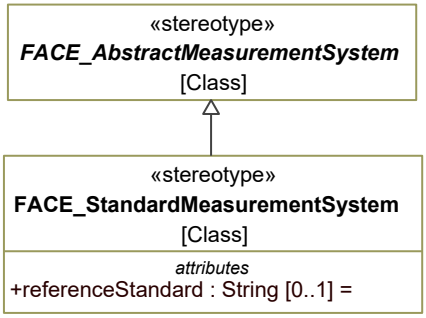

Figure 7-82: FACE_StandardMeasurementSystem

Attributes

referenceStandard : String [0..1]

FACE_StringConstraint

Package: LogicalDataModel
isAbstract: Yes
Generalization: FACE_Constraint

Description

A FACE_StringConstraint specifies a defined set of meaningful values for a String.

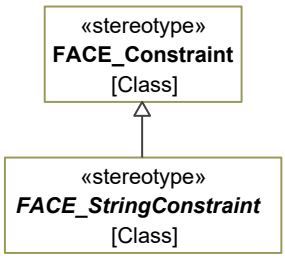

Figure 7-83: abstract FACE_StringConstraint

FACE_Unit

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_ConvertibleElement
Extension: Class

Description

A FACE_Unit is a defined magnitude of quantity used as a standard for measurement.

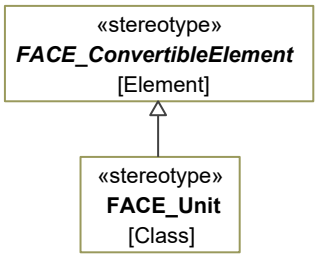

Figure 7-84: FACE_Unit

FACE Conformance/OCL Constraints

[1] FACE_Unit.nonEmptyDescription FACE_Unit must have a non-empty description.

FACE_ValueTypeEnum

Package: LogicalDataModel
isAbstract: No

Description

Indicates the logical data type associated with a property of a FACE element. Its enumeration literals are:

- Boolean -
- Character -
- String -
- Integer -
- Natural -
- Real -
- NonNegativeReal -
- Enumerated -

FACE_ValueTypeUnit

Package: LogicalDataModel
isAbstract: No
Generalization: FACE_AbstractMeasurement, FACE_LogicalElement
Extension: Class

Description

A FACE_ValueTypeUnit defines the logical representation of a FACE_MeasurementSystemAxis or FACE_MeasurementAxis value type in terms of a FACE_Unit and FACE_ValueType pair.

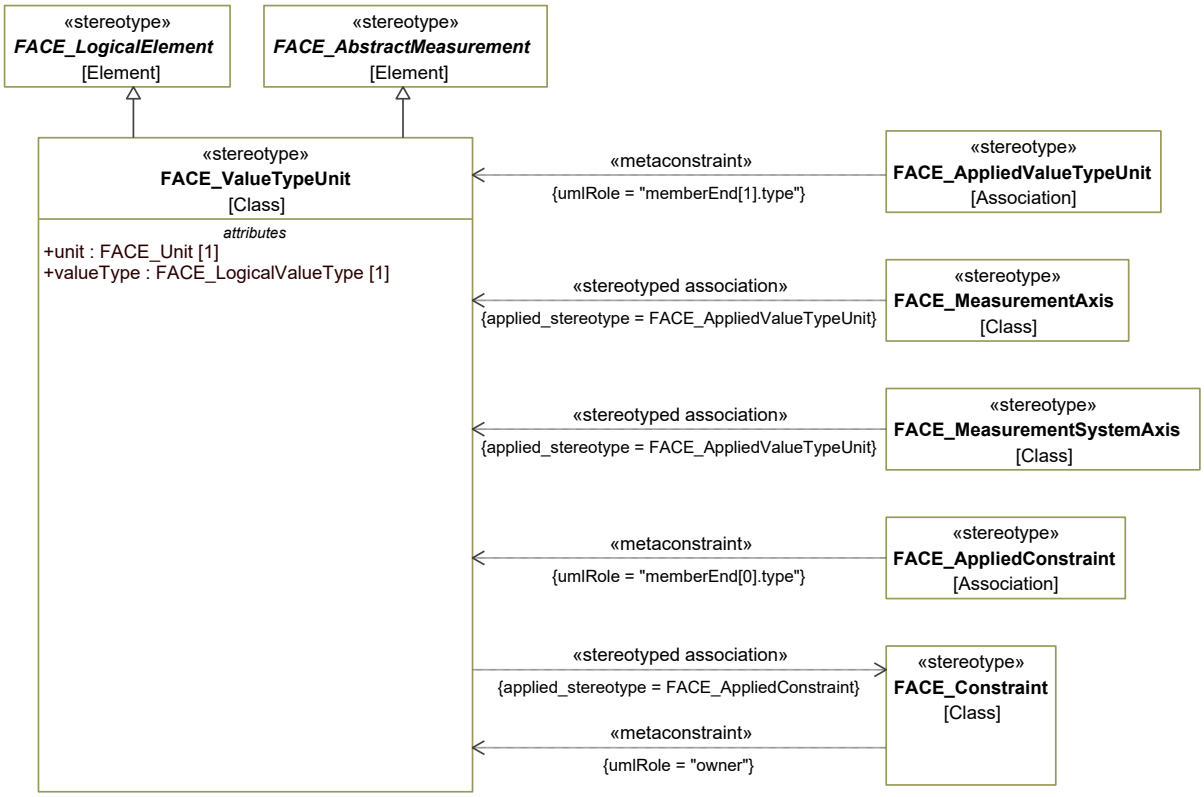

Figure 7-85: FACE_ValueTypeUnit

Attributes

unit : FACE_Unit [1]

valueType : FACE_LogicalValueType [1]

FACE Conformance/OCL Constraints

[1] FACE_ValueTypeUnit.appropriateLabelsForEnumeratedConstraint	If a FACE_ValueTypeUnit "A" contains a FACE_EnumerationConstraint, then A's valueType is a FACE_Enumeration, and the constraint's allowedValues are restricted to FACE_EnumerationLabels from that FACE_Enumeration.

7.1.1.1.3 FACE_UAF_Profile::FACE Data Architecture::FACE Data Model::PlatformDataModel

FACE_Boolean

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLPrimitive

Description

A FACE_Boolean is a data type that represents the values TRUE and FALSE.

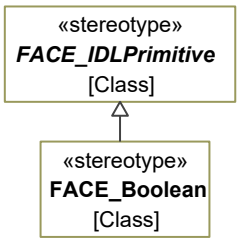

Figure 7-86: FACE_Boolean

FACE_BoundedString

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLBoundedString

Description

A FACE_BoundedString is a data type that represents a variable length sequence of Char (all 8-bit quantities except NULL). The length is a non-negative integer and is available at run-time. The length is maximally bounded.

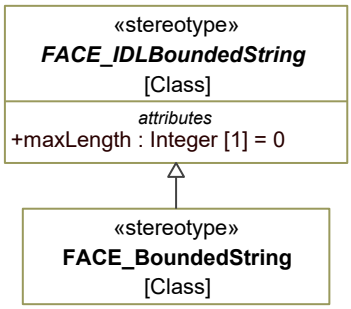

Figure 7-87: FACE_BoundedString

FACE_BoundQuery

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to relate a FACE Template view with the underlying FACE query that is its specification.

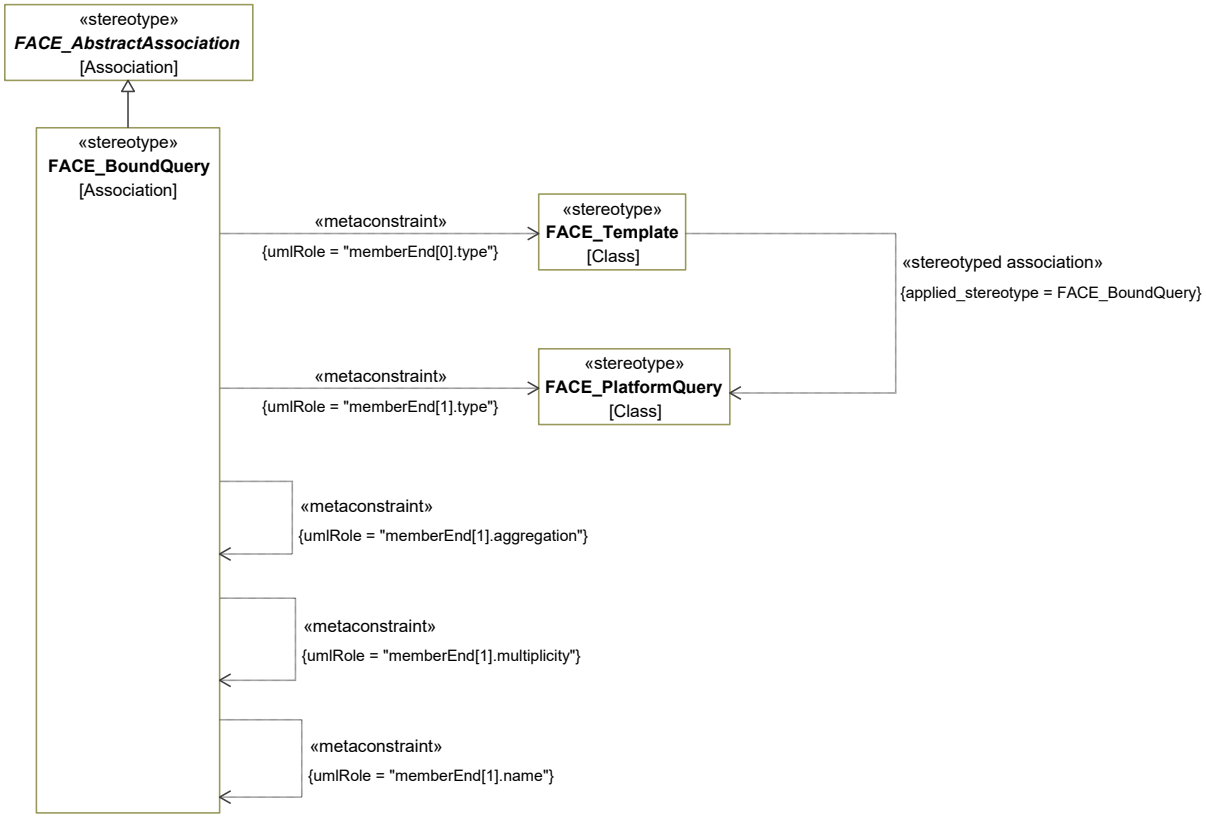

Figure 7-88: FACE_BoundQuery

Constraints

[1] FACE_BoundQuery.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_Template».
[2] FACE_BoundQuery.memberEnd[1].aggregation	none
[3] FACE_BoundQuery.memberEnd[1].multiplicity	0..1
[4] FACE_BoundQuery.memberEnd[1].name	"boundQuery"
[5] FACE_BoundQuery.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_PlatformQuery».

FACE_Char

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_CharType

Description

A FACE_Char is a data type that represents characters from any single byte character set.

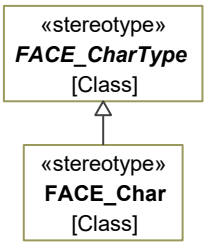

Figure 7-89: FACE_Char

FACE_CharArray

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLCharacterArray

Description

A FACE_CharArray is a data type that represents a fixed length sequence of Char (all 8-bit quantities except NULL). The length is a positive integer and is available at run-time. The length is maximally bounded.

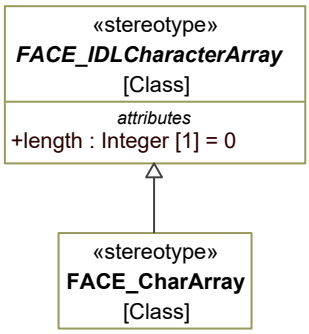

Figure 7-90: FACE_CharArray

FACE_CharType

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_IDLPrimitive

Description

A FACE_CharType is a Char.

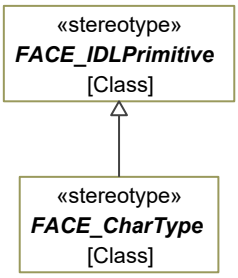

Figure 7-91: abstract FACE_CharType

FACE_CompositeTemplate

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_PlatformView
Extension: Class

Description

A FACE_CompositeTemplate is a collection of two or more FACE_Templates. The isUnion attribute specifies whether the composed Templates are to be represented as cases in a FACE_IDL union or as members of a FACE_IDL struct.

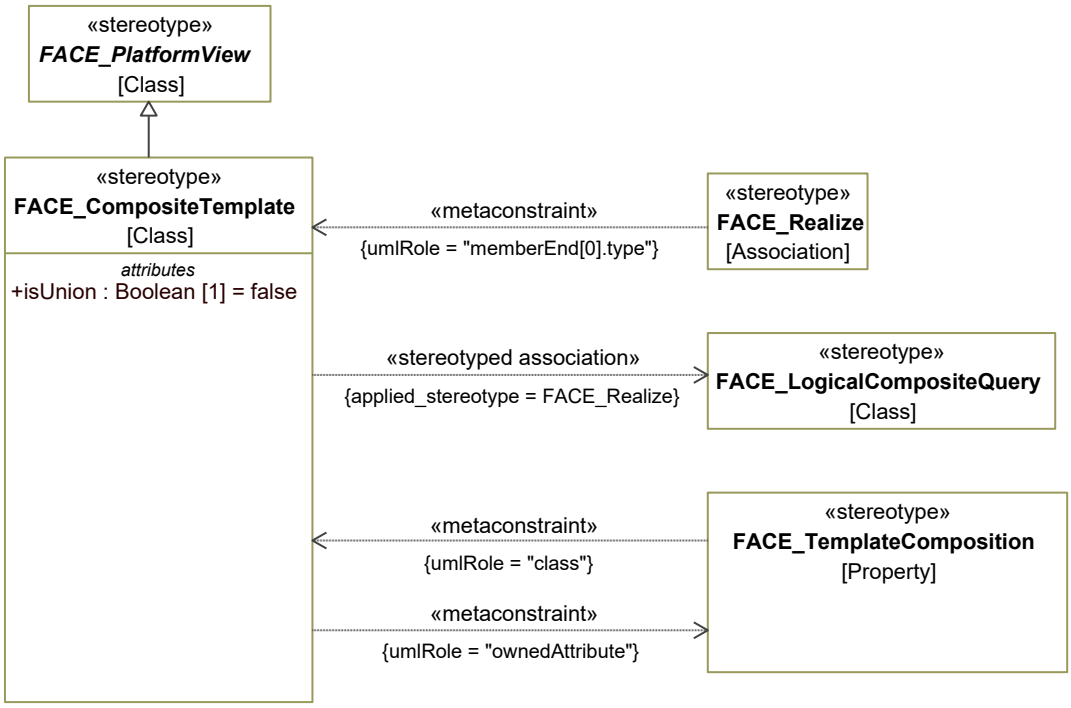

Figure 7-92: FACE_CompositeTemplate

Attributes

isUnion : Boolean [1]

Constraints

[1] FACE_CompositeTemplate.ownedAttribute	The values for the ownedAttribute metaproperty must meet the following criteria: - must be ordered list - referenced elements must be stereotyped «FACE_TemplateComposition» or its specializations - must contain 2 or more elements

FACE Conformance/OCL Constraints

[1] FACE_CompositeTemplate.compositionsConsistentWithRealization	FACE_TemplateCompositions in a platform FACE_CompositeTemplate must realize FACE_QueryCompositions in the FACE_LogicalCompositeQuery that the platform FACE_CompositeTemplate realizes.
[2] FACE_CompositeTemplate.compositionsHaveUniqueRolenames	A FACE_TemplateComposition's rolename must be unique within a FACE_CompositeTemplate.
[3] FACE_CompositeTemplate.noCyclesInConstruction	A FACE_CompositeTemplate must not compose itself.
[4] FACE_CompositeTemplate.realizationUnionConsistent	A FACE_CompositeTemplate that realizes must have the same "isUnion" property as the FACE_CompositeQuery it realizes.
[5] FACE_CompositeTemplate.realizedCompositionsHaveDifferentTypes	A FACE_CompositeTemplate may not contain two FACE_TemplateCompositions that realize the same FACE_QueryComposition.
[6] FACE_CompositeTemplate.viewComposedOnce	A FACE_CompositeTemplate must not compose the same FACE_Template more than once.

FACE_Double

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLReal

Description

A FACE_Double is a real data type that represents an IEEE double precision floating-point number.

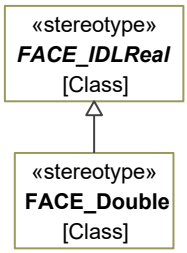

Figure 7-93: FACE_Double

FACE_EffectiveQuery

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

A FACE_EffectiveQuery is a Query that can produce the desired or intended data needed to develop the Platform FACE_Template data structures. Effective Queries are used as an optional notational reference for the modeler to help when a FACE_Template is utilizing other FACE_Templates and the resulting Query may be a complex combination of FACE_BoundQueries.

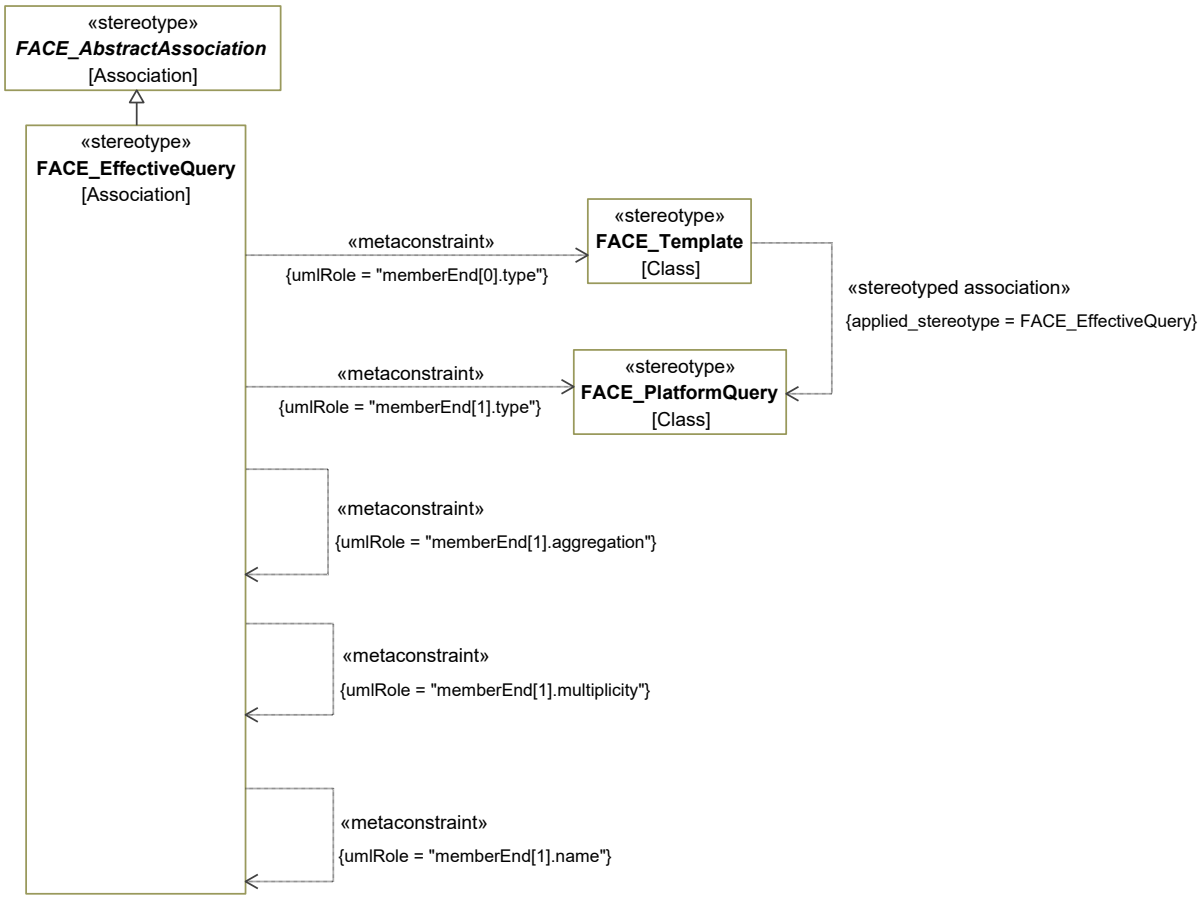

Figure 7-94: FACE_EffectiveQuery

Constraints

[1] FACE_EffectiveQuery.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_Template».
[2] FACE_EffectiveQuery.memberEnd[1].aggregation	none
[3] FACE_EffectiveQuery.memberEnd[1].multiplicity	0..1
[4] FACE_EffectiveQuery.memberEnd[1].name	"effectiveQuery"
[5] FACE_EffectiveQuery.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_PlatformQuery».

FACE_Enumeration

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLPrimitive

Description

A FACE_Enumeration is a data type that represents an ordered list of identifiers. A maximum of 2^32 identifiers may be specified in an enumeration. The order in which the identifiers are named defines the relative order of the identifiers.

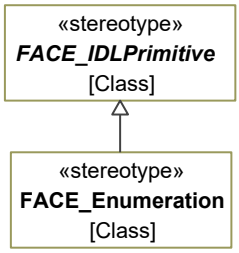

Figure 7-95: FACE_Enumeration

FACE_Fixed

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLReal

Description

A FACE_Fixed is a real data type that represents a fixed-point decimal number of up to 31 significant digits. The digits attribute defines the total number of digits, a non-negative integer value less than or equal to 31. The scale attribute defines the position of the decimal point in the number and cannot be greater than digits.

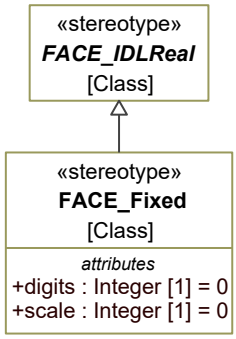

Figure 7-96: FACE_Fixed

Attributes

digits : Integer [1]

scale : Integer [1]

FACE_Float

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLReal

Description

A FACE_Float is a real data type that represents an IEEE single precision floating-point number.

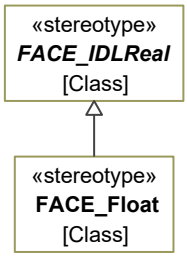

Figure 7-97: FACE_Float

FACE_IDLArray

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLPrimitive

Description

A FACE_IDLArray is used to represent an array of Octets. This can be used to realize a FACE_StandardMeasurementSystem.

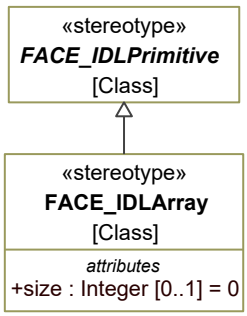

Figure 7-98: FACE_IDLArray

Attributes

size : Integer [0..1]

FACE_IDLBoundedString

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_StringType

Description

A FACE_IDLBoundedString is a BoundedString. The maxLength attribute defines the maximum length of the FACE_IDLBoundedString, an integer value greater than 0.

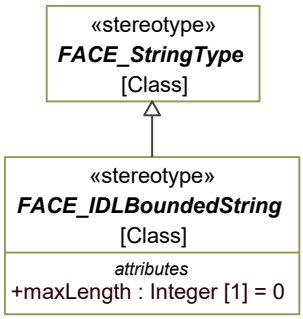

Figure 7-99: abstract FACE_IDLBoundedString

Attributes

maxLength : Integer [1]

FACE_IDLCharacterArray

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_StringType

Description

A FACE_IDLCharacterArray is a CharArray. The "length" attribute defines the length of the FACE_IDLCharacterArray, an integer value greater than 0.

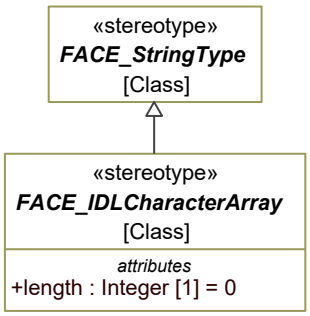

Figure 7-100: abstract FACE_IDLCharacterArray

Attributes

length : Integer [1]

FACE_IDLComposition

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_ModelElement
Extension: Property

Description

A FACE_IDLComposition is the mechanism that allows FACE_IDLStructs to be constructed from other FACE_IDLTypes. The type of a FACE_IDLComposition is the FACE_IDLType being used to construct the FACE_IDLStruct. If type is a FACE_IDLPrimitive, the precision attribute specifies a measure of the detail in which a quantity is captured.

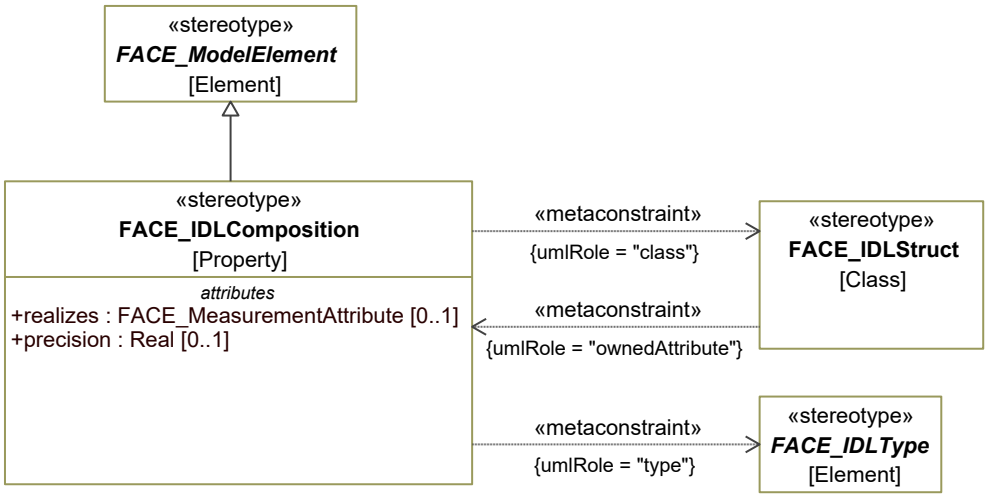

Figure 7-101: FACE_IDLComposition

Attributes

precision : Real [0..1]

realizes : FACE_MeasurementAttribute [0..1]

Constraints

[1] FACE_IDLComposition.class	Value for the class metaproperty must be stereotyped «FACE_IDLStruct»
[2] FACE_IDLComposition.type	Value for the type metaproperty must be stereotyped by a specialization of «FACE_IDLType».

FACE Conformance/OCL Constraints

[1] FACE_IDLComposition.composedIDLNumberHasPrecisionSet	A FACE_IDLComposition whose type is an IDLNumber must have a precision greater than zero.
[2] FACE_IDLComposition.typeConsistentWithRealization	If a FACE_IDLComposition realizes a FACE_MeasurementAttribute, then the FACE_IDLComposition's type must be consistent with its realization's type.

FACE_IDLInteger

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_IDLNumber

Description

A FACE_IDLInteger is an abstract meta-class from which all meta-classes representing whole numbers derive.

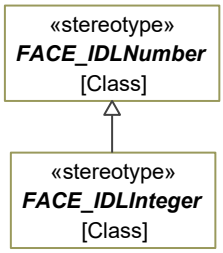

Figure 7-102: abstract FACE_IDLInteger

FACE_IDLNumber

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_IDLPrimitive

Description

A FACE_IDLNumber is an abstract meta-class from which all meta-classes representing numeric values derive.

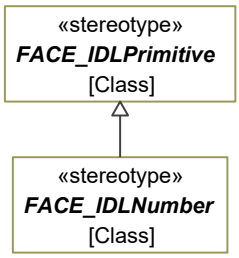

Figure 7-103: abstract FACE_IDLNumber

FACE_IDLPrimitive

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_IDLType
Extension: Class

Description

A FACE_IDLPrimitive is a platform realization of a logical FACE_AbstractMeasurement and represented as a primitive data type supported by the OMG Interface Definition Language (IDL).

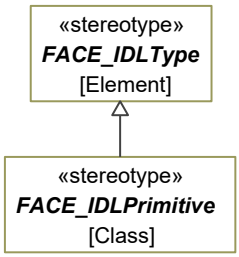

Figure 7-104: abstract FACE_IDLPrimitive

FACE_IDLReal

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_IDLNumber

Description

A FACE_IDLReal is an abstract meta-class from which all meta-classes representing real numbers derive.

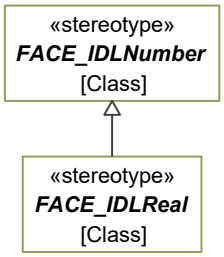

Figure 7-105: abstract FACE_IDLReal

FACE_IDLSequence

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLPrimitive

Description

A FACE_IDLSequence is used to represent a sequence of Octets. This can be used to realize a FACE_StandardMeasurementSystem.

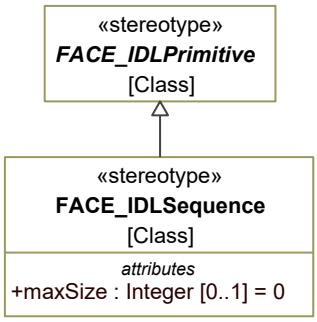

Figure 7-106: FACE_IDLSequence

Attributes

maxSize : Integer [0..1]

FACE_IDLStruct

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLType
Extension: Class

Description

A platform FACE_IDLStruct realizes a logical FACE_AbstractMeasurement in terms of FACE_IDLPrimitives and other FACE_IDLStructs composed of FACE_IDLPrimitives. A platform FACE_IDLStruct's composition hierarchy is consistent with the composition hierarchy of the logical FACE_AbstractMeasurement that it realizes. Each composed platform FACE_IDLType realizes a logical FACE_AbstractMeasurement.

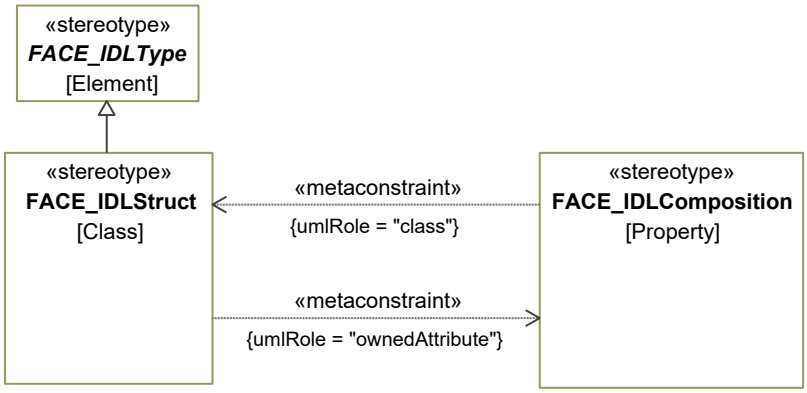

Figure 7-107: FACE_IDLStruct

Constraints

[1] FACE_IDLStruct.ownedAttribute	The values for the ownedAttribute metaproperty must meet the following criteria: - referenced elements must be stereotyped «FACE_IDLComposition» - must contain 2 or more elements

FACE Conformance/OCL Constraints

[1] FACE_IDLStruct.idlCompositionsConsistentlyRealizeMeasurementAttributes	A FACE_Measurement with FACE_MeasurementAttributes is realized by a FACE_IDLStruct with one FACE_IDLComposition per FACE_MeasurementAttribute. (Each FACE_IDLComposition (that realizes) must realize a unique attribute in the FACE_Measurement; every attribute must be realized.)

FACE_IDLType

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_PhysicalDataType

Description

A FACE_IDLType is a FACE_IDLPrimitive or a FACE_IDLStruct.

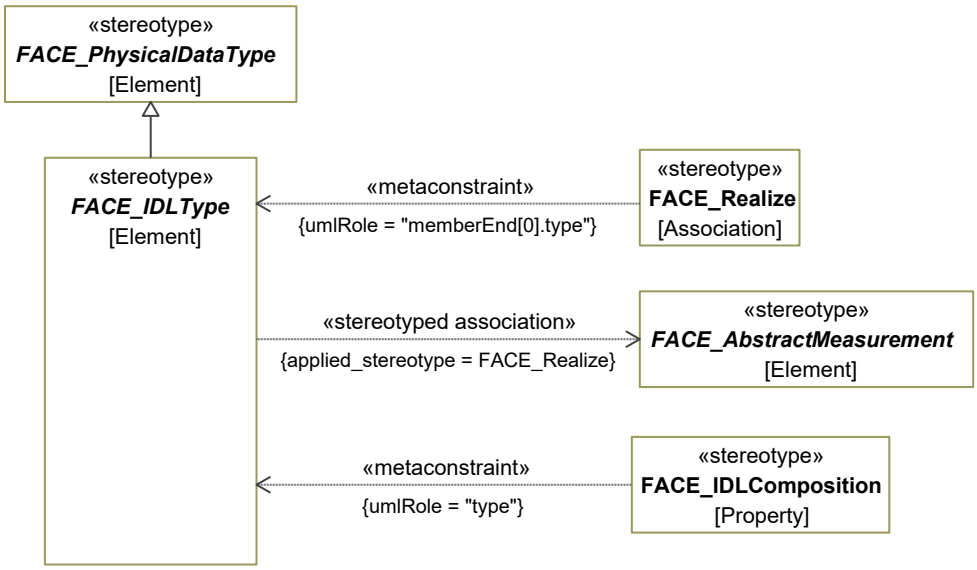

Figure 7-108: abstract FACE_IDLType

FACE Conformance/OCL Constraints

[1] FACE_IDLType.idlCollectionRealizesStandardMeasurement	A FACE_IDLArray or FACE_IDLSequence must realize a FACE_Measurement based on a FACE_StandardMeasurementSystem.
[2] FACE_IDLType.idlTypeConsistentlyRealizesMeasurement	A FACE_Measurement must be realized by a FACE_IDLStruct with one FACE_IDLComposition per FACE_MeasurementAxis. (Each FACE_IDLComposition's type must realize a

	unique axis in the FACE_Measurement; every axis must be realized.) There are two exceptions: - If a FACE_Measurement has one axis with one FACE_ValueTypeUnit (VTU) and no FACE_MeasurementAttributes, it is realized by a FACE_IDLPrimitive. - If a FACE_Measurement has one axis with multiple VTUs and no FACE_MeasurementAttributes, it is realized by a FACE_IDLStruct with one FACE_IDLComposition for each VTU in the axis. (Each FACE_IDLComposition's type must realize a unique VTU in the axis; every VTU must be realized.) Each FACE_IDLComposition's type must be consistent with the type of the VTU it realizes.
[3] FACE_IDLType.idlTypeConsistentlyRealizesMeasurementAxis	If a FACE_MeasurementAxis has one FACE_ValueTypeUnit (VTU), then it must be realized by a FACE_IDLPrimitive; if it has multiple VTUs, then it must be realized by a FACE_IDLStruct with one FACE_IDLComposition per VTU. If FACE_IDLStruct "A" realizes FACE_MeasurementAxis "B", then A must have the same number of FACE_IDLCompositions as B has VTUs, and every FACE_IDLComposition in A must realize a unique VTU in V.
[4] FACE_IDLType.vtuRealizedByIDLPrimitive	FACE_IDLTypes that realize FACE_ValueTypeUnits are FACE_IDLPrimitives.

FACE_IDLUnboundedString

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_StringType

Description

A FACE_IDLUnboundedString is a String.

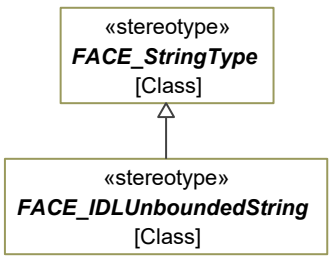

Figure 7-109: abstract FACE_IDLUnboundedString

FACE_IDLUnsignedInteger

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_IDLInteger

Description

A FACE_IDLUnsignedInteger is an abstract meta-class from which all meta-classes representing unsigned whole numbers derive.

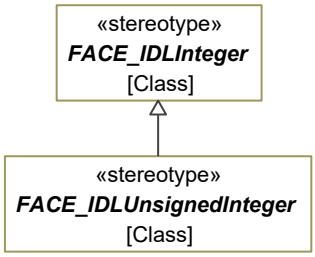

Figure 7-110: abstract FACE_IDLUnsignedInteger

FACE_Long

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLInteger

Description

A FACE_Long is an integer data type that represents integer values in the range -2^31 to (2^31 - 1).

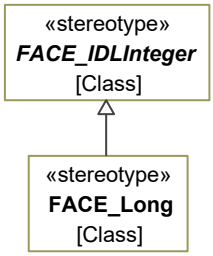

Figure 7-111: FACE_Long

FACE_LongDouble

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLReal

Description

A FACE_LongDouble is a real data type that represents an IEEE extended double precision floating-point number (having a signed fraction of at least 64 bits and an exponent of at least 15 bits).

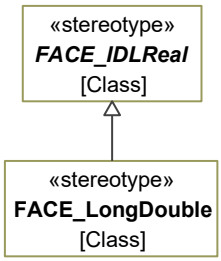

Figure 7-112: FACE_LongDouble

FACE_LongLong

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLInteger

Description

A FACE_LongLong is an integer data type that represents integer values in the range -2^63 to (2^63 - 1).

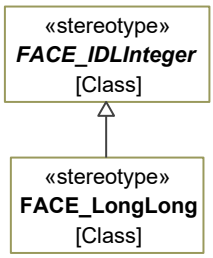

Figure 7-113: FACE_LongLong

FACE_Octet

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLPrimitive

Description

A FACE_Octet is an 8-bit quantity that is guaranteed not to undergo any conversion during transfer between systems.

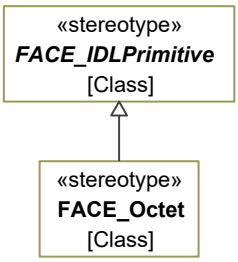

Figure 7-114: FACE_Octet

FACE_PhysicalDataType

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_PlatformComposableElement

Description

A FACE_PhysicalDataType specifies how a logical FACE_AbstractMeasurement is represented on the target platform. Each FACE_PhysicalDataType has a predefined size in bytes that is fixed.

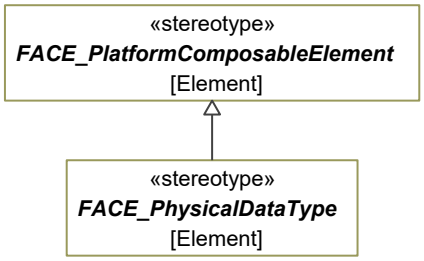

Figure 7-115: abstract FACE_PhysicalDataType

FACE_PlatformAssociation

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_PlatformEntity

Description

A FACE_PlatformAssociation represents a relationship between two or more FACE_PlatformEntities. In addition, there may be one or more FACE_PlatformComposableElements that characterize the relationship. FACE_PlatformAssociations are FACE_PlatformEntities that may also participate in other FACE_PlatformAssociations.

