Date: December 2023

FACE Profile

Version 2.0 – beta 1

OMG Document Number:

Standard document URL: https://www.omg.org/spec/FACE/

This OMG document replaces the submission document (c41/2023-06-08). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome and should be directed to issues@omg.org by December 11, 2023.

You may view the pending issues for this specification from the OMG revision issues web page https://issues.omg.org/issues/lists.

The FTF Recommendation and Report for this specification will be published in September 2024. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

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

LICENSES

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

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

PATENTS

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

GENERAL USE RESTRICTIONS

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

DISCLAIMER OF WARRANTY

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

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

RESTRICTED RIGHTS LEGEND

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

OMG

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

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

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

OMG Specifications

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

https://www.omg.org/spec

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

OMG Headquarters
9C Medway Road, PMB 274
Milford, MA 01757
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*
Certain OMG specifications are also available as ISO standards. Please consult https://www.iso.org

1 Scope

This specification defines a profile to express The Open Group® Future Airborne Capability Environment (FACE™)[1] Technical Standard, Edition 3.1 and associated Meta-Object Facility (MOF) data architecture metamodel in terms of the Object Management Group's (OMG) Unified Modeling Language (UML) metamodel, with extensions to connect FACE elements to appropriate elements of the Unified Architecture Framework (UAF).

Unless otherwise explicitly stated, all references in this document to the FACE Technical Standard shall be interpreted as references to the Future Airborne Capability Environment (FACE) Technical Standard, Edition 3.1 as listed in the References section of this document.

The data model portion of the FACE Technical Standard is based upon The Open Group Open Universal Domain Description Language (Open UDDL™)[2], Edition 1.0. As such, this standard also references the Open UDDL Standard. Unless otherwise explicitly stated, all references in this document to the UDDL Standard shall be interpreted as reference to The Open Group Open Universal Domain Description Language (Open UDDL), Edition 1.0 as listed in the References section of this document.

1.1 FACE Profile Background

The FACE Profile v2.0 specification defines a profile to express the FACE Technical Standard and its underlying UDDL Standard as expressed in their Meta-Object Facility (MOF) data architecture metamodels in terms of the Object Management Group's (OMG) Unified Architecture Framework (UAF). This profile is purposefully designed to be loosely coupled with the UAF standard and expresses FACE metamodel elements as UML with relationships to connect appropriate FACE metamodel elements to UAF profile elements. The UML portion of this standard can be stand-alone or paired with the UAF-specific extensions.

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."[3] The focus of the FACE Technical Standard is the support of real-time and safety critical software beginning with avionics, representing the software elements as modules in a layered architecture with defined interfaces between the layers. The software elements are meant to be separable and replaceable to fit changing contexts and requirements. As a result of its focus on portable software components (Units of Portability, or UoPs), the FACE Technical Standard is primarily concerned with individual components, rather than the larger contexts into which they will be integrated. Because semantic understanding of message data is so important to integration of system components, The FACE Technical Standard includes the UDDL Standard to define the data semantics underlying the elements in its message definitions. 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."[4]

The UAF standard provides the larger scope to describe the environments into which the FACE-described components fit. The Open Group UDDL Standard defines the elements, attributes, and associations for the FACE Data Architecture. The Open Group FACE Technical Standard leverages 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

[1] FACE™ is a trademark of The Open Group®.
[2] Open UDDL™ is a trademark of The Open Group®
[3] FACE FAQs | The Open Group. (2022). Opengroup.org. Retrieved 28 December 2022, from https://www.opengroup.org/content/future-airborne-capability-environment-face/faqs
[4] "Unified Architecture Framework® (UAF®) | Object Management Group. (2022). omg.org. Retrieved 28 December 2022, from https://www.omg.org/uaf/index.htm"

cohesive systems architectures.

Together, the FACE Profile and its UAF extensions 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 achieve "the benefits of interoperability, affordability, portability, increased competition and improved time-to-field"[5] promised by the FACE Technical Standard.

1.2 Intended Users

The profile enables the modeling of FACE components, data descriptions, data exchanges, integration elements, and traceability mechanisms using the UML metamodel and in the context of system-of-systems airframe architectures described in UAF. 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.

2 Conformance

The FACE Profile contains a separable portion that is dependent upon UML, and another portion that is dependent upon UAF for the connectivity to a larger systems-of-systems architecture. It defines constraints that do not conflict with application of Unified Architecture Framework (UAF) Profile (UAFP) stereotypes. There are three levels of conformance designated for the FACE Profile. The requirements for a tool to be considered as conformant with the FACE Profile at each level of conformance are detailed below.

The Conformance clause identifies which clauses of the RFC are mandatory (or conditionally mandatory) and which are optional in order for an implementation to claim conformance to the RFC.

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. The core UML elements of the profile (the FACE metamodel expressed as UML) may be separated from the UAF connection extensions for implementation as two related profiles, with the UAF extension profile dependent upon both the UAF and FACE/UML profiles
XMI data exchange	Provide XMI import and export (.xmi) of the user model and profile, including UML representations of FACE elements and UAF extensions
Fidelity of XMI exchange	Be able to import and export FACE Profile 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

[5] FACE FAQs | The Open Group. (2022). Opengroup.org. Retrieved 28 December 2022, from https://www.opengroup.org/content/future-airborne-capability-environment-face/faqs

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 Specifications for UDDL and FACE delivered with this document (document c4i/23-05-11)
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. For details of OCL Constraints, please refer to the UDDL Standard and the FACE Technical Standard.
FACE Conformance Checks in tool	FACE Conformance checking in tool using FACE Conformance/OCL Constraints

3 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 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.

List of normative references.

3.1.1 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, http://www.omg.org/spec/UML

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

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

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

Unified Architecture Framework (UAF), v1.2, July 2022, 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/

3.1.2 The Open Group Documents (Normative References)

While all documents published by The Open Group are freely available for download, The Open Group requires that users register for and use an Open Group account to download documents. Registration and document access are no-cost.

The Open Group normative references that apply to this standard are:

- FACE Technical Standard, Edition 3.1
    - Open Group FACE™ Consortium, The Open Group FACE™ (Future Airborne Capability Environment) Technical Standard, Edition 3.1, 28 July 2020, accessed 19-May-2023, ISBN: 1-947754-61-4 <https://publications.opengroup.org/standards/face/c207>
    - Unless otherwise explicitly stated, all references in this document to the FACE Technical Standard shall be interpreted as references to the Future Airborne Capability Environment (FACE) Technical Standard, Edition 3.1.
    - 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, including UDDL metamodel referenced by the FACE metamodel. Section J of the FACE Technical Standard also includes conformance criteria expressed as OCL statements. The FACE conformance criteria extend the UDDL conformance criteria in addition to introduction of conformance criteria specific to elements only found in the FACE Technical Standard.
- Open Universal Domain Description Language (Open UDDL), Edition 1.0
    - Open Group FACE™ Consortium, The Open Group Standard for the Open Universal Domain Description Language (Open UDDL), Edition 1.0, 03 July 2019, accessed 19 May 2023, ISBN: 1-947754-32-4, <https://publications.opengroup.org/standards/face/ c198>
    - Unless otherwise explicitly stated, all references in this document to the UDDL Standard shall be interpreted as reference to The Open Group Open Universal Domain Description Language (Open UDDL), Edition 1.0.
    - The written UDDL standard remains the normative standard for the FACE Technical Standard's Data Model Architecture. The purpose of the Universal Domain Description Language (UDDL) is to define a data modeling language for formally describing, querying, and communicating information. The written UDDL standard (in conjunction with the FACE Technical Standard) provide both language definition information and conformance criteria expressed as OCL statements. The profile presented in this standard follows the metamodel in section 7 of the UDDL Standard 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 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 2.1.1, 21 June 2017, accessed 19 May 2023, <http://www.opengroup.org/library/c176>
    - As mentioned in the UDDL Standard discussion, 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 metamodels in the above-listed UDDL Standard and FACE Technical Standard for all elements other than the UDDL 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 and the FACE XMI (face) file are required to translate between the two notations.

3.2 Non-normative References

List of non-normative references.

- FACE 3rd Party Tools
    - Tool listings and links found under URL: https://www.opengroup.org/face/third-party-tools. These tools are previous proof-of-concept implementations of the FACE metamodel as a UML-based profile. They are referenced to provide prospective implementers of the profile working examples of the profile.
    - MagicDraw / Cameo (NoMagic) Model Tool Integration (MTI) for FACE™ 3.1 Data Modeling NAVAIR public release 2022-554, accessed 19 May 2023, <https://archive.isis.vanderbilt.edu/sites/default/files/face_products/MTI/FACE31_MagicDraw_MTI_v2022_03_1_A.zip> This plug-in is an implementation of most of the UML portion of this standard and serves as a proof of concept for the standard.
    - FACE Edition 2.1 EA Data Model Profile and Plugins NAVAIR Public Release 2015-746, accessed 19 May 2023, <https://archive.isis.vanderbilt.edu/sites/default/files/face_products/MTI/FACE31_MagicDraw_MTI_v2022_03_1_A.zip>
    - FACE Edition 2.1 Rhapsody Data Model Profile and Plugins NAVAIR Public Release 2015-746, accessed 19 May 2023, <https://archive.isis.vanderbilt.edu/sites/default/files/face_products/downloads/FACE21RhapsodyPlugins.zip>
- FACE Consortium Conformance Publications & Tools
    - The conformance publications are listed as assistance to implementers of this profile. The conformance rules would be implemented by implementers that extend to Level AAA conformance, and the Conformance Test Suite would be used to verify that a FACE file that the tool considers to be conformant passes the FACE standard's conformance tests.
    - These publications and tools are listed located at URL: https://www.opengroup.org/face/docsandtools#collapse31, accessed 19 May 2023
    - Link to FACE™ Conformance Verification Matrix, Edition 3.1 (Revision A)

    - Link to the FACE Conformance Test Suites page that includes conformance test suites for FACE Edition 3.1
    - Link to the FACE Reference Implementation Guide (RIG) for FACE Technical Standard Edition 3.0 Volume 3 (Data Architecture)

- FACE New Users Resources
    - These resources available at URL: https://www.opengroup.org/face/softwaresuppliers, accessed 19 May 2023
    - Software Suppliers Guide
    - 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), currently only available to FACE consortium members pending global public release authorizations, <https://www.opengroup.org/face/balsa>

4 Terms and Definitions

For the purposes of this specification, the following terms and definitions apply.

Table 4-1 Acronyms in the Specification

AADL	Architecture Analysis & Design Language
ARINC	Avionics Application Standard Software Interface
BALSA	Basic Avionics Lightweight Source Archetype
CTS	Conformance Test Suite
CVM	Conformance Verification Matrix
DAL	Design Assurance Level
DoDAF	Department of Defense Architecture Framework
EA	(Sparx) Enterprise Architect
EASA	European Aviation Safety Agency
EMOF	Essential Meta-Object Facility
FAA	(U.S.) Federal Aviation Administration
FACE	Future Airborne Capability Environment
GCM	General Component Model
IOSS	Input/Output Services Segment
MARTE	Modeling and Analysis of Real-Time and Embedded systems
MODAF	(British) Ministry of Defence Architecture Framework
MOF	Meta-Object Facility
MTI	Model Tool Integration
NATO	North Atlantic Treaty Organization
OCL	Object Constraint Language
OSS	Operating System Segment
PCS	Portable Components Segment
PSSS	Platform-Specific Services Segment
RFP	Request For Proposals
RIG	Reference Implementation Guide
RTCA	Radio Technical Commission for Aeronautics
STRIDE	Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege
TSS	Transport Services Segment
UDDL	Open Universal Domain Description Language
UAF	Unified Architecture Framework

UML	Unified Modeling Language
UoC	Unit of Conformance (a FACE software component)
UoP	Unit of Portability (a cohesive unit that represents a FACE software component)
XMI	XML Metadata Interchange
XML	eXtensible Markup Language

5 Symbols

No new symbols have been required to create this specification.

6 Additional Information

6.1 Scope of this Specification

This specification covers the entire scope of the FACE Technical Standard metamodel, which includes the UDDL Standard metamodel. This specification additionally includes references to the FACE Technical Standard and UDDL Standard OCL.

Rationale for complete FACE Metamodel:

- Inclusion of the data model portion of the FACE metamodel enables expression of the semantically rigorous descriptions of the data being passed in FACE UoP messages
- The FACE Traceability Model is dependent on FACE Data Model elements
- There are existing 3rd-party UML-based profiles and import/export plugins that extend to the entire scope of the FACE metamodel
- There are existing 3rd-party Import/Export plugins provide exchange between tool-authored data architecture and "gold standard" FACE XMI format The FACE standard specifies that the file format for storage, exchange, and use of FACE Data Architecture files conform to the metamodels described in the FACE and UDDL standards. This "gold standard" format is an XMI file format that describes the data, Units of Conformance (UoPs - FACE software components) , and optionally integration and traceability information required for deployment of FACE UoPs. The ".face file XML data exchange" compliance point specified for conformance level AAA of this standard enables ingestion of .face files into models that use this profile and production of .face files from models that use this profile. As a result of implementing .face format import/export, FACE data architectures exchanged between different organizations that conform to the FACE standard can be viewed, modified, and exported between models implementing this profile and systems that implement the FACE standard.

6.2 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 UDDL Standard and the FACE Technical Standard that are the basis for the elements contained herein. These specifications include the Data Architecture specifications, Object Constraint Language (OCL) rules, and EMOF metamodels that govern the FACE artifacts, and from which all elements of this specification have been derived. After that, the UAF 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, UDDL and FACE standards provide the basic constructs used to define the FACE Profile.

6.2.1 Content Notes for this Specification

In the interest of avoiding potential inconsistencies between this specification and the FACE Technical Standard, 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 either the UDDL Standard or 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 UDDL and FACE metamodels. The content of those descriptions otherwise remains unchanged from the corresponding descriptions in the UDDL and FACE metamodels.

This specification introduces some abstract elements not found in the UDDL and FACE metamodels. 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.

6.2.2 Representing Additional Properties and Constraints on Stereotypes

The FACE Profile 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 specification diagrams for visualization purposes only. The representation in the standard varies by dependency stereotype:

- A 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.

6.2.2.1 FACE Conformance/OCL Constraints

The FACE Conformance/OCL Constraints represented in this standard are representations of the OCL Constraints listed in the UDDL Standard and FACE Technical Standard. 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 UDDL and FACE Conformance/OCL Constraints descriptions have been taken from the UDDL Standard and FACE Technical Standard, 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 UDDL and FACE Conformance/OCL Constraints, see the appropriate subsections of the UDDL Standard and FACE Technical Standard.

6.2.2.2 Metaconstraint Dependency

«metaconstraint» is a stereotype that extends the Dependency metaclass. It is used to specify constrained elements within the profile and is not part of the profile itself.

6.2.2.2.1 Definition of the Metaconstraint Dependency Stereotype

metaconstraint

Package: stereotyped dependencies

isAbstract: No

Extension: Dependency

Description

«metaconstraint» is a stereotype that extends the Dependency metaclass. It has been created for the purpose of expressing the FACE Profile 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 properties. The umlRole Tag relates to a property of the meta-type for the source stereotype, and is used in diagrams to provide a visual indication that there is a constraint on the property defined in the source stereotype for the dependency. To fully understand the «metaconstraint», the reader must review the Constraints applied to the stereotype that is its source.

Note – When stereotype extends Association or Dependency, the stereotype property umlRole has values " memberEnd[[0].role" and/or “memberEnd[1].role." The square bracketed number after the memberEnd indicates whether the metaconstraint applies to the originating end (memberEnd[0]) or the target end (memberEnd[1]) of the relationship. This convention is consistent with the subscript syntax in the Java and other programming languages.

For example metaconstraint umlRole = “memberEnd[1].multiplicity” with constraint text “memberEnd[1].multiplicity shall be 1” should be interpreted as the “the multiplicity at the target end of the stereotyped Association shall be constrained to be exactly 1”.

Figure 6-1: metaconstraint Dependency (specification stereotype)

Attributes

umlRole :	String []	UML Role (property) of the source of the Dependency that is to be constrained by a same-named Constraint applied to the stereotype. If the target of the metaconstraint is a different type than the source, the property identified by UML Role may be typed by the target type.

6.2.2.2.2 Example Usage of the Metaconstraint Dependency

An example of the «metaconstraint» dependency is a diagram for a stereotype extending the Association metaclass.

The diagram shows «Example_Association» with «metaconstraint» Dependencies that indicate constraints on the endpoint types, and the multiplicity, aggregation, and name properties of memberEnd[1]. The stereotype definition for «Example_Association» includes applied constraints named to match the metaconstraint umlRole tagged values. The plain-English constraint definitions for those constraints are shown in the anchored text box in the diagram.

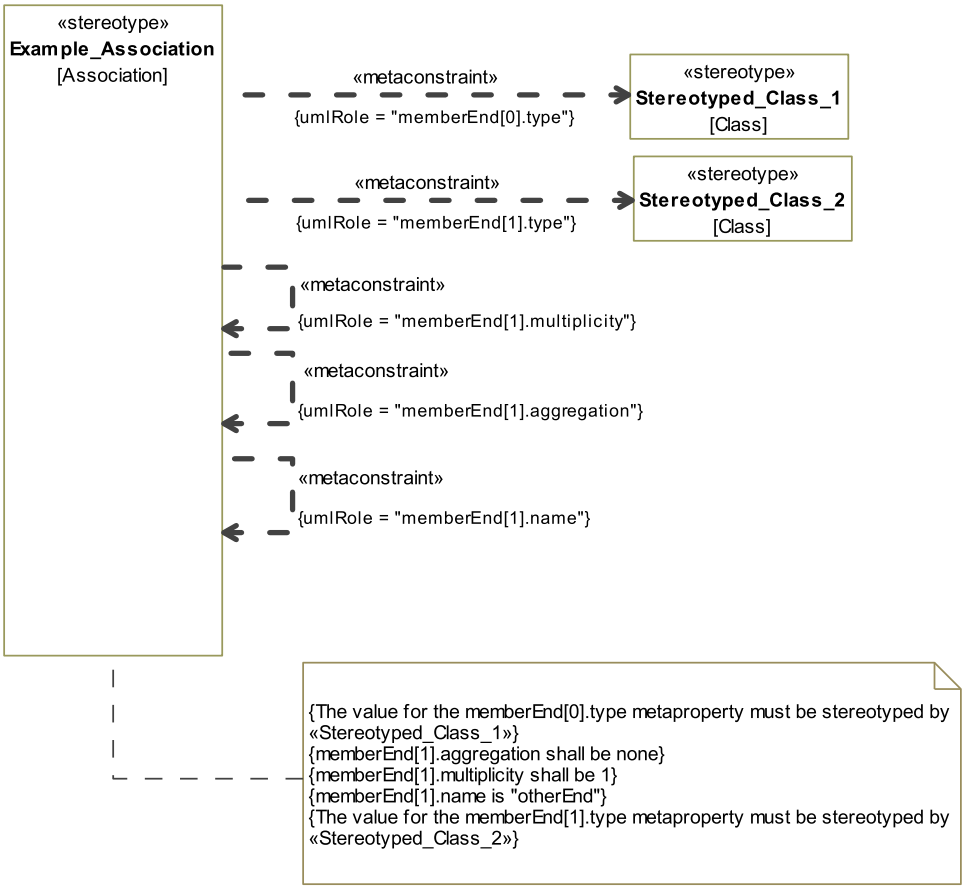

Figure 6-2 Use of «metaconstraint» dependency

6.2.2.3 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 and other elements.

6.2.2.3.1 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 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. To fully understand the relationship, the reader must examine the stereotyped relationship named in the tagged value.

Figure 6-3: stereotyped relationship

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.

6.2.2.3.2 Example Usage of the Stereotyped Relationship Dependency

The example diagram shows two different representations for a dependency stereotyped by «stereotyped relationship». The upper image shows how the «stereotyped relationship» would appear in the definition of «Stereotyped_Class_1» that has an «Example_Dependency» on «Stereotyped_Class_2». This enables readers of this standard to see that an «Example_Dependency» could be defined between the two types of shown elements.

The lower image shows how the «stereotyped relationship» would look in the diagram for the definition of «Example_Dependency». In this case the information may be redundant but provides an explicit visual showing that the «Example_Dependency» relationship is defined as a way to connect the two classes. The Constraints identified in the «metaconstraint» Dependencies would further refine the «Example_Dependency» relationship between «Stereotyped_Class_1» and «Stereotyped_Class_2».

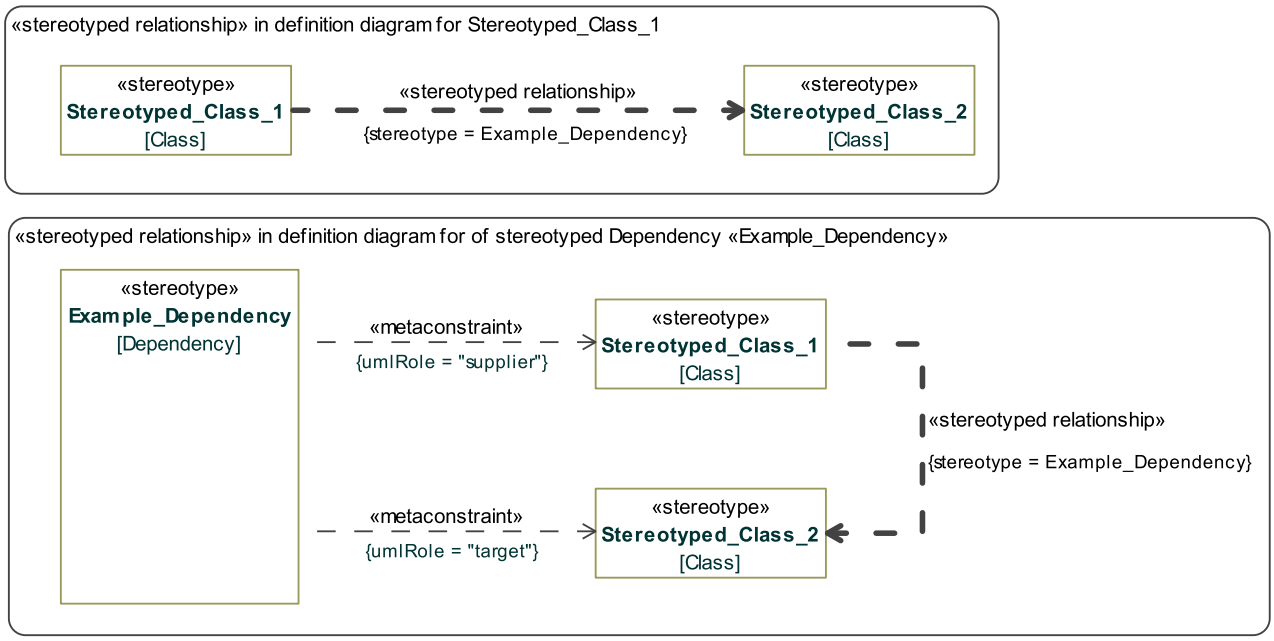

Figure 6-4 Uses of «stereotyped relationship» dependency

6.2.2.4 Stereotyped Association Dependency

There are several stereotypes in the profile specification that have the Metaclass Association. There is no profiling mechanism to visually express that the Association between a source and target is characterized by a specific (constrained) Association stereotype.. As a result, the FACE Profile standard introduces a notation to identify Associations between FACE Profile elements that are stereotyped by FACE Profile Association Stereotypes. This information is represented using «stereotyped association» dependencies.

6.2.2.4.1 Definition of the Stereotyped Association Dependency Stereotype

stereotyped association

Package: stereotyped dependencies

isAbstract: No

Extension: Dependency

Description

«stereotyped association» is a stereotype of Dependency. It has been created for the purpose of expressing the FACE Profile 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). To fully understand the association, the reader must examine the stereotyped association named in the tagged value.

Figure 6-5: stereotyped association

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.

6.2.2.4.2 Example Usage of the Stereotyped Association Dependency

The example diagram for «stereotyped association» shows two different representations for a dependency stereotyped by «stereotyped association». The upper image shows how the «stereotyped association» would appear in the definition of «Stereotyped_Class_1» that has an «Example_Association» with «Stereotyped_Class_2». This enables readers of this standard to see that an «Example_Association» could be defined between the two types of shown elements. The lower image shows how the «stereotyped association» would look in the diagram for the definition of «Example_Association». In this case the information may be redundant, but provides an explicit visual showing that the «Example_Association» relationship is one way to connect the two classes. The Constraints of «Example_Association» further characterize the defined relationship between «Stereotyped_Class_1» and «Stereotyped_Class_2».

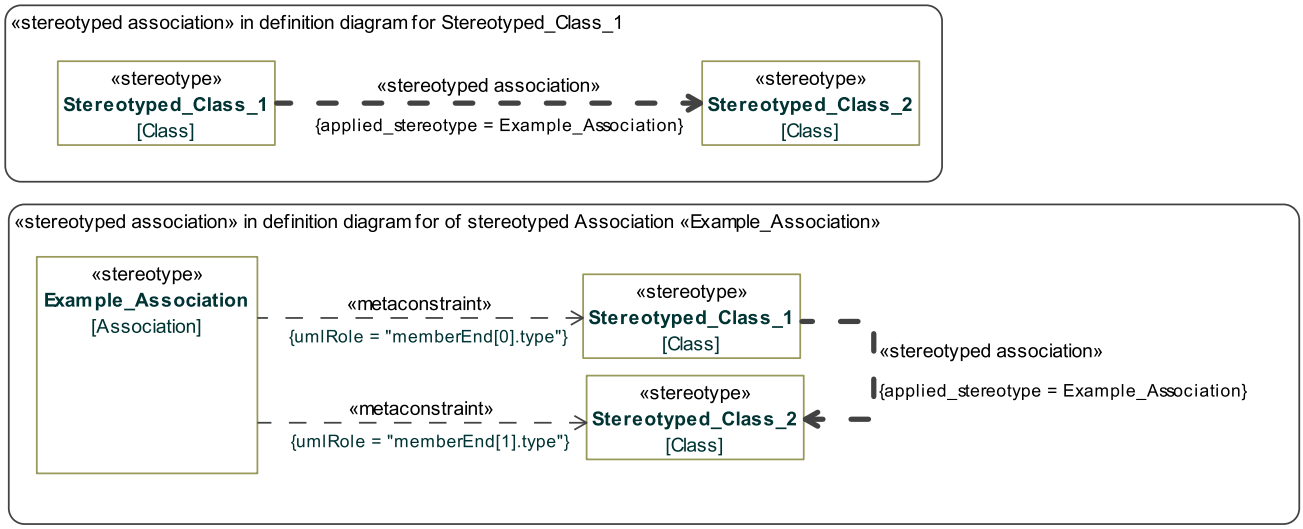

Figure 6-6 Use of «stereotyped association» dependency

6.2.2.5 Stereotyped Generalization Dependency Stereotype

The FACE Profile 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.

6.2.2.5.1 Definition of the Stereotyped Generalization Dependency

stereotyped generalization

Package: stereotyped dependencies

isAbstract: No

Extension: Dependency

Description

The stereotyped generalization stereotype has been created for the purpose of expressing the FACE Profile 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). To fully understand the generalization, the reader must examine the stereotyped generalization named in the tagged value.

Figure 6-7: stereotyped generalization

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.

6.2.2.5.2 Example Usage of the Stereotyped Generalization Dependency

The example diagram shows two different representations for a generalization stereotyped by «stereotyped generalization». The upper image shows how the «stereotyped generalization» would appear in the definition of «SemanticallySpecific» that has a «Constrained_Generalization» with «SemanticallyGeneral». This enables readers of this standard to see that a «Constrained_Generalization» could be defined between the two types of shown elements. The lower image shows how the «stereotyped generalization» would look in the diagram for the definition of «Constrained_Generalizatio». In this case the information may be redundant but provides an explicit visual showing that the «Constrained_Generalization» relationship is one way to connect the two classes. The Constraints applied to «Constrained_Generalization» further define the generalization relationship between «SemanticallySpecific» and «SemanticallyGeneral». The anchored note in the lower diagram shows an example of the sort of constraints that might be applied to the generalization relationship.

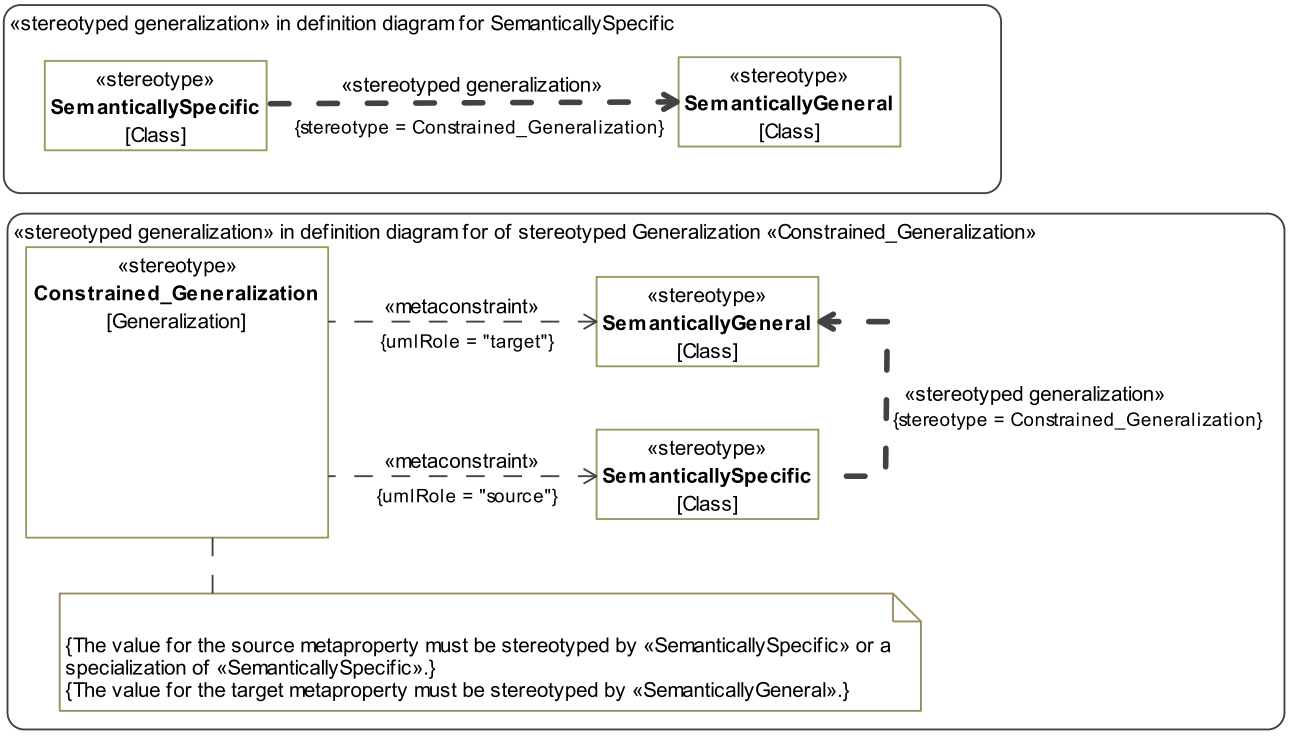

Figure 6-8 Use of «stereotyped generalization» dependency

7 FACE Profile

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

The FACE 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 and its referenced UDDL Standard.

7.1 FACE_Profile

The FACE_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 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 mimics the FACE metamodel packages.

FACE_ArchitectureModel

Package: FACE_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.

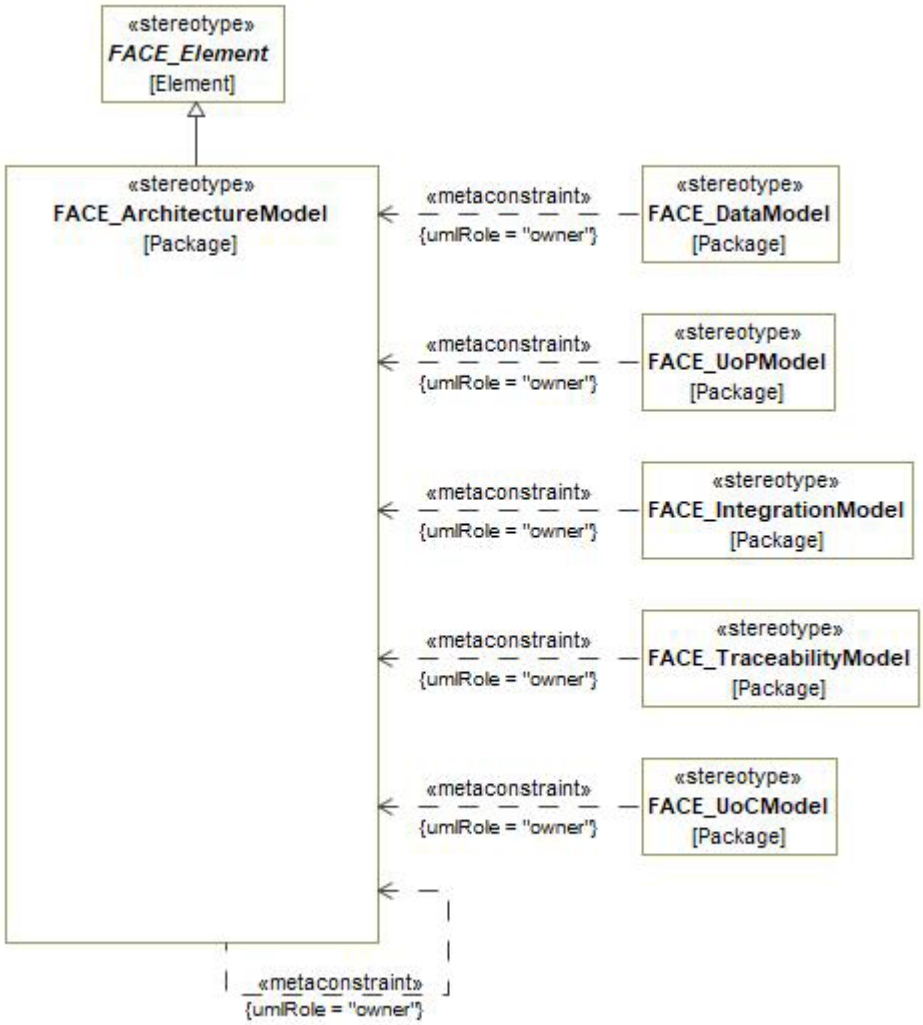

Figure 7-1: FACE_ArchitectureModel

Constraints

C01: 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

C01: FACE_ArchitectureModel.hasUniqueName	In the context of the entire FACE Architectural Model, the name of each element must be unique using case-insensitive tests.

FACE_Element

Package: FACE_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.

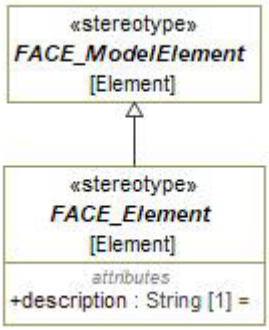

Figure 7-2: abstract FACE_Element

Attributes

description : String [1]

FACE Conformance/OCL Constraints

C01: FACE_Element.isValidIdentifier	An identifier is valid if it consists of alphanumeric characters.

7.1.1 FACE_Profile::FACE Data Architecture

The FACE Data Architecture package of the FACE Profile contains elements that represent the FACE Data Architecture as specified in the FACE metamodel. The profile packages within this package are organized to match the organization of the FACE metamodel.

FACE_AbstractAssociation

Package: FACE Data Architecture

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.

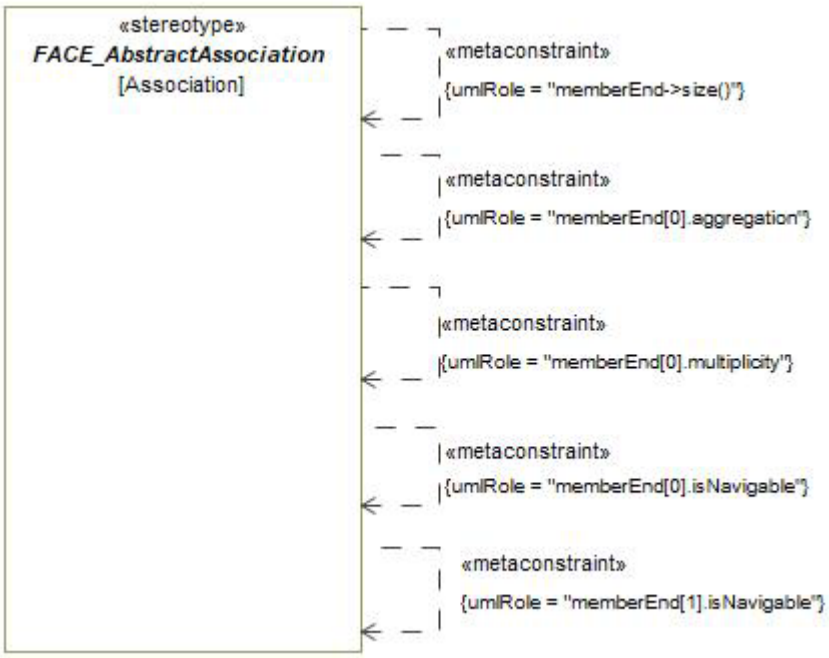

Figure 7-3: abstract FACE_AbstractAssociation

Constraints

C01: FACE_AbstractAssociation.memberEnd->size()	memberEnd.size() shall be 2
C02: FACE_AbstractAssociation.memberEnd[0].aggregation	memberEnd[0].aggregation shall be none
C03: FACE_AbstractAssociation.memberEnd[0].isNavigable	memberEnd[0].isNavigable shall be false
C04: FACE_AbstractAssociation.memberEnd[0].multiplicity	memberEnd[0].multiplicity shall be 1
C05: FACE_AbstractAssociation.memberEnd[1].isNavigable	memberEnd[1].isNavigable shall be true

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.

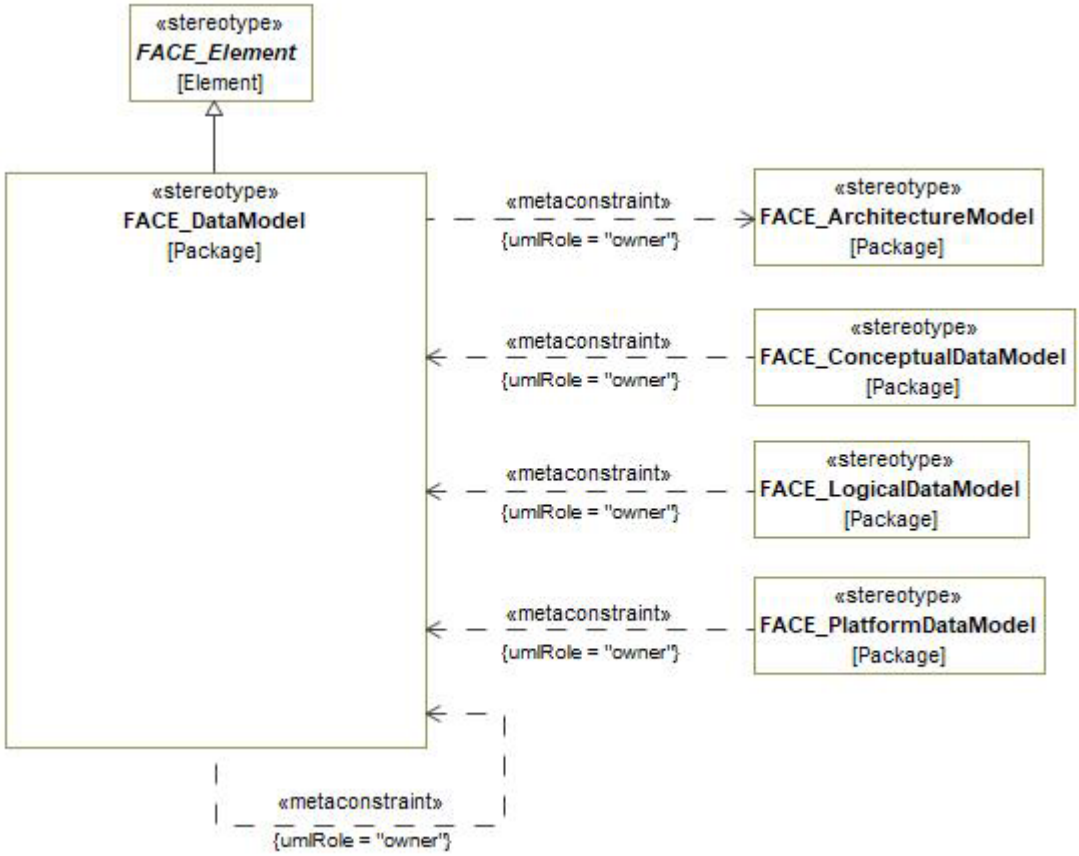

Figure 7-4: FACE_DataModel

Constraints

C01: FACE_DataModel.contains	The contained elements must by stereotyped one of the following: «FACE_DataModel» «FACE_ConceptualDataModel» «FACE_LogicalDataModel» «FACE_PlatformDataModel»
C02: 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

C01: FACE_DataModel.hasUniqueName	Each FACE Data Model Element must have a unique name as determined with case insensitivity.

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.

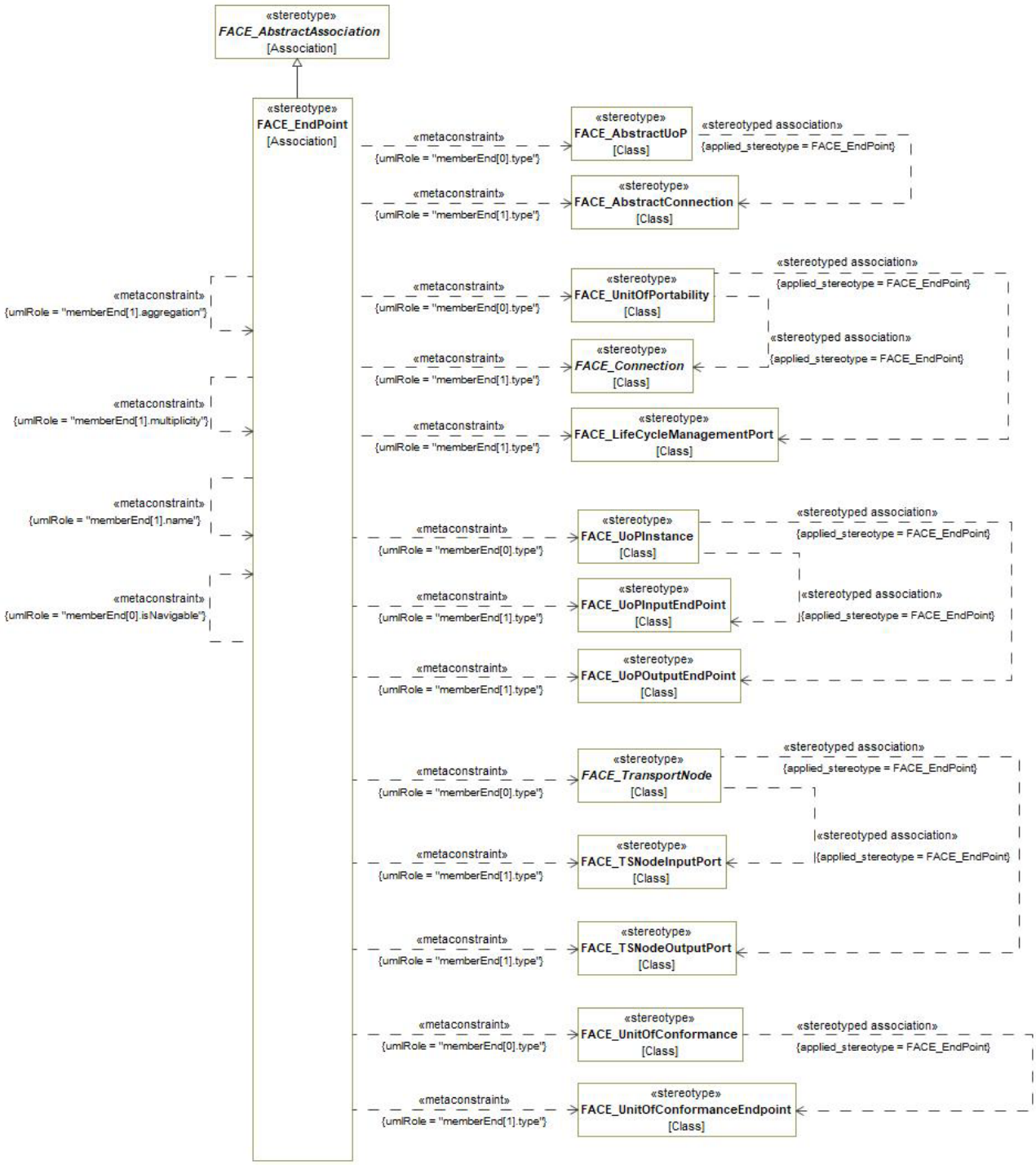

Figure 7-5: FACE_EndPoint

Constraints

C01: FACE_EndPoint.memberEnd [0].isNavigable	memberEnd[0].isNavigable shall be true

C02: 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»

C03: FACE_EndPoint.memberEnd[1].aggregation

memberEnd[1].aggregation shall be composite

C04: 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..*

C05: 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_UoPOutputEndPoint»	output
«FACE_UnitOfConformance»	«FACE_UnitOfConformanceEndpoint»	endPoint

C06: FACE_EndPoint.memberEnd[1].type

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.

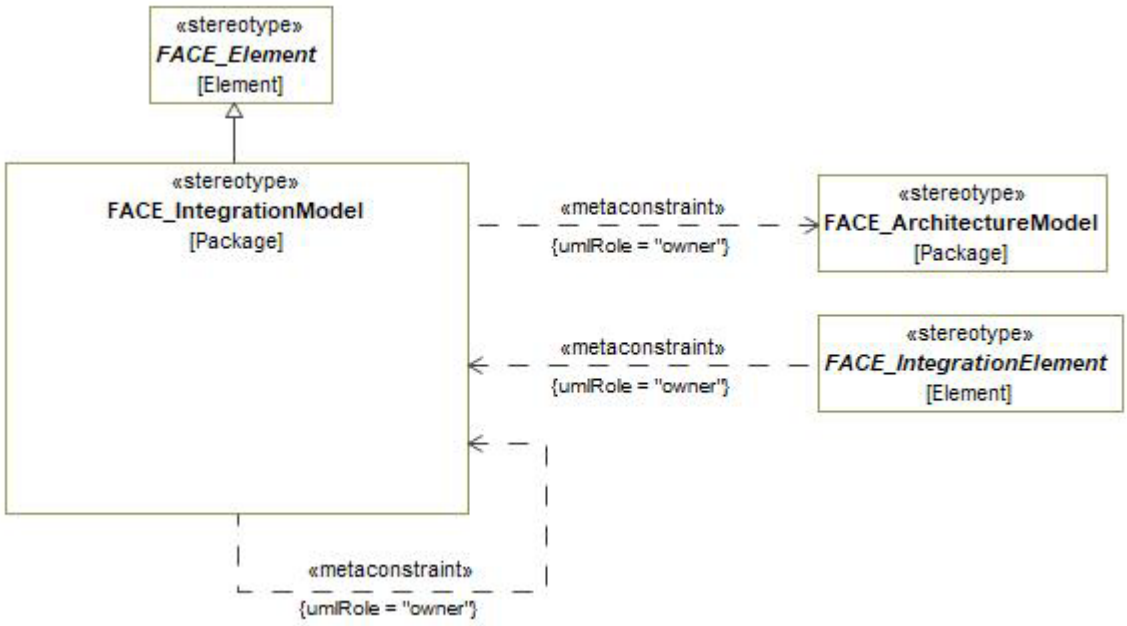

Figure 7-6: FACE_IntegrationModel

Constraints

C01: 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_UoPMessageType that specifies the data to be exchanged through a FACE_Endpoint.

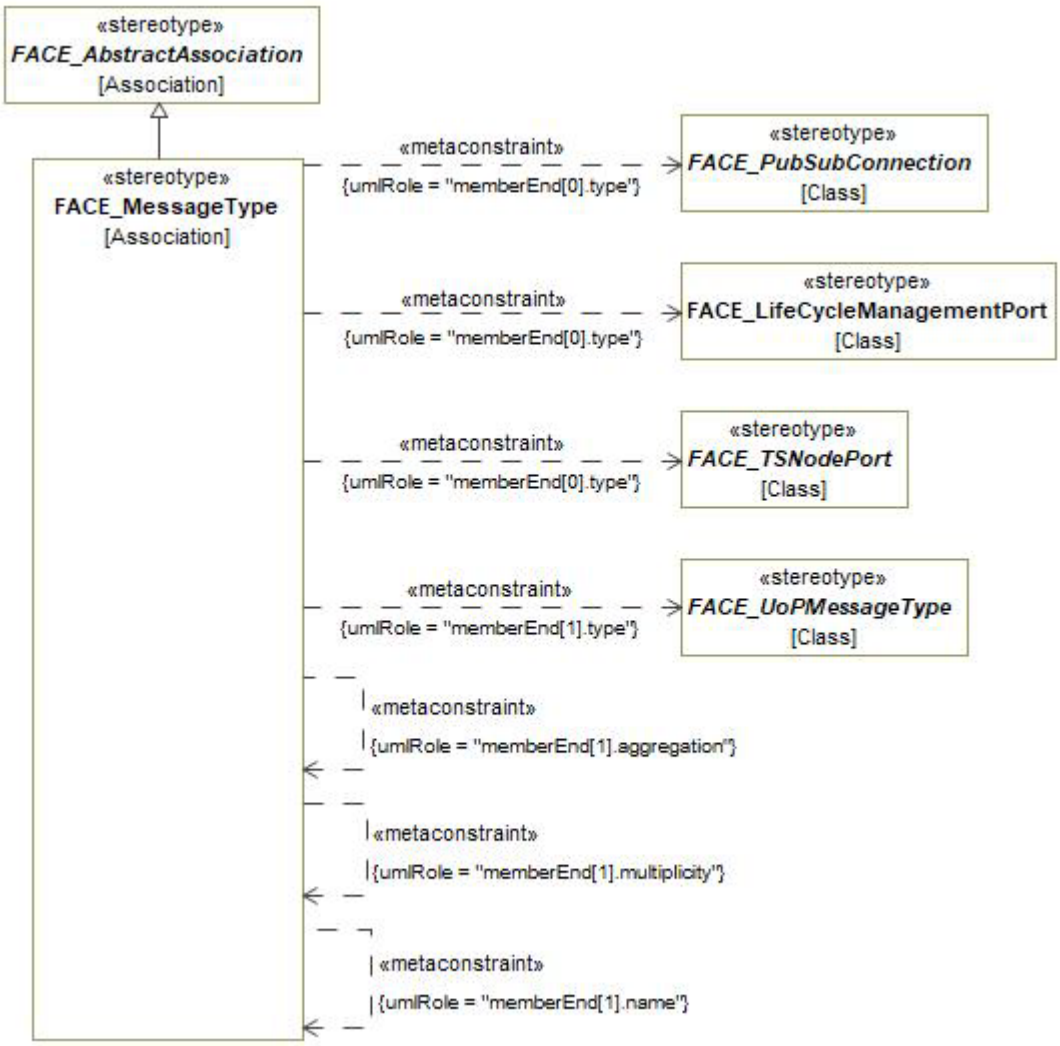

Figure 7-7: FACE_MessageType

Constraints

C01: 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»

C02: FACE_MessageType.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: FACE_MessageType.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 1
C04: 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"
C05: FACE_MessageType.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_UoPMessageType».

FACE_ModelElement

Package: FACE Data Architecture

isAbstract: Yes

Extension: Element

Description

An abstract stereotype created specifically for the FACE Profile. Used to represent the unique identity of constructed UDDL and 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 UDDL or FACE metamodel.

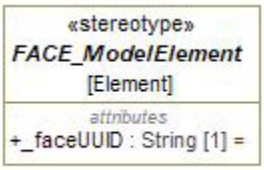

Figure 7-8: abstract FACE_ModelElement

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_Realize

Package: FACE Data Architecture

isAbstract: No

Generalization: FACE_AbstractAssociation

Extension: Association

Description

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

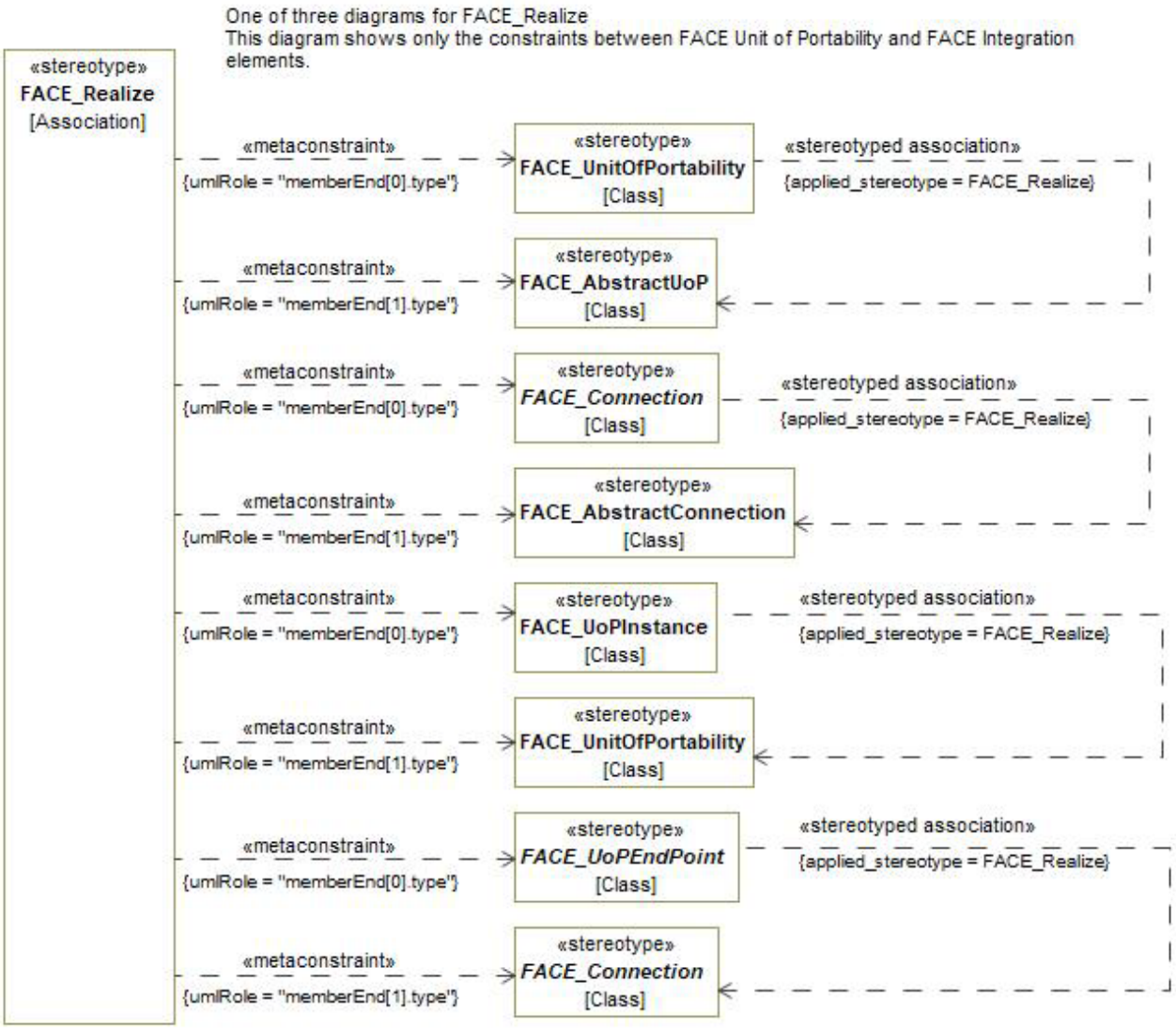

Figure 7-9: FACE_Realize

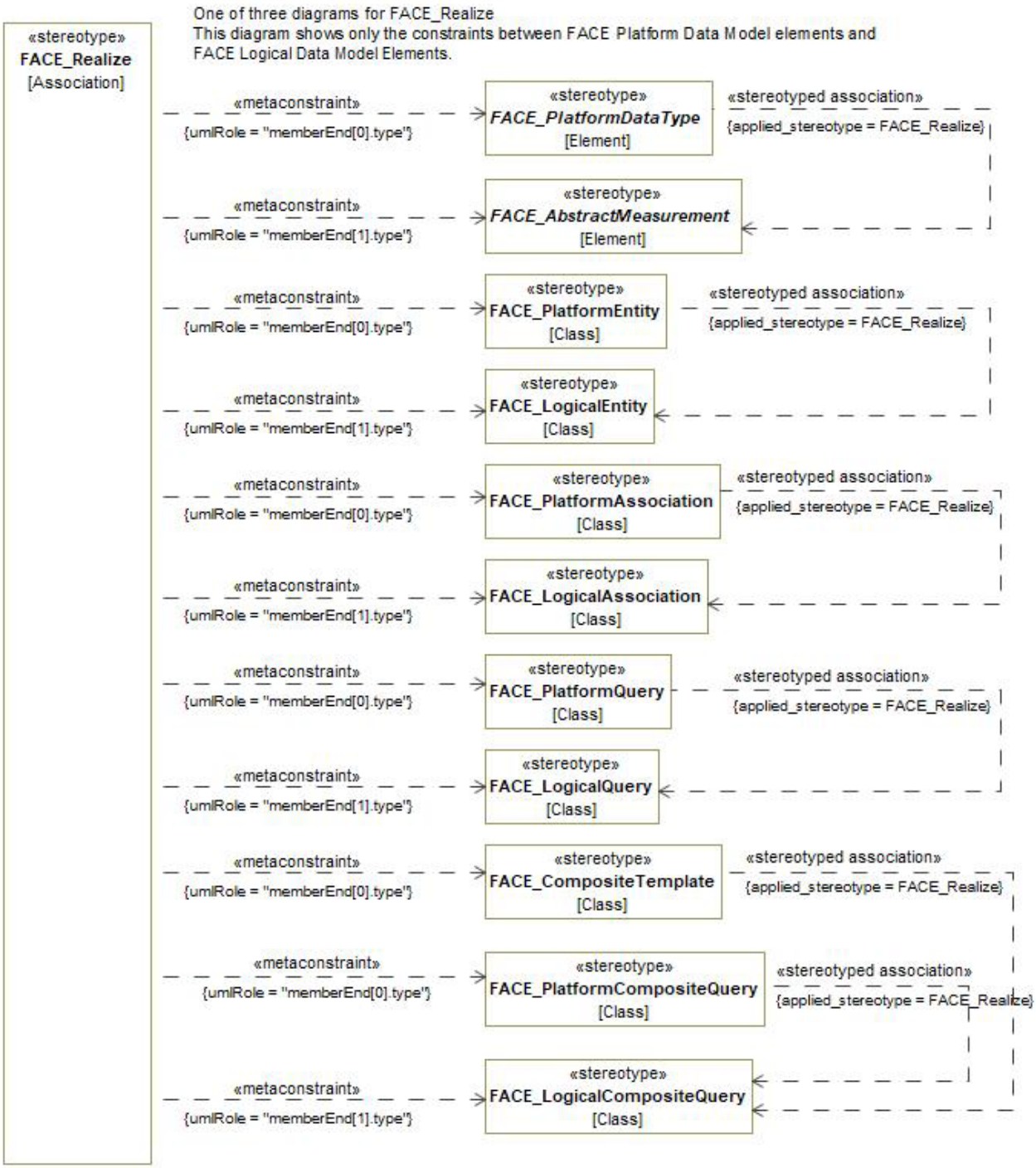

Figure 7-10: FACE_Realize

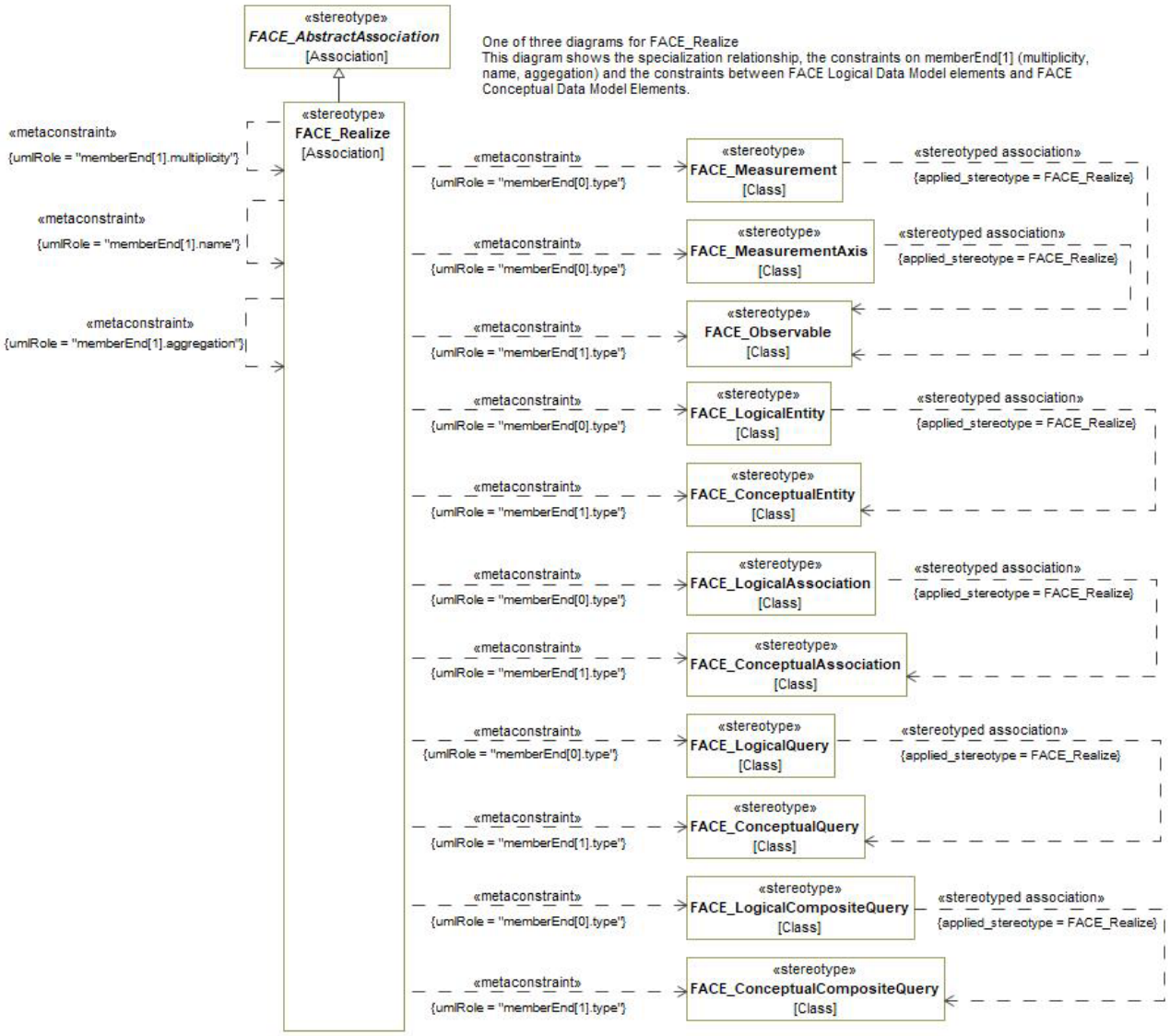

Figure 7-11: FACE_Realize

Constraints

C01: 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_PlatformDataType» «FACE_PlatformAssociation» «FACE_PlatformEntity» «FACE_PlatformQuery» «FACE_PlatformCompositeQuery» «FACE_CompositeTemplate» «FACE_UnitOfPortability» Specializations of «FACE_Connection» «FACE_UoPInstance» Specializations of «FACE_UoPEndPoint»
C02: FACE_Realize.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none

C03: FACE_Realize.memberEnd[1].multiplicity	Based on the stereotype of the memberEnd[0].type metaproperty: = «FACE_CompositeTemplate», «FACE_PlatformQuery», «FACE_PlatformCompositeQuery», «FACE_UnitOfPortability», or specialization of «FACE_Connection», memberEnd[1].multiplicity is 0..1 = specialization of «FACE_PlatformDataType», «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
C04: FACE_Realize.memberEnd[1].name	memberEnd[1].name shall be "realize"

C05: 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_PlatformDataType», 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_PlatformCompositeQuery», the memberEnd[1].type metaproperty must be stereotyped by «FACE_LogicalCompositeQuery» = «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.

