**Date:** November 2014

# UML Profile for the Global Justice Information Sharing Initiative Global Reference Architecture (GRA-UML)

*Version 1.0*

---

**OMG Document Number: gov/2014-11-01**

**Normative reference: http://www.omg.org/spec/GRA-UML/1.0/**

**Machine readable file(s):**

**Normative:**
http://www.omg.org/spec/GRA-UML/20141101/GRAProfile.xmi
http://www.omg.org/spec/GRA-UML/20141101/GRAAnnotations.xmi
http://www.omg.org/spec/GRA-UML/20141101/GRAAnnotationModel.xmi
http://www.omg.org/spec/GRA-UML/20141101/graAnnotationModelXMI.xsd
http://www.omg.org/spec/GRA-UML/20141101/graWsdlXMI.xsd
http://www.omg.org/spec/GRA-UML/20141101/GraSspModel2template.qvto
http://www.omg.org/spec/GRA-UML/20141101/GraCommon.qvto

**Non-normative:**
http://www.omg.org/spec/GRA-UML/20141101/Informative/GRATemplate.xmi
http://www.omg.org/spec/GRA-UML/20141101/Informative/StandardTemplate
http://www.omg.org/spec/GRA-UML/20141101/Informative/PetSimpleService/PetSimpleService.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/PetSimpleService/PetIEPD.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/SIRS/SIRS-Service.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/SIRS/SIRS-Information.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/Corrections/Corrections_SSP_v_1.0.0.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/Corrections/Corrections_IEPD_v_1.0.0.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/Corrections/MessageMetadata.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/Corrections/MessageMetadataReference.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/PetTest/PetTestService.mdzip

---

# OMG's Issue Reporting Procedure

# Table of Contents

# Figures and Tables

# Preface

## OMG

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

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

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

## OMG Specifications

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

*http://www.omg.org/spec*

Specifications are organized by the following categories:

### Business Modeling Specifications

### Middleware Specifications

- **CORBA/IIOP**
- **Data Distribution Services**
- **Specialized CORBA**

### IDL/Language Mapping Specifications

### Modeling and Metadata Specifications

- **UML, MOF, CWM, XMI**
- **UML Profile**

### Modernization Specifications

### Platform Independent Model (PIM), Platform Specific Model (PSM), Interface Specifications

- **CORBAServices**
- **CORBAFacilities**

**OMG Domain Specifications**

**CORBA Embedded Intelligence Specifications**

**CORBA Security Specifications**

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

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

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

## Typographical Conventions

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

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

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

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

Helvetica/Arial - 10 pt: Exceptions

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

## Issues

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

# 0 Submission-related material

## 0.1 Submission Introduction

The GRA-UML submission team is pleased to present a revised submission to the “UML Profile for GRA (GRA-UML)” Request for Proposal gov/13-09-20.

The IPR mode for this submission is Non-Assert.

Clause 0 of this document contains information specific to the OMG submission process and is not part of the proposed specification. The proposed specification starts with Clause 1. All clauses are normative unless otherwise specified.

## 0.2 Submission Team

### 0.2.1 Submitters

Model Driven Solutions, cory-c@modeldriven.com
IJIS Institute, ashwini.jarral@ijis.org

### 0.2.2 Contributors

- Hidden Symmetry Ltd.
- Open Networks Inc.

## 0.3 Resolution of Requirements

### 0.3.1 Mandatory requirements

| | |
|---|---|
| 6.5.1 Submissions shall specify a GRA-UML Logical Profile. The GRA-UML Logical Profile shall be a set of UML stereotypes and properties which support the modeling of SSPs in UML in a technology-independent way. This profile shall support modeling of any content and structure allowed by GRA SSPs, while constraining the modeling of any content and structure disallowed by GRA SSPs, as specified in 6.5.4. The use of the GRA-UML Logical Profile shall result in UML models that are free from dependency on any physical representation (such as XML Schema). In MDA terms, the GRA Logical Profile is a specification of the platform independent model (PIM). | The Logical Services Profile is explained in clause 7 and specified in clause 8. |
| 6.5.2 Submissions shall specify a GRA-UML Profile for SSPs. The GRA-UML Profile for SSPs shall be a set of UML stereotypes and properties which specify the details and metadata of a SSP specification to be produced based on logical GRA model(s). Transformations (discussed in 6.5.3) shall be designed to utilize the Profile for SSPs to parameterize the transformation from the GRA-UML Logical Profile (the PIM) to GRA-conformant SSPs (the PSM), where GRA conformance is as defined in 6.5.4. The SSP profile shall govern the inclusion in a UML model of any information necessary to properly generate GRA-conformant SSP artifacts beyond the information found in the GRA-UML Logical | Instead of a set of stereotypes, the details and metadata for SSP generation are specified by instantiating a model library. This library is explained in clause 7 and specified in clause 9. |

| | |
|---|---|
| profile. | |
| 6.5.3 Submissions shall specify a transformation from UML models using the GRA-UML profiles specified in 6.5.1 and 6.5.2 to the set of artifacts required in a conformant SSP, as defined by 6.5.4. Submissions shall utilize the GRA-UML Profiles to model at least one existing GRA SSP and demonstrate the resulting transformation to an SSP. The SSP produced must be GRA conformant, as defined in section 6.5.4, and the XML Schema set contained within the SSP must validate the same set of exchange documents as the existing SSP IEPD. It is not required that the generated SSP be structurally identical to the existing SSP. | The transformation is done in two phases, as explained in clause 7.2. |
| 6.5.4 The SSPs generated based on models conforming to the GRA-UML profile shall conform to normative GRA specifications referenced in section 6.4.1. These specifications are:<br>• GRA Service Specification Package, v1.0.0 (http://it.ojp.gov/docdownloader.aspx?ddid=1217)<br>• GRA Service Specification Guideline v1.0.0 (http://it.ojp.gov/docdownloader.aspx?ddid=1215)<br>• GRA Web-Services Service Interaction Profile v1.3 (http://it.ojp.gov/docdownloader.aspx?ddid=1173)<br>• GRA ebXML Messaging Service Interaction Profile v1.1 (http://it.ojp.gov/docdownloader.aspx?ddid=1168)<br>• GRA Reliable Secure Web Services, Service Interaction Profile v1.2 (http://it.ojp.gov/docdownloader.aspx?ddid=1134) | The generated SSPs are GRA conformant. |
| 6.5.5 Specifications shall reuse constructs and/or representations from SoaML to express the SSP logical model and will extend SoaML to meet specific requirements of the GRA where required. | SoaML is not used or extended, although some of its general principles are adopted such that SoaML models may be used with GRA-UML. Instead, normal UML models are used, with just two stereotypes to indicate the providers and consumers of services. |
| 6.5.6 Specifications shall reuse elements and/or representations from UML Profile for BPMN 2 Processes to express the SSP logical model and more specifically the SSP interaction model. | BPMN2 is not used. A process model is not required in order to generate GRA-compliant schemas. |
| 6.5.7 Specifications shall define mappings between GRA, and these industry-standard specifications - OMG SoaML, OASIS SOA-Reference Model, Open Group SOA Ontology. | Annex B provides these mappings. |
| 6.5.8 Specifications shall use OMG-QVT to specify the transformation of the UML logical model to all or part of a SSP. | The Phase-1 transformation is normative and uses QVT. Phase-2 is non-normative, uses XSLT and ANT and is customizable. |

### 0.3.2 Non-mandatory features

| | |
|---|---|
| 6.6.1 Submissions may specify a "reverse engineering" transformation from a GRA conformant SSP to UML models that conform to the GRA-UML profiles. The reverse engineering shall produce both a "PIM" (A model conforming to the GRA-UML Logical Profile) and a corresponding "PSM" (A model conforming to the GRA-UML Profile for SSPs) such that applying the forward engineering transformations to these models produces a GRA-conformant SSP semantically equivalent to the input. | No such transformation is provided. |
| 6.6.2 The submitters may propose a revision of SoaML which would benefit GRA needs and the community at large. | No such revision is proposed. The GRA logical services profile is similar to a subset of SoaML but simpler and smaller. |

## 0.4 Resolution of Discussion Issues

6.7.1 Submissions shall discuss the relationship of GRA-UML with other ongoing and related GRA standards, the existing GRA specifications and the GRA process.

GRA-UML is supported by the Global Standards Council (GSC), the organization responsible for the GRA. The GSC has been involved with and influenced the evolution of GRA-UML and its fit with GRA. One factor in this influence has been that the definition of process is allowed for but not required or emphasized in a GRA-UML specification as the GSC is in the process of moving process definition out of the GRA.