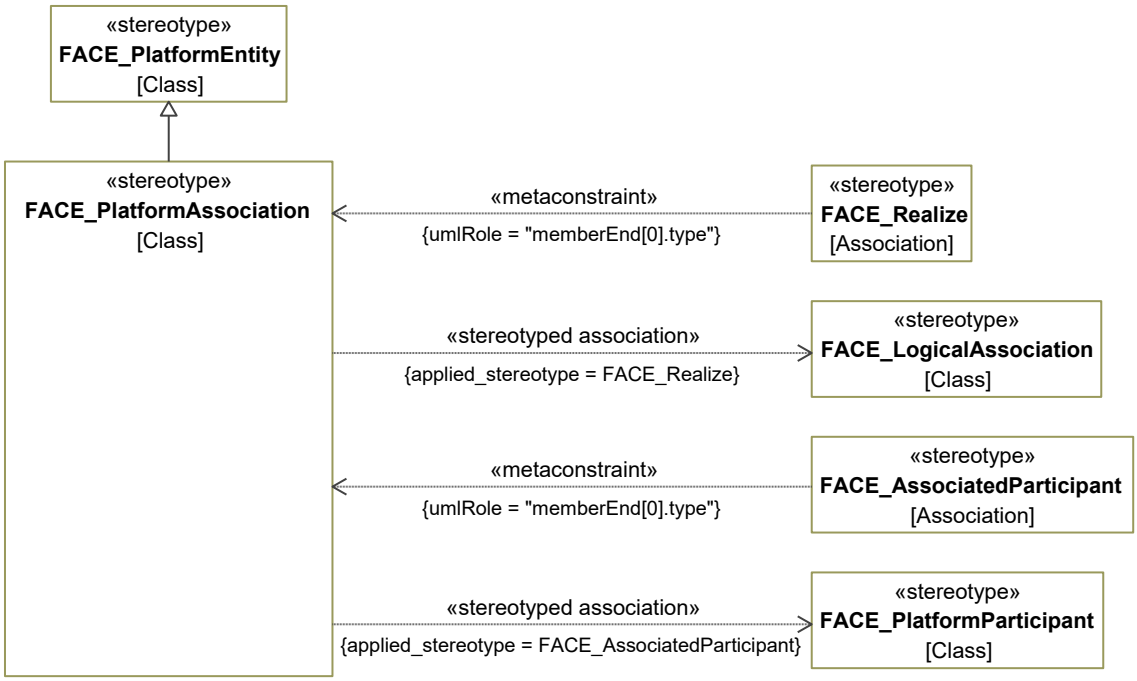

Figure 7-116: FACE_PlatformAssociation

FACE Conformance/OCL Constraints

[1] FACE_PlatformAssociation.participantsConsistentWithRealization	FACE_PlatformParticipants in a FACE_PlatformAssociation must realize FACE_LogicalParticipants in the FACE_LogicalAssociation that the FACE_PlatformAssociation realizes.
[2] FACE_PlatformAssociation.participantsRealizeUniquely	FACE_PlatformParticipants in a FACE_PlatformAssociation must realize unique FACE_LogicalParticipants.

FACE_PlatformCharacteristic

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_ModelElement

Description

A FACE_PlatformCharacteristic is a defining feature of a FACE_PlatformEntity. The rolename attribute defines the name of the FACE_PlatformCharacteristic within the scope of the FACE_PlatformEntity. The lowerBound and upperBound attributes define the multiplicity of the composed Characteristic. An upperBound multiplicity of -1 represents an unbounded sequence.

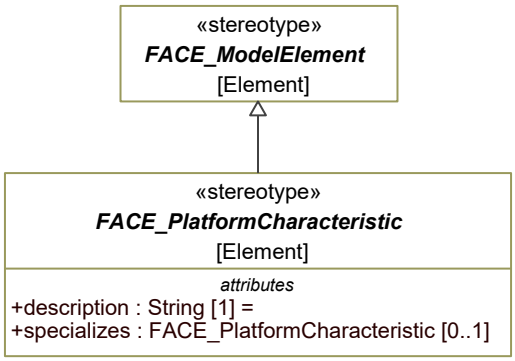

Figure 7-117: abstract FACE_PlatformCharacteristic

Attributes

description : String [1]

specializes : FACE_PlatformCharacteristic [0..1]

FACE Conformance/OCL Constraints

[1] FACE_PlatformCharacteristic.lowerBound_LTE_UpperBound	A FACE_PlatformCharacteristic's lowerBound must be less than or equal to its upperBound, unless its upperBound is -1.
[2] FACE_PlatformCharacteristic.rolenameIsNotReservedWord	The rolename of a FACE_PlatformCharacteristic must not be an IDL reserved word.
[3] FACE_PlatformCharacteristic.rolenameIsValidIdentifier	The rolename of a FACE_PlatformCharacteristic must be a valid identifier.
[4] FACE_PlatformCharacteristic.specializationConsistentWithRealization	If a FACE_PlatformCharacteristic specializes, its specialization must be consistent with its realization's specialization.
[5] FACE_PlatformCharacteristic.upperBoundValid	A FACE_PlatformCharacteristic's upperBound must be equal to -1 or greater than 1.

FACE_PlatformComposableElement

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_PlatformElement

Description

A FACE_PlatformComposableElement is a FACE_PlatformElement that is allowed to participate in a FACE_Composition relationship. In other words, these are the FACE_PlatformElements that may be a characteristic of a FACE_PlatformEntity.

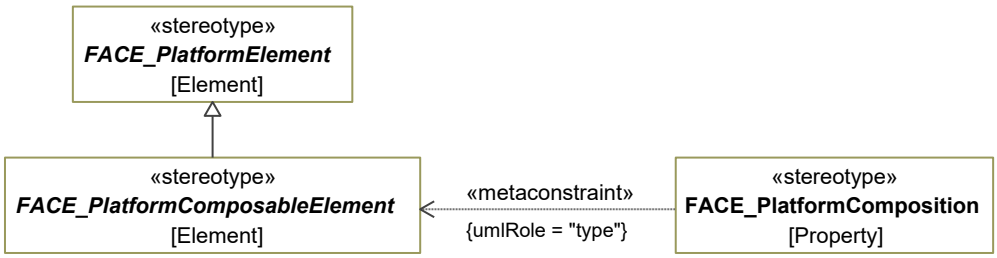

Figure 7-118: abstract FACE_PlatformComposableElement

FACE_PlatformComposition

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_PlatformCharacteristic
Extension: Property

Description

A FACE_PlatformComposition is the mechanism that allows FACE_PlatformEntities to be constructed from other FACE_PlatformComposableElements. The type of a FACE_PlatformComposition is the FACE_PlatformComposableElement being used to construct the FACE_PlatformEntity. The lowerBound and upperBound define the multiplicity of the composed FACE_PlatformEntity. An upperBound multiplicity of -1 represents an unbounded sequence. If type is a FACE_IDLPrimitive, the precision attribute specifies a measure of the detail in which a quantity is captured.

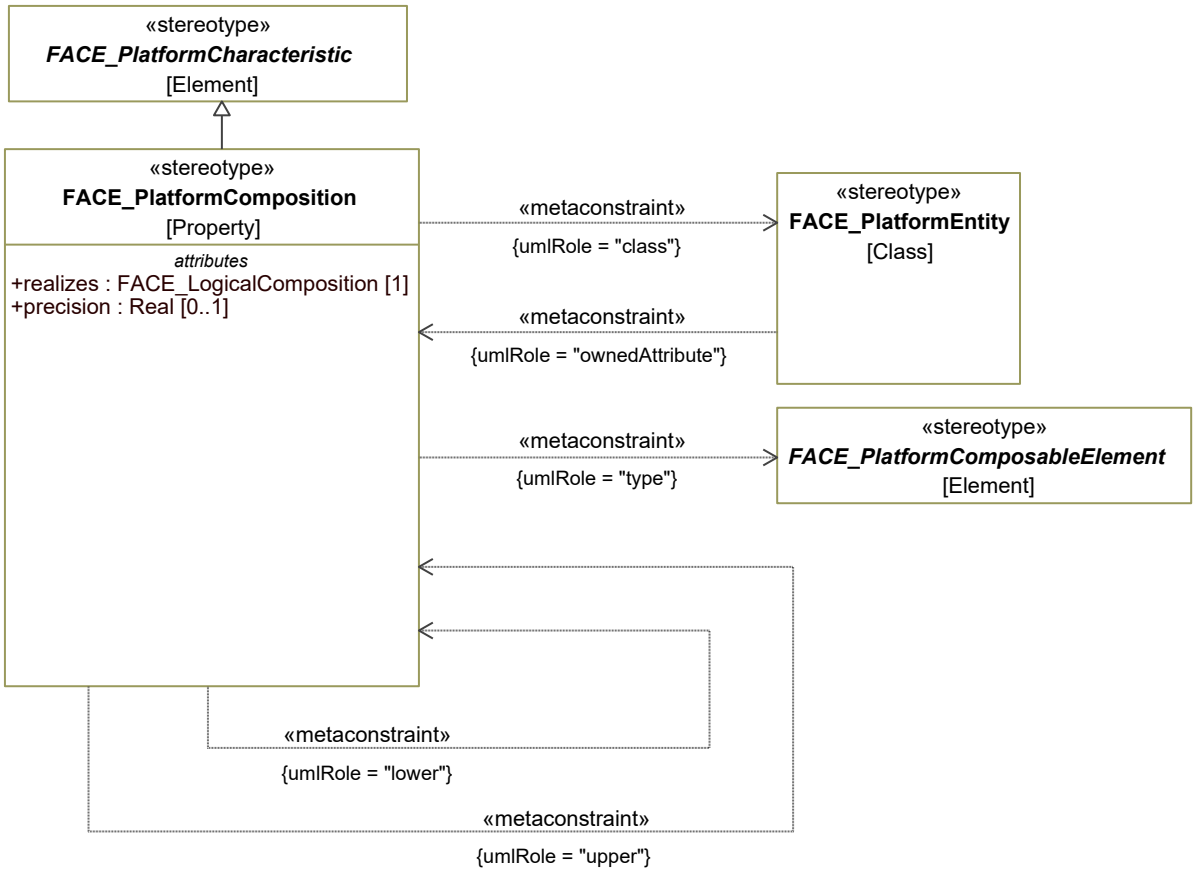

Figure 7-119: FACE_PlatformComposition

Attributes

precision : Real [0..1]

realizes : FACE_LogicalComposition [1]

Constraints

[1] FACE_PlatformComposition.class	Value for the class metaproperty must be stereotyped «FACE_PlatformEntity» or its specializations.
[2] FACE_PlatformComposition.multiplicity.lowerbound	The value for the multiplicity.lowerBound metaproperty must be an integer greater than or equal to -1.
[3] FACE_PlatformComposition.multiplicity.upperbound	The value for the multiplicity.upperBound metaproperty must be an integer greater than or equal to -1
[4] FACE_PlatformComposition.type	Value for the type metaproperty must be stereotyped «FACE_PlatformComposableElement» or its specializations.

FACE Conformance/OCL Constraints

[1] FACE_PlatformComposition.composedIDLNumberHasPrecisionSet	A FACE_PlatformComposition whose type is an IDLNumber must have a precision greater than zero.
[2] FACE_PlatformComposition.multiplicityConsistentWithRealization	A FACE_PlatformComposition's multiplicity must be at least as restrictive as the FACE_LogicalComposition it realizes.
[3] FACE_PlatformComposition.multiplicityConsistentWithSpecialization	A FACE_PlatformComposition's multiplicity must be at least as restrictive as the FACE_PlatformComposition it specializes.
[4] FACE_PlatformComposition.typeConsistentWithRealization	A FACE_PlatformComposition's type must be consistent with its realization's type.

FACE_PlatformElement

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_DataModelElement

Description

A FACE_PlatformElement is the root type for defining the FACE_PlatformElements of the FACE Data Model Language.

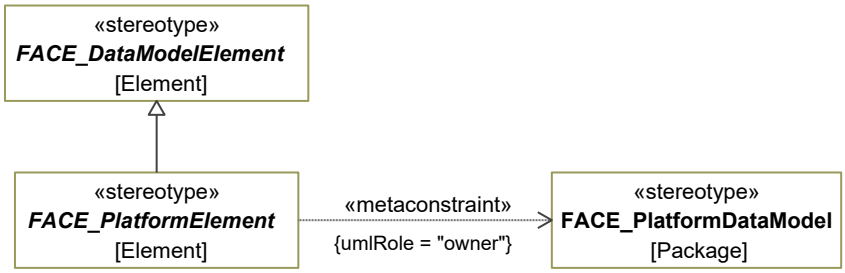

Figure 7-120: abstract FACE_PlatformElement

Constraints

[1] FACE_PlatformElement.owner	Elements that are stereotyped by specializations of this abstract stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_PlatformDataModel»

FACE Conformance/OCL Constraints

[1] FACE_PlatformElement.hasUniqueName	Each FACE_PlatformElement must have a unique name.
[2] FACE_PlatformElement.nameIsNotReservedWord	A FACE_PlatformElement's name may not be an IDL reserved word.

FACE_PlatformEntity

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_PlatformComposableElement, FACE_TraceableElement, FACE_SpecializationOwner
Extension: Class

Description

A FACE_PlatformEntity "realizes" a FACE_LogicalEntity in terms of FACE_PhysicalDataTypes and other FACE_PlatformEntities composed of FACE_PhysicalDataTypes. A FACE_PlatformEntity's composition hierarchy is consistent with the composition hierarchy of the FACE_LogicalEntity that it realizes. The FACE_PlatformEntity's composed Entities realize one to one the FACE_LogicalEntity's composed Entities; the FACE_PlatformEntity's composed FACE_PhysicalDataTypes realize many to one the FACE_LogicalEntity's composed FACE_Measurements.

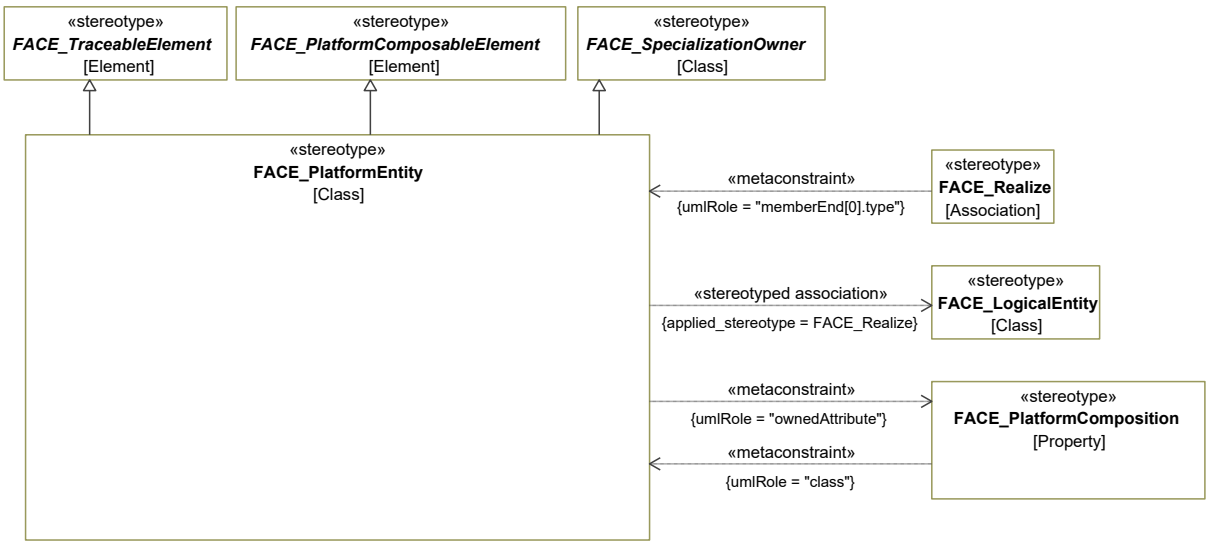

Figure 7-121: FACE_PlatformEntity

Constraints

[1] FACE_PlatformEntity.ownedAttribute	The value for the ownedAttribute metaproperty must be stereotyped «FACE_PlatformComposition» or its specializations

FACE Conformance/OCL Constraints

[1] FACE_PlatformEntity.characteristicsHaveUniqueRolenames	A FACE_PlatformCharacteristic's rolename must be unique within a FACE_PlatformEntity.
[2] FACE_PlatformEntity.compositionsConsistentWithRealization	FACE_PlatformCompositions in a FACE_PlatformEntity must realize FACE_LogicalCompositions in the FACE_LogicalEntity that the FACE_PlatformEntity realizes.
[3] FACE_PlatformEntity.hasAtLeastOneLocalCharacteristic	A FACE_PlatformEntity must have at least one FACE_PlatformCharacteristic defined locally (not through generalization), unless the

	FACE_PlatformEntity is in the "middle" of a generalization hierarchy.
[4] FACE_PlatformEntity.specializationConsistentWithRealization	If a FACE_PlatformEntity specializes, its specialization must be consistent with its realization's specialization.

FACE_PlatformParticipant

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_PlatformCharacteristic
Extension: Class

Description

A FACE_PlatformParticipant is the mechanism that allows a FACE_PlatformAssociation to be constructed between two or more FACE_PlatformEntities. The type of a FACE_PlatformParticipant is the FACE_PlatformEntity being used to construct the FACE_PlatformAssociation. The sourceLowerBound and sourceUpperBound attributes define the multiplicity of the FACE_PlatformAssociation relative to the Participant. A sourceUpperBound multiplicity of -1 represents an unbounded sequence. The path attribute of the Participant describes the chain of entity characteristics to traverse to reach the subject of the association beginning with the entity referenced by the type attribute.

The strings provided in the "path" tagged value are a representation of the full set of FACE Platform CharacteristicPathNode, ParticipantPathNode, and PathNode elements in the path attribute as specified in the Technical Standard for Future Airborne Capability Environment (FACE™), Edition 3.0. The notation used for path string is described in Section 3.6.4.1.1.3 of the Technical Standard for Future Airborne Capability Environment (FACE™), Edition 2.1. The two notations (elements and string) are interchangeable using a translation algorithm. XMI exchange mechanisms between models using the FACE Profile for UAF and the FACE XMI (face) file are required to translate between the two notations.

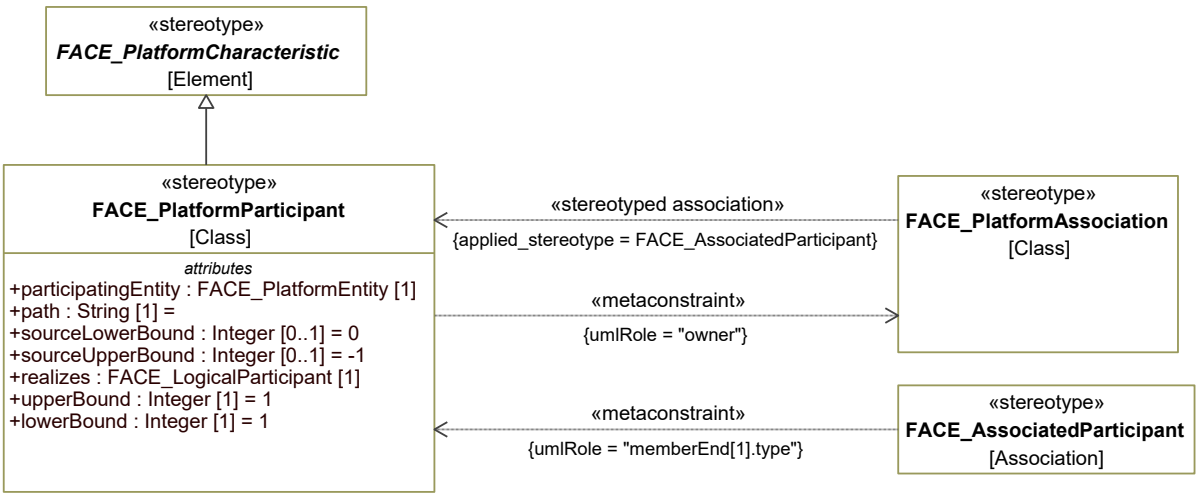

Figure 7-122: FACE_PlatformParticipant

Attributes

lowerBound : Integer [1]

participatingEntity : FACE_PlatformEntity [1]

path : String [1]

realizes : FACE_LogicalParticipant [1]

sourceLowerBound : Integer [0..1]

sourceUpperBound : Integer [0..1]

upperBound : Integer [1]

Constraints

[1] FACE_PlatformParticipant.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_PlatformAssociation»

FACE Conformance/OCL Constraints

[1] FACE_PlatformParticipant.multiplicityConsistentWithRealization	A FACE_PlatformParticipant's multiplicity must be at least as restrictive as the FACE_LogicalParticipant it realizes.
[2] FACE_PlatformParticipant.multiplicityConsistentWithSpecialization	A FACE_PlatformParticipant's multiplicity must be at least as restrictive as the FACE_PlatformParticipant it specializes.
[3] FACE_PlatformParticipant.rolenameDefined	A FACE_PlatformParticipant must have a rolename, either projected from a characteristic or defined directly on the FACE_PlatformParticipant.
[4] FACE_PlatformParticipant.typeConsistentWithRealization	If FACE_PlatformParticipant "A" realizes FACE_LogicalParticipant "B", then A's type must realize B's type, and A's PathNode sequence must "realize" B's PathNode sequence. (A PathNode sequence "A" "realizes" a sequence "B" if the projected element of each PathNode in A realizes the projected element of the corresponding PathNode in B.)

FACE_PlatformQuery

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_TraceableElement, FACE_PlatformElement
Extension: Class

Description

A FACE_PlatformQuery is a specification that defines the content of FACE_PlatformView as a set of FACE_PlatformCharacteristics projected from a selected set of related FACE_PlatformEntities. The specification attribute captures the specification of a Query as defined by the Query grammar.

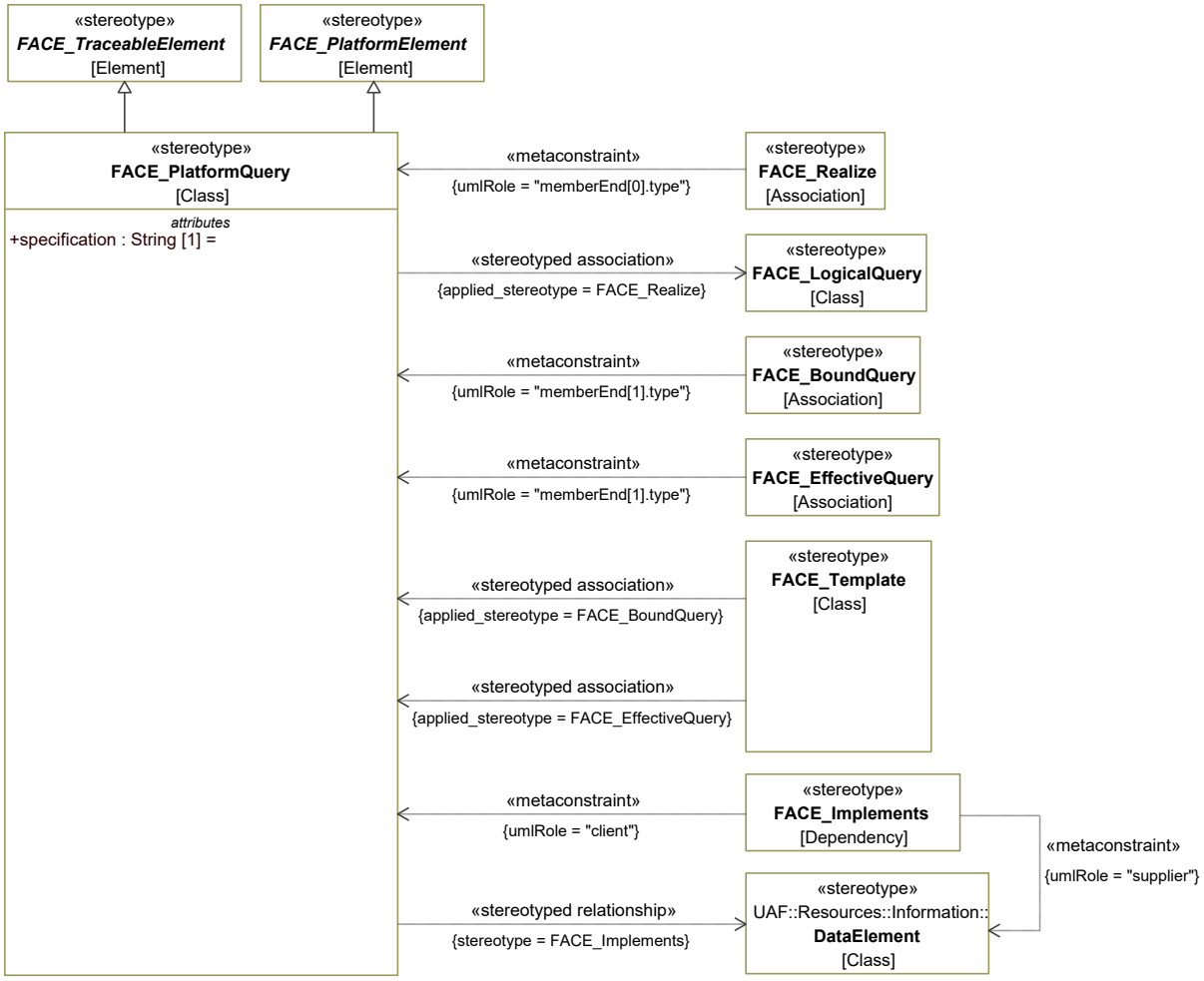

Figure 7-123: FACE_PlatformQuery

Attributes

specification : String [1]

FACE_PlatformView

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_PlatformElement, FACE_TraceableElement
Extension: Class

Description

A FACE_PlatformView is a FACE_Template or a FACE_CompositeTemplate.

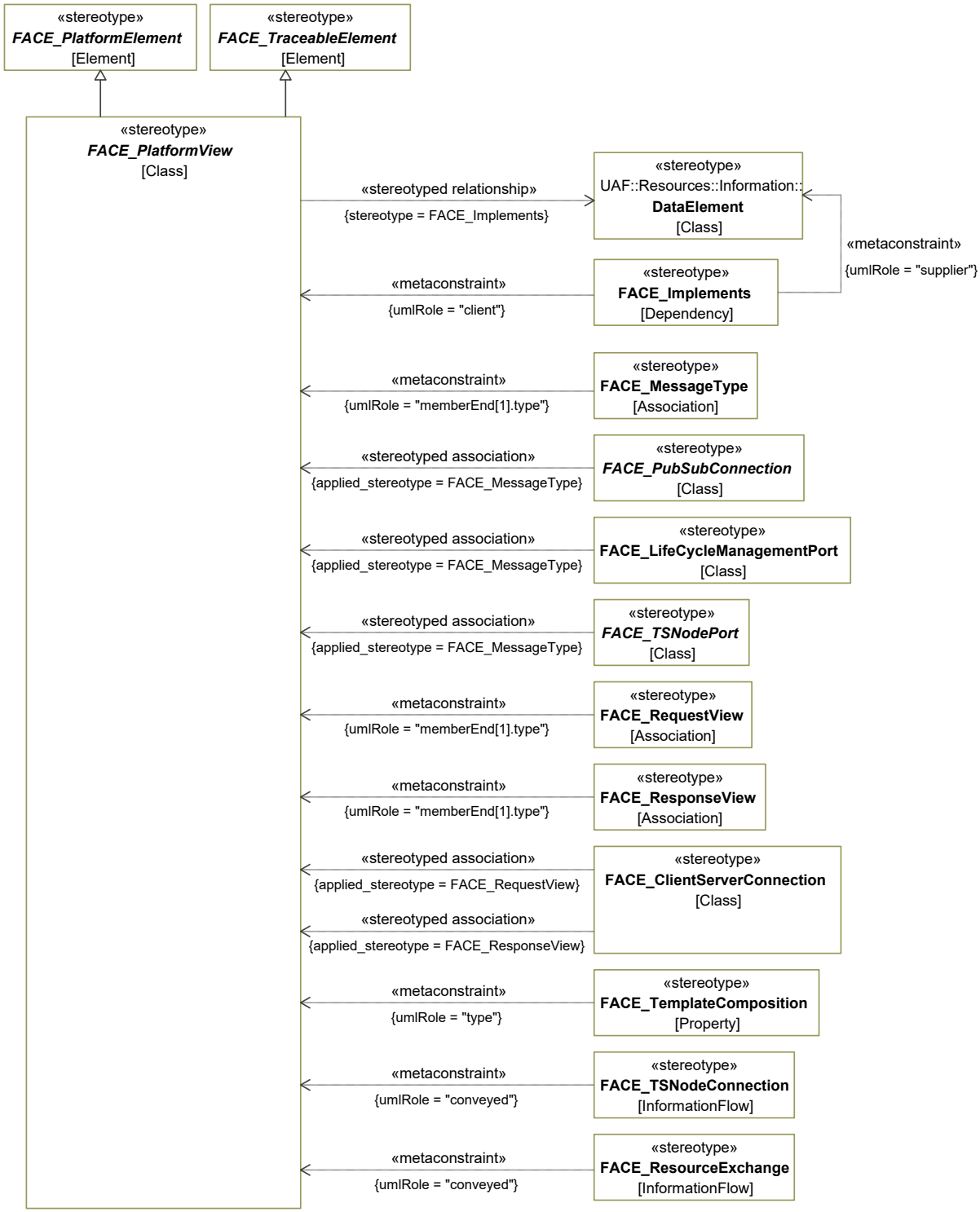

Figure 7-124: abstract FACE_PlatformView

FACE_Short

Package: PlatformDataModel
isAbstract: No

Generalization: FACE_IDLInteger

Description

A FACE_Short is an integer data type that represents integer values in the range -2^15 to (2^15 - 1).

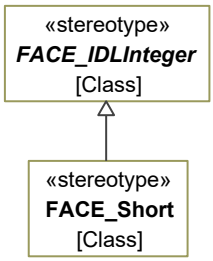

Figure 7-125: FACE_Short

FACE_String

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLUnboundedString

Description

A FACE_String is a data type that represents a variable length sequence of Char (all 8-bit quantities except NULL). The length is a non-negative integer and is available at run-time. The length is not maximally bounded.

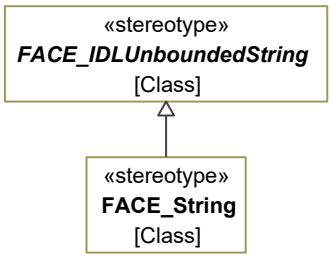

Figure 7-126: FACE_String

FACE_StringType

Package: PlatformDataModel
isAbstract: Yes
Generalization: FACE_IDLPrimitive

Description

A FACE_StringType is a FACE_IDLBoundedString, a FACE_IDLUnboundedString or a FACE_IDLCharacterArray.

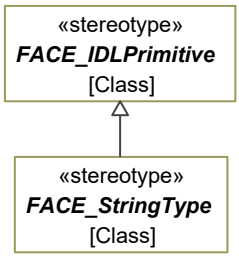

Figure 7-127: abstract FACE_StringType

FACE_Template

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_PlatformView
Extension: Class

Description

A FACE_Template is a specification that defines a structure for Characteristics projected by its boundQuery or its effectiveQuery. The specification attribute captures the specification of a Template as defined by the Template grammar in Appendix J.4.

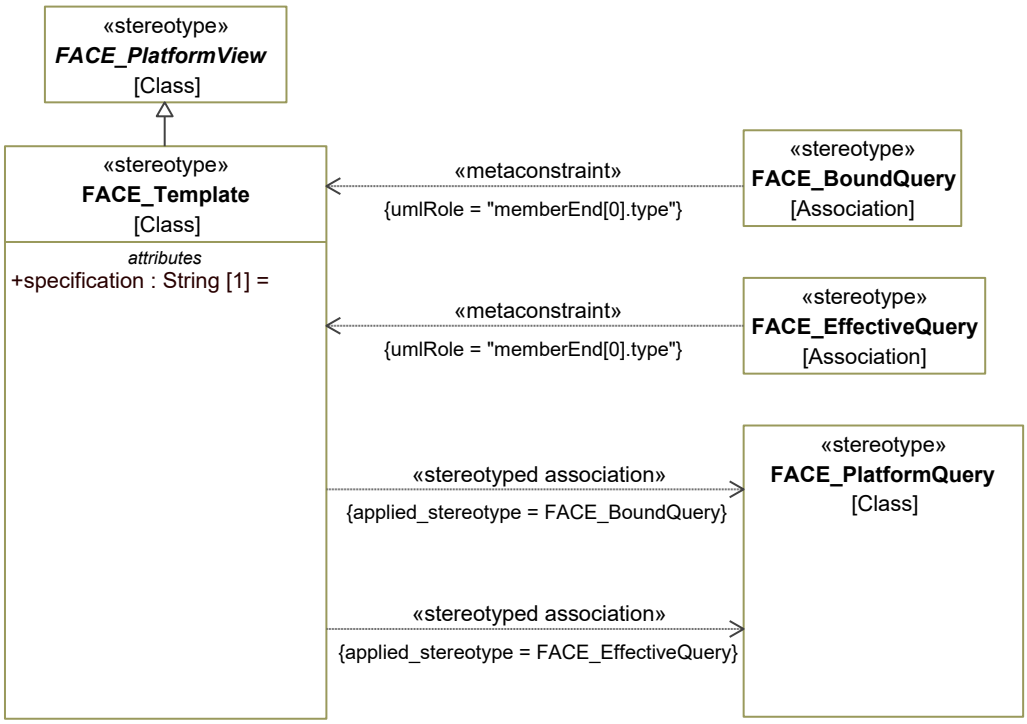

Figure 7-128: FACE_Template

Attributes

specification : String [1]

FACE Conformance/OCL Constraints

[1] FACE_Template.equivalentEntityTemplateHasNoQuery	A FACE_Template who's main_template_method_decl is a main_equivalent_entity_type_template_method_decl may not have its boundQuery or effectiveQuery set. Any other FACE_Template must have its boundQuery set.

FACE_TemplateComposition

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_ModelElement
Extension: Property

Description

A FACE_TemplateComposition is the mechanism that allows a FACE_CompositeTemplate to be constructed from FACE_Templates and other FACE_CompositeTemplates. The rolename attribute defines the name of the composed FACE_PlatformView within the scope of the composing FACE_CompositeTemplate. The type of a FACE_TemplateComposition is the FACE_PlatformView being used to construct the FACE_CompositeTemplate.

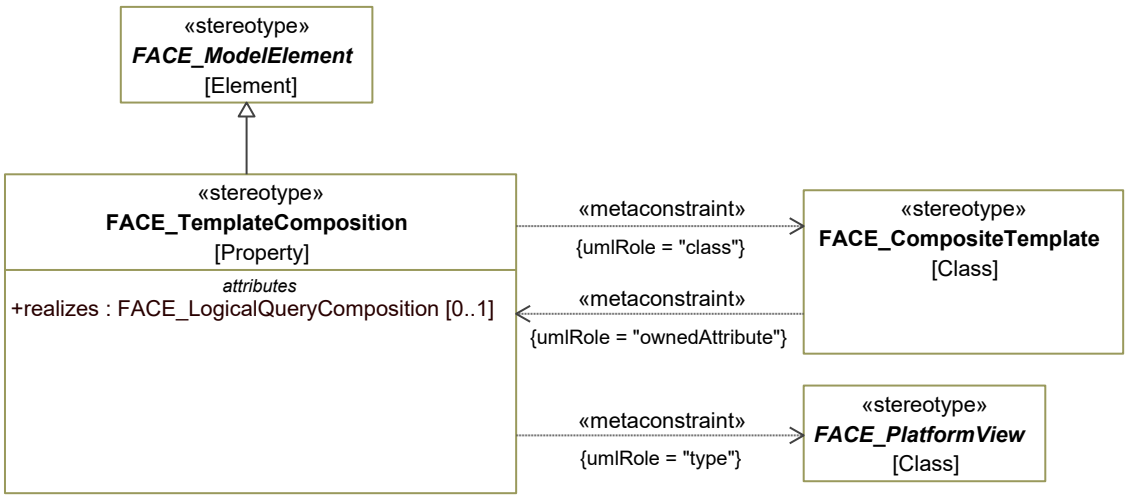

Figure 7-129: FACE_TemplateComposition

Attributes

realizes : FACE_LogicalQueryComposition [0..1]

Constraints

[1] FACE_TemplateComposition.class	Value for class metaproperty must be stereotyped «FACE_CompositeTemplate».

[2] FACE_TemplateComposition.type	Value for type metaproperty must be stereotyped «FACE_PlatformView» or its specializations.

FACE Conformance/OCL Constraints

[1] FACE_TemplateComposition.rolenameIsNotReservedWord	The rolename of a FACE_TemplateComposition may not be an IDL reserved word.
[2] FACE_TemplateComposition.rolenameIsValidIdentifier	The rolename of a FACE_TemplateComposition must be a valid identifier.
[3] FACE_TemplateComposition.typeConsistentWithRealization	If FACE_TemplateComposition "A" realizes FACE_LogicalQueryComposition "B", then if A's type is a FACE_CompositeTemplate, then A's type must realize B's type, and if A's type is a FACE_Template and defines an effectiveQuery, then A's type's effectiveQuery must realize B's type.

FACE_ULong

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLUnsignedInteger

Description

A FACE_ULong is an integer data type that represents integer values in the range 0 to (2^32 - 1).

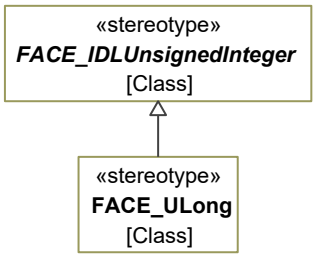

Figure 7-130: FACE_ULong

FACE_ULongLong

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLUnsignedInteger

Description

A FACE_ULongLong is an integer data type that represents integer values in the range 0 to (2^64 - 1).

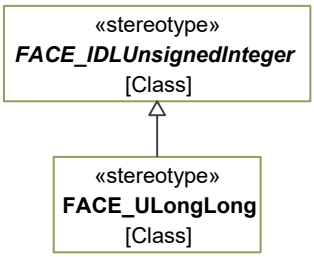

Figure 7-131: FACE_ULongLong

FACE_UShort

Package: PlatformDataModel
isAbstract: No
Generalization: FACE_IDLUnsignedInteger

Description

A FACE_UShort is an integer data type that represents integer values in the range 0 to (2^16 - 1).