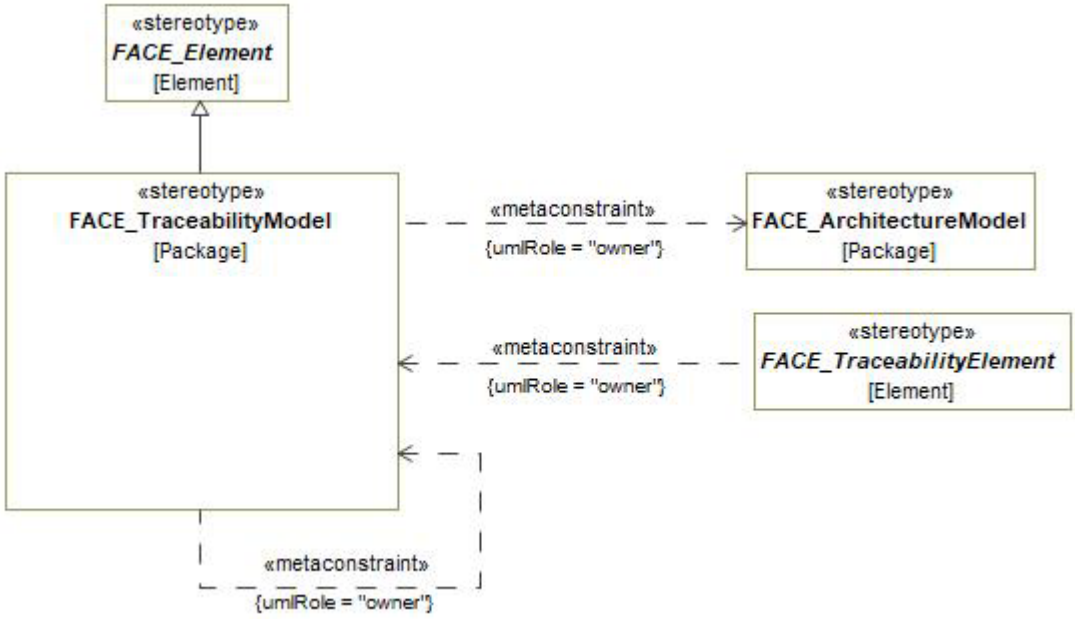

Figure 7-12: FACE_TraceabilityModel

Constraints

C01: 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.

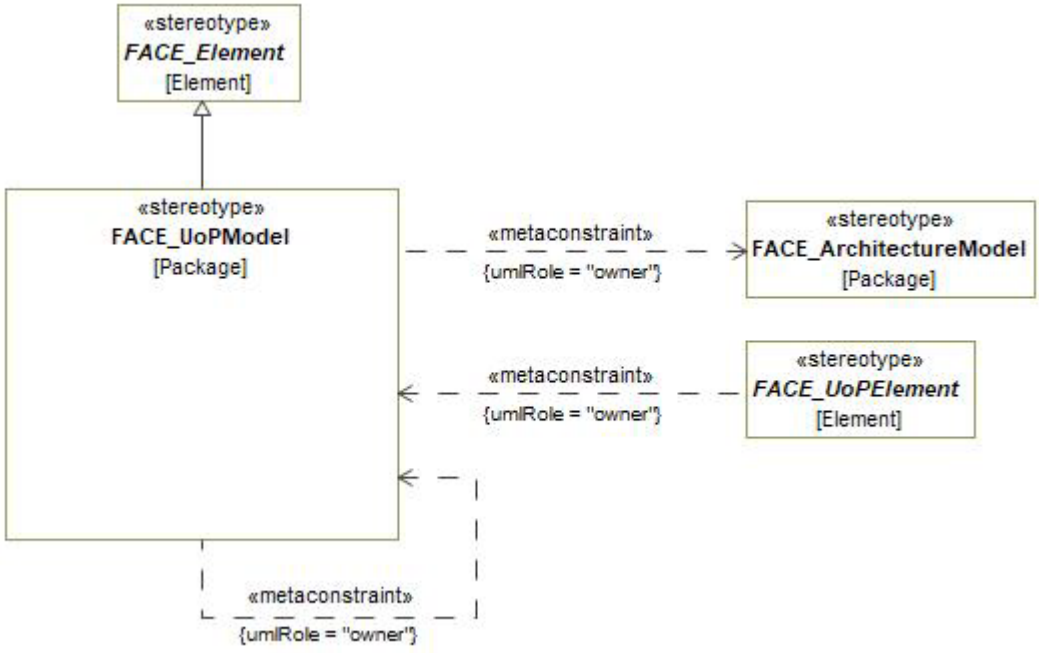

Figure 7-13: FACE_UoPModel

Constraints

C01: 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_Profile::FACE Data Architecture::FACE Data Model

The FACE Data Model package of the FACE Profile contains elements that represent the FACE Data Model package as specified in the FACE metamodel. The FACE metamodel references the UDDL specification for its content. The subpackages in this package are organized to match the FACE and UDDL metamodel organization.

FACE_ConceptualDataModel

Package: FACE Data Model

isAbstract: No

Generalization: FACE_DataModelElement

Extension: Package

Description

A FACE_ConceptualDataModel is a container for FACE_ConceptualElements.

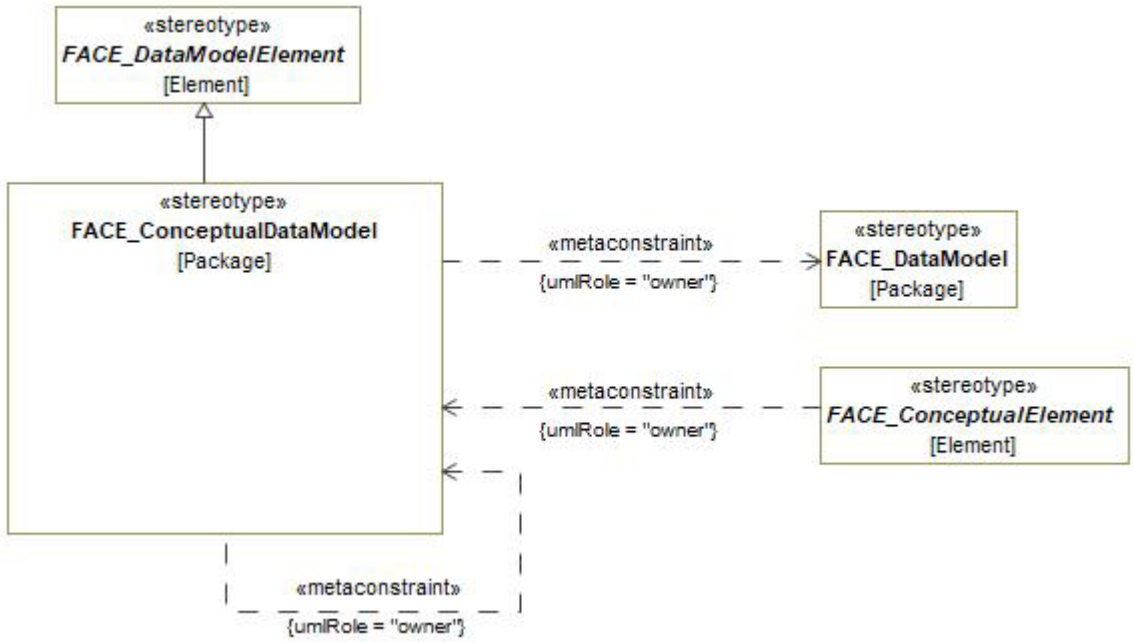

Figure 7-14: FACE_ConceptualDataModel

Constraints

C01: FACE_ConceptualDataModel.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_DataModel» «FACE_ConceptualDataModel»

FACE_DataModelElement

Package: FACE Data Model

isAbstract: Yes

Generalization: FACE_ModelElement

Description

A FACE_DatamodelElement is the root type for defining the elements of the Data Model Language. The “name” attribute in the UML metatype captures the name of the Data Model Element in the model. The “description” attribute captures a description for the Data Model Element.

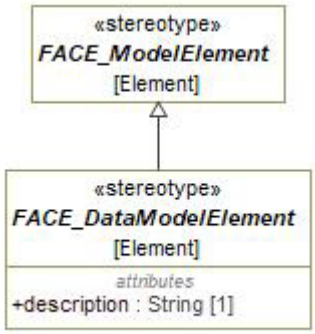

Figure 7-15: abstract FACE_DataModelElement

Attributes

description : String [1]

FACE Conformance/OCL Constraints

C01: FACE_DataModelElement.isValidIdentifier	An identifier is valid if it consists of alphanumeric characters.

C02: FACE_DataModelElement.nonEmptyDescription	The following data model elements must have a non-empty description: - Observable - Unit - Landmark - ReferencePoint - MeasurementSystem - MeasurementSystemAxis - CoordinateSystem - CoordinateSystemAxis - MeasurementSystemConversion - LogicalValueTypeUnit.value_type == Boolean - LogicalValueTypeUnit.value_type == Character - LogicalValueTypeUnit.value_type == Numeric - LogicalValueTypeUnit.value_type == Integer - LogicalValueTypeUnit.value_type == Natural - LogicalValueTypeUnit.value_type == NonNegativeReal - LogicalValueTypeUnit.value_type == Real - LogicalValueTypeUnit.value_type == String

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).

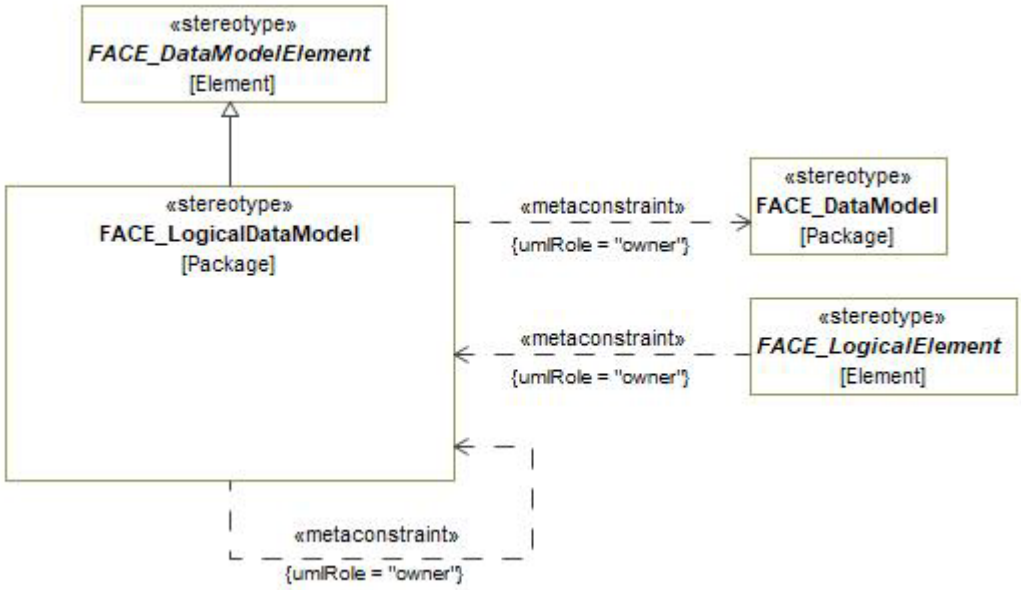

Figure 7-16: FACE_LogicalDataModel

Constraints

C01: 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).

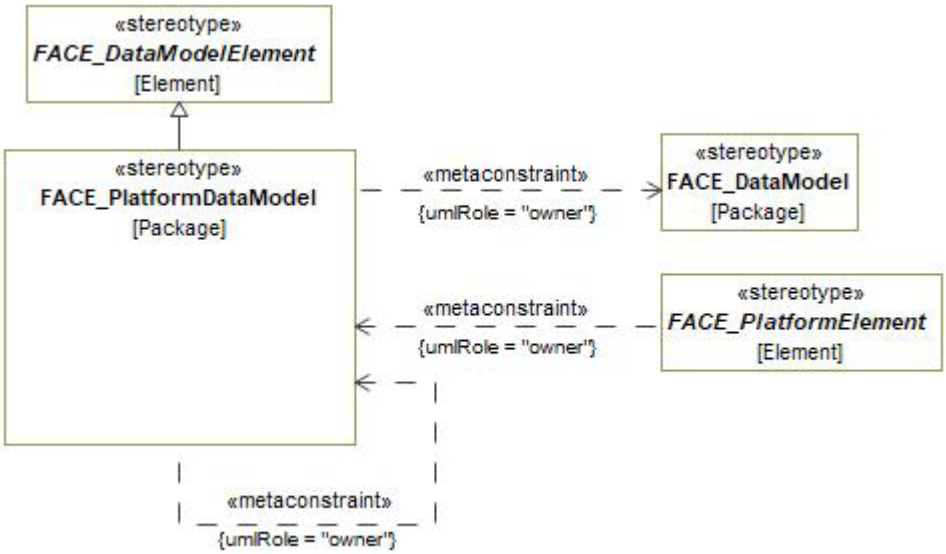

Figure 7-17: FACE_PlatformDataModel

Constraints

C01: 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.

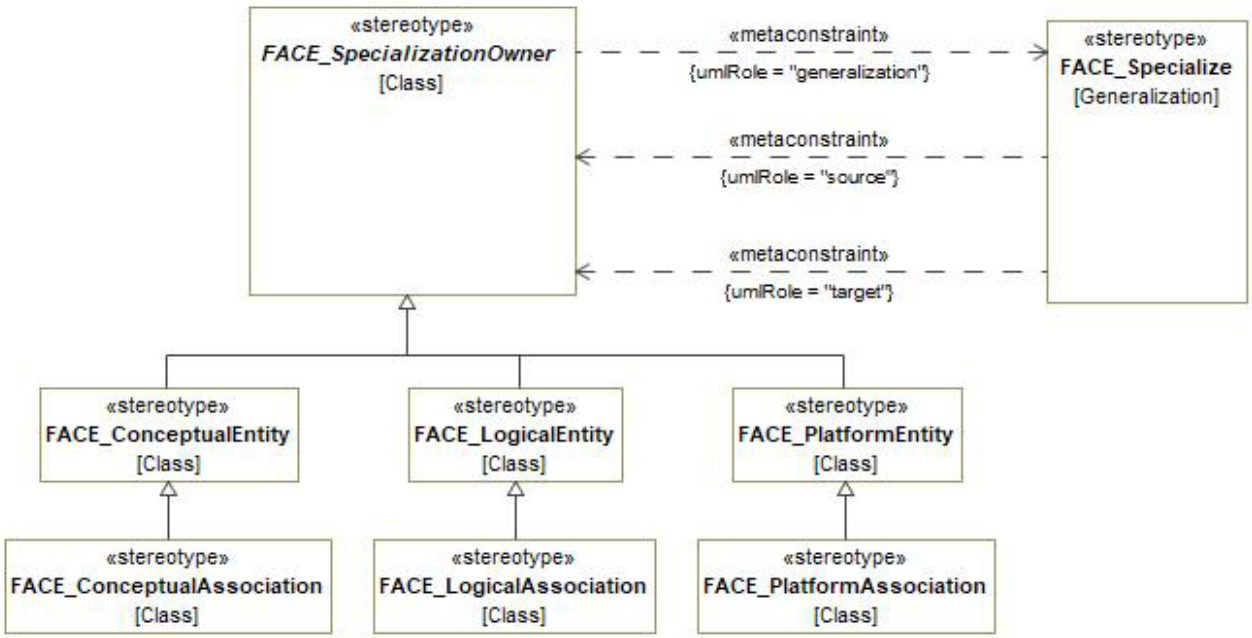

Figure 7-18: abstract FACE_SpecializationOwner

Constraints

C01: 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.

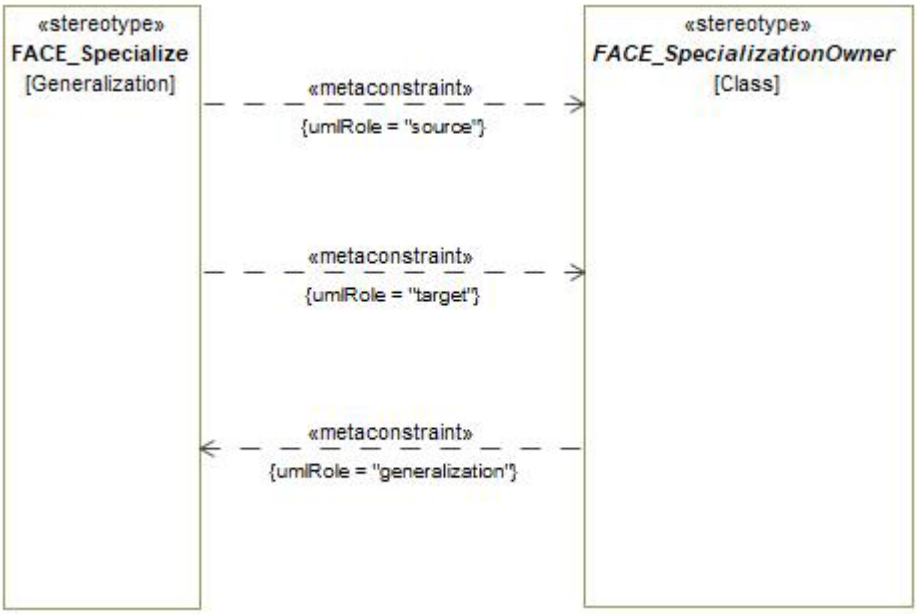

Figure 7-19: FACE_Specialize

Constraints

C01: FACE_Specialize.source	The value for the source metaproperty must be stereotyped by a specialization of «FACE_SpecializationOwner».

C02: 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_Profile::FACE Data Architecture::FACE Data Model::ConceptualDataModel

The ConceptualDataModel package of the FACE Profile contains elements that represent the Conceptual Data Model subpackage as specified in the UDDL metamodel.

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.

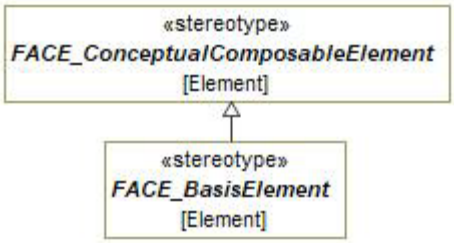

Figure 7-20: abstract FACE_BasisElement

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.

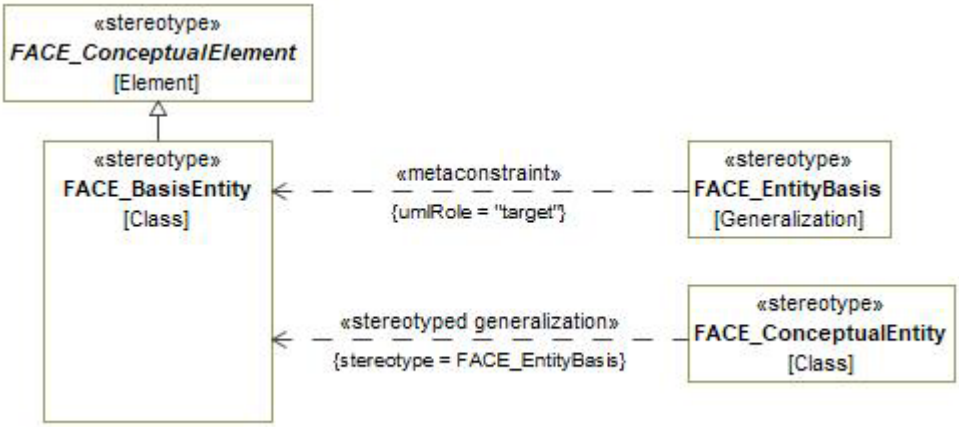

Figure 7-21: FACE_BasisEntity

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.

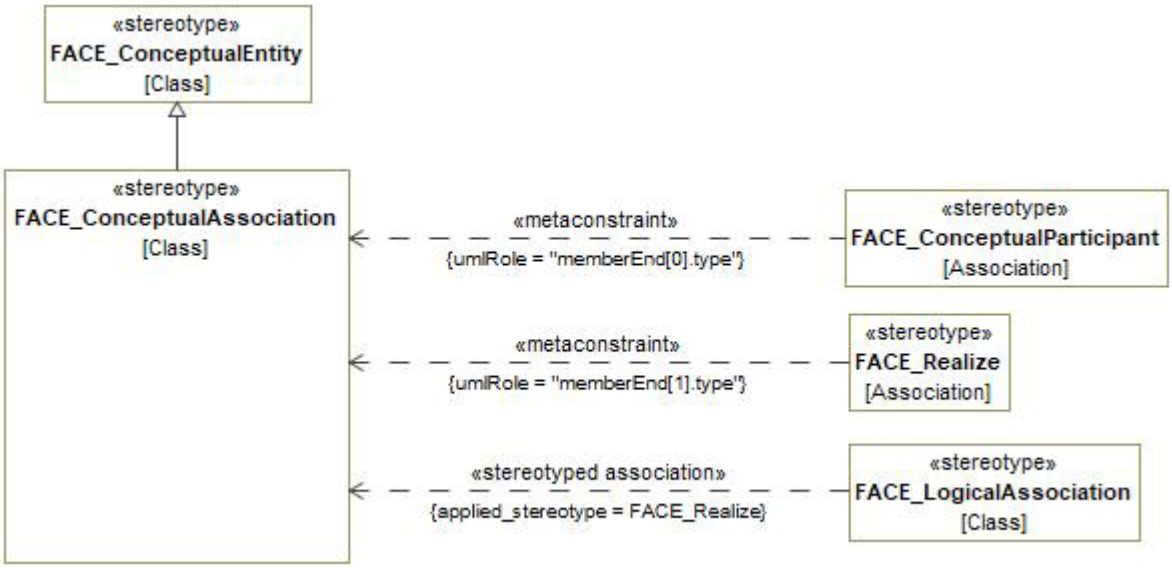

Figure 7-22: FACE_ConceptualAssociation

FACE Conformance/OCL Constraints

C01: FACE_ConceptualAssociation.hasAtLeastTwoParticipants	A FACE_ConceptualAssociation 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 "name" attribute corresponds to the UDDL Standard's "rolename" attribute and 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.

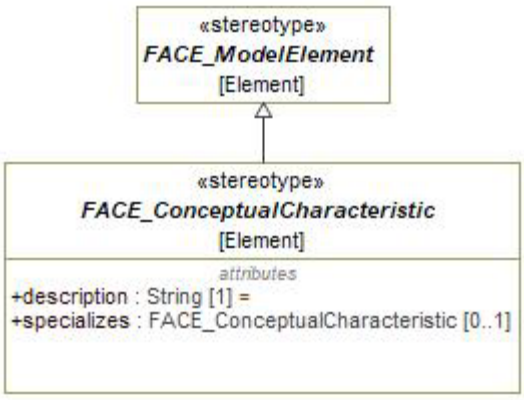

Figure 7-23: abstract FACE_ConceptualCharacteristic

Attributes

description : String [1]

specializes : FACE_ConceptualCharacteristic [0..1]

FACE Conformance/OCL Constraints

C01: FACE_ConceptualCharacteristic.lowerBoundValid	A FACE_ConceptualCharacteristic's lowerBound must be greater than or equal to zero.
C02: FACE_ConceptualCharacteristic.lowerBound_LTE_UpperBound	A FACE_ConceptualCharacteristic's lowerBound must be less than or equal to its upperBound, unless its upperBound is -1.
C03: FACE_ConceptualCharacteristic.rolenameIsValidIdentifier	The rolename of a FACE_ConceptualCharacteristic must be a valid identifier.
C04: FACE_ConceptualCharacteristic.specializeCharacteristicOnce	A FACE_ConceptualCharacteristic must be specialized no more than once in a generalization hierarchy.
C05: 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.

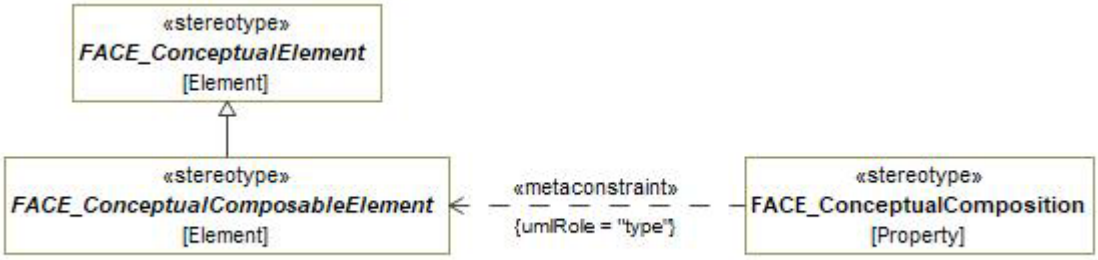

Figure 7-24: abstract FACE_ConceptualComposableElement

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 an union or as members of a struct.

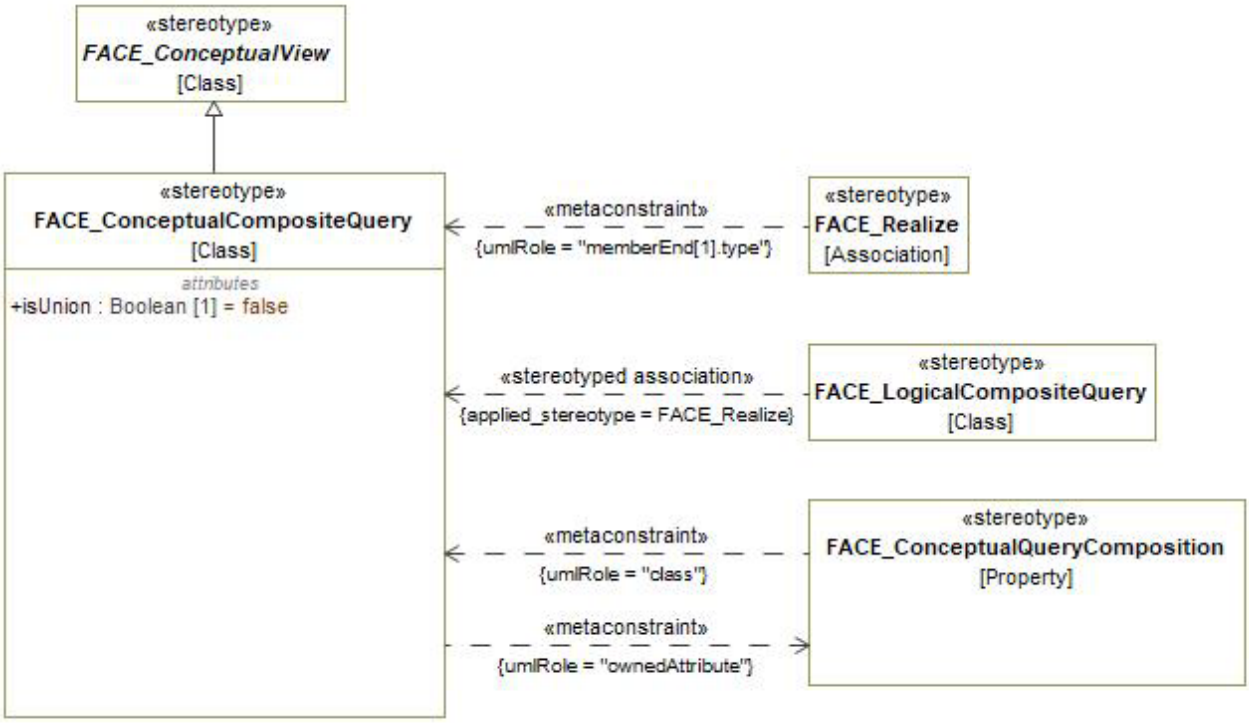

Figure 7-25: FACE_ConceptualCompositeQuery

Attributes

isUnion : Boolean [1]

Constraints

C01: 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

C01: FACE_ConceptualCompositeQuery.compositionsHaveUniqueRolenames	A FACE_ConceptualQueryComposition's rolename must be unique within a FACE_ConceptualCompositeQuery.

C02: FACE_ConceptualCompositeQuery.noCyclesInConstruction	A FACE_ConceptualCompositeQuery may not compose itself.
C03: 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.

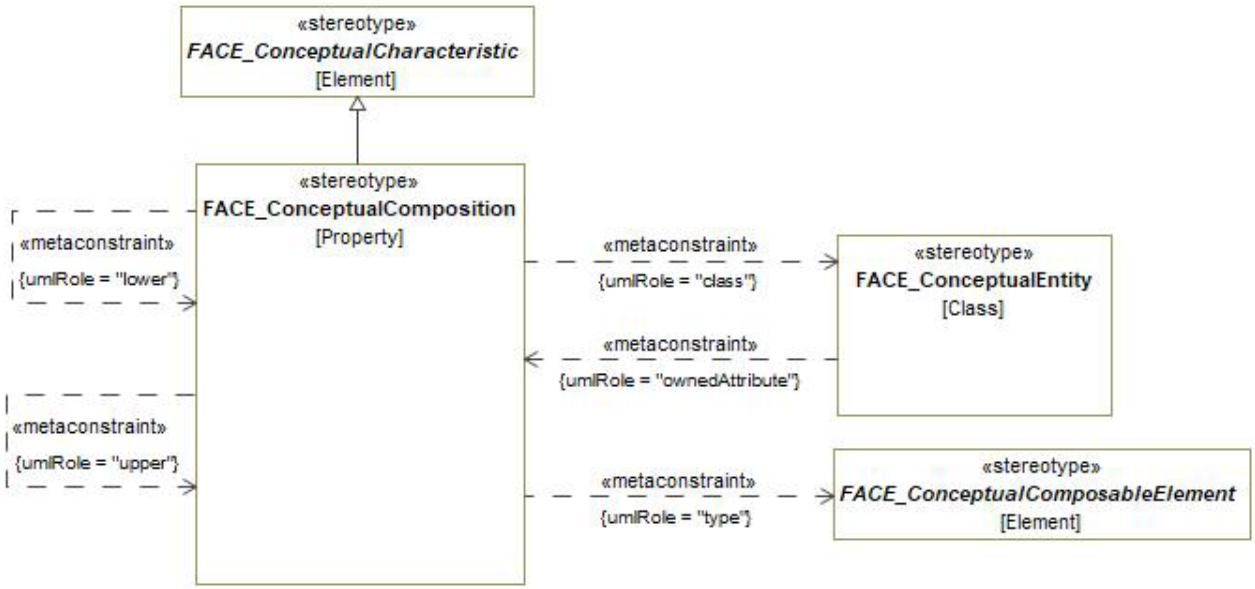

Figure 7-26: FACE_ConceptualComposition

Constraints

C01: FACE_ConceptualComposition.class	Value for the class metaproperty must be stereotyped «FACE_ConceptualEntity» or its specializations.

C02: FACE_ConceptualComposition.lower	The value for the lower (lower bound of multiplicity) metaproperty must be an integer greater than or equal to -1.
C03: FACE_ConceptualComposition.type	Value for the type metaproperty must be stereotyped «FACE_ConceptualComposableElement» or its specializations.
C04: 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

C01: FACE_ConceptualComposition.multiplicityConsistentWithSpecialization	If a FACE_ConceptualComposition specializes, its multiplicity must be at least as restrictive as the FACE_ConceptualComposition it specializes.
C02: FACE_ConceptualComposition.specializationDistinct	If a FACE_ConceptualComposition specializes, its type or multiplicity must be different from the FACE_ConceptualComposition it specializes.
C03: 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 Data Model Language.

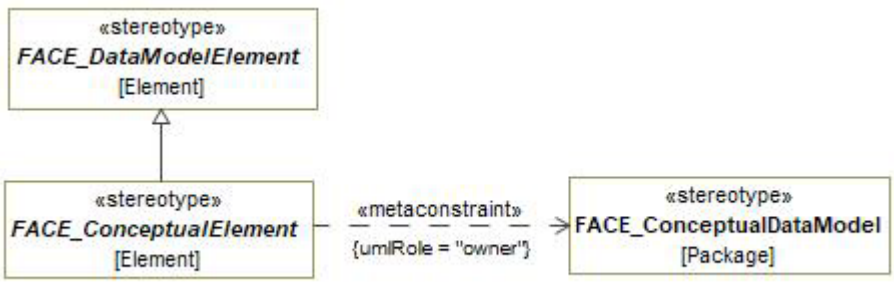

Figure 7-27: abstract FACE_ConceptualElement

Constraints

C01: 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

C01: FACE_ConceptualElement.hasUniqueName	Each FACE ConceptualElement must have a unique name, as determined using case insensitivity.

FACE_ConceptualEntity

Package: ConceptualDataModel

isAbstract: No

Generalization: FACE_ConceptualComposableElement, 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.

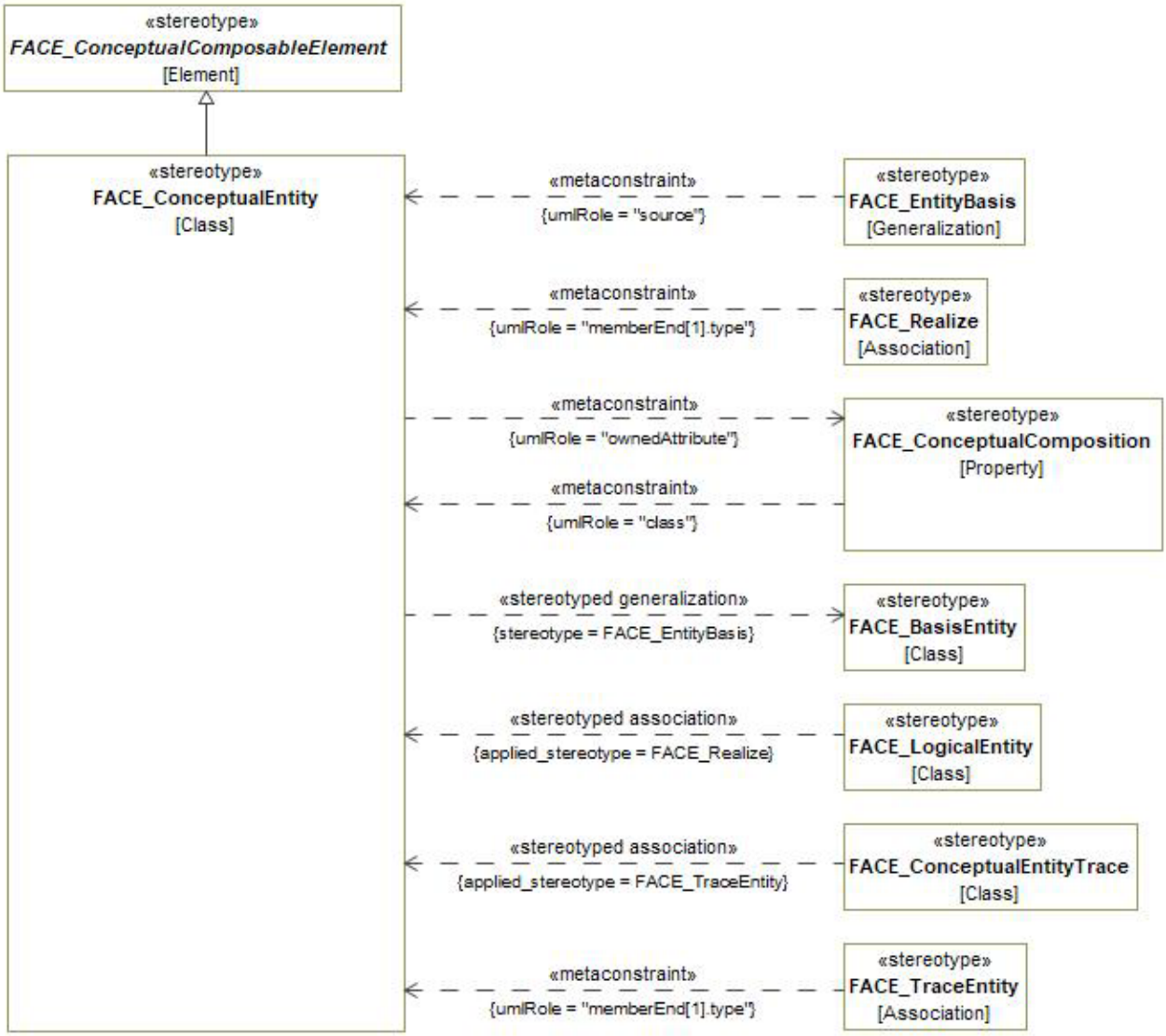

Figure 7-28: FACE_ConceptualEntity

Constraints

C01: FACE_ConceptualEntity.ownedAttribute	The value for the ownedAttribute metaproperty must be stereotyped «FACE_ConceptualComposition» or its specializations

FACE Conformance/OCL Constraints

C01: FACE_ConceptualEntity.characteristicsHaveUniqueRolenames	A Characteristic's rolename must be unique within a FACE_ConceptualEntity.

C02: 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.
C03: FACE_ConceptualEntity.hasAtLeastOneLocalCharacteristic	A FACE_ConceptualEntity must have at least one Characteristic defined locally (not through generalization).
C04: FACE_ConceptualEntity.hasUniqueID	A FACE_ConceptualEntity must contain a Composition whose type is an Observable named 'Identifier'.
C05: FACE_ConceptualEntity.noCyclesInSpecialization	A FACE_ConceptualEntity must not be a specialization of itself, directly or indirectly.
C06: FACE_ConceptualEntity.observableComposedOnce	A FACE_ConceptualEntity may not compose the same FACE_Observable more than once.
C07: 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: Association

Description

A FACE_ConceptualParticipant is the mechanism that allows a FACE_ConceptualAssociation to be constructed between two or more FACE_ConceptualEntities. The "type" (target of the directional Association) of a conceptual Participant is the conceptual Entity being used to construct the conceptual Association. Target multiplicity values represent the "sourceLowerBound" and "sourceUpperBound" attributes that define the multiplicity of the conceptual Association relative to the Participant in the UDDL metamodel. An upper multiplicity of star (*) on the target of the association is the equivalent of a "sourceUpperBound" multiplicity of -1 (which represents an unbounded sequence) in the the UDDL metamodel. 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.

FACE_ConceptualParticipant Associations are directional, from a FACE_ConceptualAssociation to a FACE_ConceptualEntity.

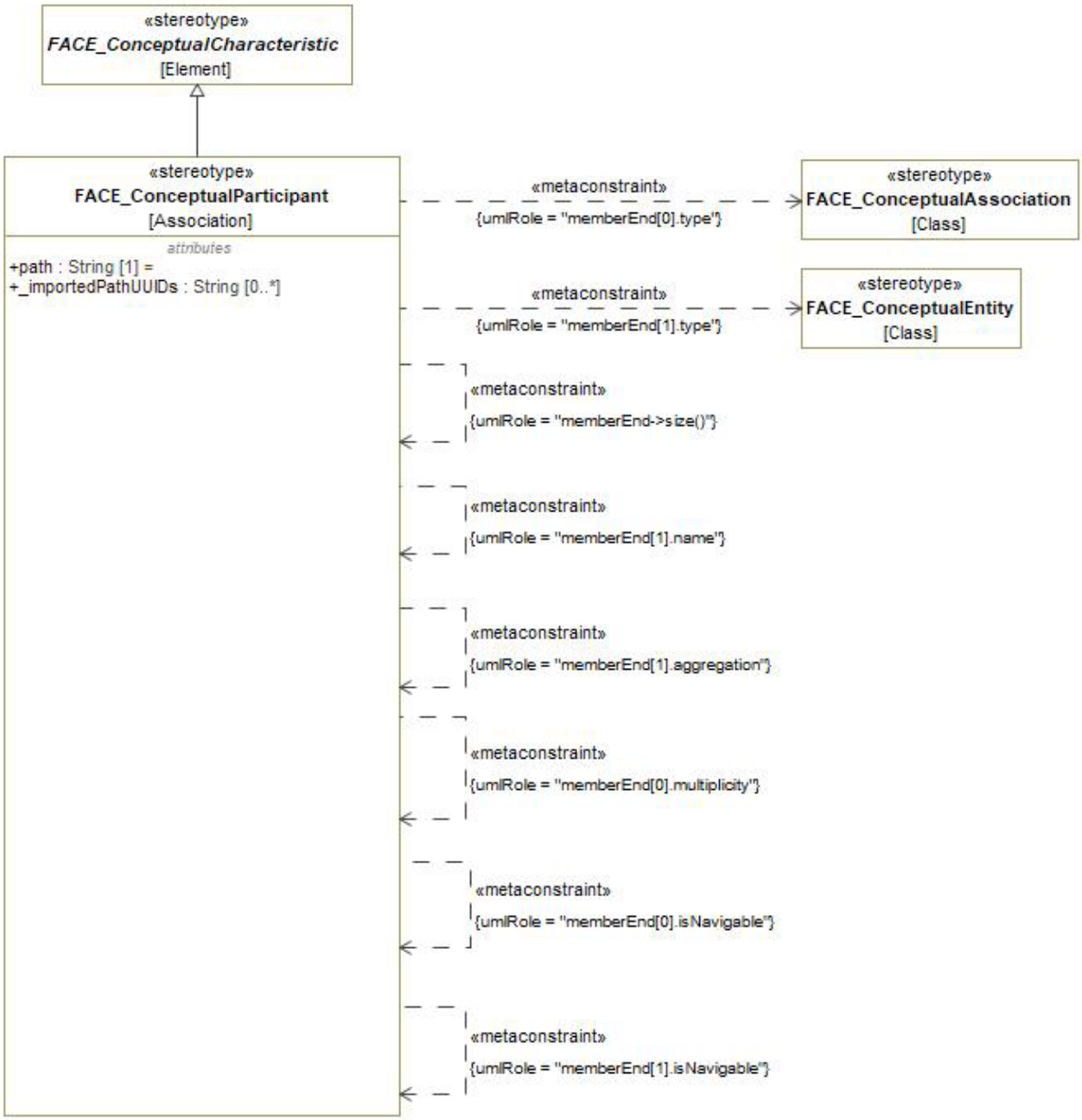

Figure 7-29: FACE_ConceptualParticipant

Attributes

path : String [1]	The "path" property indicates the portion of the target «FACE_ConceptualEntity» that is participating in the

	«FACE_ConceptualAssociation» that is the source for the «FACE_ConceptualParticipant» Association. Path strings reference Entities or Characteristics (properties of Entities). Where the path string references an Entity, it is considered to be a ParticipantPathNode. Where the path string references a Characteristic of an Entity, it is considered to be a CharacteristicPathNode. The UDDL metamodel defines PathNode, ParticipantPathNode and CharacteristicPathNode as follows: A conceptual PathNode is a single element in a chain that collectively forms a path specification. A conceptual ParticipantPathNode is a conceptual PathNode that selects a Participant that references an Entity. This provides a mechanism for reverse navigation from an Entity that participates in an Association back to the Association. A conceptual CharacteristicPathNode is a conceptual PathNode that selects a conceptual Characteristic which is directly contained in a conceptual Entity or Association. 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 UDDL Standard. 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 and the FACE XMI (face) file are required to translate between the two notations.
_importedPathUUIDs : String [0..*]	This tag is for use by import/export plug-ins in two-way translation of FACE 3.x paths to and from FACE 2.1 path strings. It is used to preserve the UUIDs of the paths imported from FACE 3.x paths when they are translated into FACE 2.1 path strings, so that they can be reconstituted for subsequent export as FACE 3.x elements. Because this tag is used exclusively by the plug-ins, its implementation is optional if a tool either does not import/export FACE format files or the tool uses an alternate means of representing and translating FACE Paths.

Constraints

C01: FACE_ConceptualParticipant.memberEnd->size()	memberEnd.size() shall be 2
C02: FACE_ConceptualParticipant.memberEnd[0].isNavigable	memberEnd[0].isNavigable shall be false

C03: FACE_ConceptualParticipant.memberEnd[0].multiplicity	memberEnd[0].multiplicity shall be 1
C04: FACE_ConceptualParticipant.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_ConceptualAssociation»
C05: FACE_ConceptualParticipant.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C06: FACE_ConceptualParticipant.memberEnd[1].isNavigable	memberEnd[1].isNavigable shall be true
C07: FACE_ConceptualParticipant.memberEnd[1].name	The memberEnd[1].name metaproperty must be an non-empty alphanumeric name string
C08: FACE_ConceptualParticipant.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_ConceptualEntity»

FACE Conformance/OCL Constraints

C01: FACE_ConceptualParticipant.multiplicityConsistentWithSpecialization	If a FACE_ConceptualParticipant specializes, its multiplicity must be at least as restrictive as the FACE_ConceptualParticipant it specializes.
C02: 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.
C03: FACE_ConceptualParticipant.rolenameDefined	A FACE_ConceptualParticipant must have a rolename, either projected from a characteristic or defined directly on the FACE_ConceptualParticipant.
C04: FACE_ConceptualParticipant.specializationDistinct	If a FACE_ConceptualParticipant specializes, its type, PathNode sequence, or multiplicity must be different from the FACE_ConceptualParticipant it specializes.
C05: 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 Section 6.1.

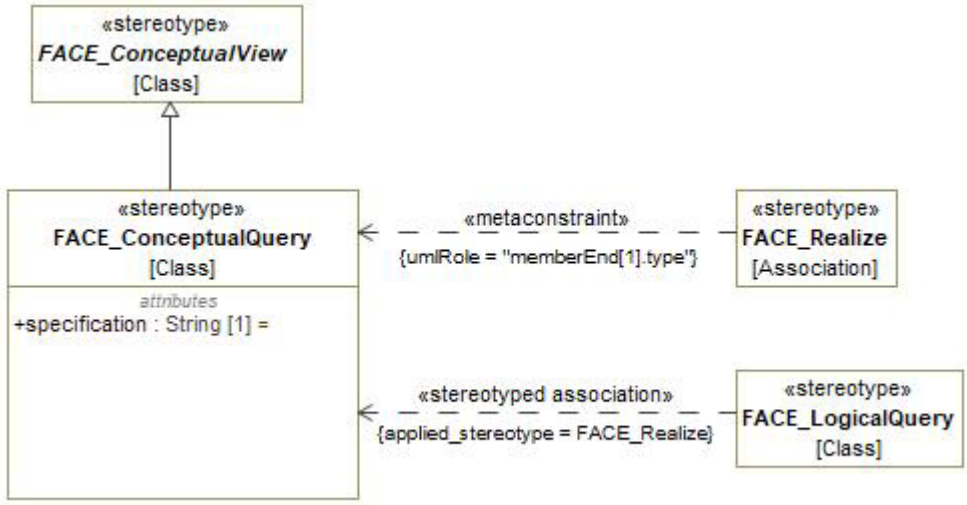

Figure 7-30: FACE_ConceptualQuery

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 metatype "name" attribute represents the UDDL "rolename" attribute that defines the name of the composed conceptual View within the scope of the composing conceptual CompositeQuery. The "type" of a conceptual QueryComposition is the conceptual View being used to construct the conceptual CompositeQuery.

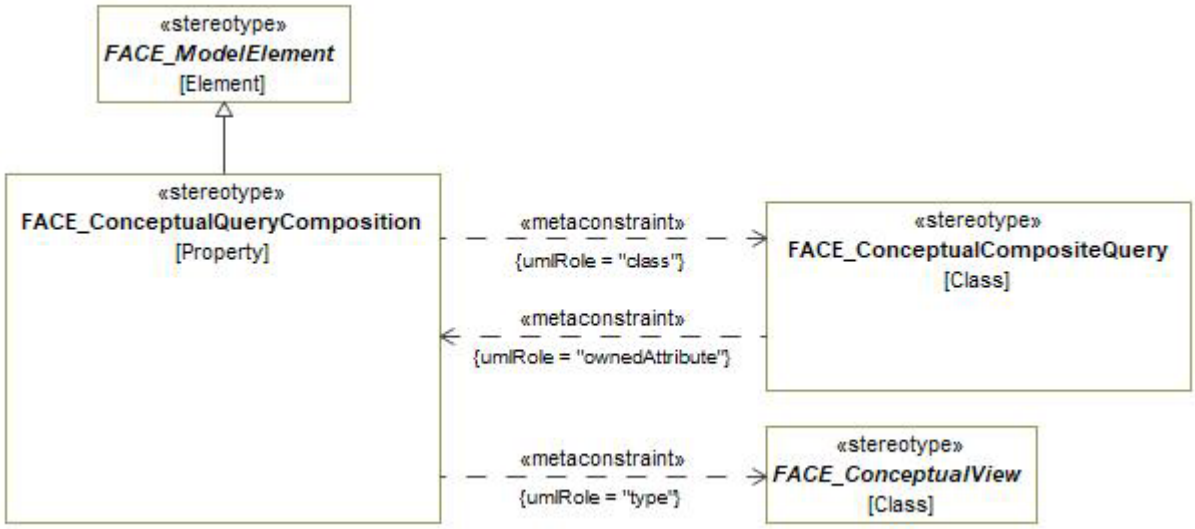

Figure 7-31: FACE_ConceptualQueryComposition

Constraints

C01: FACE_ConceptualQueryComposition.class	Value for the class metaproperty must be stereotyped «FACE_ConceptualCompositeQuery».
C02: FACE_ConceptualQueryComposition.type	Value for the type metaproperty must be stereotyped «FACE_ConceptualView» or its specializations.

FACE Conformance/OCL Constraints

C01: FACE_ConceptualQueryComposition.rolenameIsValidIdentifier	The rolename of a FACE_ConceptualQueryComposition must be a valid identifier.

FACE_ConceptualView

Package: ConceptualDataModel

isAbstract: Yes

Generalization: FACE_ConceptualElement

Extension: Class

Description

A FACE_ConceptualView is a FACE_ConceptualQuery or a FACE_ConceptualCompositeQuery.

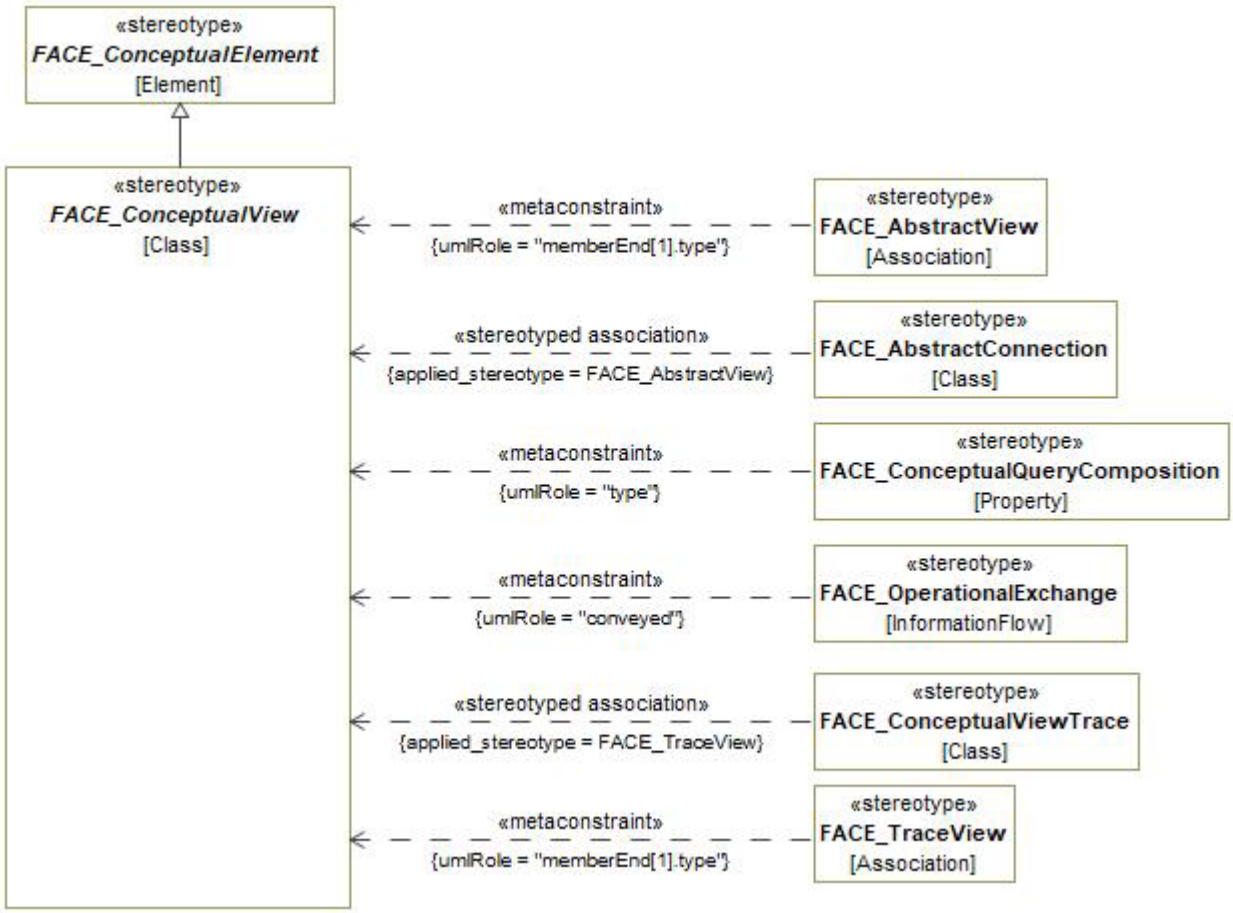

Figure 7-32: abstract FACE_ConceptualView

FACE_Domain

Package: ConceptualDataModel

isAbstract: No

Generalization: FACE_ConceptualElement

Extension: Class

Description

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

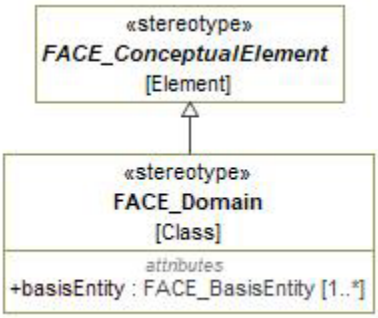

Figure 7-33: FACE_Domain

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.

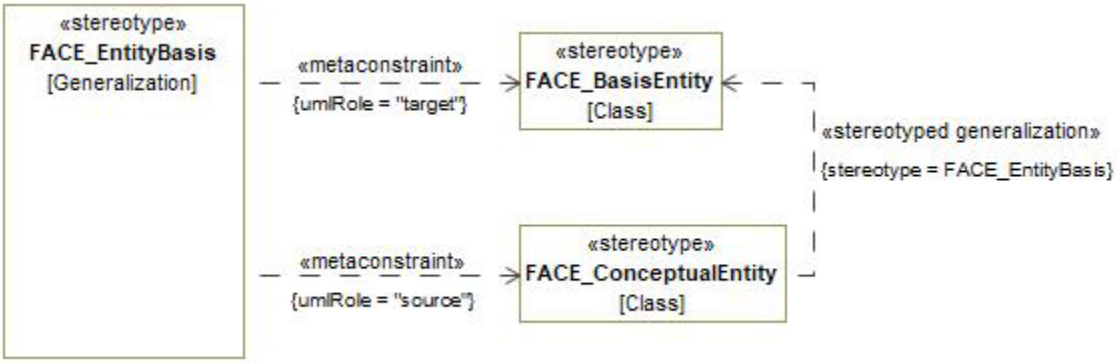

Figure 7-34: FACE_EntityBasis

Constraints

C01: FACE_EntityBasis.source	The value for the source metaproperty must be stereotyped by «FACE_ConceptualEntity» or a specialization of «FACE_ConceptualEntity».

C02: 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.

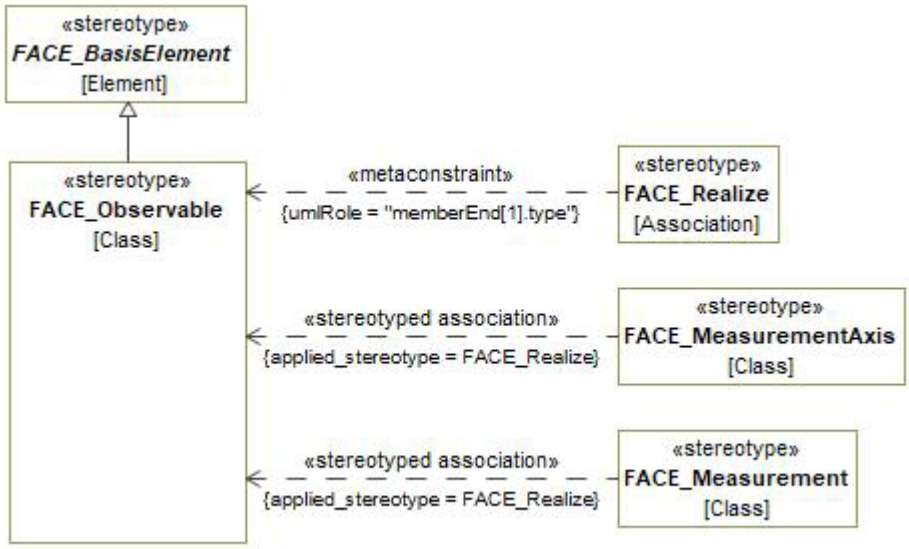

Figure 7-35: FACE_Observable

7.1.1.1.2 FACE_Profile::FACE Data Architecture::FACE Data Model::LogicalDataModel

The LogicalDataModel package of the FACE Profile contains elements that represent the Logical Data Model subpackage as specified in the UDDL metamodel.

FACE_AbstractMeasurement

Package: LogicalDataModel

isAbstract: Yes

Extension: Element

Description

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

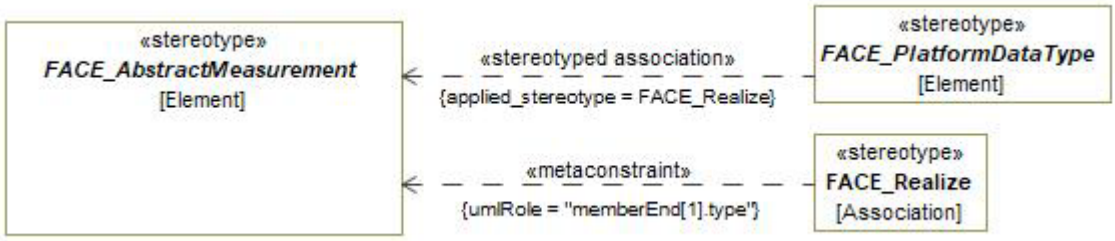

Figure 7-36: abstract FACE_AbstractMeasurement

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.

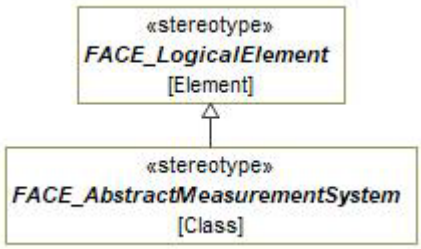

Figure 7-37: abstract FACE_AbstractMeasurementSystem

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.