6.7.2 Submissions shall discuss their relationship with other relevant standards including but not limited to the Unified Profile for DoDAF/MODAF (UPDM)

Alignment with DoDAF/MODAF (UPDM) has not been evaluated.

6.7.3 If a GRA SIP for the Representational State Transfer (REST) protocol becomes available during the submission process, submissions may discuss its impact on the submission.

No such SIP exists at present. The two-phase generation approach specified by GRA-UML should be able to incorporate such technology developments.

6.7.4 Submissions shall discuss their relationship with other relevant standards including but not limited to the IEPPV.

There is no substantive relationship with IEPPV. The policy approach specified by GRA-UML is aligned with GRA.

# 1 Scope

## 1.1 GRA-UML Background

The Global Reference Architecture (GRA) is an information exchange solution designed to reduce implementation time and costs for state and local justice agencies to share information through reuse of established practices in IT architecture and design. It is developed and maintained under the governance of the Global Standards Council (GSC), part of the Global Justice Information Sharing Initiative (Global) which serves as a Federal Advisory Committee (FAC) and advises the U.S. Attorney General on justice information sharing and integration initiatives.

GRA solutions to information exchange are made up of a combination of the connection method (often Web Services), the exchange language (use of NIEM is encouraged), and the security specifications (encryption at the transport layer, data layer, etc.). These specifications are packaged into a GRA solution that can be customized to meet a community's need for interoperability and information exchange.  These packages are called Service Specification Packages (SSPs). A Service Specification Package Template and a number of reference Service Specification Packages are available as part of the GRA and can be found on the GRA website at http://www.it.ojp.gov/gra. The GRA SSPs are used by a number of justice and public safety agencies to define and implement interoperable information sharing services.

Prior to GRA-UML there were no standard based tools which could be used to model and generate GRA SSPs. As a result creating, reusing and implementing GRA SSPs was a very manual process, and requires comprehensive understanding of the GRA.

In GRA-UML, SSPs are generated from models, thus simplifying and improving the consistency of their implementation. The models are created from two viewpoints:

- The Logical Service Model is a platform-independent description of the services automated by the SSP.
- The Annotation Model is a description of the additional metadata needed to interpret the Logical Service Model and generate a complete and conformant SSP.

Each of these models is created using a subset of UML. GRA-UML defines these subsets together with the transformations and workflows used to generate a complete and conformant SSP from the models.

## 1.2 Intended Users of GRA-UML

GRA-UML enables the construction of an SSP to be distributed amongst architects and developers with different skillsets. The Logical Service Model specifies business requirements for the services, including the various participating individuals, organizations and software systems, the capabilities offered by these participants, and the interactions between them.  These are modeled in a technology-agnostic way using a subset of UML; the modelers need only understand this subset of UML and the business requirements to be modeled. The Annotation Model adds the detail required to generate the complete SSP. This includes metadata, technology and policy choices for the SSP in cases where these choices cannot be inferred from the Logical Service Model by applying defaults.  The annotation modeler needs a good understanding of the structure of SSPs and the various policy and implementation choices available.  The final step in the production of an SSP involves the application of an ANT script and XSLT transformations to generate the actual SSP artifacts; this ANT script and XSLT transformations may be customized by a template developer, expert in XSLT and in the target technology, in cases where the detailed implementation choices provided by the pre-packaged templates require changing.

## 1.3 GRA-UML Profile, Library and Transformations

A GRA-UML Logical Service Model is created using the Logical Services UML Profile described in Clause 8.  This profile defines a subset of UML that includes SOA constructs such as Actors, Use Cases, Components, Ports, Interfaces etc., and two UML stereotypes, together with modeling conventions about how to use the profile to describe service participants and their interactions.

A GRA-UML Annotation Model is created by using the GRA Annotation Library described in Clause 9. Metadata describing the SSP requirements is created by instantiating types from the Annotation Library as UML InstanceSpecifications, linked together into a UML instance model that portrays the overall structure of the metadata for the SSP.

Given a Logical Services Model and Annotation Model, a *two-phase provisioning* process is used. First, QVT transformations are used to convert the models into a standardized intermediate format that provides all of the metadata needed to construct a complete valid SSP. Secondly, XSLT transformations are applied to create the final SSP artifacts based on the ANT script.

Clause 7 provides a detailed rationale and description for each of these models, conventions, workflows and transformations.

# 2 Conformance

All clauses of this specification that are not marked as "informative" are normative, and a conforming implementation shall satisfy all of them.

# 3 Normative References

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

| | |
|---|---|
| [MOF] | OMG Meta Object Facility (MOF) Core Specification v2.4.2, formal/2014-04-03, http://www.omg.org/spec/MOF/2.4.2/PDF/ |
| [UML] | OMG Unified Modeling Language (UML) Superstructure v2.4.1, formal/2011-08-06, http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/ |
| [OCL] | OMG Object Constraint Language (OCL) v2.4, formal/2014-02-03, http://www.omg.org/spec/OCL/2.4/PDF/ |
| [QVT] | OMG Meta Object Facility (MOF) Query/View/Transformation v1.2, ptc/2014-03-38, http://www.omg.org/spec/QVT/1.2/Beta/PDF |
| [GRA] | Global Reference Architecture, https://it.ojp.gov/gra |
| [NIEM] | National Information Exchange Model, www.niem.gov |
| [NIEM-UML] | UML Profile for NIEM, http://www.omg.org/spec/NIEM-UML/ |
| [SoaML] | Service Oriented Architecture Modeling Language, http://www.omg.org/spec/SoaML/1.0.1/ |

# 4 Terms and Definitions

For the purposes of this specification, the following terms and definitions apply. For terms labelled (GRA), full definitions are found at https://it.ojp.gov/gist/43/The-Global-Reference-Architecture--GRA--Service-Specification-Guideline-V-1-0-0.

**IEPD (NIEM)**

Information Exchange Package Documentation.

**LSM**

Logical Service Model.

**SOA**

Service Oriented Architecture.

**SDD (GRA)**

Service Description Document.

**SIDD (GRA)**

Service Interface Description Document.

**SSP (GRA)**

Service Specification Package.

**SIP (GRA)**

Service Interaction Profile.

**WSDL**

Web Services Description Language, a W3C standard XML-based interface definition language used for describing the functionality offered by a web service.

**XHTML**

Extensible HyperText Markup Language.

# 5 Symbols

There are no symbols defined in this specification.

# 6 Additional Information

## 6.1 Acknowledgements

The following entities have played a significant role in driving the development of this specification:

- Submitters
    - Model Driven Solutions
    - The Integrated Justice Information Systems Institute (IJIS)
- Government Stakeholders
    - U.S. Department of Justice
    - U.S. Information Sharing Environment
- Contributors
    - Hidden Symmetry Ltd.
    - Open Networks Inc.

# 7 GRA-UML Modeling Guide

## 7.1 GRA SSP Overview

The purpose of a GRA-UML tool is to enable GRA architects and developers to model the business and technological requirements for a GRA SSP using UML, and to provision the complete valid GRA SSP as an output. This section briefly describes the contents and structure of a GRA SSP: it is necessary to understand these in order to understand the process used to create them. Full details of the GRA SSP can be found at https://it.ojp.gov/gist/43/The-Global-Reference-Architecture--GRA--Service-Specification-Guideline-V-1-0-0.

The files that constitute a valid GRA SSP must be arranged in the predefined folder structure shown in Figure 1.