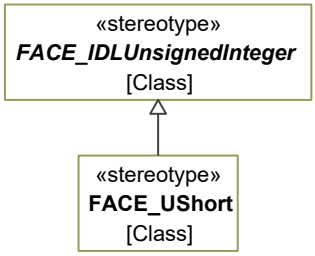

Figure 7-132: FACE_UShort

7.1.1.2 FACE_UAF_Profile::FACE Data Architecture::Integration Model

FACE_IntegrationContext

Package: Integration Model
isAbstract: No
Generalization: FACE_IntegrationElement
Extension: Package

Description

A FACE_IntegrationContext is a container used to group a set of FACE_TransportNodes and FACE_TSNodeConnections related to each other by a common, integrator defined context (e.g., collection and distribution of navigation data).

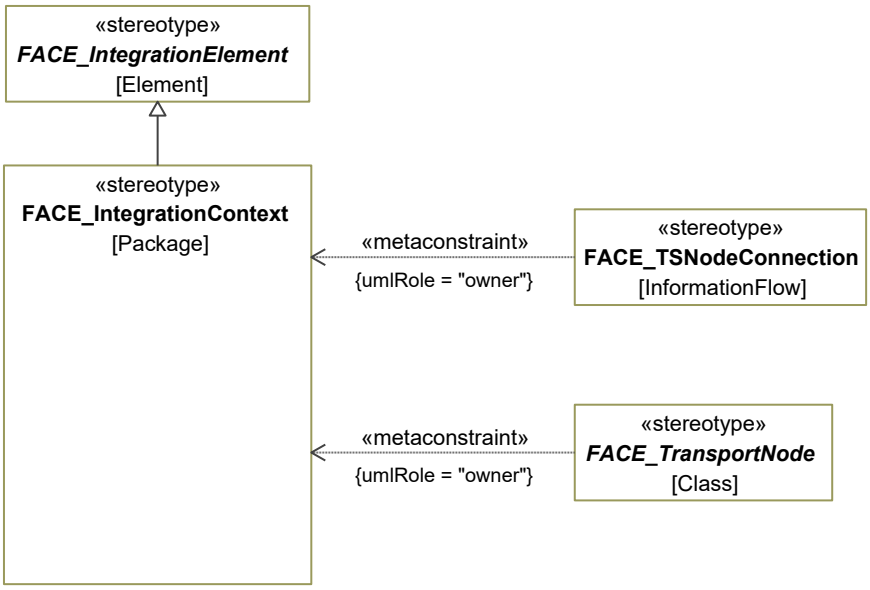

Figure 7-133: FACE_IntegrationContext

FACE_IntegrationElement

Package: Integration Model
isAbstract: Yes
Generalization: FACE_Element

Description

A FACE_IntegrationElement is the root type for defining the integration elements of the FACE_ArchitectureModel.

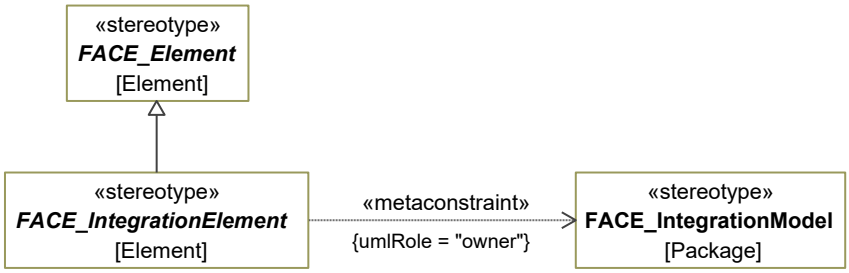

Figure 7-134: abstract FACE_IntegrationElement

Constraints

[1] FACE_IntegrationElement.owner	Elements that are stereotyped by specializations of this abstract stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_IntegrationModel»

FACE Conformance/OCL Constraints

[1] FACE_IntegrationElement.hasUniqueName	All FACE Integration Elements must have a unique name.

FACE_TransportChannel

Package: Integration Model
isAbstract: No
Generalization: FACE_IntegrationElement
Extension: Class

Description

A FACE_TransportChannel is a place holder for an integrator supplied configuration between transport end points.

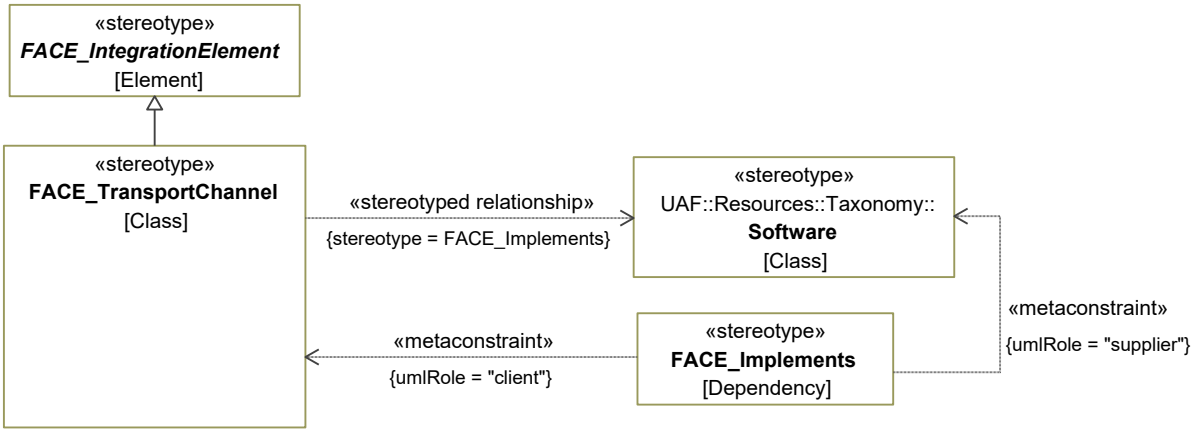

Figure 7-135: FACE_TransportChannel

FACE_TransportNode

Package: Integration Model
isAbstract: Yes
Generalization: FACE_Element
Extension: Class

Description

A FACE_TransportNode is an abstraction of a node that performs a function along a path of communication from source FACE_UnitOfPortability (UoPs) to destination UoPs.

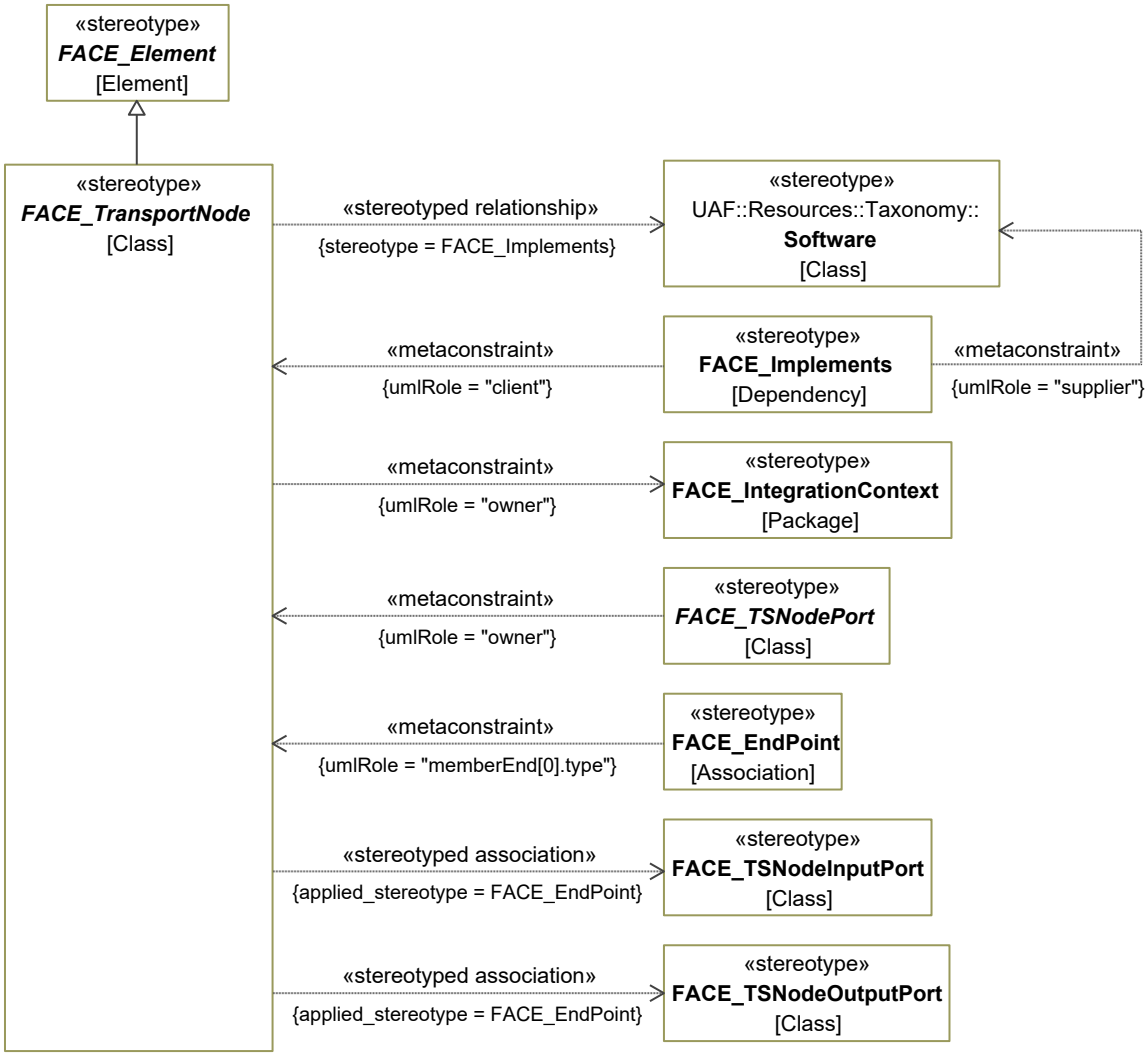

Figure 7-136: abstract FACE_TransportNode

Constraints

[1] FACE_TransportNode.owner	Elements that are stereotyped by specializations of this abstract stereotype may only be contained in (owned by) elements stereotyped by «FACE_IntegrationContext»

FACE Conformance/OCL Constraints

[1] FACE_TransportNode.hasCorrectInputCountTransportNode.hasCorrectInputCount	A FACE_ViewSource may have no inputs. A FACE_ViewSink, FACE_ViewFilter, FACE_ViewTransformation, or FACE_ViewTransporter may have one input. A FACE_ViewAggregation may have more than one input.

[2] FACE_TransportNode.hasCorrectOutputCount	A FACE_ViewSink may have no outputs. A FACE_ViewSource, FACE_ViewFilter, FACE_ViewAggregation, FACE_ViewTransformation, or FACE_ViewTransporter may have one output.
[3] FACE_TransportNode.noCycles	An FACE_IntegrationContext may contain no cycles.

FACE_TSNodeConnection

Package: Integration Model
isAbstract: No
Generalization: FACE_ModelElement
Extension: InformationFlow

Description

A FACE_TSNodeConnection represents a connection between two FACE_TransportNodes.

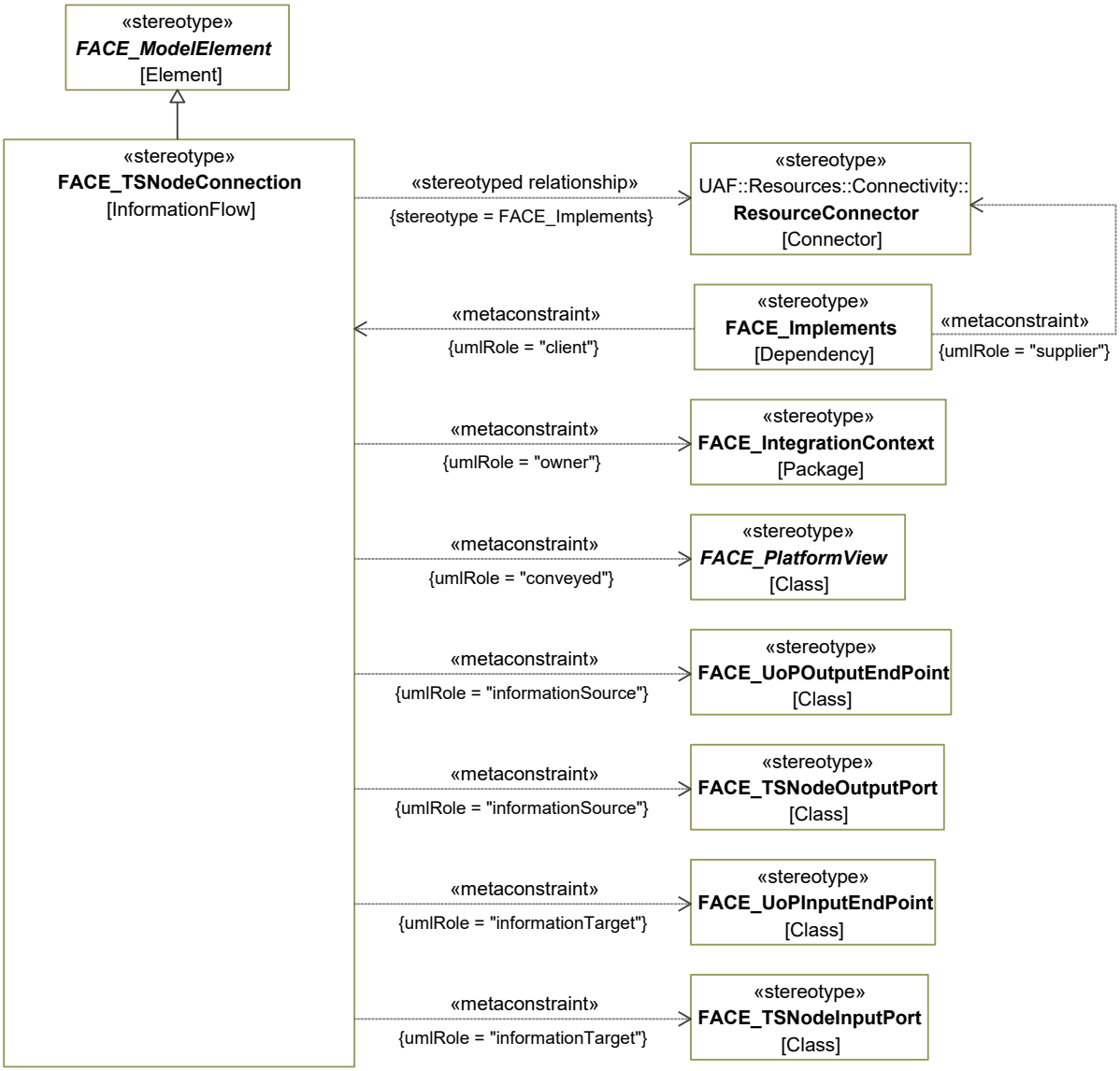

Figure 7-137: FACE_TSNodeConnection

Constraints

[1] FACE_TSNodeConnection.conveyed	Value for the conveyed metaproperty must be stereotyped by a specialization of «FACE_PlatformView».
[2] FACE_TSNodeConnection.informationSource	The value for the informationSource metaproperty must be stereotyped by one of the following: «FACE_UoPOutputEndPoint» «FACE_TSNodeOutputPort»
[3] FACE_TSNodeConnection.informationTarget	The value for the informationTarget metaproperty must be stereotyped by one of the following: «FACE_UoPInputEndPoint» «FACE_TSNodeInputPort»

[4] FACE_TSNodeConnection.owner	Elements with this stereotype may only be contained in (owned by) elements stereotyped by «FACE_IntegrationContext»

FACE Conformance/OCL Constraints

[1] FACE_TSNodeConnection.connectWithinSameContext	A FACE_TSNodeConnection may connect only FACE_TransportNodes that are in the same FACE_IntegrationContext as the FACE_TSNodeConnection.
[2] FACE_TSNodeConnection.destinationIsInput	A FACE_TSNodeConnection's destination must be an input.
[3] FACE_TSNodeConnection.sourceIsOutput	A FACE_TSNodeConnection's source must be an output
[4] FACE_TSNodeConnection.sourceViewMatchesDestinationView	A FACE_TSNodeConnection must use the same View on its source and destination.
[5] FACE_TSNodeConnection.transporterOnPath	There must be at least one FACE_ViewTransporter on a path between any two FACE_UoPInstances.

FACE_TSNodeInputPort

Package: Integration Model
isAbstract: No
Generalization: FACE_TSNodePort

Description

A FACE_TSNodeInputPort is a specialization of a FACE_TSNodePort providing an endpoint which is used to input data to a FACE_TransportNode.

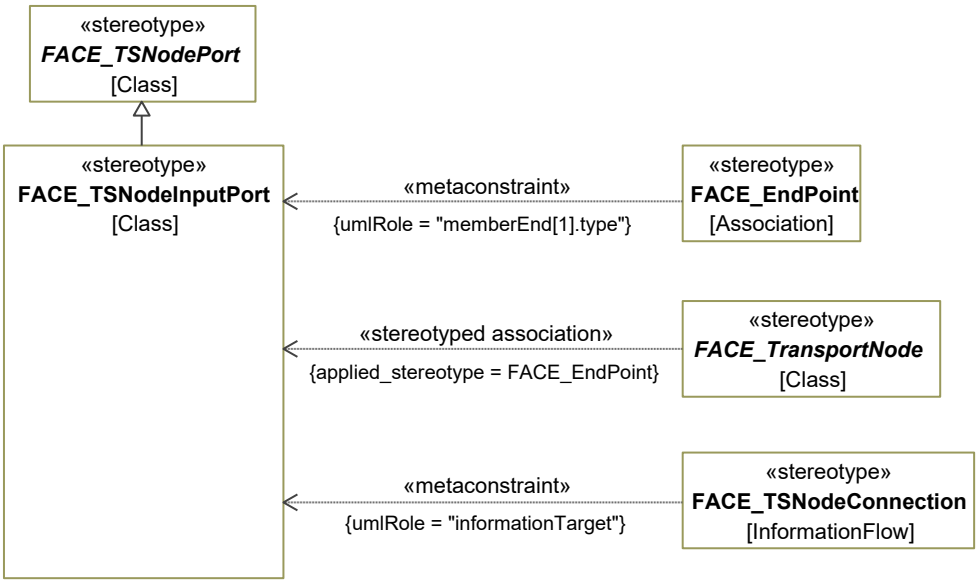

Figure 7-138: FACE_TSNodeInputPort

FACE Conformance/OCL Constraints

[1] FACE_TSNodeInputPort.onlyOneConnection	A FACE_TSNodeInputPort may be the destination of at most one FACE_TSNodeConnection.

FACE_TSNodeOutputPort

Package: Integration Model
isAbstract: No
Generalization: FACE_TSNodePort

Description

A FACE_TSNodeOutputPort is a specialization of a FACE_TSNodePort providing an endpoint which is used to output data from a FACE_TransportNode.

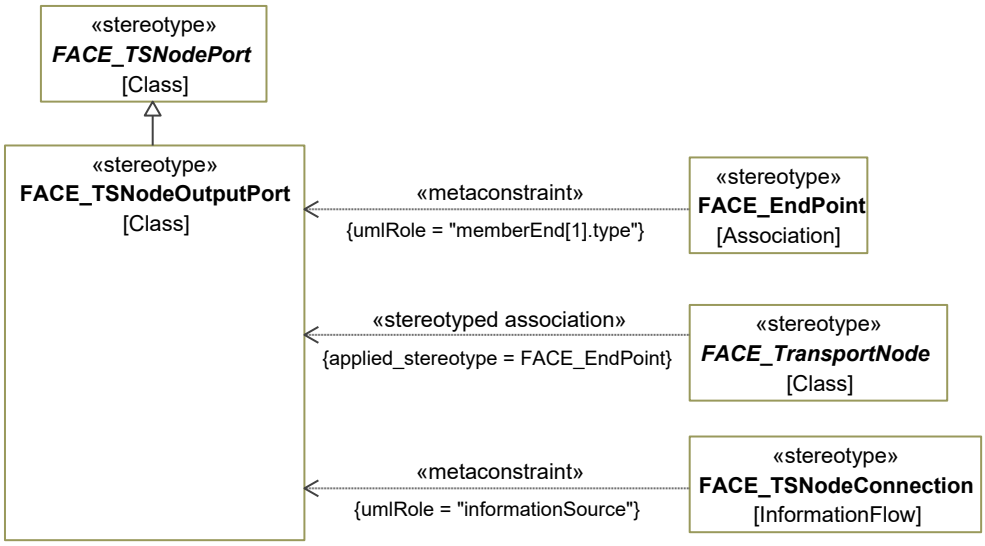

Figure 7-139: FACE_TSNodeOutputPort

FACE_TSNodePort

Package: Integration Model
isAbstract: Yes
Generalization: FACE_TSNodePortBase

Description

A FACE_TSNodePort is a port that provides a connection point to a FACE_TransportNode. A FACE_TSNodePort is typed by the view it references.

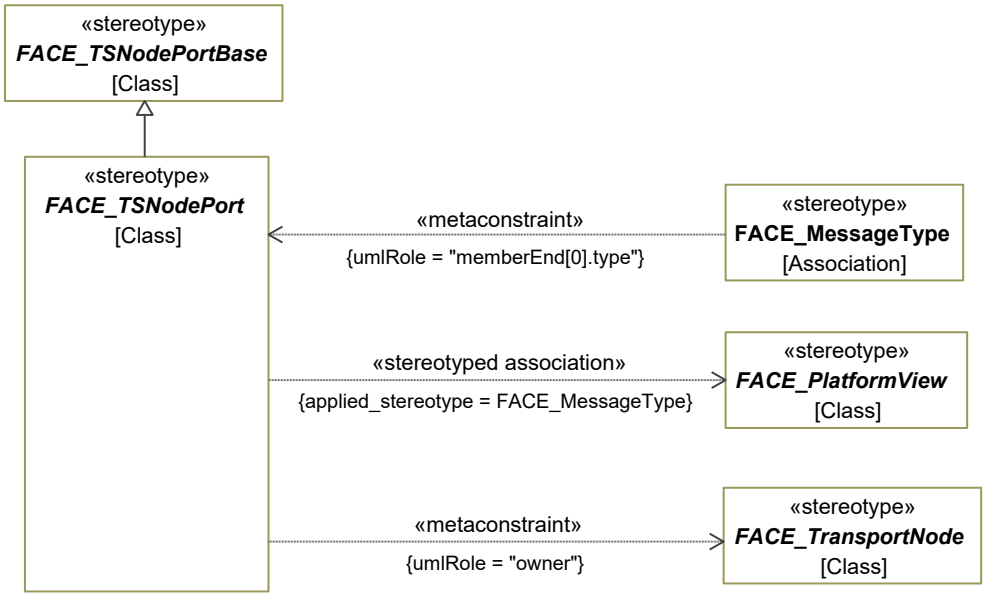

Figure 7-140: abstract FACE_TSNodePort

Constraints

[1] FACE_TSNodePort.owner	Elements that are stereotyped by specializations of this abstract stereotype may only be contained in (owned by) elements stereotyped by «FACE_TransportNode»

FACE_TSNodePortBase

Package: Integration Model
isAbstract: Yes
Generalization: FACE_ModelElement
Extension: Class

Description

A FACE_TSNodePortBase is a port that can be used to connect a FACE_TransportNode and a FACE_UoPEndPoint together using a FACE_TSNodeConnection.

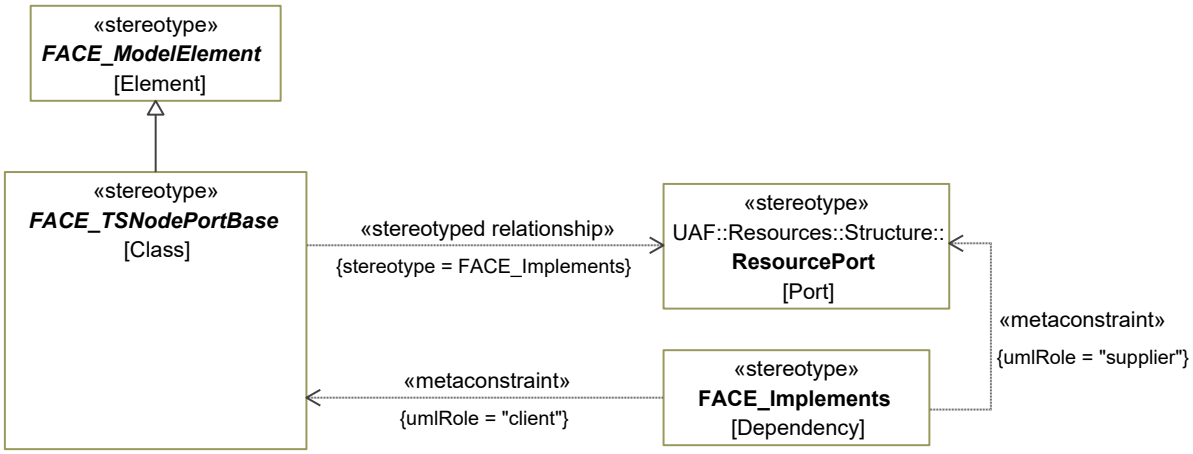

Figure 7-141: abstract FACE_TSNodePortBase

FACE Conformance/OCL Constraints

[1] FACE_TSNodePortBase.isConnected	A FACE_TSNodePortBase must be connected by a FACE_TSNodeConnection.

FACE_UoPEndPoint

Package: Integration Model
isAbstract: Yes
Generalization: FACE_TSNodePortBase

Description

A FACE_UoPEndPoint is a specialization of aFACE_ TSNodePortBase that allows connections in a UoPInstance to be part of a FACE_TSNodeConnection. This supports connecting FACE_UnitOfPortability (UoP) input and output end points to each other and to transport node input and output ports.

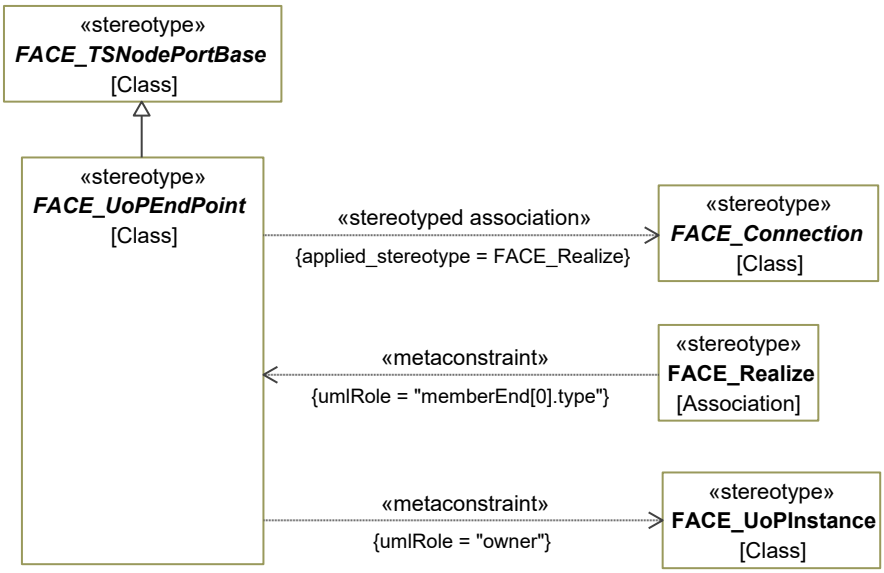

Figure 7-142: abstract FACE_UoPEndPoint

Constraints

[1] FACE_UoPEndPoint.owner	Elements that are stereotyped by specializations of this abstract stereotype may only be contained in (owned by) elements stereotyped by «FACE_UoPInstance»

FACE_UoPInputEndPoint

Package: Integration Model
isAbstract: No
Generalization: FACE_UoPEndPoint

Description

A FACE_UoPInputEndPoint is a specialization of a FACE_UoPEndPoint providing an endpoint which is used to input data to a FACE_UnitOfPortability (UoP).

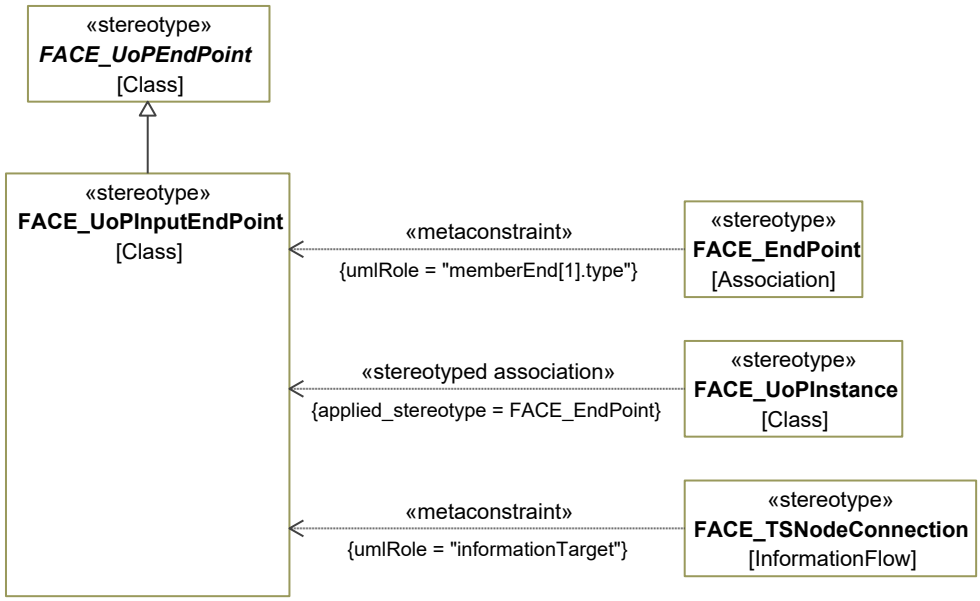

Figure 7-143: FACE_UoPInputEndPoint

FACE Conformance/OCL Constraints

[1] FACE_UoPInputEndPoint.uoPEndPointConsistentWithRealization	A FACE_UoPInputEndPoint's connection may be either a FACE_ClientServerConnection or a FACE_PubSubConnection whose messageExchangeType is OutboundMessage.

FACE_UoPInstance

Package: Integration Model
isAbstract: No
Generalization: FACE_IntegrationElement
Extension: Class

Description

A FACE_UoPInstance represents an instance of a specific FACE_UnitOfPortability (UoP) within the system bounded by an integration model. An integration model can contain multiple instances of the same UoP.

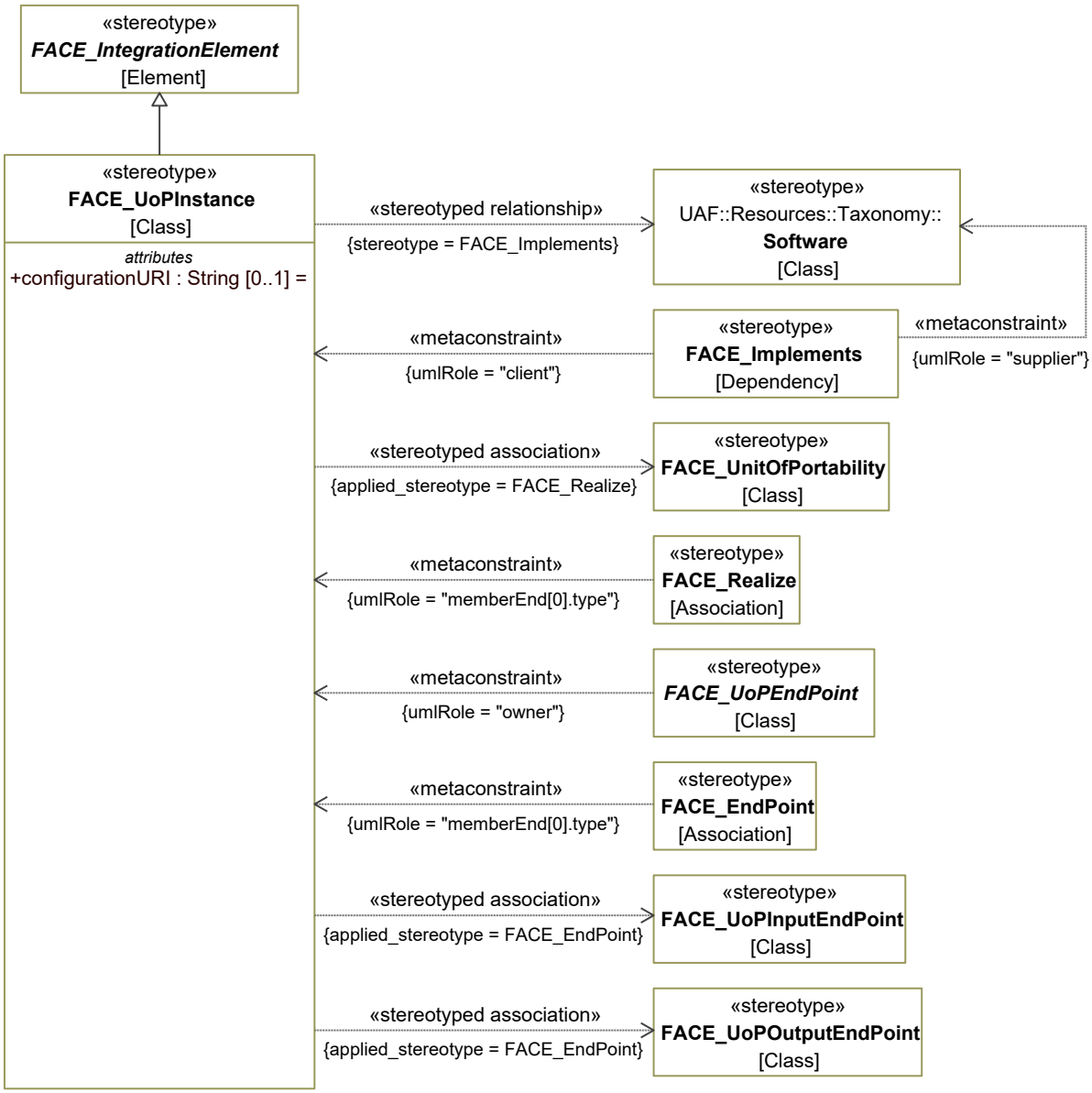

Figure 7-144: FACE_UoPInstance

Attributes

configurationURI : String [0..1]

FACE Conformance/OCL Constraints

[1] FACE_UoPInstance.endPointsConsistentWithRealization	If a FACE_UoPInstance "A" realizes a FACE_UnitOfPortability "B", then A must have one unique FACE_UoPEndPoint that realizes each of B's FACE_PubSubConnections, one unique FACE_UoPInputEndPoint that realizes each of B's FACE_ClientServerConnections, and one FACE_UoPOutputEndPoint that realizes each of B's

	FACE_ClientServerConnections. A FACE_UoPInstance may have no additional FACE_UoPEndPoints.

FACE_UoPOutputEndPoint

Package: Integration Model
isAbstract: No
Generalization: FACE_UoPEndPoint

Description

A FACE_UoPOutputEndPoint is a specialization of a FACE_UoPEndPoint providing an endpoint which is used to output data from a FACE_UnitOfPortability (UoP).

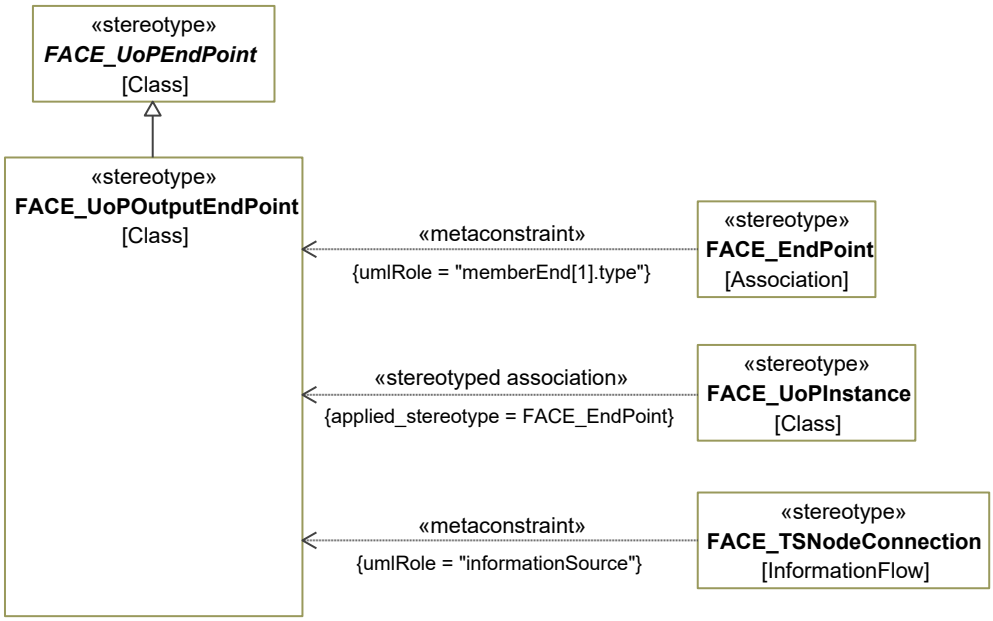

Figure 7-145: FACE_UoPOutputEndPoint

FACE Conformance/OCL Constraints

[1] FACE_UoPOutputEndPoint.onlyOneConnection	A FACE_UoPInputEndPoint may be the destination of at most one FACE_TSNodeConnection.
[2] FACE_UoPOutputEndPoint.uoPEndPointConsistentWithRealization	A FACE_UoPInputEndPoint's connection may be either a FACE_ClientServerConnection or a FACE_PubSubConnection whose messageExchangeType is InboundMessage.

FACE_ViewAggregation

Package: Integration Model
isAbstract: No
Generalization: FACE_TransportNode

Description

A FACE_ViewAggregation represents of an instance of aggregation of data from multiple incoming views into a single outgoing view type, including transformation of input data to that required by the output view type.

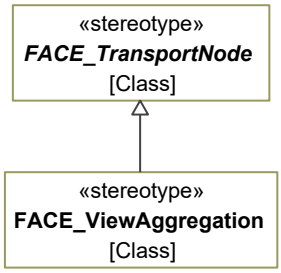

Figure 7-146: FACE_ViewAggregation

FACE_ViewFilter

Package: Integration Model
isAbstract: No
Generalization: FACE_TransportNode

Description

A FACE_ViewFilter represents of an instance of a filter of data allowing a view to either pass through a filter, or to be filtered out (i.e., not passed through). A FACE_ViewFilter performs no transformation of data.