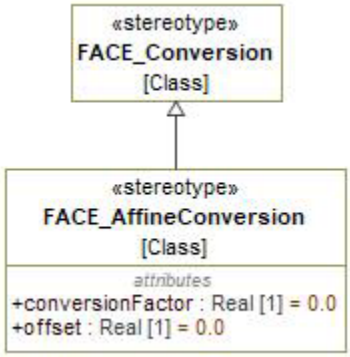

Figure 7-38: FACE_AffineConversion

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.

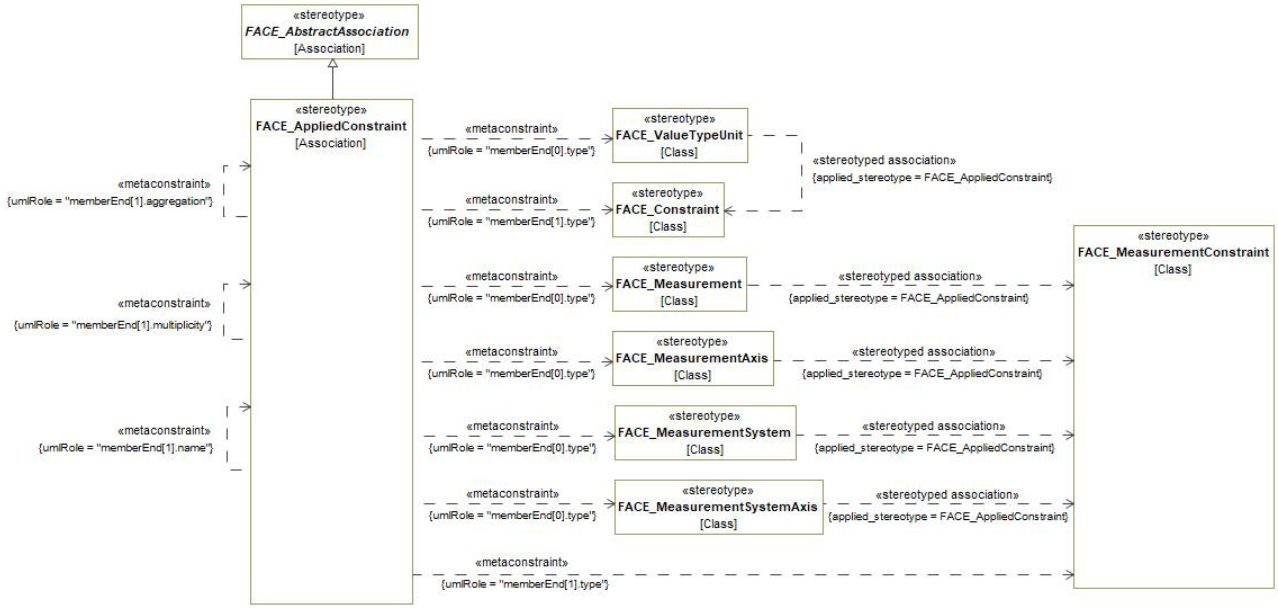

Figure 7-39: FACE_AppliedConstraint

Constraints

C01: 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»
C02: FACE_AppliedConstraint.memberEnd[1].aggregation	memberEnd[1].aggregation shall be composite
C03: FACE_AppliedConstraint.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 0..*
C04: FACE_AppliedConstraint.memberEnd[1].name	memberEnd[1].name shall be "constraint"

C05: 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.

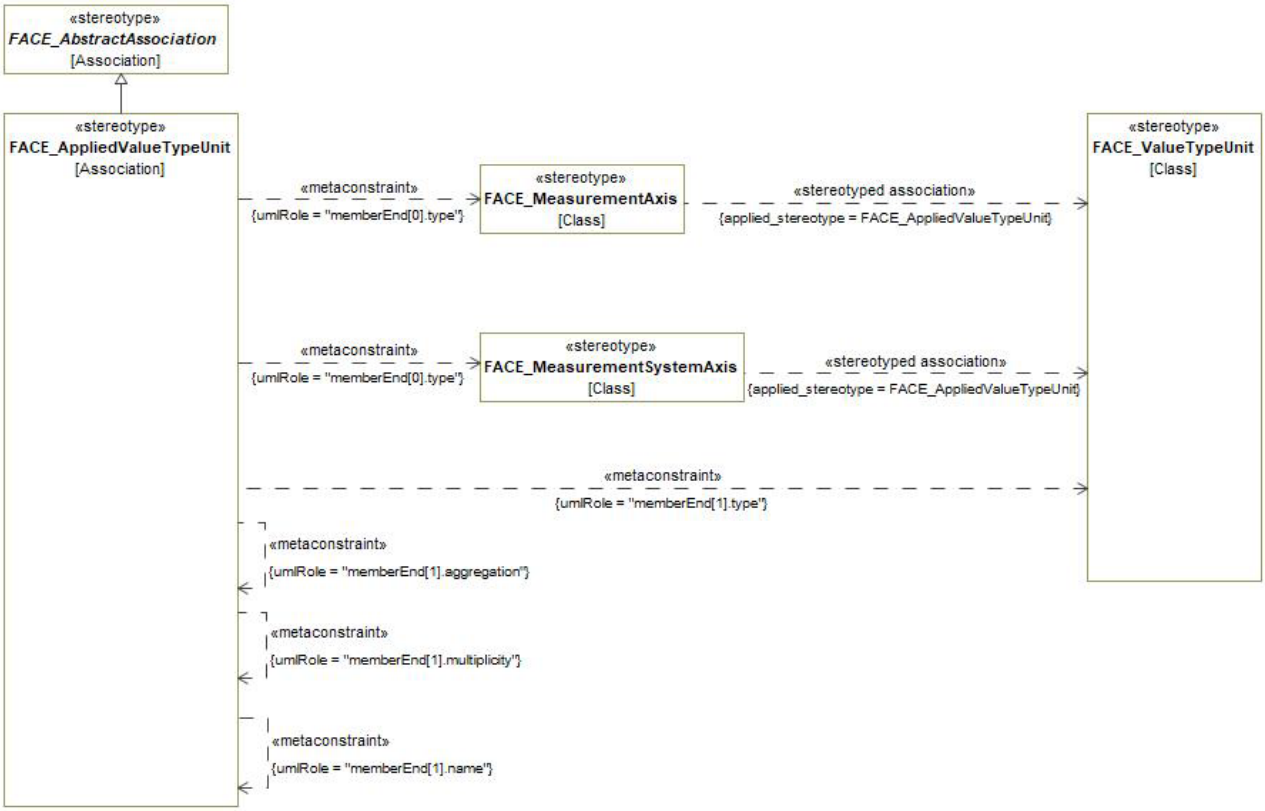

Figure 7-40: FACE_AppliedValueTypeUnit

Constraints

C01: 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»
C02: FACE_AppliedValueTypeUnit.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: 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..*

C04: 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"
C05: 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.

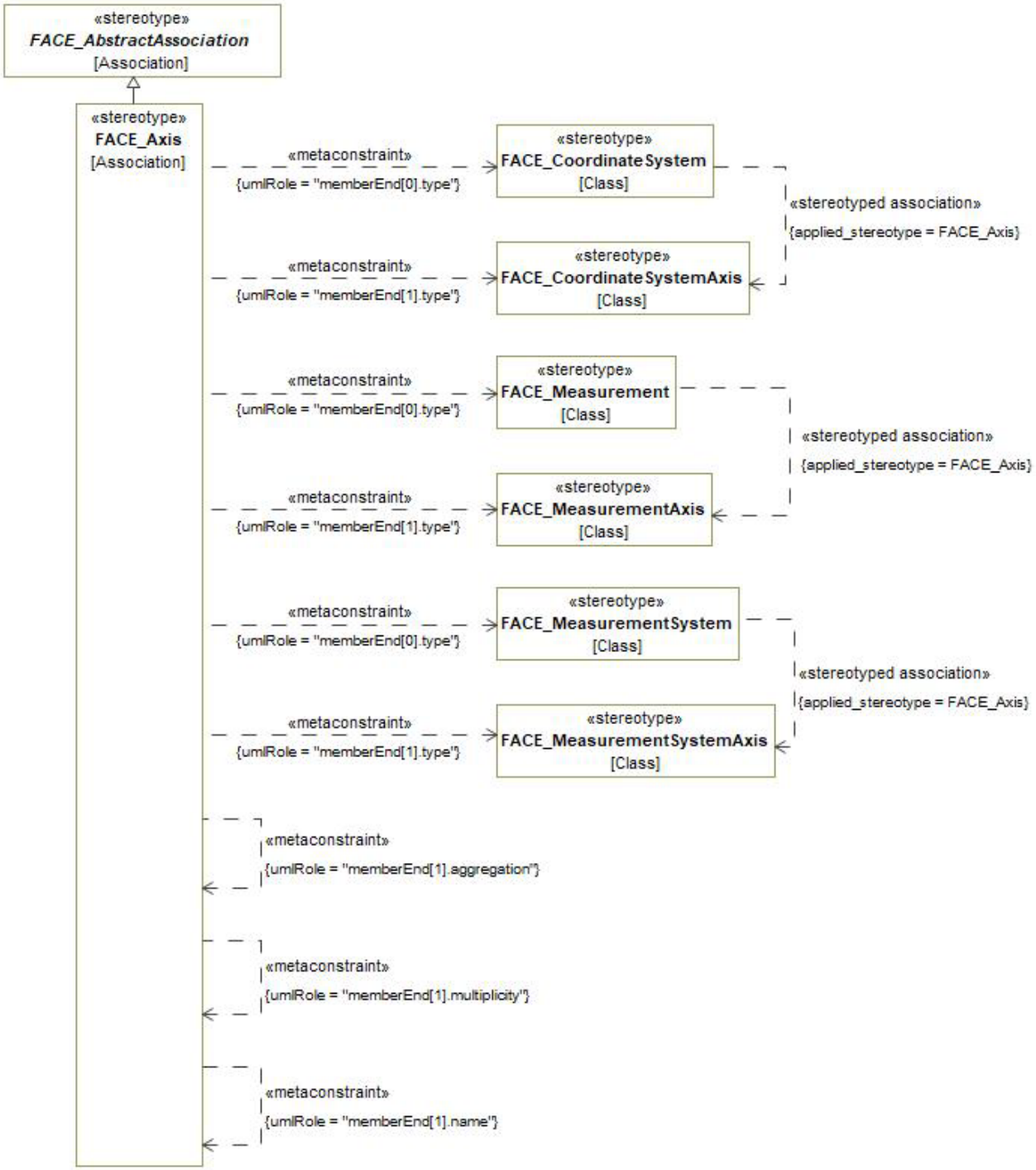

Figure 7-41: FACE_Axis

Constraints

C01: 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»
C02: FACE_Axis.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: 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
C04: 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"
C05: 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_DataModelElement

Extension: Class

Description

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

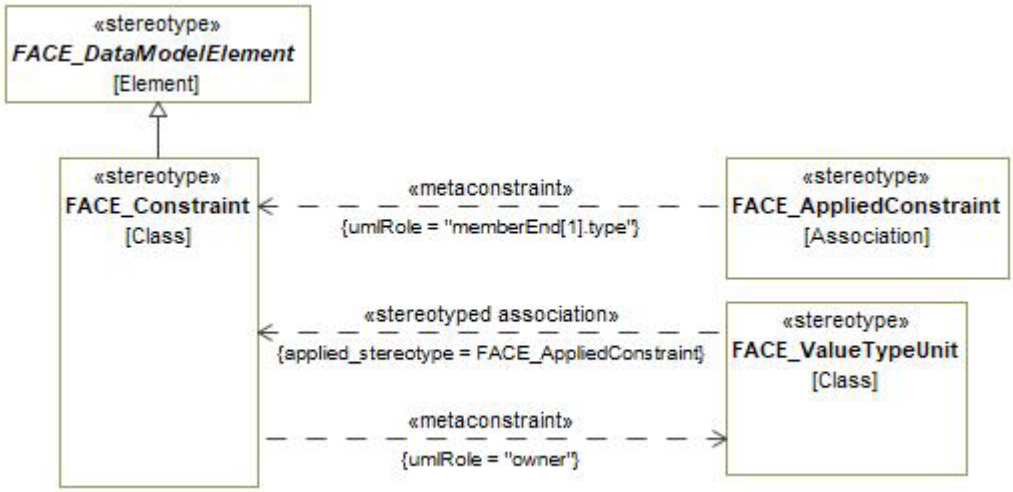

Figure 7-42: FACE_Constraint

Constraints

C01: 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.

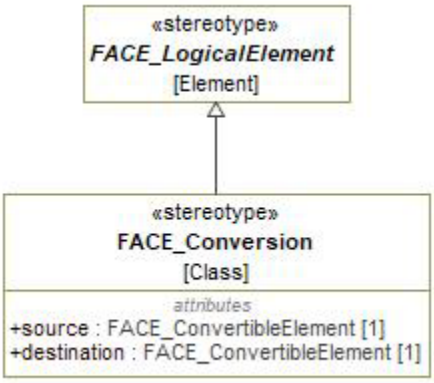

Figure 7-43: FACE_Conversion

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.

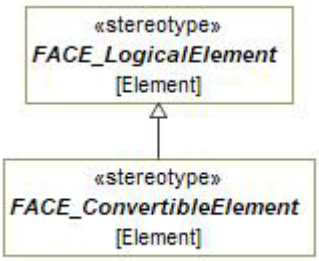

Figure 7-44: abstract FACE_ConvertibleElement

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.

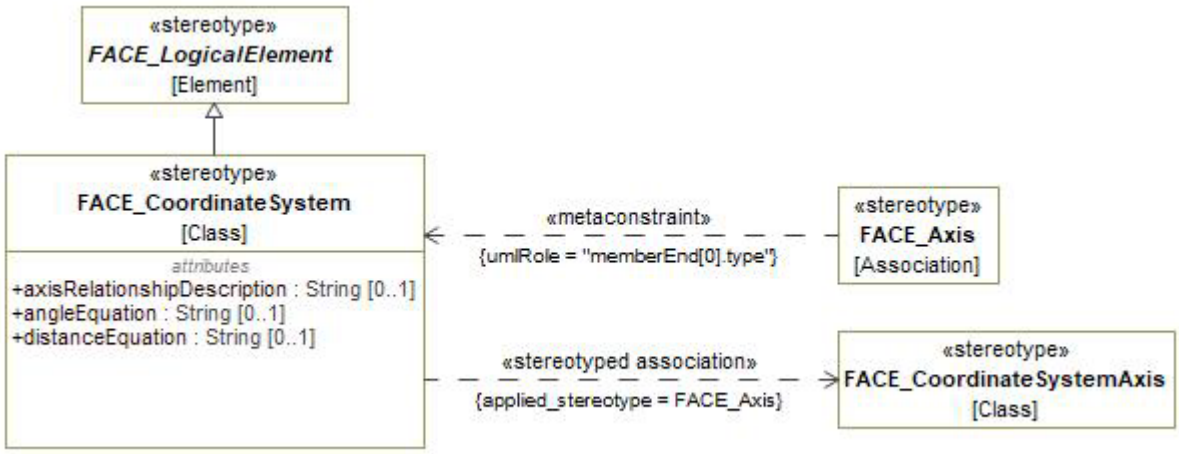

Figure 7-45: FACE_CoordinateSystem

Attributes

angleEquation : String [0..1]

axisRelationshipDescription : String [0..1]

distanceEquation : String [0..1]

FACE_CoordinateSystemAxis

Package: LogicalDataModel

isAbstract: No

Generalization: FACE_LogicalElement

Extension: Class

Description

A FACE_CoordinateSystemAxis represents a dimension within a FACE_CoordinateSystem.

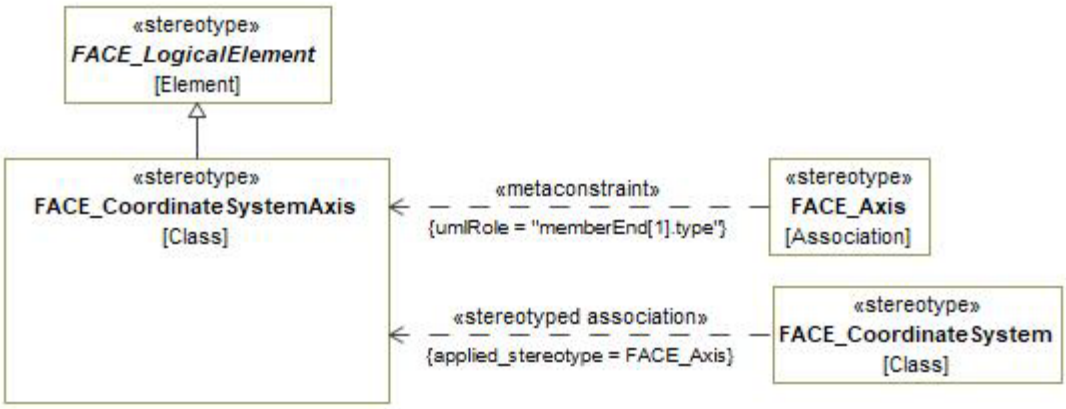

Figure 7-46: FACE_CoordinateSystemAxis

FACE_DefinedReferencePoint

Package: LogicalDataModel

isAbstract: No

Generalization: FACE_AbstractAssociation

Extension: Association

Description

Used to identify the reference point that characterizes a Measurement System.

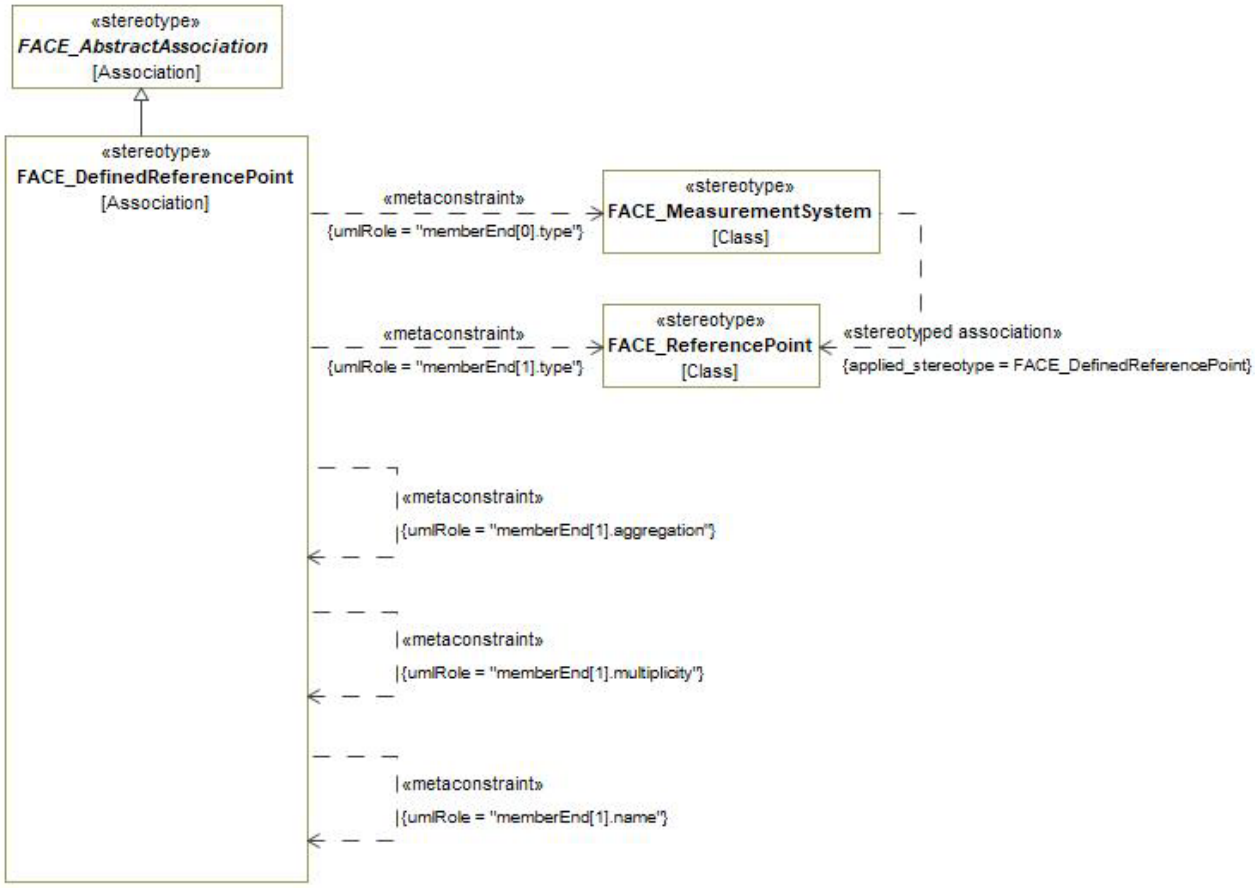

Figure 7-47: FACE_DefinedReferencePoint

Constraints

C01: FACE_DefinedReferencePoint.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_MeasurementSystem».
C02: FACE_DefinedReferencePoint.memberEnd[1].aggregation	memberEnd[1].aggregation shall be composite
C03: FACE_DefinedReferencePoint.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 0..*
C04: FACE_DefinedReferencePoint.memberEnd[1].name	memberEnd[1].name shall be "referencePoint"
C05: 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.

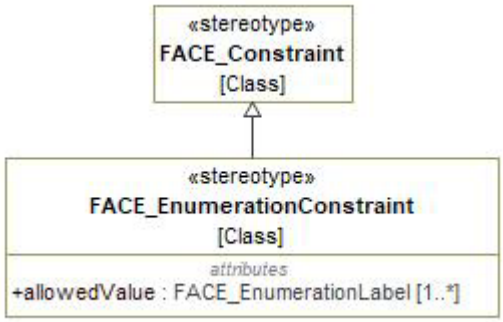

Figure 7-48: FACE_EnumerationConstraint

Attributes

allowedValue : FACE_EnumerationLabel [1..*]

FACE_EnumerationLabel

Package: LogicalDataModel

isAbstract: No

Generalization: FACE_DataModelElement

Extension: Property

Description

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

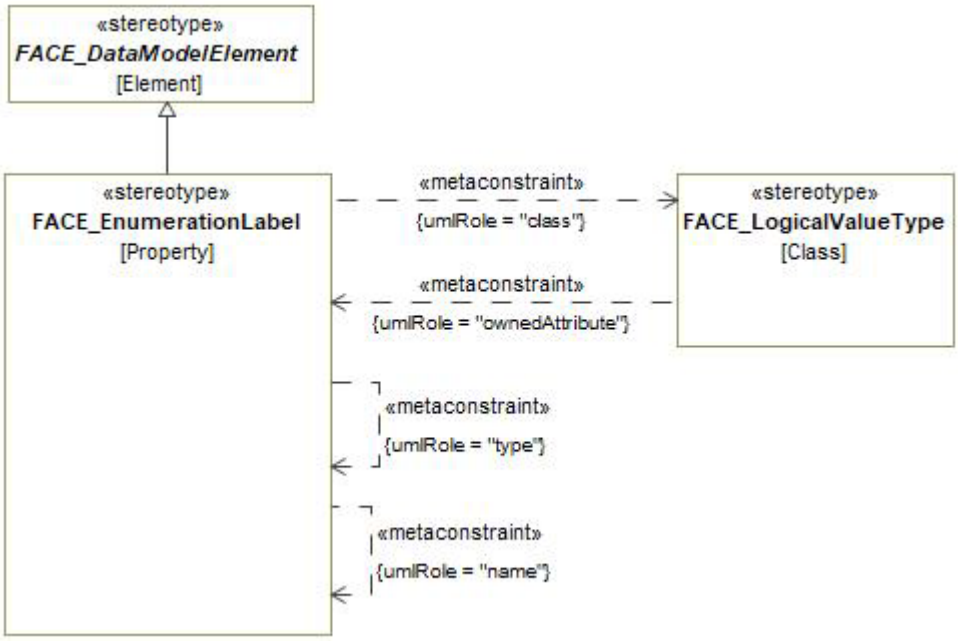

Figure 7-49: FACE_EnumerationLabel

Constraints

C01: FACE_EnumerationLabel.class	Value for the class metaproperty must be stereotyped «FACE_LogicalValueType»
C02: FACE_EnumerationLabel.name	Value for the name metaproperty must not be null and must be unique within the owning class.
C03: FACE_EnumerationLabel.type	Value for the type metaproperty must be null. (The name metaproperty is the only valid information.)

FACE Conformance/OCL Constraints

C01: 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.

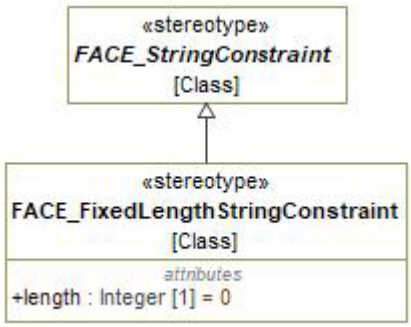

Figure 7-50: FACE_FixedLengthStringConstraint

Attributes

length : Integer [1]

FACE Conformance/OCL Constraints

C01: 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.

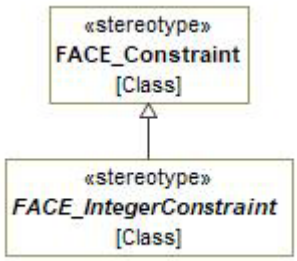

Figure 7-51: abstract FACE_IntegerConstraint

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.

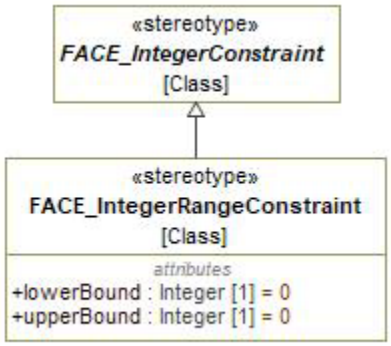

Figure 7-52: FACE_IntegerRangeConstraint

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.

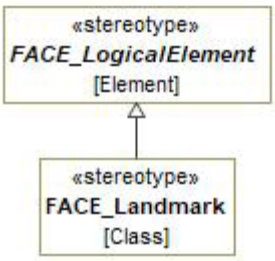

Figure 7-53: FACE_Landmark

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.

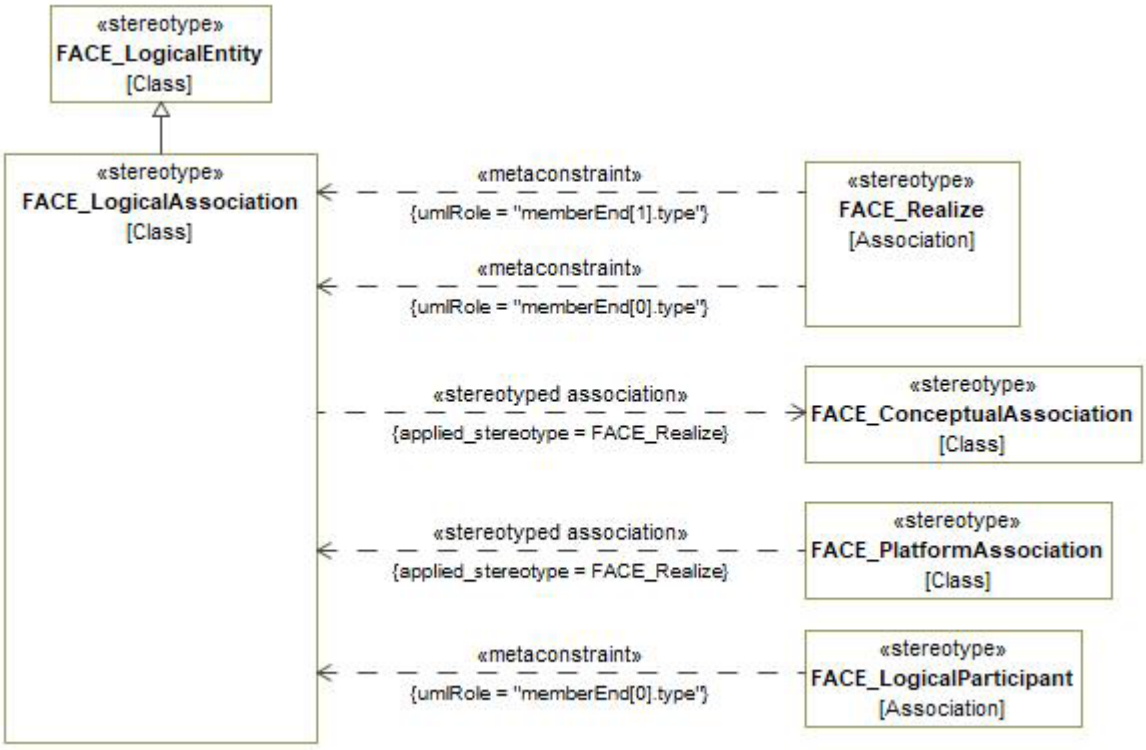

Figure 7-54: FACE_LogicalAssociation

FACE Conformance/OCL Constraints

C01: FACE_LogicalAssociation.participantsConsistentWithRealization	FACE_LogicalParticipants in a FACE_LogicalAssociation must realize FACE_ConceptualParticipants in the FACE_LogicalAssociation that the FACE_LogicalAssociation realizes.
C02: 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 "name" metatype attribute represents the data model "rolename" attribute that defines the name of the logical Characteristic within the scope of the logical Entity. The "lowerBound" and "upperBound" attributes define the multiplicity of the composed Characteristic. An "upperBound" multiplicity of -1 represents an unbounded sequence.

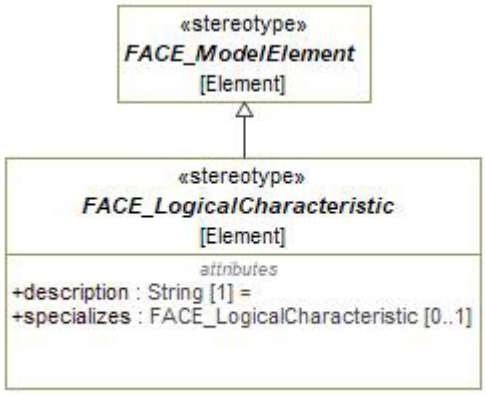

Figure 7-55: abstract FACE_LogicalCharacteristic

Attributes

description : String [1]

specializes : FACE_LogicalCharacteristic [0..1]

FACE Conformance/OCL Constraints

C01: FACE_LogicalCharacteristic.lowerBound_LTE_UpperBound	A FACE_LogicalCharacteristic's lowerBound must be less than or equal to its upperBound, unless its upperBound is -1.
C02: FACE_LogicalCharacteristic.rolenameIsValidIdentifier	The rolename of a FACE_LogicalCharacteristic must be a valid identifier.
C03: FACE_LogicalCharacteristic.specializationConsistentWithRealization	If a FACE_LogicalCharacteristic specializes, its specialization must be consistent with its realization's specialization.
C04: 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.

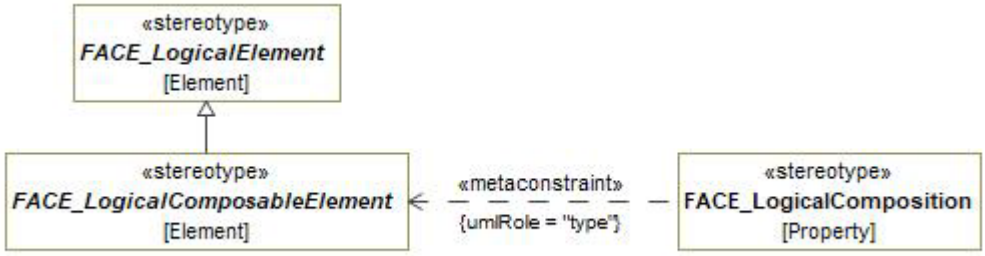

Figure 7-56: abstract FACE_LogicalComposableElement

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 union or as members of a struct.

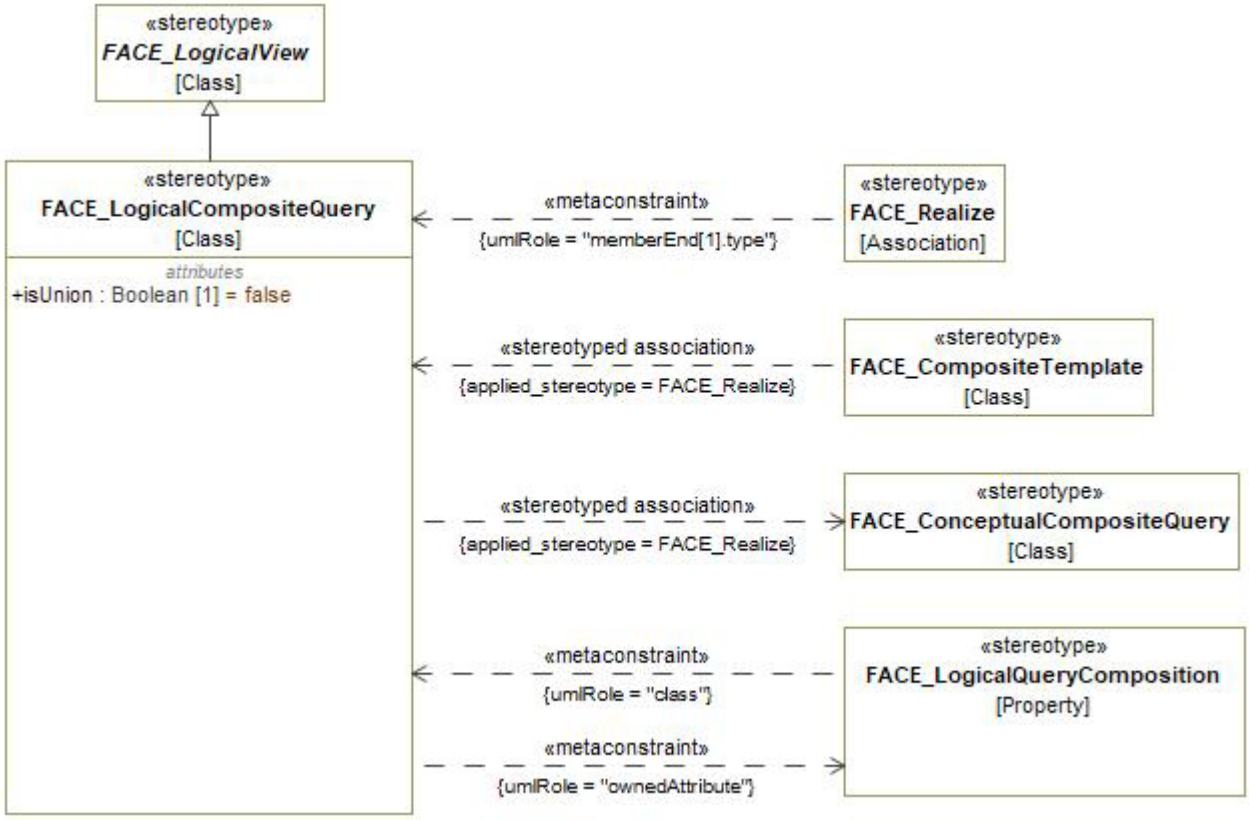

Figure 7-57: FACE_LogicalCompositeQuery

Attributes

isUnion : Boolean [1]

Constraints

C01: 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

C01: FACE_LogicalCompositeQuery.compositionsConsistentWithRealization	FACE_LogicalQueryCompositions in a FACE_LogicalCompositeQuery must realize FACE_ConceptualQueryCompositions in the FACE_ConceptualCompositeQuery that the FACE_LogicalCompositeQuery realizes.
C02: FACE_LogicalCompositeQuery.compositionsHaveUniqueRolenames	A FACE_LogicalQueryComposition's rolename must be unique within a FACE_LogicalCompositeQuery.
C03: FACE_LogicalCompositeQuery.noCyclesInConstruction	A FACE_LogicalCompositeQuery must not compose itself directly or indirectly.
C04: FACE_LogicalCompositeQuery.realizationUnionConsistent	A FACE_LogicalCompositeQuery that realizes must have the same "isUnion" property as the FACE_LogicalCompositeQuery it realizes.
C05: FACE_LogicalCompositeQuery.realizedCompositionsHaveDifferentTypes	A FACE_LogicalCompositeQuery must not contain two FACE_LogicalQueryCompositions that realize the same FACE_ConceptuallQueryComposition.
C06: 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 Logical Composition is the Logical ComposableElement being used to construct the logical Entity. The "lowerBound" and "upperBound" define the multiplicity of the composed logical Entity. An "upperBound" multiplicity of -1 represents an unbounded sequence.

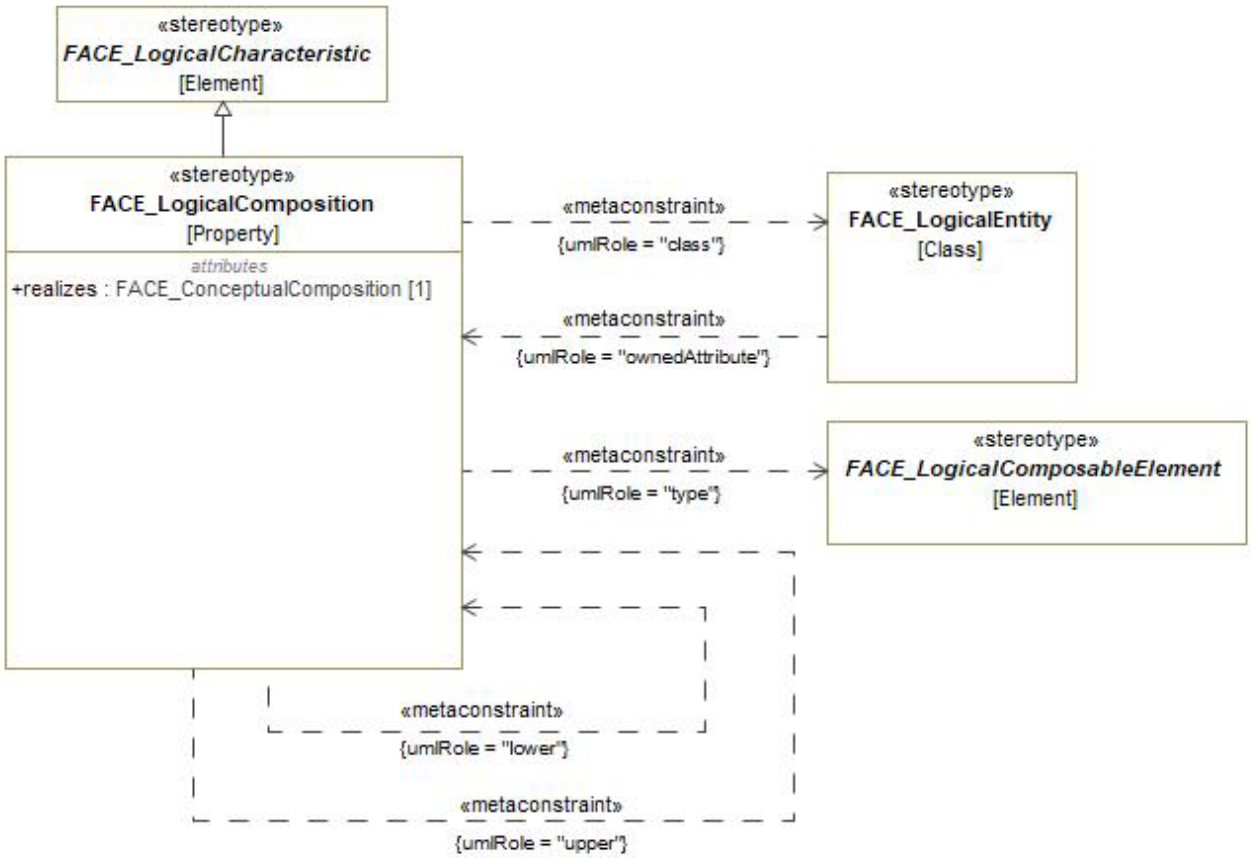

Figure 7-58: FACE_LogicalComposition

Attributes

realizes : FACE_ConceptualComposition [1]

Constraints

C01: FACE_LogicalComposition.class	Value for the class metaproperty must be stereotyped «FACE_LogicalEntity» or its specializations.
C02: FACE_LogicalComposition.lower	The value for the lower (lower bound of multiplicity) metaproperty must be an integer greater than or equal to -1.
C03: FACE_LogicalComposition.type	Value for the type metaproperty must be stereotyped «FACE_LogicalComposableElement» or its specializations.

C04: 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

C01: FACE_LogicalComposition.multiplicityConsistentWithRealization	A FACE_LogicalComposition's multiplicity must be at least as restrictive as the FACE_ConceptualComposition it realizes
C02: FACE_LogicalComposition.multiplicityConsistentWithSpecialization	A FACE_LogicalComposition's multiplicity must be at least as restrictive as the FACE_LogicalComposition of which it is a specialization.
C03: 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 Data Model Language.

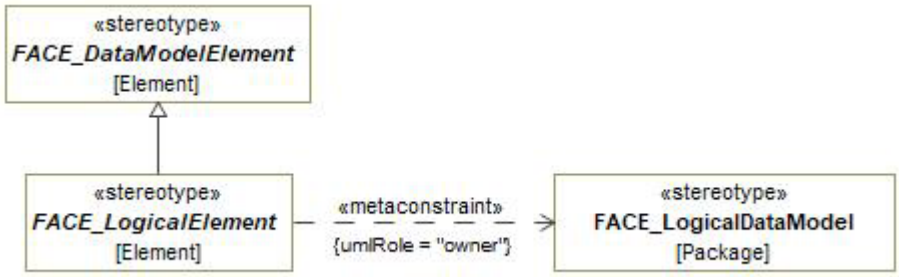

Figure 7-59: abstract FACE_LogicalElement

Constraints

C01: 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

C01: 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_SpecializationOwner

Extension: Class

Description

A FACE_LogicalEntity "realizes" a FACE_ConceptualEntity in terms of Measurements and other 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.

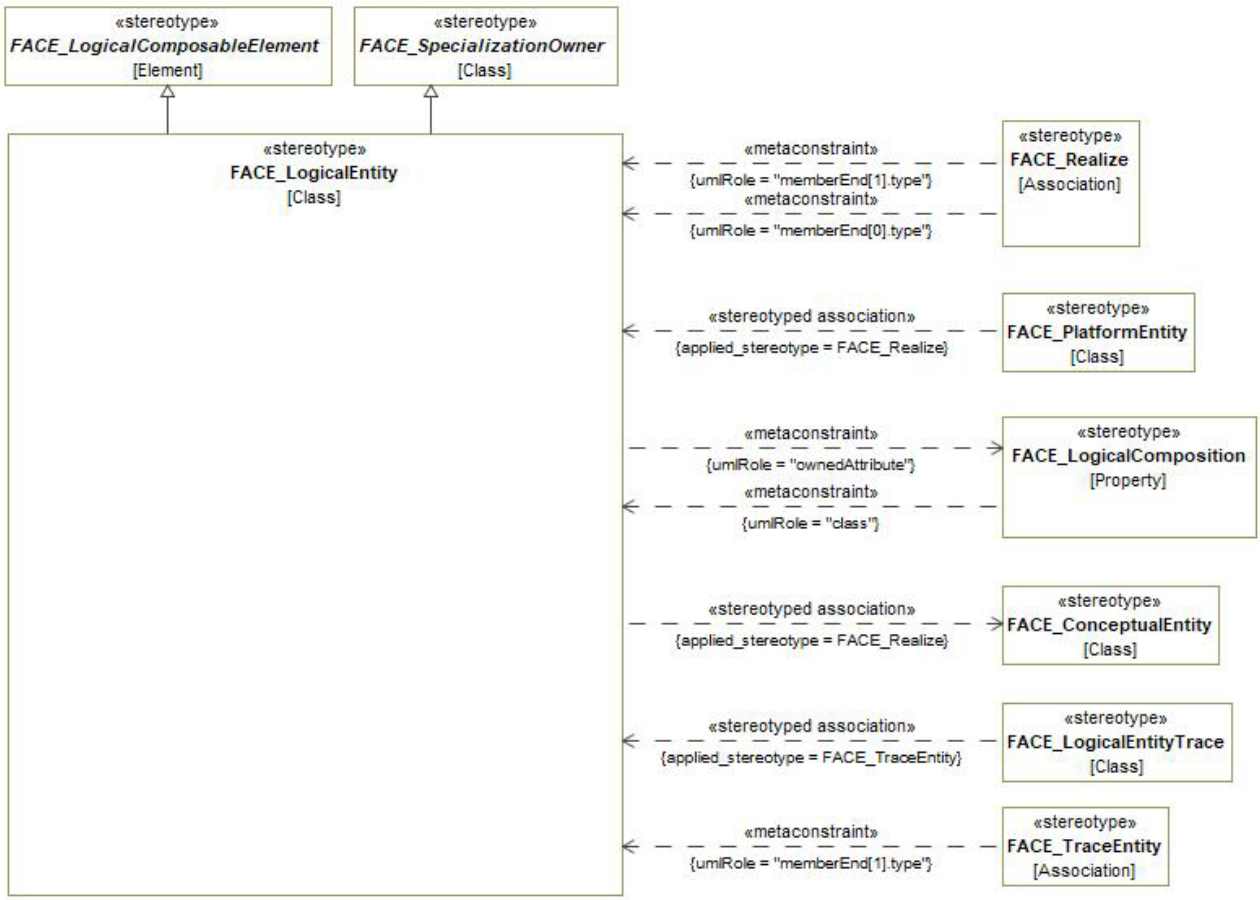

Figure 7-60: FACE_LogicalEntity

Constraints

C01: FACE_LogicalEntity.ownedAttribute	The value for the ownedAttribute metaproperty must be stereotyped «FACE_LogicalComposition» or its specializations

FACE Conformance/OCL Constraints

C01: FACE_LogicalEntity.characteristicsHaveUniqueRolenames	A FACE_LogicalCharacteristic's rolename must be unique within a FACE_LogicalEntity.
C02: FACE_LogicalEntity.compositionsConsistentWithRealization	FACE_LogicalCompositions in a FACE_LogicalEntity must realize FACE_ConceptualCompositions in the conceptual FACE_ConceptualEntity that the FACE_LogicalEntity realizes.
C03: FACE_LogicalEntity.hasAtLeastOneLocalCharacteristic	A FACE_LogicalEntity must have at least one Characteristic defined locally (not through generalization), unless the Entity is in the "middle" of a generalization hierarchy.
C04: 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.
C05: 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: Association

Description

A FACE_LogicalParticipant is the mechanism that allows a FACE_LogicalAssociation to be constructed between two or more FACE_LogicalEntities. The "type" (target of the directional Association) of a logical Participant is the logical Entity being used to construct the logical Association. Target multiplicity values represent the "sourceLowerBound" and "sourceUpperBound" attributes that define the multiplicity of the logical Association relative to the Participant in the UDDL metamodel. An upper multiplicity of star (*) on the target of the association is the equivalent of a "sourceUpperBound" multiplicity of -1 (which represents an unbounded sequence) in the the UDDL metamodel. 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.

FACE_LogicalParticipant Associations are directional, from a FACE_LogicalAssociation to a FACE_LogicalEntity.

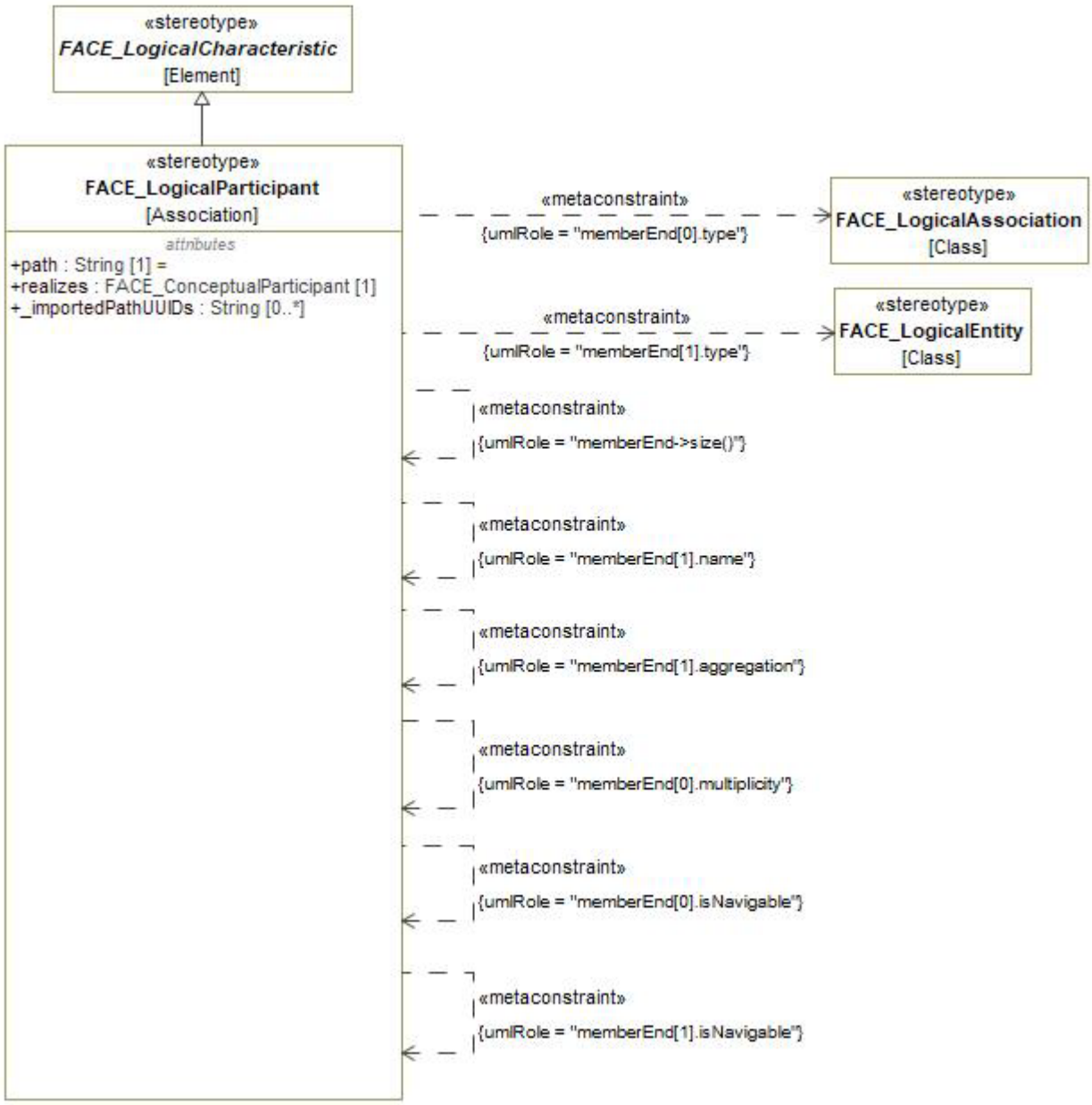

Figure 7-61: FACE_LogicalParticipant

Attributes

path : String [1]	The "path" property indicates the portion of the target «FACE_LogicalEntity» that is participating in the «FACE_LogicalAssociation» that is the source for the

	«FACE_LogicalParticipant» Association. Path strings reference Entities or Characteristics (properties of Entities). Where the path string references an Entity, it is considered to be a ParticipantPathNode. Where the path string references a Characteristic of an Entity, it is considered to be a CharacteristicPathNode. The UDDL metamodel defines PathNode, ParticipantPathNode and CharacteristicPathNode as follows: A logical PathNode is a single element in a chain that collectively forms a path specification. A logical ParticipantPathNode is a logical PathNode that selects a Participant that references an Entity. This provides a mechanism for reverse navigation from an Entity that participates in an Association back to the Association. A logical CharacteristicPathNode is a logical PathNode that selects a logical Characteristic which is directly contained in a logical Entity or Association. The strings provided in the "path" tagged value are a representation of the full set of Logical CharacteristicPathNode, ParticipantPathNode, and PathNode elements in the path attribute as specified in the UDDL Standard. 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 and the FACE XMI (face) file are required to translate between the two notations.
realizes : FACE_ConceptualParticipant [1]	
_importedPathUUIDs : String [0..*]	This tag is for use by import/export plug-ins in two-way translation of FACE 3.x paths to and from FACE 2.1 path strings. It is used to preserve the UUIDs of the paths imported from FACE 3.x paths when they are translated into FACE 2.1 path strings, so that they can be reconstituted for subsequent export as FACE 3.x elements. Because this tag is used exclusively by the plug-ins, its implementation is optional if a tool either does not import/export FACE format files or the tool uses an alternate means of representing and translating FACE Paths.

Constraints

C01: FACE_LogicalParticipant.memberEnd->size()	memberEnd.size() shall be 2
C02: FACE_LogicalParticipant.memberEnd[0].isNavigable	memberEnd[0].isNavigable shall be false

C03: FACE_LogicalParticipant.memberEnd[0].multiplicity	memberEnd[0].multiplicity shall be 1
C04: FACE_LogicalParticipant.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_LogicalAssociation»
C05: FACE_LogicalParticipant.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C06: FACE_LogicalParticipant.memberEnd[1].isNavigable	memberEnd[1].isNavigable shall be true
C07: FACE_LogicalParticipant.memberEnd[1].name	The memberEnd[1].name metaproperty must be an non-empty alphanumeric name string
C08: FACE_LogicalParticipant.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_LogicalEntity»

FACE Conformance/OCL Constraints

C01: FACE_LogicalParticipant.multiplicityConsistentWithRealization	A FACE_LogicalParticipant's multiplicity must be at least as restrictive as the FACE_ConceptualParticipant it realizes.
C02: FACE_LogicalParticipant.multiplicityConsistentWithSpecialization	A FACE_LogicalParticipant's multiplicity must be at least as restrictive as the FACE_LogicalParticipant it specializes.
C03: FACE_LogicalParticipant.rolenameDefined	A FACE_LogicalParticipant must have a rolename, either projected from a characteristic or defined directly on the FACE_LogicalParticipant.
C04: 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.

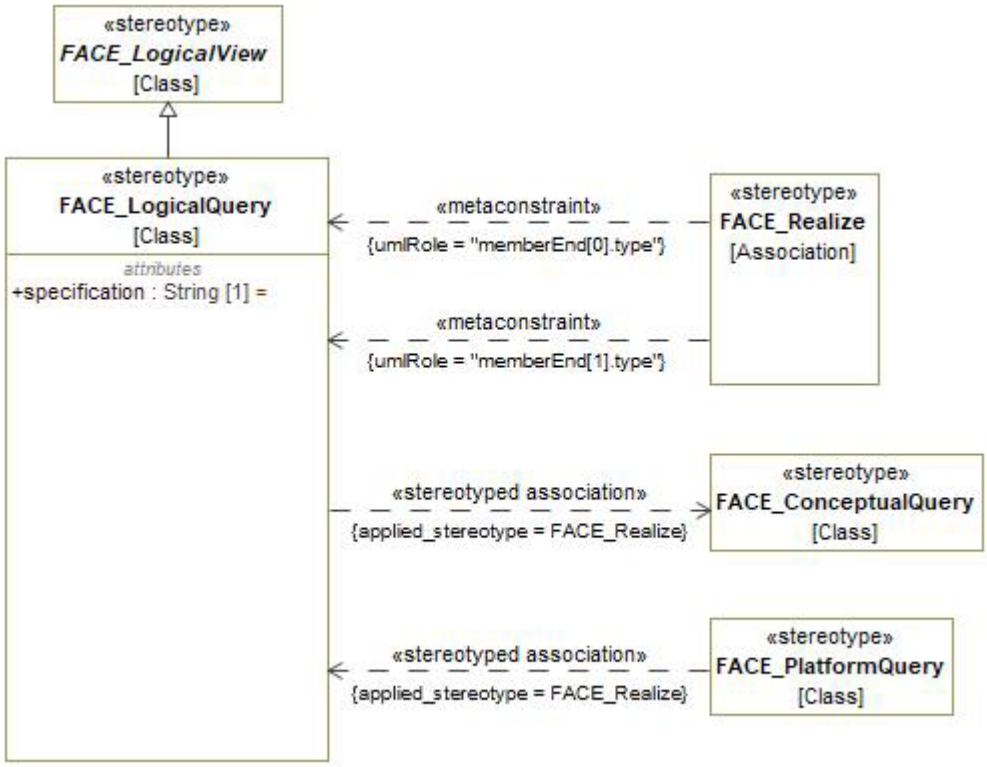

Figure 7-62: FACE_LogicalQuery

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 "name" metamodel attribute represents the "rolename" attribute in UDDL that 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.

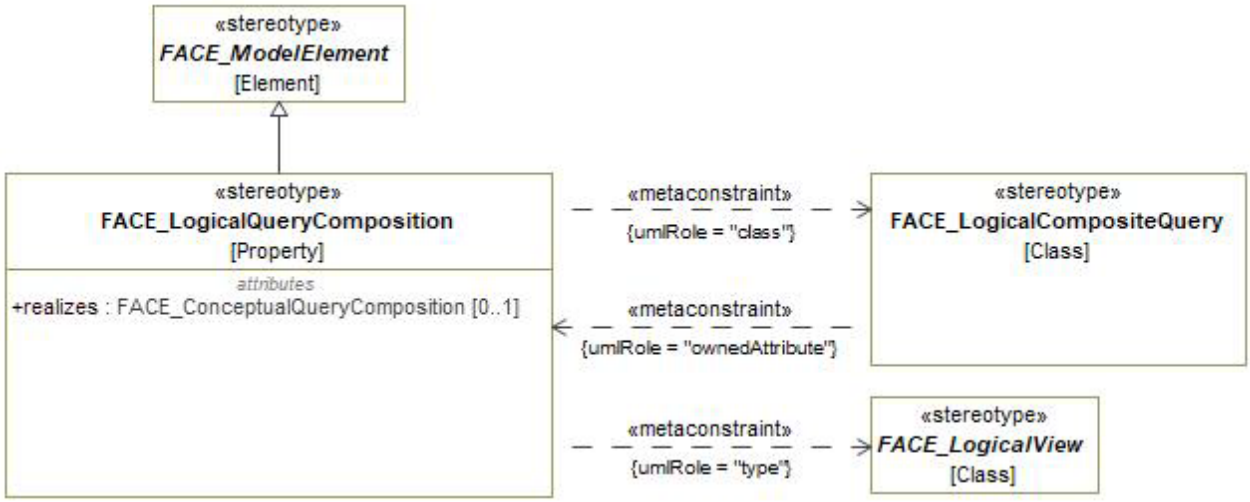

Figure 7-63: FACE_LogicalQueryComposition

Attributes

realizes : FACE_ConceptualQueryComposition [0..1]

Constraints

C01: FACE_LogicalQueryComposition.class	Value for the class metaproperty must be stereotyped «FACE_LogicalCompositeQuery».
C02: FACE_LogicalQueryComposition.type	Value for the type metaproperty must be stereotyped «FACE_LogicalView» or its specializations.

FACE Conformance/OCL Constraints

C01: FACE_LogicalQueryComposition.rolenameIsValidIdentifier	The rolename of a FACE_LogicalQueryComposition must be a valid identifier.
C02: 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 UDDL Standard.

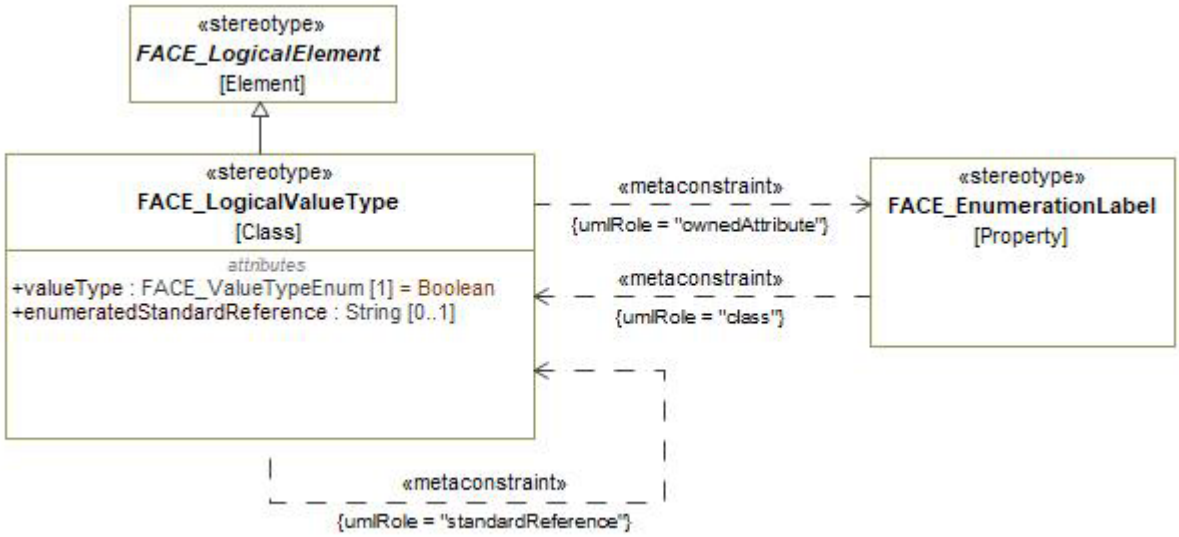

Figure 7-64: FACE_LogicalValueType

Attributes

enumeratedStandardReference : String [0..1]

valueType : FACE_ValueTypeEnum [1]

Constraints

C01: 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».
C02: FACE_LogicalValueType.standardReference	standardReference may only have a value if valueType = Enumerated

FACE Conformance/OCL Constraints

C01: FACE_LogicalValueType.enumerationLabelNameUnique	If the value type is Enumeration (value_type == FACE_ValueTypeEnum.Enumerated), all owned attribute FACE_EnumerationLabels must have unique names.
C02: FACE_LogicalValueType.nameIsNotReservedWord	If the value type is Enumeration (value_type == FACE_ValueTypeEnum.Enumerated), the Enumerated's name must not be an IDL reserved word.
C03: FACE_LogicalValueType.nameOfValueTypeMatchesNameOfMetaclass	A FACE_LogicalValueType must be named the same as its metatype. (e.g. a String must be named "String")

FACE_LogicalView

Package: LogicalDataModel

isAbstract: Yes

Generalization: FACE_LogicalElement

Extension: Class

Description

A FACE_LogicalView is a FACE_LogicalQuery or a FACE_LogicalCompositeQuery.