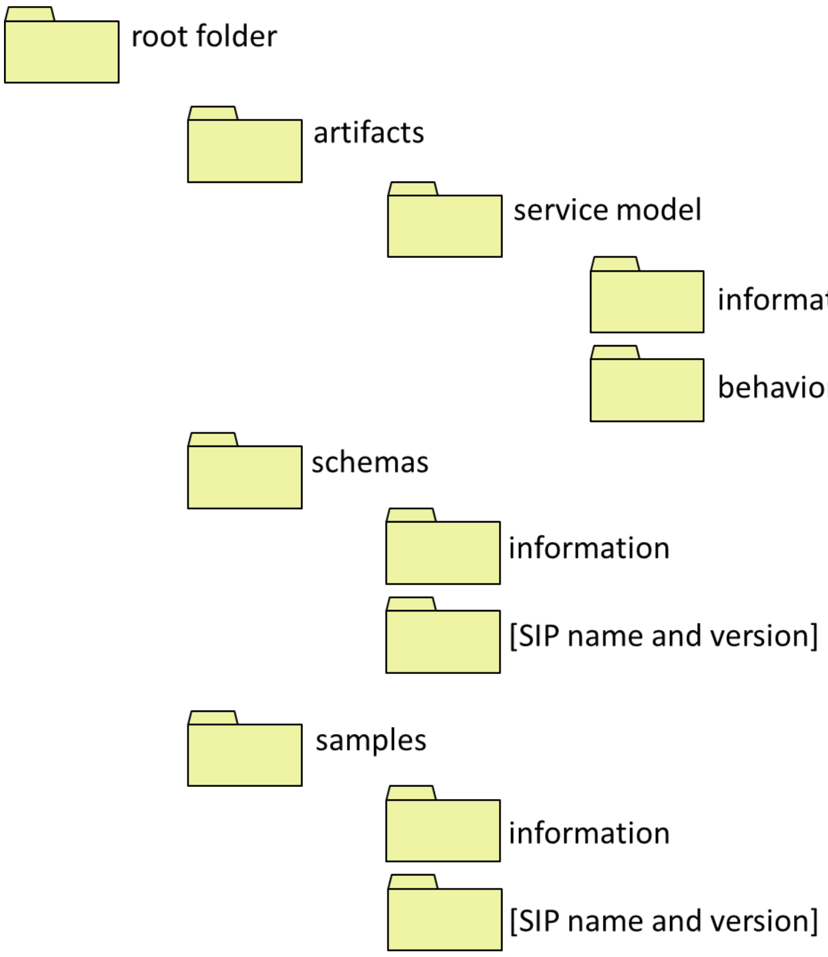


**Figure 1 Mandatory SSP folder structure**

A complete GRA SSP (Service Specification Package) comprises at least the following files:

- A Service Description, manifested in a Service Description Document (SDD). This is a human-readable file contained in the artifacts folder. For GRA-UML, XHTML format is generated.
- One or more Service Interface Descriptions, manifested in Service Interface Description Documents (SIDDs). Each Service Interface Description must conform to one or more Service Interaction Profiles (SIPs). SIDDs are human-readable files contained in the artifacts folder. For GRA-UML, XHTML format is generated.

- A Service Metadata file. This is called metadata.xml and contained in the root folder, which also contains a copy of the GRA standard file metadata.xsd.
- A Service Catalog. This consists of two files, a machine-readable catalog.xml and a human-readable catalog.html, both contained in the root folder.
- A Service Specification Information Model, which must be a NIEM-conformant IEPD. This is packaged in a zip file or folder and contained in the artifacts/service model/information model folder. In the case of GRA-UML it is a folder whose contents conform to transformations specified in the [NIEM-UML] profile.
- A Service Specification Behavior Model. In the case of GRA-UML, this is the Logical Service Model (see clause 7.3). It is contained in the artifacts/service model folder.
- Schemas. Web Service descriptions in WSDL format that are generated from the annotated GRA-UML model are contained in the schemas/[SIP name and version] folder, where the name of the folder corresponds to the SIP used to generate the WSDL.

Other optional files may be included and the folder structure may be extended according to rules specified by GRA. Such optional files and folders are not consumed or provisioned by GRA-UML and their description is omitted from this specification.

## 7.2 Two-Phase SSP Provisioning

GRA-UML has two potentially conflicting goals: to generate a fully GRA-conformant SSP, while providing flexibility in how SSPs are produced, what technologies are used and how these technologies are used. The reason for this flexibility as that GRA is essentially open ended with respect to the service interaction profiles (SIPs) used as well as what specific technologies are used within a SIP. In addition, SSP architects frequently have additional requirements or styles with respect to the generated documentation.

To resolve this dilemma the generation and provisioning of the final SSP is divided into two steps, called *Phase-1* and *Phase-2* as shown in Figure 2 below. Phase-1 uses a standardized algorithm and is implemented in QVT; Phase-2 is customizable and the default version is implemented using XSLT and ANT.

The GRA-UML services architect creates a model of the SSP containing two main aspects that separate concerns between the logical services and the GRA-specific requirements and metadata:

- The Logical Service Model - a platform-independent specification of the services to be automated by an implementation of the SSP, in terms of SOA constructs such as Actors, Use Cases, Components, Ports, Interfaces etc. It contains normal UML structures that are used to define services by applying simple conventions and a couple of GRA-specific stereotypes.
    - o *Note: this same UML model could be used and extended for purposes other than producing GRA specifications, such as producing implementations. Use of SoaML to produce this model is encouraged but not required.*
- The Annotation Model – a modeled specification of GRA-specific metadata, including technology and policy choices for the SSP, in cases where these choices cannot be inferred from the Logical Service Model by applying defaults.

Phase-1 uses a standardized algorithm to generate an intermediate format representation of the SSP. It uses the OMG’s QVT (Query/View/Transformation) to transform the GRA-UML model into a standardized XML-based format that contains all of the data required to generate the complete SSP. This intermediate format uses XMI (XML Metadata Interchange) corresponding to the OMG’s EMOF (Essential MOF) specification. XMI is an XML format with a schema that provides a convenient and standardized intermediate format for consumption by Phase-2.

Phase-2 uses a set of transformations to generate the complete final SSP. The baseline template provided by GRA-UML consists of a folder structure whose top-level folder is called StandardTemplate. Within this folder structure are located the baseline ANT script and XSLT transformations, as well as various other artifacts needed to execute Phase-2. The details of the processing done by this baseline template are contained in clause 9.5. Some or all of this baseline template may be replaced or extended to customize the resulting SSP to handle different profiles, technologies, requirements or document styles.

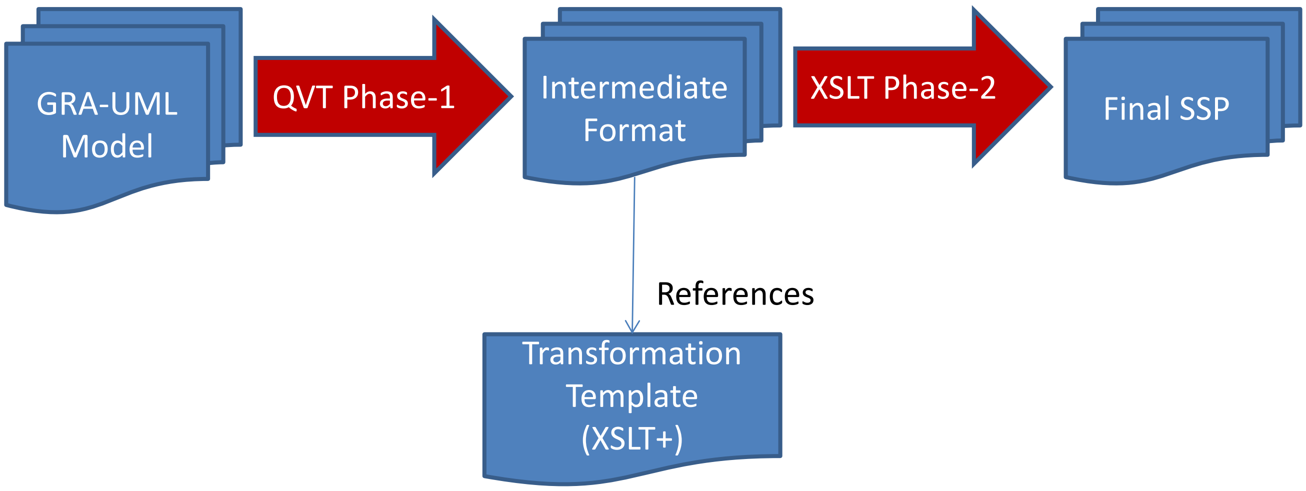


**Figure 2 Two-phase provisioning**

The envisaged workflow for a user of GRA-UML is as follows. A much more detailed description appears in clause 10 of this document.

- User loads GRA-UML, which consists of the Logical Services Profile and the GRA Annotation Library, into a UML tool.
- User loads a starter model which contains a starter Logical Service Model and a set of starter annotations, which are InstanceSpecifications classified by classes in the GRA Annotation Library. A starter model is provided with the specification.
- User uses the Logical Services Profile and extends the starter model to create the Logical Services platform-independent model – the PIM.
- User edits and adds annotation elements and properties, and connects annotation elements to elements in the PIM using Realization and Usage dependencies.
- User invokes an action which causes the tool to, for each ServiceDescription:
  - Execute Phase-1 to generate the intermediate format annotation instance file (annotations.xmi) and the other artifacts produced by Phase-1.
  - Invoke Phase-2 to produce the final SSP artifacts and to place all artifacts correctly in the folder organization.