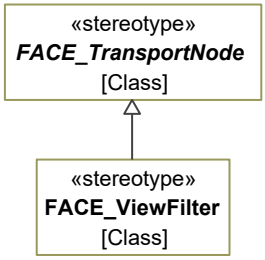

Figure 7-147: FACE_ViewFilter

FACE Conformance/OCL Constraints

[1] FACE_ViewFilter.viewIsConsistent	A FACE_ViewFilter must use the same FACE_PlatformView on its input and output.

FACE_ViewSink

Package: Integration Model
isAbstract: No
Generalization: FACE_TransportNode

Description

A FACE_ViewSink is a FACE_TransportNode that only receives a View.

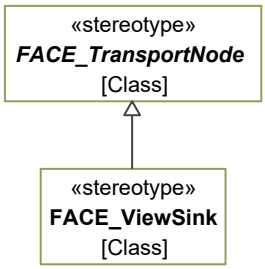

Figure 7-148: FACE_ViewSink

FACE Conformance/OCL Constraints

[1] FACE_ViewSink.viewSinkConnectedToUoPOutputEndPoint	A FACE_ViewSink may only be connected to a FACE_UoPOutputEndPoint.

FACE_ViewSource

Package: Integration Model
isAbstract: No
Generalization: FACE_TransportNode

Description

A FACE_ViewSource is a TransportNode that only provides a View.

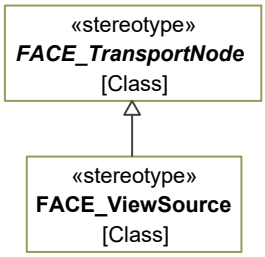

Figure 7-149: FACE_ViewSource

FACE Conformance/OCL Constraints

[1] FACE_ViewSource.viewSourceConnectedToUoPInputEndPoint	A FACE_ViewSource may only be connected to a FACE_UoPInputEndPoint.

FACE_ViewTransformation

Package: Integration Model
isAbstract: No
Generalization: FACE_TransportNode

Description

A FACE_ViewTransformation represents an instance of transformation of data from one view type to another.

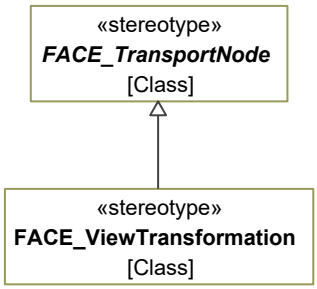

Figure 7-150: FACE_ViewTransformation

FACE_ViewTransporter

Package: Integration Model
isAbstract: No
Generalization: FACE_TransportNode

Description

A FACE_ViewTransporter represents the use of a TransportChannel with the intent of moving a view over it.

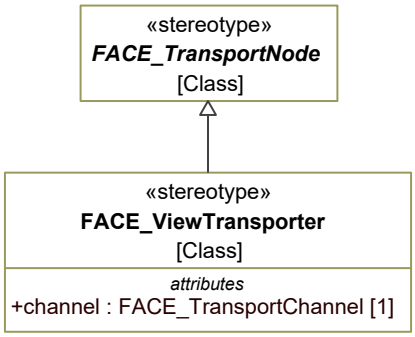

Figure 7-151: FACE_ViewTransporter

Attributes

channel : FACE_TransportChannel [1]

FACE Conformance/OCL Constraints

[1] FACE_ViewTransporter.viewIsConsistent	A FACE_ViewTransporter must use the same FACE_PlatformView on its input and output.

7.1.1.3 FACE_UAF_Profile::FACE Data Architecture::Traceability Model

FACE_ConnectionTrace

Package: Traceability Model
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to connect FACE_ConnectionTraceabilitySet elements to their associated FACE_Connections.

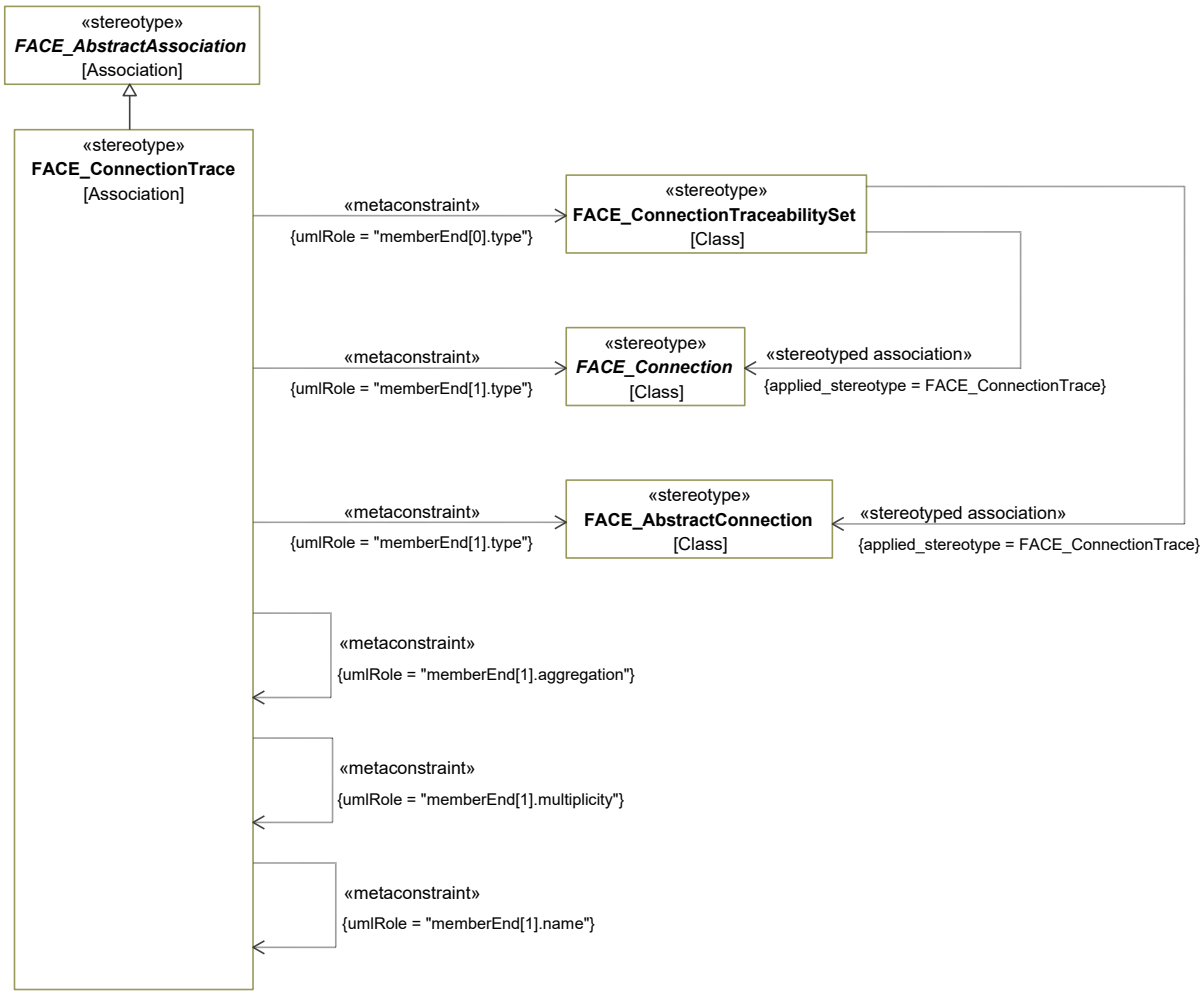

Figure 7-152: FACE_ConnectionTrace

Constraints

[1] FACE_ConnectionTrace.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_ConnectionTraceabilitySet».
[2] FACE_ConnectionTrace.memberEnd[1].aggregation	none
[3] FACE_ConnectionTrace.memberEnd[1].multiplicity	0..*
[4] FACE_ConnectionTrace.memberEnd[1].name	Based on the stereotype of the memberEnd[1].type metaproperty: = specialization of «FACE_Connection», memberEnd[1].name is "Connection" = «FACE_AbstractConnection», memberEnd[1].name is "abstractConnection"

[5] FACE_ConnectionTrace.memberEnd[1].type	The value for the memberEnd[1].type metaproperty must be stereotyped by one of the following: A specialization of «FACE_Connection» «FACE_AbstractConnection»

FACE_ConnectionTraceabilitySet

Package: Traceability Model
isAbstract: No
Generalization: FACE_TraceabilityElement, FACE_TraceableElement
Extension: Class

Description

A FACE_ConnectionTraceabilitySet is used to relate a set of FACE_Connections and/or FACE_AbstractConnections to a set of FACE_TraceabilityPoints.

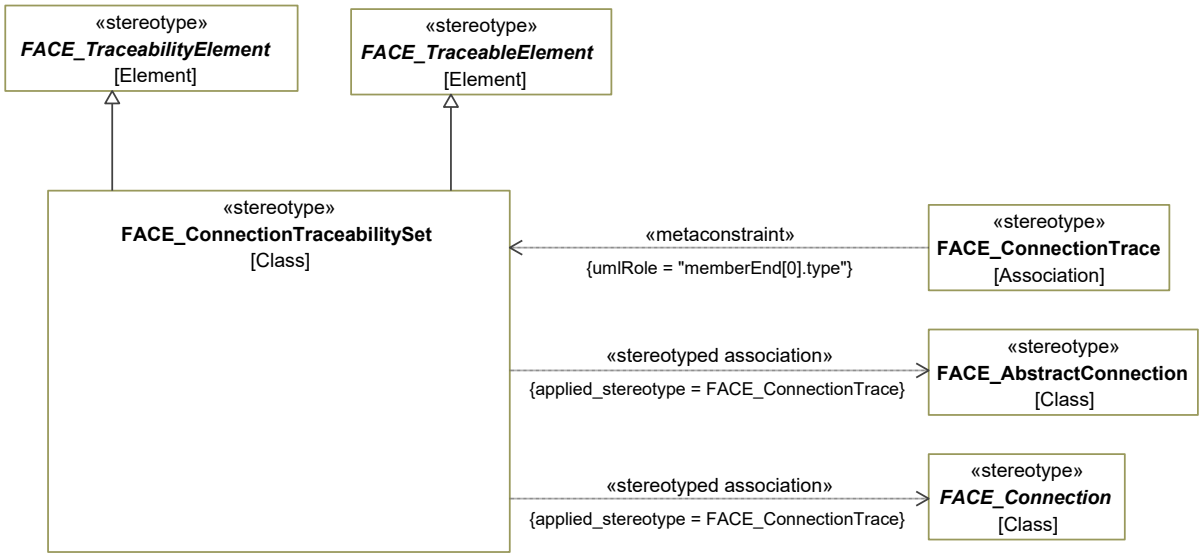

Figure 7-153: FACE_ConnectionTraceabilitySet

FACE_TraceabilityElement

Package: Traceability Model
isAbstract: Yes
Generalization: FACE_Element

Description

A FACE_TraceabilityElement is the root type for defining the FACE_TraceabilityElements of the FACE Architecture Model.

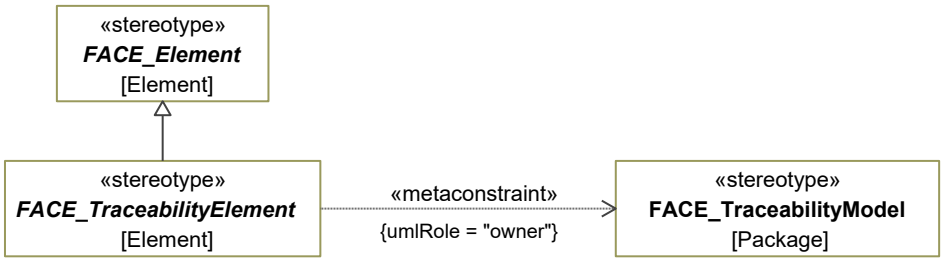

Figure 7-154: abstract FACE_TraceabilityElement

Constraints

[1] FACE_TraceabilityElement.owner	Elements that are stereotyped by specializations of this abstract stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_TraceabilityModel»

FACE Conformance/OCL Constraints

[1] FACE_TraceabilityElement.hasUniqueName	All FACE Traceability Elements must have a unique name.

FACE_TraceabilityPoint

Package: Traceability Model
isAbstract: No
Generalization: FACE_ModelElement
Extension: Class

Description

A FACE_TraceabilityPoint is used to document the relationship between a FACE_TraceableElement and an external model. The reference attribute is a reference to the external model. The rationale attribute is used to document the reasoning behind the Trace.

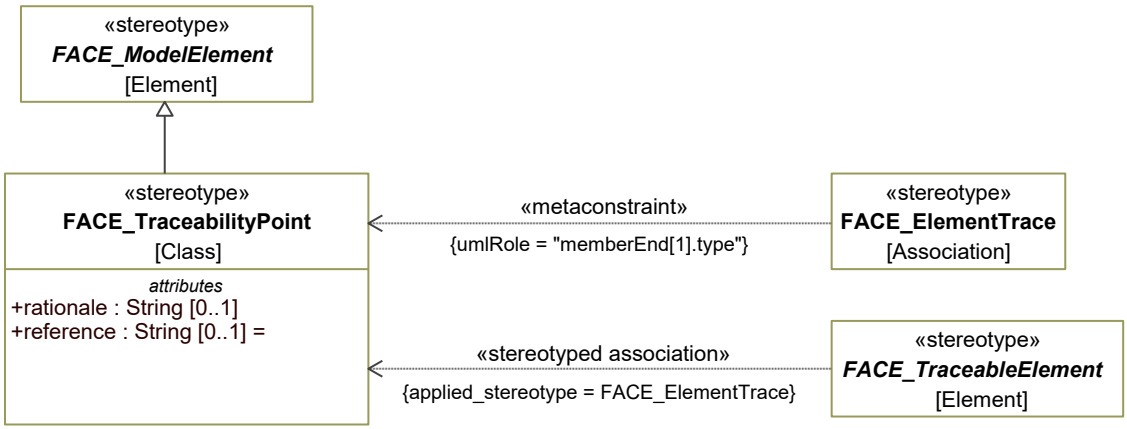

Figure 7-155: FACE_TraceabilityPoint

Attributes

rationale : String [0..1]

reference : String [0..1]

FACE_TraceableElement

Package: Traceability Model
isAbstract: Yes
Extension: Element

Description

A FACE_TraceableElement is used to capture traceability to other models.

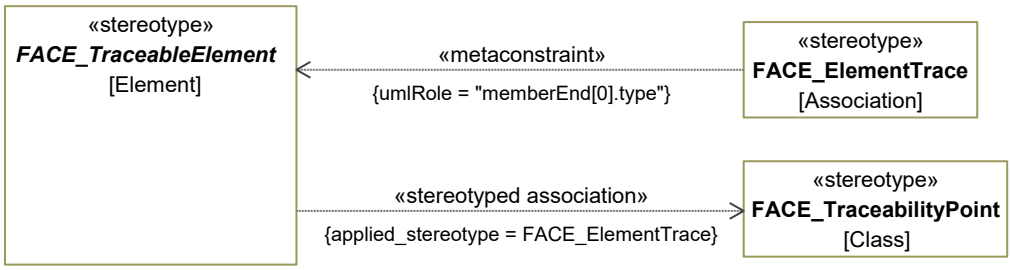

Figure 7-156: abstract FACE_TraceableElement

FACE_UoPTrace

Package: Traceability Model
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to connect FACE_UoPTraceabilitySets to their associated FACE_UnitOfPortability (UoPs).

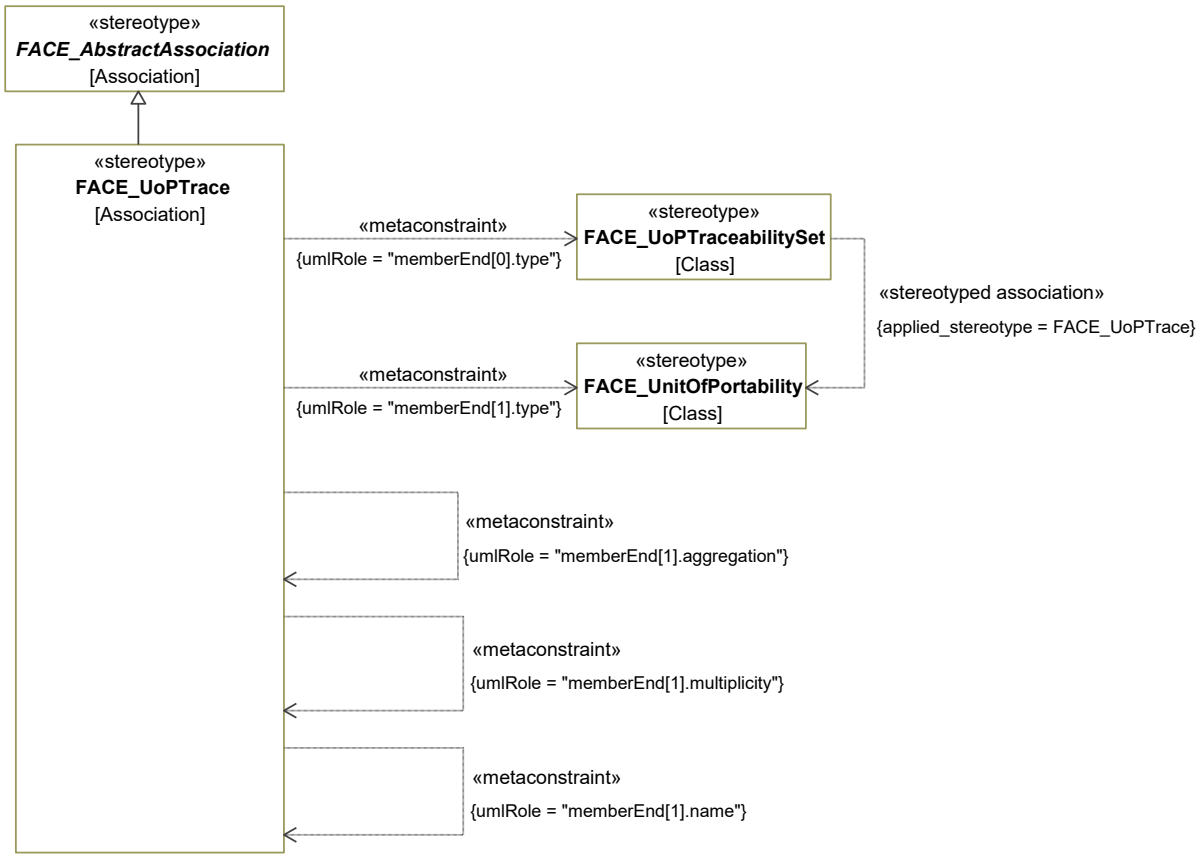

Figure 7-157: FACE_UoPTrace

Constraints

[1] FACE_UoPTrace.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_UoPTraceabilitySet».
[2] FACE_UoPTrace.memberEnd[1].aggregation	none
[3] FACE_UoPTrace.memberEnd[1].multiplicity	0..*
[4] FACE_UoPTrace.memberEnd[1].name	"uop"
[5] FACE_UoPTrace.memberEnd[1].type	The value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_UnitOfPortability».

FACE_UoPTraceabilitySet

Package: Traceability Model
isAbstract: No
Generalization: FACE_TraceabilityElement, FACE_TraceableElement
Extension: Class

Description

A FACE_UoPTraceabilitySet is used to relate a set of FACE_UnitOfPortability (UoPs) and/or FACE_AbstractUoPs to a set of FACE_TraceabilityPoints.

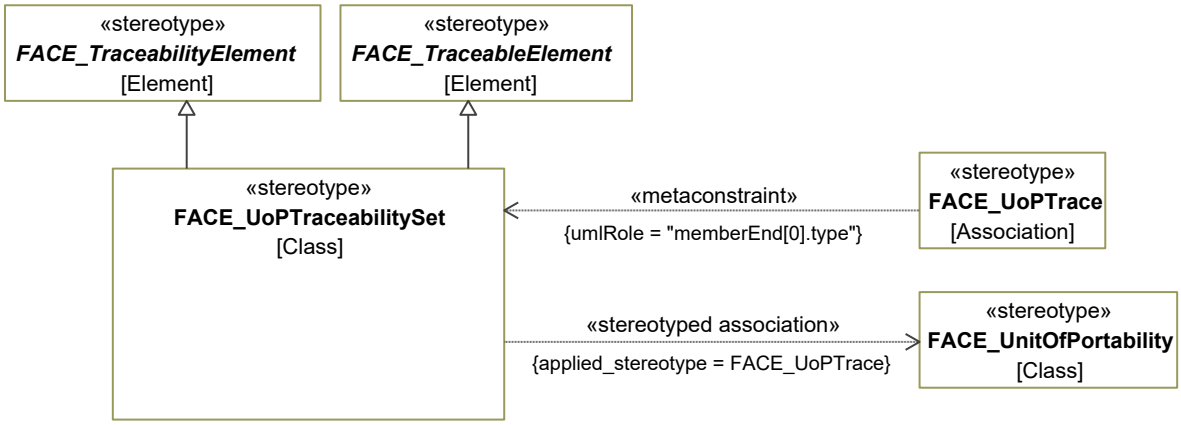

Figure 7-158: FACE_UoPTraceabilitySet

7.1.1.4 FACE_UAF_Profile::FACE Data Architecture::UoP Model

FACE_AbstractConnection

Package: UoP Model
isAbstract: No
Generalization: FACE_Element, FACE_TraceableElement
Extension: Class

Description

A FACE_AbstractConnection captures the input and output characteristics of a FACE_AbstractUoP by specifying data at a Logical or Conceptual level.

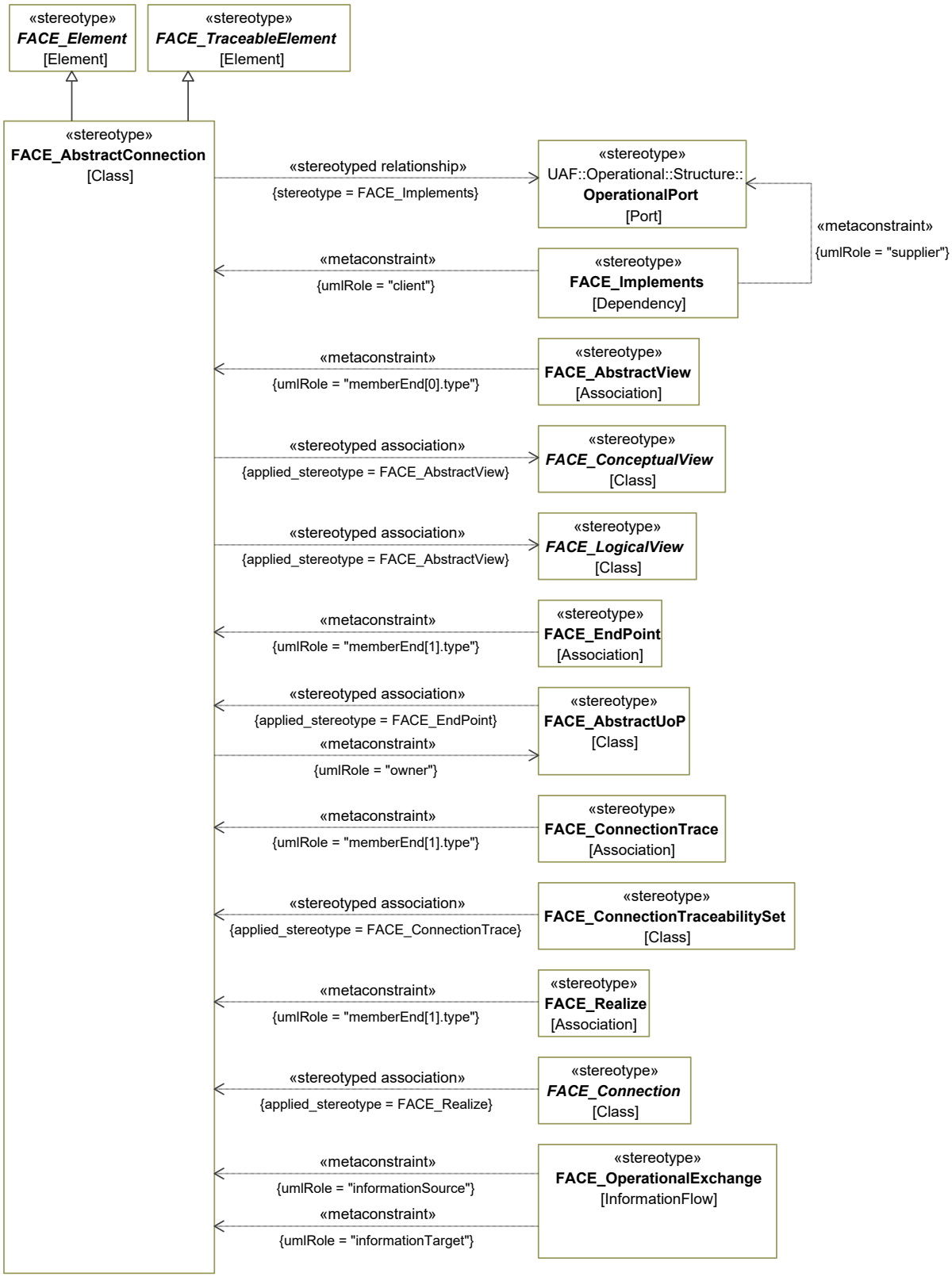

Figure 7-159: FACE_AbstractConnection

Constraints

[1] FACE_AbstractConnection.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_AbstractUoP»

FACE_AbstractUoP

Package: UoP Model
isAbstract: No
Generalization: FACE_UoPElement, FACE_TraceableElement
Extension: Class

Description

A FACE_AbstractUoP is used to capture the logical specification of a FACE_UnitOfPortability (UoP).

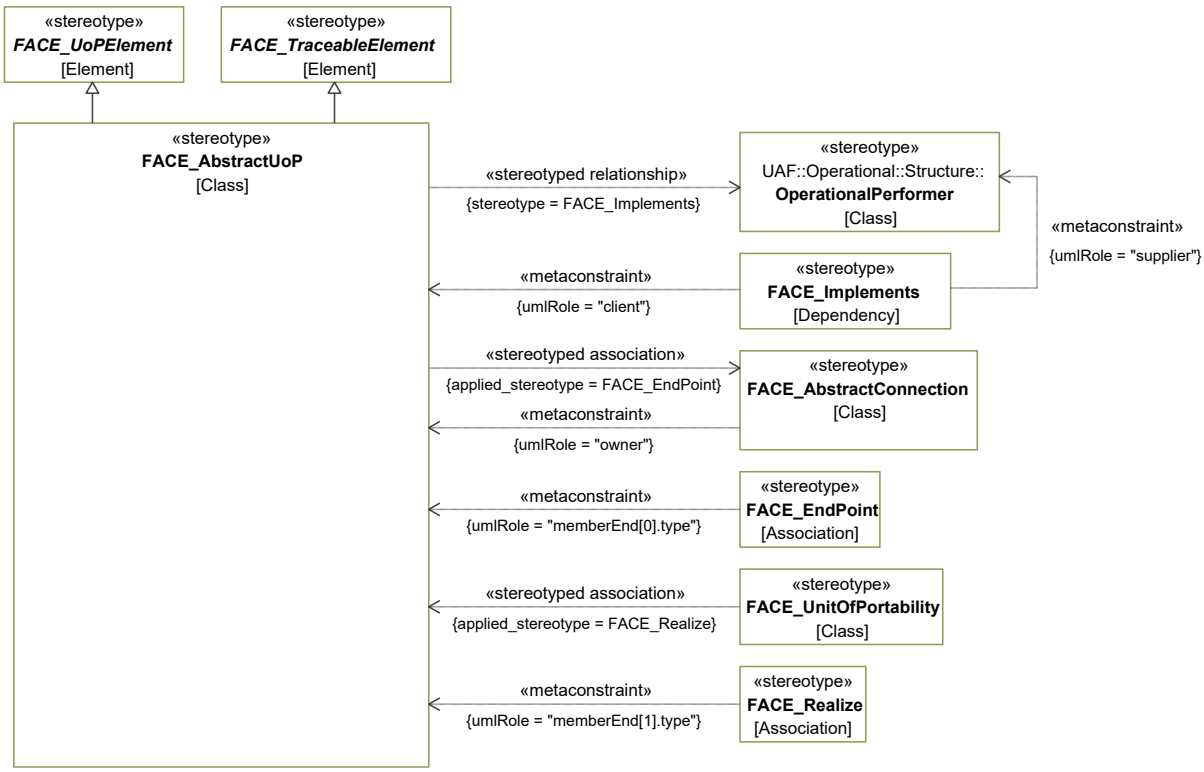

Figure 7-160: FACE_AbstractUoP

FACE Conformance/OCL Constraints

[1] FACE_AbstractUoP.onlyLogicalOrOnlyConceptual	A FACE_AbstractUoP must be entirely logical or entirely conceptual. (Its FACE_AbstractConnections all must have their logicalView set and conceptualView not set or all must have their conceptualView set and their logicalView not set.)

FACE_AbstractView

Package: UoP Model
isAbstract: No

Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to identify the FACE conceptual and FACE_LogicalViews that express the data exchanges for FACE_AbstractConnection components.

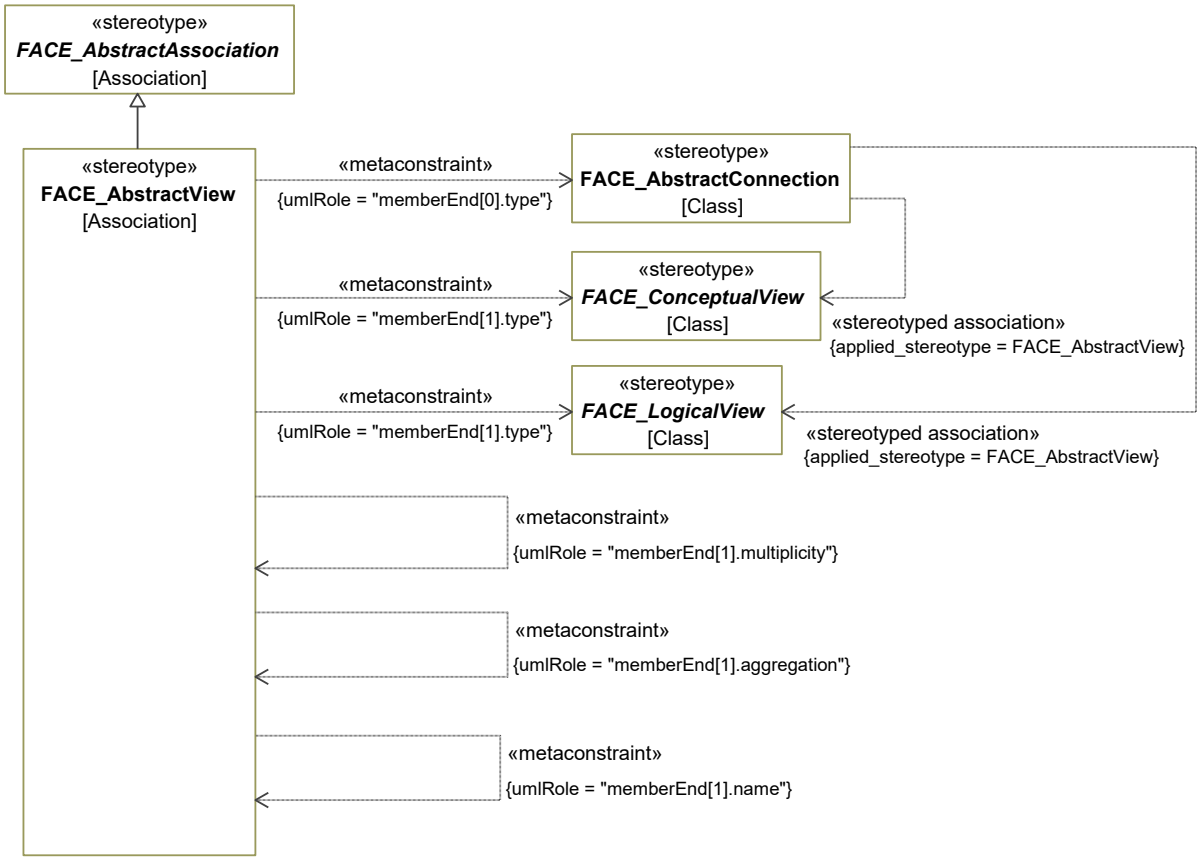

Figure 7-161: FACE_AbstractView

Constraints

[1] FACE_AbstractView.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_AbstractConnection».
[2] FACE_AbstractView.memberEnd[1].aggregation	none
[3] FACE_AbstractView.memberEnd[1].multiplicity	0..1
[4] FACE_AbstractView.memberEnd[1].name	Based on the stereotype of the memberEnd[1].type metaproperty: = Specialization of «FACE_ConceptualView», memberEnd[1].name is "conceptualView" = Specialization of «FACE_LogicalView», memberEnd[1].name is "logicalView"

[5] FACE_AbstractView.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by one of the following: Specialization of «FACE_ConceptualView» Specialization of «FACE_LogicalView»

FACE_BackingComponent

Package: UoP Model
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

The FACE_BackingComponent identifies the FACE_SupportingComponents that are required for a FACE_UnitOfPortability.

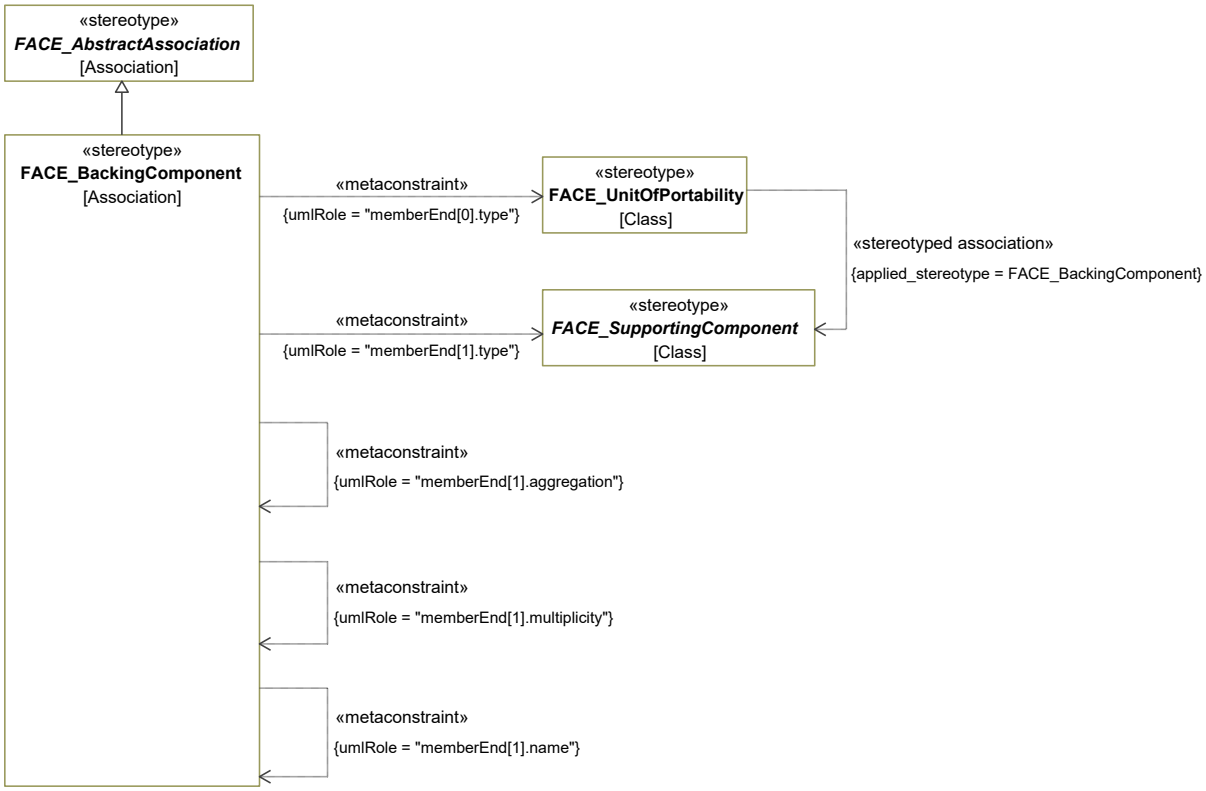

Figure 7-162: FACE_BackingComponent

Constraints

[1] FACE_BackingComponent.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_UnitOfPortability».
[2] FACE_BackingComponent.memberEnd[1].aggregation	none
[3] FACE_BackingComponent.memberEnd[1].multiplicity	0..*

[4] FACE_BackingComponent.memberEnd[1].name	"supportingComponent"
[5] FACE_BackingComponent.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_SupportingComponent».

FACE_ClientServerConnection

Package: UoP Model
isAbstract: No
Generalization: FACE_Connection

Description

A FACE_ClientServerConnection is a Request/Reply Connection as defined in Section 3.7 of the FACE Technical Standard, Edition 3.0.

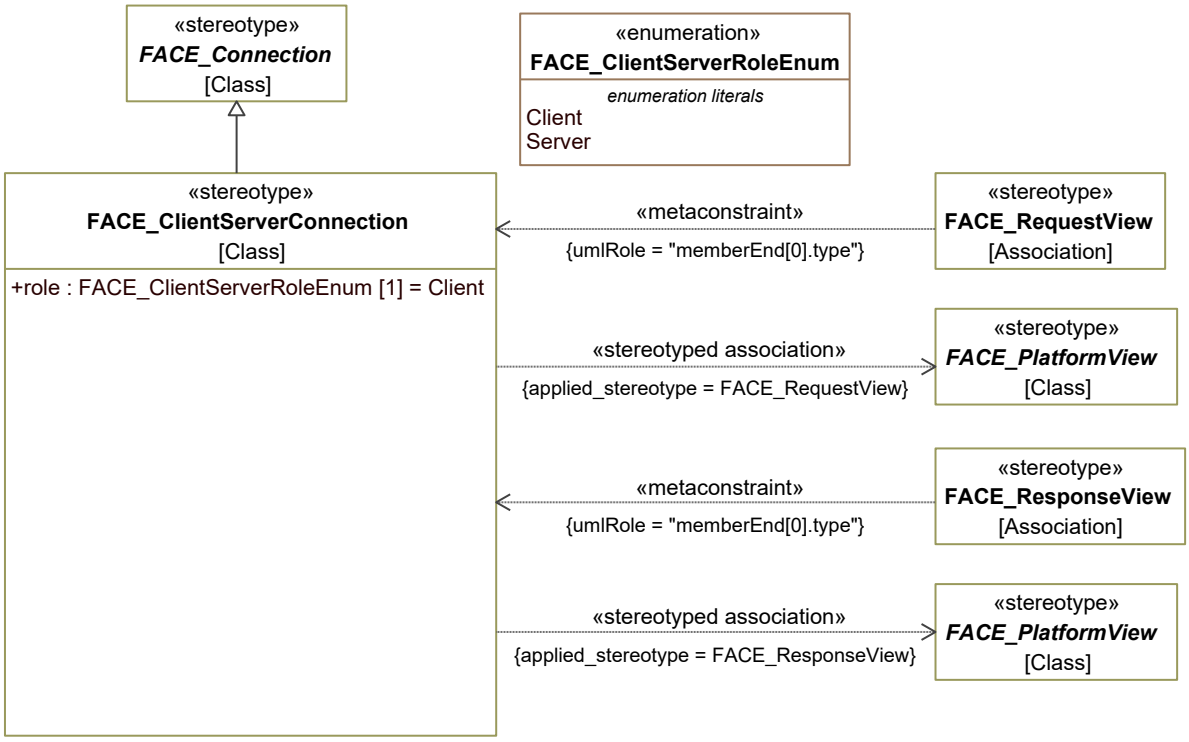

Figure 7-163: FACE_ClientServerConnection

Attributes

role : FACE_ClientServerRoleEnum [1]

FACE_ClientServerRoleEnum

Package: UoP Model
isAbstract: No

Description

Indicates the component role in a Client/Server communication pattern. Its enumeration literals are:

- Client -
- Server -

FACE_ComponentFramework

Package: UoP Model
isAbstract: No
Generalization: FACE_SupportingComponent
Extension: Class

Description

A FACE_ComponentFramework is an application framework as defined in Section 3.2.4 of the FACE Technical Standard, Edition 3.0.