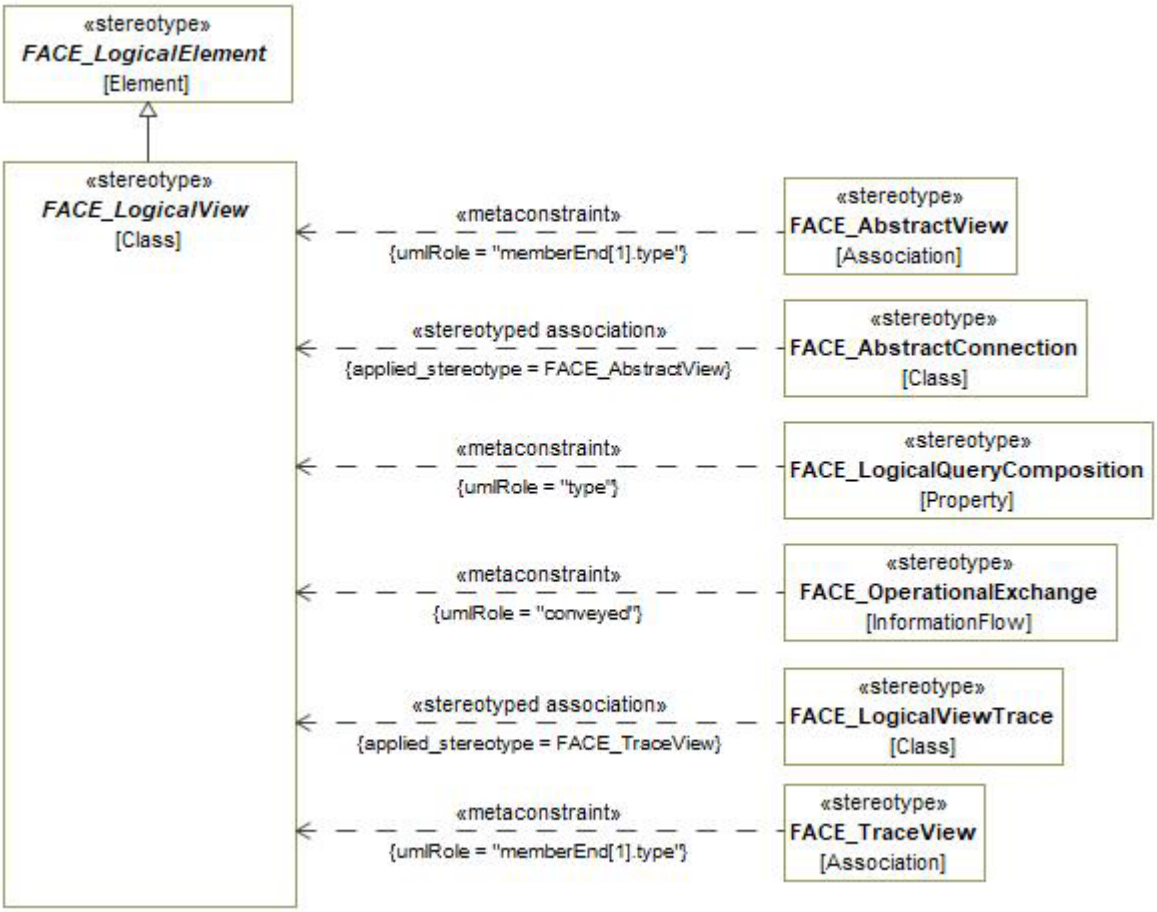

Figure 7-65: abstract FACE_LogicalView

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.

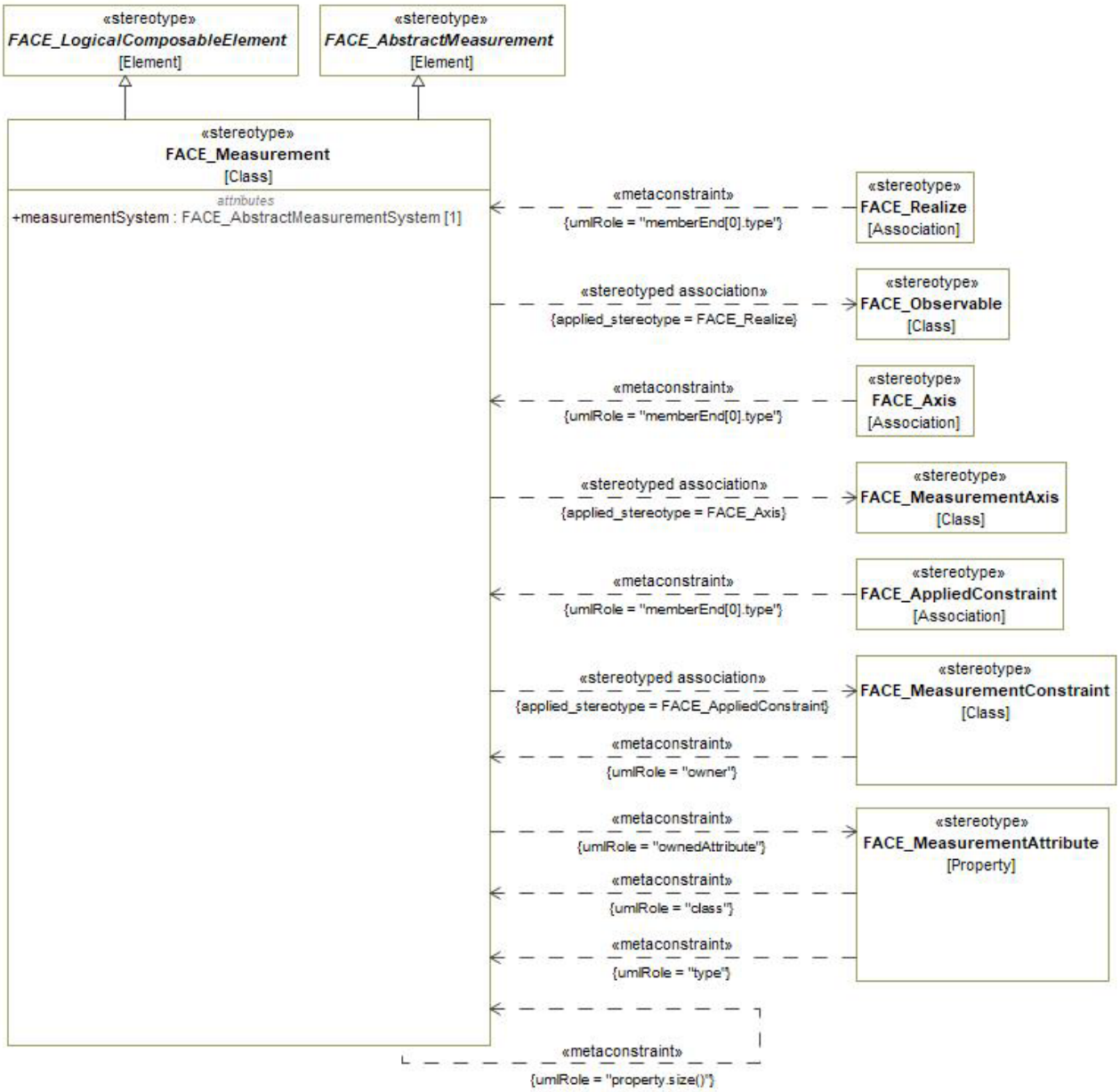

Figure 7-66: FACE_Measurement

Attributes

measurementSystem : FACE_AbstractMeasurementSystem [1]

Constraints

C01: 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

C01: FACE_Measurement.enumeratedMeasurementUsesEnumeratedMeasurementSystem	A Measurement that uses an Enumerated ValueType in any of its axes must be based on the 'AbstractDiscreteSetMeasurementSystem' MeasurementSystem.
C02: FACE_Measurement.measurementAttributesHaveUniqueRolenames	A FACE_Measurement's attributes must have unique rolenames.
C03: 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.
C04: 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.

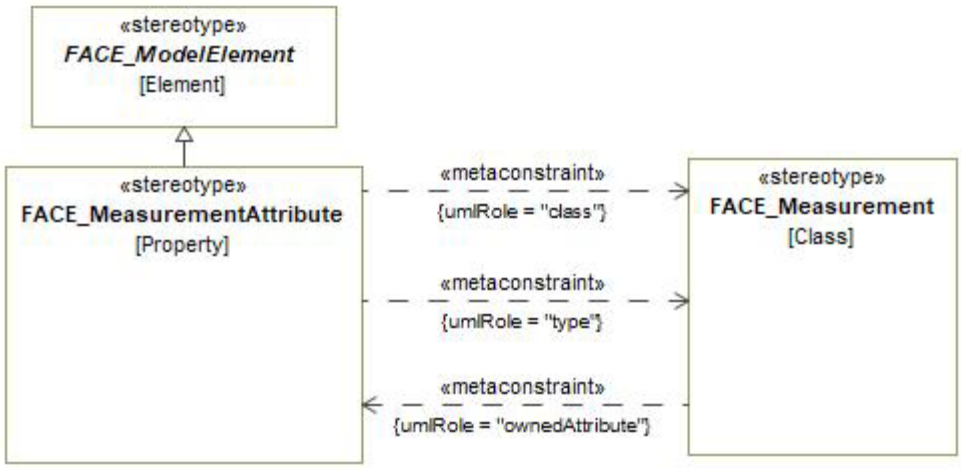

Figure 7-67: FACE_MeasurementAttribute

Constraints

C01: FACE_MeasurementAttribute.class	Value for the class metaproperty must be stereotyped «FACE_Measurement»
C02: 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.

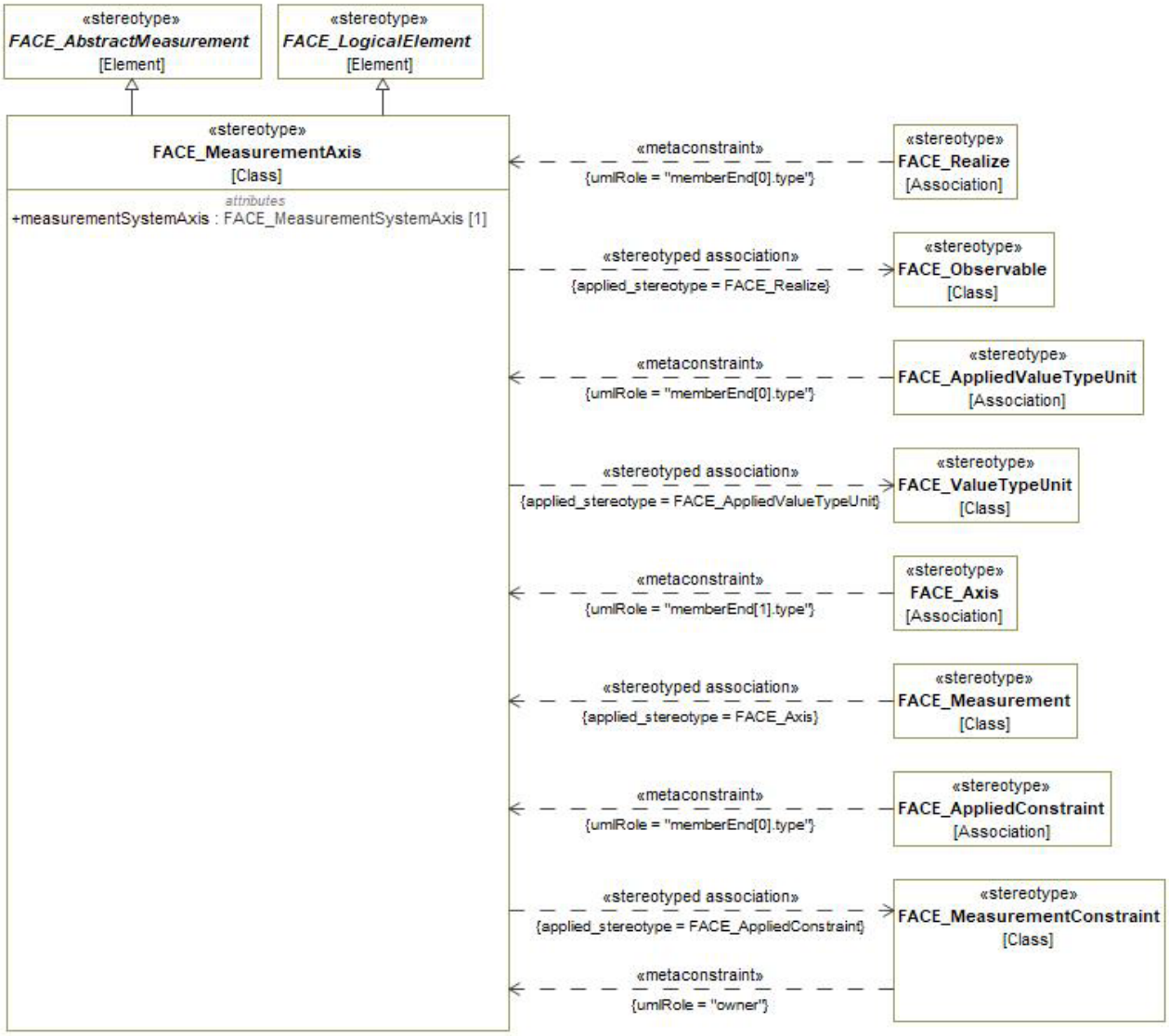

Figure 7-68: FACE_MeasurementAxis

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.

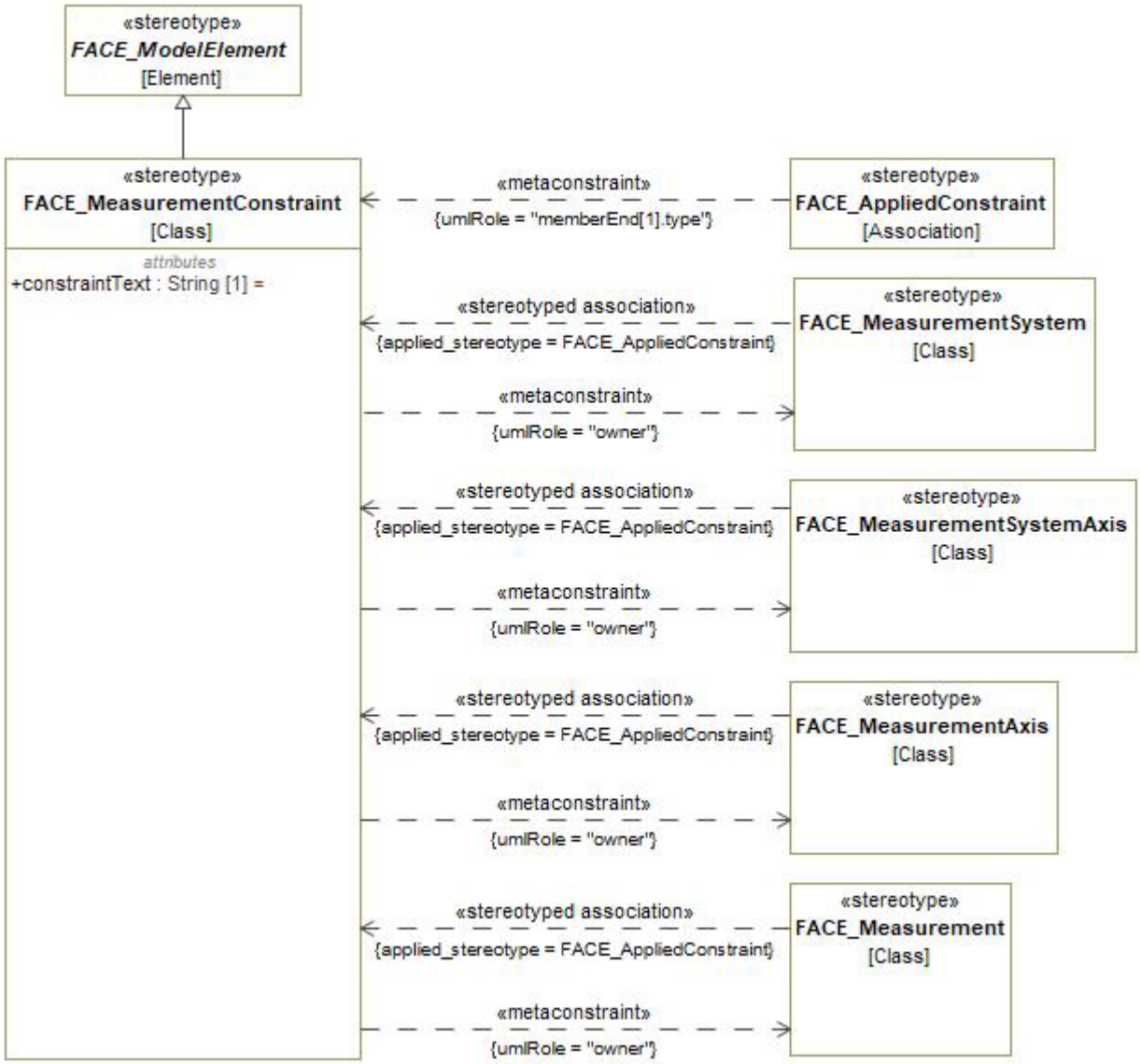

Figure 7-69: FACE_MeasurementConstraint

Attributes

constraintText : String [1]

Constraints

C01: 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.

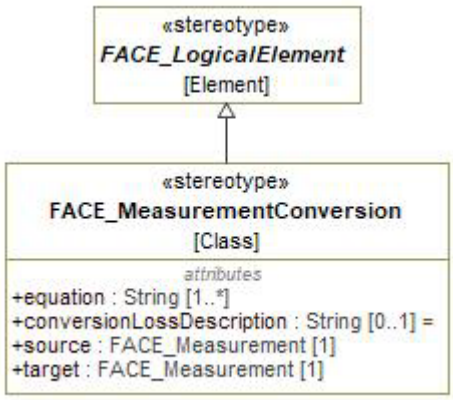

Figure 7-70: FACE_MeasurementConversion

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.

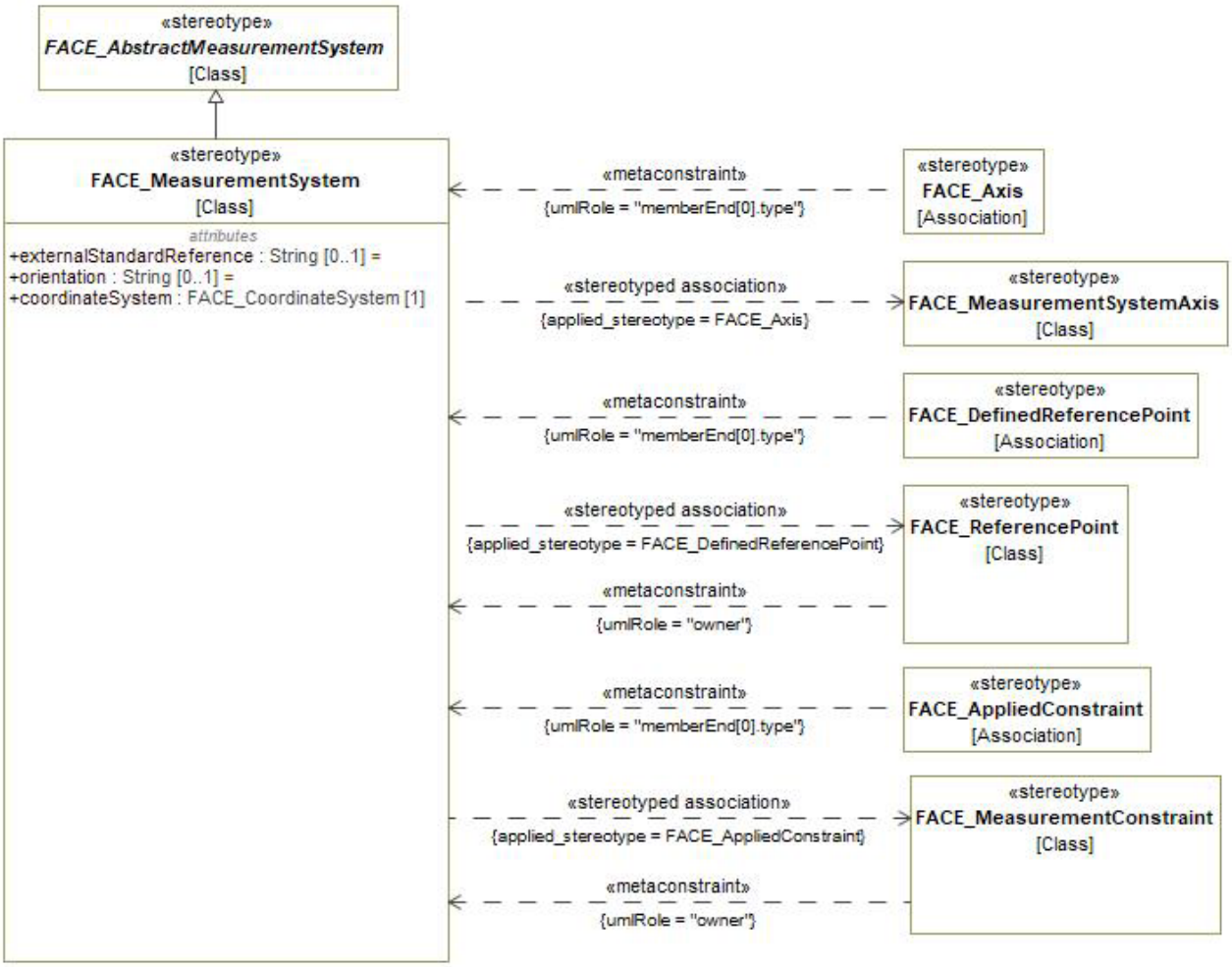

Figure 7-71: FACE_MeasurementSystem

Attributes

coordinateSystem : FACE_CoordinateSystem [1]

externalStandardReference : String [0..1]

orientation : String [0..1]

FACE Conformance/OCL Constraints

C01: FACE_MeasurementSystem.hasSufficientReferencePoints	If a FACE_MeasurementSystem has FACE_ReferencePoints, then it must have at least as many FACE_ReferencePoints as it has axes.
C02: 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.
C03: 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.
C04: 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.
C05: 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.

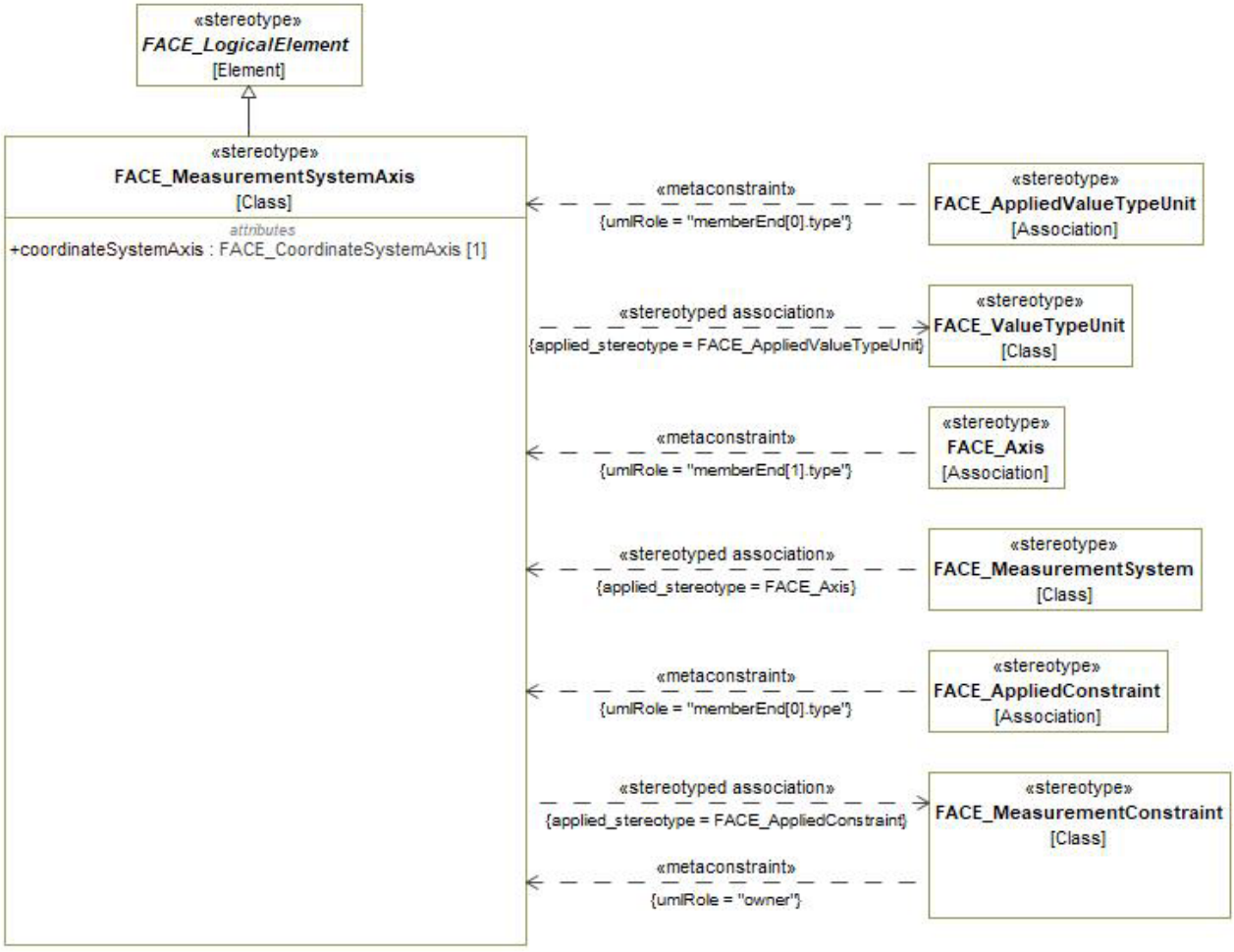

Figure 7-72: FACE_MeasurementSystemAxis

Attributes

coordinateSystemAxis : FACE_CoordinateSystemAxis [1]

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.

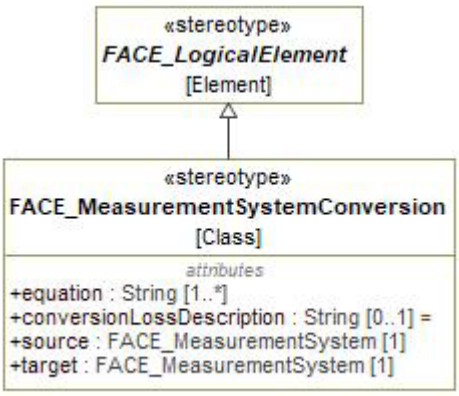

Figure 7-73: FACE_MeasurementSystemConversion

Attributes

conversionLossDescription : String [0..1]

equation : String [1..*]

source : FACE_MeasurementSystem [1]

target : FACE_MeasurementSystem [1]

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.

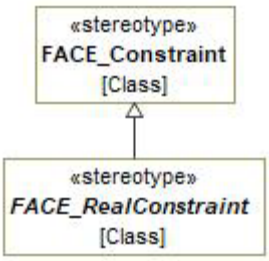

Figure 7-74: abstract FACE_RealConstraint

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".

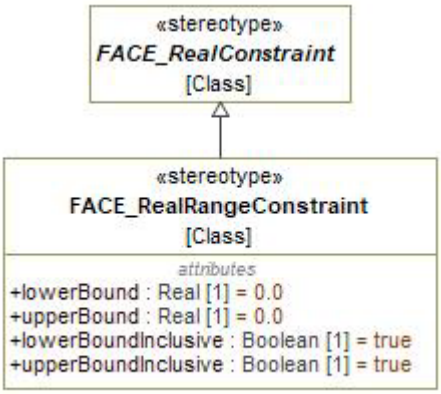

Figure 7-75: FACE_RealRangeConstraint

Attributes

lowerBound : Real [1]

lowerBoundInclusive : Boolean [1]

upperBound : Real [1]

upperBoundInclusive : Boolean [1]

FACE_ReferencePoint

Package: LogicalDataModel

isAbstract: No

Generalization: FACE_DataModelElement

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.

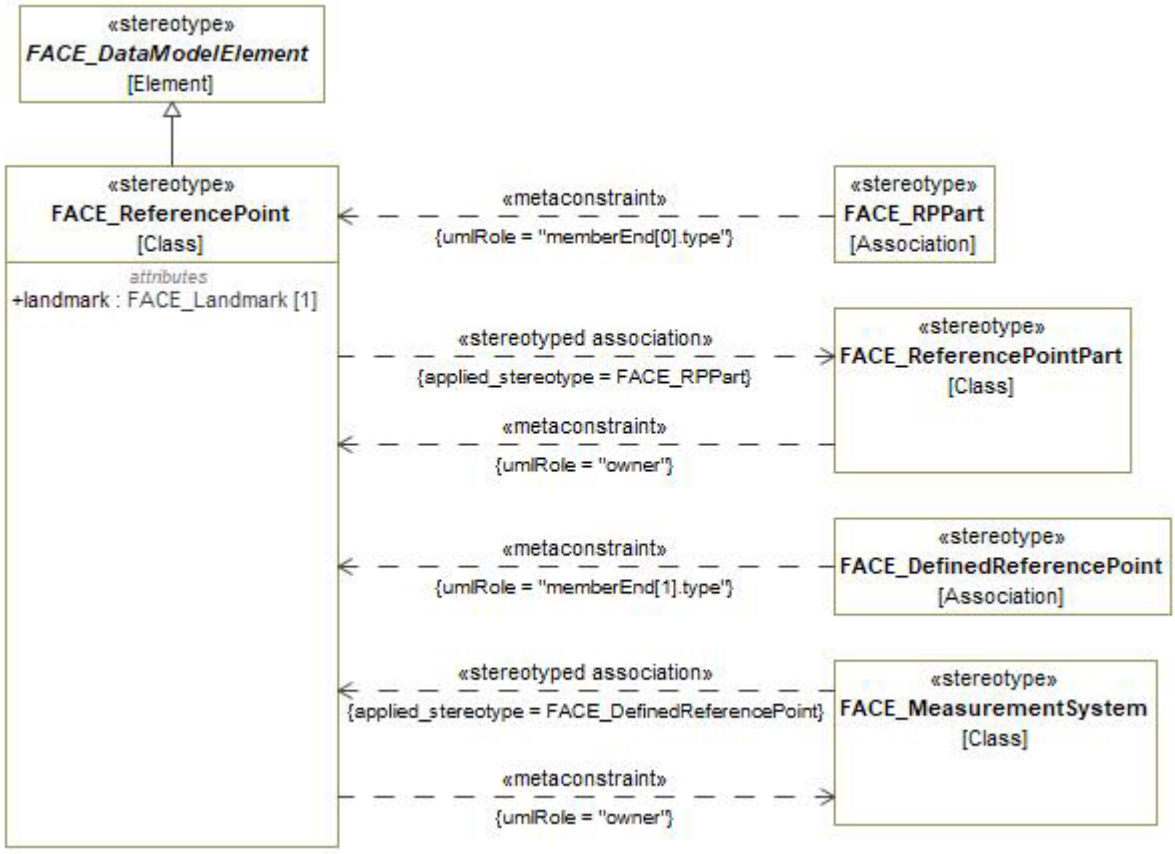

Figure 7-76: FACE_ReferencePoint

Attributes

landmark : FACE_Landmark [1]

Constraints

C01: FACE_ReferencePoint.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_MeasurementSystem»

FACE Conformance/OCL Constraints

C01: FACE_ReferencePoint.noAmbiguousVTUReference	If two ReferencePointParts in a FACE_ReferencePoint refer to the same VTU, then they must refer to distinct (non-null) axes.

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.

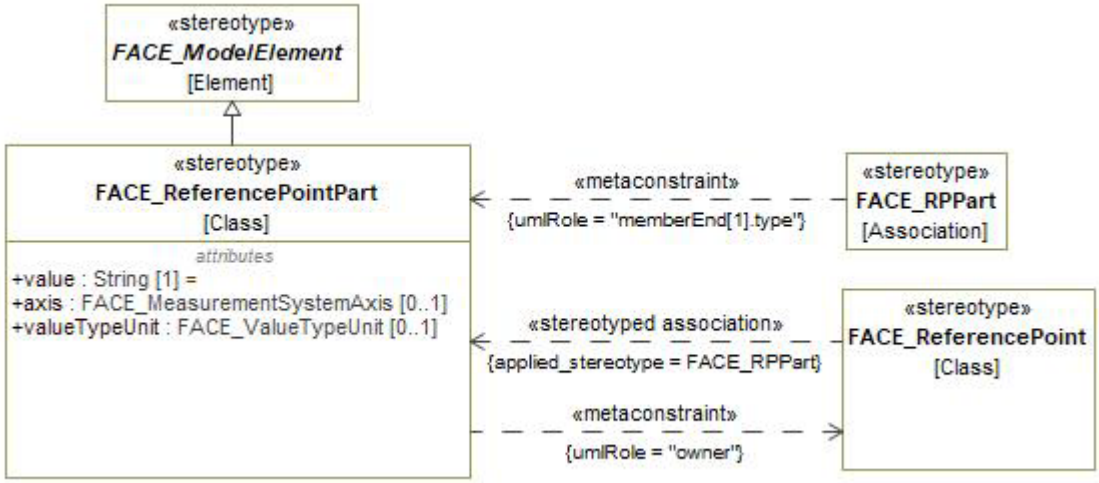

Figure 7-77: FACE_ReferencePointPart

Attributes

axis : FACE_MeasurementSystemAxis [0..1]

value : String [1]

valueTypeUnit : FACE_ValueTypeUnit [0..1]

Constraints

C01: FACE_ReferencePointPart.owner	This element may only be contained in (owned by) elements with the stereotype «FACE_ReferencePoint»

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.

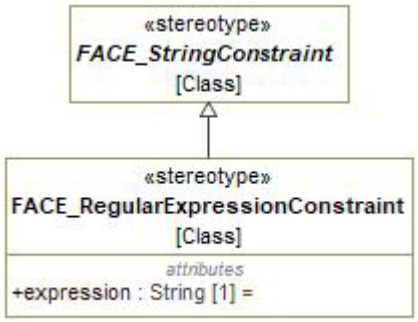

Figure 7-78: FACE_RegularExpressionConstraint

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.

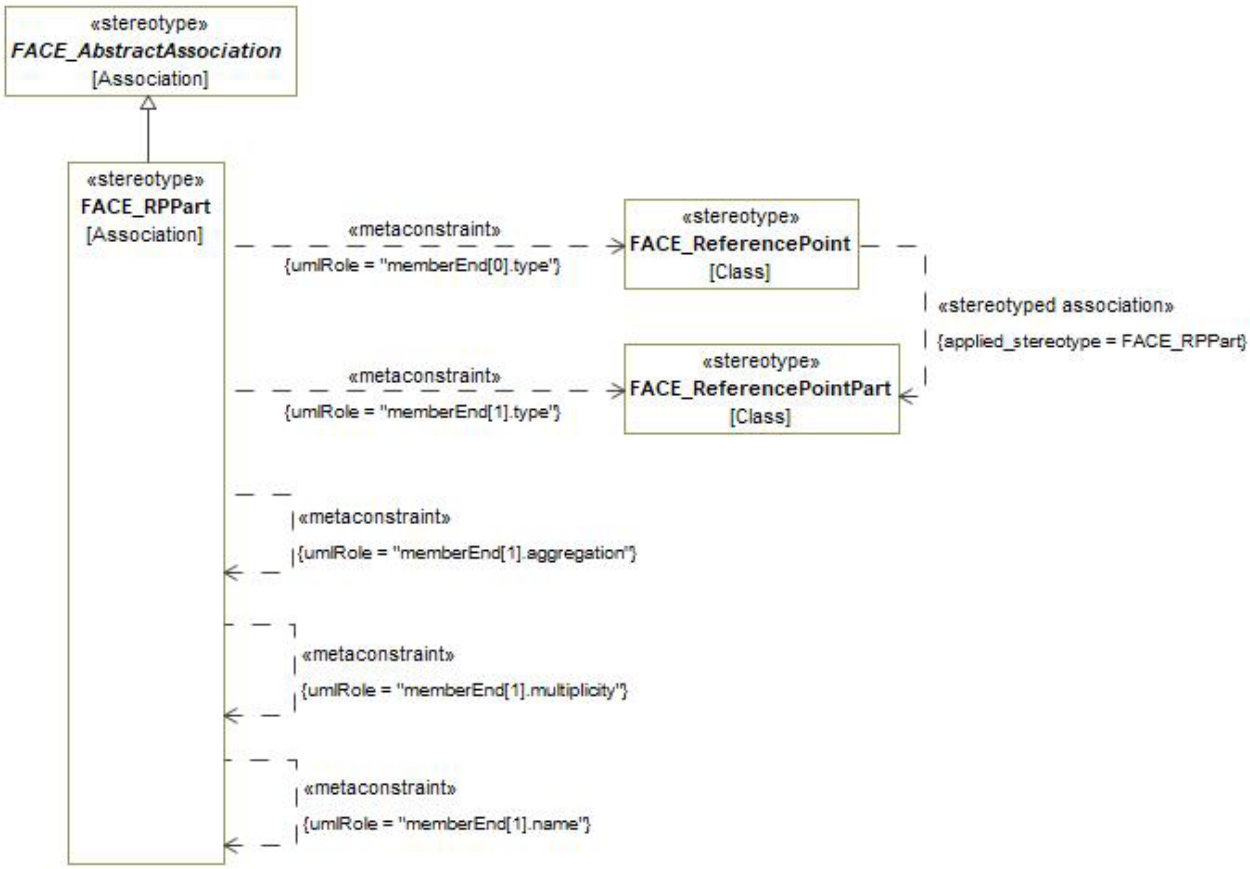

Figure 7-79: FACE_RPPart

Constraints

C01: FACE_RPPart.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_ReferencePoint».
C02: FACE_RPPart.memberEnd[1].aggregation	memberEnd[1].aggregation shall be composite
C03: FACE_RPPart.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 1..*
C04: FACE_RPPart.memberEnd[1].name	memberEnd[1].name shall be "referencePointPart"
C05: 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 measurement system without requiring the detailed modeling of the measurement system. The reference should be unambiguous and allows for full comprehension of the underlying measurement system.

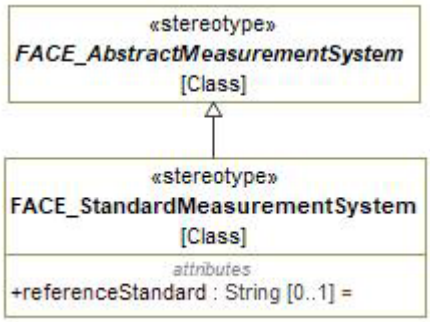

Figure 7-80: FACE_StandardMeasurementSystem

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.

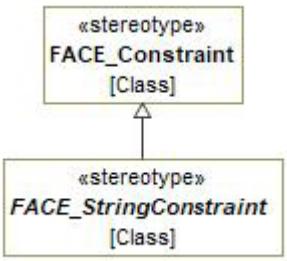

Figure 7-81: abstract FACE_StringConstraint

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.

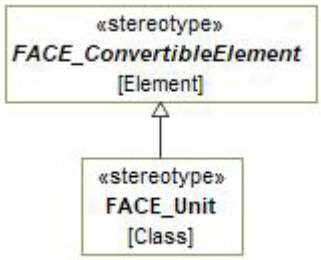

Figure 7-82: FACE_Unit

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 -

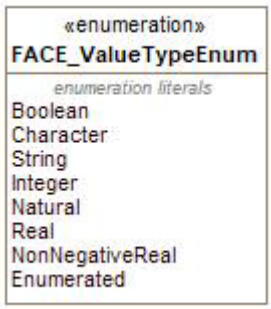

Figure 7-83: FACE_ValueTypeEnum

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.

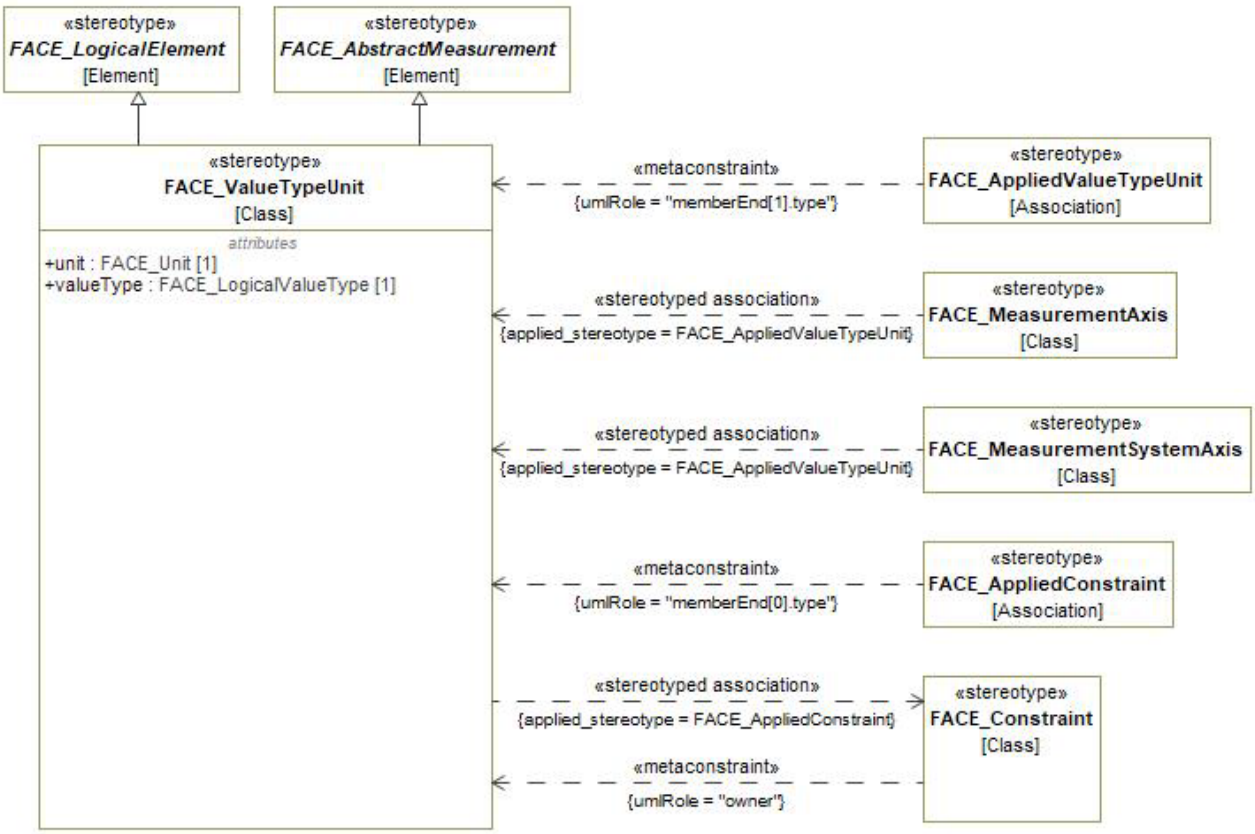

Figure 7-84: FACE_ValueTypeUnit

Attributes

unit : FACE_Unit [1]

valueType : FACE_LogicalValueType [1]

FACE Conformance/OCL Constraints

C01: 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_Profile::FACE Data Architecture::FACE Data Model::PlatformDataModel

The PlatformDataModel package of the FACE Profile contains elements that represent the Platform Data Model subpackage as specified in the UDDL metamodel.

FACE_Array

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Primitive

Description

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

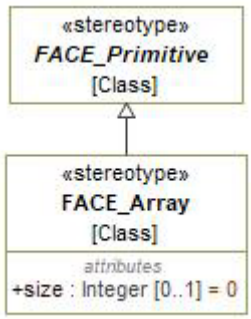

Figure 7-85: FACE_Array

Attributes

size : Integer [0..1]

FACE_Boolean

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Primitive

Description

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

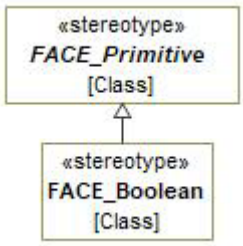

Figure 7-86: FACE_Boolean

FACE_BoundedString

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_StringType

Description

A 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.

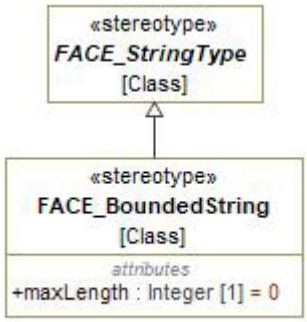

Figure 7-87: FACE_BoundedString

Attributes

maxLength : Integer [1]

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.

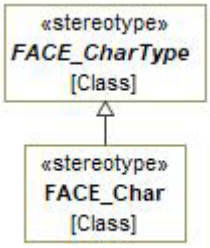

Figure 7-88: FACE_Char

FACE_CharArray

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_StringType

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.

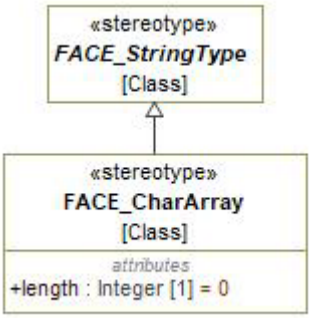

Figure 7-89: FACE_CharArray

Attributes

length : Integer [1]

FACE_CharType

Package: PlatformDataModel

isAbstract: Yes

Generalization: FACE_Primitive

Description

A FACE_CharType is a Char.

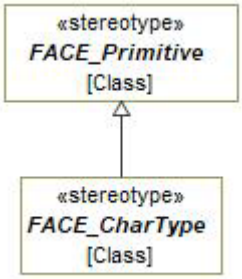

Figure 7-90: abstract FACE_CharType

FACE_Double

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Real

Description

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

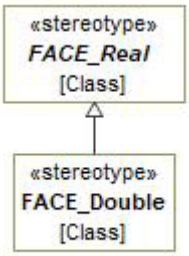

Figure 7-91: FACE_Double

FACE_Enumeration

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Primitive

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.

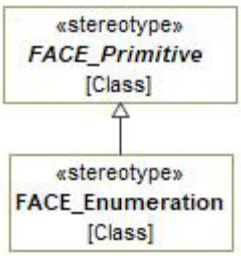

Figure 7-92: FACE_Enumeration

FACE_Fixed

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Real

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.

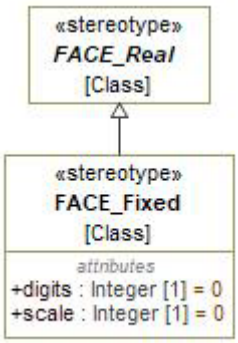

Figure 7-93: FACE_Fixed

Attributes

digits : Integer [1]

scale : Integer [1]

FACE_Float

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Real

Description

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

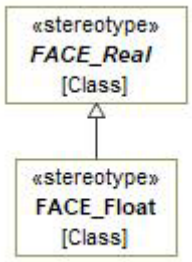

Figure 7-94: FACE_Float

FACE_Integer

Package: PlatformDataModel

isAbstract: Yes

Generalization: FACE_Number

Description

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

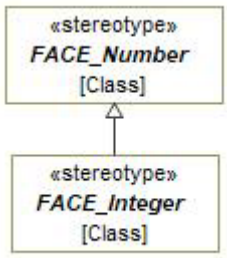

Figure 7-95: abstract FACE_Integer

FACE_Long

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Integer

Description

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

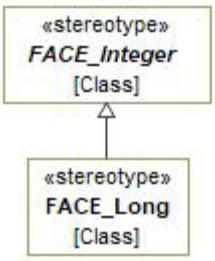

Figure 7-96: FACE_Long

FACE_LongDouble

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Real

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).

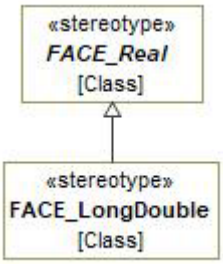

Figure 7-97: FACE_LongDouble

FACE_LongLong

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Integer

Description

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

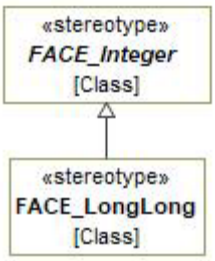

Figure 7-98: FACE_LongLong

FACE_Number

Package: PlatformDataModel

isAbstract: Yes

Generalization: FACE_Primitive

Description

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

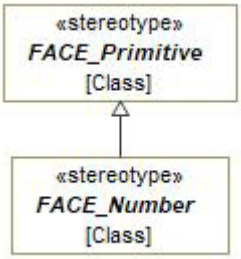

Figure 7-99: abstract FACE_Number

FACE_Octet

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Primitive

Description

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

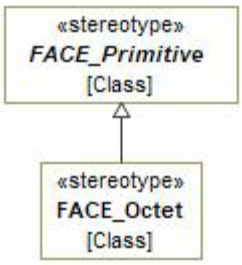

Figure 7-100: FACE_Octet

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.

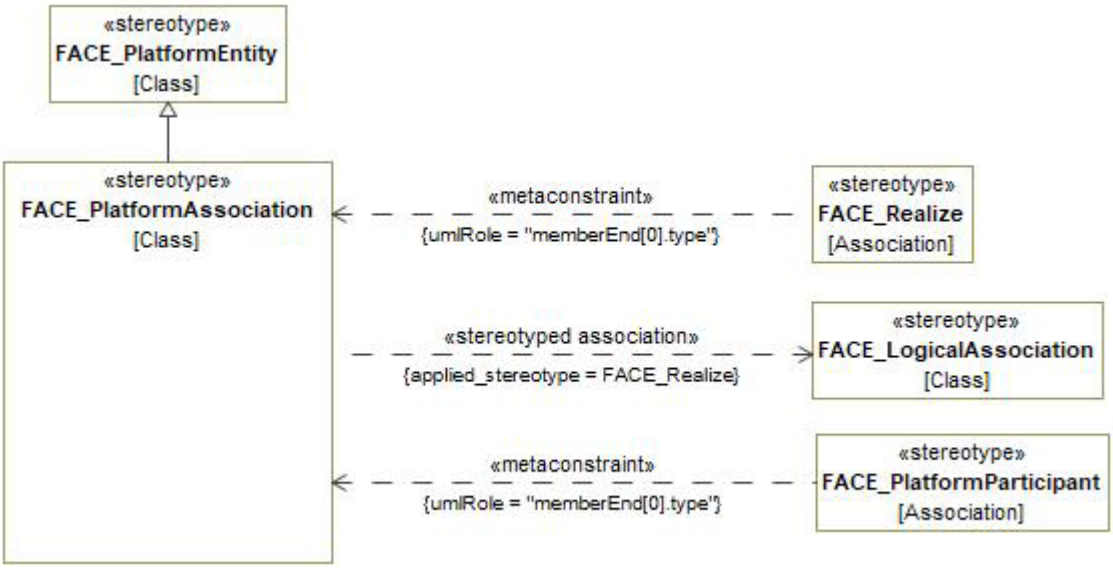

Figure 7-101: FACE_PlatformAssociation

FACE Conformance/OCL Constraints

C01: FACE_PlatformAssociation.participantsConsistentWithRealization	FACE_PlatformParticipants in a FACE_PlatformAssociation must realize FACE_LogicalParticipants in the FACE_LogicalAssociation that the FACE_PlatformAssociation realizes.
C02: 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 "name" metamodel attribute represents the FACE "rolename" attribute that defines the name of the platform Characteristic within the scope of the platform Entity. The "lowerBound" and "upperBound" attributes define the multiplicity of the composed Characteristic. An "upperBound" multiplicity of -1 represents an unbounded sequence.

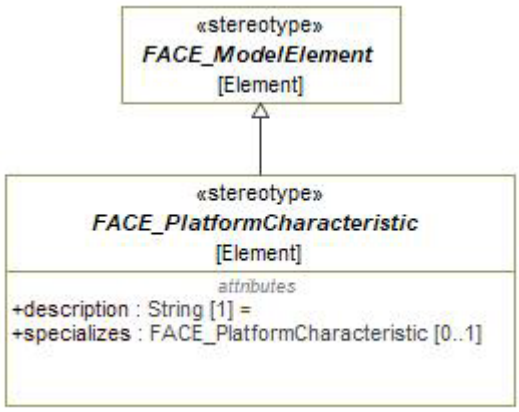

Figure 7-102: abstract FACE_PlatformCharacteristic

Attributes

description : String [1]

specializes : FACE_PlatformCharacteristic [0..1]

FACE Conformance/OCL Constraints

C01: FACE_PlatformCharacteristic.lowerBound_LTE_UpperBound	A FACE_PlatformCharacteristic's lowerBound must be less than or equal to its upperBound, unless its upperBound is -1.
C02: FACE_PlatformCharacteristic.rolenameIsNotReservedWord	The rolename of a FACE_PlatformCharacteristic must not be an IDL reserved word.
C03: FACE_PlatformCharacteristic.rolenameIsValidIdentifier	The rolename of a FACE_PlatformCharacteristic must be a valid identifier.
C04: FACE_PlatformCharacteristic.specializationConsistentWithRealization	If a FACE_PlatformCharacteristic specializes, its specialization must be consistent with its realization's specialization.
C05: 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.

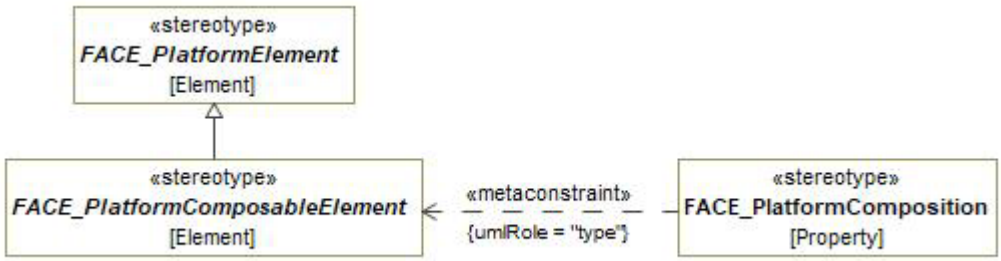

Figure 7-103: abstract FACE_PlatformComposableElement

FACE_PlatformCompositeQuery

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_PlatformView

Extension: Class

Description

A FACE_PlatformCompositeQuery is a collection of two or more platform Queries. The "isUnion" attribute specifies whether the composed platform Queries are intended to be represented as cases in an union or as members of a struct.

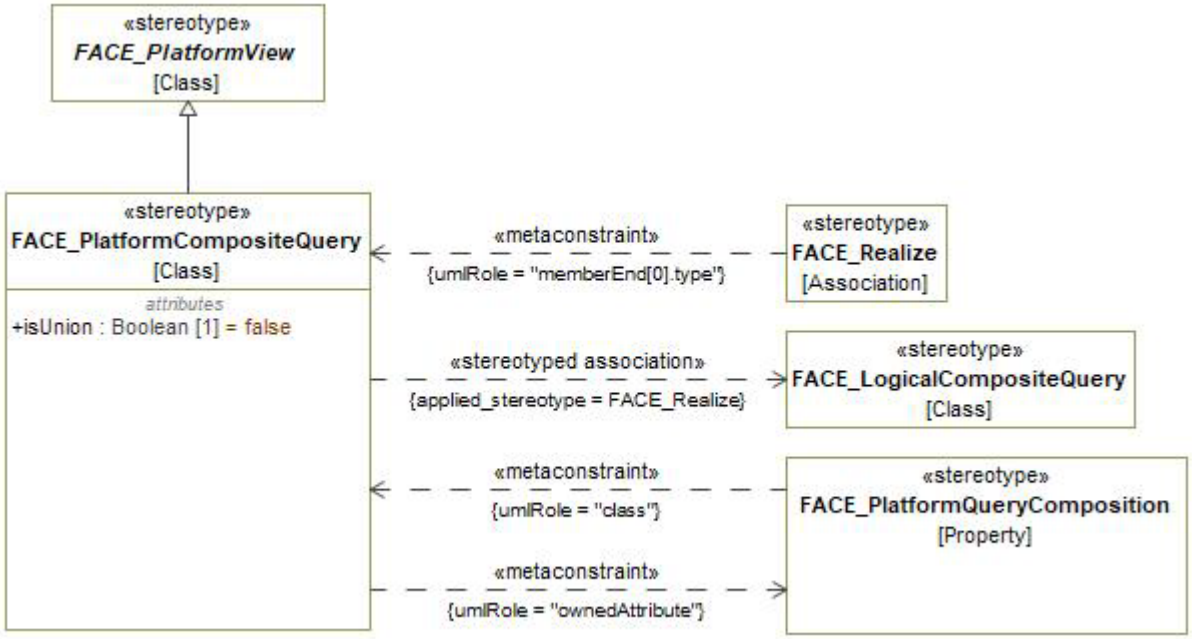

Figure 7-104: FACE_PlatformCompositeQuery

Attributes

isUnion : Boolean [1]

Constraints

C01: FACE_PlatformCompositeQuery.ownedAttribute	The values for the ownedAttribute metaproperty must meet the following criteria: - must be ordered list - referenced elements must be stereotyped «FACE_PlatformQueryComposition» or its specializations - must contain 2 or more elements

FACE Conformance/OCL Constraints

C01: FACE_PlatformCompositeQuery.compositionsConsistentWithRealization	FACE_PlatformQueryCompositions in a FACE_PlatformCompositeQuery must realize FACE_LogicalQueryCompositions in the FACE_LogicalCompositeQuery that the FACE_PlatformCompositeQuery realizes.
C02: FACE_PlatformCompositeQuery.compositionsHaveUniqueRolenames:	All contained rolenames must be unique within a FACE_PlatformCompositeQuery.
C03: FACE_PlatformCompositeQuery.noCyclesInConstruction	A FACE_PlatformCompositeQuery must not compose itself directly or indirectly.
C04: FACE_PlatformCompositeQuery.realizationUnionConsistent	A FACE_PlatformCompositeQuery that realizes must have the same "isUnion" property as the FACE_PlatformCompositeQuery it realizes.
C05: FACE_PlatformCompositeQuery.realizedCompositionsHaveDifferentTypes	A FACE_PlatformCompositeQuery must not contain two FACE_PlatformQueryCompositions that realize the same FACE_LogicalQueryComposition.
C06: FACE_PlatformCompositeQuery.viewComposedOnce	A FACE_PlatformCompositeQuery must not compose the same FACE_PlatformView more than once.

FACE_PlatformComposition

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_PlatformCharacteristic

Extension: Property

Description

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

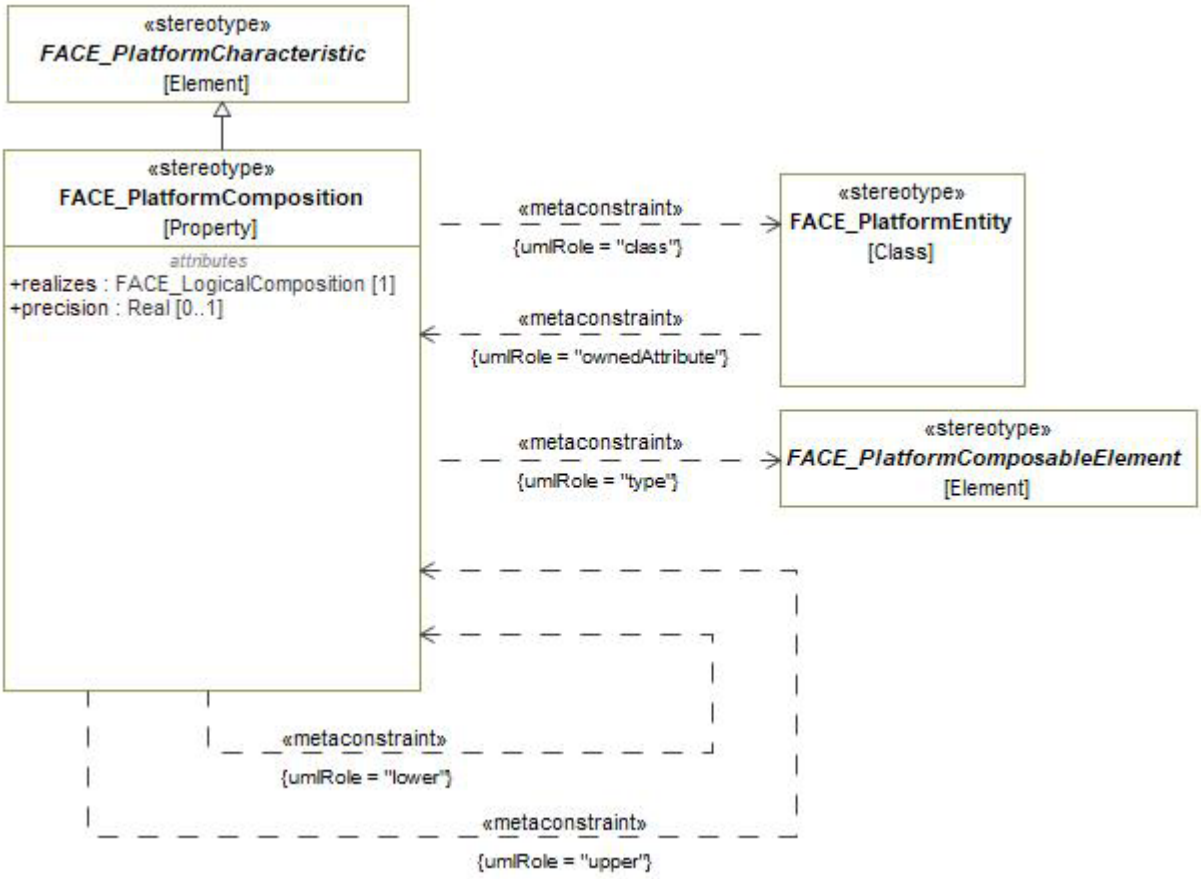

Figure 7-105: FACE_PlatformComposition

Attributes

precision : Real [0..1]

realizes : FACE_LogicalComposition [1]

Constraints

C01: FACE_PlatformComposition.class	Value for the class metaproperty must be stereotyped «FACE_PlatformEntity» or its specializations.
C02: FACE_PlatformComposition.multiplicity.lowerbound	The value for the multiplicity.lowerBound metaproperty must be an integer greater than or equal to -1.