## 7.3 The Logical Service Profile

A UML Profile is a means to customize the general-purpose modeling language UML so that it can be used to create models that relate to a particular domain. A UML Profile has the following characteristics:

- It may define a subset of UML which is applicable to the intended domain.
- It may define a set of Stereotypes. These are labeled bundles of information which may be applied to particular kinds of UML element, carrying additional metadata relevant to the intended domain.
- It may define a library of domain-specific types for use within the model.

GRA-UML does all three of these.  It defines:

- A subset of UML, used for modeling services at a logical, platform-independent level of abstraction.
- Two stereotypes, needed for distinguishing between service consumers and providers.
- The GRA Annotation Library, used to create elements that represent the detailed metadata required to provision SSPs. This library is described in clause 7.4 and specified in detail in clause 9.

This sub-clause describes the use of the first two of these constituents, which are together known as the Logical Services Profile.

Logical Service Models (LSMs) may contain any kind of UML element, but only those elements that are consumed by the transformations defined in this specification will influence the machine-generated outputs provisioned to the SSP. Logical Service Models use existing UML concepts in familiar ways and should be easily understood by typical UML users.

The Logical Service Profile is similar to, but considerably simpler than, OMG's existing SoaML specification. It contains just two stereotypes called «Provider» and «Consumer» which are intended to be applied to UML Associations in use cases as shown in 7.3.2.  No additional metadata is required to model the Logical Services Model using normal UML elements as shown in the following sections.

### 7.3.1 Actors

Actors are the business participants in the SSP's subject community. They are typically people, organizations or roles of people or organizations. Figure 3 shows Actors, some of which are organized in an inheritance hierarchy.

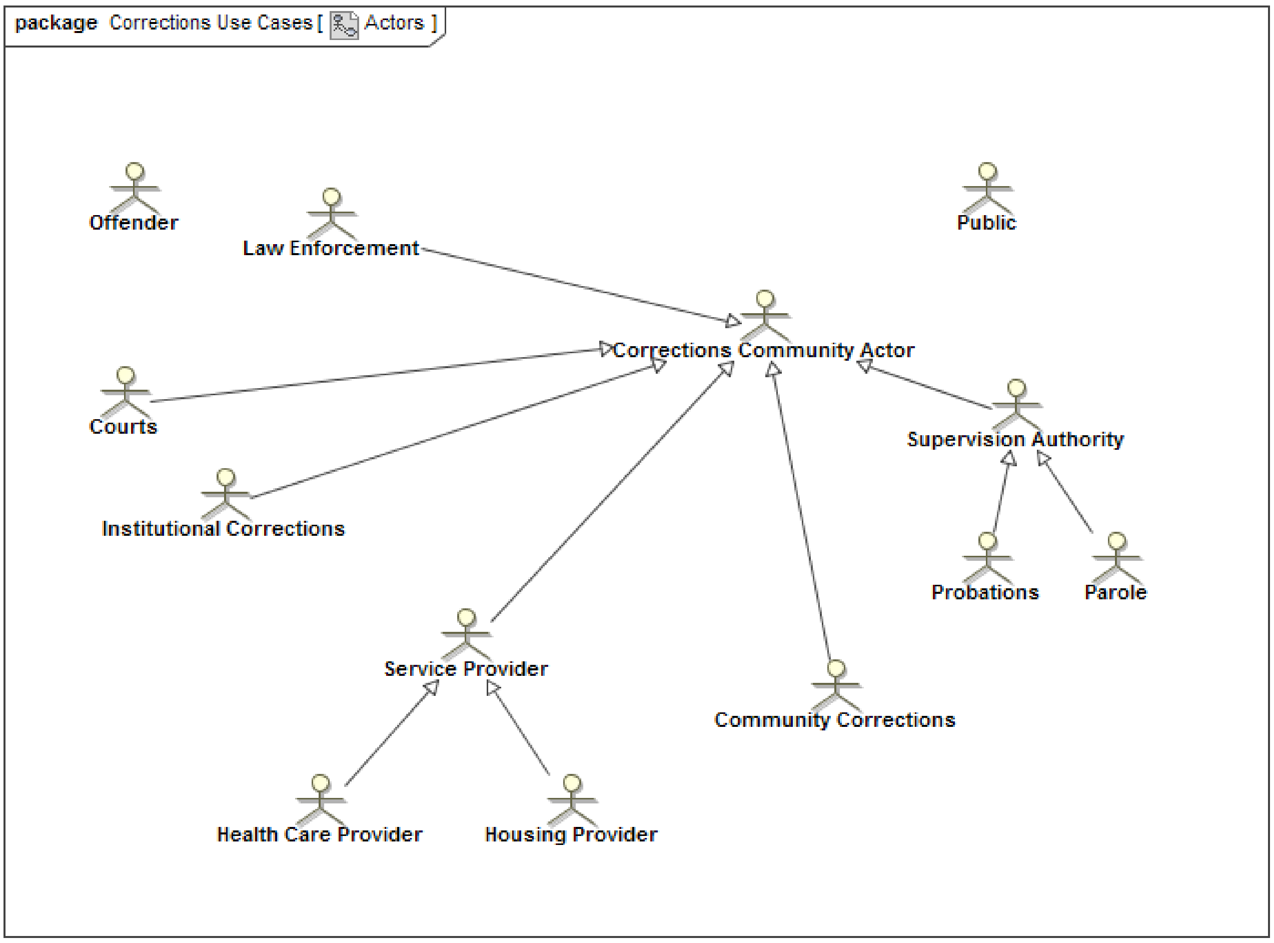


**Figure 3 Actors**

### 7.3.2 Use cases

The *real-world effects* of services are modeled as UseCases involving Actors. Each UseCase (oval) is a real-world effect and a business interaction. Each association between a UseCase and an Actor may be marked with a stereotype to indicate whether the Actor is a «Provider» or «Consumer» of the service. Figure 4 shows an example that involves some of the Actors from Figure 3.