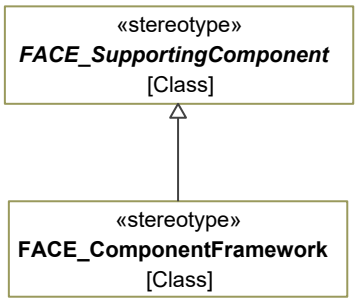

Figure 7-164: FACE_ComponentFramework

FACE_ComponentTypeEnum

Package: UoP Model
isAbstract: No

Description

Indicates the FACE-Specific component type of the component. Its enumeration literals are:

- PortableComponent -
- PlatformSpecificComponent -

FACE_Connection

Package: UoP Model
isAbstract: Yes
Generalization: FACE_Element, FACE_TraceableElement
Extension: Class

Description

A FACE_Connection is a communication endpoint on a FACE_UnitOfPortability (UoP). A FACE_Connection is either a Publisher, Subscriber, Client, or Server. The messageType specifies the FACE_PlatformView that is transmitted through the

endpoint. If period is not specified, the endpoint is aperiodic. If period is specified, the value is the period of the endpoint in seconds.

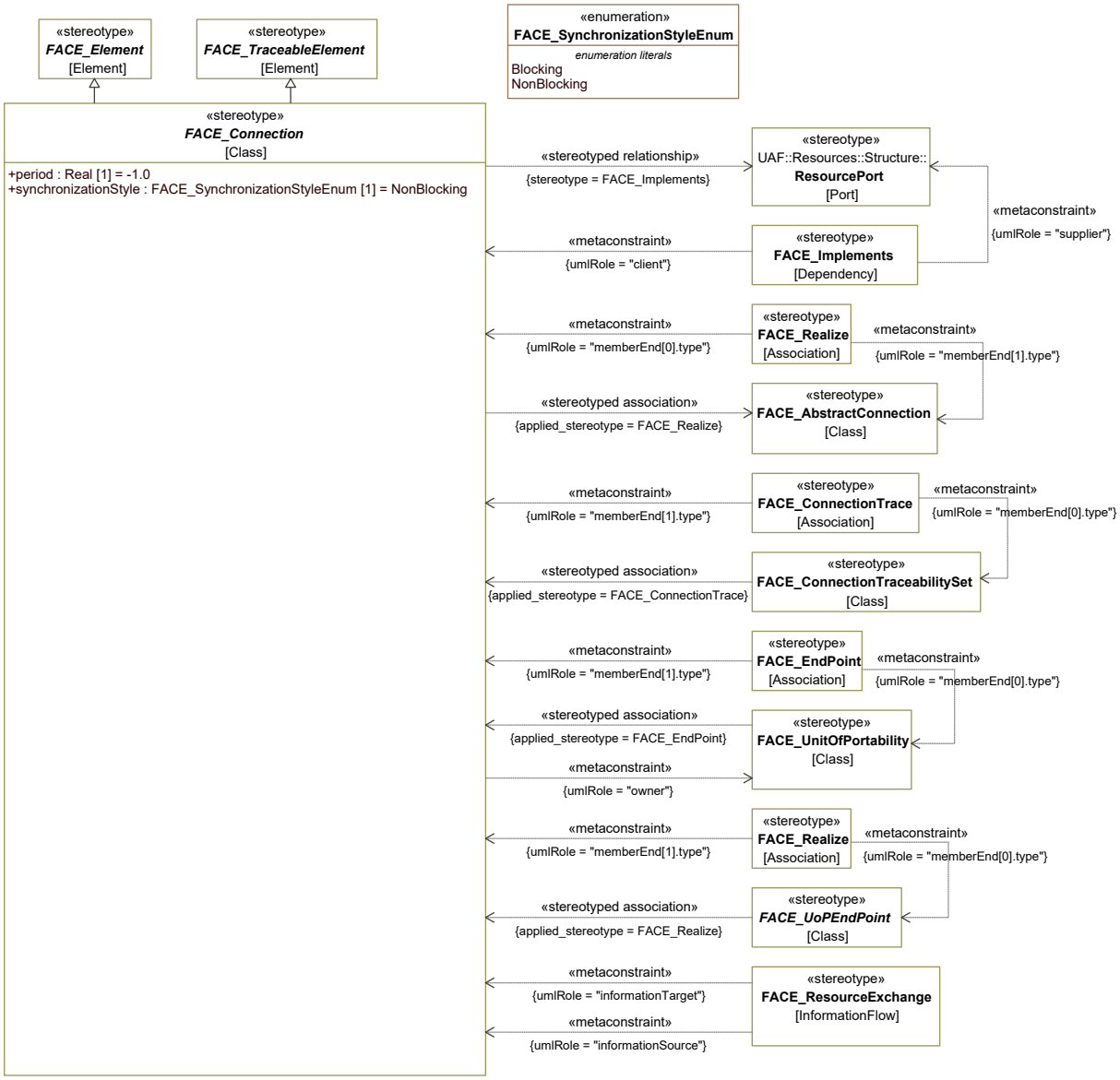

Figure 7-165: abstract FACE_Connection

Attributes

period : Real [1]

synchronizationStyle : FACE_SynchronizationStyleEnum [1]

Constraints

[1] FACE_Connection.owner	Elements that are stereotyped by specializations of this stereotype may only be contained in (owned by) elements with the stereotype «FACE_UnitOfPortability»

FACE Conformance/OCL Constraints

[1] FACE_Connection.realizationTypeConsistent	If a FACE_Connection realizes an FACE_AbstractConnection, its requestType or responseType or both (for FACE_ClientServerConnections) or its messageType (for FACE_PubSubConnections) must realize either the FACE_AbstractConnection's logicalView or a logical View that must realize the FACE_AbstractConnection's conceptualView.

FACE_DesignAssuranceLevelEnum

Package: UoP Model
isAbstract: No

Description

Indicates the safety and hazard Design Assurance Level (DAL) assigned to a component. Its enumeration literals are:

- A -
- B -
- C -
- D -
- E -

FACE_DesignAssuranceStandardEnum

Package: UoP Model
isAbstract: No

Description

Indicates the FACE-pertinent safety-critical Design Assurance Standard that applies to a component. Its enumeration literals are:

- DO_178B_ED_12B -
- DO_178C_ED_12C -

FACE_LanguageRunTime

Package: UoP Model
isAbstract: No
Generalization: FACE_SupportingComponent
Extension: Class

Description

A FACE_LanguageRunTime is a language run-time as defined in Section 3.2.3 of the FACE Technical Standard, Edition 3.0.

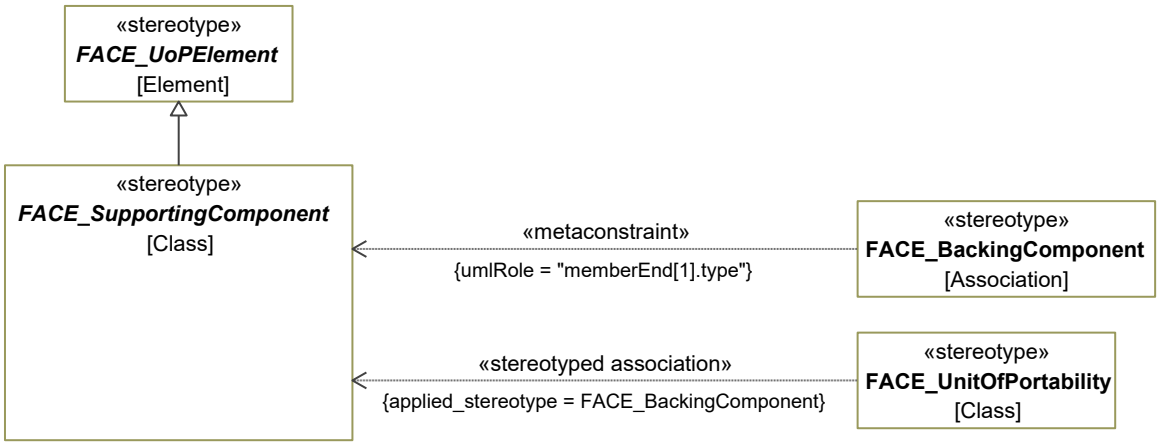

Figure 7-166: FACE_LanguageRunTime

FACE_LifeCycleManagementPort

Package: UoP Model
isAbstract: No
Generalization: FACE_ModelElement
Extension: Class

Description

A FACE_LifeCycleManagementPort is used to define the life-cycle interface for the component. The messageExchangeType attribute defines the direction of the life-cycle message relative to the FACE_UnitOfPortability (UoP).

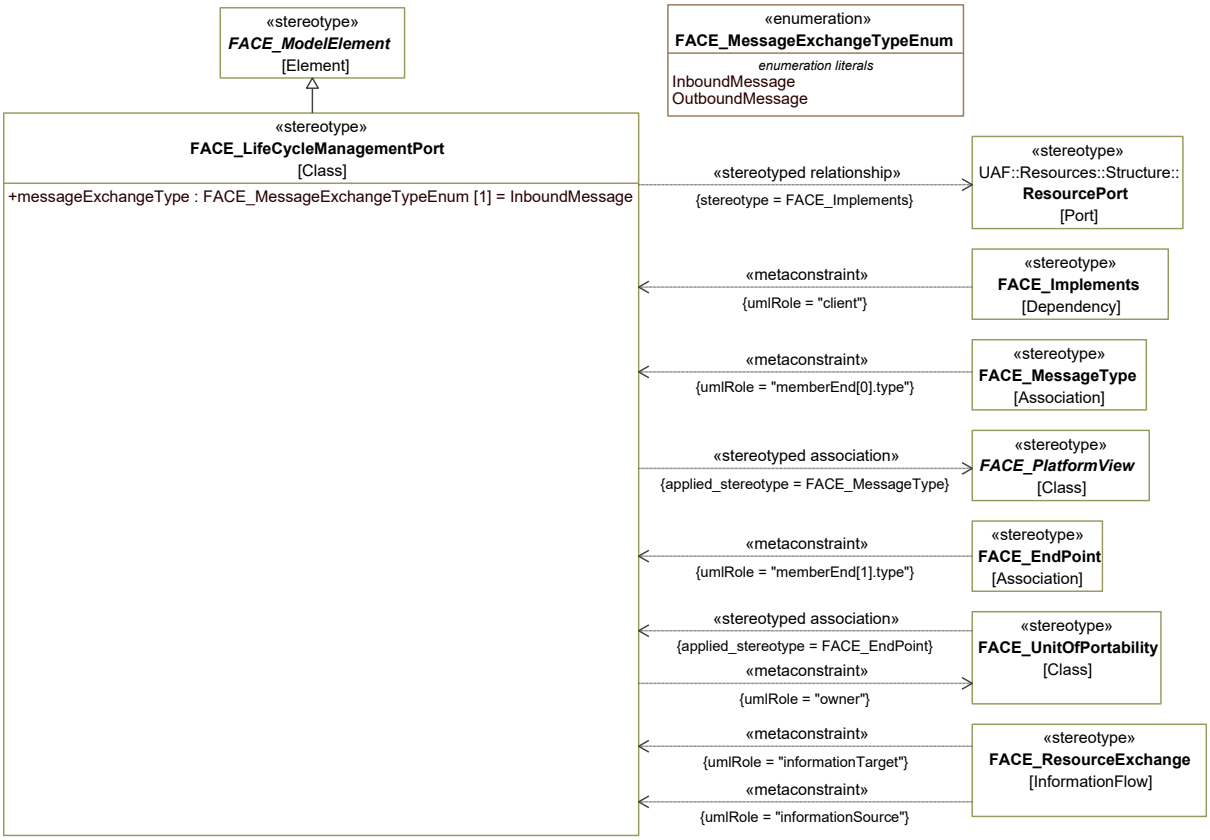

Figure 7-167: FACE_LifeCycleManagementPort

Attributes

messageExchangeType : FACE_MessageExchangeTypeEnum [1]

Constraints

[1] FACE_LifeCycleManagementPort.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_UnitOfPortability»

FACE_MessageExchangeTypeEnum

Package: UoP Model
isAbstract: No

Description

The FACE_MessageExchangeTypeEnum enumeration captures the options for the message exchange type of a FACE_UnitOfPortability (UoP) port as defined by the TS Interface. Its enumeration literals are:

- InboundMessage -
- OutboundMessage -

FACE_PartitionTypeEnum

Package: UoP Model
isAbstract: No

Description

The FACE_PartitionTypeEnum enumeration captures the OS API types for a FACE_UnitOfPortability (UoP) as defined by the FACE Operating System Segment (OSS). Its enumeration literals are:

- POSIX -
- ARINC653 -

FACE_ProfileEnum

Package: UoP Model
isAbstract: No

Description

The FACE_FaceProfileEnum enumeration captures the OS API subsets for a FACE_UnitOfPortability (UoP) as defined by the Operating System Segment (OSS). Its enumeration literals are:

- GeneralPurpose -
- Security -
- SafetyBase -
- SafetyExtended -

FACE_ProgrammingLanguageEnum

Package: UoP Model
isAbstract: No

Description

The FACE_ProgrammingLanguageEnum enumeration captures the options for programming language API bindings as defined by Section 3.14 of the FACE Technical Standard, Edition 3.0. Its enumeration literals are:

- C -
- CPP -
- Java -
- Ada -

FACE_PubSubConnection

Package: UoP Model
isAbstract: Yes
Generalization: FACE_Connection

Description

A FACE_PubSubConnection is a FACE_QueuingConnection or a FACE_SingleInstanceMessageConnection. The messageExchangeType attribute defines the direction of the message relative to the FACE_UnitOfPortability (UoP).

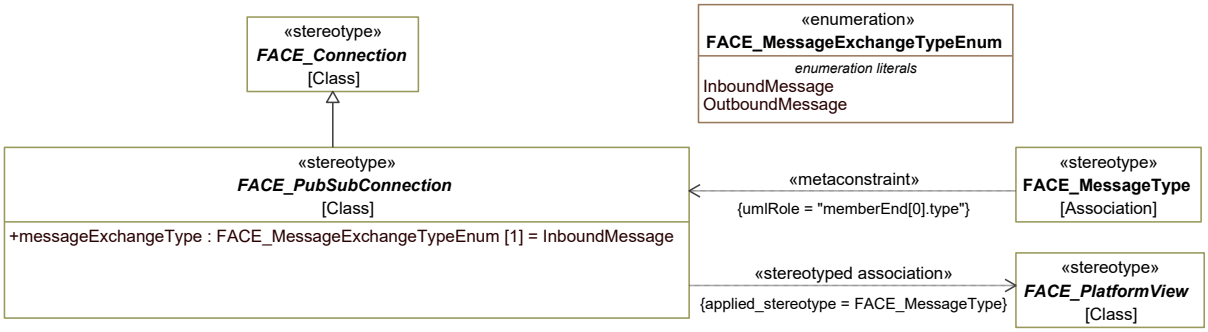

Figure 7-168: abstract FACE_PubSubConnection

Attributes

messageExchangeType : FACE_MessageExchangeTypeEnum [1]

FACE_QueuingConnection

Package: UoP Model
isAbstract: No
Generalization: FACE_PubSubConnection

Description

A FACE_QueuingConnection is a FACE_PubSubConnection that supports buffering/queuing as defined in Section 3.8 of the FACE Technical Standard, Edition 3.0.

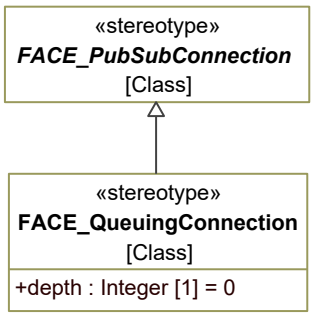

Figure 7-169: FACE_QueuingConnection

Attributes

depth : Integer [1]

FACE Conformance/OCL Constraints

[1] FACE_QueuingConnection.depthValid A FACE_QueuingConnection's queue depth must be greater than zero.

FACE_RAMMemoryRequirements

Package: UoP Model
isAbstract: No
Generalization: FACE_ModelElement
Extension: Class

Description

A FACE_RAMMemoryRequirements defines memory resources required by a FACE_UnitOfPortability (UoP).

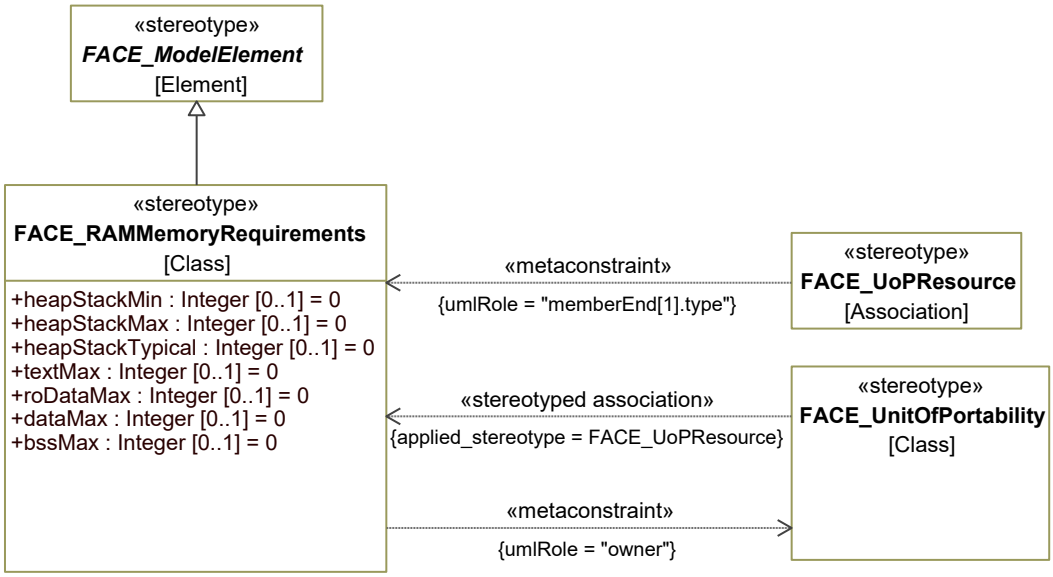

Figure 7-170: FACE_RAMMemoryRequirements

Attributes

bssMax : Integer [0..1]

dataMax : Integer [0..1]

heapStackMax : Integer [0..1]

heapStackMin : Integer [0..1]

heapStackTypical : Integer [0..1]

roDataMax : Integer [0..1]

textMax : Integer [0..1]

Constraints

[1] FACE_RAMMemoryRequirements.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_UnitOfPortability»

FACE_RequestView

Package: UoP Model
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to identify the FACE_PlatformView that specifies the request message for a FACE Client/Server connection.

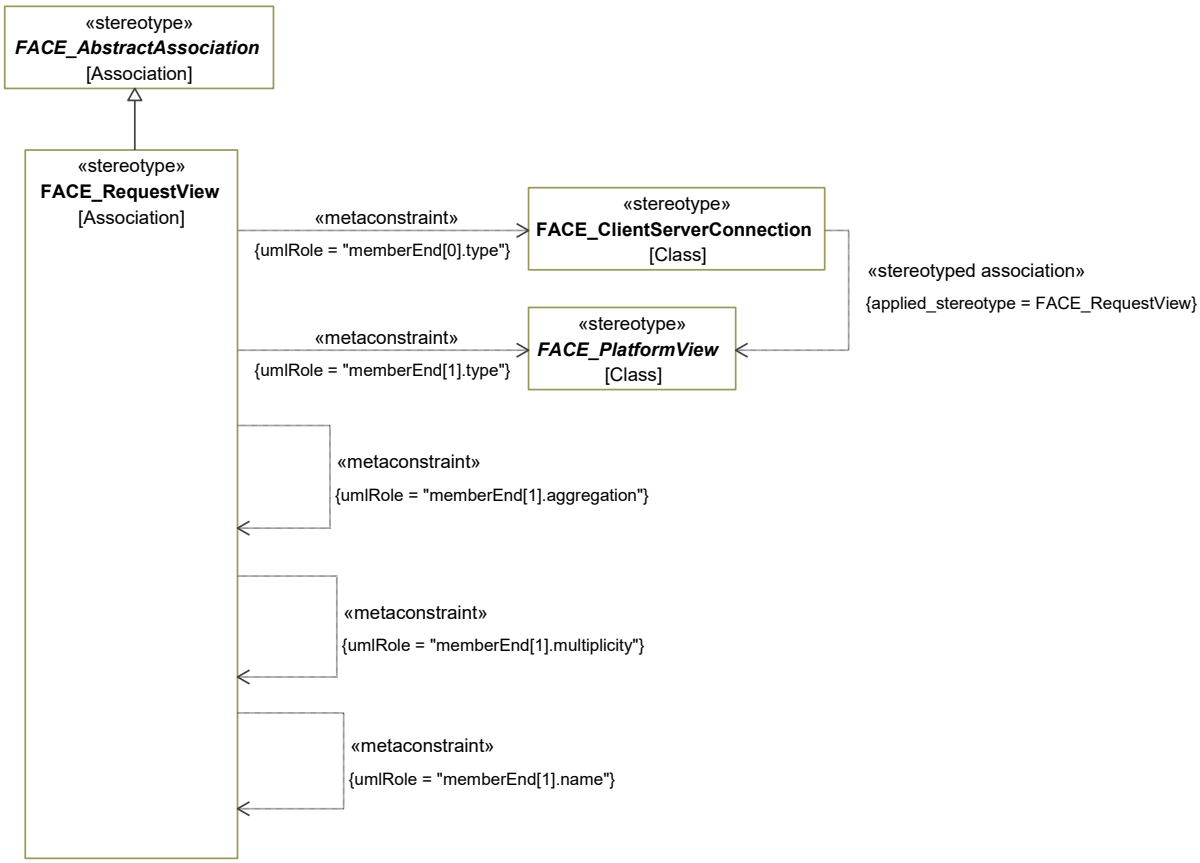

Figure 7-171: FACE_RequestView

Constraints

[1] FACE_RequestView.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_ClientServerConnection».
[2] FACE_RequestView.memberEnd[1].aggregation	none
[3] FACE_RequestView.memberEnd[1].multiplicity	1
[4] FACE_RequestView.memberEnd[1].name	"requestType"

[5] FACE_RequestView.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_PlatformView».

FACE_ResponseView

Package: UoP Model
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to identify the FACE_PlatformView that specifies the expected response message for a FACE Client/Server connection.

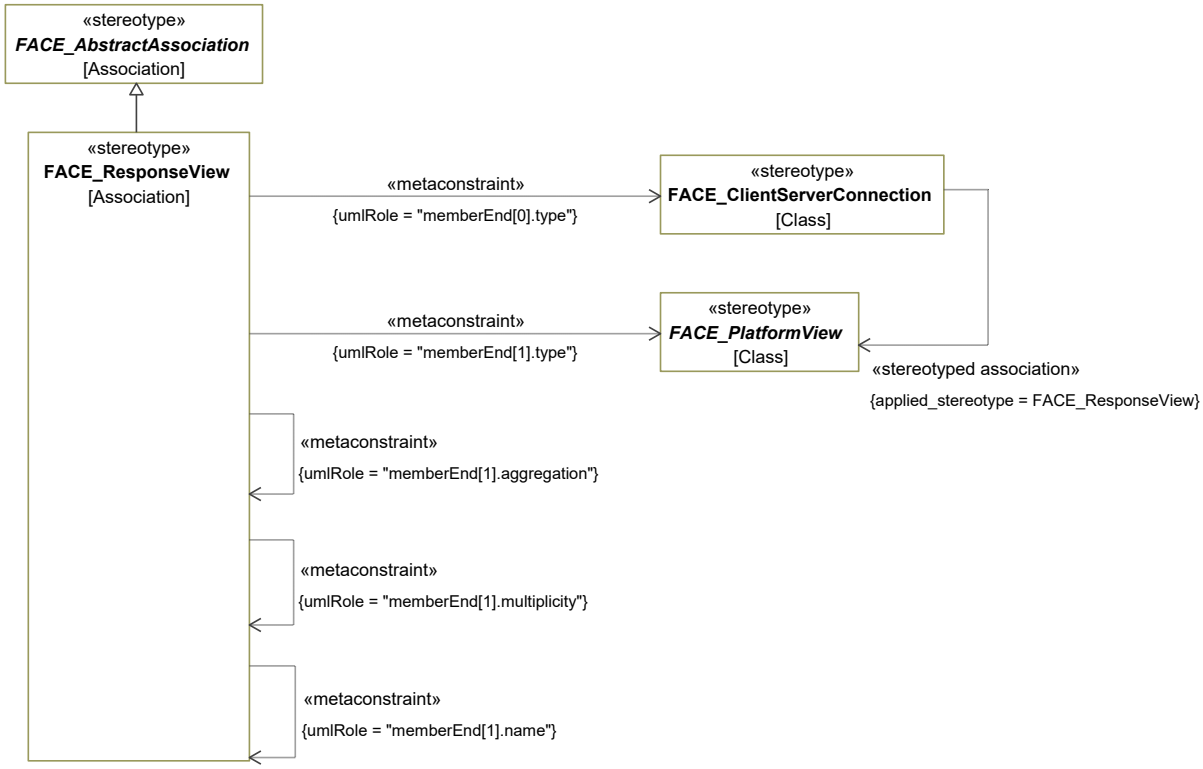

Figure 7-172: FACE_ResponseView

Constraints

[1] FACE_ResponseView.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_ClientServerConnection».
[2] FACE_ResponseView.memberEnd[1].aggregation	none
[3] FACE_ResponseView.memberEnd[1].multiplicity	1
[4] FACE_ResponseView.memberEnd[1].name	"responseType"

[5] FACE_ResponseView.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_PlatformView».

FACE_SingleInstanceMessageConnection

Package: UoP Model
isAbstract: No
Generalization: FACE_PubSubConnection

Description

A FACE_SingleInstanceMessageConnection is a FACE_PubSubConnection that supports single instance messaging as defined in Section 3.8 of the FACE Technical Standard, Edition 3.0.

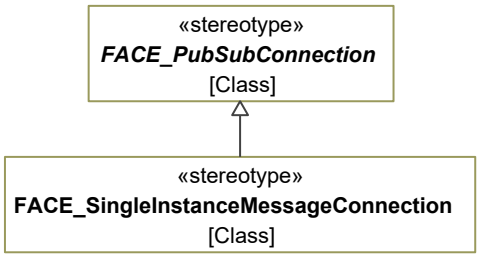

Figure 7-173: FACE_SingleInstanceMessageConnection

FACE_SupportingComponent

Package: UoP Model
isAbstract: Yes
Generalization: FACE_UoPElement
Extension: Class

Description

A FACE_SupportingComponent is a LanguageRunTime or ApplicationFramework. The version attribute is the version of the FACE_SupportingComponent.

Figure 7-174: abstract FACE_SupportingComponent

Attributes

version : String [1]

FACE_SynchronizationStyleEnum

Package: UoP Model
isAbstract: No

Description

The FACE_SynchronizationStyleEnum enumeration captures the options for the synchronization style of a FACE_UnitOfPortability (UoP) port as defined by the Transport Services (TS) Interface. Its enumeration literals are:

- Blocking -
- NonBlocking -

FACE_Thread

Package: UoP Model
isAbstract: No
Generalization: FACE_ModelElement
Extension: Class

Description

A FACE_Thread defines the properties for the scheduling of a thread.

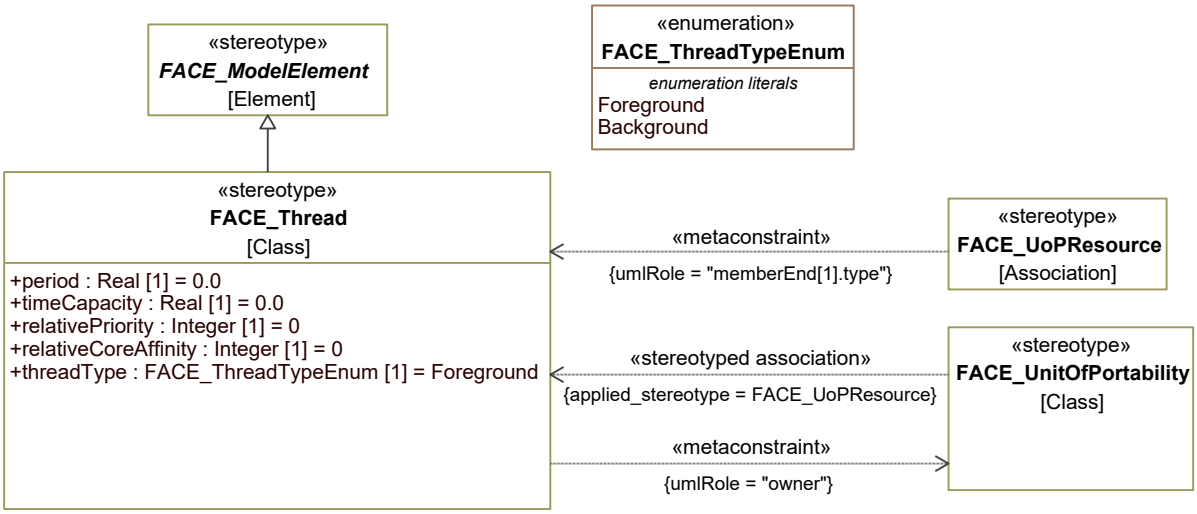

Figure 7-175: FACE_Thread

Attributes

period : Real [1]

relativeCoreAffinity : Integer [1]

relativePriority : Integer [1]

threadType : FACE_ThreadTypeEnum [1]

timeCapacity : Real [1]

Constraints

[1] FACE_Thread.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_UnitOfPortability»

FACE_ThreadTypeEnum

Package: UoP Model
isAbstract: No

Description

Indicates the thread runtime foreground/background characteristic for a component. Its enumeration literals are:

- Foreground -
- Background -

FACE_UnitOfPortability

Package: UoP Model
isAbstract: No
Generalization: FACE_UoPElement, FACE_TraceableElement
Extension: Class

Description

A FACE_UnitOfPortability is a PlatformSpecificComponent or PortableComponent.

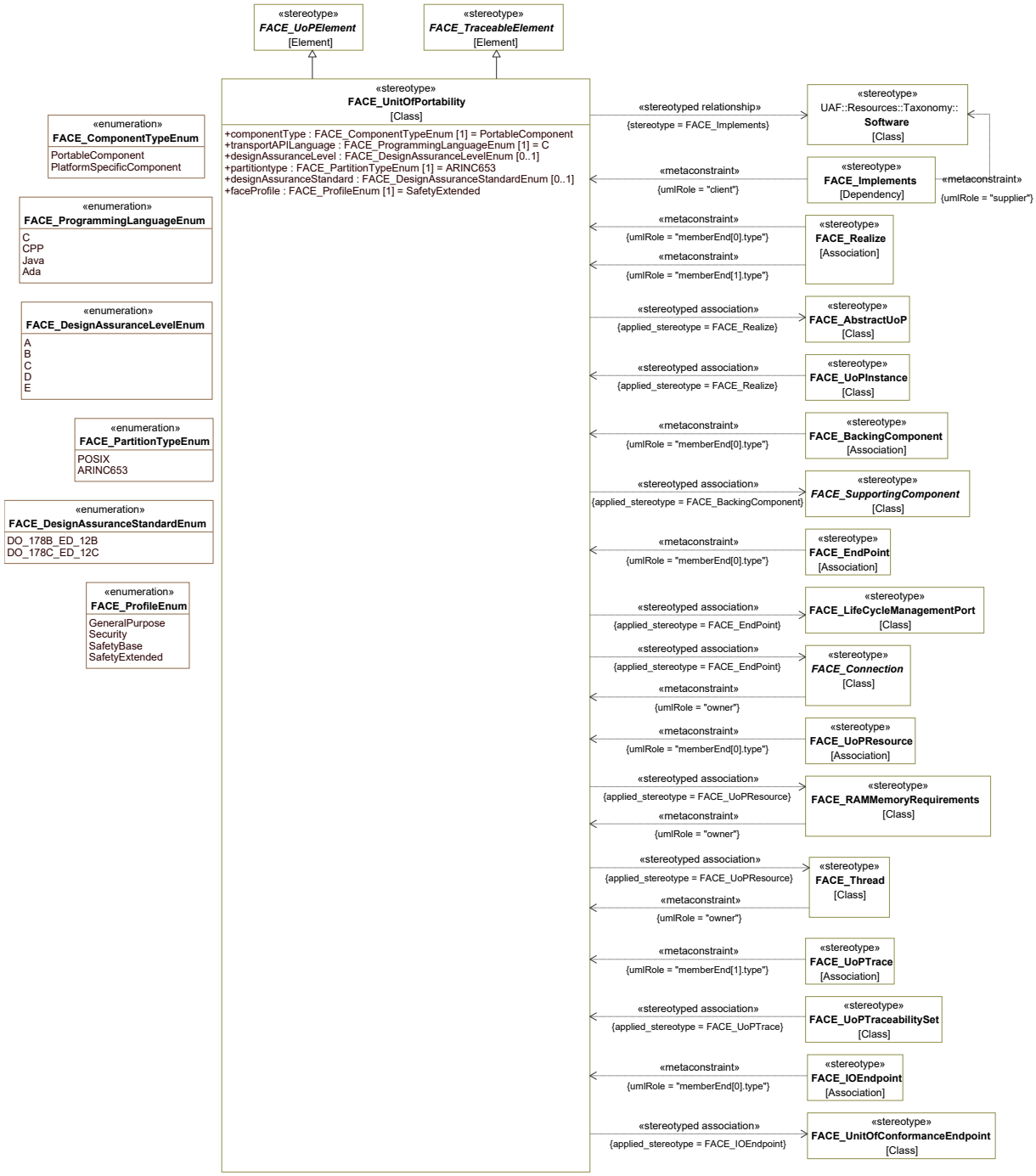

Figure 7-176: FACE_UnitOfPortability

Attributes

componentType : FACE_ComponentTypeEnum [1]

designAssuranceLevel : FACE_DesignAssuranceLevelEnum [0..1]

designAssuranceStandard : FACE_DesignAssuranceStandardEnum [0..1]

faceProfile : FACE_ProfileEnum [1]

partitiontype : FACE_PartitionTypeEnum [1]

transportAPILanguage : FACE_ProgrammingLanguageEnum [1]

FACE Conformance/OCL Constraints

[1] FACE_UnitOfPortability.connectionsConsistentWithUoPRealization	If a FACE_UnitOfPortability "A" realizes a FACE_AbstractUoP "B", then A and B must have the same number of connections, and every FACE_Connection in A must realize a unique FACE_AbstractConnection in B. If a FACE_UnitOfPortability does not realize a FACE_AbstractUoP, none of its FACE_Connections may realize.

FACE_UoPElement

Package: UoP Model
isAbstract: Yes
Generalization: FACE_Element

Description

A FACE_UoPElement is the root type for defining the component elements of the UoPModel.

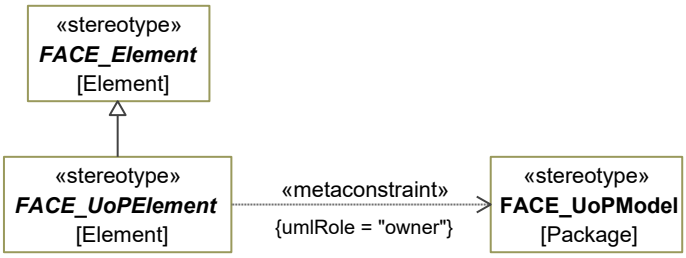

Figure 7-177: abstract FACE_UoPElement

Constraints

[1] FACE_UoPElement.owner	Elements that are stereotyped by specializations of this abstract stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_UoPModel»

FACE Conformance/OCL Constraints

[1] FACE_UoPElement.hasUniqueName	All FACE UoP Elements must have a unique name.

FACE_UoPResource

Package: UoP Model
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

Used to identify system requirements for FACE_UnitOfPortability (UoP) components.