C03: FACE_PlatformComposition.multiplicity.upperbound	The value for the multiplicity.upperBound metaproperty must be an integer greater than or equal to -1
C04: FACE_PlatformComposition.type	Value for the type metaproperty must be stereotyped «FACE_PlatformComposableElement» or its specializations.

FACE Conformance/OCL Constraints

C01: FACE_PlatformComposition.composedNumberHasPrecisionSet	A FACE_PlatformComposition whose type is a Number must have a precision greater than zero.
C02: FACE_PlatformComposition.multiplicityConsistentWithRealization	A FACE_PlatformComposition's multiplicity must be at least as restrictive as the FACE_LogicalComposition it realizes.
C03: FACE_PlatformComposition.multiplicityConsistentWithSpecialization	A FACE_PlatformComposition's multiplicity must be at least as restrictive as the FACE_PlatformComposition it specializes.
C04: FACE_PlatformComposition.typeConsistentWithRealization	A FACE_PlatformComposition's type must be consistent with its realization's type.

FACE_PlatformDataType

Package: PlatformDataModel

isAbstract: Yes

Generalization: FACE_PlatformComposableElement

Description

A FACE_PlatformDataType is a FACE_Primitive or a FACE_Struct.

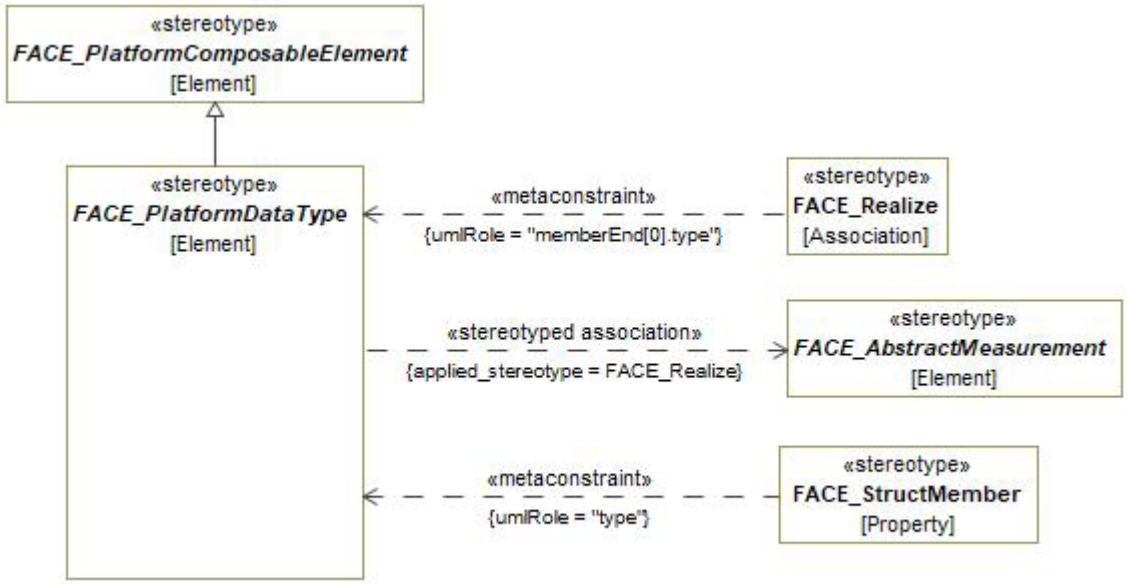

Figure 7-106: abstract FACE_PlatformDataType

FACE Conformance/OCL Constraints

C01: FACE_PlatformDataType.collectionRealizesStandardMeasurement	A FACE_Array or FACE_Sequence must realize a FACE_Measurement based on a FACE_StandardMeasurementSystem.
C02: FACE_PlatformDataType.platformDataTypeConsistentlyRealizesMeasurement	A FACE_Measurement must be realized by a FACE_Struct with one FACE_StructMember per FACE_MeasurementAxis. (Each FACE_StructMember'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_Primitive . - If a FACE_Measurement has one axis with multiple VTUs and no FACE_MeasurementAttributes, it is realized by a FACE_Struct with one FACE_StructMember for each VTU in the axis. (Each FACE_StructMember's type must realize a unique VTU in the axis; every VTU must be realized.) Each FACE_StructMember's type must be consistent with the type of the VTU it realizes.

C03: FACE_PlatformDataType.platformDataTypeConsistentlyRealizesMeasurementAxis	If a FACE_MeasurementAxis has one FACE_ValueTypeUnit (VTU), then it must be realized by a FACE_Primitive ; if it has multiple VTUs, then it must be realized by a FACE_Struct with one FACE_StructMember per VTU. If FACE_Struct "A" realizes FACE_MeasurementAxis "B", then A must have the same number of FACE_Compositions as B has VTUs, and every FACE_StructMember in A must realize a unique VTU in V.
C04: FACE_PlatformDataType.vtuRealizedByPrimitive	FACE_PlatformDataTypes that realize FACE_ValueTypeUnits are FACE_Primitives.

FACE_PlatformElement

Package: PlatformDataModel

isAbstract: Yes

Generalization: FACE_DataModelElement

Description

A FACE_PlatformElement is the root type for defining the platform-level elements of the FACE Data Model Language.

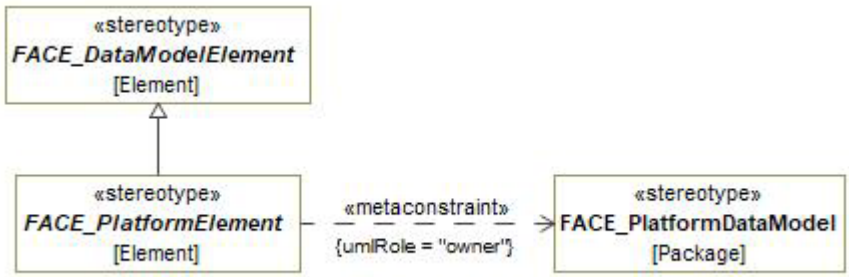

Figure 7-107: abstract FACE_PlatformElement

Constraints

C01: 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

C01: FACE_PlatformElement.hasUniqueName	Each FACE_PlatformElement must have a unique name.

C02: 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_SpecializationOwner

Extension: Class

Description

A FACE_PlatformEntity "realizes" a FACE_LogicalEntity in terms of FACE_PlatformDataTypes and other FACE_PlatformEntities composed of FACE_PlatformDataTypes. 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_PlatformDataTypes realize many to one the FACE_LogicalEntity's composed FACE_Measurements.

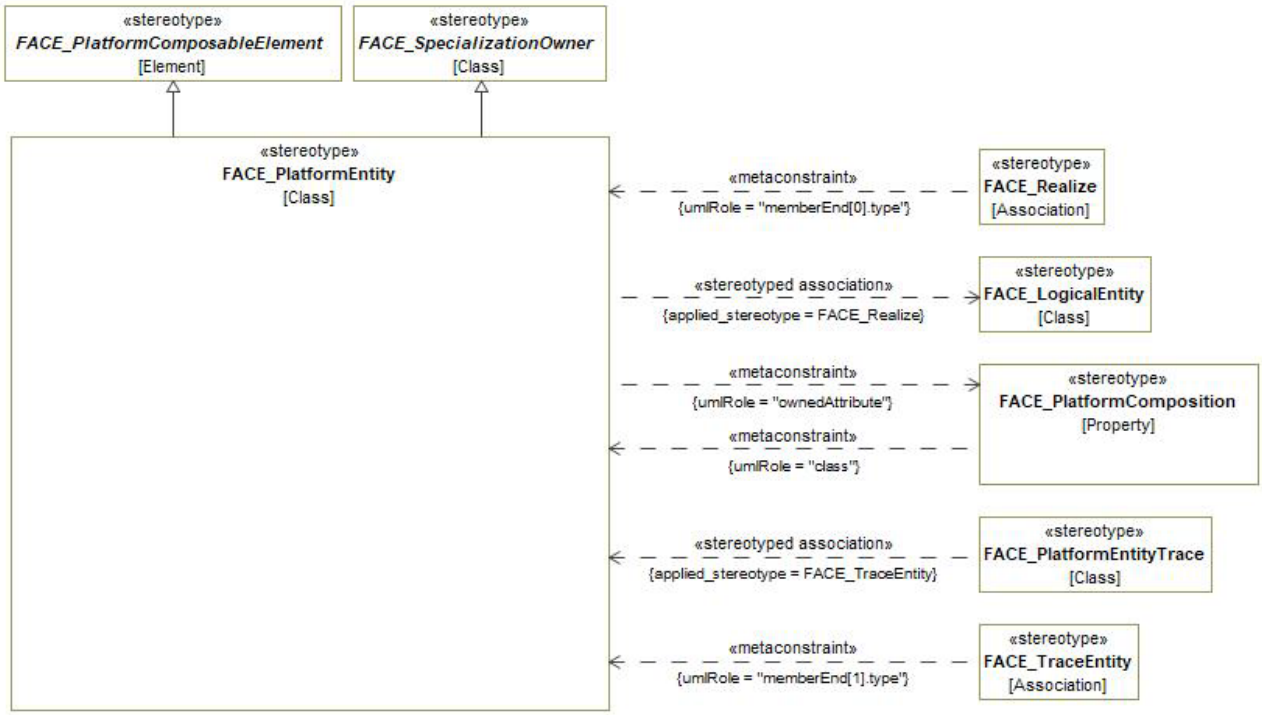

Figure 7-108: FACE_PlatformEntity

Constraints

C01: FACE_PlatformEntity.ownedAttribute	The value for the ownedAttribute metaproperty must be stereotyped «FACE_PlatformComposition» or its specializations

FACE Conformance/OCL Constraints

C01: FACE_PlatformEntity.characteristicsHaveUniqueRolenames	A FACE_PlatformCharacteristic's rolename must be unique within a FACE_PlatformEntity.
C02: FACE_PlatformEntity.compositionsConsistentWithRealization	FACE_PlatformCompositions in a FACE_PlatformEntity must realize FACE_LogicalCompositions in the FACE_LogicalEntity that the FACE_PlatformEntity realizes.
C03: 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.
C04: FACE_PlatformEntity.realizedCompositionsHaveDifferentTypes	A FACE_PlatformEntity may not contain two FACE_PlatformCompositions that realize the same lFACE_LogicalCompositions unless their types are different PlatformDataTypes and their multiplicities are equal.
C05: 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: Association

Description

A FACE_PlatformParticipant is the mechanism that allows a FACE_PlatformAssociation to be constructed between two or more FACE_PlatformEntities. The "type" (target of the directional Association) of a platform Participant is the platform Entity being used to construct the platform Association. Target multiplicity values represent the "sourceLowerBound" and "sourceUpperBound" attributes that define the multiplicity of the platform Association relative to the Participant in the UDDL metamodel. An upper multiplicity of star (*) on the target of the association is the equivalent of a "sourceUpperBound" multiplicity of -1 (which represents an unbounded sequence) in the the UDDL metamodel. 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.

FACE_PlatformParticipant Associations are directional, from a FACE_PlatformAssociation to a FACE_PlatformEntity.

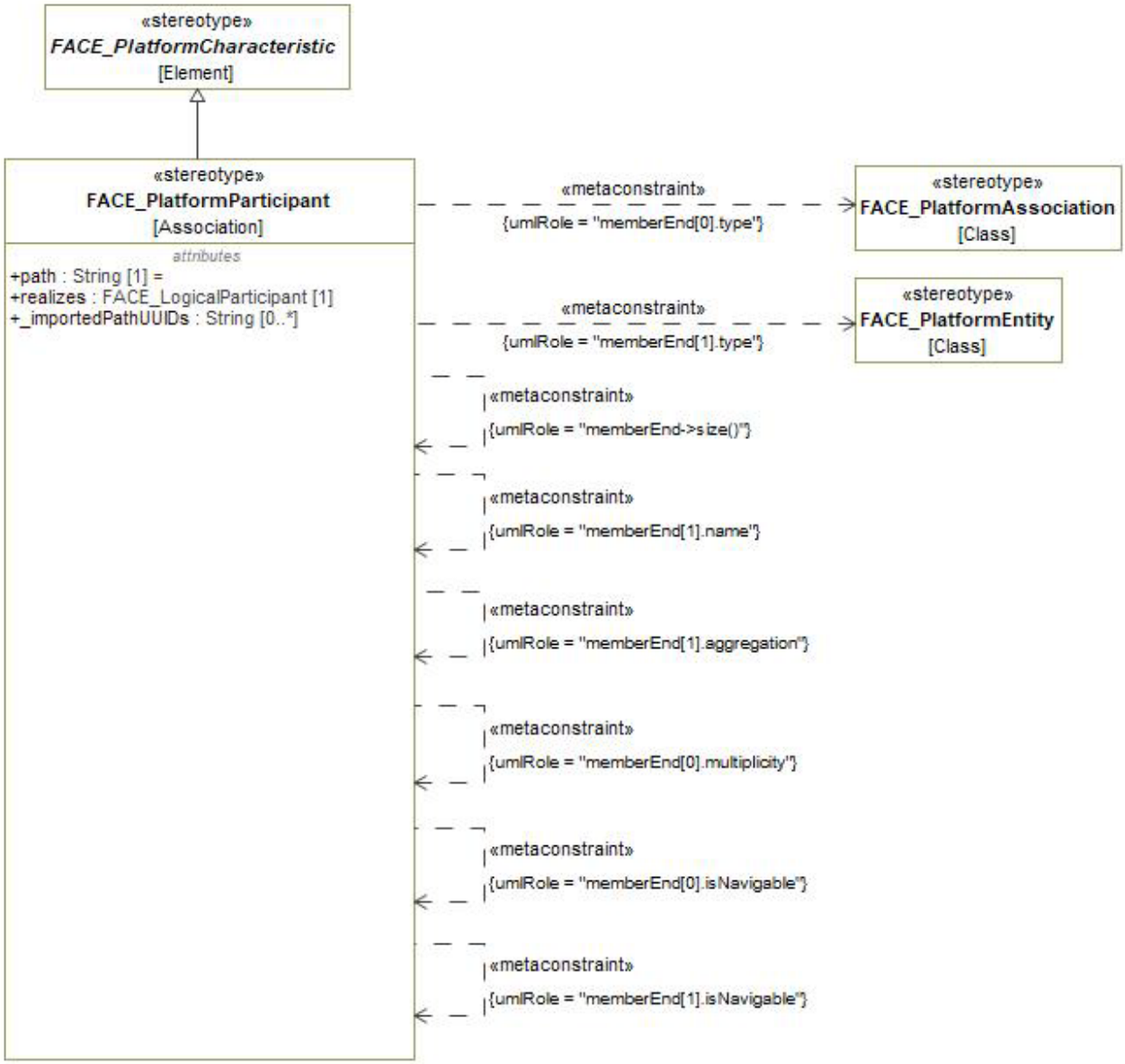

Figure 7-109: FACE_PlatformParticipant

Attributes

path : String [1]	The "path" property indicates the portion of the target «FACE_PlatformEntity» that is participating in the «FACE_PlatformAssociation» that is the source for the «FACE_PlatformParticipant» Association. Path strings reference Entities or Characteristics (properties of Entities). Where the path string references an Entity, it is considered to be a ParticipantPathNode. Where the path string references a Characteristic of an Entity, it is considered to be a CharacteristicPathNode.

	The UDDL metamodel defines PathNode, ParticipantPathNode and CharacteristicPathNode as follows: A platform PathNode is a single element in a chain that collectively forms a path specification. A platform ParticipantPathNode is a platform PathNode that selects a Participant that references an Entity. This provides a mechanism for reverse navigation from an Entity that participates in an Association back to the Association. A platform CharacteristicPathNode is a platform PathNode that selects a platform Characteristic which is directly contained in a platform Entity or Association. The strings provided in the "path" tagged value are a representation of the full set of Platform CharacteristicPathNode, ParticipantPathNode, and PathNode elements in the path attribute as specified in the UDDL Standard. 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 and the FACE XMI (face) file are required to translate between the two notations.
realizes : FACE_LogicalParticipant [1]	
_importedPathUUIDs : String [0..*]	This tag is for use by import/export plug-ins in two-way translation of FACE 3.x paths to and from FACE 2.1 path strings. It is used to preserve the UUIDs of the paths imported from FACE 3.x paths when they are translated into FACE 2.1 path strings, so that they can be reconstituted for subsequent export as FACE 3.x elements. Because this tag is used exclusively by the plug-ins, its implementation is optional if a tool either does not import/export FACE format files or the tool uses an alternate means of representing and translating FACE Paths.

Constraints

C01: FACE_PlatformParticipant.memberEnd->size()	memberEnd.size() shall be 2
C02: FACE_PlatformParticipant.memberEnd[0].isNavigable	memberEnd[0].isNavigable shall be false
C03: FACE_PlatformParticipant.memberEnd[0].multiplicity	memberEnd[0].multiplicity shall be 1
C04: FACE_PlatformParticipant.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_PlatformAssociation»

C05: FACE_PlatformParticipant.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C06: FACE_PlatformParticipant.memberEnd[1].isNavigable	memberEnd[1].isNavigable shall be true
C07: FACE_PlatformParticipant.memberEnd[1].name	The memberEnd[1].name metaproperty must be an non-empty alphanumeric name string
C08: FACE_PlatformParticipant.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_PlatformEntity»

FACE Conformance/OCL Constraints

C01: FACE_PlatformParticipant.multiplicityConsistentWithRealization	A FACE_PlatformParticipant's multiplicity must be at least as restrictive as the FACE_LogicalParticipant it realizes.
C02: FACE_PlatformParticipant.multiplicityConsistentWithSpecialization	A FACE_PlatformParticipant's multiplicity must be at least as restrictive as the FACE_PlatformParticipant it specializes.
C03: FACE_PlatformParticipant.rolenameDefined	A FACE_PlatformParticipant must have a rolename, either projected from a characteristic or defined directly on the FACE_PlatformParticipant.
C04: 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_PlatformView

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 data model Query grammar.

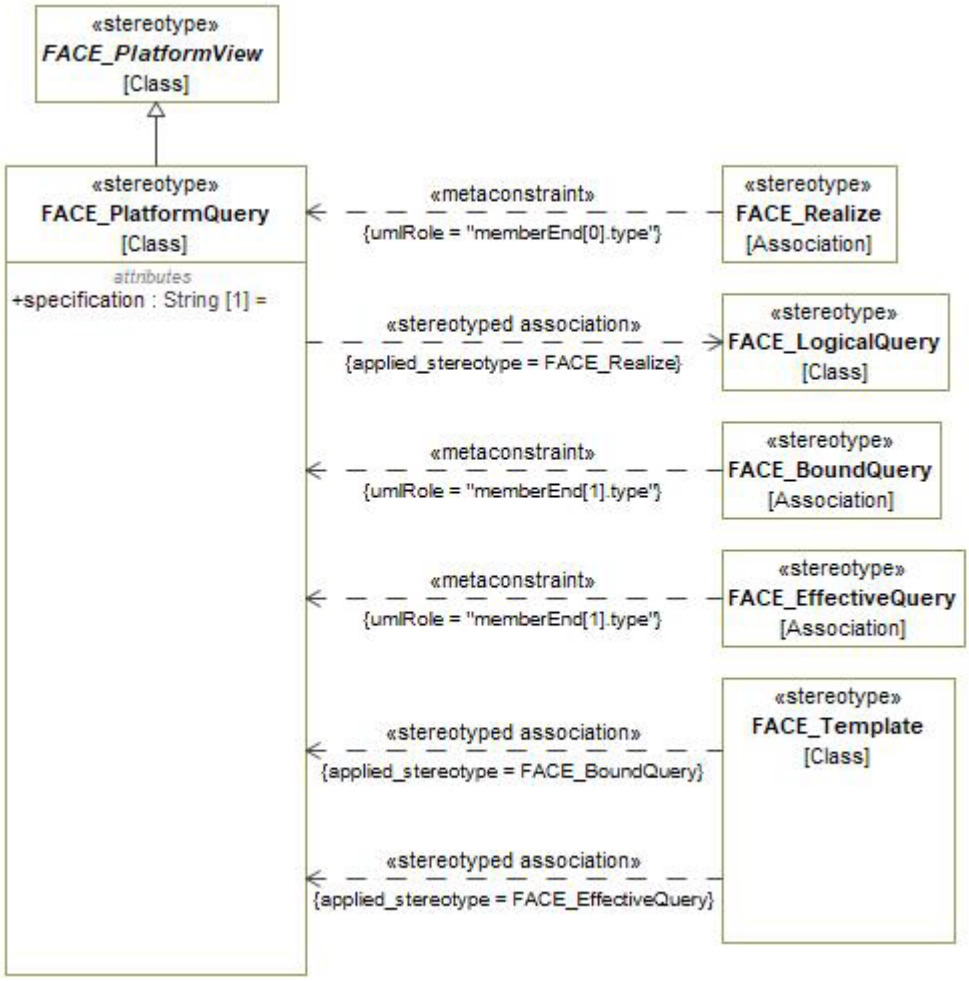

Figure 7-110: FACE_PlatformQuery

Attributes

specification : String [1]

FACE_PlatformQueryComposition

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_ModelElement

Extension: Property

Description

A FACE_PlatformQueryComposition is the mechanism that allows a FACE_PlatformCompositeQuery to be constructed from FACE_PlatformQueries and other FACE_PlatformCompositeQueries. The "rolename" attribute defines the name of the composed platform View within the scope of the composing platform CompositeQuery. The "type" of a

FACE_PlatformQueryComposition is the FACE_PlatformView being used to construct the FACE_PlatformCompositeQuery.

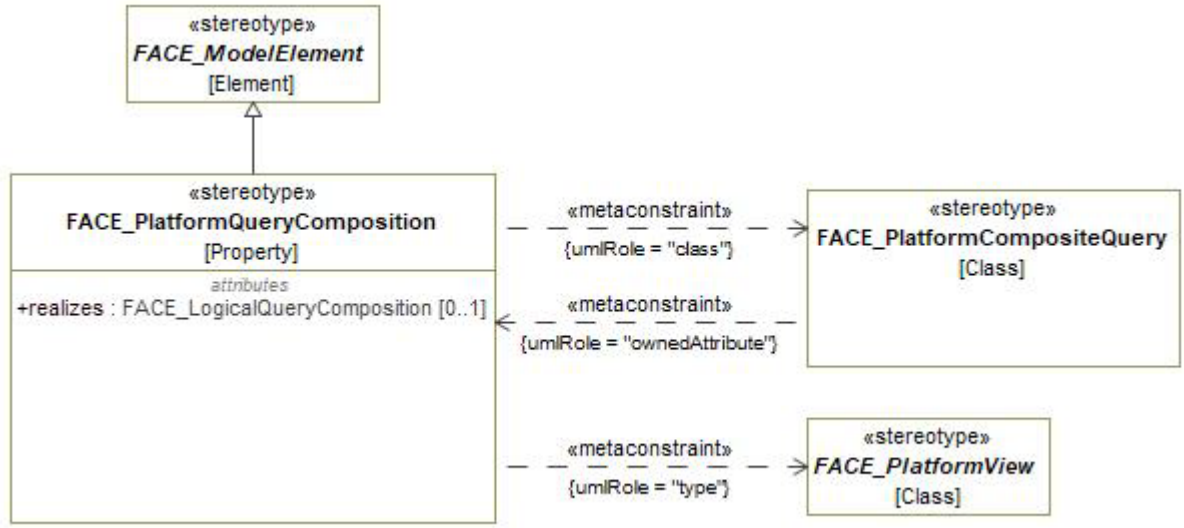

Figure 7-111: FACE_PlatformQueryComposition

Attributes

realizes : FACE_LogicalQueryComposition [0..1]

Constraints

C01: FACE_PlatformQueryComposition.class	Value for class metaproperty must be stereotyped «FACE_PlatformCompositeQuery».
C02: FACE_PlatformQueryComposition.type	Value for type metaproperty must be stereotyped «FACE_PlatformView» or its specializations.

FACE Conformance/OCL Constraints

C01: FACE_PlatformQueryComposition.rolenameIsValidIdentifier	The rolename of a FACE_PlatformQueryComposition must be a valid identifier.
C02: FACE_PlatformQueryComposition.typeConsistentWithRealization	If FACE_PlatformQueryComposition "A" realizes FACE_LogicalQueryComposition "B", then A's type must realize B's type.

FACE_PlatformView

Package: PlatformDataModel

isAbstract: Yes

Generalization: FACE_PlatformElement

Extension: Class

Description

A FACE_PlatformView is a platform Query or a platform CompositeQuery.

Figure 7-112: abstract FACE_PlatformView

FACE_Primitive

Package: PlatformDataModel

isAbstract: Yes

Generalization: FACE_PlatformDataType

Extension: Class

Description

A FACE_Primitive is a platform realization of a logical FACE_AbstractMeasurement,, and represented as a primitive data type (e.g. Boolean, Char, Float, Double…).

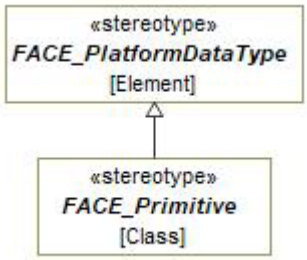

Figure 7-113: abstract FACE_Primitive

FACE_Real

Package: PlatformDataModel

isAbstract: Yes

Generalization: FACE_Number

Description

A FACE_Real is an abstract meta-class from which all meta-classes representing real / floating-point numbers derive.

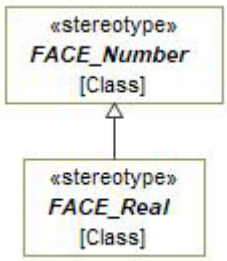

Figure 7-114: abstract FACE_Real

FACE_Sequence

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Primitive

Description

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

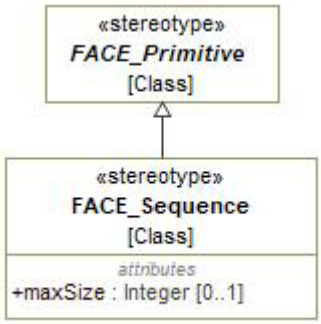

Figure 7-115: FACE_Sequence

Attributes

maxSize : Integer [0..1]

FACE_Short

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_Integer

Description

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

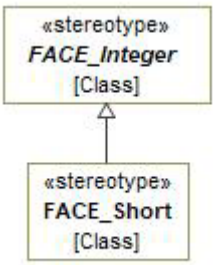

Figure 7-116: FACE_Short

FACE_String

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_StringType

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.

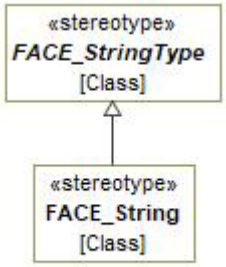

Figure 7-117: FACE_String

FACE_StringType

Package: PlatformDataModel

isAbstract: Yes

Generalization: FACE_Primitive

Description

A FACE_StringType is a representation for CharArray, BoundedString, or String.

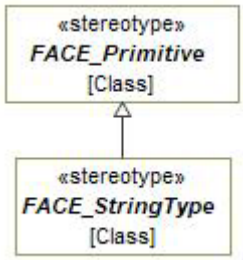

Figure 7-118: abstract FACE_StringType

FACE_Struct

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_PlatformDataType

Extension: Class

Description

A platform FACE_Struct "realizes" a logical FACE_AbstractMeasurement in terms of FACE_Primitives and other FACE_Structs composed of FACE_Primitives. A platform FACE_Struct's composition hierarchy is consistent with the composition hierarchy of the logical AbstractMeasurement that it realizes. Each composed platform FACE_PlatformDataType realizes a logical FACE_AbstractMeasurement.

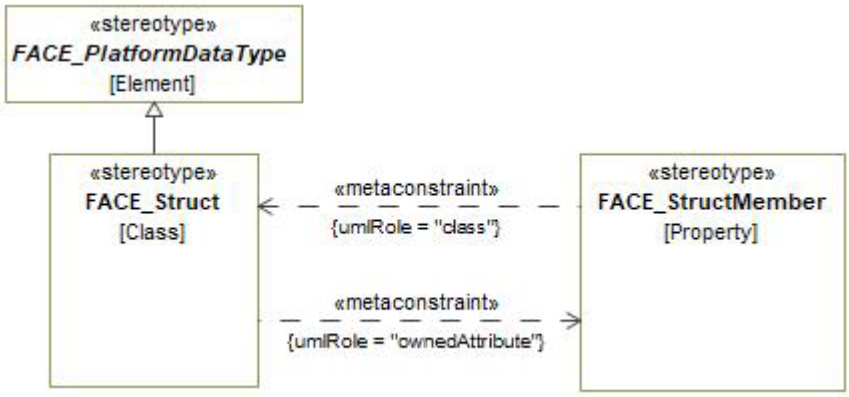

Figure 7-119: FACE_Struct

Constraints

C01: FACE_Struct.ownedAttribute	The values for the ownedAttribute metaproperty must meet the following criteria: - referenced elements must be stereotyped «FACE_StructMember» - must contain 2 or more elements

FACE Conformance/OCL Constraints

C01: FACE_Struct.structMembersConsistentlyRealizeMeasurementAttributes	A FACE_Measurement with FACE_MeasurementAttributes is realized by a FACE_Struct with one FACE_StructMember per FACE_MeasurementAttribute. (Each FACE_StructMember (that realizes) must realize a unique attribute in the FACE_Measurement; every attribute must be realized.)

FACE_StructMember

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_ModelElement

Extension: Property

Description

A FACE_StructMember is the mechanism that allows FACE_Structs to be constructed from other FACE_PlatformDataTypes. The "type" property of a FACE_StructMember is the FACE_PlatformDataType being used to construct the FACE_StructMember. If "type" is a FACE_Primitive, the precision attribute specifies a measure of the detail in which a quantity is captured.

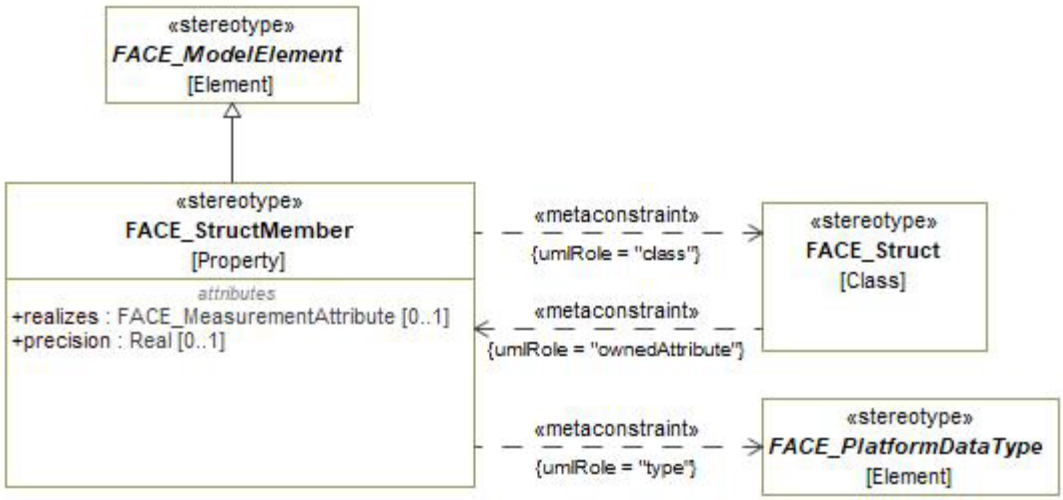

Figure 7-120: FACE_StructMember

Attributes

precision : Real [0..1]

realizes : FACE_MeasurementAttribute [0..1]

Constraints

C01: FACE_StructMember.class	Value for the class metaproperty must be stereotyped «FACE_Struct»
C02: FACE_StructMember.type	Value for the type metaproperty must be stereotyped by a specialization of «FACE_PlatformDataType».

FACE Conformance/OCL Constraints

C01: FACE_StructMember.composedNumberHasPrecisionSet	A FACE_StructMember whose type is a Number must have a precision greater than zero.
C02: FACE_StructMember.typeConsistentWithRealization	If a FACE_StructMember realizes a FACE_MeasurementAttribute, then the FACE_StructMember's type must be consistent with its realization's type.

FACE_ULong

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_UnsignedInteger

Description

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

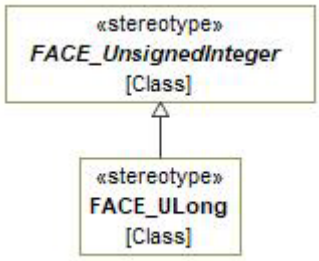

Figure 7-121: FACE_ULong

FACE_ULongLong

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_UnsignedInteger

Description

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

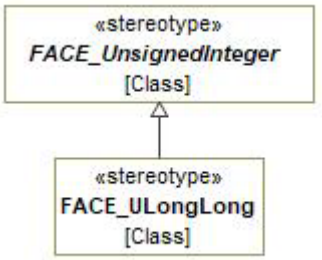

Figure 7-122: FACE_ULongLong

FACE_UnsignedInteger

Package: PlatformDataModel

isAbstract: Yes

Generalization: FACE_Integer

Description

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

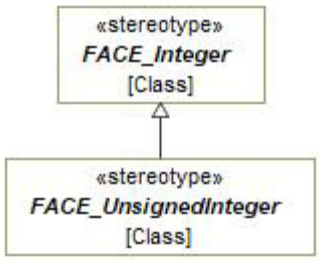

Figure 7-123: abstract FACE_UnsignedInteger

FACE_UShort

Package: PlatformDataModel

isAbstract: No

Generalization: FACE_UnsignedInteger

Description

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

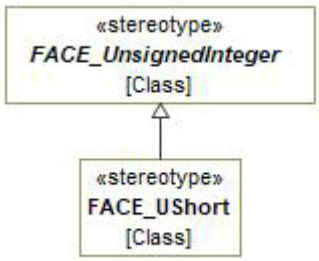

Figure 7-124: FACE_UShort

7.1.1.2 FACE_Profile::FACE Data Architecture::Integration Model

The Integration Model package of the FACE Profile contains elements that represent the Integration Model subpackage as specified in the FACE metamodel.

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).

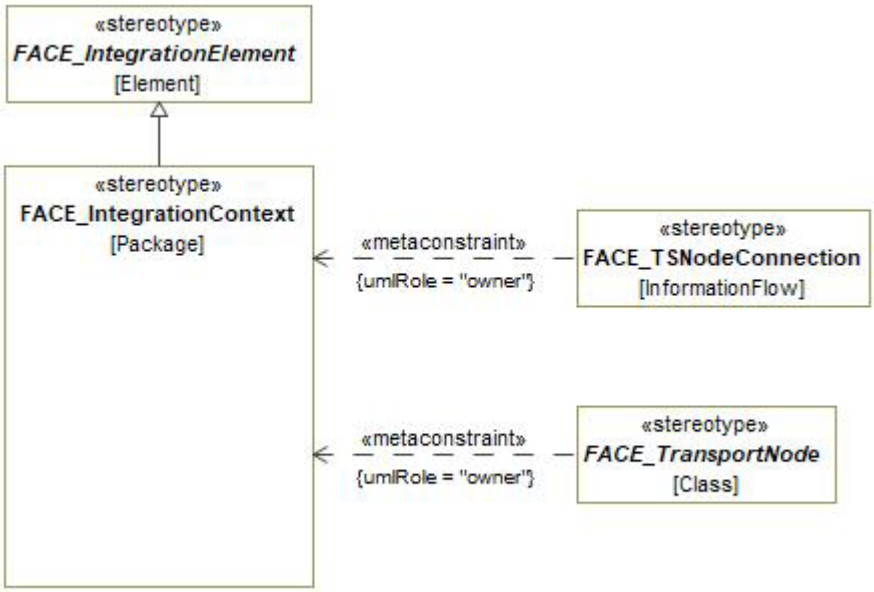

Figure 7-125: FACE_IntegrationContext

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.

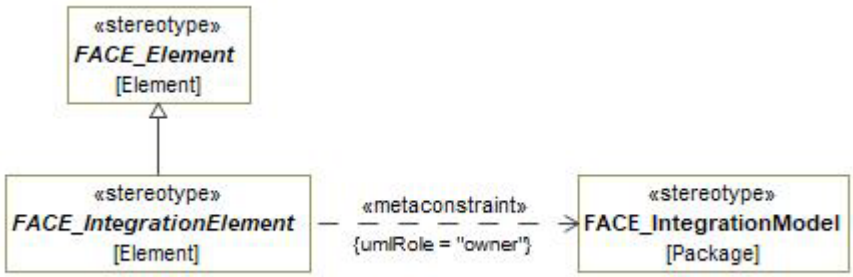

Figure 7-126: abstract FACE_IntegrationElement

Constraints

C01: 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

C01: 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.

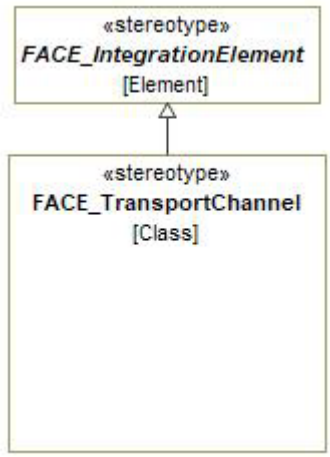

Figure 7-127: FACE_TransportChannel

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.

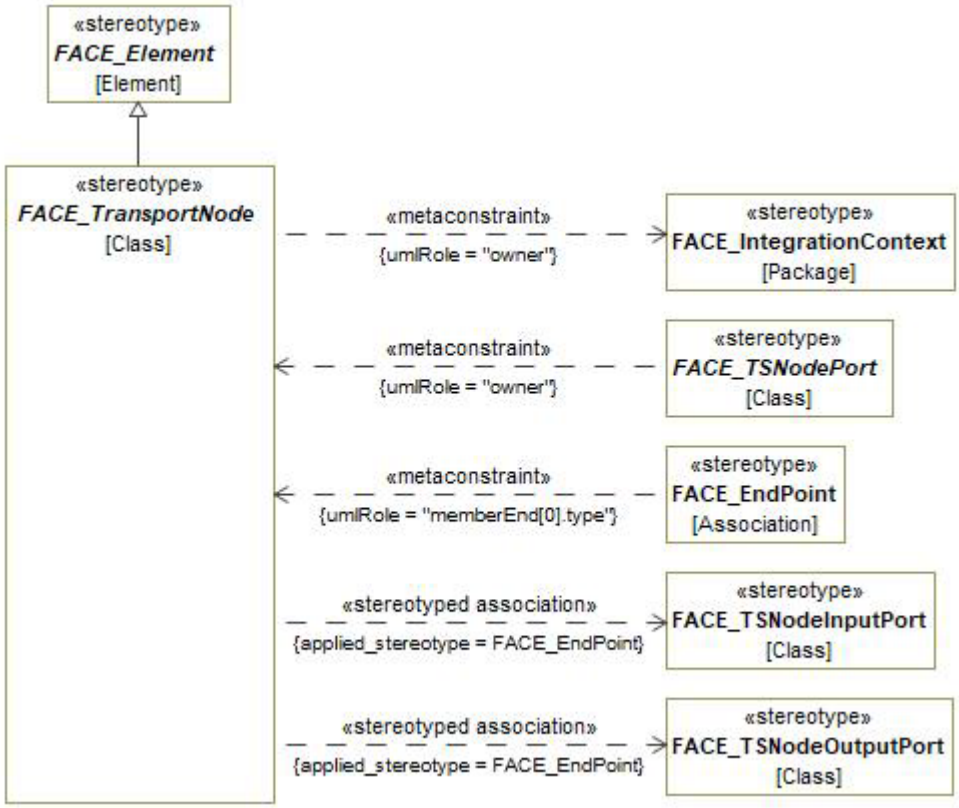

Figure 7-128: abstract FACE_TransportNode

Constraints

C01: 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

C01: FACE_TransportNode.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.
C02: 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.
C03: 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.

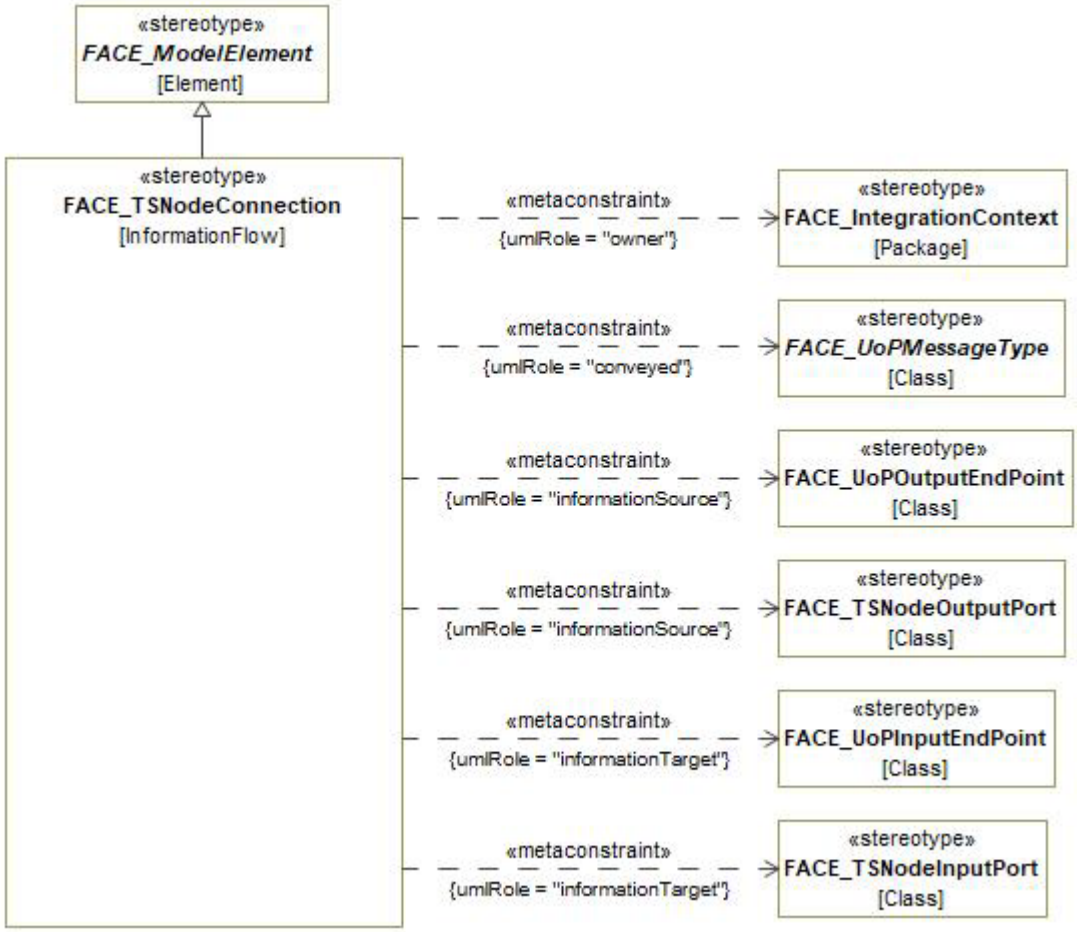

Figure 7-129: FACE_TSNodeConnection

Constraints

C01: FACE_TSNodeConnection.conveyed	Value for the conveyed metaproperty must be stereotyped by a specialization of «FACE_MessageType».
C02: FACE_TSNodeConnection.informationSource	The value for the informationSource metaproperty must be stereotyped by one of the following: «FACE_UoPOutputEndPoint» «FACE_TSNodeOutputPort»

C03: FACE_TSNodeConnection.informationTarget	The value for the informationTarget metaproperty must be stereotyped by one of the following: «FACE_UoPInputEndPoint» «FACE_TSNodeInputPort»
C04: FACE_TSNodeConnection.owner	Elements with this stereotype may only be contained in (owned by) elements stereotyped by «FACE_IntegrationContext»

FACE Conformance/OCL Constraints

C01: FACE_TSNodeConnection.connectWithinSameContext	A FACE_TSNodeConnection may connect only FACE_TransportNodes that are in the same FACE_IntegrationContext as the FACE_TSNodeConnection.
C02: FACE_TSNodeConnection.destinationIsInput	A FACE_TSNodeConnection's destination must be an input.
C03: FACE_TSNodeConnection.sourceIsOutput	A FACE_TSNodeConnection's source must be an output
C04: FACE_TSNodeConnection.sourceViewMatchesDestinationView	A FACE_TSNodeConnection must use the same View on its source and destination.
C05: 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.

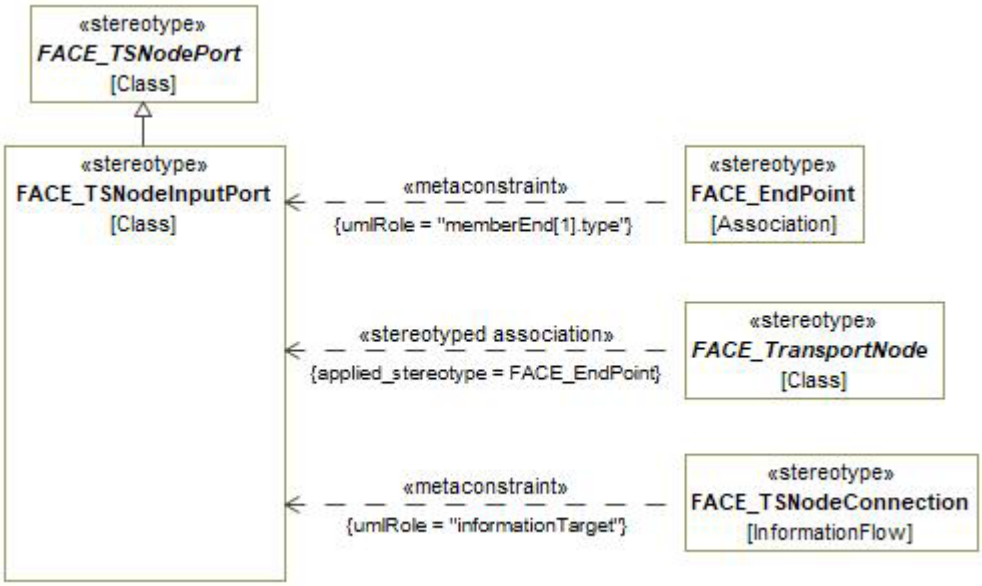

Figure 7-130: FACE_TSNodeInputPort

FACE Conformance/OCL Constraints

C01: 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.

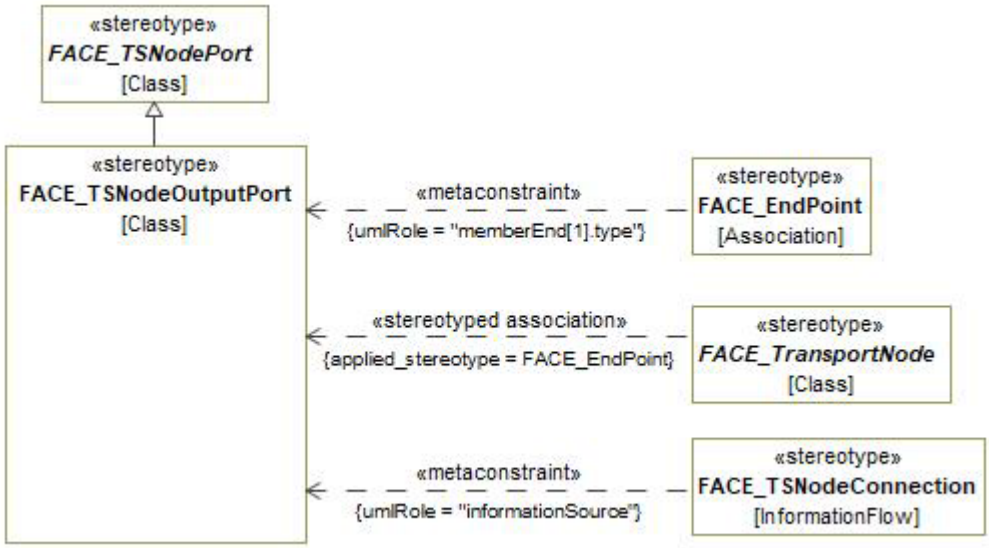

Figure 7-131: FACE_TSNodeOutputPort

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. The type property of a FACE_TSNodePort is the FACE_UoPMessageType it references.

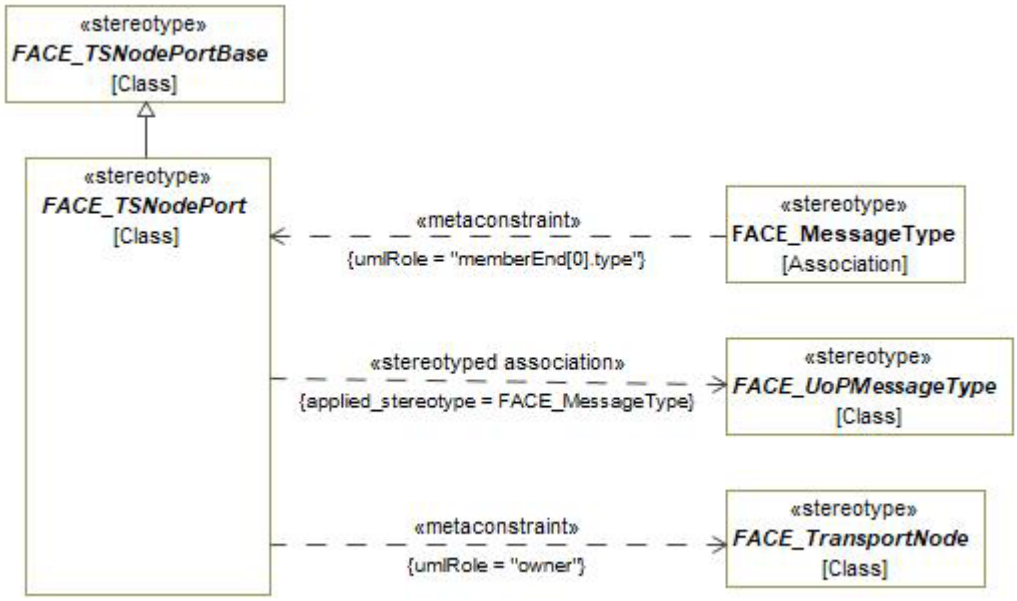

Figure 7-132: abstract FACE_TSNodePort

Constraints

C01: 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.

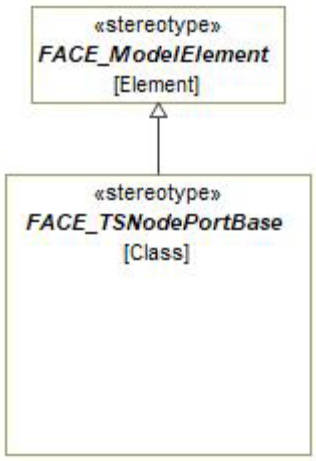

Figure 7-133: abstract FACE_TSNodePortBase

FACE Conformance/OCL Constraints

C01: 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.

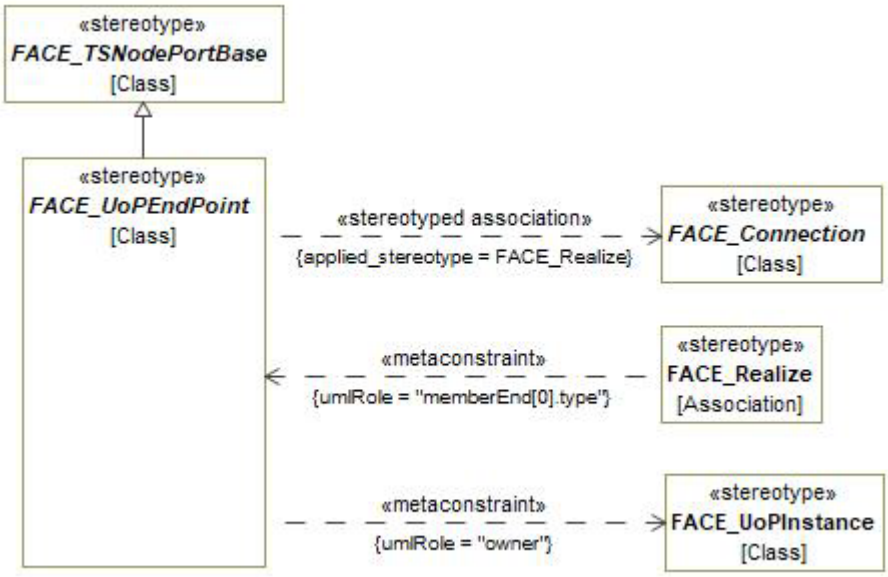

Figure 7-134: abstract FACE_UoPEndPoint

Constraints

C01: 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).

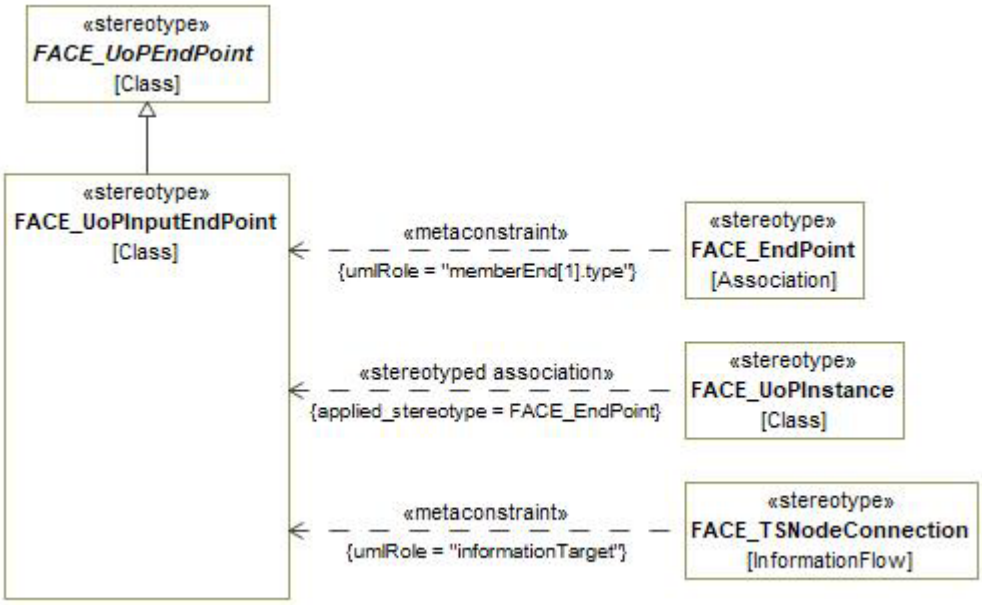

Figure 7-135: FACE_UoPInputEndPoint

FACE Conformance/OCL Constraints

C01: FACE_UoPInputEndPoint.onlyOneConnection	A FACE_UoPInputEndPoint's may be the destination of at most one TSNodeConnection.
C02: 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.

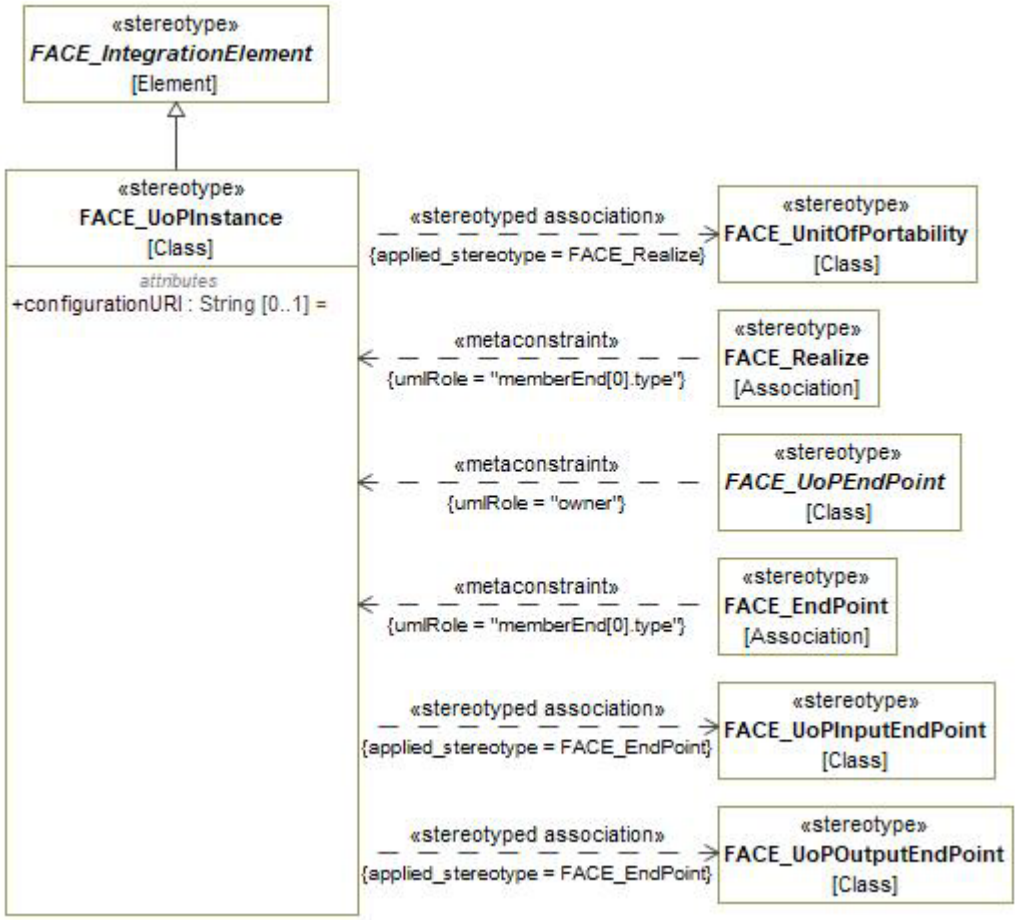

Figure 7-136: FACE_UoPInstance

Attributes

configurationURI : String [0..1]

FACE Conformance/OCL Constraints

C01: 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).

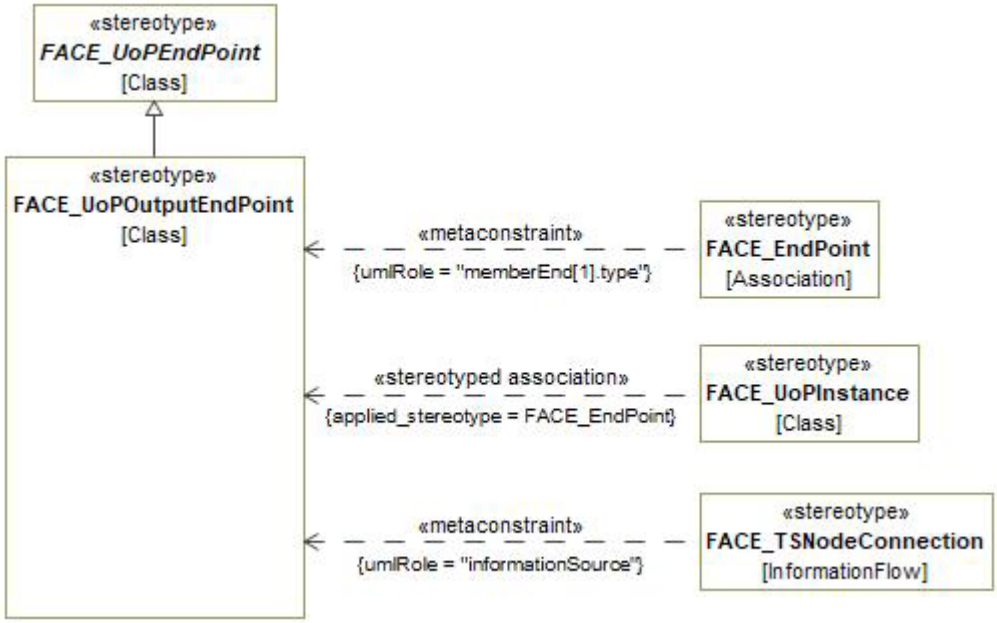

Figure 7-137: FACE_UoPOutputEndPoint

FACE Conformance/OCL Constraints

C01: 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.

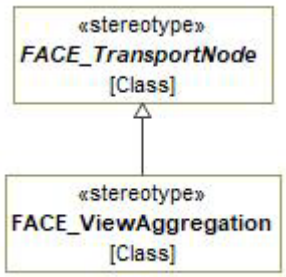

Figure 7-138: FACE_ViewAggregation

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.