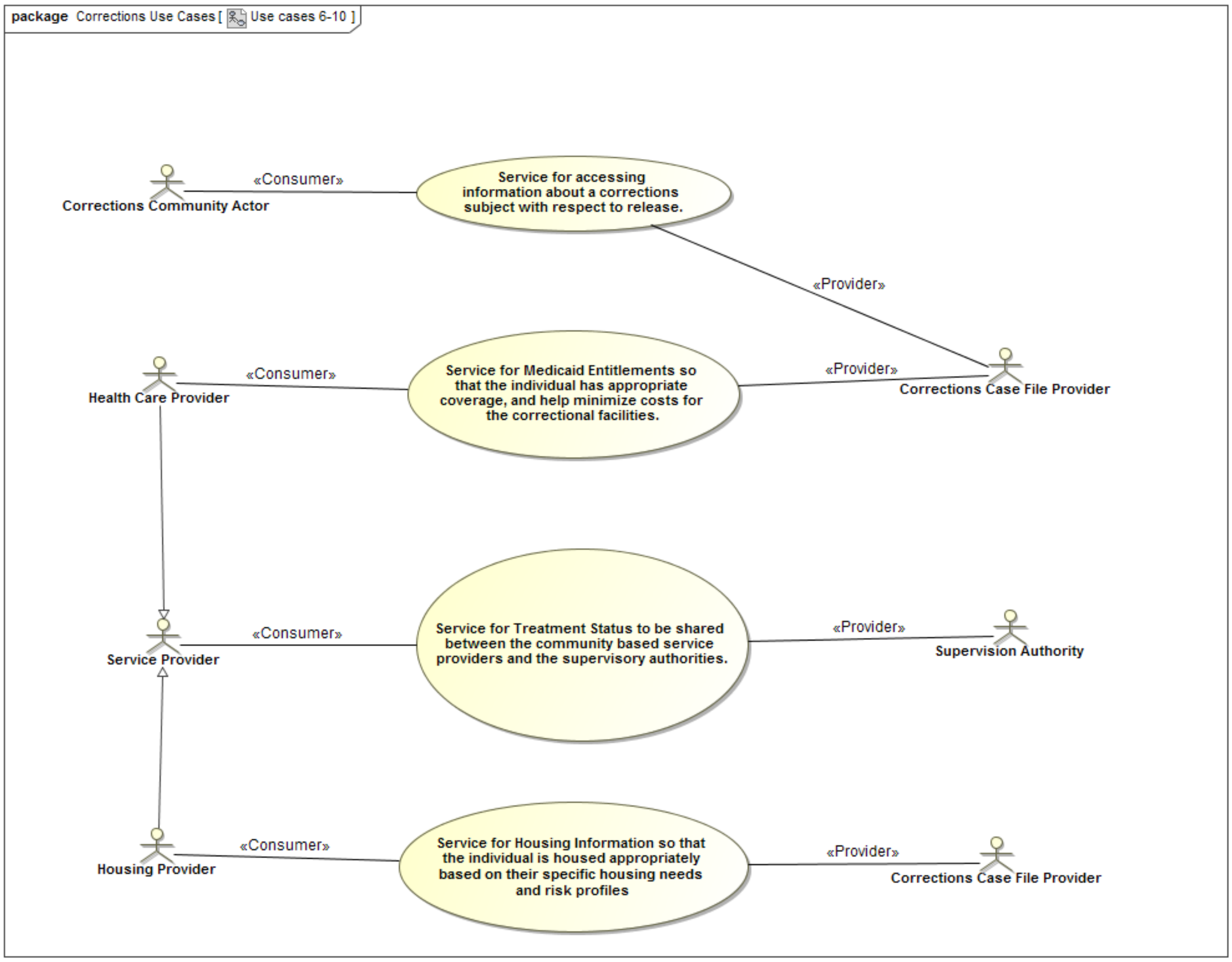

package Corrections Use Cases [ Use cases 6-10 ]
Corrections Community Actor
«Consumer»
Service for accessing information about a corrections subject with respect to release.
«Provider»
Health Care Provider
«Consumer»
Service for Medicaid Entitlements so that the individual has appropriate coverage, and help minimize costs for the correctional facilities.
«Provider»
Corrections Case File Provider
Service Provider
«Consumer»
Service for Treatment Status to be shared between the community based service providers and the supervisory authorities.
«Provider»
Supervision Authority
Housing Provider
«Consumer»
Service for Housing Information so that the individual is housed appropriately based on their specific housing needs and risk profiles
«Provider»
Corrections Case File Provider

**Figure 4 Use Cases**

### 7.3.3 Interfaces

UML Interfaces define the operations and signal receptions that service providers implement to expose a service. The arguments of operations and receptions directly reference [NIEM-UML] message types as their content. Figure 5 shows an example with two interfaces, one containing an operation and the other a reception. The figure also shows the types used by the two interfaces.

### 7.3.4 Components and ports

UML Components represent the service provided by an Actor. Each Component has a set of Ports that provide or use UML Interfaces. A Component may realize an Actor, to signify in the model that it represents a service provided by that Actor. Figure 6 illustrates two Components each with Ports and Interfaces, and realizations to the Actors that provide the service. Because both of these Components offer Receptions in their provided interfaces, they are marked as active.

### 7.3.5 Collaborations & Interactions

Collaborations are used to define the *community* for which the SSP is intended, providing a property for each Actor and service Component playing a role in that community. Figure 7 shows such a Collaboration with properties (in UML terminology *parts*) that correspond to the Actors and Components that appear in Figure 6. In this Collaboration, the parts typed by the Actors are un-named. The parts typed by the Components have the same names as the realized Actors: this is purely an informal convention to facilitate readability of this example.

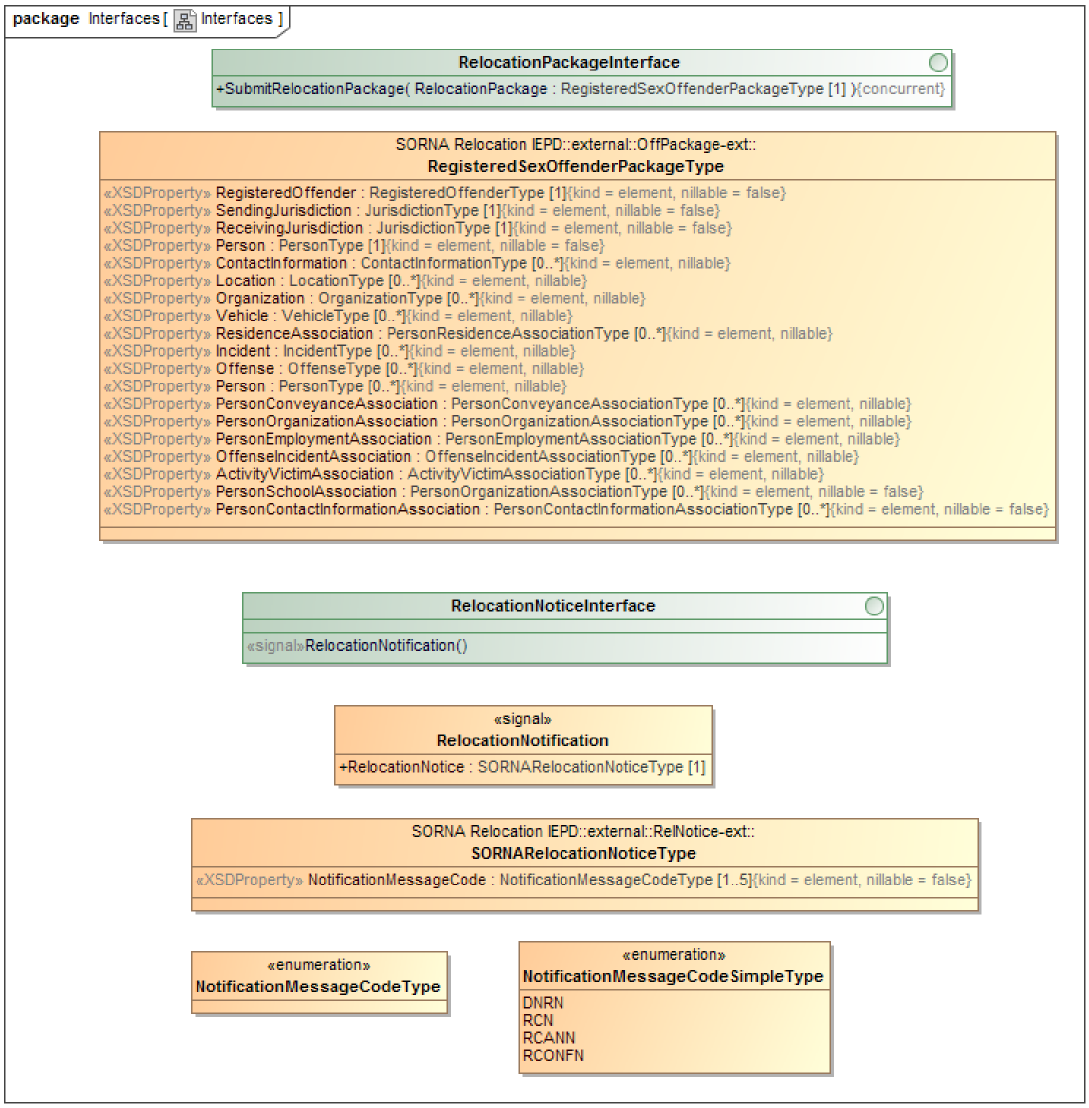

package Interfaces [ Interfaces ]
RelocationPackageInterface
+SubmitRelocationPackage( RelocationPackage : RegisteredSexOffenderPackageType [1] ){concurrent}
SORNA Relocation IEPD::external::OffPackage-ext::
RegisteredSexOffenderPackageType
«XSDProperty» RegisteredOffender : RegisteredOffenderType [1]{kind = element, nillable = false}
«XSDProperty» SendingJurisdiction : JurisdictionType [1]{kind = element, nillable = false}
«XSDProperty» ReceivingJurisdiction : JurisdictionType [1]{kind = element, nillable = false}
«XSDProperty» Person : PersonType [1]{kind = element, nillable = false}
«XSDProperty» ContactInformation : ContactInformationType [0..*]{kind = element, nillable}
«XSDProperty» Location : LocationType [0..*]{kind = element, nillable}
«XSDProperty» Organization : OrganizationType [0..*]{kind = element, nillable}
«XSDProperty» Vehicle : VehicleType [0..*]{kind = element, nillable}
«XSDProperty» ResidenceAssociation : PersonResidenceAssociationType [0..*]{kind = element, nillable}
«XSDProperty» Incident : IncidentType [0..*]{kind = element, nillable}
«XSDProperty» Offense : OffenseType [0..*]{kind = element, nillable}
«XSDProperty» Person : PersonType [0..*]{kind = element, nillable}
«XSDProperty» PersonConveyanceAssociation : PersonConveyanceAssociationType [0..*]{kind = element, nillable}
«XSDProperty» PersonOrganizationAssociation : PersonOrganizationAssociationType [0..*]{kind = element, nillable}
«XSDProperty» PersonEmploymentAssociation : PersonEmploymentAssociationType [0..*]{kind = element, nillable}
«XSDProperty» OffenseIncidentAssociation : OffenseIncidentAssociationType [0..*]{kind = element, nillable}
«XSDProperty» ActivityVictimAssociation : ActivityVictimAssociationType [0..*]{kind = element, nillable}
«XSDProperty» PersonSchoolAssociation : PersonOrganizationAssociationType [0..*]{kind = element, nillable = false}
«XSDProperty» PersonContactInformationAssociation : PersonContactInformationAssociationType [0..*]{kind = element, nillable = false}
RelocationNoticeInterface
«signal»RelocationNotification()
«signal»
RelocationNotification
+RelocationNotice : SORNARelocationNoticeType [1]
SORNA Relocation IEPD::external::RelNotice-ext::
SORNARelocationNoticeType
«XSDProperty» NotificationMessageCode : NotificationMessageCodeType [1..5]{kind = element, nillable = false}
«enumeration»
NotificationMessageCodeType
«enumeration»
NotificationMessageCodeSimpleType
DNRN
RCN
RCANN
RCONFN