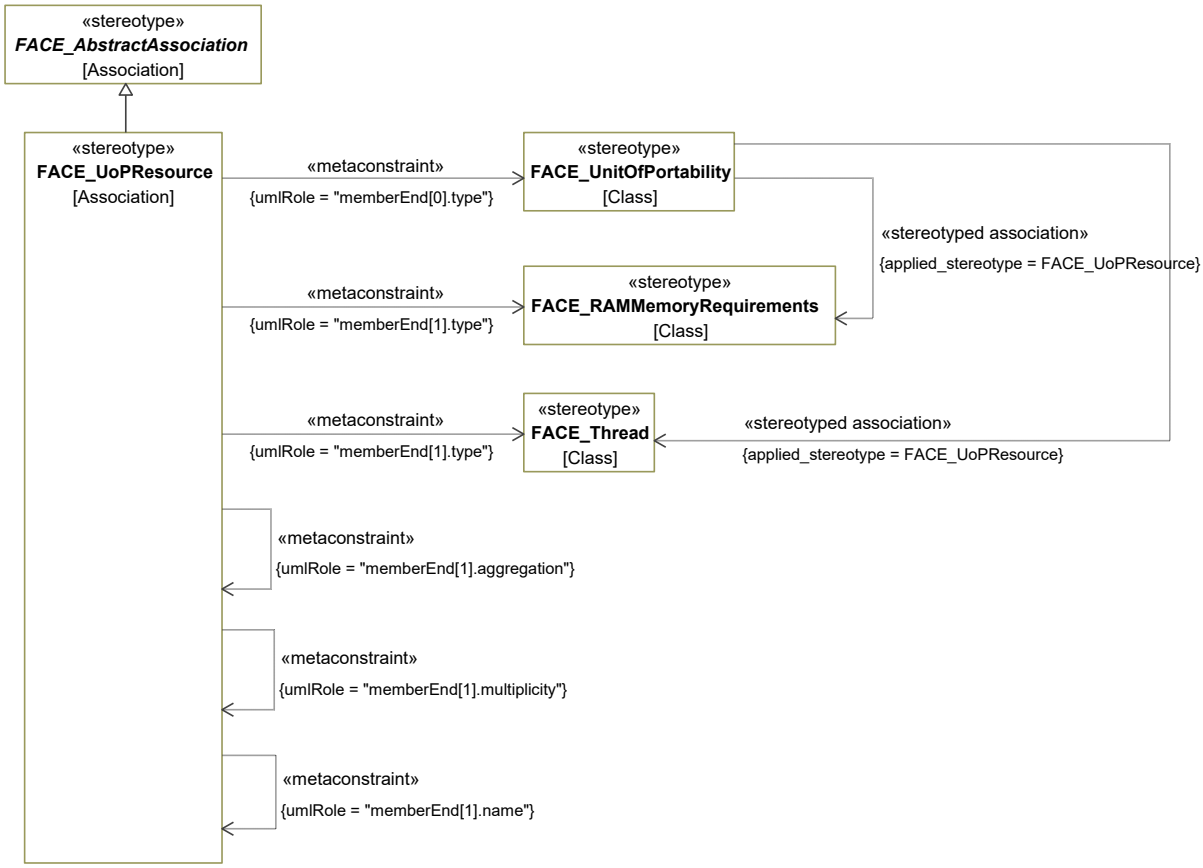

Figure 7-178: FACE_UoPResource

Constraints

[1] FACE_UoPResource.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_UnitOfPortability».
[2] FACE_UoPResource.memberEnd[1].aggregation	composite
[3] FACE_UoPResource.memberEnd[1].multiplicity	Based on the EndPoint.memberEnd[1].type value's stereotype: = «FACE_RAMMemoryRequirements», memberEnd[1].multiplicity must be 1 = «FACE_Thread», memberEnd[1].multiplicity must be 1..*
[4] FACE_UoPResource.memberEnd[1].name	Based on the EndPoint.memberEnd[1].type value's stereotype:

	= «FACE_RAMMemoryRequirements», memberEnd[1].name must be "memoryRequirements" = «FACE_Thread», memberEnd[1].name must be "thread"
[5] FACE_UoPResource.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by one of the following: «FACE_RAMMemoryRequirements» «FACE_Thread»

7.1.2 FACE_UAF_Profile::UAF_FACE_Extended_Stereotypes

This package contains stereotypes for elements not found in the FACE metamodel but supplement the FACE metamodel with elements that recognize the larger context of a UAF system-of-systems. The supplemental elements either represent FACE segments that are not explicitly represented in the FACE metamodel or provide connection between FACE Components and other components of the system-of-systems. The FACE_Implements «stereotyped relationship» dependencies in the steretoype defintions express the correspondence between FACE and UAF metatypes, with additional constraints for the application of FACE stereotypes.

FACE_Implements

Package: UAF_FACE_Extended_Stereotypes
isAbstract: No
Extension: Dependency

Description

This dependency indicates that the referencing FACE element is an implementation of the referenced UAF architectural element. This dependency and its constraints constitute the mapping from FACE stereotyped elements to UAF stereotyped elements.

The allowed dependencies in this stereotype include some implementation relationships that cross metatypes. Because the profile for the FACE adheres as closely as possible to the FACE metamodel, the type of a FACE profile element might differ from its corresponding application in a UAF context. The use of Dependency relationships to indicate implementation enables the representation of the intent of the FACE element correctly in the UAFP context.

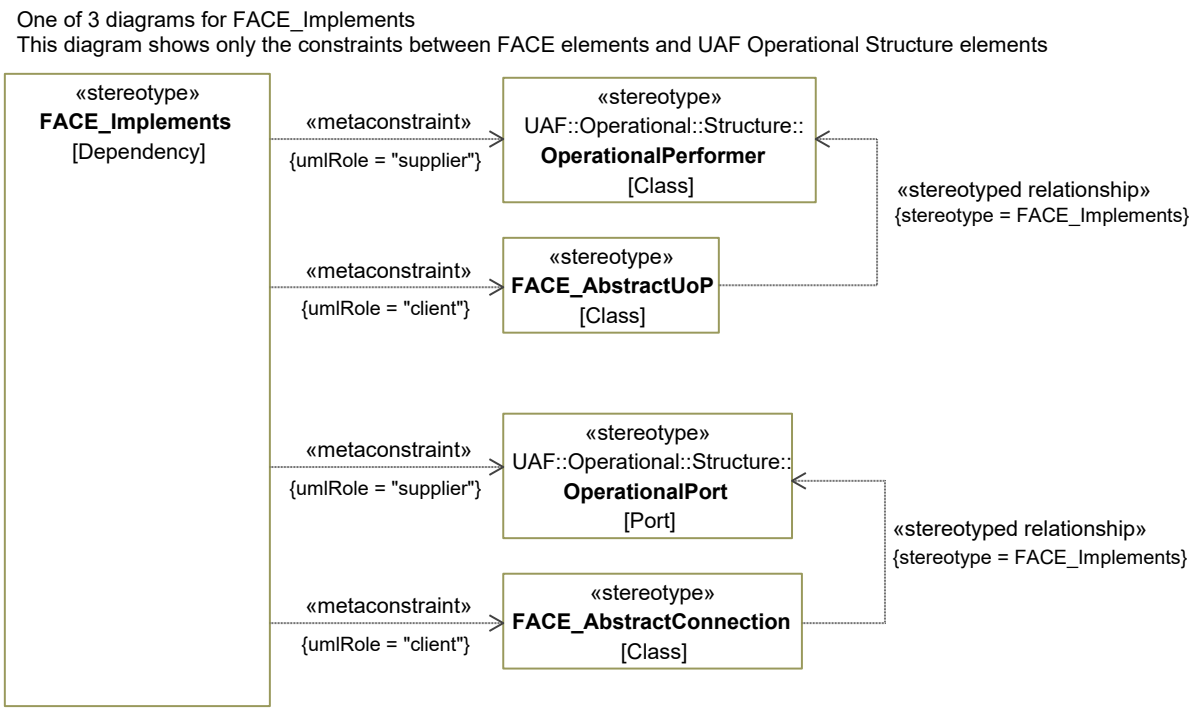

Figure 7-179: FACE_Implements diagram 1 of 3

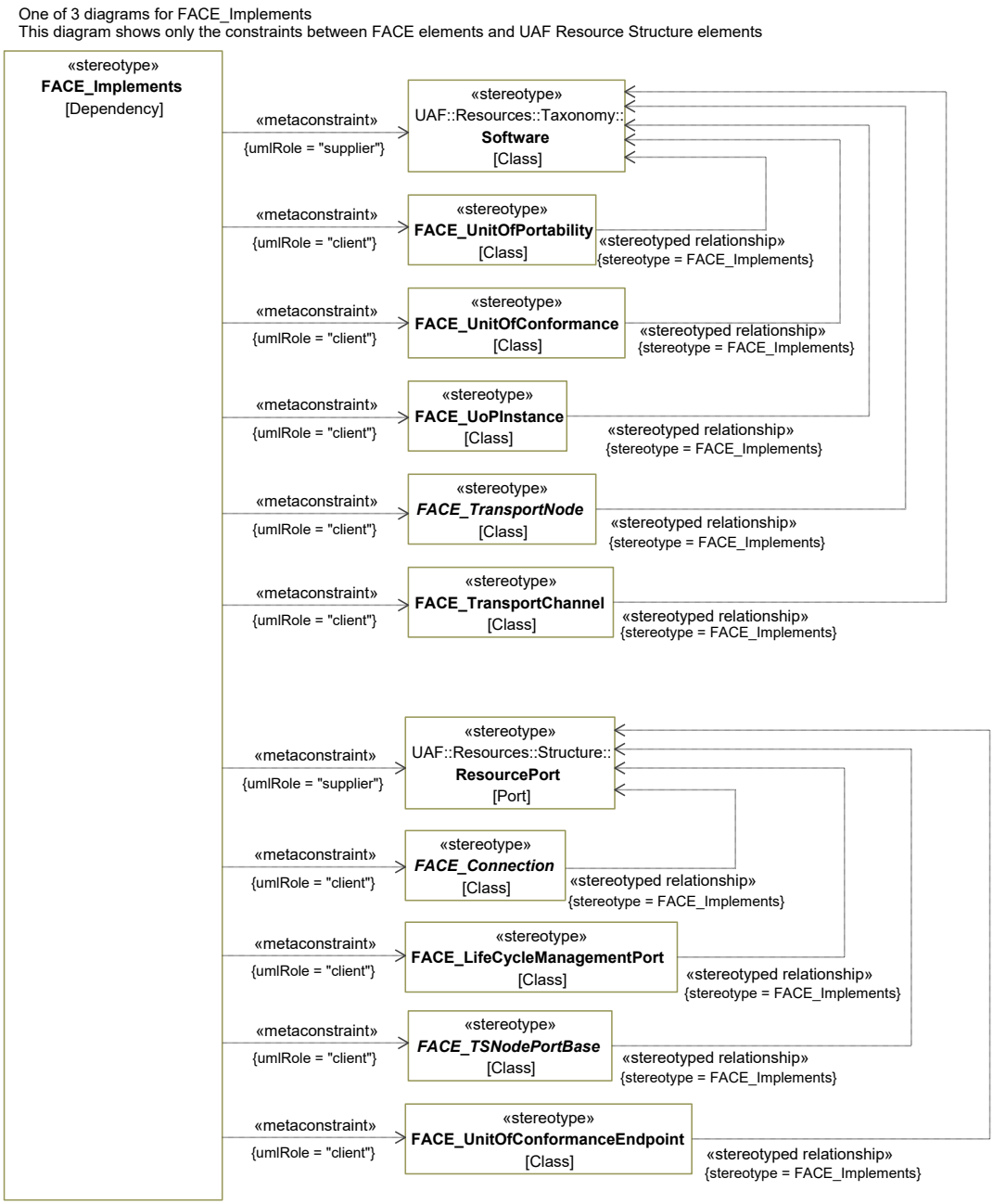

Figure 7-180: FACE_Implements, diagram 2 of 3

One of 3 diagrams for FACE_Implements
This diagram shows only the constraints between FACE elements and UAF data and connection related elements

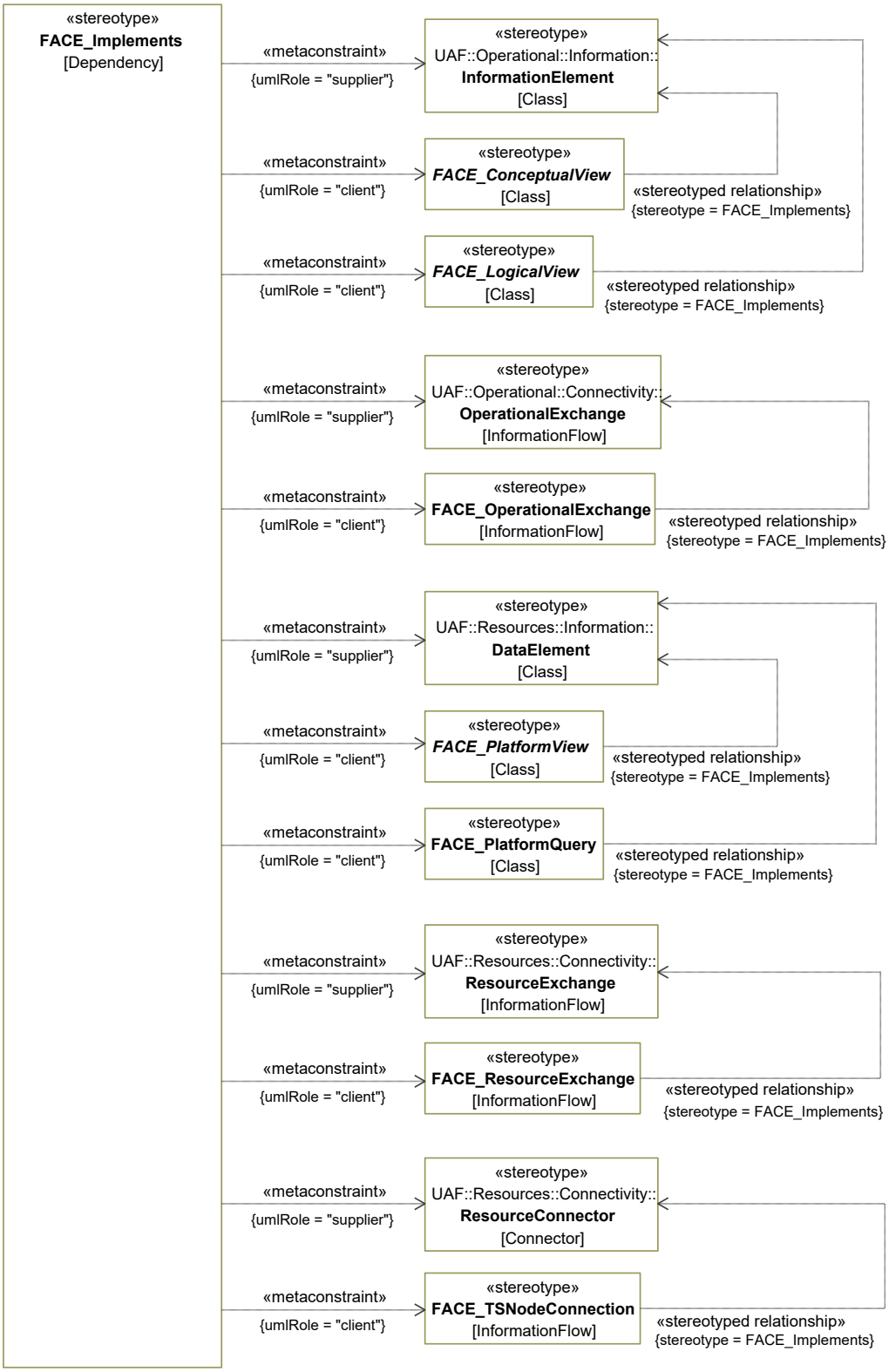

Figure 7-181: FACE_Implements, diagram 3 of 3

Constraints

[1] FACE_Implements.client	Value for the client metaproperty must be stereotyped by one of the following: «FACE_AbstractUoP» «FACE_AbstractConnection» «FACE_UnitOfPortability» «FACE_UnitOfConformance» «FACE_UoPInstance» Specializations of «FACE_TransportNode» «FACE_TransportChannel» Specializations of «FACE_Connection» «FACE_LifeCycleManagementPort» Specializationa of «FACE_TSNodePortBase» «FACE_UnitOfConformanceEndpoint» Specializations of «FACE_ConceptualView» Specializations of «FACE_LogicalView» «FACE_OperationalExchange» Specializations of «FACE_PlatformView» «FACE_PlatformQuery» «FACE_TSNodeConnection» «FACE_ResourceExchange»
[2] FACE_Implements.supplier	Based on the stereotype of the client metaproperty: = «FACE_AbstractUoP», the supplier metaproperty must be stereotyped by (UAF::Operational::Structure) «OperationalPerformer» = «FACE_AbstractConnection», the supplier metaproperty must be stereotyped by (UAF::Operational::Structure) «OperationalPort» = «FACE_UnitOfPortability», «FACE_UnitOfConformance», «FACE_UoPInstance», a specialization of «FACE_TransportNode», or «FACE_TransportChannel», the supplier metaproperty must be stereotyped by (UAF::Resources::Taxonomy) «Software» = A specialization of «FACE_Connection», «FACE_LifeCycleManagementPort», a specialization of «FACE_TSNodePortBase», or «FACE_UnitOfConformanceEndpoint», the supplier metaproperty must be stereotyped by (UAF::Resources::Structure) «ResourcePort» = A specialization of «FACE_ConceptualView», or a specialization of «FACE_LogicalView», the supplier metaproperty must be stereotyped by (UAF::Operational::Information) «InformationElement» = «FACE_OperationalExchange», the supplier metaproperty must be stereotyped by (UAF::Operational::Connectivity) «OperationalExchange» = a specialization of «FACE_PlatformView», or «FACE_PlatformQuery», the supplier metaproperty must be stereotyped by (UAF::Resources::Information) «DataElement» = «FACE_ResourceExchange», the supplier metaproperty must be stereotyped by (UAF::Resources::Connectivity) «ResourceExchange» = «FACE_TSNodeConnection», the supplier metaproperty must be stereotyped by (UAF::Resources::Connectivity) «ResourceConnector»

FACE_IOEndpoint

Package: UAF_FACE_Extended_Stereotypes
isAbstract: No
Generalization: FACE_AbstractAssociation
Extension: Association

Description

The FACE standard states that Platform-Specific Services Segment (PSSS) Components may exchange information with the Input-Output Segment (IOS) Components, but the FACE 3.0 metamodel does not include a mechanism to express the connection. This association provides additional connections through FACE_UnitOfConformanceEndpoint elements through which PSSS FACE_UnitOfPortability elements may exchange information with IOS FACE_UnitOfConformance components elements.

In addition to aggregation and multiplicity specifications on memberEnd[1], this association differs from the default FACE_AbstractAssociation in that it is bi-directionally navigable.

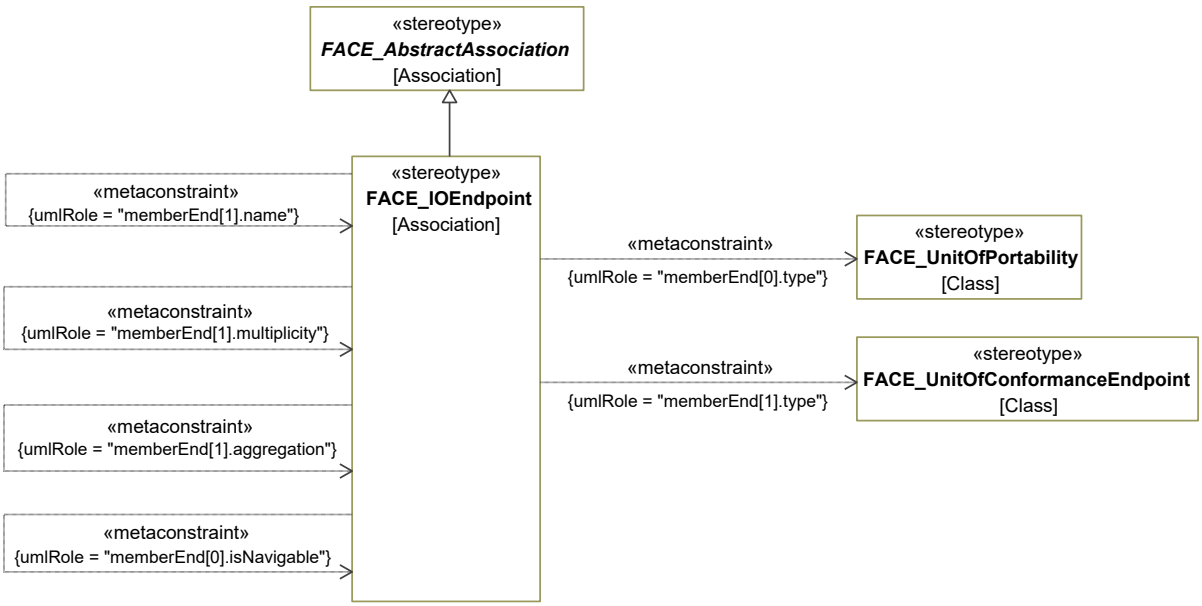

Figure 7-182: FACE_IOEndpoint

Constraints

[1] FACE_IOEndpoint.memberEnd[0].isNavigable	true
[2] FACE_IOEndpoint.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_UnitOfPortability» and memberEnd[0].componentType must be PlatformSpecificComponent.
[3] FACE_IOEndpoint.memberEnd[1].aggregation	composite
[4] FACE_IOEndpoint.memberEnd[1].multiplicity	[0..*]

[5] FACE_IOEndpoint.memberEnd[1].name	ioEndpoint
[6] FACE_IOEndpoint.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_SystemComponentEndpoint» and memberEnd[1].endPointType must be IOSEndpoint.

FACE_OperationalExchange

Package: UAF_FACE_Extended_Stereotypes
isAbstract: No
Extension: InformationFlow

Description

A type of OperationalExchange that asserts information exchange between two FACE_AbstractConnections. This has no corresponding metatype in the FACE Technical Standard because the FACE standard represents components without system context. This exchange enables expression of information exchanges between FACE elements at the system-of-systems level.

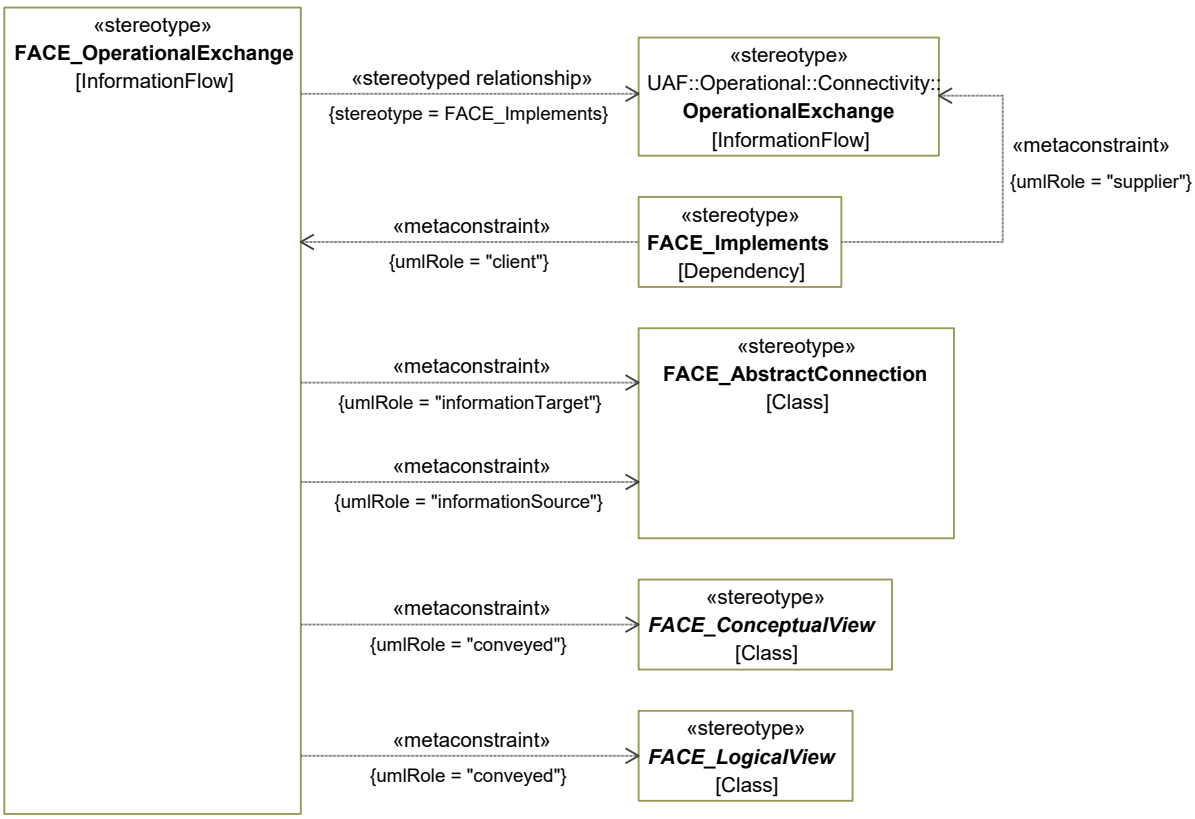

Figure 7-183: FACE_OperationalExchange

Constraints

[1] FACE_OperationalExchange.conveyed	Value for the conveyed metaproperty must be stereotyped by either the specialization of «FACE_ConceptualView» or the specialization of «FACE_LogicalView».

[2] FACE_OperationalExchange.exchangeKind	Value for the exchangeKind attribute defaults to "InformationExchange".
[3] FACE_OperationalExchange.informationSource	Value for the informationSource metaproperty must be stereotyped by «FACE_AbstractConnection».
[4] FACE_OperationalExchange.informationTarget	Value for the informationTarget metaproperty must be stereotyped by «FACE_AbstractConnection».

FACE_ResourceExchange

Package: UAF_FACE_Extended_Stereotypes
isAbstract: No
Extension: InformationFlow

Description

A type of ResourceExchange that asserts information exchange and among FACE_UnitOfPortability (via subclass of Connection) and FACE_UnitOfConformance Transport Services Segment (TSS) elements (via UnitOfConformanceEndpoint). This has no corresponding metatype in the FACE Technical Standard because the FACE standard represents components without system context. This exchange enables expression of information exchanges between FACE elements at the system-of-systems level.

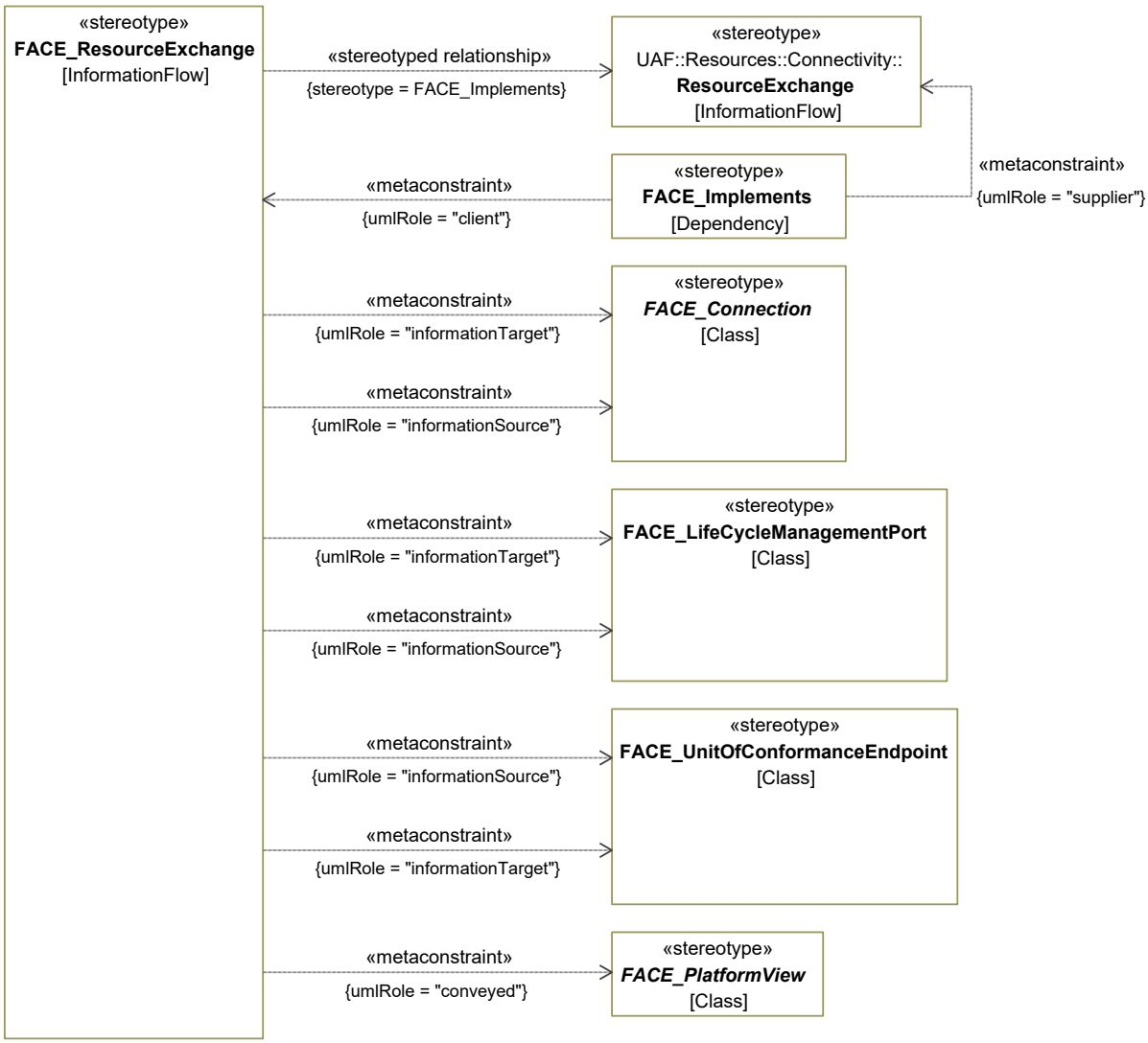

Figure 7-184: FACE_ResourceExchange

Constraints

[1] FACE_ResourceExchange.conveyed	Value for the conveyed metaproperty must be stereotyped by the specialization of «FACE_PlatformView».
[2] FACE_ResourceExchange.exchangeKind	Value for the exchangeKind attribute defaults to "FACEResourceCommunication".
[3] FACE_ResourceExchange.informationSource	Value for the informationSource metaproperty must be stereotyped by «FACE_LifeCycleManagementPort», a specialization of «FACE_Connection», or a «FACE_UnitOfConformanceEndpoint» that has endPointType = TSSEndpoint.
[4] FACE_ResourceExchange.informationTarget	Value for the informationSource metaproperty must be stereotyped by «FACE_LifeCycleManagementPort», a specialization of «FACE_Connection», or a «FACE_UnitOfConformanceEndpoint» that has endPointType = TSSEndpoint.

FACE_UnitOfConformance

Package: UAF_FACE_Extended_Stereotypes
isAbstract: No
Generalization: FACE_UoCElement
Extension: Class

Description

The FACE Technical Standard, Edition 3.0 discusses segments and component Units of Conformance (UoCs) for every segment in the FACE Data Architecture, but the FACE 3.0 metamodel includes only Portable Component Segment (PCS) and Platform-Specific Services Segment (PSSS) components. This stereotype represents FACE Components (UoCs) that are that are pertinent to a system-of-systems architecture and are allocated to segments of the FACE standard that are not represented in the FACE 3.0 metamodel.

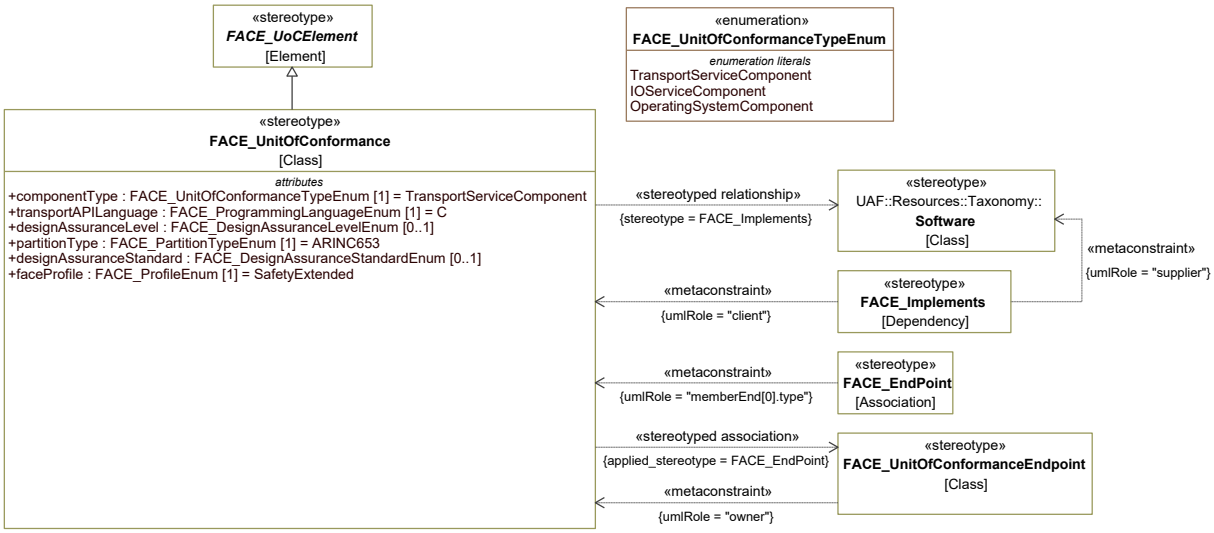

Figure 7-185: FACE_UnitOfConformance

Attributes

componentType : FACE_UnitOfConformanceTypeEnum [1]	The component type that corresponds to a segment in the FACE segment architecture. Indicates the segment into which the described Component is intended to be placed. For more details, see the enumerated type descriptions for UnitOfConformanceTypeEnum.
designAssuranceLevel : FACE_DesignAssuranceLevelEnum [0..1]	The design assurance level attributed to safety/security sensitive components. Indicates the impact of a failure condition of the described component.
designAssuranceStandard : FACE_DesignAssuranceStandardEnum [0..1]	The design assurance standard that applies to a safety/security sensitive system and that by which the design and testing of the system is judged to be safety or security certified.

faceProfile : FACE_ProfileEnum [1]	The criticality designation used by FACE to tailor the operating system to be deployed for a set of components. For more information about the details of each potential designation, please refer to the FACE Technical Standard, Edition 3.0.
partitionType : FACE_PartitionTypeEnum [1]	The operating system type for which the described component was developed.
transportAPILanguage : FACE_ProgrammingLanguageEnum [1]	The programming language to be used for the component's communications.

FACE_UnitOfConformanceEndpoint

Package: UAF_FACE_Extended_Stereotypes
isAbstract: No
Extension: Class

Description

The FACE Technical Standard, Edition 3.0 discusses segments and component Units of Conformance (UoCs) but the FACE 3.0 metamodel does not include components for every segment. This stereotype represents an aspect of component in a segment of the FACE standard that is pertinent to a system-of-systems architecture but is not represented in the FACE 3.0 metamodel.

A FACE_UnitOfConformanceEndpoint is a communication endpoint on a FACE component that is part of the Transport Services, IOServices, or Operating Services segments in FACE. These endpoints are the conduits through which information flows between FACE components in designated segments. The communication paths for FACE components are strictly governed by the FACE standard and are reflected in related stereotypes in this standard.

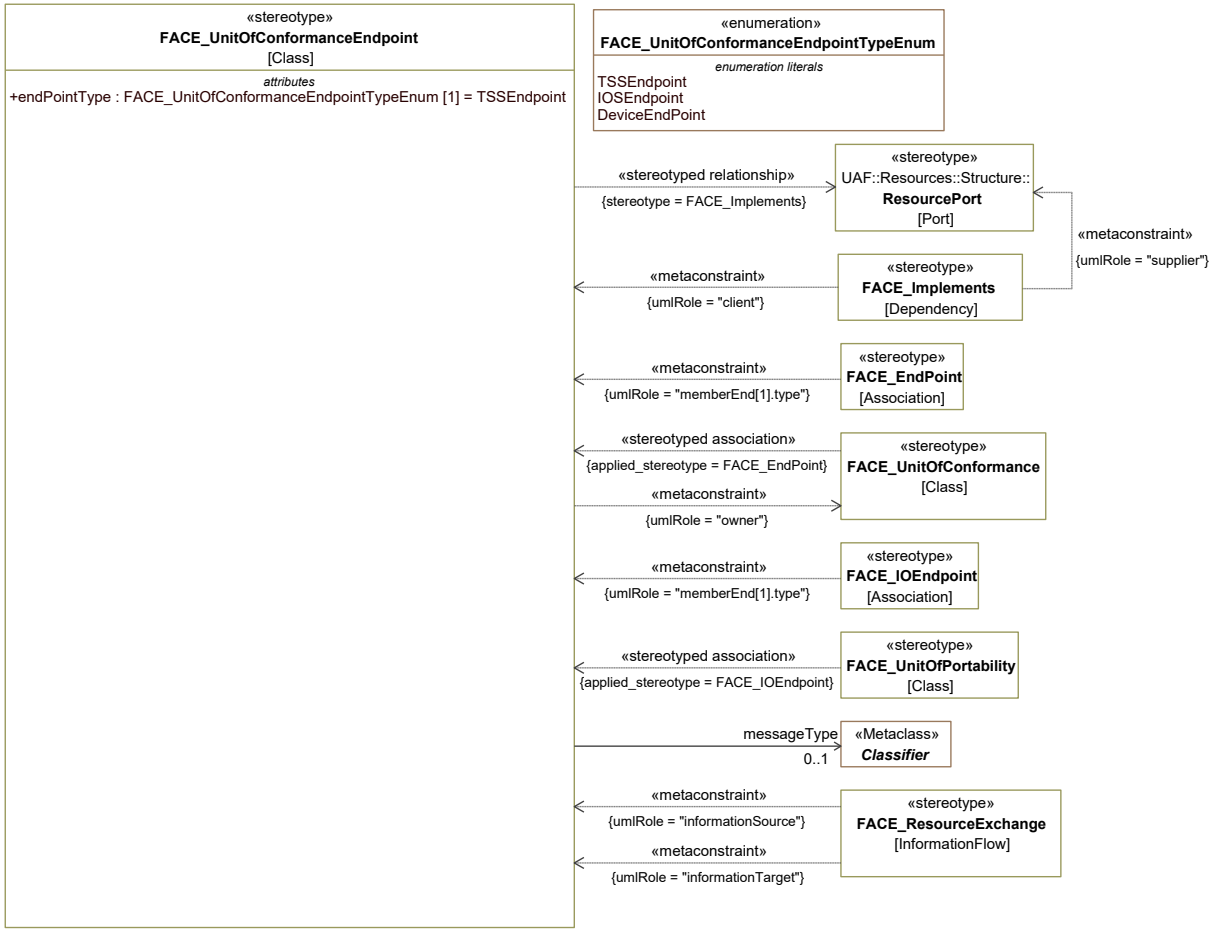

Figure 7-186: FACE_UnitOfConformanceEndpoint

Attributes

endPointType :	FACE_UnitOfConformanceEndpointTypeEnum [1]	The component type that corresponds to a the segment in the FACE architecture with which this endpoint is intended to connect. For more details, see the enumerated type descriptions for UnitOfConformanceEndpointTypeEnum.