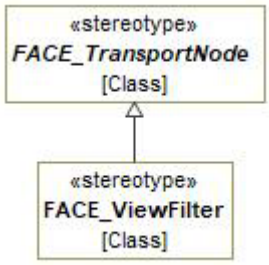

Figure 7-139: FACE_ViewFilter

FACE Conformance/OCL Constraints

C01: 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.

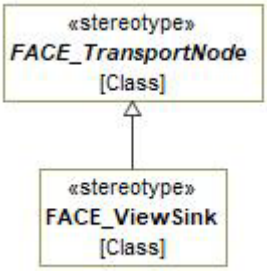

Figure 7-140: FACE_ViewSink

FACE Conformance/OCL Constraints

C01: 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.

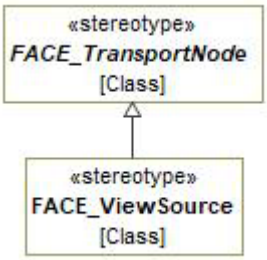

Figure 7-141: FACE_ViewSource

FACE Conformance/OCL Constraints

C01: 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.

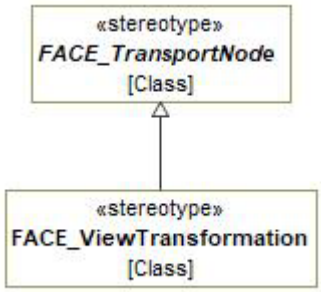

Figure 7-142: FACE_ViewTransformation

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.

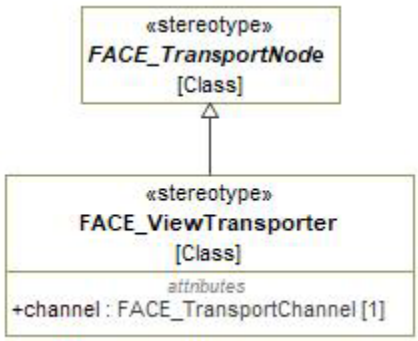

Figure 7-143: FACE_ViewTransporter

Attributes

channel : FACE_TransportChannel [1]

FACE Conformance/OCL Constraints

C01: FACE_ViewTransporter.viewIsConsistent	A FACE_ViewTransporter must use the same FACE_PlatformView on its input and output.

7.1.1.3 FACE_Profile::FACE Data Architecture::Traceability Model

The Traceability Model package of the FACE Profile contains elements that represent the Traceability Model subpackage as specified in the FACE metamodel.

FACE_ConceptualEntityTrace

Package: Traceability Model

isAbstract: No

Generalization: FACE_TraceableElement, FACE_TraceabilityElement

Extension: Class

Description

Because the Data Model (based on UDDL) may not reference any FACE elements, this element exists to identify a Conceptual Entity in the Data Model that has a traceability relationship to some other model.

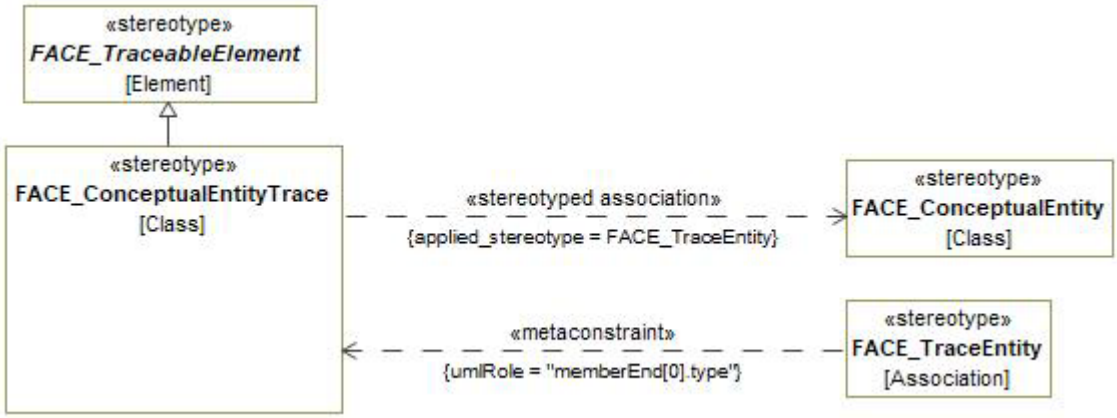

Figure 7-144: FACE_ConceptualEntityTrace

FACE_ConceptualViewTrace

Package: Traceability Model

isAbstract: No

Generalization: FACE_TraceableElement, FACE_TraceabilityElement

Extension: Class

Description

Because the Data Model (based on UDDL) may not reference any FACE elements, this element exists to identify a Conceptual View in the Data Model that has a traceability relationship to some other model.

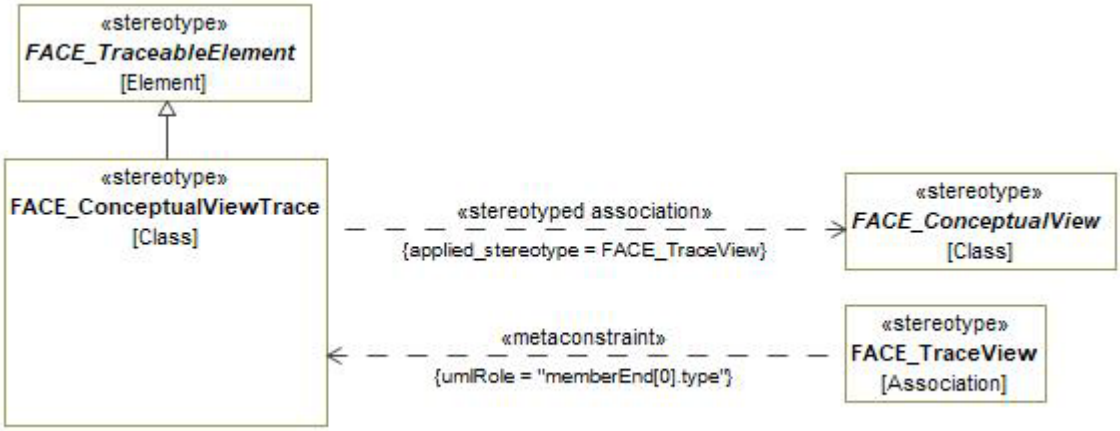

Figure 7-145: FACE_ConceptualViewTrace

FACE_ConnectionTrace

Package: Traceability Model

isAbstract: No

Generalization: FACE_AbstractAssociation

Extension: Association

Description

Used to connect FACE_ConnectionTraceabilitySet elements to their associated FACE_Connections.

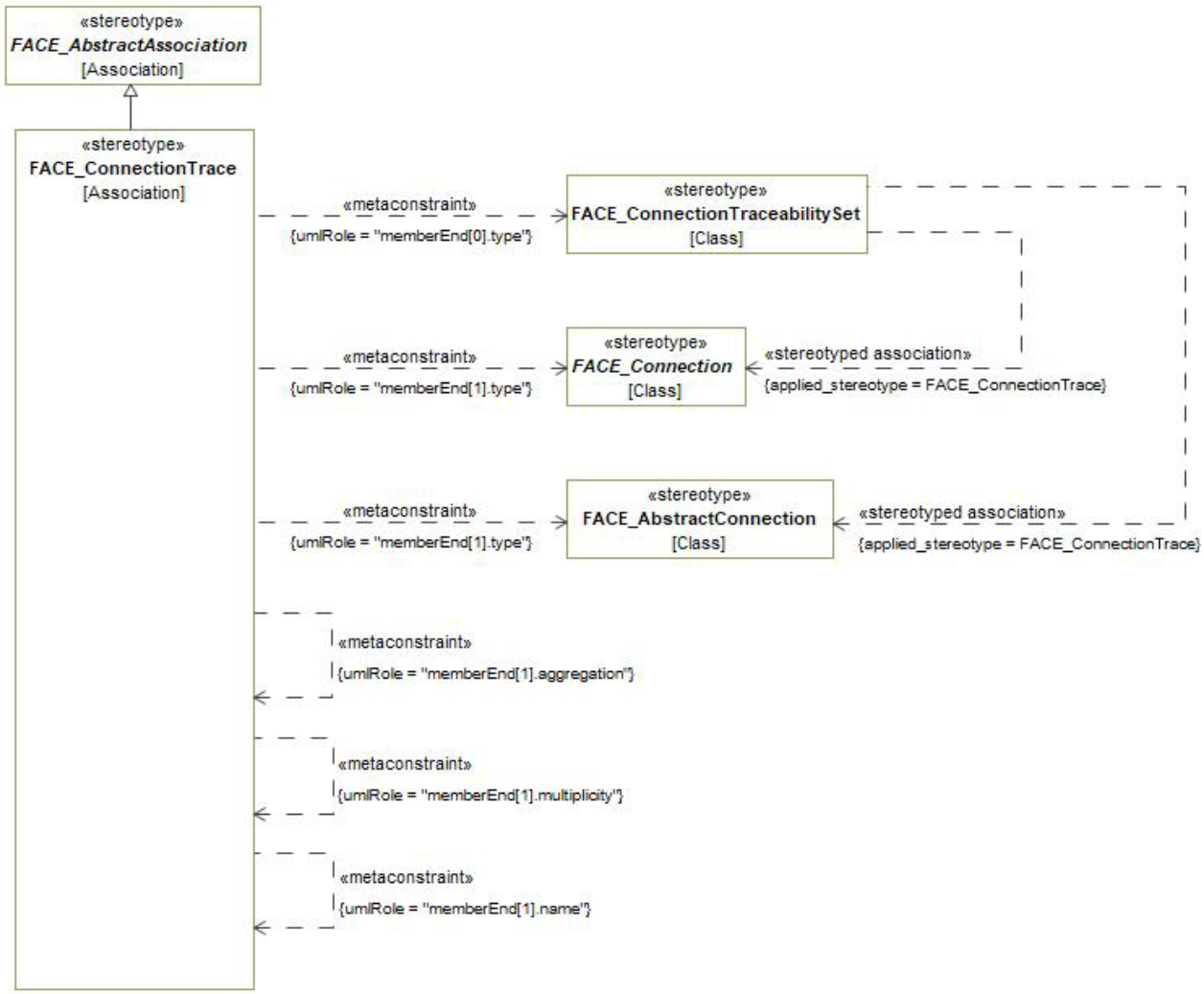

Figure 7-146: FACE_ConnectionTrace

Constraints

C01: FACE_ConnectionTrace.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_ConnectionTraceabilitySet».
C02: FACE_ConnectionTrace.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: FACE_ConnectionTrace.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 0..*

C04: 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"
C05: 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.

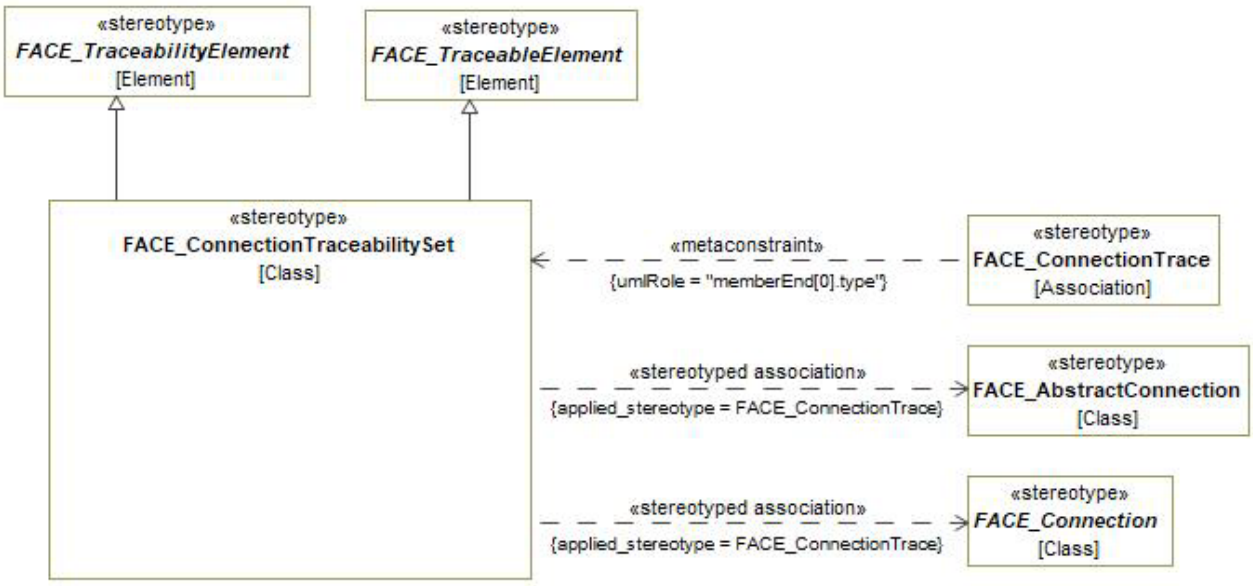

Figure 7-147: FACE_ConnectionTraceabilitySet

FACE_ElementTrace

Package: Traceability Model

isAbstract: No

Generalization: FACE_AbstractAssociation

Extension: Association

Description

Used to connect Traceable Elements to Traceability Points.

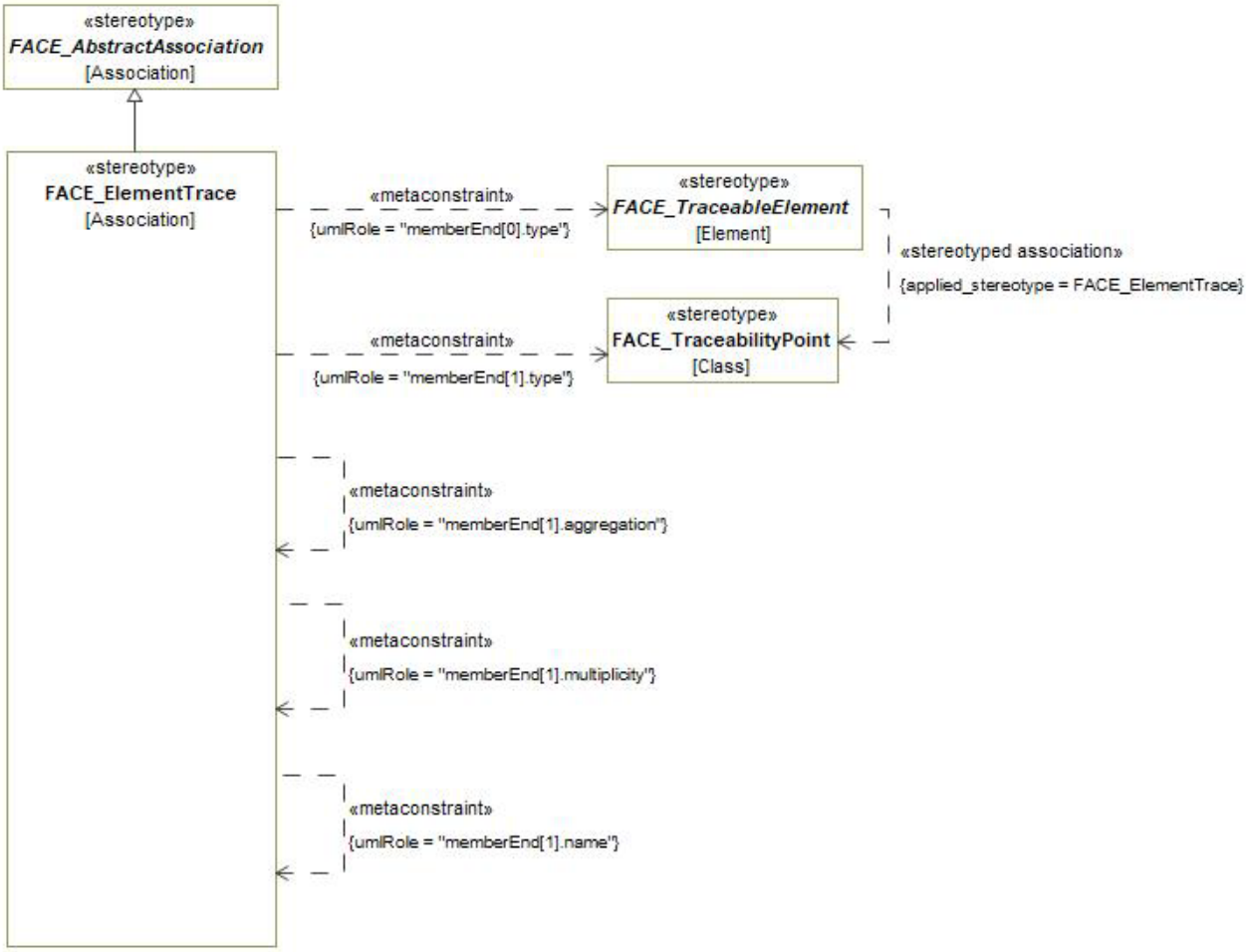

Figure 7-148: FACE_ElementTrace

Constraints

C01: FACE_ElementTrace.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by a specialization of «FACE_TraceableElement».

C02: FACE_ElementTrace.memberEnd[1].aggregation	memberEnd[1].aggregation shall be composite
C03: FACE_ElementTrace.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 0..*
C04: FACE_ElementTrace.memberEnd[1].name	memberEnd[1].name shall be "traceabilityPoint"
C05: FACE_ElementTrace.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_TraceabilityPoint».

FACE_LogicalEntityTrace

Package: Traceability Model

isAbstract: No

Generalization: FACE_TraceableElement, FACE_TraceabilityElement

Extension: Class

Description

Because the Data Model (based on UDDL) may not reference any FACE elements, this element exists to identify a Logical Entity in the Data Model that has a traceability relationship to some other model.

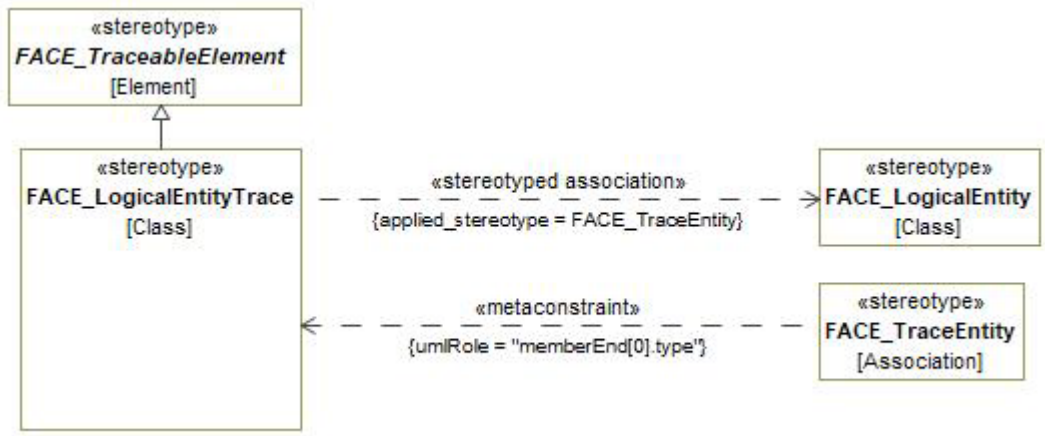

Figure 7-149: FACE_LogicalEntityTrace

FACE_LogicalViewTrace

Package: Traceability Model

isAbstract: No

Generalization: FACE_TraceableElement, FACE_TraceabilityElement

Extension: Class

Description

Because the Data Model (based on UDDL) may not reference any FACE elements, this element exists to identify a Logical View in the Data Model that has a traceability relationship to some other model.

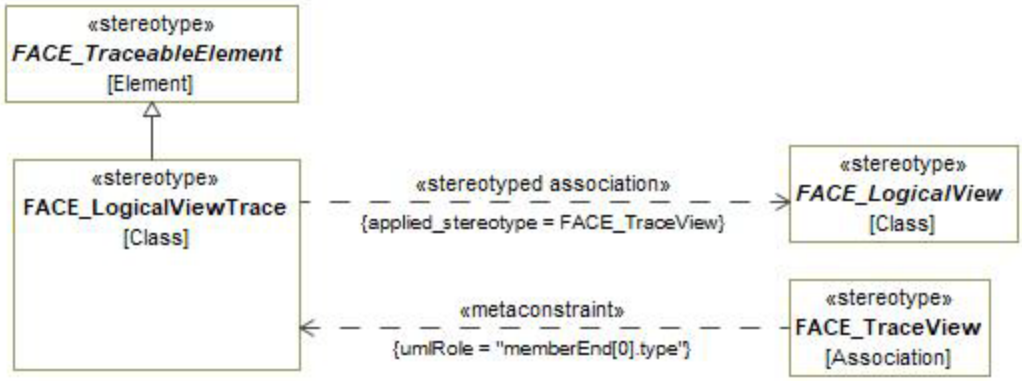

Figure 7-150: FACE_LogicalViewTrace

FACE_PlatformEntityTrace

Package: Traceability Model

isAbstract: No

Generalization: FACE_TraceableElement, FACE_TraceabilityElement

Extension: Class

Description

Because the Data Model (based on UDDL) may not reference any FACE elements, this element exists to identify a Platform Entity in the Data Model that has a traceability relationship to some other model.

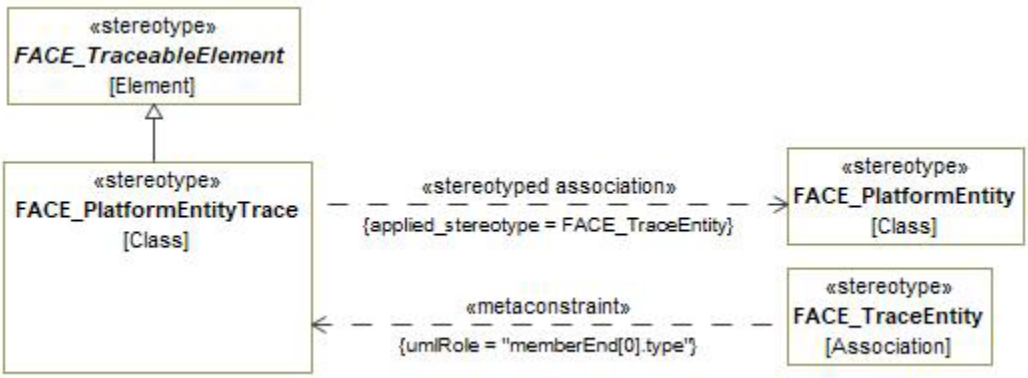

Figure 7-151: FACE_PlatformEntityTrace

FACE_PlatformViewTrace

Package: Traceability Model

isAbstract: No

Generalization: FACE_TraceableElement, FACE_TraceabilityElement

Extension: Class

Description

Because the Data Model (based on UDDL) may not reference any FACE elements, this element exists to identify a Platform View in the Data Model that has a traceability relationship to some other model.

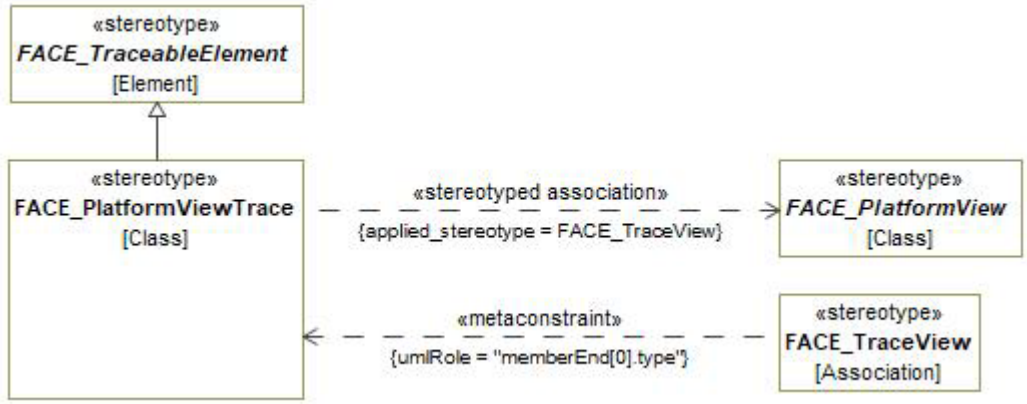

Figure 7-152: FACE_PlatformViewTrace

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.

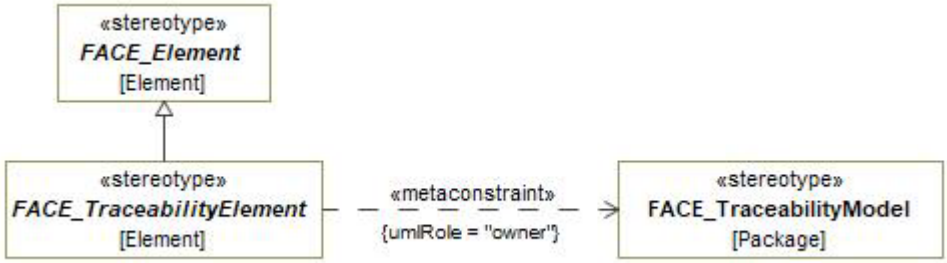

Figure 7-153: abstract FACE_TraceabilityElement

Constraints

C01: 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

C01: 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.

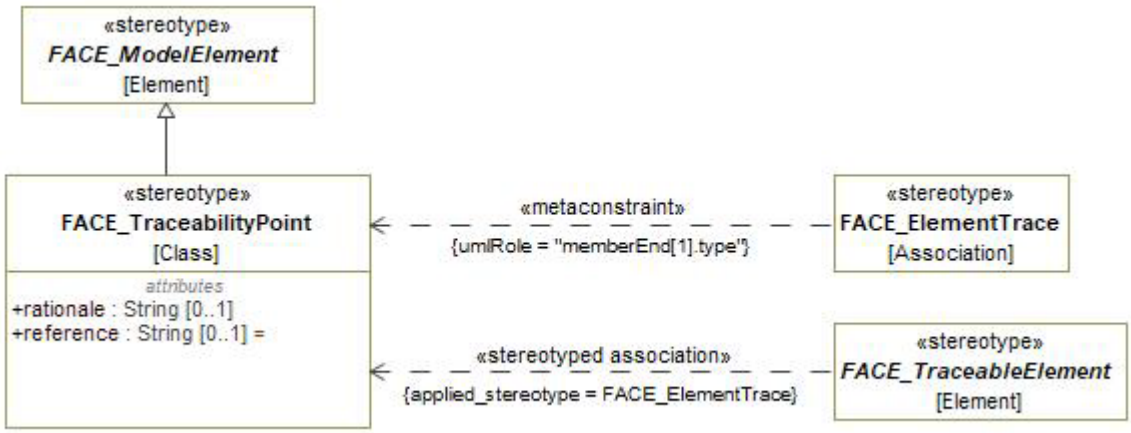

Figure 7-154: FACE_TraceabilityPoint

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.

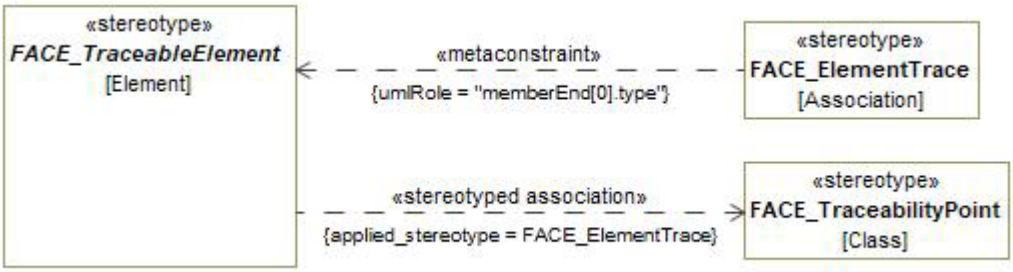

Figure 7-155: abstract FACE_TraceableElement

FACE_TraceEntity

Package: Traceability Model

isAbstract: No

Generalization: FACE_AbstractAssociation

Extension: Association

Description

Used to connect FACE_xxxEntityTraces elements to their associated data model entities.

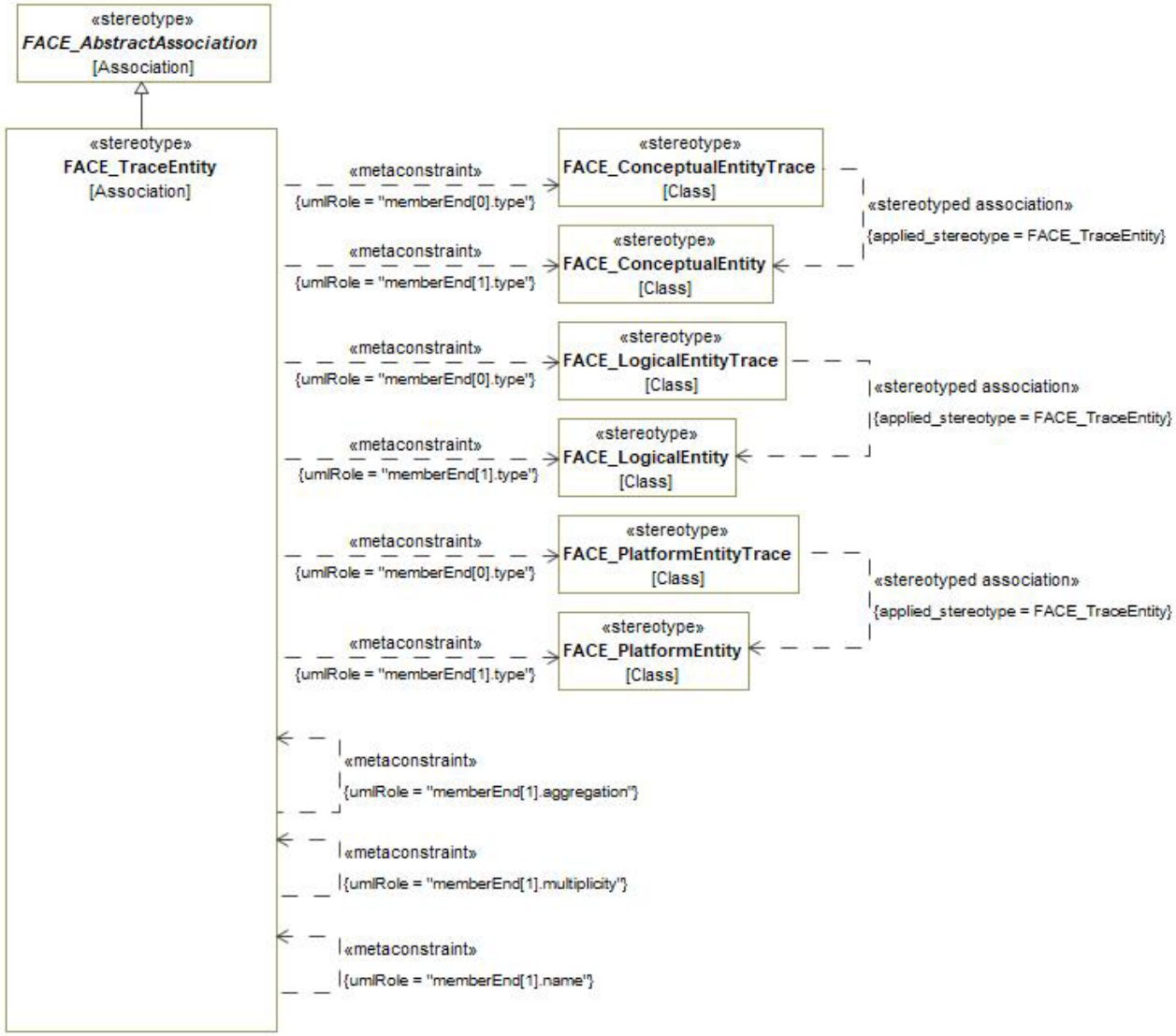

Figure 7-156: FACE_TraceEntity

Constraints

C01: FACE_DMEntityTraceAssoc.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by one of the following: «FACE_ConceptualEntityTrace» «FACE_LogicalEntityTrace» «FACE_PlatformEntityTrace»
C02: FACE_DMEntityTraceAssoc.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none

C03: FACE_DMEntityTraceAssoc.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 1
C04: FACE_DMEntityTraceAssoc.memberEnd[1].name	memberEnd[1].name is "entity"
C05: FACE_DMEntityTraceAssoc.memberEnd[1].type	Based on the memberEnd[0].type value's stereotype: = «FACE_ConceptualEntityTrace», the memberEnd[1].type metaproperty must be stereotyped by «FACE_ConceptualEntity» = «FACE_LogicalEntityTrace», the memberEnd[1].type metaproperty must be stereotyped by «FACE_LogicalEntity» = «FACE_PlatformEntityTrace», the memberEnd[1].type metaproperty must be stereotyped by «FACE_PlatformEntity»

FACE_TraceView

Package: Traceability Model

isAbstract: No

Generalization: FACE_AbstractAssociation

Extension: Association

Description

Used to connect FACE_xxxViewTraces elements to their associated data model Views.

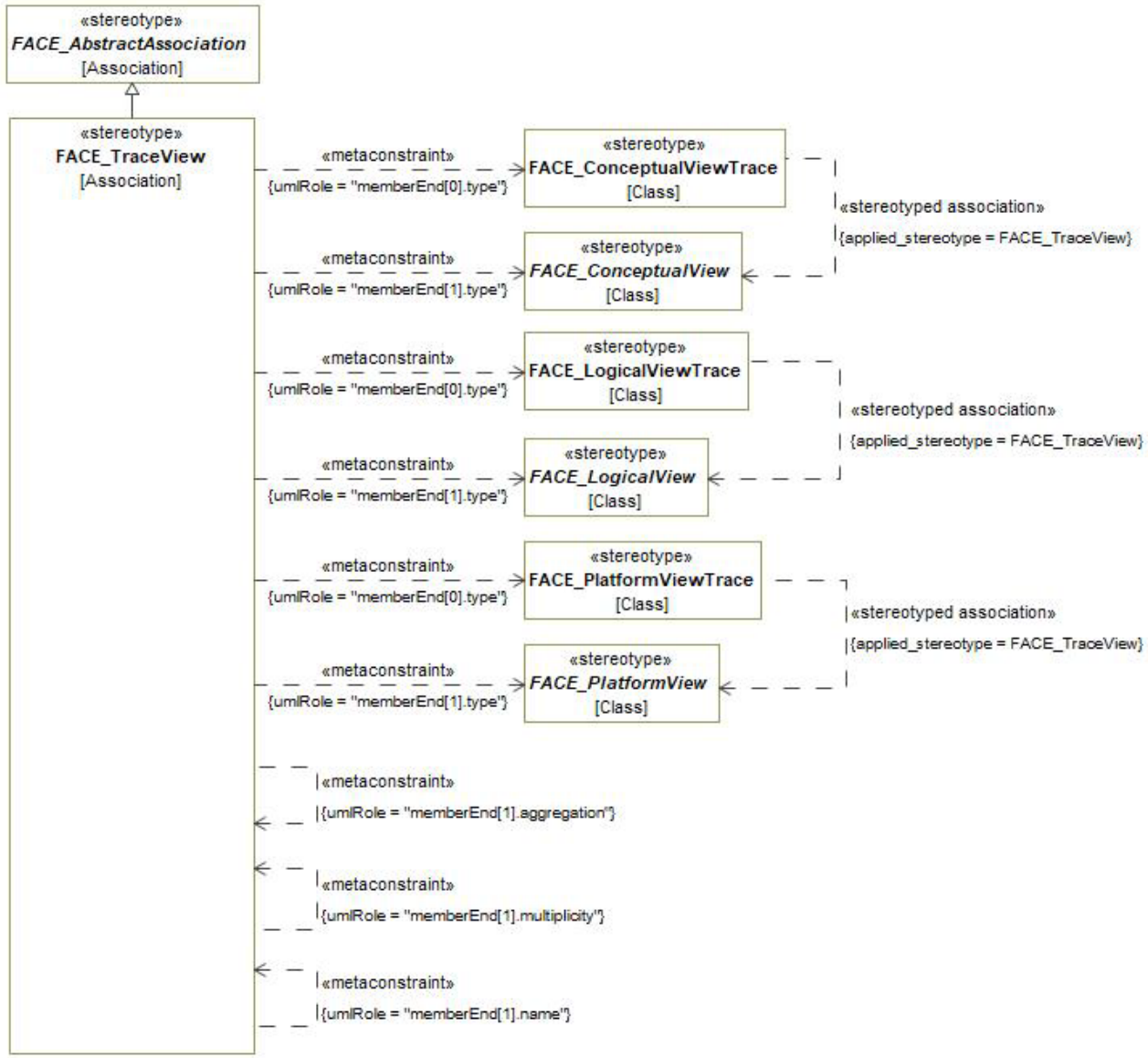

Figure 7-157: FACE_TraceView

Constraints

C01: FACE_DMViewTraceAssoc.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by one of the following: «FACE_ConceptualViewTrace» «FACE_LogicalViewTrace» «FACE_PlatformViewTrace»

C02: FACE_DMViewTraceAssoc.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: FACE_DMViewTraceAssoc.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 1
C04: FACE_DMViewTraceAssoc.memberEnd[1].name	memberEnd[1].name is "view"
C05: FACE_DMViewTraceAssoc.memberEnd[1].type	Based on the memberEnd[0].type value's stereotype: = «FACE_ConceptualViewTrace», the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_ConceptualView» = «FACE_LogicalViewTrace», the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_LogicalView» = «FACE_PlatformViewTrace», the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_PlatformView»

FACE_UoPTrace

Package: Traceability Model

isAbstract: No

Generalization: FACE_AbstractAssociation

Extension: Association

Description

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

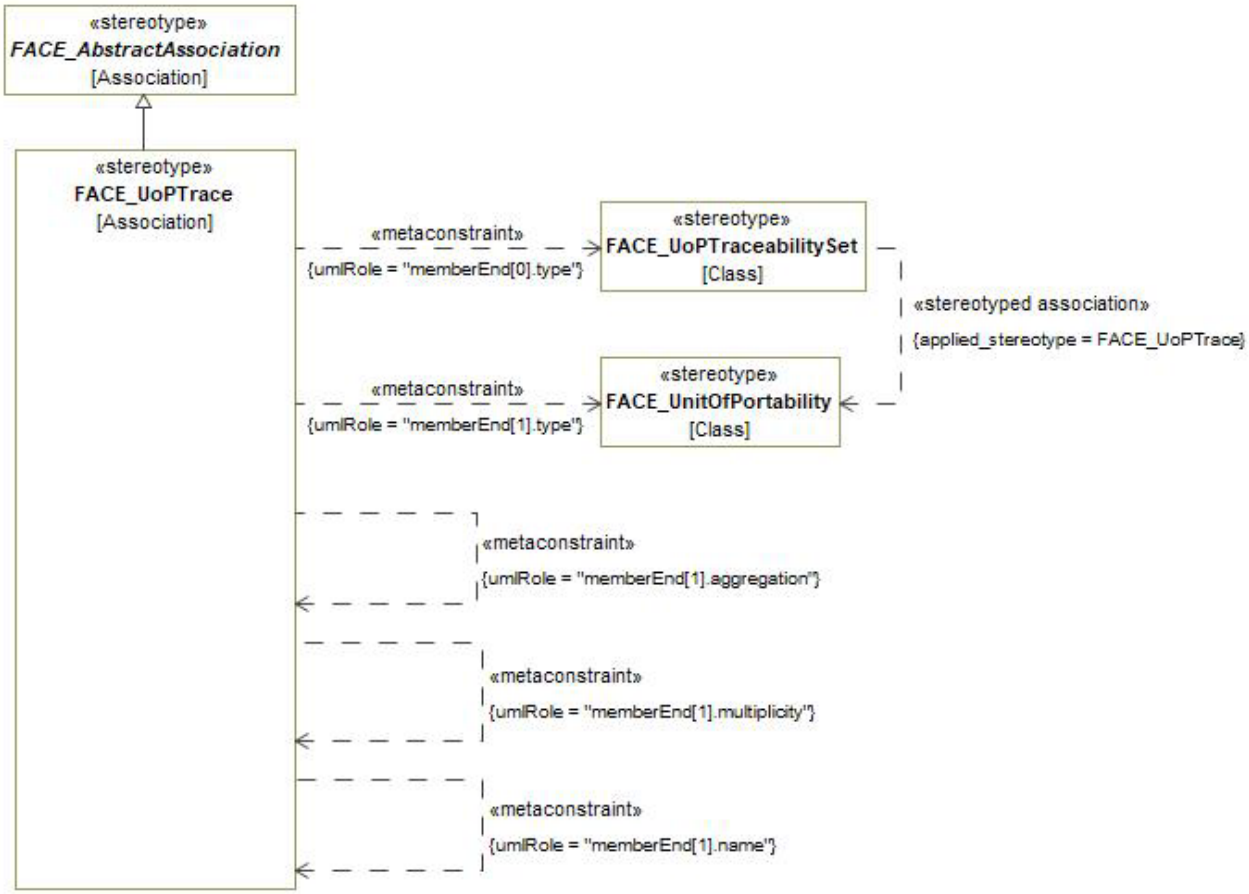

Figure 7-158: FACE_UoPTrace

Constraints

C01: FACE_UoPTrace.memberEnd[0].type	The value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_UoPTraceabilitySet».
C02: FACE_UoPTrace.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: FACE_UoPTrace.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 0..*
C04: FACE_UoPTrace.memberEnd[1].name	memberEnd[1].name shall be "uop"
C05: 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.

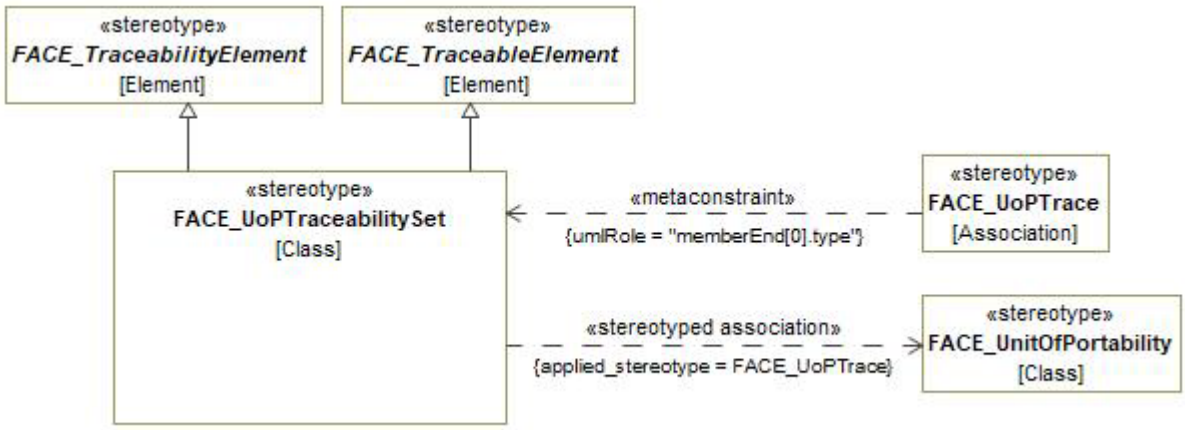

Figure 7-159: FACE_UoPTraceabilitySet

7.1.1.4 FACE_Profile::FACE Data Architecture::UoP Model

The UoP Model package of the FACE Profile contains elements that represent the UoP Model subpackage as specified in the FACE metamodel.

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.

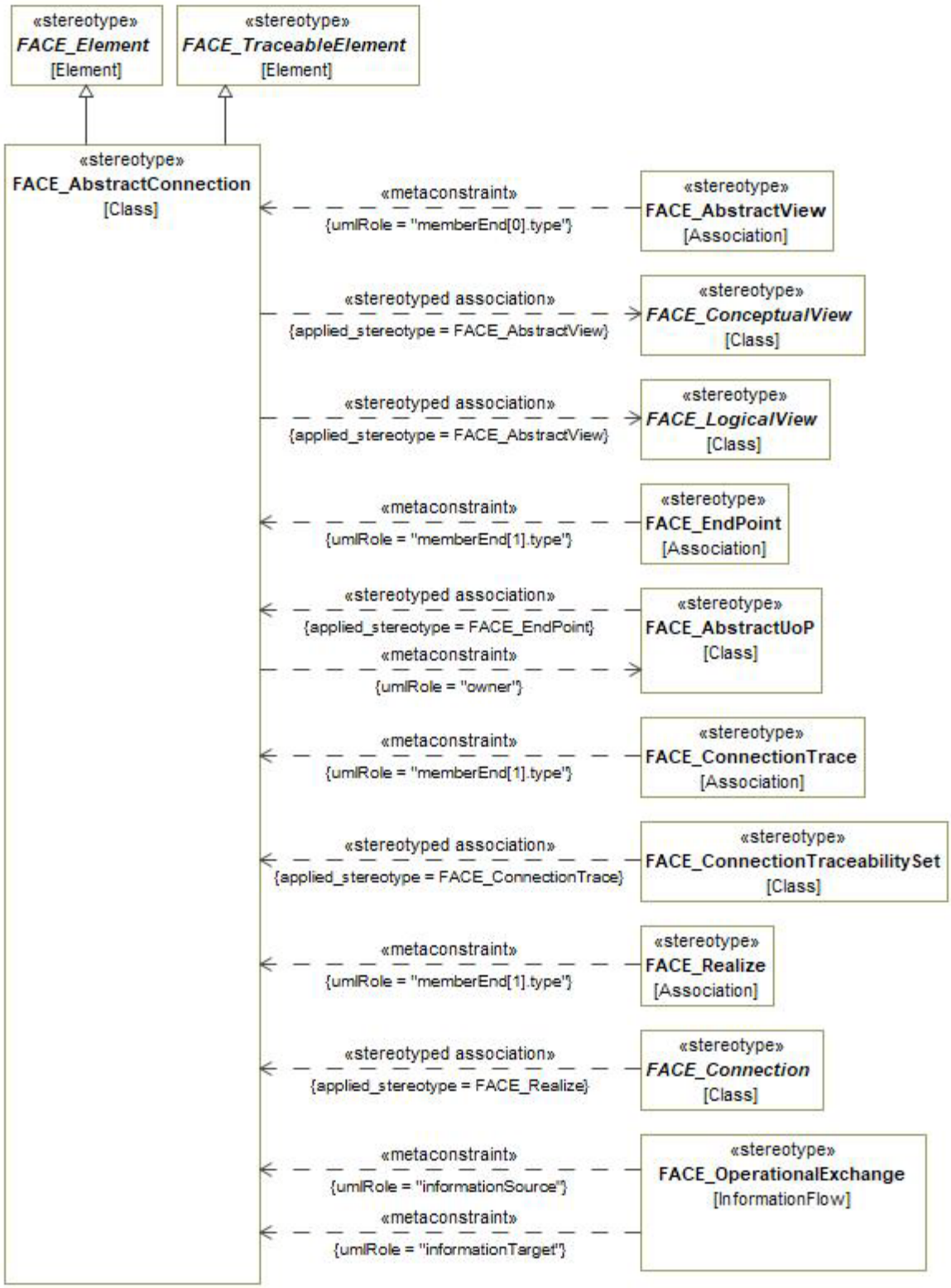

Figure 7-160: FACE_AbstractConnection

Constraints

C01: 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).

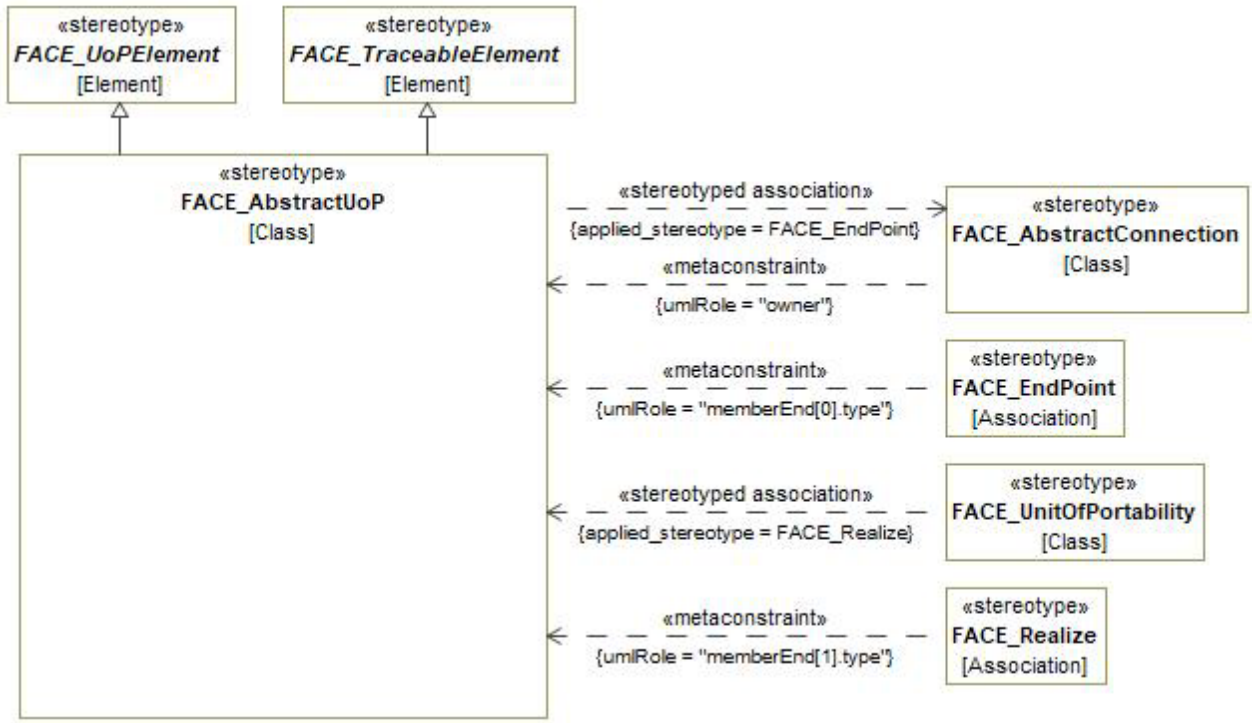

Figure 7-161: FACE_AbstractUoP

FACE Conformance/OCL Constraints

C01: FACE_AbstractUoP.onlyLogicalOrOnlyConceptual	A FACE_AbstractUoP must be entirely logical or entirely conceptual. (Its AbstractConnections all have their logicalView set and conceptualView not set or all have their conceptualView set and 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.

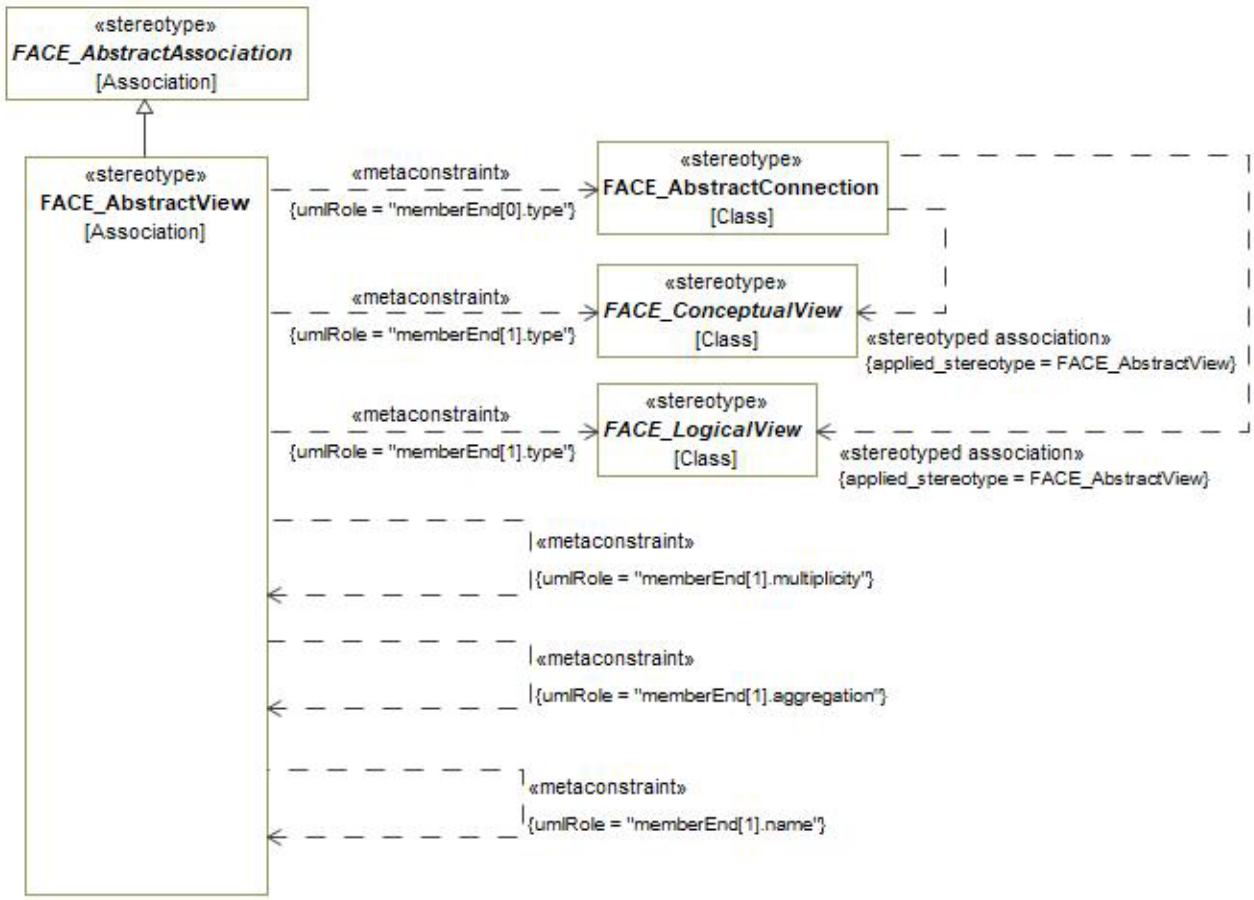

Figure 7-162: FACE_AbstractView

Constraints

C01: FACE_AbstractView.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_AbstractConnection».

C02: FACE_AbstractView.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: FACE_AbstractView.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 0..1
C04: 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"
C05: 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.

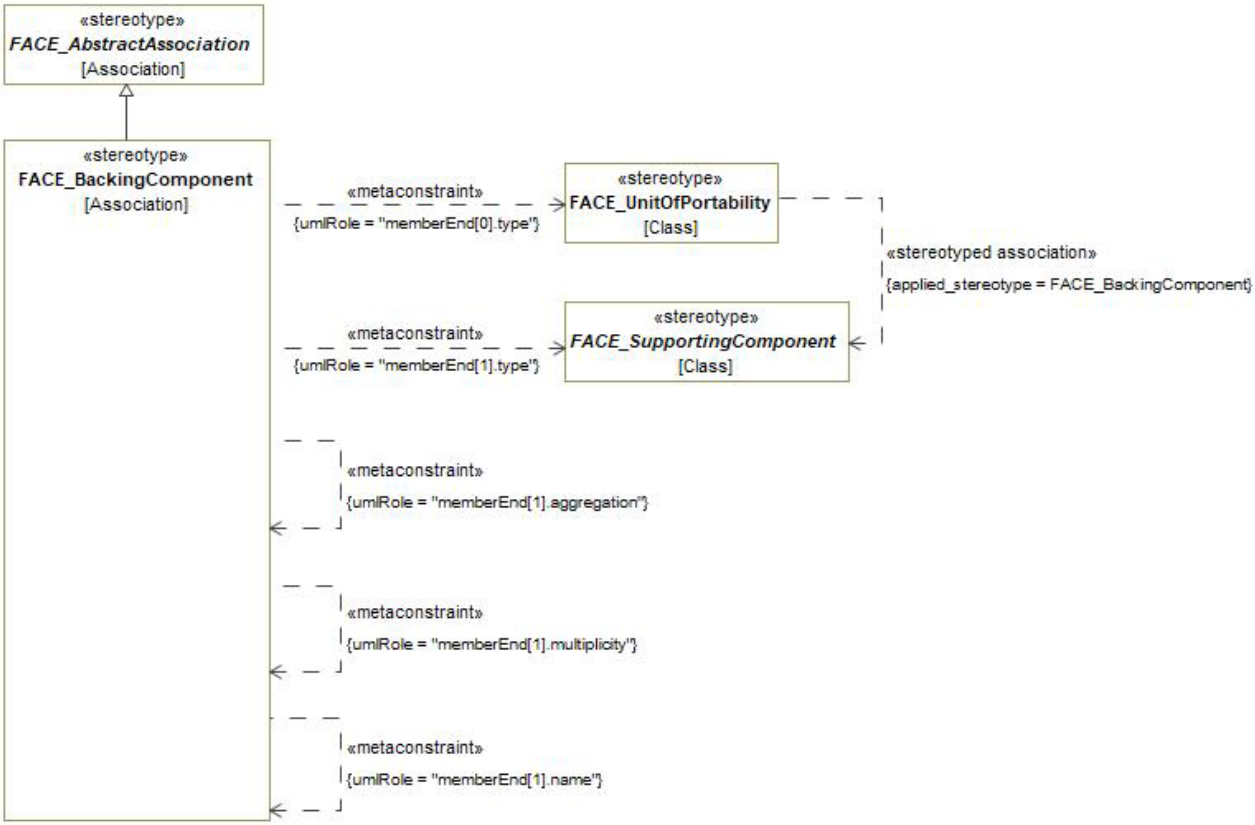

Figure 7-163: FACE_BackingComponent

Constraints

C01: FACE_BackingComponent.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_UnitOfPortability».
C02: FACE_BackingComponent.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: FACE_BackingComponent.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 0..*
C04: FACE_BackingComponent.memberEnd[1].name	memberEnd[1].name shall be "supportingComponent"
C05: FACE_BackingComponent.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_SupportingComponent».

FACE_BoundQuery

Package: UoP Model

isAbstract: No

Generalization: FACE_AbstractAssociation

Extension: Association

Description

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

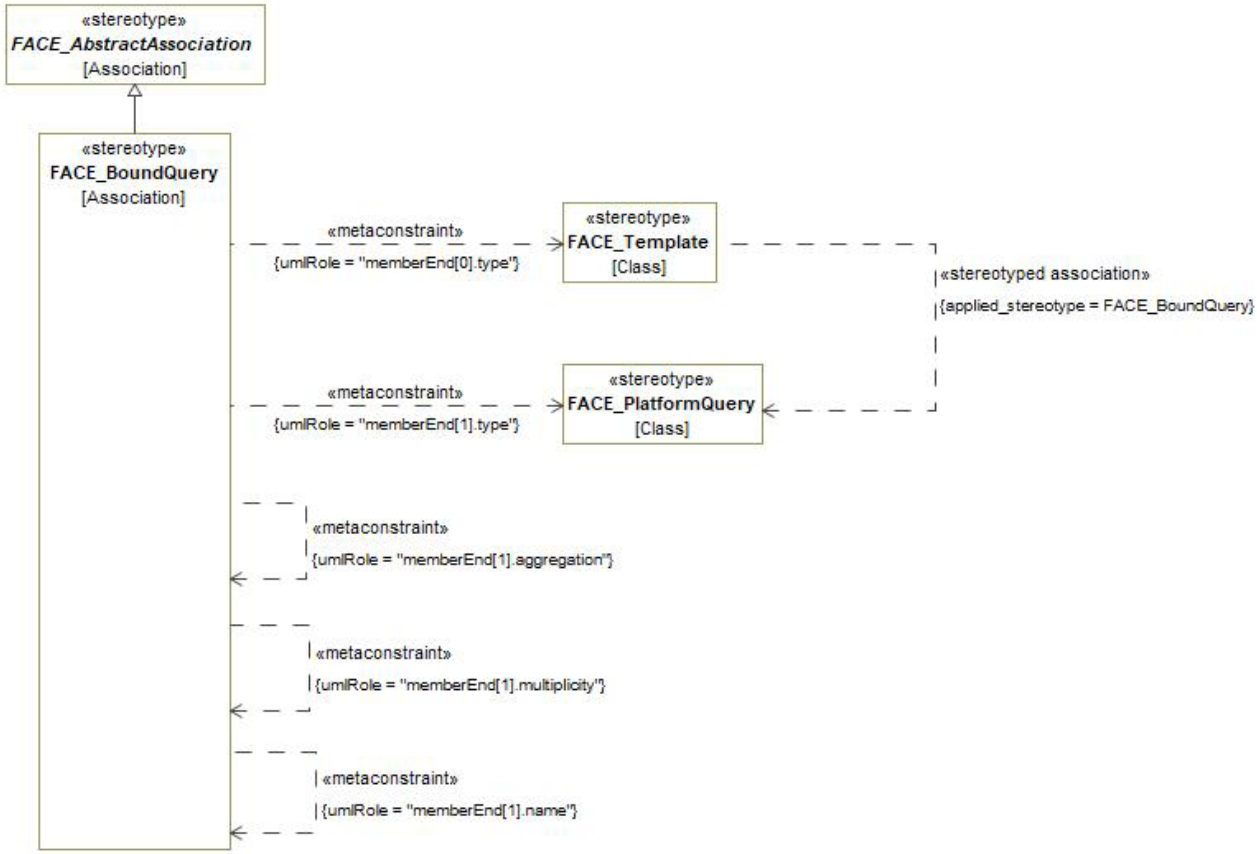

Figure 7-164: FACE_BoundQuery

Constraints

C01: FACE_BoundQuery.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_Template».
C02: FACE_BoundQuery.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: FACE_BoundQuery.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 0..1
C04: FACE_BoundQuery.memberEnd[1].name	memberEnd[1].name shall be "boundQuery"

C05: FACE_BoundQuery.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_PlatformQuery».

FACE_ClientServerConnection

Package: UoP Model

isAbstract: No

Generalization: FACE_Connection

Description

A FACE_ClientServerConnection is a Request/Reply Connection as defined in Section 4.7 of the FACE Technical Standard.