**Figure 5 Interfaces**

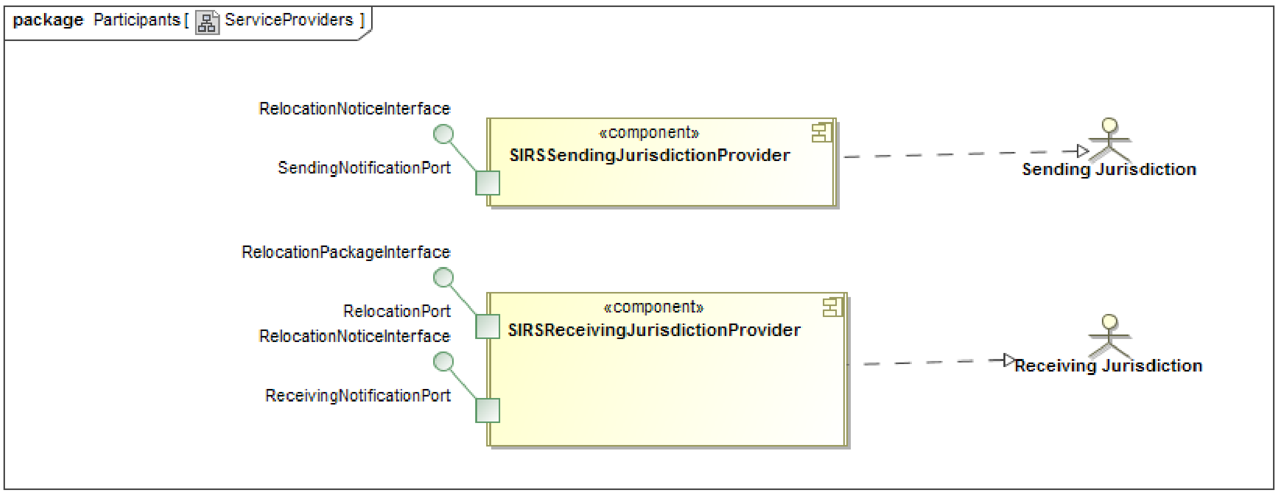

package Participants [ ServiceProviders ]
RelocationNoticeInterface
SendingNotificationPort
«component»
SIRSSendingJurisdictionProvider
Sending Jurisdiction
RelocationPackageInterface
RelocationPort
RelocationNoticeInterface
ReceivingNotificationPort
«component»
SIRSReceivingJurisdictionProvider
Receiving Jurisdiction

**Figure 6 Components, Ports and Actors**

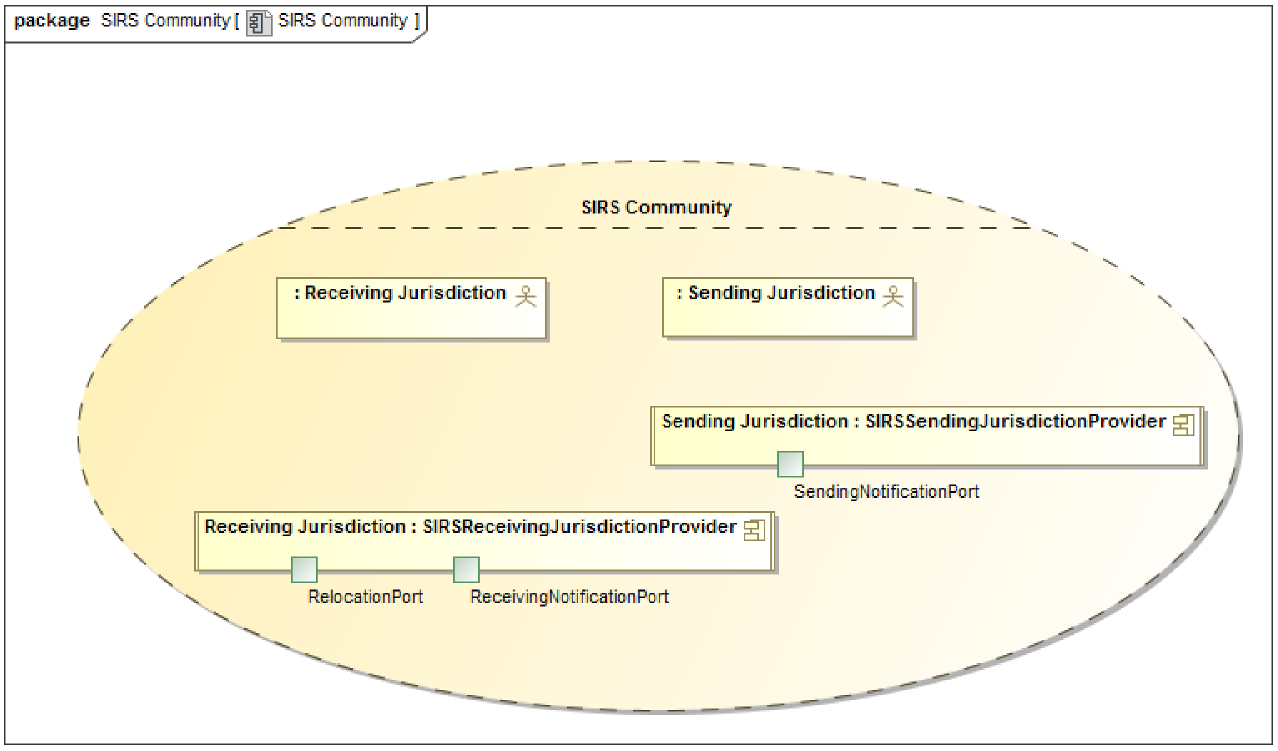

package SIRS Community [ SIRS Community ]
SIRS Community
: Receiving Jurisdiction
: Sending Jurisdiction
Sending Jurisdiction : SIRSSendingJurisdictionProvider
SendingNotificationPort
Receiving Jurisdiction : SIRSReceivingJurisdictionProvider
RelocationPort
ReceivingNotificationPort

**Figure 7 Collaboration**

The Collaboration that represents the community owns a set of Interactions that define the choreography of exchanges to implement the UseCases. Figure 8 shows an example that corresponds to Figure 7, where the lifelines represent the parts typed by Components.