Associations

messageType :	The classifier that describes the information/resource being exchanged through the endpoint. Characterized as Classifier because, depending on the endPointType, the exchange could be characterized in a variety of ways. Multiplicity of [0..1] because the exchange might not be characterized at this time.

Constraints

[1] FACE_UnitOfConformanceEndpoint.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_UnitOfConformance»

FACE_UnitOfConformanceEndpointTypeEnum

Package: UAF_FACE_Extended_Stereotypes
isAbstract: No

Description

This Enumeration provides types for the endpoints/connections owned by FACE components that are described in the FACE Technical Standard, Edition 3.0 but are not part of the FACE 3.0 metamodel. Each FACE component has 1 or more connections to other FACE components. The intended FACE segment for that communication is indicated by this enumerated type. Its enumeration literals are:

- TSSEndpoint - Indicates that the endpoint represents FACE Transport Services Segment (TSS) communications.
- IOSEndpoint - Indicates that the endpoint represents a communications conduit between a FACE Input/Output Services Segment (IOSS) element and a FACE Platform-Specific Segment (PSSS) element.
- DeviceEndPoint - Indicates a communications conduit between an Input/Output Services Segment (IOSS) element and a device or device driver. The target of communications from a Device endpoint may not be FACE component.

FACE_UnitOfConformanceTypeEnum

Package: UAF_FACE_Extended_Stereotypes
isAbstract: No

Description

The FACE Technical Standard, Edition 3.0 discusses segments and component Units of Conformance (UoCs) but the FACE 3.0 metamodel does not include components for every segment. This stereotype represents an aspect of a component in a segment of the FACE standard that is pertinent to a system-of-systems architecture but is not represented in the FACE 3.0 metamodel.

This enumeration represents the FACE component types that are part of the FACE 3.0 Architecture but are not represented in the FACE 3.0 metamodel.
Its enumeration literals are:

- TransportServiceComponent - Indicates that a component is a FACE Transport Services Segment (TSS) Component. TSS components provide communication between and among FACE Portable Components Segment (PCS) and Platform-Specific Services Segment (PSSS) components.
- IOServiceComponent - Indicates that a component is a FACE Input/Output Services Segment (IOSS) Component. IOSS components provide the interface between vendor-supplied device drivers (hosted in the Operating System Segment/OSS) and the Platform-Specific Services Segment (PSSS) components.
- OperatingSystemComponent - Indicates that a component is a FACE Operating System Segment (OSS) Component. OSS components include operating system services, device drivers, and other vendor-supplied software. An OSS component provides and controls access to the computing platform itself.

FACE_UoCElement

Package: UAF_FACE_Extended_Stereotypes
isAbstract: Yes
Extension: Element

Description

A FACE_UoCElement is the root type for defining the non-metamodel system elements of the ArchitectureModel.

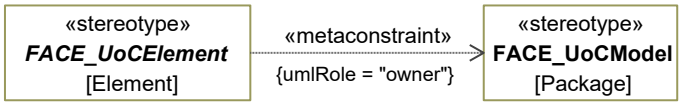

Figure 7-187: abstract FACE_UoCElement

Constraints

[1] FACE_UoCElement.owner	Elements that are stereotyped by specializations of this abstract stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_UoCModel»

FACE_UoCModel

Package: UAF_FACE_Extended_Stereotypes
isAbstract: No
Extension: Package

Description

This package holds descriptions of FACE components that are called for in the FACE Technical Standard, Edition 3.0 but that are not represented in the FACE 3.0 metamodel. These descriptions are separated from the rest of the FACE model elements to differentiate them from metamodel-represented elements.

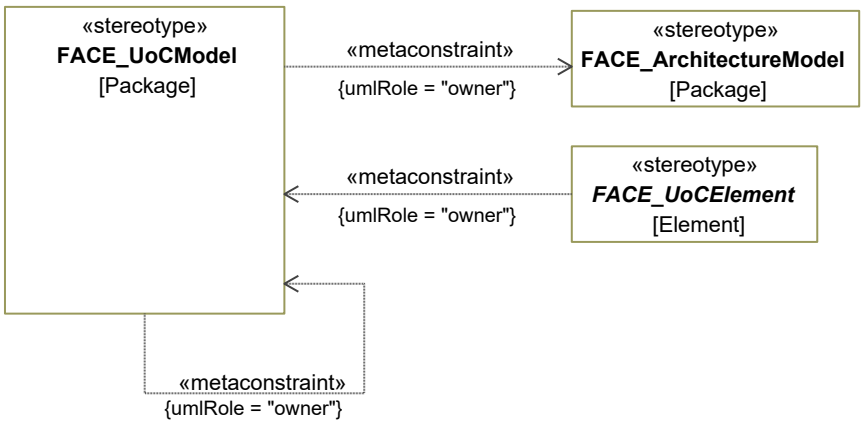

Figure 7-188: FACE_UoCModel

Constraints

[1] FACE_UoCModel.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_ArchitectureModel» «FACE_UoCModel»

7.2 View Customizations

This section addresses the requirements from the RFP that call for tables that aggregate FACE Constructs. The tables called for include:

- All FACE Components (Units of Conformance (UoCs)/Units of Portability (UoPs) elements)
- All FACE Components (UoC/UoP elements) that reside in a particular FACE Segment (PCS, PSSS, IOSS, …)
- All usages of particular FACE Interfaces or FACE Data Exchanges

In addition, the RFP calls for specific information to be included in the tables. This is detailed below:

- Safety/Security Stance (DAL and/or FACE Profile) for all FACE UoC/UoP
- FOR ALL TABLES INCLUDING UoCs/UoPs in the PSSS layer, include target layer for exchange
- FOR ALL TABLES INCLUDING MULTIPLE FACE LAYERS, include source layer of data exchange

This specification further identifies the properties of the FACE elements that it expects to see detailed in the provided tables. While this information is included in the individual view specifications, it is summarized below:

- Tables specifying only UnitOfPortability elements (with no data exchange information): UnitOfPortability Name, Layer = FACE Segment (PCS/PSSS/TSS/IOSS/OSS) , TransportAPILanguage, FACEProfile, DesignAssuranceStandard, DAL Level, PartitionType (POSIX/ARINC)
- Tables specifying message flows between FACE UnitsOfPortability or AbstractUoPs: Element Name, Connection Name (if any), MessageType, and MessageDirection (Inbound/Outbound)

Because the FACE Profile for UAF specifies FACE implementation of portions of a UAF architecture but is not comprised of UAF elements, the views specified in this section are not expressed as UAF views.

7.2.1 View Specifications::FACE Data Architecture

7.2.1.1 View Specifications::All FACE Components View

Stakeholders: Systems Engineers, Software Engineers
Concerns: Identification of FACE Components
Definition: Allows identification of all FACE Components in an architecture and their characteristics
Recommended Implementation: Tabular Format
Characteristics to Display: For all «UAF::Resources::Taxonomy::Software» stereotyped elements in user-selected scope, if «Software» is the supplier for a «FACE_Implements» relationship and the client is stereotyped by «FACE_UnitOfPortability» or «FACE_UnitOfConformance», display the following attributes of the client «FACE_UnitOfPortability» or «FACE_UnitOfConformance»::

- <element>.name
- <element>.componentType
- <element>.transportAPILanguage
- <element>.faceProfile
- <element>.designAssuranceStandard
- <element>.designAssuranceLevel
- <element>.PartitionType

Stereotypes of elements and relationships to use when constructing All FACE Components View

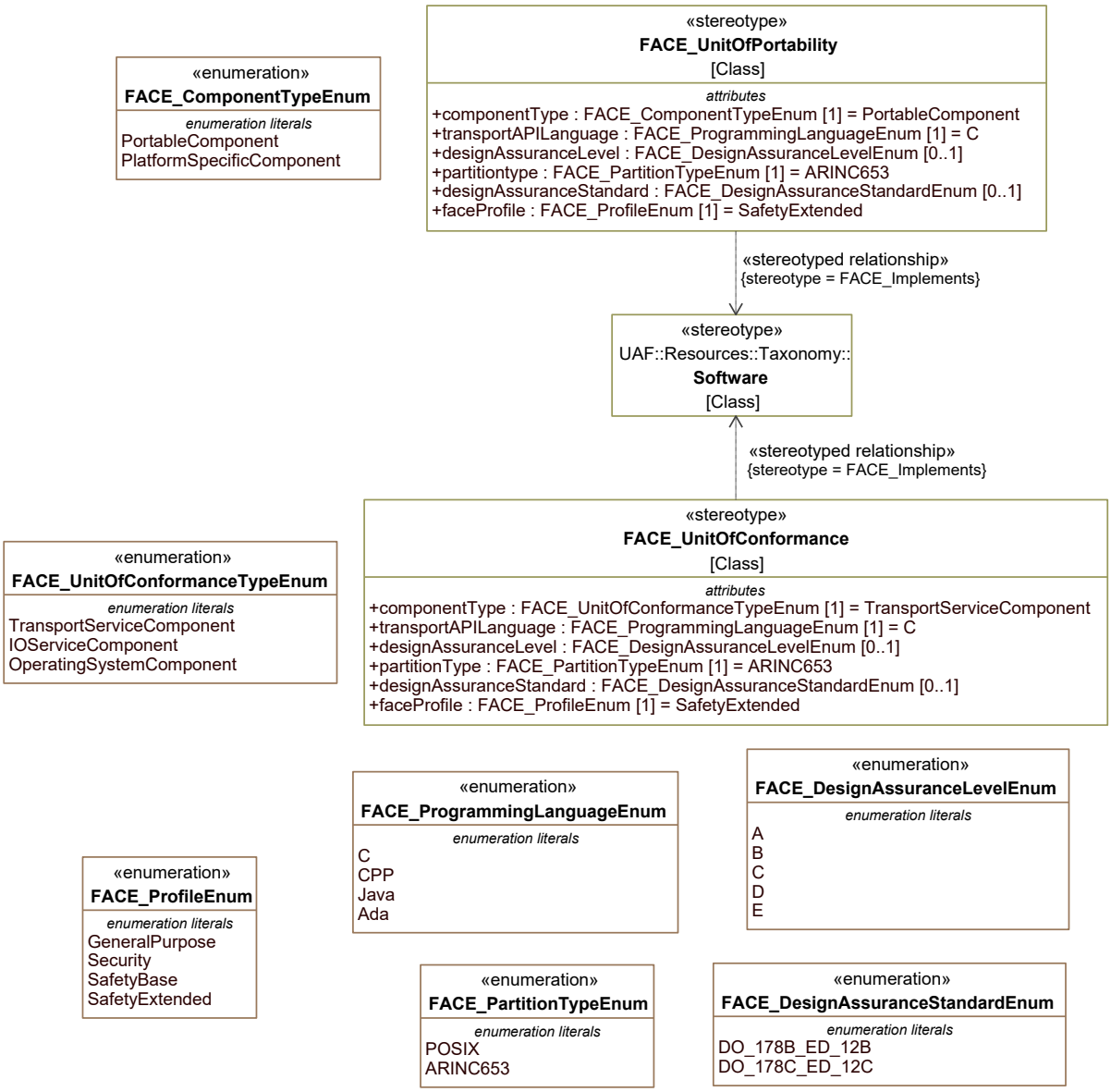

Figure A-1: All FACE Components View

Elements

- FACE_ComponentTypeEnum
- FACE_DesignAssuranceLevelEnum
- FACE_DesignAssuranceStandardEnum
- FACE_PartitionTypeEnum
- FACE_ProfileEnum
- FACE_ProgrammingLanguageEnum
- FACE_UnitOfConformance
- FACE_UnitOfConformanceTypeEnum
- FACE_UnitOfPortability
- Software

7.2.1.2 View Specifications::FACE Components Per Segment View

Stakeholders: Systems Engineers, Software Engineers
Concerns: Categorization of FACE Components
Definition: Allows identification and characterization of all FACE Components in a specific FACE Segment (of a specific ComponentType) of an architecture
Recommended Implementation: Tabular Format
Characteristics to Display: For all «UAF::Resources::Taxonomy::Software» stereotyped elements in user-selected scope, if «Software» is the supplier for a «FACE_Implements» relationship and the «FACE_Implements».client is stereotyped by «FACE_UnitOfPortability» or «FACE_UnitOfConformance» AND the client <element>.componentType matches the user-specified ComponentTypeEnum or UnitOfConformanceTypeEnum value, display for the client element:

- <element>.name
- <element>.componentType
- <element>.transportAPILanguage
- <element>.faceProfile
- <element>.designAssuranceStandard
- <element>.designAssuranceLevel
- <element>.PartitionType

Stereotypes of elements and relationships to use when constructing FACE Components Per Segment View

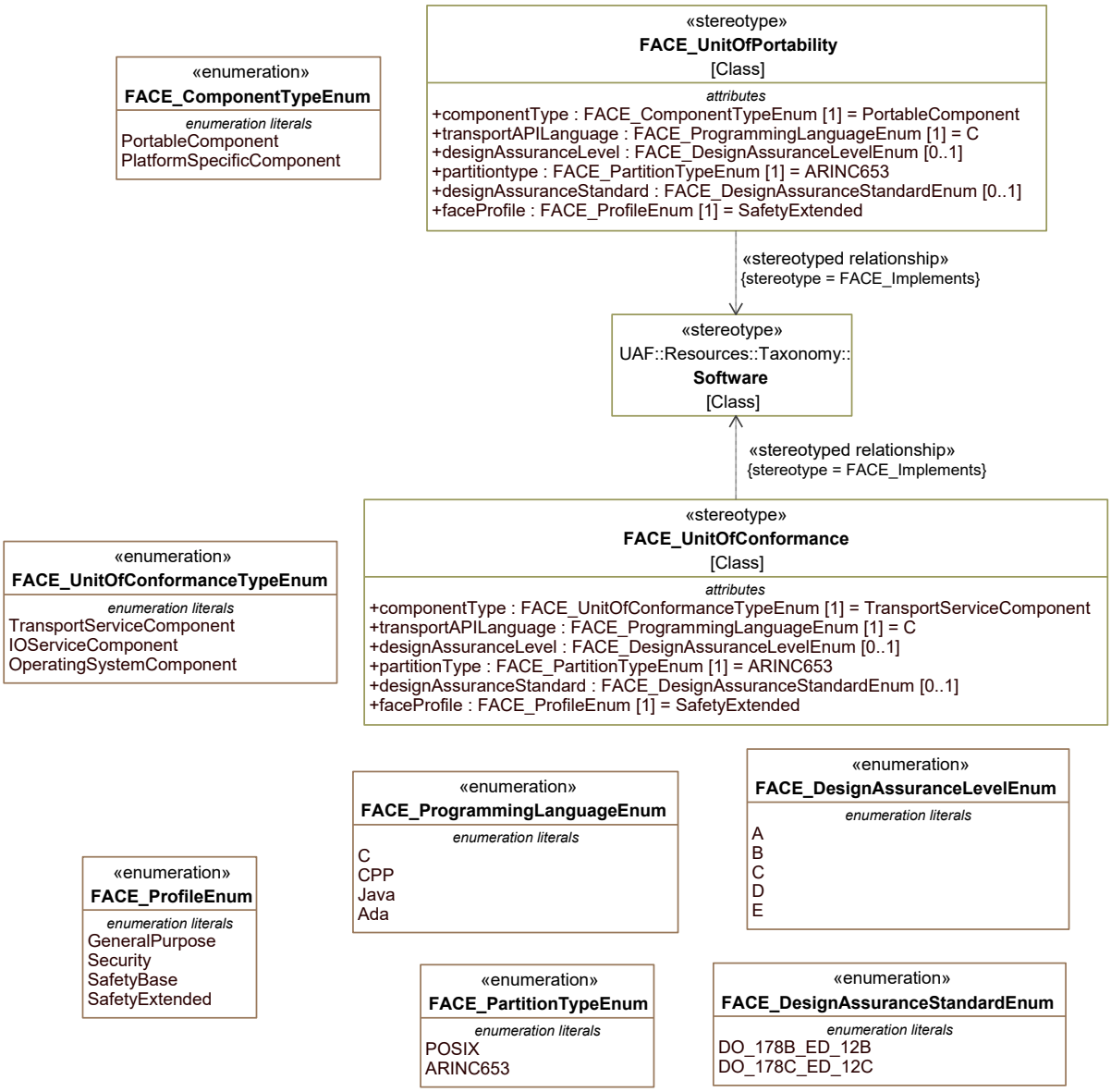

Figure A-2: FACE Components Per Segment View

Elements

- FACE_ComponentTypeEnum
- FACE_DesignAssuranceLevelEnum
- FACE_DesignAssuranceStandardEnum
- FACE_PartitionTypeEnum
- FACE_ProfileEnum
- FACE_ProgrammingLanguageEnum
- FACE_UnitOfConformance
- FACE_UnitOfConformanceTypeEnum
- FACE_UnitOfPortability
- Software

7.2.1.3 View Specifications::FACE Logical Interfaces View

Stakeholders: Systems Architects, Systems Engineers
Concerns: Identifies logical interfaces between FACE Abstract UoP components in the architecture
Definition: Shows the connections between abstract FACE Components in the architecture
Recommended Implementation: Tabular Format
Desired information is found by navigating from OperationalExchanges in the selected UAF scope and navigation to «FACE_OperationalExchange» elements via «FACE_Implements» relationships:
For each OperationalExchange in the selected UAF scope, for each «FACE_Implements» relationship in which the ResourceExchange is the supplier and a «FACE_OperationalExchange» element is the client, desired information for the «FACE_OperationalExchange» client of the «FACE_Implements» relationship:

- (Source UoP Name) <FACE_OperationalExchange>.informationSource->(AbstractConnection).EndPoint->memberEnd[0].type->(AbstractUop).name
- (Target UoP Name) <FACE_OperationalExchange>.informationTarget->(AbstractConnection).EndPoint->memberEnd[0].type->(AbstractUop).name
- (MessageType) <FACE_OperationalExchange>.conveyed.type
- Message direction is implied by the Operational Exchange direction

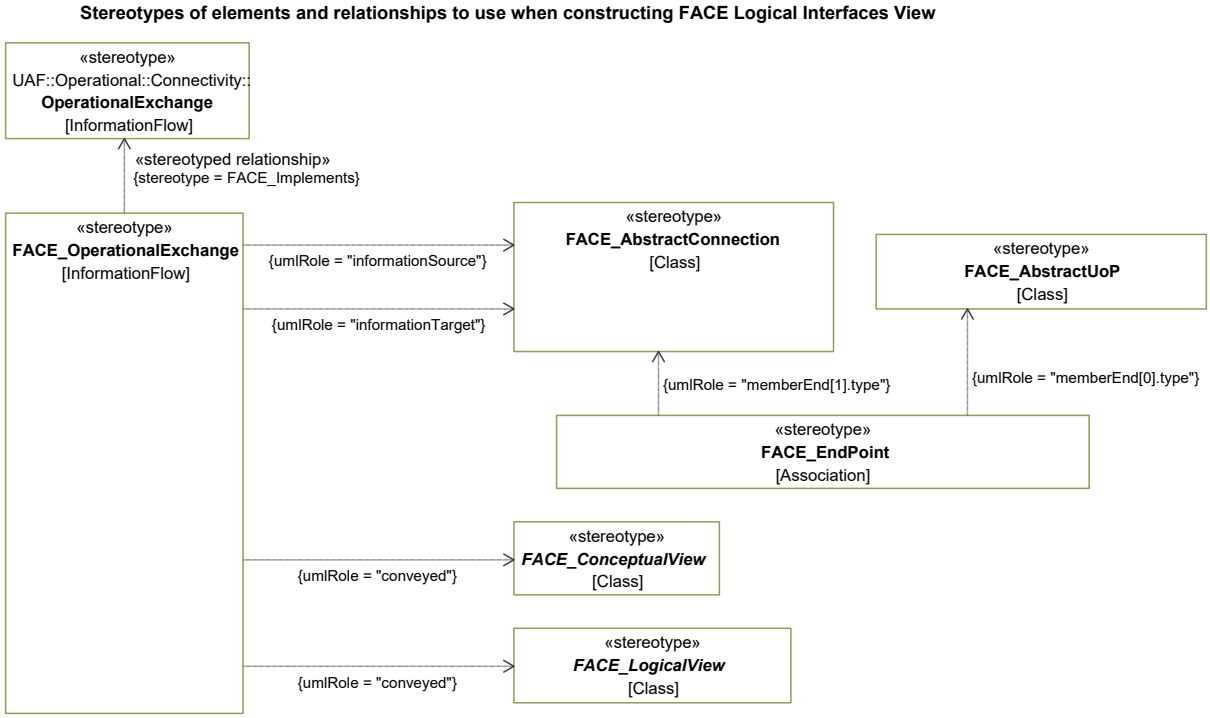

Figure A-3: FACE Logical Interfaces View

Elements

- FACE_AbstractConnection
- FACE_AbstractUoP
- FACE_ConceptualView
- FACE_EndPoint
- FACE_LogicalView
- FACE_OperationalExchange
- OperationalExchange

7.2.1.4 View Specifications::FACE Physical Interfaces View

Stakeholders: Systems Architects, Systems Engineers
Concerns: Identifies resource-level interfaces between FACE components in the architecture
Definition: Shows the connections between FACE Components in the architecture and identifies the layered segments in which the source and targets of the interactions reside.
Recommended Implementation: Tabular Format
Desired information is based on ResourceExchanges in the selected UAF scope and navigation via «FACE_Implements» relationship:
For each ResourceExchange in the selected UAF scope, for each «FACE_Implements» relationship in which the ResourceExchange is the supplier and a «FACE_ResourceExchange» element is the client, desired information for the «FACE_ResourceExchange» client of the «FACE_Implements» relationship:

- (Source Component Name) <FACE_ResourceExchange>.informationSource->(<connection element>).EndPoint->memberEnd[0].type->(UnitOfPortability/UnitOfConformance).name
- (Source Component Layer) <FACE_ResourceExchange>.informationSource->(<connection element>).EndPoint->memberEnd[0].type->(UnitOfPortability/UnitOfConformance).componentType
- (Target Component Name) <FACE_ResourceExchange>.informationSource->(<connection element>).EndPoint->memberEnd[0].type->(UnitOfPortability/UnitOfConformance).name
- (Target Component Layer) <FACE_ResourceExchange>.informationSource->(<connection element>).EndPoint->memberEnd[0].type->(UnitOfPortability/UnitOfConformance).componentType
- (MessageType) <FACE_ResourceExchange>.conveyed->name
- Message direction is implied by the FACE_ResourceExchange direction

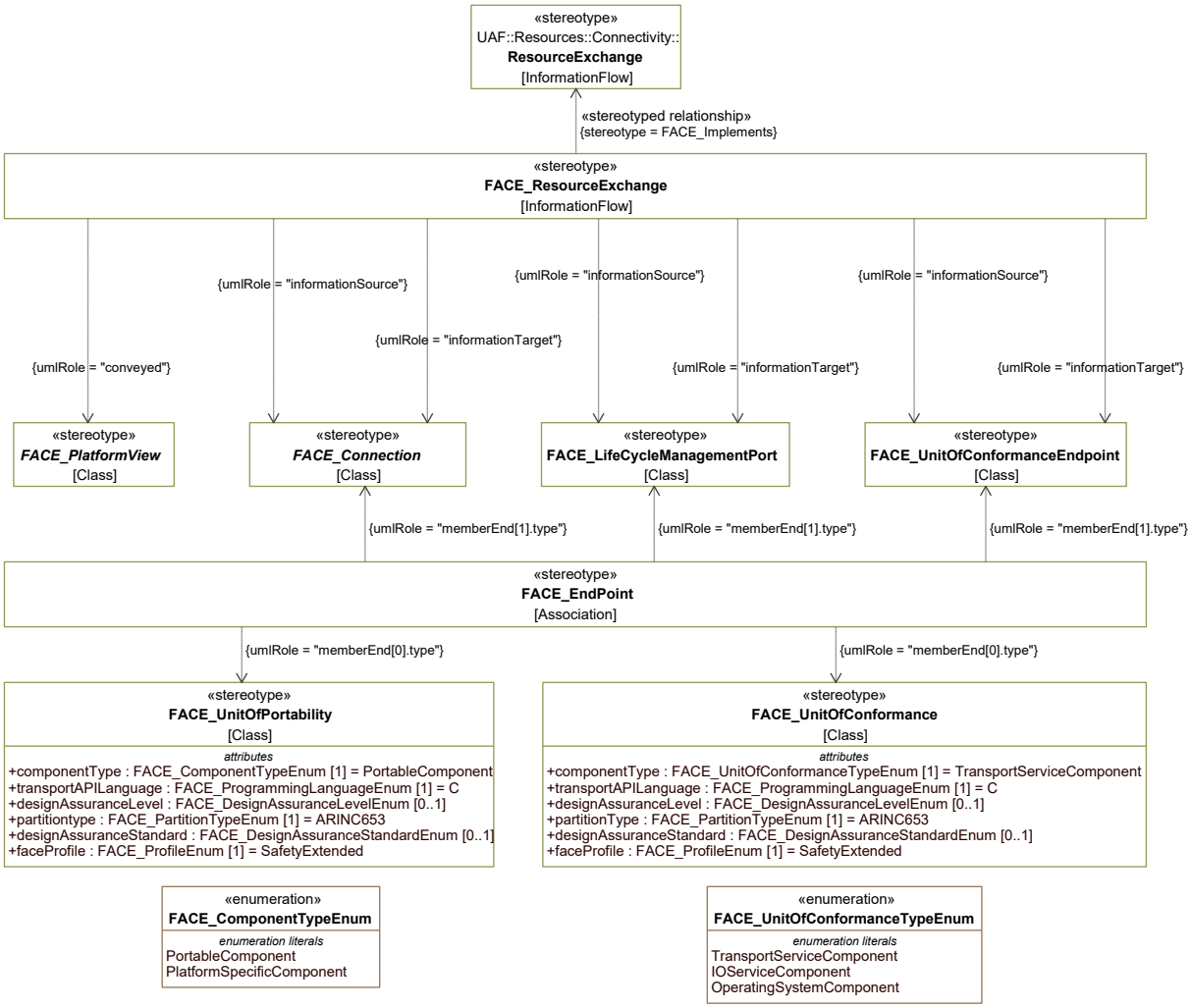

Figure A-4: FACE Physical Interfaces View

Elements

- FACE_ComponentTypeEnum
- FACE_Connection
- FACE_EndPoint
- FACE_LifeCycleManagementPort
- FACE_PlatformView
- FACE_ResourceExchange
- FACE_UnitOfConformance
- FACE_UnitOfConformanceEndpoint
- FACE_UnitOfConformanceTypeEnum
- FACE_UnitOfPortability
- ResourceExchange

8. Design Considerations (Non-Normative)

This section addresses the items in section 6.7 (Issues to be discussed) of the FACE™ Profile for UAF Request For Proposal (RFP), OMG document c4i-18-09-03.

8.1 Relationships to UAF profile: How the FACE Profile for UAF Enhances Related to Architectures

This section responds to the RFP section 6.7.1 Relationships to UAF profile, which requests that the specification discuss how inclusion of FACE Profile elements in UAF models enhance general architecture, DoDAF, MODAF, and NATO Architecture models.

The FACE technical standard defines a layered architecture that is separated into several segments: PCS - Portable Component Segment (presentation-layer applications), TSS - Transport Services Segment (middleware), PSSS - Platform-Specific Software Segment (platform-specific services), IOSS - Input/Output Services Segment (hardware device drivers), and OSS - Operating Systems Segment (foundational system services and vendor-supplied software). This is a level of granularity that is not specified in the UAF metamodel and which can be of value when specifying requirements for individual components within a system-of-systems. By linking the FACE profile's differentiations between layers and the information-transform representations of the FACE Integration Model, the FACE Profile for UAF enhances the representation of layered architecture elements and the flow of information throughout a system of systems.

8.2 Support for Cyber Security within the System: Security Analysis enhancements from FACE Profile for UAF

This section responds to the RFP section 6.7.3 Support for Cyber Security within the System, which requests that the specification discuss how FACE Profile enhances system-of-systems security analysis when combined with UML-based security and risk/threat analysis technologies.

The FACE standard addresses the specification of avionics systems components with respect to safety, security, partitioning, integration, and semantic documentation of information exchanges. The FACE profile brings this enhanced specification information to UAF architectures. Further, by enabling expression of FACE components using OMG technologies, FACE components can be further elaborated within a UAF architecture through the application of the MARTE profile (Modeling and Analysis of Real-Time and Embedded systems). The FACE profile for UAF along with the MARTE profile will enable architects to associate information within the UAF database with implementation mechanisms that express the architecture in terms of layers, connectivity, partitioning strategy and hardware/software typing. The MARTE specification General Component Model (GCM) includes detailed information of components. The MARTE profile complements the FACE profile by providing detailed specification of any Design Assurance Standard and Design Assurance Level (DAL) associated with a FACE-profile component, as well as introducing other analysis-related attributes to the architecture.

8.3 Combining FACE Profile for UAF with MARTE markings to feed AADL analysis

This section responds to the RFP section 6.7.2 Relationships to MARTE profile, which requests that the specification discuss how FACE Profile information could be combined with MARTE markings to feed AADL safety analysis tools.

The mapping of FACE elements into a UAF architecture enables finer-grained description of real-time avionics systems components with respect to safety, security, partitioning, integration, and semantic documentation of information exchanges. Within the context of a combined FACE and UAF model, the combination of the FACE profile with the MARTE will enable architects to associate information within the UAF database with implementation mechanisms that express the architecture in terms of layers, connectivity, partitioning strategy and hardware/software typing. There are mechanisms by which information can then be transferred from a UAF-FACE combined model that uses MARTE to an Architecture Analysis & Design Language (AADL) modeling tool to support safety analysis using AADL tool capabilities. MARTE provides many of the tagging keys which are used by AADL to support the proper transfer of information. The MARTE profile combined with the structuring information provided by a FACE profile gives identified structure and meaning needed by an AADL safety analysis tool to generate such information as (Avionics Application Standard Software Interface) ARINC 653 partition parameters needed to meet safety requirements needed for proper timing design.

8.4 Non-Profile Tool implementation aspects of the FACE Technical Standard

This section discusses non-Profile tool implementation aspects of the specification, in response to the items in section 6.7.4 (Tool implementation of aspects the FACE Technical Specification that are outside the bounds of a profile) from FACE™ Profile for UAF Request For Proposal, document c4i-18-09-03.

8.4.1 Suggested Approaches for Enforcement of OCL Constraints from FACE Technical Standard

The application of OCL constraints from the FACE Technical Standard, Edition 3.0 is not a requirement of this specification's profile itself, nor is it a requirement for Level A conformance to this standard. Application of FACE OCL constraints is required for Conformance levels AA and AAA of this specification. This section describes possible approaches by which implementations of this standard at higher levels of conformance might implement and possibly enforce these constraints.

8.4.1.1 Level AA Conformance application of FACE OCL Constraints

Level AA Conformance provides the minimum support needed by the users of FACE data architecture models in order to use the authored information in a FACE integration effort. There is no requirement to implement the FACE OCL Constraints directly in the modeling tool at Level AA Conformance. Conformance Level AA enables the use of FACE Consortium conformance checking tools that ensure model OCL correctness. This is enabled by the export/import of the FACE model elements to/from the FACE XML format as specified in the normative MOF specification derived from the FACE Technical Standard, Edition 3.0.

The recommended approach for application of FACE OCL Constraints under Level AA Conformance is to export the model to the FACE XML-formatted (.face) file format and direct the user to the FACE Conformance Test Suite (CTS) for OCL constraint checking. The notional steps in this process are listed below:

1) Ensure that all FACE Elements are contained in the FACE Architecture Package
2) Provide mechanism to perform export of FACE Architecture to FACE XML (.face) format using plug-ins
3) Direct the user to independently use the FACE Conformance Test Suite to check model adherence to OCL constraints
4) User modifies model in tool to address issues
5) User would repeat export-test-modify as needed to address all FACE conformance model issues

8.4.1.2 Level AAA Conformance application of FACE OCL Constraints

Level AAA Conformance supports the rapid development of FACE architecture, data models, and software development through application of the FACE/OCL Constraints during the architecture modeling process. Level AAA Conformance of this specification includes implementation of FACE OCL Constraints directly in the modeling tool. There are a few different approaches that an implementer of the standard at Level AAA Conformance might wish to consider in the implementation of these constraints. The potential approaches listed below are suggestions for application of the constraints and are not meant to exclude alternate approaches. Possible approaches include:

1) Apply the OCL Constraints from the FACE Technical Standard, Edition 3.0 to check the entire set of FACE Model Elements in the tool. Add a plug-in to perform all FACE OCL Constraint checks upon request and provide the constraint check results to the user. The user addresses issues in the model and repeats the constraint test as needed. The benefit of this approach is that it minimizes rework of FACE OCL Constraints that apply to the entire FACE model, minimizes lag due to long-running constraint checks, and provides user control over when constraint checking will occur.
2) Apply the OCL Constraints from the FACE Technical Standard, Edition 3.0 to each FACE Model Element individually in the tool. Perform OCL Constraint checks for each element upon modification. The user addresses the constraint violations as they are identified.

The benefit of this approach is that it minimizes the time between authoring a model element and notification of constraint violation.

3) Apply the OCL Constraints from the FACE Technical Standard, Edition 3.0 to FACE Model Elements in a hybrid fashion. This is a combination of approaches 2 and 3. Apply constraints that are highly localized (quick running) on an element-by-element basis and a plug-in to perform all FACE OCL Constraint checks upon request and provide the constraint check results to the user.
   This approach combines the benefits of both approaches 2 and 3.

8.4.2 Recommended mechanism to generate content into FACE Profile tabular views

Users of the FACE Profile might wish to see tables of elements that support specific FACE Profile enumerated types (General, Safety-Base, Safety-Extended, Security). Most modeling tools provide a mechanism to generate tabular views of selected information from the model and to display it with or without filters. The steps below outline one possible mechanism for implementers of the profile to provide tables of FACE-stereotyped components to users:

1) Use the Tool-Native Table and plug-in extension capabilities.
2) Provide FACE-profile-specific table as selection option in “New Diagram” menu(s).
3) For each FACE UoP or Abstract UoP in the (singleton) FACE Architecture package, plug-in identifies the FACE security stance and places the name and security stance in a table as appropriate to the intended table contents. Tables may be created containing all FACE modules or may be specific to a single security stance selected by the user. Tool-native filtering and sorting may be applied by the user after table creation, as can extension of module properties displayed in the table.

8.4.3 Inclusion of the FACE vertical architecture image in tool implementations

For reference purposes, FACE Profile users might need access to a graphical view of the general FACE vertical architecture. The FACE Technical Standard, Edition 3.0 contains an image of the FACE Vertical Architecture, labeled “FACE Architectural Segments” in the standard. Figure 8-1 shows that image, and informational files included with this standard provide additional details. Tools that implement the FACE profile could include a copy of the image as/in a diagram that users request via plug-in support menus.

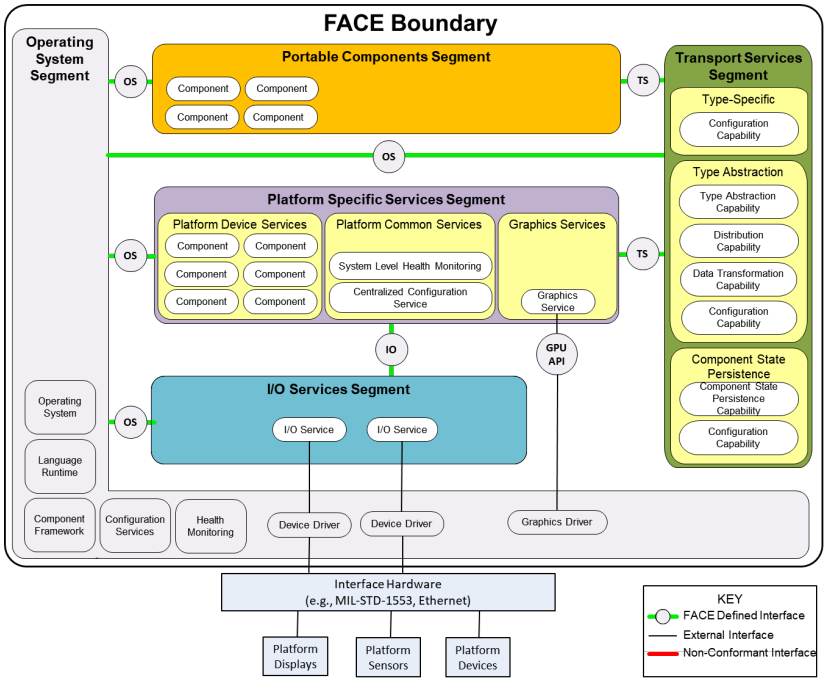

Figure 8-1: FACE Technical Interchange Meeting Architectural Diagram Template Example

A FACE Profile Mapping Tables (Informational / Non-Normative)

This chapter provides information about the relationship between the FACE Consortium FACE Metamodel elements, the FACE Profile elements, and the UAF elements in tabular form. It is meant to provide this information in an easy-to-consume format for enhanced understanding of these relationships.

A.1 FACE Metamodel to FACE Profile Mapping

This section provides the mapping between the FACE 3.0 metamodel elements and the corresponding FACE Profile elements in tabular form. The order of the metamodel elements in the table corresponds to their order in in the FACE Technical Standard, Edition 3.0. The FACE elements are generally implemented using a single stereotype to represent the element itself, with additional stereotypes listed if used to represent attributes or associations from the FACE metamodel.

A.1.1 FACE Metamodel path elements

The FACE 3.0 Metamodel path elements named CharacteristicPathNode, Participant, ParticipantPathNode, and PathNode have an alternate-syntax representation called a CharacteristicProjection. This notation is described in Section 3.6.4.1.1.3 of the Technical Standard for Future Airborne Capability Environment (FACE™), Edition 2.1 and fully expresses the paths as described using the FACE 3.0 path metamodel elements. The two notations (elements and string) are interchangeable using a translation algorithm. The CharacteristicProjection syntax is used in the FACE Profile for UAF instead of the corresponding FACE Metamodel elements. XMI exchange mechanisms between models using the FACE Profile for UAF and the FACE XMI (.face) file are required to translate between the two notations.

The following table shows the FACE metamodel path elements and their corresponding CharacteristicPathNode-syntax FACE Profile for UAF elements.