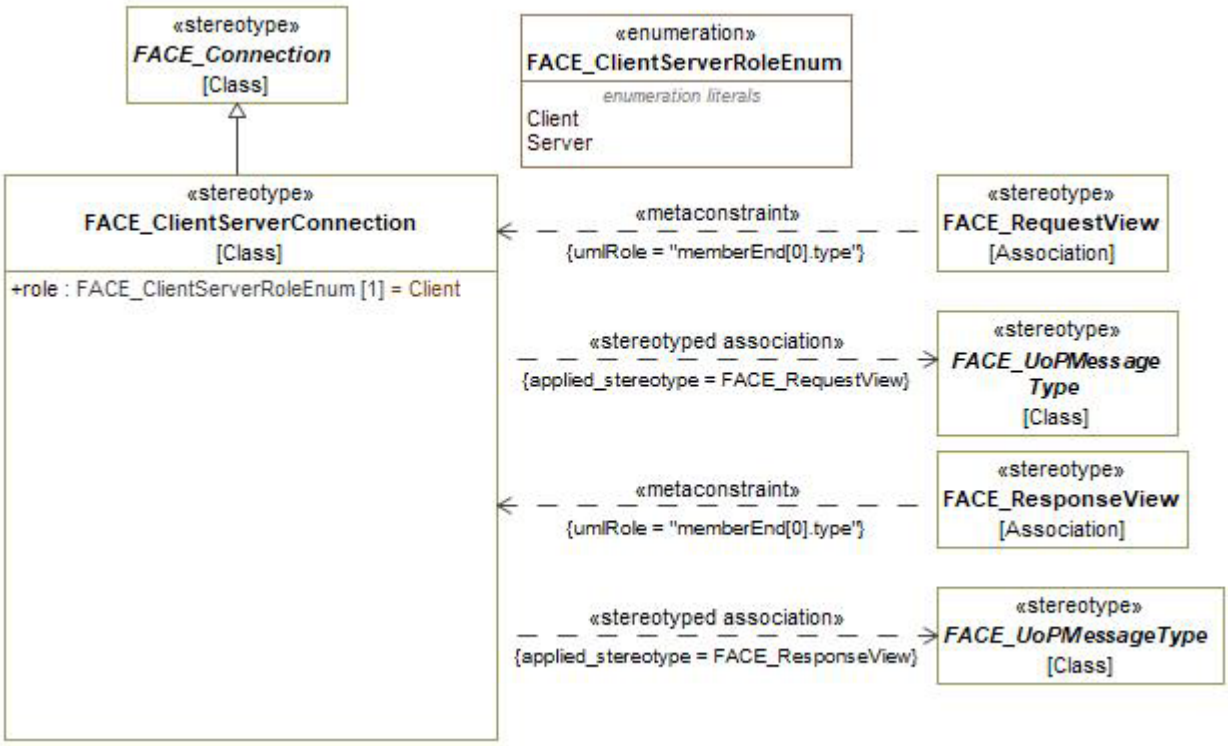

Figure 7-165: FACE_ClientServerConnection

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 -

Figure 7-166: FACE_ClientServerRoleEnum

FACE_ComponentFramework

Package: UoP Model

isAbstract: No

Generalization: FACE_SupportingComponent

Extension: Class

Description

A FACE_ComponentFramework is a component framework as defined in Section 4.2.4 of the FACE Technical Standard.

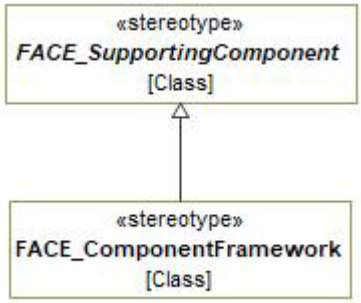

Figure 7-167: FACE_ComponentFramework

FACE_ComponentTypeEnum

Package: UoP Model

isAbstract: No

Description

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

PortableComponent -
PlatformSpecificComponent -

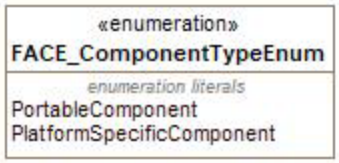

Figure 7-168: FACE_ComponentTypeEnum

FACE_CompositeTemplate

Package: UoP Model

isAbstract: No

Generalization: FACE_UoPMessageType

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 an IDL union or as members of an IDL struct.

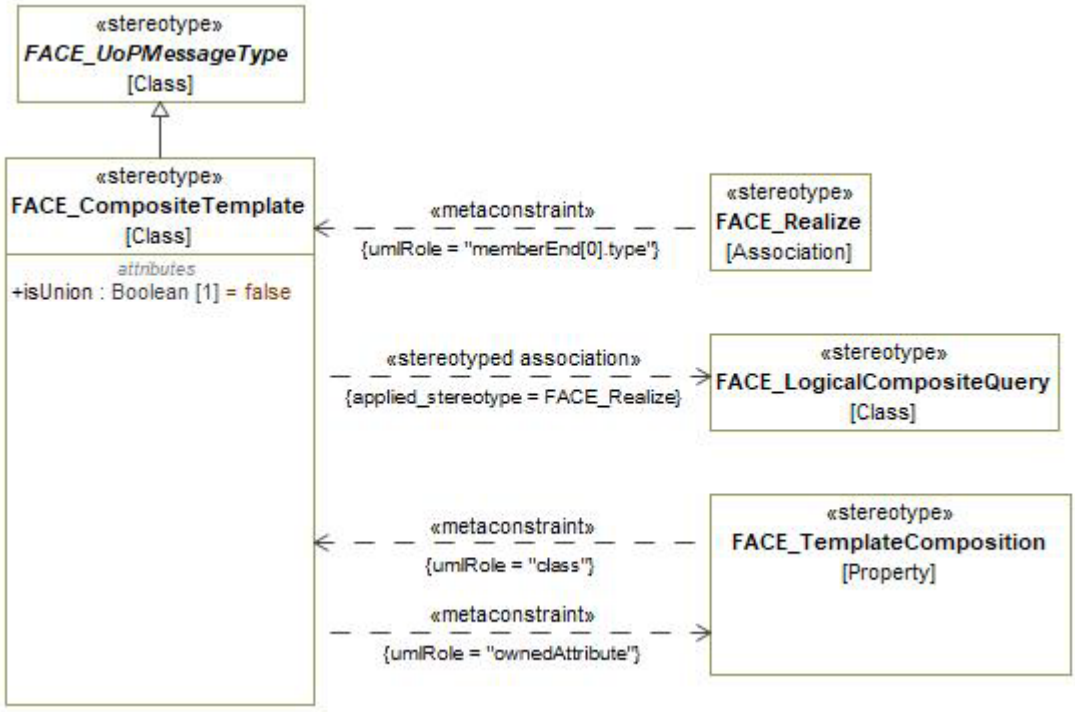

Figure 7-169: FACE_CompositeTemplate

Attributes

isUnion : Boolean [1]

Constraints

C01: 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

C01: FACE_CompositeTemplate.compositionsConsistentWithRealization	FACE_TemplateCompositions in a platform FACE_CompositeTemplate must realize FACE_QueryCompositions in the FACE_LogicalCompositeQuery that the platform FACE_CompositeTemplate realizes.
C02: FACE_CompositeTemplate.compositionsHaveUniqueRolenames	A FACE_TemplateComposition's rolename must be unique within a FACE_CompositeTemplate.
C03: FACE_CompositeTemplate.noCyclesInConstruction	A FACE_CompositeTemplate must not compose itself, directly or indirectly.
C04: FACE_CompositeTemplate.realizationUnionConsistent	A FACE_CompositeTemplate that realizes must have the same "isUnion" property as the FACE_CompositeQuery it realizes.
C05: FACE_CompositeTemplate.realizedCompositionsHaveDifferentTypes	A FACE_CompositeTemplate may not contain two FACE_TemplateCompositions that realize the same FACE_QueryComposition.
C06: FACE_CompositeTemplate.viewComposedOnce	A FACE_CompositeTemplate must not compose the same FACE_Template more than once.

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 metatype's "type" attribute represents the FACE "messageType" attribute that

specifies the FACE_MessageType 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.

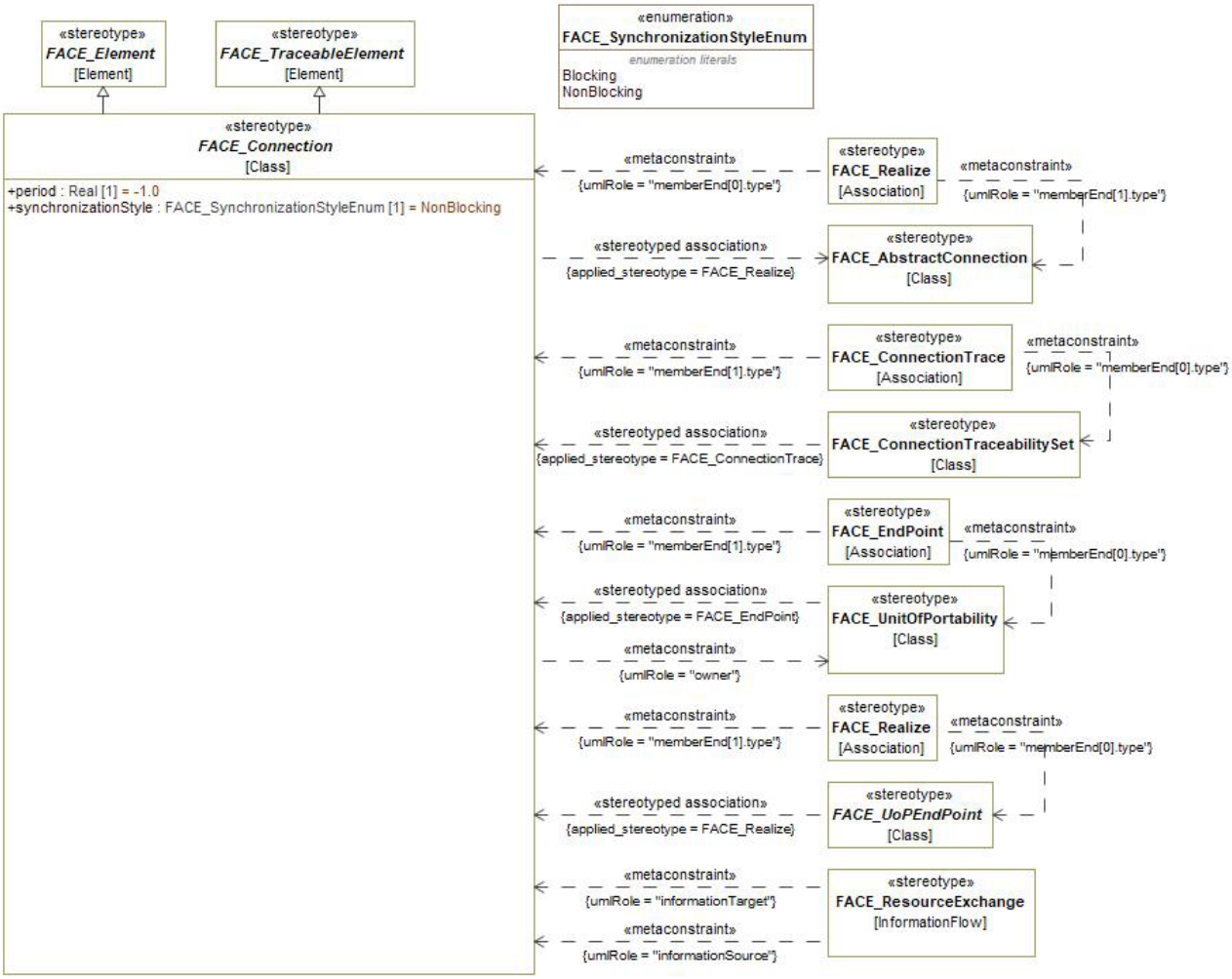

Figure 7-170: abstract FACE_Connection

Attributes

period : Real [1]

synchronizationStyle : FACE_SynchronizationStyleEnum [1]

Constraints

C01: 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

C01: 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 -

Figure 7-171: FACE_DesignAssuranceLevelEnum

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 -

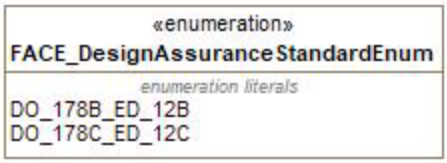

Figure 7-172: FACE_DesignAssuranceStandardEnum

FACE_EffectiveQuery

Package: UoP Model

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.

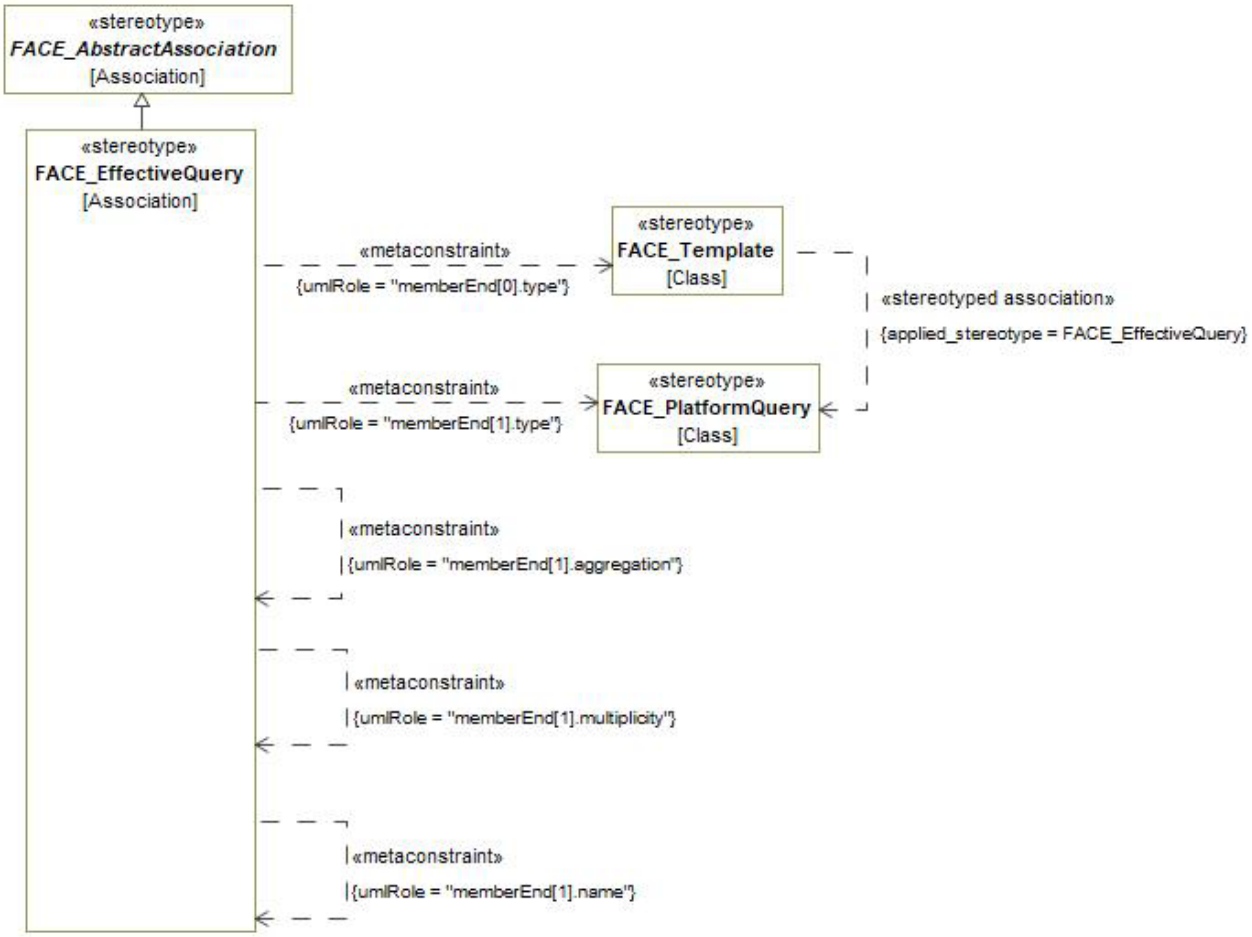

Figure 7-173: FACE_EffectiveQuery

Constraints

C01: FACE_EffectiveQuery.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_Template».
C02: FACE_EffectiveQuery.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: FACE_EffectiveQuery.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 0..1
C04: FACE_EffectiveQuery.memberEnd[1].name	memberEnd[1].name shall be "effectiveQuery"
C05: FACE_EffectiveQuery.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by «FACE_PlatformQuery».

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 4.2.3 of the FACE Technical Standard.

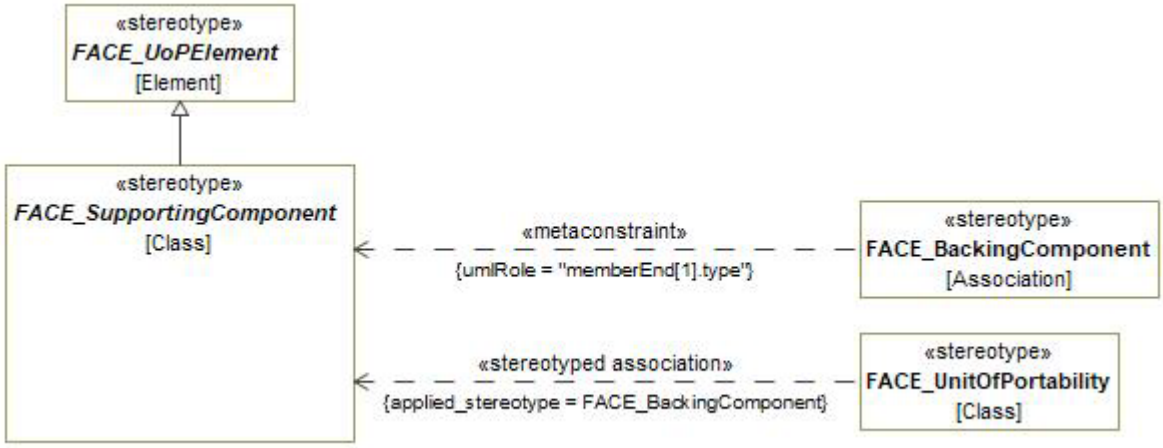

Figure 7-174: FACE_LanguageRunTime

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).

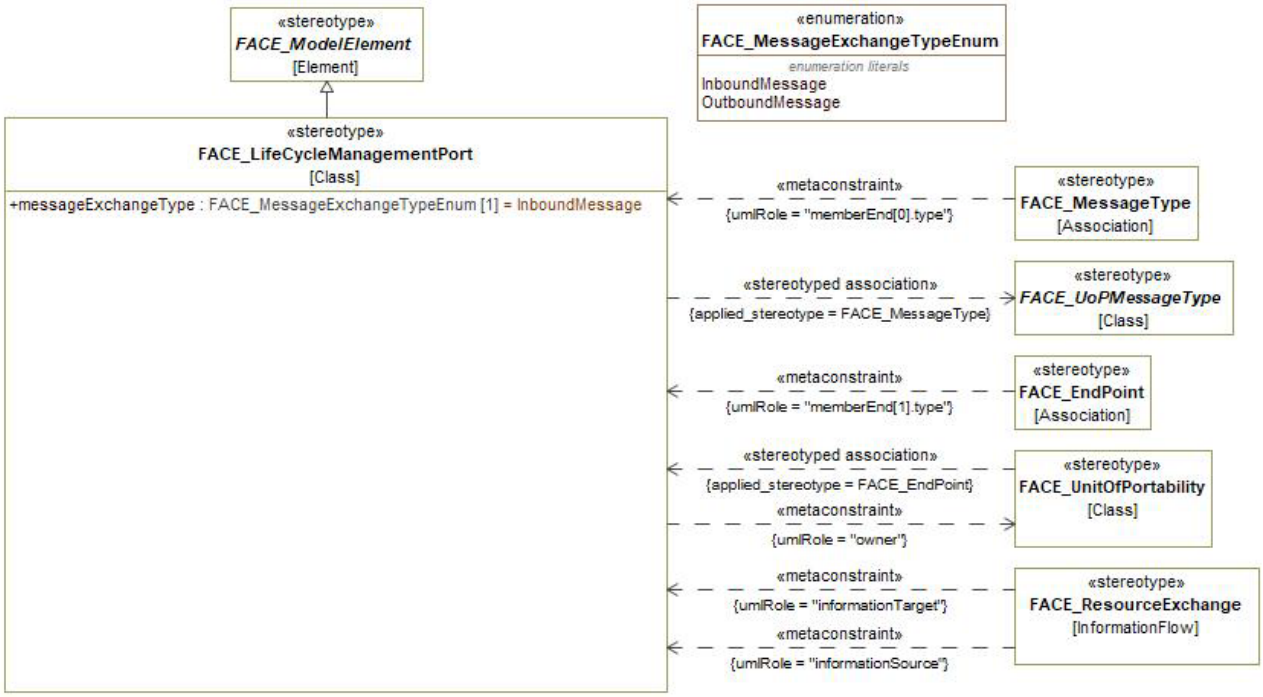

Figure 7-175: FACE_LifeCycleManagementPort

Attributes

messageExchangeType : FACE_MessageExchangeTypeEnum [1]

Constraints

C01: 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 -

Figure 7-176: FACE_MessageExchangeTypeEnum

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 -

Figure 7-177: FACE_PartitionTypeEnum

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 -

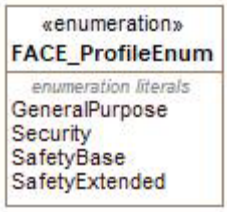

Figure 7-178: FACE_ProfileEnum

FACE_ProgrammingLanguageEnum

Package: UoP Model

isAbstract: No

Description

The FACE_ProgrammingLanguageEnum enumeration captures the options for programming language API bindings as defined by Section 4.14 of the FACE Technical Standard. Its enumeration literals are:

C -
CPP -
Java -
Ada -

Figure 7-179: FACE_ProgrammingLanguageEnum

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).

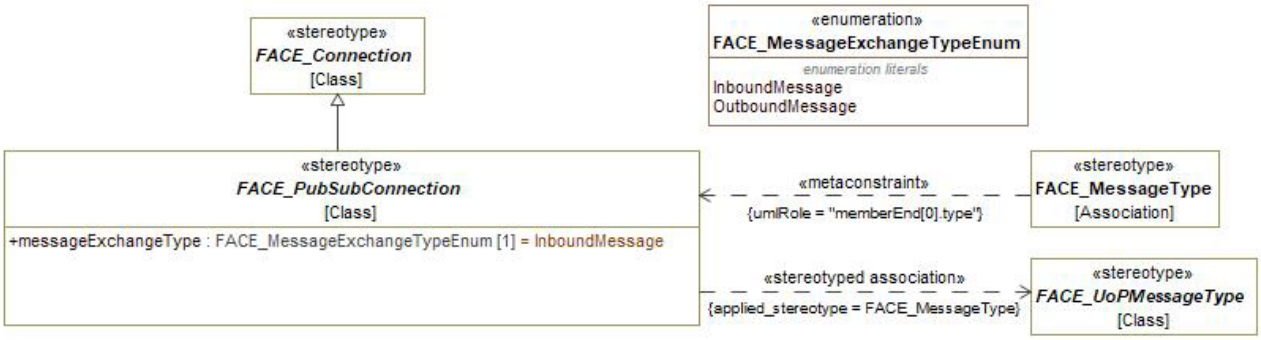

Figure 7-180: abstract FACE_PubSubConnection

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 4.8 of the FACE Technical Standard.

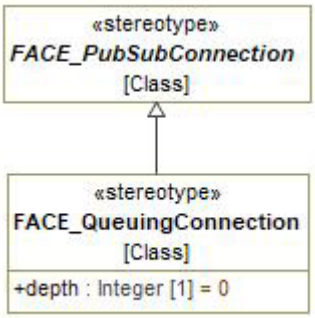

Figure 7-181: FACE_QueuingConnection

Attributes

depth : Integer [1]

FACE Conformance/OCL Constraints

C01: 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).

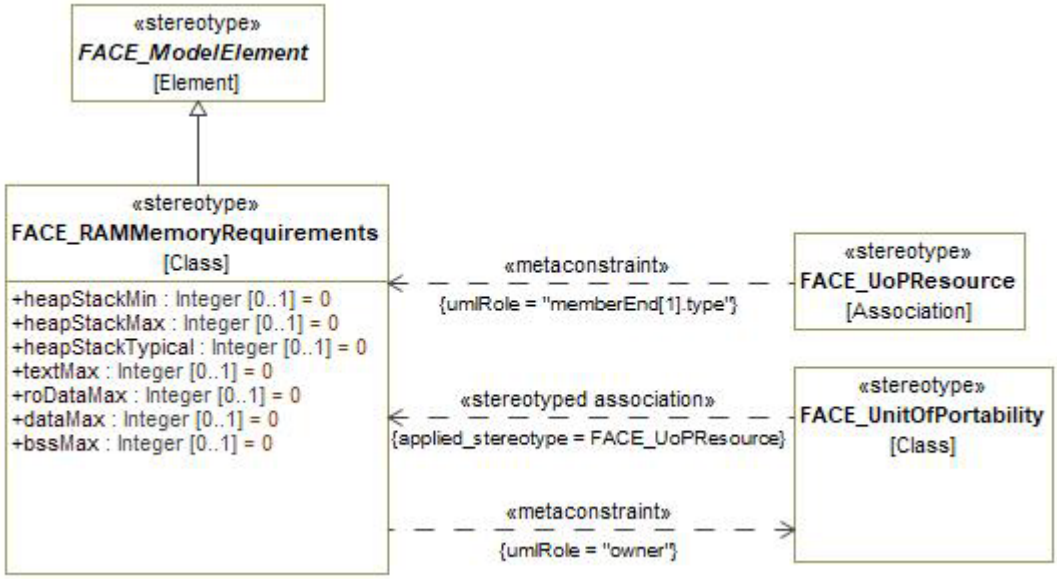

Figure 7-182: FACE_RAMMemoryRequirements

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

C01: 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.

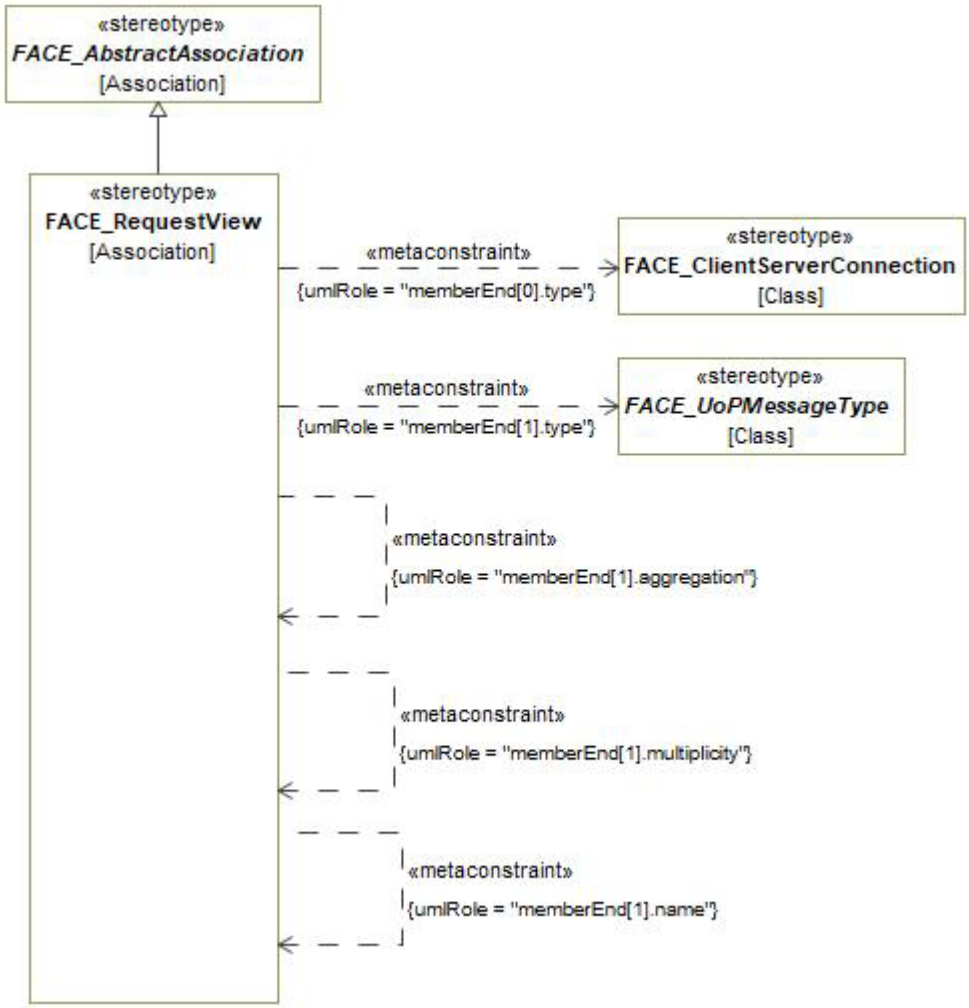

Figure 7-183: FACE_RequestView

Constraints

C01: FACE_RequestView.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_ClientServerConnection».
C02: FACE_RequestView.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: FACE_RequestView.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 1
C04: FACE_RequestView.memberEnd[1].name	memberEnd[1].name shall be "requestType"

C05: FACE_RequestView.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_MessageType».

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.

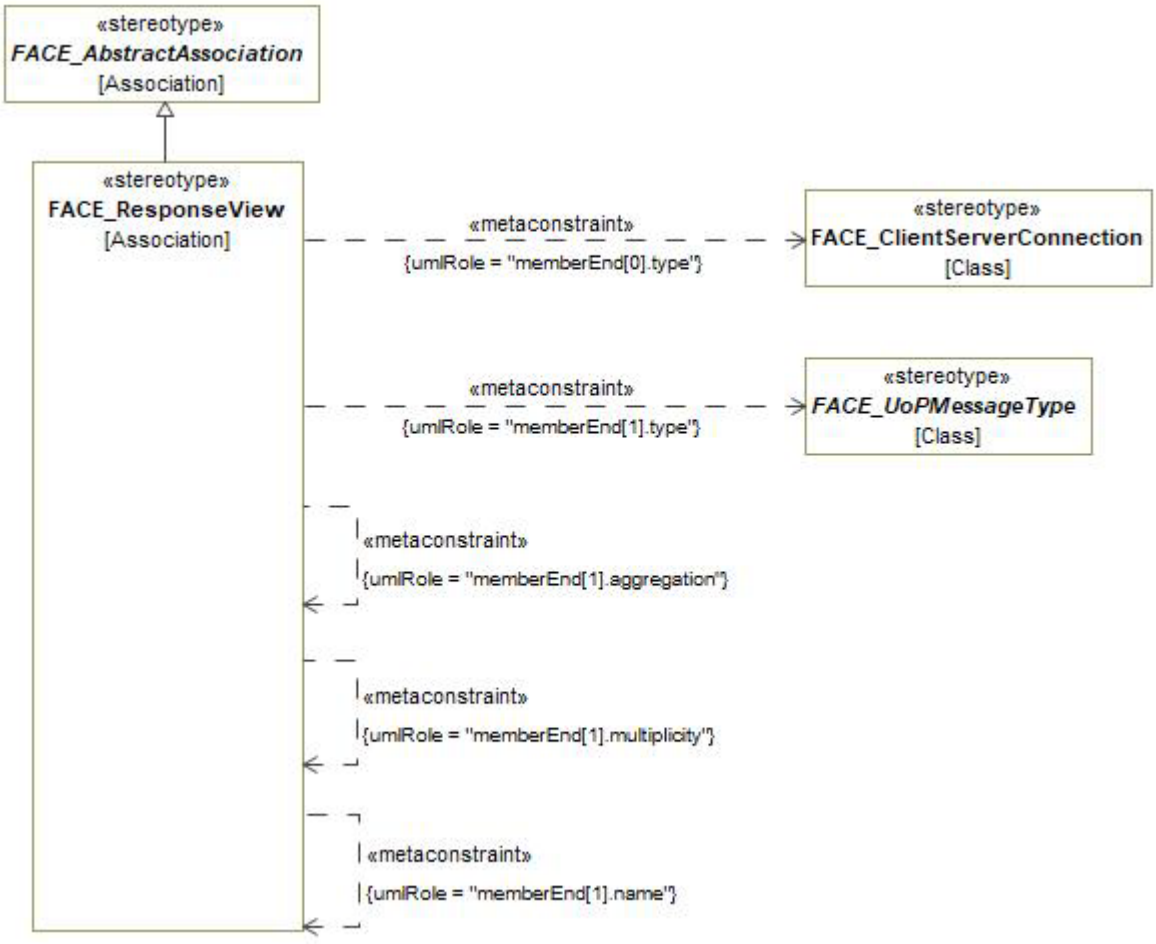

Figure 7-184: FACE_ResponseView

Constraints

C01: FACE_ResponseView.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_ClientServerConnection».
C02: FACE_ResponseView.memberEnd[1].aggregation	memberEnd[1].aggregation shall be none
C03: FACE_ResponseView.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 1
C04: FACE_ResponseView.memberEnd[1].name	memberEnd[1].name shall be "responseType"
C05: FACE_ResponseView.memberEnd[1].type	Value for the memberEnd[1].type metaproperty must be stereotyped by a specialization of «FACE_MessageType».

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 4.8 of the FACE Technical Standard.

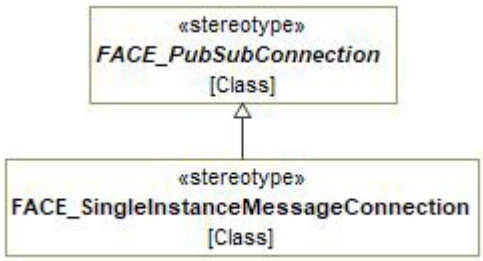

Figure 7-185: FACE_SingleInstanceMessageConnection

FACE_SupportingComponent

Package: UoP Model

isAbstract: Yes

Generalization: FACE_UoPElement

Extension: Class

Description

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

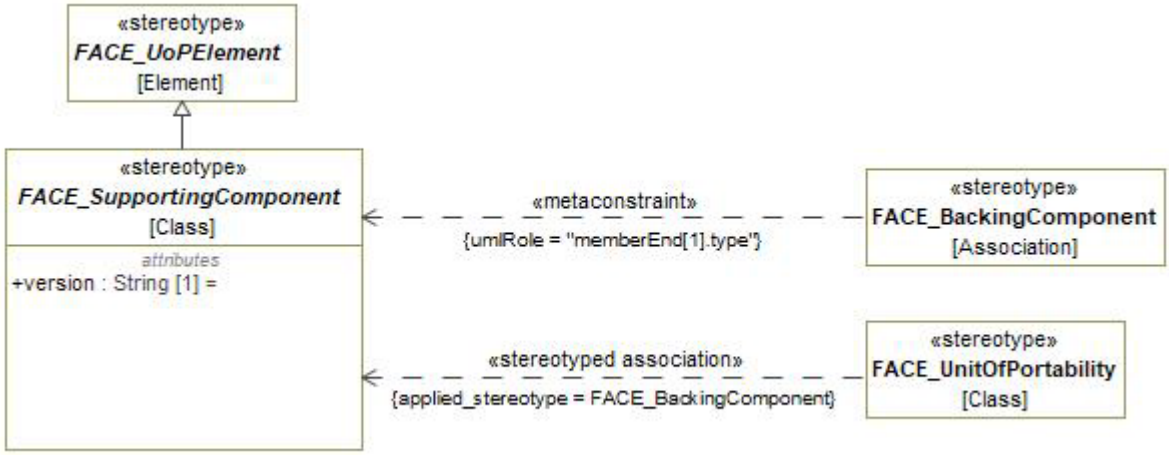

Figure 7-186: abstract FACE_SupportingComponent

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 -

Figure 7-187: FACE_SynchronizationStyleEnum

FACE_Template

Package: UoP Model

isAbstract: No

Generalization: FACE_UoPMessageType

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 of the FACE Technical Standard.

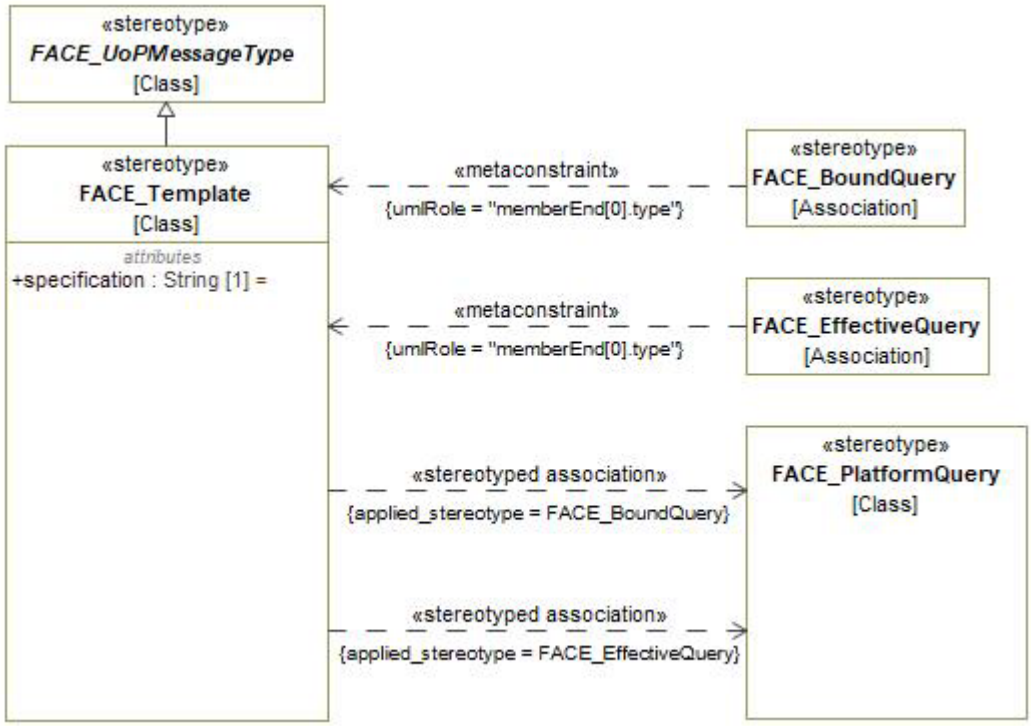

Figure 7-188: FACE_Template

Attributes

specification : String [1]

FACE_TemplateComposition

Package: UoP Model

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 "name" property represents the "rolename" attribute that defines the name of the composed platform View within the scope of the composing CompositeTemplate. The “type” of a TemplateComposition is the platform View being used to construct the CompositeTemplate.

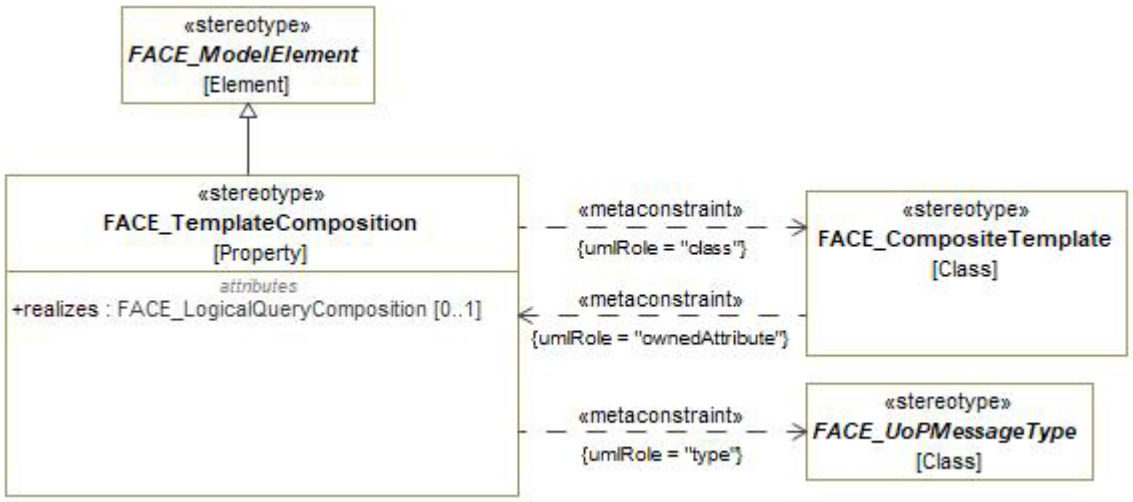

Figure 7-189: FACE_TemplateComposition

Attributes

realizes : FACE_LogicalQueryComposition [0..1]

Constraints

C01: FACE_TemplateComposition.class	Value for class metaproperty must be stereotyped «FACE_CompositeTemplate».
C02: FACE_TemplateComposition.type	Value for type metaproperty must be stereotyped «FACE_MessageType» or its specializations.

FACE Conformance/OCL Constraints

C01: FACE_TemplateComposition.rolenameIsNotReservedWord	The rolename of a FACE_TemplateComposition may not be an IDL reserved word.
C02: FACE_TemplateComposition.rolenameIsValidIdentifier	The rolename of a FACE_TemplateComposition must be a valid identifier.

C03: 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_Thread

Package: UoP Model

isAbstract: No

Generalization: FACE_ModelElement

Extension: Class

Description

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

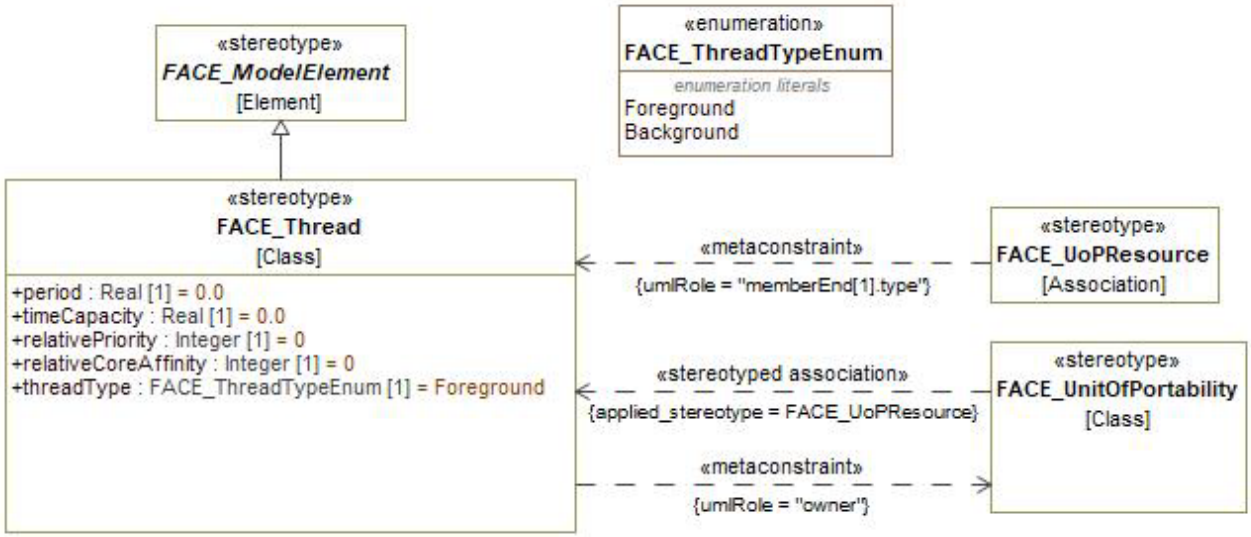

Figure 7-190: FACE_Thread

Attributes

period : Real [1]

relativeCoreAffinity : Integer [1]

relativePriority : Integer [1]

threadType : FACE_ThreadTypeEnum [1]

timeCapacity : Real [1]

Constraints

C01: 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 -

Figure 7-191: FACE_ThreadTypeEnum

FACE_UnitOfPortability

Package: UoP Model

isAbstract: No

Generalization: FACE_UoPElement, FACE_TraceableElement

Extension: Class

Description

A FACE_UnitOfPortability is a PlatformSpecificComponent or PortableComponent.

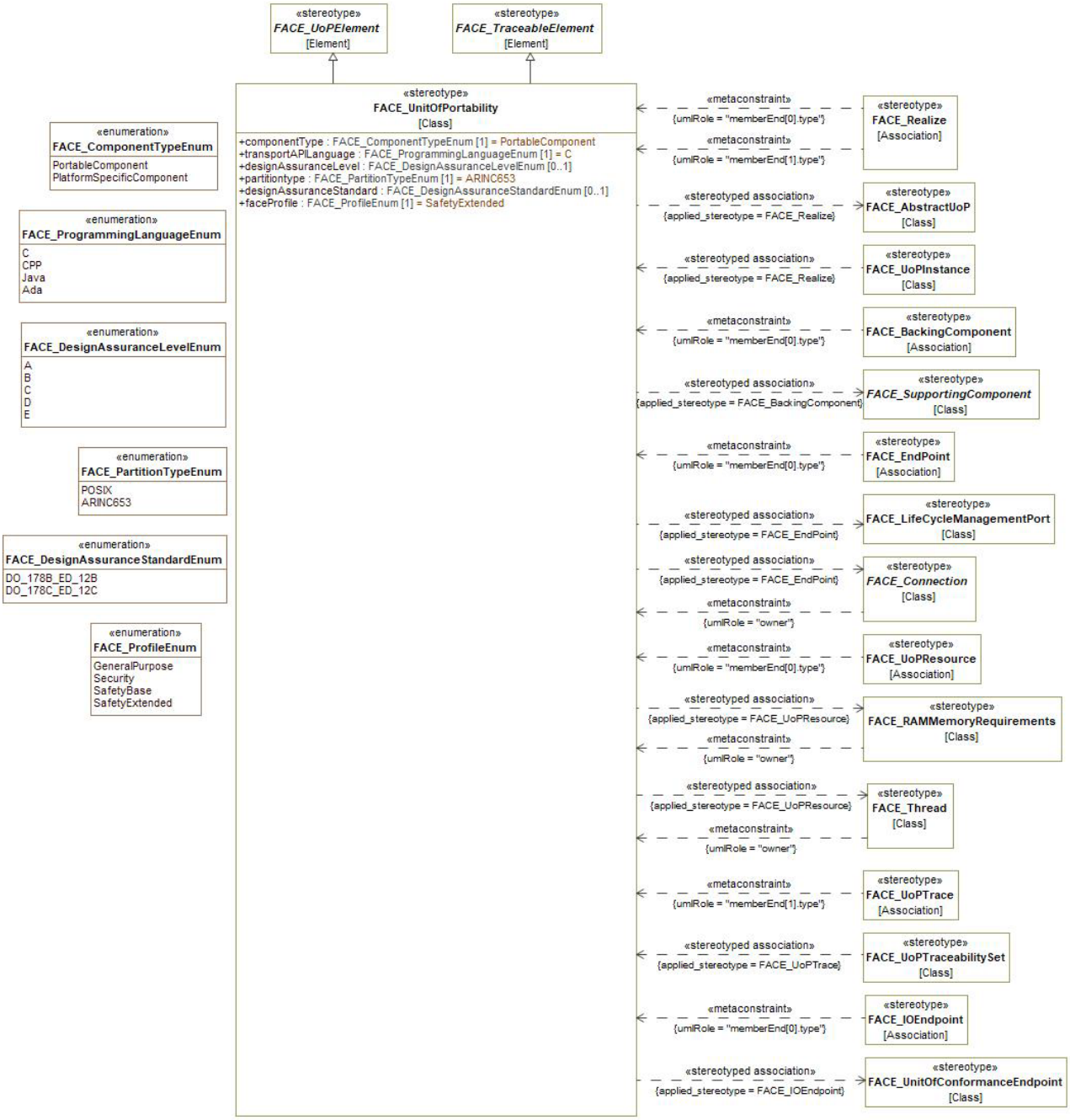

Figure 7-192: FACE_UnitOfPortability

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

C01: 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 UoPMode lthe FACE ArchitectureModel.

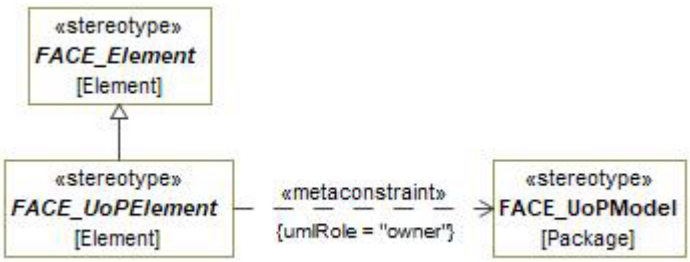

Figure 7-193: abstract FACE_UoPElement

Constraints

C01: 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

C01: FACE_UoPElement.hasUniqueName	All FACE UoP Elements must have a unique name.

FACE_UoPMessageType

Package: UoP Model

isAbstract: Yes

Generalization: FACE_UoPElement, FACE_TraceableElement

Extension: Class

Description

A UoP Message Type is a UoP Template or a UoP CompositeTemplate.

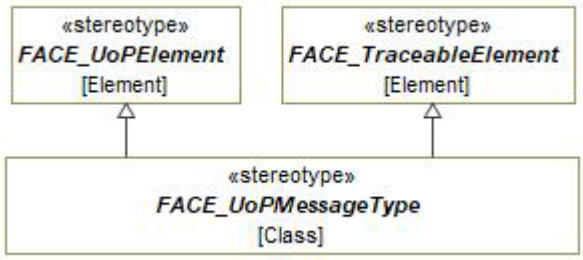

Figure 7-194: abstract FACE_UoPMessageType

FACE Conformance/OCL Constraints

C01: FACE_UoPMessageType.nameIsNotReservedWord	A UoP's Message name may not be an IDL reserved word.

FACE_UoPResource

Package: UoP Model

isAbstract: No

Generalization: FACE_AbstractAssociation

Extension: Association

Description

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

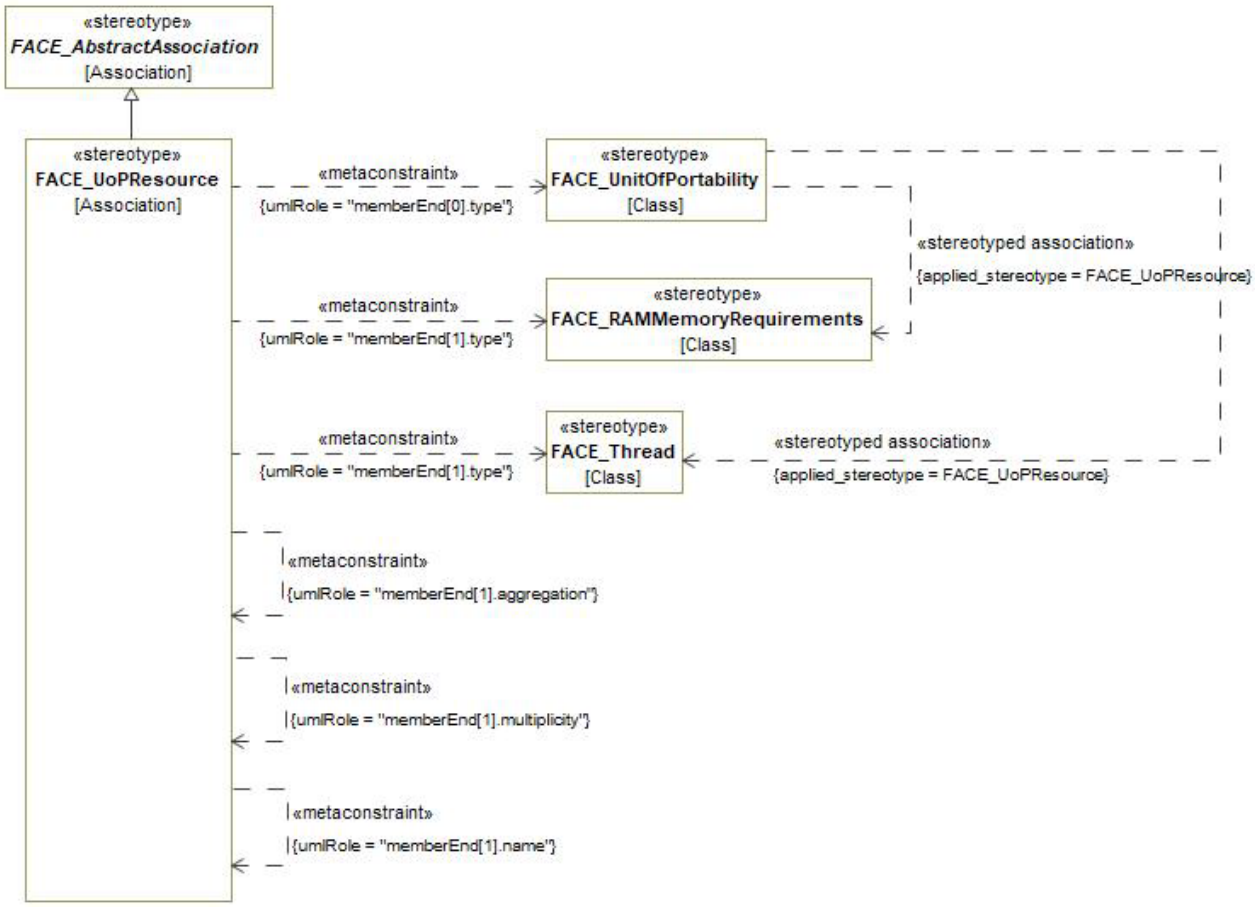

Figure 7-195: FACE_UoPResource

Constraints

C01: FACE_UoPResource.memberEnd[0].type	Value for the memberEnd[0].type metaproperty must be stereotyped by «FACE_UnitOfPortability».
C02: FACE_UoPResource.memberEnd[1].aggregation	memberEnd[1].aggregation shall be composite

C03: 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..*
C04: 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"
C05: 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_Profile::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 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.

FACE_IOEndpoint

Package: 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 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.

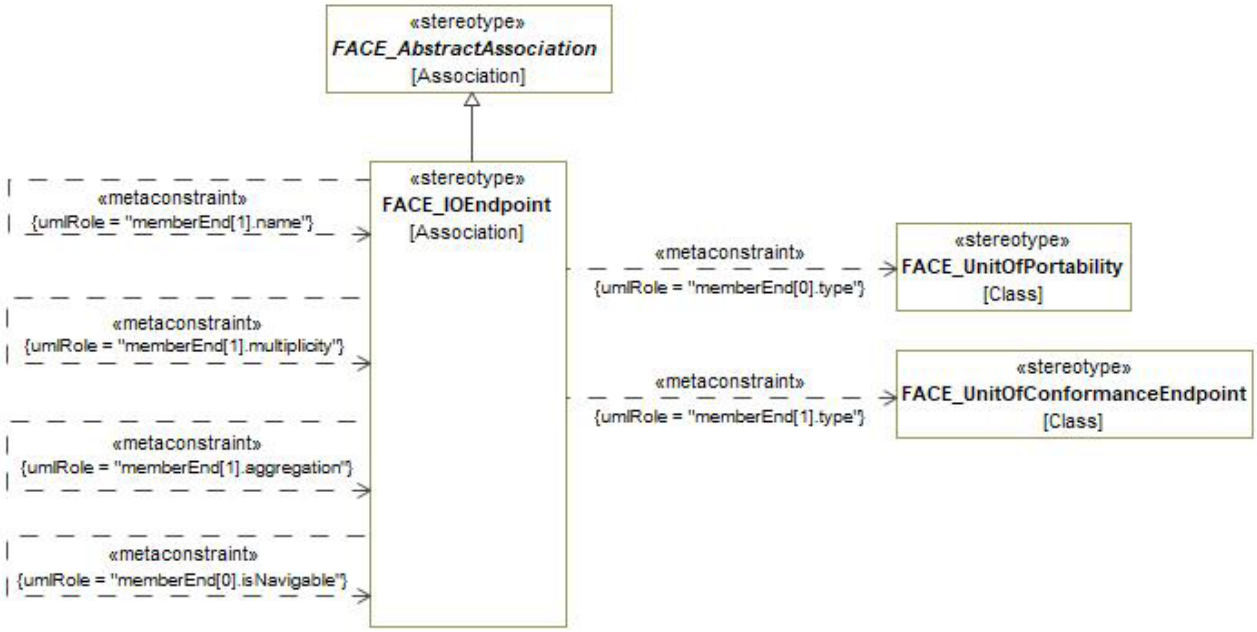

Figure 7-196: FACE_IOEndpoint

Constraints

C01: FACE_IOEndpoint.memberEnd[0].isNavigable	memberEnd[1].isNavigable shall be true
C02: 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.
C03: FACE_IOEndpoint.memberEnd[1].aggregation	memberEnd[1].aggregation shall be composite
C04: FACE_IOEndpoint.memberEnd[1].multiplicity	memberEnd[1].multiplicity shall be 0..*
C05: FACE_IOEndpoint.memberEnd[1].name	memberEnd[1].name shall be "ioEndpoint"
C06: 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: 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.

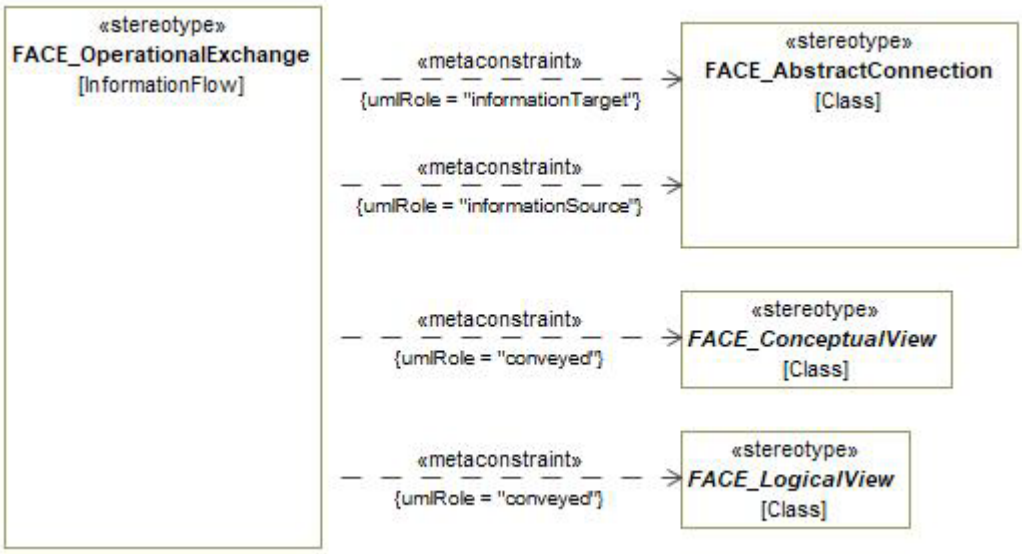

Figure 7-197: FACE_OperationalExchange

Constraints

C01: FACE_OperationalExchange.conveyed	Value for the conveyed metaproperty must be stereotyped by either the specialization of «FACE_ConceptualView» or the specialization of «FACE_LogicalView».
C02: FACE_OperationalExchange.exchangeKind	Value for the exchangeKind attribute defaults to "InformationExchange".
C03: FACE_OperationalExchange.informationSource	Value for the informationSource metaproperty must be stereotyped by «FACE_AbstractConnection».

C04: FACE_OperationalExchange.informationTarget	Value for the informationTarget metaproperty must be stereotyped by «FACE_AbstractConnection».

FACE_ResourceExchange

Package: 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 7-198: FACE_ResourceExchange

Constraints

C01: FACE_ResourceExchange.conveyed	Value for the conveyed metaproperty must be stereotyped by the specialization of «FACE_MessageType».
C02: FACE_ResourceExchange.exchangeKind	Value for the exchangeKind attribute defaults to "FACEResourceCommunication".
C03: 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.
C04: 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: FACE_Extended_Stereotypes

isAbstract: No

Generalization: FACE_UoCElement

Extension: Class

Description

The FACE Technical Standard discusses segments and component Units of Conformance (UoCs) for every segment in the FACE Data Architecture, but the FACE 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 metamodel.

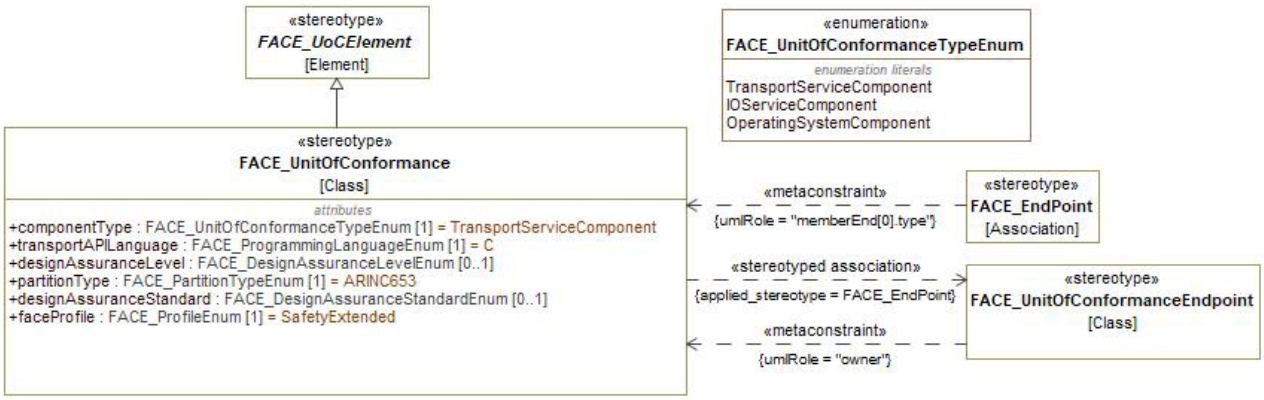

Figure 7-199: FACE_UnitOfConformance

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.
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: FACE_Extended_Stereotypes

isAbstract: No

Extension: Class

Description

The FACE Technical Standard discusses segments and component Units of Conformance (UoCs) but the FACE 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 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.