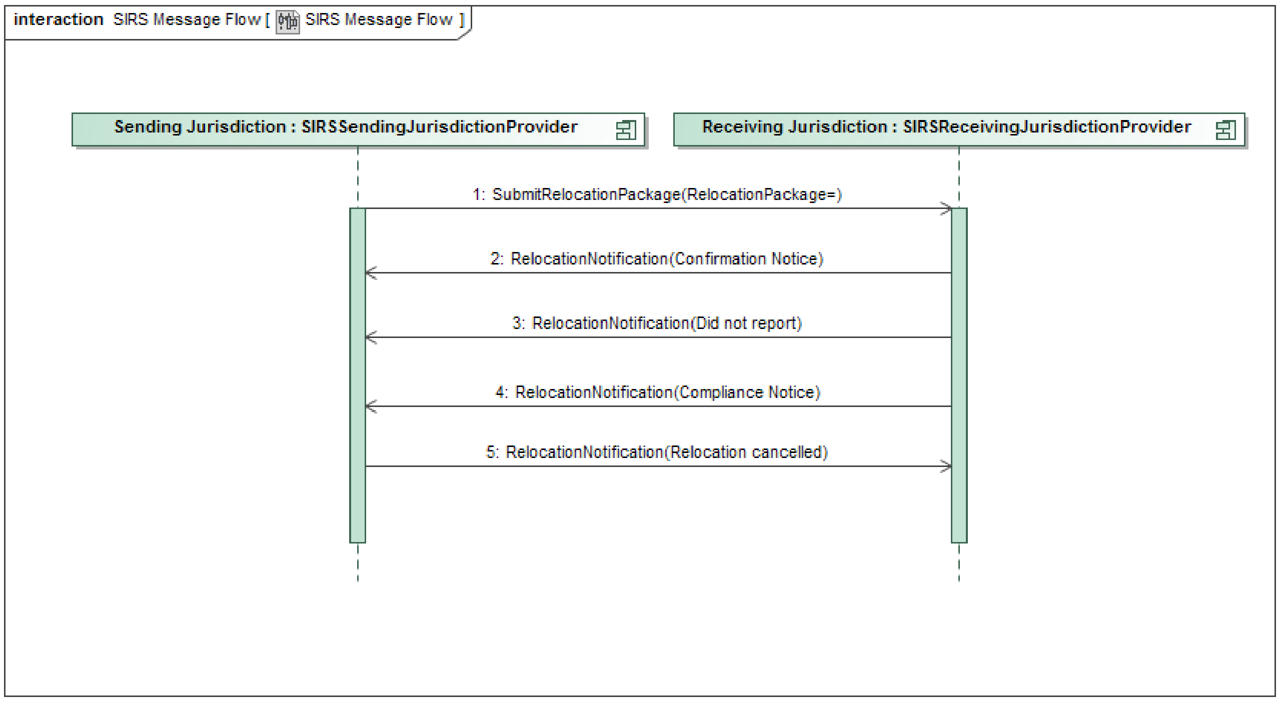

interaction SIRS Message Flow [ SIRS Message Flow ]
Sending Jurisdiction : SIRSSendingJurisdictionProvider
Receiving Jurisdiction : SIRSReceivingJurisdictionProvider
1: SubmitRelocationPackage(RelocationPackage=)
2: RelocationNotification(Confirmation Notice)
3: RelocationNotification(Did not report)
4: RelocationNotification(Compliance Notice)
5: RelocationNotification(Relocation cancelled)

**Figure 8 Interactions**

## 7.4 The GRA Annotation Library

### 7.4.1 Overview

The GRA Annotation Library performs two roles in the overall workflow, and hence exists in two similar forms that differ in their details. The two forms are called the Annotations and the Intermediate Model. Both libraries contain the same UML classes and enumerations, with some additional classes in the Intermediate Model and additional associations in the Annotations. The Annotations are designed to simplify the modeling task by avoiding duplication of information and assuming that defaults are applied wherever possible. The Intermediate Model, which is produced from the Annotations by the execution of Phase-1, is essentially the abstract syntax of a GRA SSP and is designed to provide a convenient representation for interpretation by Phase-2, which normally uses ANT and XSLT. So, for example, instances of the Intermediate Model are nested and avoid cross-references, whereas no such restriction applies to the Annotations.

The purpose of the Annotations is to provide the basis for the end user to model the GRA-specific metadata needed for generating the SSP. The end-user accomplishes this by creating UML InstanceSpecification elements that instantiate classes in the Annotations model.

*Note: For readability in describing the use of the annotation library, the word “instance” will be used to mean UML InstanceSpecification, unless this interpretation would cause confusion. See Figure 9 below for examples.*

Having created instances, the user gives values to properties in those instances, links them together, and provides Realization dependencies between specific instances and elements in the Logical Service model, and Usage dependencies to the Information Model.

The purpose of the Intermediate Model is to act as a metamodel for the intermediate annotations.xmi format. The Phase-1 transformation consumes the end-user’s model created using the logical model and Annotations, and creates (in memory) additional InstanceSpecifications, links and property values that represent all of the annotations needed to generate the SSP. The resulting model is then serialized as XMI in OMG EMOF format, where each UML InstanceSpecification is serialized as an EMOF instance, and UML links and slots serialized as EMOF properties.

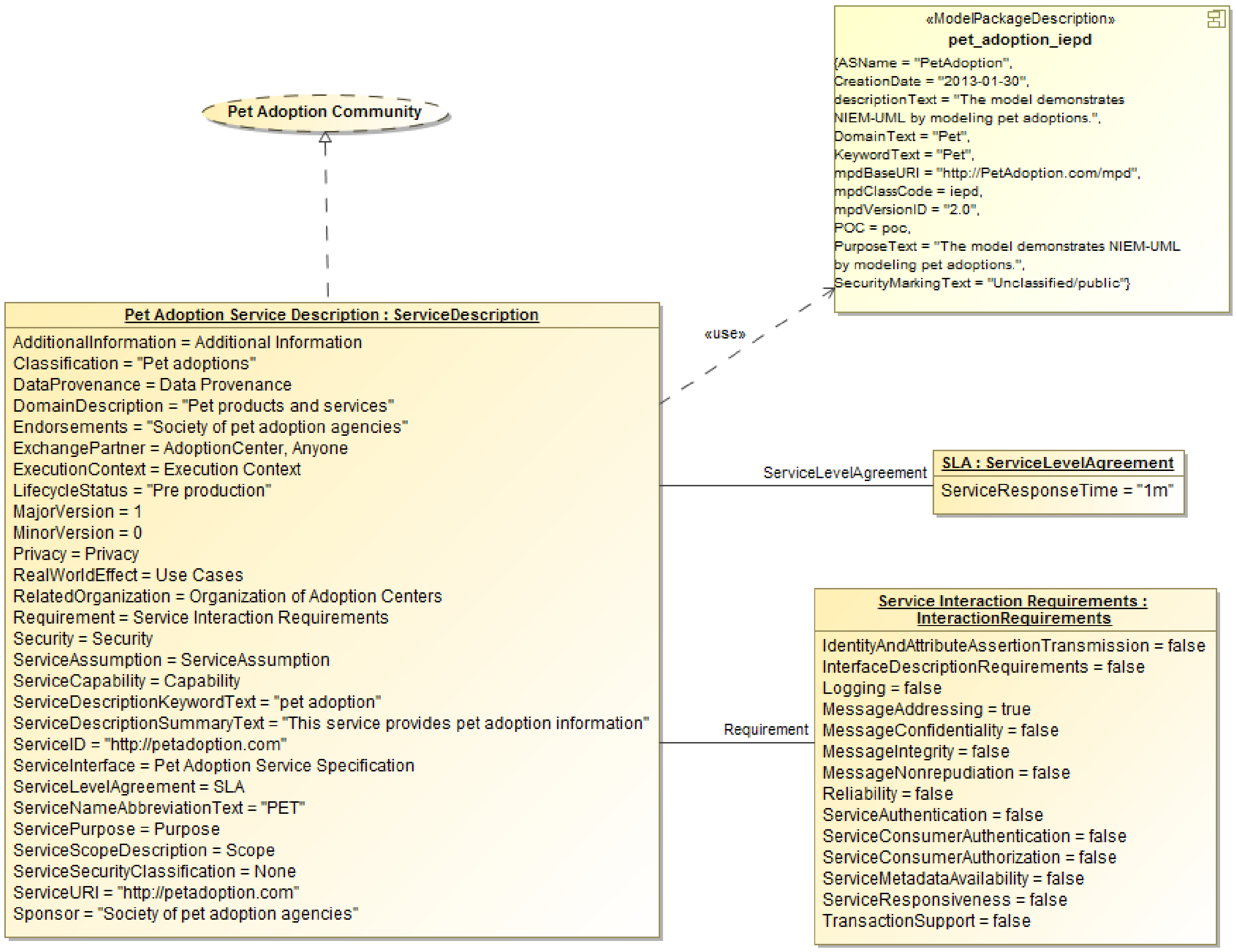
Although it would simplify this specification if these two libraries were identical, this would place an additional burden on the GRA-UML modeler to know about and ignore those elements that are only used in the Intermediate Model. The authors of GRA-UML consider this burden to be excessive, and hence have decided to keep the libraries separate, although for convenience they are documented together in Clause 9 of this specification.

The Annotation Library, in both of its forms, is specified in two packages: GRAAnnotationModel, containing platform-independent annotations, and GRAWsdl, containing WSDL-specific annotations.