Table 8-1 FACE Metamodel Path Elements mapping to FACE Profile Stereotype containing equivalent string syntax

FACE Metamodel Package	FACE Metamodel Element Names	FACE Profile Stereotype
face.datamodel.conceptual	Participant CharacteristicPathNode ParticipantPathNode PathNode	FACE_ConceptualParticipant [Class]
face.datamodel.logical	Participant CharacteristicPathNode ParticipantPathNode PathNode	FACE_LogicalParticipant [Class]
face.datamodel.platform	Participant CharacteristicPathNode ParticipantPathNode PathNode	FACE_PlatformParticipant [Class]

A.1.2 Full Mapping of FACE Metamodel to FACE Profile

The table below shows the FACE metamodel elements as listed in the FACE Technical Standard, Edition 3.0 and their mapping to stereotypes that, in part or whole, realize the metamodel element and its relationships in the FACE Profile for UAF. The order of the elements in the table corresponds to the order of the metamodel elements in the FACE Technical Standard, Edition 3.0, Appendix J.

Table 8-2 FACE Metamodel to FACE Profile element mapping

FACE Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face	ArchitectureModel	FACE_ArchitectureModel [Package]
face	Element	FACE_Element [Element] FACE_ModelElement [Element]
face	DataModel	FACE_DataModel [Package]
face.datamodel	Element	FACE_DataModelElement [Element]
face.datamodel	ConceptualDataModel	FACE_ConceptualDataModel [Package]
face.datamodel	LogicalDataModel	FACE_LogicalDataModel [Package]
face.datamodel	PlatformDataModel	FACE_PlatformDataModel [Package]
face.datamodel.conceptual	Element	FACE_ConceptualElement [Element]
face.datamodel.conceptual	ComposableElement	FACE_ConceptualComposableElement [Element]
face.datamodel.conceptual	BasisElement	FACE_BasisElement [Element]
face.datamodel.conceptual	BasisEntity	FACE_BasisEntity [Class]
face.datamodel.conceptual	Domain	FACE_Domain [Class]
face.datamodel.conceptual	Observable	FACE_Observable [Class]
face.datamodel.conceptual	Characteristic	FACE_ConceptualCharacteristic [Element]
face.datamodel.conceptual	Entity	FACE_ConceptualEntity [Class] FACE_EntityBasis [Generalization] FACE_Specialize [Generalization] FACE_ConceptualComposition [Property] FACE_SpecializationOwner [Class]
face.datamodel.conceptual	Composition	FACE_ConceptualComposition [Property] FACE_ConceptualComposableElement [Element]
face.datamodel.conceptual	Association	FACE_ConceptualAssociation [Class] FACE_AssociatedParticipant [Association] FACE_Specialize [Generalization] FACE_SpecializationOwner [Class]
face.datamodel.conceptual	Participant	FACE_ConceptualParticipant [Class]
face.datamodel.conceptual	PathNode	FACE_ConceptualParticipant [Class]
face.datamodel.conceptual	ParticipantPathNode	FACE_ConceptualParticipant [Class]
face.datamodel.conceptual	CharacteristicPathNode	FACE_ConceptualParticipant [Class]
face.datamodel.conceptual	View	FACE_ConceptualView [Class]
face.datamodel.conceptual	Query	FACE_ConceptualQuery [Class]
face.datamodel.conceptual	CompositeQuery	FACE_ConceptualCompositeQuery [Class] FACE_ConceptualQueryComposition [Property]
face.datamodel.conceptual	QueryComposition	FACE_ConceptualQueryComposition [Property] FACE_ConceptualView [Class]
face.datamodel.logical	Element	FACE_LogicalElement [Element]
face.datamodel.logical	ConvertibleElement	FACE_ConvertibleElement [Element]
face.datamodel.logical	Unit	FACE_Unit [Class]

FACE Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.datamodel.logical	Conversion	FACE_Conversion [Class]
face.datamodel.logical	AffineConversion	FACE_AffineConversion [Class]
face.datamodel.logical	ValueType	FACE_ValueTypeEnum
face.datamodel.logical	String	FACE_LogicalValueType [Class] FACE_ValueTypeEnum
face.datamodel.logical	Character	FACE_LogicalValueType [Class] FACE_ValueTypeEnum
face.datamodel.logical	Boolean	FACE_LogicalValueType [Class] FACE_ValueTypeEnum
face.datamodel.logical	Numeric	FACE_LogicalValueType [Class] FACE_ValueTypeEnum
face.datamodel.logical	Integer	FACE_LogicalValueType [Class] FACE_ValueTypeEnum
face.datamodel.logical	Natural	FACE_LogicalValueType [Class] FACE_ValueTypeEnum
face.datamodel.logical	Real	FACE_LogicalValueType [Class] FACE_ValueTypeEnum
face.datamodel.logical	NonNegativeReal	FACE_LogicalValueType [Class] FACE_ValueTypeEnum
face.datamodel.logical	Enumerated	FACE_LogicalValueType [Class] FACE_ValueTypeEnum
face.datamodel.logical	EnumerationLabel	FACE_EnumerationLabel [Property]
face.datamodel.logical	CoordinateSystem	FACE_CoordinateSystem [Class] FACE_Axis [Association]
face.datamodel.logical	CoordinateSystemAxis	FACE_CoordinateSystemAxis [Class]
face.datamodel.logical	AbstractMeasurementSystem	FACE_AbstractMeasurementSystem [Class]
face.datamodel.logical	StandardMeasurementSystem	FACE_StandardMeasurementSystem [Class]
face.datamodel.logical	Landmark	FACE_Landmark [Class]
face.datamodel.logical	MeasurementSystem	FACE_MeasurementSystem [Class] FACE_Axis [Association] FACE_DefinedReferencePoint [Association] FACE_AppliedConstraint [Association]
face.datamodel.logical	MeasurementSystemAxis	FACE_MeasurementSystemAxis [Class] FACE_AppliedValueTypeUnit [Association] FACE_AppliedConstraint [Association]
face.datamodel.logical	ReferencePoint	FACE_ReferencePoint [Class] FACE_RPPart [Association]
face.datamodel.logical	ReferencePointPart	FACE_ReferencePointPart [Class]
face.datamodel.logical	ValueTypeUnit	FACE_ValueTypeUnit [Class] FACE_AppliedConstraint [Association]
face.datamodel.logical	Constraint	FACE_Constraint [Class]
face.datamodel.logical	IntegerConstraint	FACE_IntegerConstraint [Class]
face.datamodel.logical	IntegerRangeConstraint	FACE_IntegerRangeConstraint [Class]
face.datamodel.logical	RealConstraint	FACE_RealConstraint [Class]
face.datamodel.logical	RealRangeConstraint	FACE_RealRangeConstraint [Class]
face.datamodel.logical	StringConstraint	FACE_StringConstraint [Class]

FACE Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.datamodel.logical	RegularExpressionConstraint	FACE_RegularExpressionConstraint [Class]
face.datamodel.logical	FixedLengthStringConstraint	FACE_FixedLengthStringConstraint [Class]
face.datamodel.logical	EnumerationConstraint	FACE_EnumerationConstraint [Class]
face.datamodel.logical	MeasurementConstraint	FACE_MeasurementConstraint [Class]
face.datamodel.logical	MeasurementSystemConversion	FACE_MeasurementSystemConversion [Class]
face.datamodel.logical	AbstractMeasurement	FACE_AbstractMeasurement [Element]
face.datamodel.logical	Measurement	FACE_Measurement [Class] FACE_Axis [Association] FACE_AppliedConstraint [Association] FACE_Realize [Association]
face.datamodel.logical	MeasurementAxis	FACE_MeasurementAxis [Class] FACE_AppliedValueTypeUnit [Association] FACE_AppliedConstraint [Association] FACE_Realize [Association]
face.datamodel.logical	MeasurementAttribute	FACE_MeasurementAttribute [Property]
face.datamodel.logical	MeasurementConversion	FACE_MeasurementConversion [Class]
face.datamodel.logical	ComposableElement	FACE_LogicalComposableElement [Element]
face.datamodel.logical	Characteristic	FACE_LogicalCharacteristic [Element]
face.datamodel.logical	Entity	FACE_LogicalEntity [Class] FACE_Realize [Association] FACE_Specialize [Generalization] FACE_LogicalComposition [Property] FACE_SpecializationOwner [Class]
face.datamodel.logical	Composition	FACE_LogicalComposition [Property] FACE_LogicalComposableElement [Element]
face.datamodel.logical	Association	FACE_LogicalAssociation [Class] FACE_AssociatedParticipant [Association] FACE_Realize [Association] FACE_Specialize [Generalization] FACE_SpecializationOwner [Class]
face.datamodel.logical	Participant	FACE_LogicalParticipant [Class]
face.datamodel.logical	PathNode	FACE_LogicalParticipant [Class]
face.datamodel.logical	ParticipantPathNode	FACE_LogicalParticipant [Class]
face.datamodel.logical	CharacteristicPathNode	FACE_LogicalParticipant [Class]
face.datamodel.logical	View	FACE_LogicalView [Class]
face.datamodel.logical	Query	FACE_LogicalQuery [Class] FACE_Realize [Association]
face.datamodel.logical	CompositeQuery	FACE_LogicalCompositeQuery [Class] FACE_Realize [Association] FACE_LogicalQueryComposition [Property]
face.datamodel.logical	QueryComposition	FACE_LogicalQueryComposition [Property] FACE_LogicalView [Class]
face.datamodel.platform	Element	FACE_PlatformElement [Element]
face.datamodel.platform	ComposableElement	FACE_PlatformComposableElement [Element]
face.datamodel.platform	PhysicalDataType	FACE_PhysicalDataType [Element]

FACE Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.datamodel.platform	IDLType	FACE_IDLType [Element] FACE_Realize [Association]
face.datamodel.platform	IDLPrimitive	FACE_IDLPrimitive [Class]
face.datamodel.platform	Boolean	FACE_Boolean [Class]
face.datamodel.platform	Octet	FACE_Octet [Class]
face.datamodel.platform	CharType	FACE_CharType [Class]
face.datamodel.platform	Char	FACE_Char [Class]
face.datamodel.platform	StringType	FACE_StringType [Class]
face.datamodel.platform	IDLUnboundedString	FACE_IDLUnboundedString [Class]
face.datamodel.platform	String	FACE_String [Class]
face.datamodel.platform	IDLBoundedString	FACE_IDLBoundedString [Class]
face.datamodel.platform	BoundedString	FACE_BoundedString [Class]
face.datamodel.platform	IDLCharacterArray	FACE_IDLCharacterArray [Class]
face.datamodel.platform	CharArray	FACE_CharArray [Class]
face.datamodel.platform	Enumeration	FACE_Enumeration [Class]
face.datamodel.platform	IDLNumber	FACE_IDLNumber [Class]
face.datamodel.platform	IDLInteger	FACE_IDLInteger [Class]
face.datamodel.platform	Short	FACE_Short [Class]
face.datamodel.platform	Long	FACE_Long [Class]
face.datamodel.platform	LongLong	FACE_LongLong [Class]
face.datamodel.platform	IDLReal	FACE_IDLReal [Class]
face.datamodel.platform	Double	FACE_Double [Class]
face.datamodel.platform	LongDouble	FACE_LongDouble [Class]
face.datamodel.platform	Float	FACE_Float [Class]
face.datamodel.platform	Fixed	FACE_Fixed [Class]
face.datamodel.platform	IDLUnsignedInteger	FACE_IDLUnsignedInteger [Class]
face.datamodel.platform	UShort	FACE_UShort [Class]
face.datamodel.platform	ULong	FACE_ULong [Class]
face.datamodel.platform	ULongLong	FACE_ULongLong [Class]
face.datamodel.platform	IDLSequence	FACE_IDLSequence [Class]
face.datamodel.platform	IDLArray	FACE_IDLArray [Class]
face.datamodel.platform	IDLStruct	FACE_IDLStruct [Class]
face.datamodel.platform	IDLComposition	FACE_IDLComposition [Property]
face.datamodel.platform	Characteristic	FACE_PlatformCharacteristic [Element]
face.datamodel.platform	Entity	FACE_PlatformEntity [Class] FACE_Realize [Association] FACE_Specialize [Generalization] FACE_PlatformComposition [Property] FACE_SpecializationOwner [Class]
face.datamodel.platform	Composition	FACE_PlatformComposition [Property] FACE_PlatformComposableElement [Element]

FACE Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.datamodel.platform	Association	FACE_PlatformAssociation [Class] FACE_AssociatedParticipant [Association] FACE_Realize [Association] FACE_Specialize [Generalization] FACE_SpecializationOwner [Class]
face.datamodel.platform	Participant	FACE_PlatformParticipant [Class]
face.datamodel.platform	PathNode	FACE_PlatformParticipant [Class]
face.datamodel.platform	ParticipantPathNode	FACE_PlatformParticipant [Class]
face.datamodel.platform	CharacteristicPathNode	FACE_PlatformParticipant [Class]
face.datamodel.platform	View	FACE_PlatformView [Class]
face.datamodel.platform	Query	FACE_PlatformQuery [Class] FACE_Realize [Association]
face.datamodel.platform	CompositeTemplate	FACE_CompositeTemplate [Class] FACE_Realize [Association] FACE_TemplateComposition [Property]
face.datamodel.platform	TemplateComposition	FACE_TemplateComposition [Property] FACE_PlatformView [Class]
face.datamodel.platform	Template	FACE_Template [Class] FACE_BoundQuery [Association] FACE_EffectiveQuery [Association]
face.uop	ClientServerRole	FACE_ClientServerRoleEnum
face.uop	FaceProfile	FACE_ProfileEnum
face.uop	DesignAssuranceLevel	FACE_DesignAssuranceLevelEnum
face.uop	DesignAssuranceStandard	FACE_DesignAssuranceStandardEnum
face.uop	MessageExchangeType	FACE_MessageExchangeTypeEnum
face.uop	PartitionType	FACE_PartitionTypeEnum
face.uop	ProgrammingLanguage	FACE_ProgrammingLanguageEnum
face.uop	SynchronizationStyle	FACE_SynchronizationStyleEnum
face.uop	ThreadType	FACE_ThreadTypeEnum
face	UoPModel	FACE_UoPModel [Package]
face.uop	Element	FACE_UoPElement [Element]
face.uop	SupportingComponent	FACE_SupportingComponent [Class]
face.uop	LanguageRunTime	FACE_LanguageRunTime [Class]
face.uop	ComponentFramework	FACE_ComponentFramework [Class]
face.uop	AbstractUoP	FACE_AbstractUoP [Class] FACE_EndPoint [Association]
face.uop	AbstractConnection	FACE_AbstractConnection [Class] FACE_AbstractView [Association]

FACE Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.uop	UnitOfPortability	FACE_UnitOfPortability [Class] FACE_BackingComponent [Association] FACE_Realize [Association] FACE_UoPResource [Association] FACE_EndPoint [Association] FACE_ComponentTypeEnum FACE_ProgrammingLanguageEnum FACE_DesignAssuranceLevelEnum FACE_DesignAssuranceStandardEnum FACE_PartitionTypeEnum FACE_ProfileEnum
face.uop	PortableComponent	FACE_ComponentTypeEnum FACE_UnitOfPortability [Class]
face.uop	PlatformSpecificComponent	FACE_ComponentTypeEnum FACE_UnitOfPortability [Class]
face.uop	Thread	FACE_Thread [Class]
face.uop	RAMMemoryRequirements	FACE_RAMMemoryRequirements [Class]
face.uop	Connection	FACE_Connection [Class] FACE_TraceabilityModel [Package] FACE_Realize [Association]
face.uop	ClientServerConnection	FACE_ClientServerConnection [Class] FACE_RequestView [Association] FACE_ResponseView [Association]
face.uop	PubSubConnection	FACE_PubSubConnection [Class] FACE_MessageExchangeTypeEnum FACE_MessageType [Association]
face.uop	QueuingConnection	FACE_QueuingConnection [Class]
face.uop	SingleInstanceMessageConnection	FACE_SingleInstanceMessageConnection [Class]
face.uop	LifeCycleManagementPort	FACE_LifeCycleManagementPort [Class] FACE_MessageType [Association]
face	IntegrationModel	FACE_IntegrationModel [Package]
face.integration	Element	FACE_IntegrationElement [Element]
face.integration	IntegrationContext	FACE_IntegrationContext [Package] FACE_TSNodeConnection [InformationFlow] FACE_TransportNode [Class]
face.integration	TSNodeConnection	FACE_TSNodeConnection [InformationFlow]
face.integration	TSNodePortBase	FACE_TSNodePortBase [Class] FACE_TSNodeConnection [InformationFlow]
face.integration	UoPInstance	FACE_UoPInstance [Class] FACE_Realize [Association] FACE_EndPoint [Association]
face.integration	UoPEndPoint	FACE_UoPEndPoint [Class] FACE_Realize [Association]
face.integration	UoPInputEndPoint	FACE_UoPInputEndPoint [Class]
face.integration	UoPOutputEndPoint	FACE_UoPOutputEndPoint [Class]

FACE Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.integration	TransportNode	FACE_TransportNode [Class] FACE_EndPoint [Association]
face.integration	TSNodePort	FACE_TSNodePort [Class] FACE_MessageType [Association]
face.integration	TSNodeInputPort	FACE_TSNodeInputPort [Class]
face.integration	TSNodeOutputPort	FACE_TSNodeOutputPort [Class]
face.integration	ViewAggregation	FACE_ViewAggregation [Class]
face.integration	ViewFilter	FACE_ViewFilter [Class]
face.integration	ViewSource	FACE_ViewSource [Class]
face.integration	ViewSink	FACE_ViewSink [Class]
face.integration	ViewTransformation	FACE_ViewTransformation [Class]
face.integration	ViewTransporter	FACE_ViewTransporter [Class]
face.integration	TransportChannel	FACE_TransportChannel [Class]
face	TraceabilityModel	FACE_TraceabilityModel [Package]
face.traceability	Element	FACE_TraceabilityElement [Element]
face.traceability	TraceableElement	FACE_TraceableElement [Element] FACE_ElementTrace [Association]
face.traceability	TraceabilityPoint	FACE_TraceabilityPoint [Class]
face.traceability	UoPTraceabilitySet	FACE_UoPTraceabilitySet [Class] FACE_UoPTrace [Association]
face.traceability	ConnectionTraceabilitySet	FACE_ConnectionTraceabilitySet [Class] FACE_ConnectionTrace [Association]
Not from the Metamodel, created for System-of-Systems	<Created for System-of-Systems Connectivity>	FACE_OperationalExchange [InformationFlow] FACE_ResourceExchange [InformationFlow]
Not from the Metamodel, created for System-of-Systems	<Derived from FACE Technical Standard, Edition 3.0>	FACE_IOEndpoint [Association] FACE_UnitOfConformance [Class] FACE_UnitOfConformanceEndpoint [Class] FACE_UnitOfConformanceEndpointTypeEnum FACE_UnitOfConformanceTypeEnum FACE_UoCElement [Element] FACE_UoCModel [Package]
Not from the Metamodel, created for UAF Mapping	<Created for UAF Mapping>	FACE_Implements [Dependency]

A.2 FACE Profile to FACE Metamodel Mapping

This section provides a tabular description of the mapping between the FACE Profile for UAF elements to their corresponding FACE 3.0 metamodel elements as well as showing the profile element mappings to UAF elements. (The UAF Mappings are represented by the «FACE_Implements» [Dependency] stereotype and its constraints.) The order of the profile elements in the table corresponds to the package organization of the FACE Profile for UAF specification. The FACE metamodel elements shown are realized in whole or part by the listed FACE Profile for UAF element. The UAF element shown represents the mapping from the FACE Profile for UAF element to a corresponding UAF stereotype in the UAFP. The bracketed strings following the UAF element names are the metatype of the UAFP element and the UAFP package in which the UAF element resides.

Table 8-3 FACE Profile Elements -to- FACE Metamodel Mappings

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile	FACE_AbstractAssociation	Association		
FACE_Profile	FACE_ArchitectureModel	Package	face.ArchitectureModel	
FACE_Profile	FACE_Element	Element	face.Element	
FACE_Profile	FACE_ModelElement	Element	face.Element	
FACE_Profile.FACE Data Architecture	FACE_DataModel	Package	face.DataModel	
FACE_Profile.FACE Data Architecture	FACE_DataModelElement	Element	face.datamodel.Element	
FACE_Profile.FACE Data Architecture	FACE_ElementTrace	Association	face.traceability.TraceableElement	
FACE_Profile.FACE Data Architecture	FACE_EndPoint	Association	face.uop.AbstractUoP face.uop.UnitOfPortability face.integration.UoPInstance face.integration.TransportNode	
FACE_Profile.FACE Data Architecture	FACE_IntegrationModel	Package	face.IntegrationModel	
FACE_Profile.FACE Data Architecture	FACE_MessageType	Association	face.uop.PubSubConnection face.uop.LifeCycleManagementPort face.integration.TSNodePort	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture	FACE_Realize	Association	face.datamodel.logical.Association face.datamodel.logical.Query face.datamodel.logical.Measurement face.datamodel.logical.MeasurementAxis face.datamodel.logical.CompositeQuery face.datamodel.logical.Entity face.datamodel.platform.Association face.datamodel.platform.Entity face.datamodel.platform.IDLType face.datamodel.platform.Query face.datamodel.platform.CompositeTemplate face.uop.UnitOfPortability face.uop.Connection face.integration.UoPInstance face.integration.UoPEndPoint	
FACE_Profile.FACE Data Architecture	FACE_TraceabilityModel	Package	face.TraceabilityModel face.uop.Connection	
FACE_Profile.FACE Data Architecture	FACE_UoPModel	Package	face.UoPModel	
FACE_Profile.FACE Data Architecture.FACE Data Model	FACE_AssociatedParticipant	Association	face.datamodel.conceptual.Association face.datamodel.logical.Association face.datamodel.platform.Association	
FACE_Profile.FACE Data Architecture.FACE Data Model	FACE_ConceptualDataModel	Package	face.datamodel.ConceptualDataModel	
FACE_Profile.FACE Data Architecture.FACE Data Model	FACE_LogicalDataModel	Package	face.datamodel.LogicalDataModel	
FACE_Profile.FACE Data Architecture.FACE Data Model	FACE_PlatformDataModel	Package	face.datamodel.PlatformDataModel	
FACE_Profile.FACE Data Architecture.FACE Data Model	FACE_SpecializationOwner	Class	face.datamodel.conceptual.Entity face.datamodel.conceptual.Association face.datamodel.logical.Association face.datamodel.logical.Entity face.datamodel.platform.Association face.datamodel.platform.Entity	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.FACE Data Model	FACE_Specialize	Generalization	face.datamodel.conceptual.Entity face.datamodel.conceptual.Association face.datamodel.logical.Entity face.datamodel.logical.Association face.datamodel.platform.Entity face.datamodel.platform.Association	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_BasisElement	Element	face.datamodel.conceptual.BasisElement	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_BasisEntity	Class	face.datamodel.conceptual.BasisEntity	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualAssociation	Class	face.datamodel.conceptual.Association	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualCharacteristic	Element	face.datamodel.conceptual.Characteristic	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualComposableElement	Element	face.datamodel.conceptual.ComposableElement face.datamodel.conceptual.Composition	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualCompositeQuery	Class	face.datamodel.conceptual.CompositeQuery	InformationElement [Class] [UAF::Operational::Information]
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualComposition	Property	face.datamodel.conceptual.Composition face.datamodel.conceptual.Entity	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualElement	Element	face.datamodel.conceptual.Element	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualEntity	Class	face.datamodel.conceptual.Entity	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualParticipant	Class	face.datamodel.conceptual.Participant face.datamodel.conceptual.CharacteristicPathNode face.datamodel.conceptual.ParticipantPathNode face.datamodel.conceptual.PathNode	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualQuery	Class	face.datamodel.conceptual.Query	InformationElement [Class] [UAF::Operational::Information]
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualQueryComposition	Property	face.datamodel.conceptual.QueryComposition face.datamodel.conceptual.CompositeQuery	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualView	Class	face.datamodel.conceptual.View face.datamodel.conceptual.QueryComposition	InformationElement [Class] [UAF::Operational::Information]
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_Domain	Class	face.datamodel.conceptual.Domain	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_EntityBasis	Generalization	face.datamodel.conceptual.Entity	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_Observable	Class	face.datamodel.conceptual.Observable	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_AbstractMeasurement	Element	face.datamodel.logical.AbstractMeasurement	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_AbstractMeasurementSystem	Class	face.datamodel.logical.AbstractMeasurementSystem	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_AffineConversion	Class	face.datamodel.logical.AffineConversion	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_AppliedConstraint	Association	face.datamodel.logical.Measurement face.datamodel.logical.MeasurementAxis face.datamodel.logical.MeasurementSystem face.datamodel.logical.MeasurementSystemAxis face.datamodel.logical.ValueTypeUnit	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_AppliedValueTypeUnit	Association	face.datamodel.logical.MeasurementAxis face.datamodel.logical.MeasurementSystemAxis	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_Axis	Association	face.datamodel.logical.CoordinateSystem face.datamodel.logical.Measurement face.datamodel.logical.MeasurementSystem	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_Constraint	Class	face.datamodel.logical.Constraint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_Conversion	Class	face.datamodel.logical.Conversion	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_ConvertibleElement	Element	face.datamodel.logical.ConvertibleElement	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_CoordinateSystem	Class	face.datamodel.logical.CoordinateSystem	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_CoordinateSystemAxis	Class	face.datamodel.logical.CoordinateSystemAxis	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_DefinedReferencePoint	Association	face.datamodel.logical.MeasurementSystem	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_EnumerationConstraint	Class	face.datamodel.logical.EnumerationConstraint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_EnumerationLabel	Property	face.datamodel.logical.EnumerationLabel	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_FixedLengthStringConstraint	Class	face.datamodel.logical.FixedLengthStringConstraint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_IntegerConstraint	Class	face.datamodel.logical.IntegerConstraint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_IntegerRangeConstraint	Class	face.datamodel.logical.IntegerRangeConstraint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_Landmark	Class	face.datamodel.logical.Landmark	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalAssociation	Class	face.datamodel.logical.Association	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalCharacteristic	Element	face.datamodel.logical.Characteristic	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalComposableElement	Element	face.datamodel.logical.ComposableElement face.datamodel.logical.Composition	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalCompositeQuery	Class	face.datamodel.logical.CompositeQuery	InformationElement [Class] [UAF::Operational::Information]
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalComposition	Property	face.datamodel.logical.Composition face.datamodel.logical.Entity	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalElement	Element	face.datamodel.logical.Element	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalEntity	Class	face.datamodel.logical.Entity	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalParticipant	Class	face.datamodel.logical.CharacteristicPathNode face.datamodel.logical.Participant face.datamodel.logical.ParticipantPathNode face.datamodel.logical.PathNode	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalQuery	Class	face.datamodel.logical.Query	InformationElement [Class] [UAF::Operational::Information]
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalQueryCompositio n	Property	face.datamodel.logical.QueryComposition face.datamodel.logical.CompositeQuery	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalValueType	Class	face.datamodel.logical.Boolean face.datamodel.logical.Character face.datamodel.logical.Enumerated face.datamodel.logical.Integer face.datamodel.logical.Natural face.datamodel.logical.NonNegativeReal face.datamodel.logical.Numeric face.datamodel.logical.Real face.datamodel.logical.String	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalView	Class	face.datamodel.logical.View face.datamodel.logical.QueryComposition	InformationElement [Class] [UAF::Operational::Information]
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_Measurement	Class	face.datamodel.logical.Measurement	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_MeasurementAttribute	Property	face.datamodel.logical.MeasurementAttribute	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_MeasurementAxis	Class	face.datamodel.logical.MeasurementAxis	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_MeasurementConstraint	Class	face.datamodel.logical.MeasurementConstraint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_MeasurementConversio n	Class	face.datamodel.logical.MeasurementConversion	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_MeasurementSystem	Class	face.datamodel.logical.MeasurementSystem	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_MeasurementSystemAxi s	Class	face.datamodel.logical.MeasurementSystemAxis	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_MeasurementSystemConversion	Class	face.datamodel.logical.MeasurementSystemConversion	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_RealConstraint	Class	face.datamodel.logical.RealConstraint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_RealRangeConstraint	Class	face.datamodel.logical.RealRangeConstraint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_ReferencePoint	Class	face.datamodel.logical.ReferencePoint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_ReferencePointPart	Class	face.datamodel.logical.ReferencePointPart	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_RegularExpressionConstraint	Class	face.datamodel.logical.RegularExpressionConstraint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_RPPart	Association	face.datamodel.logical.ReferencePoint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_StandardMeasurementSystem	Class	face.datamodel.logical.StandardMeasurementSystem	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_StringConstraint	Class	face.datamodel.logical.StringConstraint	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_Unit	Class	face.datamodel.logical.Unit	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_ValueTypeEnum		face.datamodel.logical.ValueType face.datamodel.logical.Boolean face.datamodel.logical.Character face.datamodel.logical.Enumerated face.datamodel.logical.Integer face.datamodel.logical.Natural face.datamodel.logical.NonNegativeReal face.datamodel.logical.Numeric face.datamodel.logical.Real face.datamodel.logical.String	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_ValueTypeUnit	Class	face.datamodel.logical.ValueTypeUnit	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Boolean	Class	face.datamodel.platform.Boolean	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_BoundedString	Class	face.datamodel.platform.BoundedString	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_BoundQuery	Association	face.datamodel.platform.Template	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Char	Class	face.datamodel.platform.Char	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_CharArray	Class	face.datamodel.platform.CharArray	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_CharType	Class	face.datamodel.platform.CharType	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_CompositeTemplate	Class	face.datamodel.platform.CompositeTemplate	DataElement [Class] [UAF::Resources::Information]
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Double	Class	face.datamodel.platform.Double	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_EffectiveQuery	Association	face.datamodel.platform.Template	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Enumeration	Class	face.datamodel.platform.Enumeration	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Fixed	Class	face.datamodel.platform.Fixed	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Float	Class	face.datamodel.platform.Float	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLArray	Class	face.datamodel.platform.IDLArray	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLBoundedString	Class	face.datamodel.platform.IDLBoundedString	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLCharacterArray	Class	face.datamodel.platform.IDLCharacterArray	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLComposition	Property	face.datamodel.platform.IDLComposition	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLInteger	Class	face.datamodel.platform.IDLInteger	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLNumber	Class	face.datamodel.platform.IDLNumber	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLPrimitive	Class	face.datamodel.platform.IDLPrimitive	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLReal	Class	face.datamodel.platform.IDLReal	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLSequence	Class	face.datamodel.platform.IDLSequence	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLStruct	Class	face.datamodel.platform.IDLStruct	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLType	Element	face.datamodel.platform.IDLType	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLUnboundedString	Class	face.datamodel.platform.IDLUnboundedString	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_IDLUnsignedInteger	Class	face.datamodel.platform.IDLUnsignedInteger	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Long	Class	face.datamodel.platform.Long	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_LongDouble	Class	face.datamodel.platform.LongDouble	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_LongLong	Class	face.datamodel.platform.LongLong	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Octet	Class	face.datamodel.platform.Octet	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PhysicalDataType	Element	face.datamodel.platform.PhysicalDataType	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformAssociation	Class	face.datamodel.platform.Association	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformCharacteristic	Element	face.datamodel.platform.Characteristic	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformComposableElement	Element	face.datamodel.platform.ComposableElement face.datamodel.platform.Composition	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformComposition	Property	face.datamodel.platform.Composition face.datamodel.platform.Entity	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformElement	Element	face.datamodel.platform.Element	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformEntity	Class	face.datamodel.platform.Entity	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformParticipant	Class	face.datamodel.platform.CharacteristicPathNode face.datamodel.platform.PathNode face.datamodel.platform.ParticipantPathNode face.datamodel.platform.Participant	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformQuery	Class	face.datamodel.platform.Query	DataElement [Class] [UAF::Resources::Information]
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformView	Class	face.datamodel.platform.View face.datamodel.platform.TemplateComposition	DataElement [Class] [UAF::Resources::Information]
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Short	Class	face.datamodel.platform.Short	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_String	Class	face.datamodel.platform.String	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_StringType	Class	face.datamodel.platform.StringType	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Template	Class	face.datamodel.platform.Template	DataElement [Class] [UAF::Resources::Information]
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_TemplateComposition	Property	face.datamodel.platform.TemplateComposition face.datamodel.platform.CompositeTemplate	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_ULong	Class	face.datamodel.platform.ULong	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_ULongLong	Class	face.datamodel.platform.ULongLong	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_UShort	Class	face.datamodel.platform.UShort	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.Integration Model	FACE_IntegrationContext	Package	face.integration.IntegrationContext	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_IntegrationElement	Element	face.integration.Element	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TransportChannel	Class	face.integration.TransportChannel	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TransportNode	Class	face.integration.IntegrationContext face.integration.TransportNode	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TSNodeConnection	InformationFlow	face.integration.IntegrationContext face.integration.TSNodeConnection face.integration.TSNodePortBase	ResourceConnector [Connector] [UAF::Resources::Connectivity]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TSNodeInputPort	Class	face.integration.TSNodeInputPort	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TSNodeOutputPort	Class	face.integration.TSNodeOutputPort	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TSNodePort	Class	face.integration.TSNodePort	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TSNodePortBase	Class	face.integration.TSNodePortBase	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_UoPEndPoint	Class	face.integration.UoPEndPoint	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_UoPInputEndPoint	Class	face.integration.UoPInputEndPoint	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_UoPInstance	Class	face.integration.UoPInstance	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_UoPOutputEndPoint	Class	face.integration.UoPOutputEndPoint	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewAggregation	Class	face.integration.ViewAggregation	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewFilter	Class	face.integration.ViewFilter	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewSink	Class	face.integration.ViewSink	Software [Class] [UAF::Resources::Taxonomy]

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewSource	Class	face.integration.ViewSource	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewTransformation	Class	face.integration.ViewTransformation	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewTransporter	Class	face.integration.ViewTransporter	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_ConnectionTrace	Association	face.traceability.ConnectionTraceabilitySet	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_ConnectionTraceabilitySet	Class	face.traceability.ConnectionTraceabilitySet	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_TraceabilityElement	Element	face.traceability.Element	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_TraceabilityPoint	Class	face.traceability.TraceabilityPoint	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_TraceableElement	Element	face.traceability.TraceableElement	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_UoPTrace	Association	face.traceability.UoPTraceabilitySet	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_UoPTraceabilitySet	Class	face.traceability.UoPTraceabilitySet	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_AbstractConnection	Class	face.uop.AbstractConnection	OperationalPerformer [Class] [UAF::Operational::Structure]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_AbstractUoP	Class	face.uop.AbstractUoP	OperationalPort [Port] [UAF::Operational::Structure]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_AbstractView	Association	face.uop.AbstractConnection	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_BackingComponent	Association	face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ClientServerConnection	Class	face.uop.ClientServerConnection	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ClientServerRoleEnum		face.uop.ClientServerRole	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ComponentFramework	Class	face.uop.ComponentFramework	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ComponentTypeEnum		face.uop.UnitOfPortability face.uop.PlatformSpecificComponent face.uop.PortableComponent	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_Connection	Class	face.uop.Connection	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_DesignAssuranceLevelEnum		face.uop.DesignAssuranceLevel face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_DesignAssuranceStandardEnum		face.uop.DesignAssuranceStandard face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_LanguageRunTime	Class	face.uop.LanguageRunTime	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_LifeCycleManagementPort	Class	face.uop.LifeCycleManagementPort	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_MessageExchangeTypeEnum		face.uop.MessageExchangeType face.uop.PubSubConnection	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_PartitionTypeEnum		face.uop.PartitionType face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ProfileEnum		face.uop.FaceProfile face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ProgrammingLanguageEnum		face.uop.ProgrammingLanguage face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_PubSubConnection	Class	face.uop.PubSubConnection	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_QueuingConnection	Class	face.uop.QueuingConnection	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_RAMMemoryRequirements	Class	face.uop.RAMMemoryRequirements	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_RequestView	Association	face.uop.ClientServerConnection	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ResponseView	Association	face.uop.ClientServerConnection	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.UoP Model	FACE_SingleInstanceMessageConnection	Class	face.uop.SingleInstanceMessageConnection	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_SupportingComponent	Class	face.uop.SupportingComponent	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_SynchronizationStyleEnum		face.uop.SynchronizationStyle	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_Thread	Class	face.uop.Thread	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ThreadTypeEnum		face.uop.ThreadType	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_UnitOfPortability	Class	face.uop.UnitOfPortability face.uop.PlatformSpecificComponent face.uop.PortableComponent	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_UoPElement	Element	face.uop.Element	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_UoPResource	Association	face.uop.UnitOfPortability	
FACE_Profile.UAF_FACE_Extended_Stereotypes	FACE_Implements	Dependency	<Not from the Metamodel, created for UAF Mapping>	
FACE_Profile.UAF_FACE_Extended_Stereotypes	FACE_IOEndpoint	Association	<Not from the Metamodel, derived from FACE Technical Standard, Edition 3.0 for System-of-Systems >	
FACE_Profile.UAF_FACE_Extended_Stereotypes	FACE_OperationalExchange	InformationFlow	<Not from the Metamodel, created for System-of-Systems Connectivity>	OperationalExchange [InformationFlow] [UAF::Operational::Connectivity]
FACE_Profile.UAF_FACE_Extended_Stereotypes	FACE_ResourceExchange	InformationFlow	<Not from the Metamodel, created for System-of-Systems Connectivity>	ResourceExchange [InformationFlow] [UAF::Resources::Connectivity]
FACE_Profile.UAF_FACE_Extended_Stereotypes	FACE_UnitOfConformance	Class	<Not from the Metamodel, derived from FACE Technical Standard, Edition 3.0 for System-of-Systems >	Software [Class] [UAF::Resources::Taxonomy]
FACE_Profile.UAF_FACE_Extended_Stereotypes	FACE_UnitOfConformanceEndpoint	Class	<Not from the Metamodel, derived from FACE Technical Standard, Edition 3.0 for System-of-Systems >	ResourcePort [Port] [UAF::Resources::Structure]
FACE_Profile.UAF_FACE_Extended_Stereotypes	FACE_UnitOfConformanceEndpointTypeEnum		<Not from the Metamodel, derived from FACE Technical Standard, Edition 3.0 for System-of-Systems >	
FACE_Profile.UAF_FACE_Extended_Stereotypes	FACE_UnitOfConformanceTypeEnum		<Not from the Metamodel, derived from FACE Technical Standard, Edition 3.0 for System-of-Systems >	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.UAF_FACE_Extended_Stereotypes	FACE_UoCElement	Element	<Not from the Metamodel, derived from FACE Technical Standard, Edition 3.0 for System-of-Systems >	
FACE_Profile.UAF_FACE_Extended_Stereotypes	FACE_UoCModel	Package	<Not from the Metamodel, derived from FACE Technical Standard, Edition 3.0 for System-of-Systems >	

This page intentionally left blank.