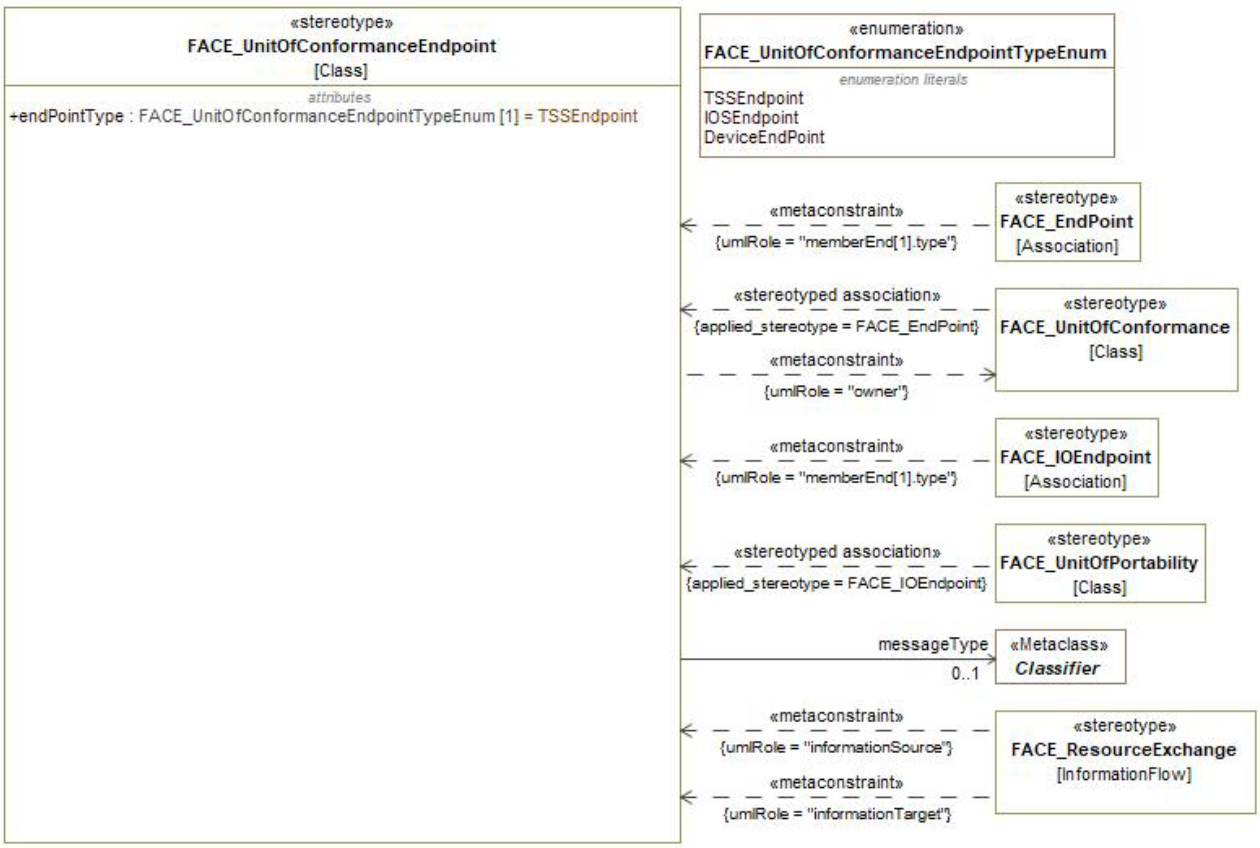

Figure 7-200: FACE_UnitOfConformanceEndpoint

Attributes

endPointType : FACE_UnitOfConformanceEndpointTypeEnum [1]	The component type that corresponds to 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

C01: FACE_UnitOfConformanceEndpoint.owner	Elements with this stereotype may only be contained in (owned by) elements with the stereotype «FACE_UnitOfConformance»

FACE_UnitOfConformanceEndpointTypeEnum

Package: 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 but are not part of the FACE metamodel. Each FACE component has 1 or more connections to other FACE components. The intended FACE segment for that communication is indicated by the 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: FACE_Extended_Stereotypes

isAbstract: No

Description

The FACE Technical Standard discusses segments and component Units of Conformance (UoCs) but the FACE 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 metamodel.

This enumeration represents the FACE component types that are part of the FACE Data Architecture but are not represented in the FACE 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: 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.

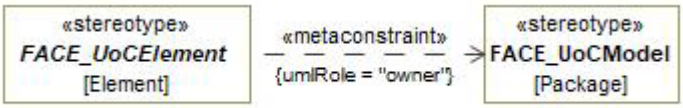

Figure 7-201: abstract FACE_UoCElement

Constraints

C01: 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: FACE_Extended_Stereotypes

isAbstract: No

Extension: Package

Description

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

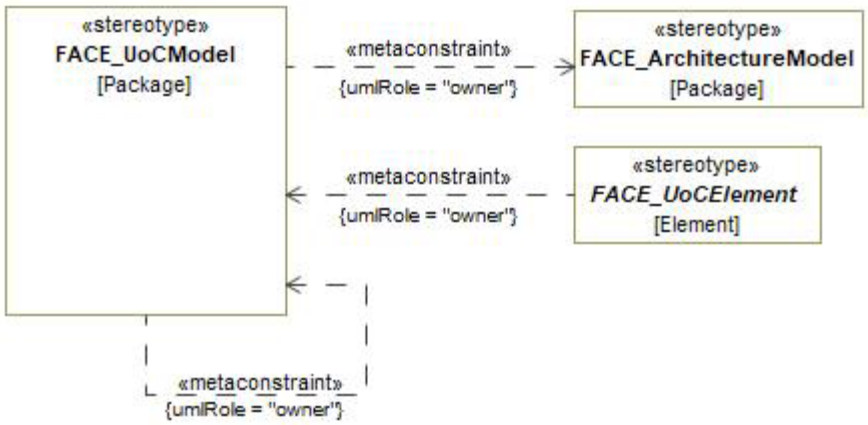

Figure 7-202: FACE_UoCModel

Constraints

C01: FACE_UoCModel.owner	Elements with this stereotype may only be contained in (owned by) elements with the following stereotypes: «FACE_ArchitectureModel» «FACE_UoCModel»

7.1.3 FACE_Profile::UAF_Extensions

This package contains stereotypes for representing FACE elements in a UAF context. The connection between the FACE Profile and UAF is loosely coupled and accomplished using a dependency between FACE elements and UAF elements. The FACE_Implements «stereotyped relationship» dependencies in the stereotype definitions express the correspondence between FACE and UAF metatypes, with additional constraints for the application of FACE stereotypes. The FACE_Implements have been omitted from the FACE element diagrams outside of this section to prevent confusion about the scope of their implementation. These relationships are meant to be implemented only for a separable UAF extension to the FACE Profile.

FACE_Implements

Package: UAF_Extensions

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.

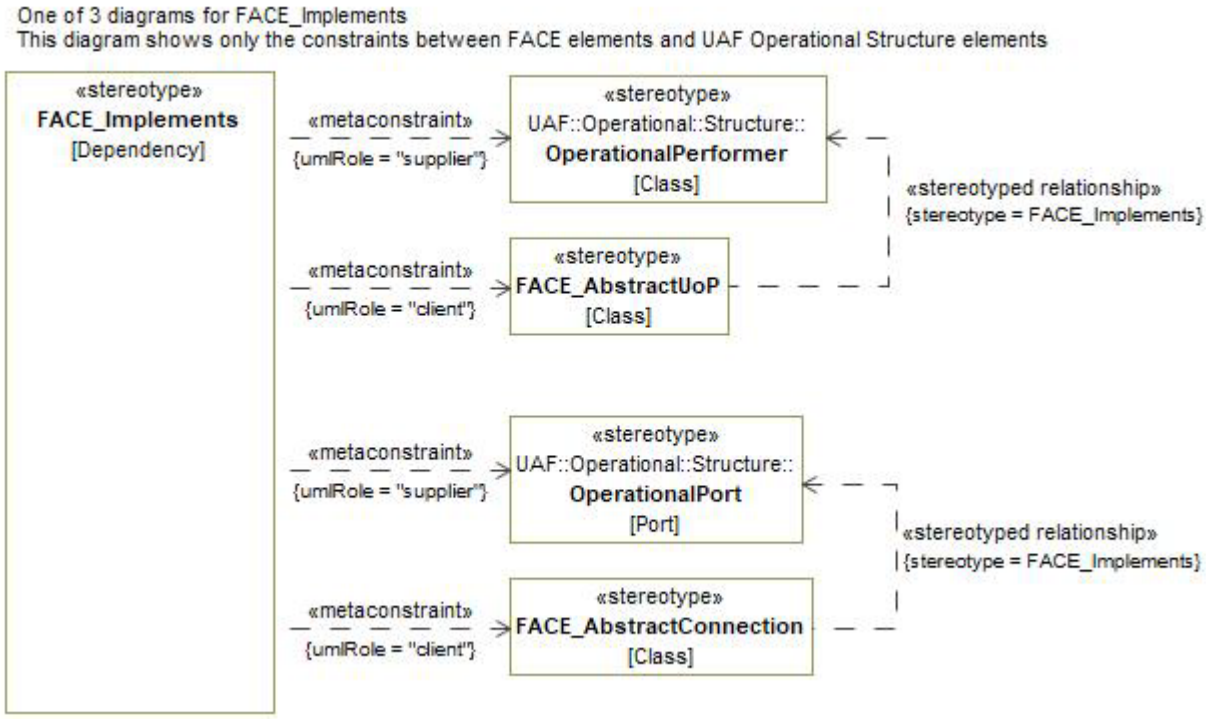

Figure 7-203: FACE_Implements

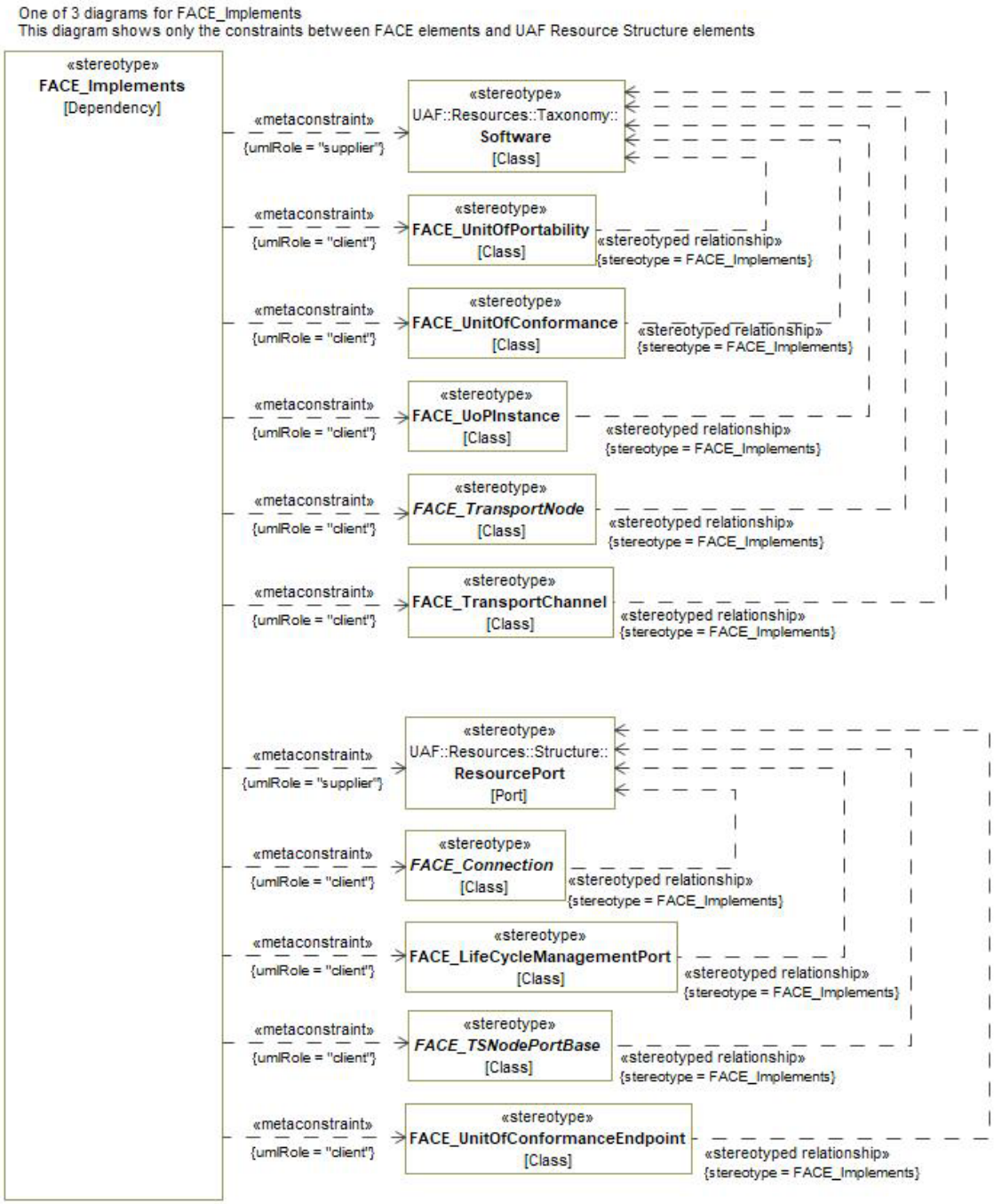

Figure 7-204: FACE_Implements

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

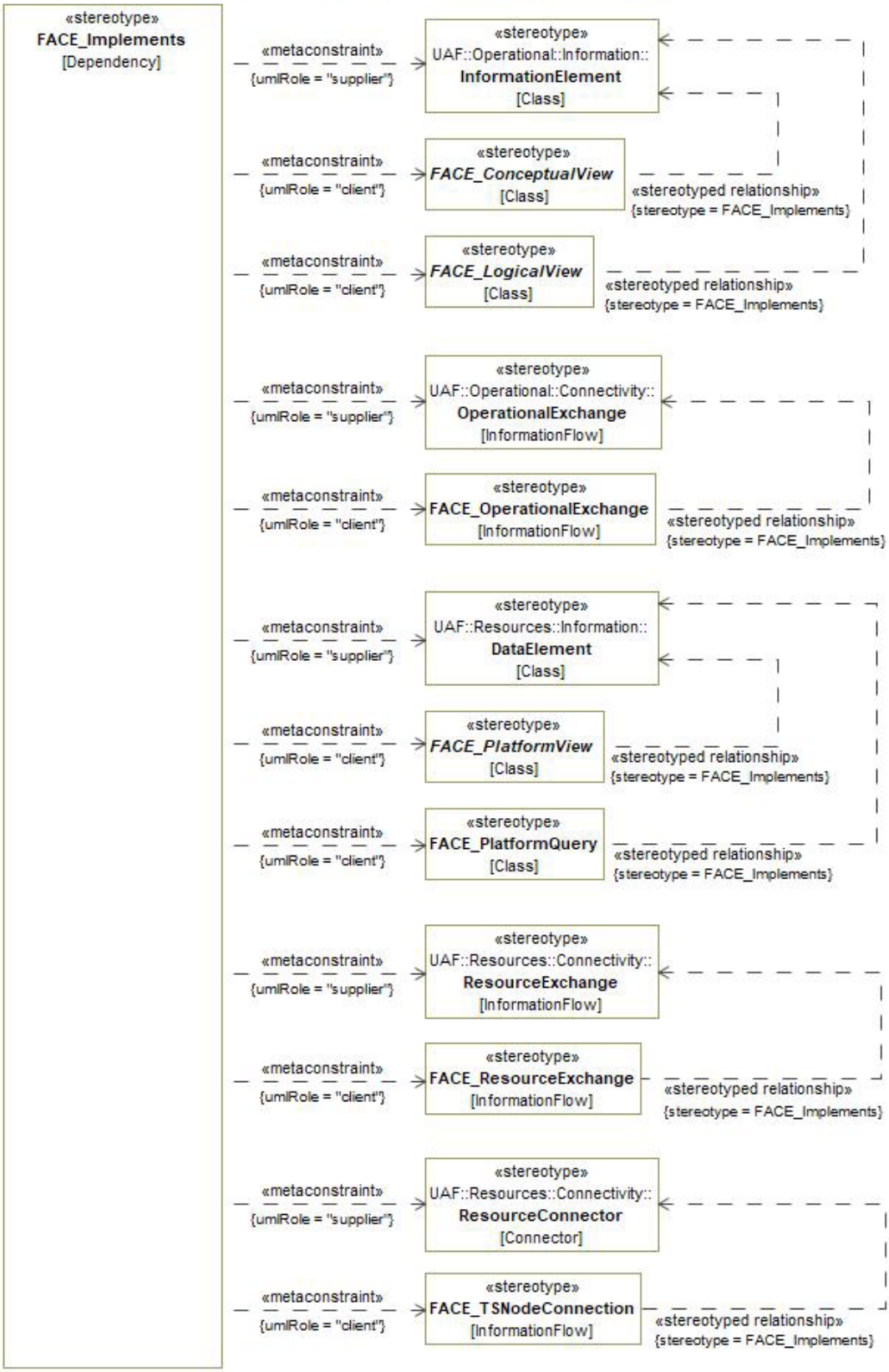

Figure 7-205: FACE_Implements

Constraints

C01: 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» Specializations 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»

C02: 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»

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 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 a UAF 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

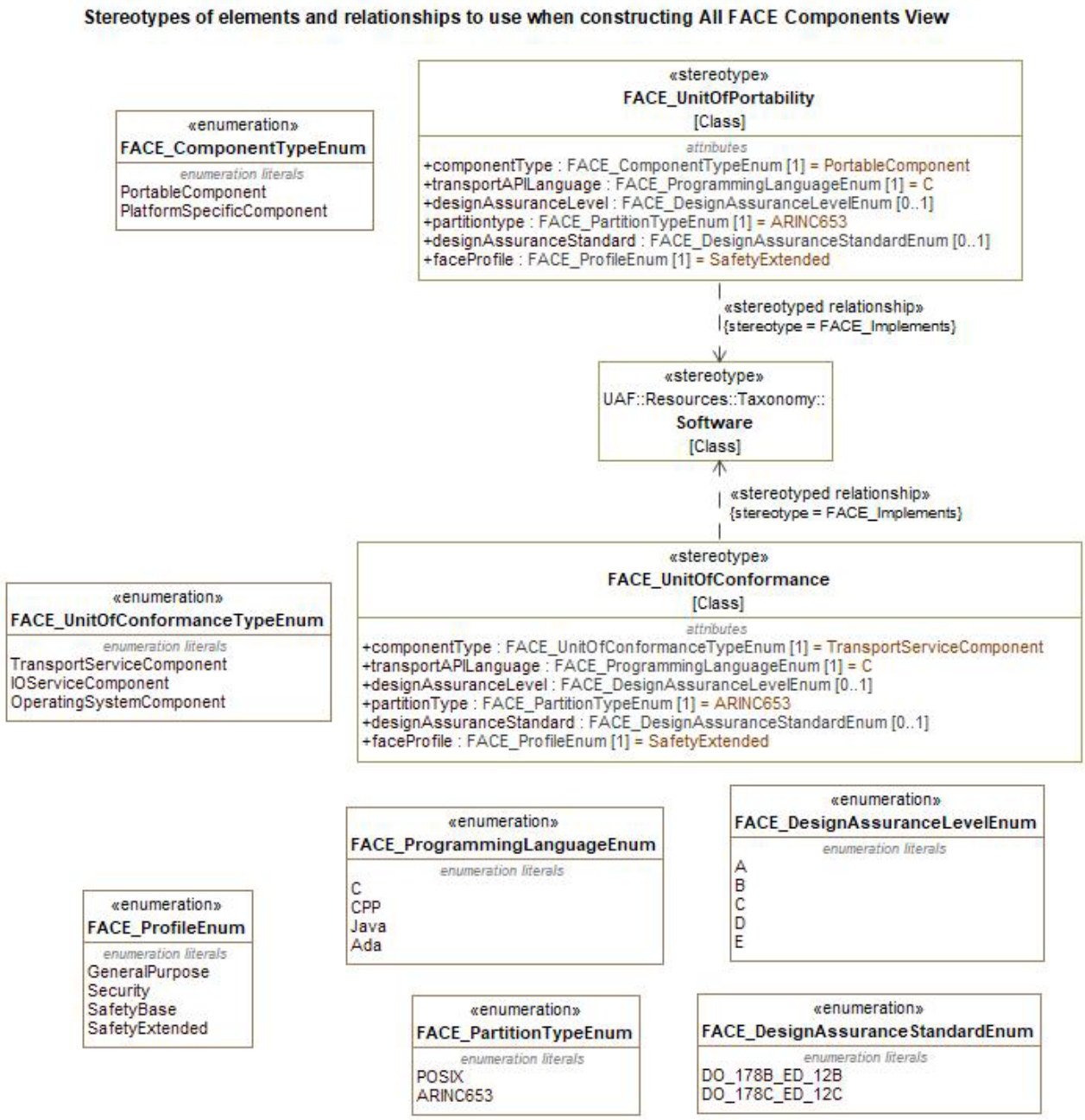

Figure 7-206: All FACE Components View

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 a UAF 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

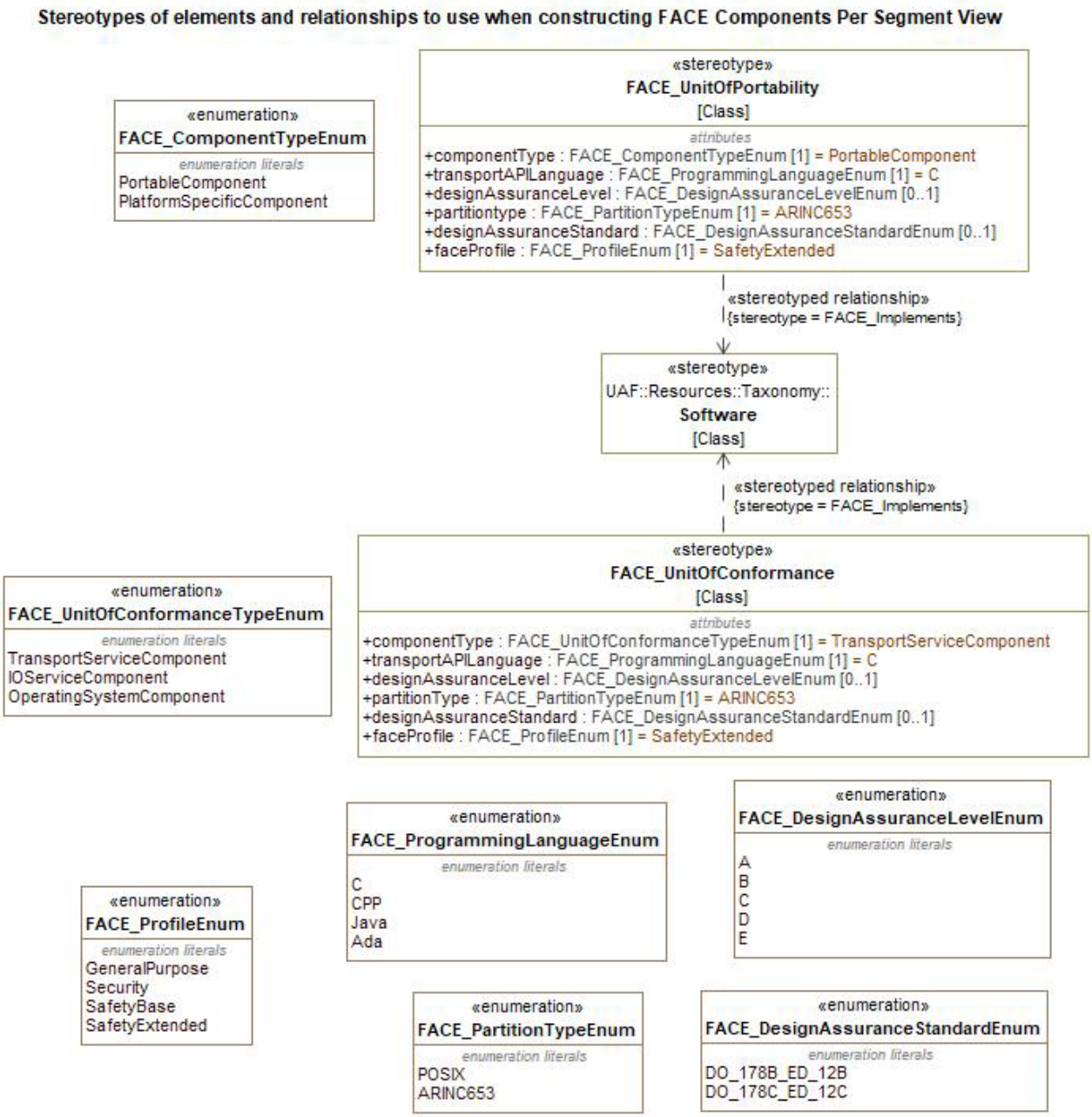

Figure 7-207: FACE Components Per Segment View

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 components identified as part of a UAF architecture
Definition: Shows the connections between abstract FACE Components in a UAF 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

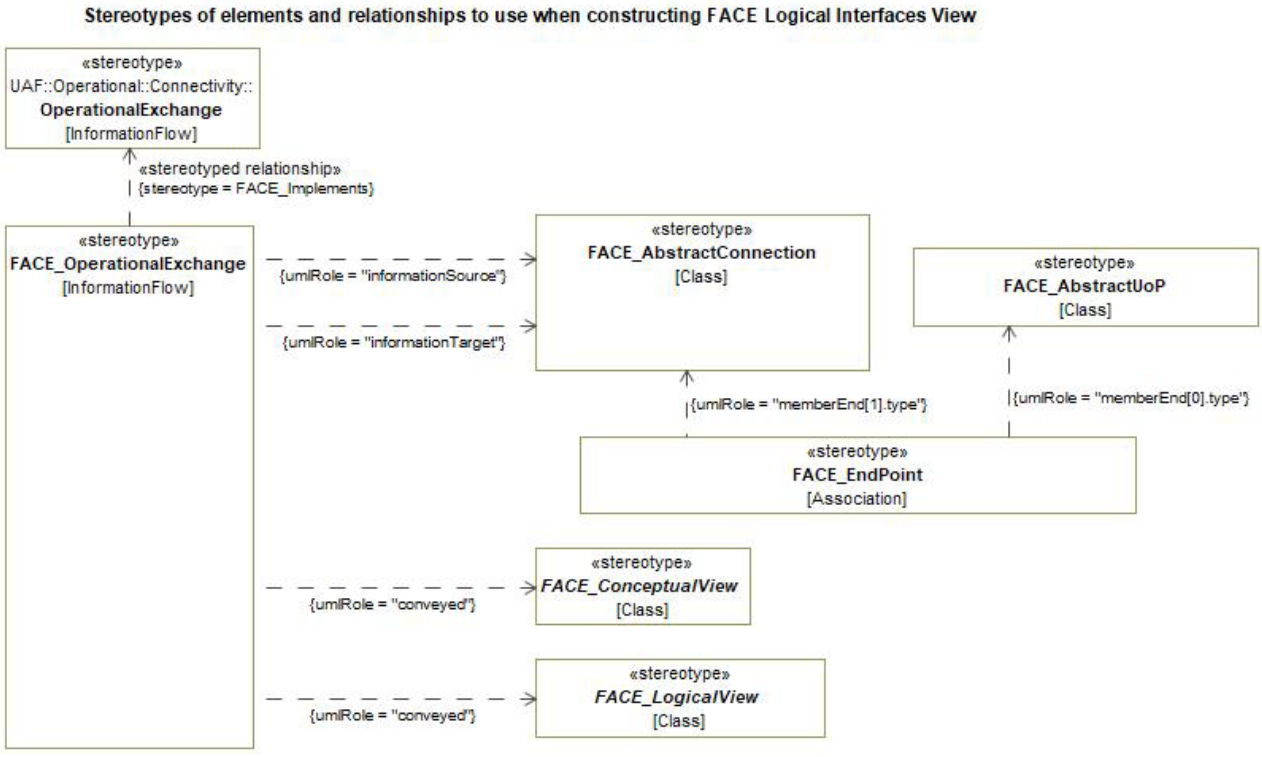

Figure 7-208: FACE Logical Interfaces View

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 identified as part of a UAF architecture
Definition: Shows the connections between FACE Components in a UAF 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

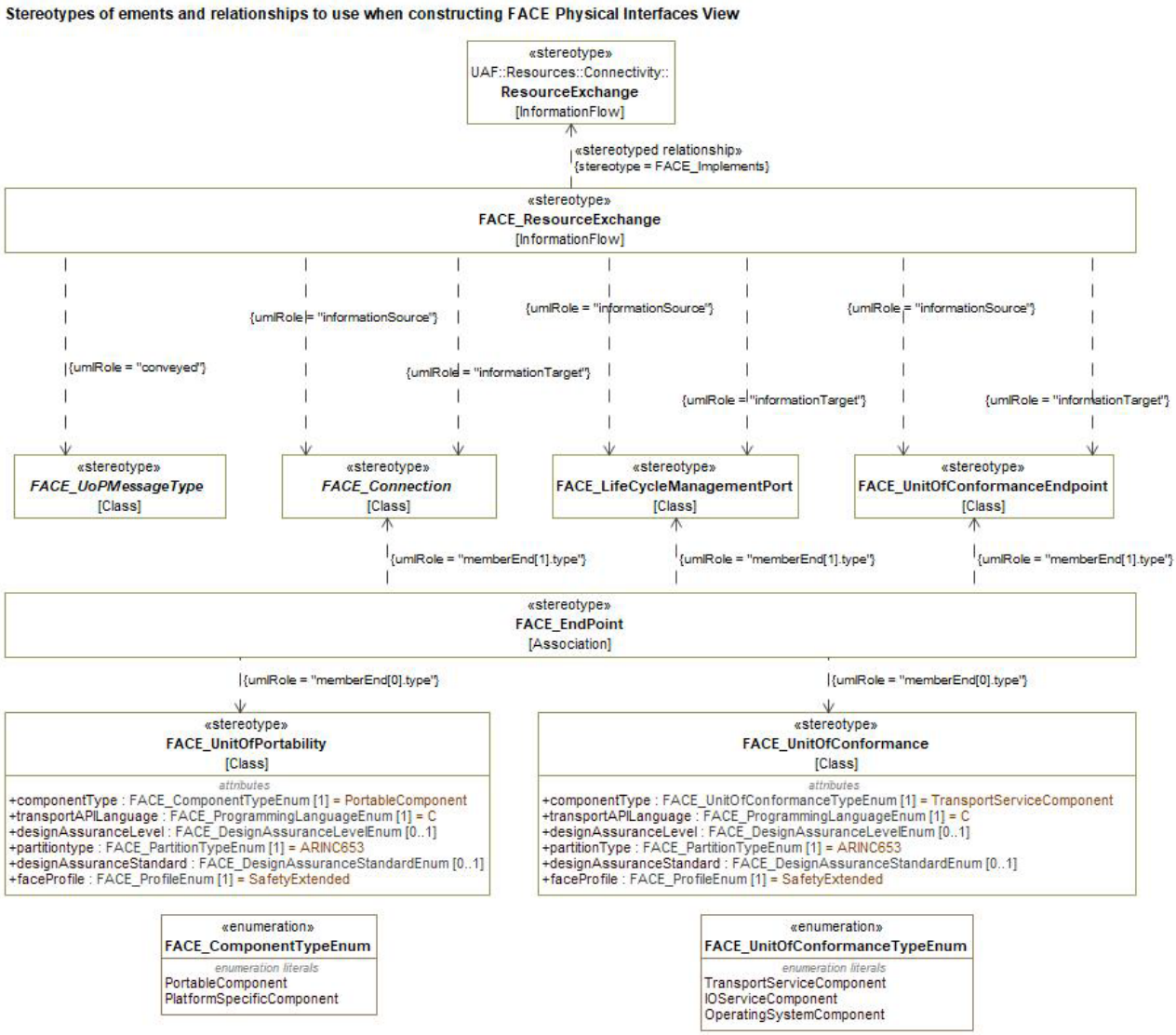

Figure 7-209: FACE Physical Interfaces View

Elements

- FACE_ComponentTypeEnum
- FACE_Connection
- FACE_EndPoint
- FACE_LifeCycleManagementPort
- FACE_ResourceExchange
- FACE_UnitOfConformance
- FACE_UnitOfConformanceEndpoint
- FACE_UnitOfConformanceTypeEnum
- FACE_UnitOfPortability
- FACE_UoPMessageType
- 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 UAF Extensions Enhance Related 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, such as Department of Defense Architecture Framework (DoDAF), The British Ministry of Defence Architecture Framework (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 extensions to the UML portion of the FACE Profile, coupled with the UAF extensions enhance the representation of layered architecture elements and the flow of information throughout a system of systems. AAA Conformance also supports the modeling of data sent and received by avionics components to improve interoperability.

This specification enables the development of tools that make it easier for modelers to create more detailed and accurate models, as well as enable model sharing across the general architecture models.

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

Just as UAF supports systems of systems modeling, additional views for safety and cybersecurity are supported. Because the FACE Profile allows the representation of software components, data models, and integration models, additional cybersecurity modeling Frameworks such as STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege) can be applied with respect to avionics across a system of systems. Likewise, this can facilitate the development of cybersecurity solutions that can be developed conformant with the FACE technical standard in way such that the solution may be applied across disparate air, space, land, and sea platforms.

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 an architecture through the application of the MARTE profile (Modeling and Analysis of Real-Time and Embedded systems). The FACE Profile's UAF extensions 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 FACE Profile enables the development of model-based artifacts to support the Radio Technical Commission for Aeronautics (RTCA) DO-178 (Software Considerations in Airborne Systems and Equipment Certification) and DO-331 (the Model-Based Development and Verification Supplement to DO-178C and DO-278A) used for safety of flight certification by the U.S. Federal Aviation Administration (FAA) and the European Aviation Safety Agency (EASA). 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 with MARTE markings to feed AADL analysis

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.

FACE Profile facilitates the development of models and artifacts to support compliance with security standards such as Standard IEC 62443 - Cybersecurity for Industry, RTCA DO-326A Airworthiness Security Process Specification, and its supplement Airworthiness Security Methods and Considerations. 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, to address tool implementation of aspects the FACE Technical Specification that are outside the bounds of a profile but may be implemented using tool-specific capabilities. It discusses approaches to implementation of Conformance levels AA and AAA described earlier in this specification, as well as implementation of tabular views described above and a recommended inclusion of a FACE segment architecture view for user reference..

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

The application of OCL constraints from the FACE Technical Standard 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 UDDL and FACE Technical Standards.

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 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 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 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 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.

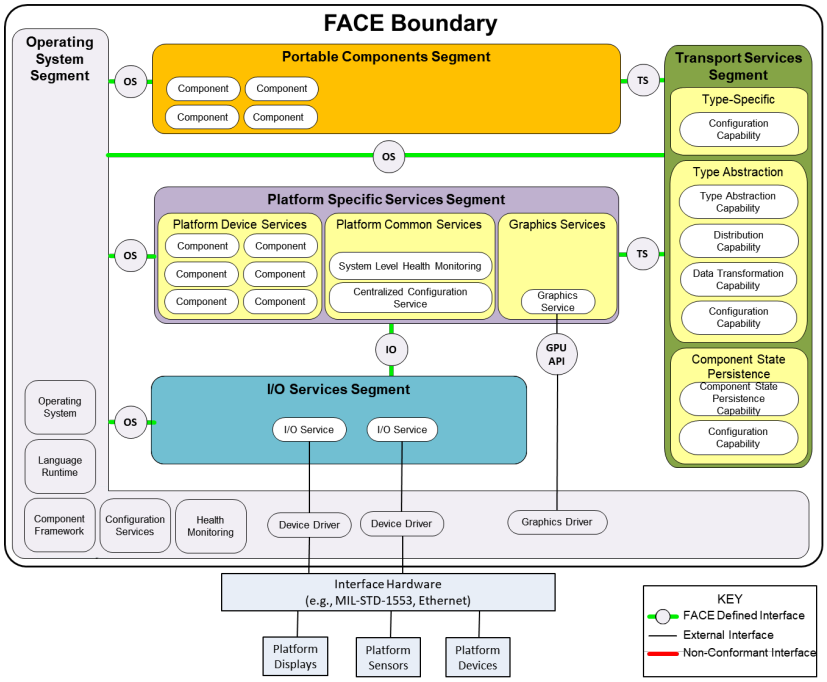

Figure 8-1: FACE Technical Interchange Meeting Architectural Diagram Template Example

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 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. 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 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 path metamodel elements. The two notations (elements and string) are interchangeable using a translation algorithm. The CharacteristicProjection syntax is used in the FACE Profile instead of the corresponding FACE Metamodel elements. XMI exchange mechanisms between models using the FACE Profile 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 elements.

Table A-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 [Association]
face.datamodel.logical	Participant CharacteristicPathNode ParticipantPathNode PathNode	FACE_LogicalParticipant [Association]
face.datamodel.platform	Participant CharacteristicPathNode ParticipantPathNode PathNode	FACE_PlatformParticipant [Association]

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 (with embedded UDDL Standard elements) and their mapping to stereotypes that, in part or whole, realize the metamodel element and its relationships in the FACE Profile. Elements in the face.datamodel package correspond to elements in the UDDL Standard. The order of the elements in the table corresponds to the order of the metamodel elements in the FACE Technical Standard and, by reference from the FACE Technical Standard, the UDDL Standard.

Table A-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_ConceptualComposition [Property] FACE_ConceptualEntity [Class] FACE_EntityBasis [Generalization] FACE_SpecializationOwner [Class] FACE_Specialize [Generalization]
face.datamodel.conceptual	Composition	FACE_ConceptualComposableElement [Element] FACE_ConceptualComposition [Property]
face.datamodel.conceptual	Association	FACE_ConceptualAssociation [Class] FACE_SpecializationOwner [Class] FACE_Specialize [Generalization]
face.datamodel.conceptual	Participant	FACE_ConceptualParticipant [Association]
face.datamodel.conceptual	PathNode	FACE_ConceptualParticipant [Association]
face.datamodel.conceptual	ParticipantPathNode	FACE_ConceptualParticipant [Association]
face.datamodel.conceptual	CharacteristicPathNode	FACE_ConceptualParticipant [Association]
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.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 Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
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_AbstractAssociation [Association] FACE_Axis [Association] FACE_CoordinateSystem [Class]
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_AbstractAssociation [Association] FACE_AppliedConstraint [Association] FACE_Axis [Association] FACE_DefinedReferencePoint [Association] FACE_MeasurementSystem [Class]
face.datamodel.logical	MeasurementSystemAxis	FACE_AbstractAssociation [Association] FACE_AppliedConstraint [Association] FACE_AppliedValueTypeUnit [Association] FACE_MeasurementSystemAxis [Class]
face.datamodel.logical	ReferencePoint	FACE_AbstractAssociation [Association] FACE_RPPart [Association] FACE_ReferencePoint [Class]
face.datamodel.logical	ReferencePointPart	FACE_ReferencePointPart [Class]
face.datamodel.logical	ValueTypeUnit	FACE_AbstractAssociation [Association] FACE_AppliedConstraint [Association] FACE_ValueTypeUnit [Class]
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.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 Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.datamodel.logical	Measurement	FACE_AbstractAssociation [Association] FACE_AppliedConstraint [Association] FACE_Axis [Association] FACE_Measurement [Class] FACE_Realize [Association]
face.datamodel.logical	MeasurementAxis	FACE_AbstractAssociation [Association] FACE_AppliedConstraint [Association] FACE_AppliedValueTypeUnit [Association] FACE_MeasurementAxis [Class] 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_AbstractAssociation [Association] FACE_LogicalComposition [Property] FACE_LogicalEntity [Class] FACE_Realize [Association] FACE_SpecializationOwner [Class] FACE_Specialize [Generalization]
face.datamodel.logical	Composition	FACE_LogicalComposableElement [Element] FACE_LogicalComposition [Property]
face.datamodel.logical	Association	FACE_AbstractAssociation [Association] FACE_LogicalAssociation [Class] FACE_Realize [Association] FACE_SpecializationOwner [Class] FACE_Specialize [Generalization]
face.datamodel.logical	Participant	FACE_LogicalParticipant [Association]
face.datamodel.logical	PathNode	FACE_LogicalParticipant [Association]
face.datamodel.logical	ParticipantPathNode	FACE_LogicalParticipant [Association]
face.datamodel.logical	CharacteristicPathNode	FACE_LogicalParticipant [Association]
face.datamodel.logical	View	FACE_LogicalView [Class]
face.datamodel.logical	Query	FACE_AbstractAssociation [Association] FACE_LogicalQuery [Class] FACE_Realize [Association]
face.datamodel.logical	CompositeQuery	FACE_AbstractAssociation [Association] FACE_LogicalCompositeQuery [Class] FACE_LogicalQueryComposition [Property] FACE_Realize [Association]
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	PlatformDataType	FACE_AbstractAssociation [Association] FACE_PlatformDataType [Element] FACE_Realize [Association]
face.datamodel.platform	Primitive	FACE_Primitive [Class]
face.datamodel.platform	Boolean	FACE_Boolean [Class]
face.datamodel.platform	Octet	FACE_Octet [Class]
face.datamodel.platform	CharType	FACE_CharType [Class]

FACE Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.datamodel.platform	Char	FACE_Char [Class]
face.datamodel.platform	StringType	FACE_StringType [Class]
face.datamodel.platform	String	FACE_String [Class]
face.datamodel.platform	BoundedString	FACE_BoundedString [Class]
face.datamodel.platform	CharArray	FACE_CharArray [Class]
face.datamodel.platform	Enumeration	FACE_Enumeration [Class]
face.datamodel.platform	Number	FACE_Number [Class]
face.datamodel.platform	Integer	FACE_Integer [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	Real	FACE_Real [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	UnsignedInteger	FACE_UnsignedInteger [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	Sequence	FACE_Sequence [Class]
face.datamodel.platform	Array	FACE_Array [Class]
face.datamodel.platform	Struct	FACE_Struct [Class]
face.datamodel.platform	StructMember	FACE_StructMember [Property]
face.datamodel.platform	Characteristic	FACE_PlatformCharacteristic [Element]
face.datamodel.platform	Entity	FACE_AbstractAssociation [Association] FACE_PlatformComposition [Property] FACE_PlatformEntity [Class] FACE_Realize [Association] FACE_SpecializationOwner [Class] FACE_Specialize [Generalization]
face.datamodel.platform	Composition	FACE_PlatformComposableElement [Element] FACE_PlatformComposition [Property]
face.datamodel.platform	Association	FACE_AbstractAssociation [Association] FACE_PlatformAssociation [Class] FACE_Realize [Association] FACE_SpecializationOwner [Class] FACE_Specialize [Generalization]
face.datamodel.platform	Participant	FACE_PlatformParticipant [Association]
face.datamodel.platform	PathNode	FACE_PlatformParticipant [Association]
face.datamodel.platform	ParticipantPathNode	FACE_PlatformParticipant [Association]
face.datamodel.platform	CharacteristicPathNode	FACE_PlatformParticipant [Association]
face.datamodel.platform	View	FACE_PlatformView [Class]
face.datamodel.platform	Query	FACE_AbstractAssociation [Association] FACE_PlatformQuery [Class] FACE_Realize [Association]
face.datamodel.platform	CompositeQuery	FACE_AbstractAssociation [Association] FACE_PlatformCompositeQuery [Class] FACE_PlatformQueryComposition [Property] FACE_Realize [Association]

FACE Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.datamodel.platform	QueryComposition	FACE_PlatformQueryComposition [Property] FACE_PlatformView [Class]
face	UoPModel	FACE_UoPModel [Package]
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.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_AbstractAssociation [Association] FACE_AbstractConnection [Class] FACE_AbstractView [Association]
face.uop	UnitOfPortability	FACE_AbstractAssociation [Association] FACE_BackingComponent [Association] FACE_ComponentTypeEnum FACE_DesignAssuranceLevelEnum FACE_DesignAssuranceStandardEnum FACE_EndPoint [Association] FACE_PartitionTypeEnum FACE_ProfileEnum FACE_ProgrammingLanguageEnum FACE_Realize [Association] FACE_UnitOfPortability [Class] FACE_UoPResource [Association]
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_AbstractAssociation [Association] FACE_Connection [Class] FACE_Realize [Association]
face.uop	ClientServerConnection	FACE_AbstractAssociation [Association] FACE_ClientServerConnection [Class] FACE_RequestView [Association] FACE_ResponseView [Association]
face.uop	PubSubConnection	FACE_AbstractAssociation [Association] FACE_MessageExchangeTypeEnum FACE_MessageType [Association] FACE_PubSubConnection [Class]

FACE Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.uop	QueuingConnection	FACE_QueuingConnection [Class]
face.uop	SingleInstanceMessageConnection	FACE_SingleInstanceMessageConnection [Class]
face.uop	LifeCycleManagementPort	FACE_AbstractAssociation [Association] FACE_LifeCycleManagementPort [Class] FACE_MessageType [Association]
face.uop	MessageType	FACE_UoPMessageType [Class]
face.uop	CompositeTemplate	FACE_AbstractAssociation [Association] FACE_CompositeTemplate [Class] FACE_Realize [Association] FACE_TemplateComposition [Property]
face.uop	TemplateComposition	FACE_TemplateComposition [Property] FACE_UoPMessageType [Class]
face.uop	Template	FACE_AbstractAssociation [Association] FACE_BoundQuery [Association] FACE_EffectiveQuery [Association] FACE_Template [Class]
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_TSNodeConnection [InformationFlow] FACE_TSNodePortBase [Class]
face.integration	UoPInstance	FACE_AbstractAssociation [Association] FACE_EndPoint [Association] FACE_Realize [Association] FACE_UoPInstance [Class]
face.integration	UoPEndPoint	FACE_AbstractAssociation [Association] FACE_Realize [Association] FACE_UoPEndPoint [Class]
face.integration	UoPInputEndPoint	FACE_UoPInputEndPoint [Class]
face.integration	UoPOutputEndPoint	FACE_UoPOutputEndPoint [Class]
face.integration	TransportNode	FACE_AbstractAssociation [Association] FACE_EndPoint [Association] FACE_TransportNode [Class]
face.integration	TSNodePort	FACE_AbstractAssociation [Association] FACE_MessageType [Association] FACE_TSNodePort [Class]
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 Metamodel Package	FACE Metamodel Element Name	FACE Profile Stereotype(s)
face.traceability	Element	FACE_Connection [Class] FACE_TraceabilityElement [Element]
face.traceability	TraceableElement	FACE_AbstractAssociation [Association] FACE_ElementTrace [Association] FACE_TraceableElement [Element]
face.traceability	TraceabilityPoint	FACE_TraceabilityPoint [Class]
face.traceability	UoPTraceabilitySet	FACE_AbstractAssociation [Association] FACE_UoPTrace [Association] FACE_UoPTraceabilitySet [Class]
face.traceability	ConnectionTraceabilitySet	FACE_AbstractAssociation [Association] FACE_ConnectionTrace [Association] FACE_ConnectionTraceabilitySet [Class]
face.traceability	ConceptualEntityTrace	FACE_AbstractAssociation [Association] FACE_ConceptualEntityTrace [Class] FACE_TraceEntity [Association]
face.traceability	ConceptualViewTrace	FACE_AbstractAssociation [Association] FACE_ConceptualViewTrace [Class] FACE_TraceView [Association]
face.traceability	LogicalEntityTrace	FACE_AbstractAssociation [Association] FACE_LogicalEntityTrace [Class] FACE_TraceEntity [Association]
face.traceability	LogicalViewTrace	FACE_AbstractAssociation [Association] FACE_LogicalViewTrace [Class] FACE_TraceView [Association]
face.traceability	PlatformEntityTrace	FACE_AbstractAssociation [Association] FACE_PlatformEntityTrace [Class] FACE_TraceEntity [Association]
face.traceability	PlatformViewTrace	FACE_AbstractAssociation [Association] FACE_PlatformViewTrace [Class] FACE_TraceView [Association]
Not from the Metamodel, created for System-of-Systems	<Derived from FACE Technical Standard>	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 System-of-Systems	<Created for System-of-Systems Connectivity>	FACE_OperationalExchange [InformationFlow] FACE_ResourceExchange [InformationFlow]
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 elements to their corresponding FACE and UDDL 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 specification. The FACE metamodel elements shown are realized in whole or part by the listed FACE Profile element. The UAF element shown represents the mapping from the FACE Profile 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 A-3 FACE Profile Elements -to- FACE Metamodel Mappings

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

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.CompositeQuery face.datamodel.logical.Entity face.datamodel.logical.Measurement face.datamodel.logical.MeasurementAxis face.datamodel.logical.Query face.datamodel.platform.Association face.datamodel.platform.CompositeQuery face.datamodel.platform.Entity face.datamodel.platform.PlatformDataType face.datamodel.platform.Query face.integration.UoPEndPoint face.integration.UoPInstance face.uop.CompositeTemplate face.uop.Connection face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture	FACE_TraceabilityModel	Package	face.TraceabilityModel	
FACE_Profile.FACE Data Architecture	FACE_UoPModel	Package	face.UoPModel	

FACE_Profile.FACE Data Architecture	FACE_AbstractAssociation	Association	face.datamodel.logical.Association face.datamodel.logical.CompositeQuery face.datamodel.logical.CoordinateSystem face.datamodel.logical.Entity face.datamodel.logical.Measurement face.datamodel.logical.MeasurementAxis face.datamodel.logical.MeasurementSystem face.datamodel.logical.MeasurementSystemAxis face.datamodel.logical.Query face.datamodel.logical.ReferencePoint face.datamodel.logical.ValueTypeUnit face.datamodel.platform.Association face.datamodel.platform.CompositeQuery face.datamodel.platform.Entity face.datamodel.platform.PlatformDataType face.datamodel.platform.Query face.integration.TSNodePort face.integration.TransportNode face.integration.UoPEndPoint face.integration.UoPInstance face.traceability.ConceptualEntityTrace face.traceability.ConceptualViewTrace face.traceability.ConnectionTraceabilitySet face.traceability.LogicalEntityTrace face.traceability.LogicalViewTrace face.traceability.PlatformEntityTrace face.traceability.PlatformViewTrace face.traceability.TraceableElement face.traceability.UoPTraceabilitySet face.uop.AbstractConnection face.uop.ClientServerConnection face.uop.CompositeTemplate face.uop.Connection face.uop.LifeCycleManagementPort face.uop.PubSubConnection face.uop.Template face.uop.UnitOfPortability	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.FACE Data Model	FACE_ConceptualDataModel	Package	face.datamodel.ConceptualDataModel	
FACE_Profile.FACE Data Architecture.FACE Data Model	FACE_DataModelElement	Element	face.datamodel.Element	
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_Specialize	Generalization	face.datamodel.conceptual.Association face.datamodel.conceptual.Entity face.datamodel.logical.Association face.datamodel.logical.Entity face.datamodel.platform.Association face.datamodel.platform.Entity	
FACE_Profile.FACE Data Architecture.FACE Data Model	FACE_SpecializationOwner	Class	face.datamodel.conceptual.Association face.datamodel.conceptual.Entity face.datamodel.logical.Association face.datamodel.logical.Entity face.datamodel.platform.Association face.datamodel.platform.Entity	
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	
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 Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
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	Association	face.datamodel.conceptual.CharacteristicPathNode face.datamodel.conceptual.Participant face.datamodel.conceptual.ParticipantPathNode face.datamodel.conceptual.PathNode	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualQuery	Class	face.datamodel.conceptual.Query	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualQueryComposition	Property	face.datamodel.conceptual.CompositeQuery face.datamodel.conceptual.QueryComposition	
FACE_Profile.FACE Data Architecture.FACE Data Model.ConceptualDataModel	FACE_ConceptualView	Class	face.datamodel.conceptual.QueryComposition face.datamodel.conceptual.View	InformationElement [Class]
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 Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
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.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	
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	Association	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	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_LogicalQueryComposition	Property	face.datamodel.logical.CompositeQuery face.datamodel.logical.QueryComposition	
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.QueryComposition face.datamodel.logical.View	InformationElement [Class]
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_MeasurementConversion	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_MeasurementSystemAxis	Class	face.datamodel.logical.MeasurementSystemAxis	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_MeasurementSystemConversion	Class	face.datamodel.logical.MeasurementSystemConversion	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
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	Enumeration	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.datamodel.logical.ValueType	
FACE_Profile.FACE Data Architecture.FACE Data Model.LogicalDataModel	FACE_ValueTypeUnit	Class	face.datamodel.logical.ValueTypeUnit	
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_Char	Class	face.datamodel.platform.Char	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_CharType	Class	face.datamodel.platform.CharType	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
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_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_Array	Class	face.datamodel.platform.Array	
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_CharArray	Class	face.datamodel.platform.CharArray	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_StructMember	Property	face.datamodel.platform.StructMember	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Integer	Class	face.datamodel.platform.Integer	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Number	Class	face.datamodel.platform.Number	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Primitive	Class	face.datamodel.platform.Primitive	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Real	Class	face.datamodel.platform.Real	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Sequence	Class	face.datamodel.platform.Sequence	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_Struct	Class	face.datamodel.platform.Struct	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformDataType	Element	face.datamodel.platform.PlatformDataType	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_UnsignedInteger	Class	face.datamodel.platform.UnsignedInteger	
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 Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
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_PlatformAssociation	Class	face.datamodel.platform.Association	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformCharacteristic	Element	face.datamodel.platform.Characteristic	
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	Association	face.datamodel.platform.CharacteristicPathNode face.datamodel.platform.Participant face.datamodel.platform.ParticipantPathNode face.datamodel.platform.PathNode	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformQuery	Class	face.datamodel.platform.Query	DataElement [Class]
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformView	Class	face.datamodel.platform.QueryComposition face.datamodel.platform.View	DataElement [Class]
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_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.FACE Data Model.PlatformDataModel	FACE_PlatformCompositeQuery	Class	face.datamodel.platform.CompositeQuery	
FACE_Profile.FACE Data Architecture.FACE Data Model.PlatformDataModel	FACE_PlatformQueryComposition	Property	face.datamodel.platform.CompositeQuery face.datamodel.platform.QueryComposition	
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]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TransportNode	Class	face.integration.IntegrationContext face.integration.TransportNode	Software [Class]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TSNodeConnection	InformationFlow	face.integration.IntegrationContext face.integration.TSNodeConnection face.integration.TSNodePortBase	ResourceConnector [Connector]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TSNodeInputPort	Class	face.integration.TSNodeInputPort	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TSNodeOutputPort	Class	face.integration.TSNodeOutputPort	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TSNodePort	Class	face.integration.TSNodePort	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_TSNodePortBase	Class	face.integration.TSNodePortBase	ResourcePort [Port]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_UoPEndPoint	Class	face.integration.UoPEndPoint	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_UoPInputEndPoint	Class	face.integration.UoPInputEndPoint	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_UoPInstance	Class	face.integration.UoPInstance	Software [Class]
FACE_Profile.FACE Data Architecture.Integration Model	FACE_UoPOutputEndPoint	Class	face.integration.UoPOutputEndPoint	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewAggregation	Class	face.integration.ViewAggregation	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewFilter	Class	face.integration.ViewFilter	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewSink	Class	face.integration.ViewSink	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewSource	Class	face.integration.ViewSource	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewTransformation	Class	face.integration.ViewTransformation	
FACE_Profile.FACE Data Architecture.Integration Model	FACE_ViewTransporter	Class	face.integration.ViewTransporter	
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_ElementTrace	Association	face.traceability.TraceableElement	
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.Traceability Model	FACE_ConceptualEntityTrace	Class	face.traceability.ConceptualEntityTrace	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_ConceptualViewTrace	Class	face.traceability.ConceptualViewTrace	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_LogicalEntityTrace	Class	face.traceability.LogicalEntityTrace	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_LogicalViewTrace	Class	face.traceability.LogicalViewTrace	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_PlatformEntityTrace	Class	face.traceability.PlatformEntityTrace	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_PlatformViewTrace	Class	face.traceability.PlatformViewTrace	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_TraceEntity	Association	face.traceability.ConceptualEntityTrace face.traceability.LogicalEntityTrace face.traceability.PlatformEntityTrace	
FACE_Profile.FACE Data Architecture.Traceability Model	FACE_TraceView	Association	face.traceability.ConceptualViewTrace face.traceability.LogicalViewTrace face.traceability.PlatformViewTrace	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_AbstractConnection	Class	face.uop.AbstractConnection	OperationalPort [Port]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_AbstractUoP	Class	face.uop.AbstractUoP	OperationalPerformer [Class]
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_BoundQuery	Association	face.uop.Template	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ClientServerConnection	Class	face.uop.ClientServerConnection	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ClientServerRoleEnum	Enumeration	face.uop.ClientServerRole	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ComponentFramework	Class	face.uop.ComponentFramework	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ComponentTypeEnum	Enumeration	face.uop.PlatformSpecificComponent face.uop.PortableComponent face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_CompositeTemplate	Class	face.uop.CompositeTemplate	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_Connection	Class	face.traceability.Element face.uop.Connection	ResourcePort [Port]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_DesignAssuranceLevelEnum	Enumeration	face.uop.DesignAssuranceLevel face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_DesignAssuranceStandardEnum	Enumeration	face.uop.DesignAssuranceStandard face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_EffectiveQuery	Association	face.uop.Template	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ProfileEnum	Enumeration	face.uop.FaceProfile face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_LanguageRunTime	Class	face.uop.LanguageRunTime	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_LifeCycleManagementPort	Class	face.uop.LifeCycleManagementPort	ResourcePort [Port]
FACE_Profile.FACE Data Architecture.UoP Model	FACE_MessageExchangeTypeEnum	Enumeration	face.uop.MessageExchangeType face.uop.PubSubConnection	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_PartitionTypeEnum	Enumeration	face.uop.PartitionType face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ProgrammingLanguageEnum	Enumeration	face.uop.ProgrammingLanguage face.uop.UnitOfPortability	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_PubSubConnection	Class	face.uop.PubSubConnection	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_QueuingConnection	Class	face.uop.QueuingConnection	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_RAMMemoryRequirements	Class	face.uop.RAMMemoryRequirements	
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.FACE Data Architecture.UoP Model	FACE_SingleInstanceMessageConnection	Class	face.uop.SingleInstanceMessageConnection	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_SupportingComponent	Class	face.uop.SupportingComponent	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_SynchronizationStyleEnum	Enumeration	face.uop.SynchronizationStyle	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_Template	Class	face.uop.Template	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_TemplateComposition	Property	face.uop.CompositeTemplate face.uop.TemplateComposition	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_Thread	Class	face.uop.Thread	
FACE_Profile.FACE Data Architecture.UoP Model	FACE_ThreadTypeEnum	Enumeration	face.uop.ThreadType	

FACE Profile Package	Profile Element Name	Metaclass	FACE Metamodel Element(s)	UAF Mapping
FACE_Profile.FACE Data Architecture.UoP Model	FACE_UnitOfPortability	Class	face.uop.PlatformSpecificComponent face.uop.PortableComponent face.uop.UnitOfPortability	Software [Class]
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.FACE Data Architecture.UoP Model	FACE_UoPMessageType	Class	face.uop.MessageType face.uop.TemplateComposition	
FACE_Profile.FACE_Extended_Stereotypes	FACE_IOEndpoint	Association	Not from the Metamodel, created for System-of-Systems.<Derived from FACE Technical Standard>	
FACE_Profile.FACE_Extended_Stereotypes	FACE_OperationalExchange	InformationFlow	Not from the Metamodel, created for System-of-Systems.<Created for System-of-Systems Connectivity>	OperationalExchange [InformationFlow]
FACE_Profile.FACE_Extended_Stereotypes	FACE_ResourceExchange	InformationFlow	Not from the Metamodel, created for System-of-Systems.<Created for System-of-Systems Connectivity>	ResourceExchange [InformationFlow]
FACE_Profile.FACE_Extended_Stereotypes	FACE_UnitOfConformance	Class	Not from the Metamodel, created for System-of-Systems.<Derived from FACE Technical Standard>	Software [Class]
FACE_Profile.FACE_Extended_Stereotypes	FACE_UnitOfConformanceEndpoint	Class	Not from the Metamodel, created for System-of-Systems.<Derived from FACE Technical Standard>	ResourcePort [Port]
FACE_Profile.FACE_Extended_Stereotypes	FACE_UnitOfConformanceEndpointTypeEnum	Enumeration	Not from the Metamodel, created for System-of-Systems.<Derived from FACE Technical Standard>	
FACE_Profile.FACE_Extended_Stereotypes	FACE_UnitOfConformanceTypeEnum	Enumeration	Not from the Metamodel, created for System-of-Systems.<Derived from FACE Technical Standard>	
FACE_Profile.FACE_Extended_Stereotypes	FACE_UoCElement	Element	Not from the Metamodel, created for System-of-Systems.<Derived from FACE Technical Standard>	
FACE_Profile.FACE_Extended_Stereotypes	FACE_UoCModel	Package	Not from the Metamodel, created for System-of-Systems.<Derived from FACE Technical Standard>	
FACE_Profile.UAF_Extensions	FACE_Implements	Dependency	Not from the Metamodel, created for UAF Mapping.<Created for UAF Mapping>	