The pivotal class in the Annotations is ServiceDescription. The user creates an instance of this class to represent the overall service description which ultimately leads to the creation of an SSP. This ServiceDescription instance must be related to a Collaboration in the Logical Service Model by a UML Realization, signifying the community that the ServiceDescription relates to. It must also be related to the corresponding information model – one or more NIEM-conformant IEPDs – by means of UML Usages. Linked to the ServiceDescription is a set of instances that represent the metadata describing capabilities, requirements, exchange partners and so on; these instances, together with the logical model and properties of the ServiceDescription instance, are used to drive provisioning of the SDD documentation and artifacts.

Accompanying the ServiceDescription instance are ServiceInterfaceSpecification instances that correspond to the Service Interfaces of the SPP. Each of these has properties and related instances that carry metadata to drive the provisioning of documentation and artifacts. It also has related Service instances and a ServiceInteractionProfile instance that together drive the provisioning of WSDL schemas. As far as possible, default generation rules are applied, so that the user only needs to model exceptions from the default situation.

## Example



«ModelPackageDescription»
**pet_adoption_iepd**
{ASName = "PetAdoption",
CreationDate = "2013-01-30",
descriptionText = "The model demonstrates NIEM-UML by modeling pet adoptions.",
DomainText = "Pet",
KeywordText = "Pet",
mpdBaseURI = "http://PetAdoption.com/mpd",
mpdClassCode = iepd,
mpdVersionID = "2.0",
POC = poc,
PurposeText = "The model demonstrates NIEM-UML by modeling pet adoptions.",
SecurityMarkingText = "Unclassified/public"}

Pet Adoption Service Description : ServiceDescription
AdditionalInformation = Additional Information
Classification = "Pet adoptions"
DataProvenance = Data Provenance
DomainDescription = "Pet products and services"
Endorsements = "Society of pet adoption agencies"
ExchangePartner = AdoptionCenter, Anyone
ExecutionContext = Execution Context
LifecycleStatus = "Pre production"
MajorVersion = 1
MinorVersion = 0
Privacy = Privacy
RealWorldEffect = Use Cases
RelatedOrganization = Organization of Adoption Centers
Requirement = Service Interaction Requirements
Security = Security
ServiceAssumption = ServiceAssumption
ServiceCapability = Capability
ServiceDescriptionKeywordText = "pet adoption"
ServiceDescriptionSummaryText = "This service provides pet adoption information"
ServiceID = "http://petadoption.com"
ServiceInterface = Pet Adoption Service Specification
ServiceLevelAgreement = SLA
ServiceNameAbbreviationText = "PET"
ServicePurpose = Purpose
ServiceScopeDescription = Scope
ServiceSecurityClassification = None
ServiceURI = "http://petadoption.com"
Sponsor = "Society of pet adoption agencies"

SLA : ServiceLevelAgreement
ServiceResponseTime = "1m"

Service Interaction Requirements : InteractionRequirements
IdentityAndAttributeAssertionTransmission = false
InterfaceDescriptionRequirements = false
Logging = false
MessageAddressing = true
MessageConfidentiality = false
MessageIntegrity = false
MessageNonrepudiation = false
Reliability = false
ServiceAuthentication = false
ServiceConsumerAuthentication = false
ServiceConsumerAuthorization = false
ServiceMetadataAvailability = false
ServiceResponsiveness = false
TransactionSupport = false

**Figure 9 Example annotations**

Figure 9 shows a partial example to illustrate some of these principles. A much more complete and comprehensive version of this example appears in Annex A

In Figure 9, Pet Adoption Service Description is an instance of the class ServiceDescription, using the UML notation for InstanceSpecifications. This contains values for the properties defined in the class ServiceDescription, such as ServiceURI = “http://petadoption.com”.

Some of the properties are also represented as links. In the example, the property ServiceLevelAgreement refers to the instance named SLA, an instance of the class ServiceLevelAgreement. The property Requirement is similarly shown.

Also shown on the diagram is a Realization from the Pet Adoption Service Description instance to the Collaboration called Pet Adoption Community. This is a part of the Logical Service Model which contains elements as described in clause 7.3. The Realization is notated as a dashed line with open triangular arrowhead.

Additionally a Usage dependency connects the Pet Adoption Service Description instance to a ModelPackageDescription component from the Information Model. This component is constructed using the UML Profile for NIEM, which defines the «ModelPackageDescription» stereotype shown on the component.

Shown below is a fragment of the annotations.xmi file that corresponds to this example. The top-level element of this fragment corresponds to the Pet Adoption Service Description instance. Within that element are attributes that correspond to many of the properties defined by the model. There is also a ModelReference element, derived by Phase-1, resulting from the Realization between the SIRSServiceDescription instance and the SIRS Community collaboration. Such derivations are discussed in the next section. The model reference contains a modelUri property which contains the name of a model file which will be located in the artifacts/service model folder.

```
<graa:ServiceDescription
       xmlns:xmi="http://www.omg.org/spec/XMI/20110701"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:graa="http://ijis.org/GRA/Annotations"
       xmi:id="_0"
       name="Pet Adoption Service Description"
       documentation="This service provides information sharing with pet adoption centers"
       serviceId="http://petadoption.com"
       serviceUri="http://petadoption.com"
       serviceNameAbbreviationText="PET"
       serviceDescriptionSummaryText="This service provides pet adoption information"
       domainDescription="Pet products and services"
       majorVersion="1"
       lifecycleStatus="Pre production">
  <requirement xmi:id="_0.requirement" messageAddressing="true"/>
  <modelReference xmi:id="_0.modelReference"
       name="Pet Adoption Community"
       elementId="_17_0_5_1_3ba019e_1407187737027_669498_17425">
    <model xmi:id="_0.modelReference.model"
           modelUri="PetSimpleService.xmi"
           label="PetSimpleService"/>
  </modelReference>
  ....

</graa:ServiceDescription> >
```

### 7.4.2 Derived properties

Many properties in the intermediate model are derived by Phase-1. For example, the property GRAServiceAnnotationBase::ModelReference is derived for any instance that has a Realization to an element in the Logical Service Model. The type of this property is the class ModelReference, which encapsulates a set of metadata used for traceability that identifies and describes the realized logical service element. This property is derived by Phase-1 and emitted into the intermediate annotations.xmi for consumption by Phase-2. The Annotations contains neither this property nor the type ModelReference: all the modeler needs to do is create the correct Realization.

Another example involves Documentation properties. In the user's model, text may be entered as documentation into a comment attached to the annotation instance: many UML tools distinguish the first comment of a model element through a special documentation field in the user interface and allow for HTML tags to be used, which is supported by GRA-UML. Alternatively the appropriate documentation might be found as a comment attached to a logical service element realized by the annotation instance. Either way, Phase-1 extracts the text from the appropriate comment and places it into a derived Documentation property for the Phase-2 XSLT to consume.

Table 2 specifies all of the properties that are derived by Phase-1, together with specifications of their derivations.

### 7.4.3 Defaults

To make the modeling process simpler and also to provide for flexibility in the technology and documentation artifacts produced, GRA-UML utilizes defaults at multiple levels. The following sections are intended to clarify the default mechanisms.

Some of these defaults are processed in Phase-1 whereas others are processed in Phase-2. Note that using the "template" mechanism of Phase-2, the Phase-2 defaults many be modified by the GRA-UML architect by providing an alternative template whereas the Phase-1 defaults are specified by the GRA-UML standard and are not modifiable.

Defaults fall into the following categories:

- Service hierarchy defaults, expanded in Phase-1
- Interaction Requirements defaults, expanded in Phase-2
- Property value defaults, provided in Phase-2

### 7.4.4 Service hierarchy defaults

A service specification and the typical WSDL artifacts produced from it follow a hierarchical pattern of:

- Service interface specification
  - Service
    - Port
      - Interface
        - Operation or Reception
          - Parameter
            - Message

The *output* from Phase-1will have an annotation instance for *every* service, port, interface, operation, parameter and message defined by the logical service model. These will correspond to the

- Component realized by the service
- Ports on those service components
- Interfaces provided by those ports
- Operations and Receptions contained within those interfaces
- Parameters on the operations
- Messages that are the data types of the parameters

It is not necessary for the user to construct the corresponding hierarchy of annotations corresponding to the complete LSM: Phase-1 does that, and will provide default annotations for every element in the hierarchy, together with ModelReferences to the logical elements they correspond to. So in the simplest case the user only needs to specify the Service annotation, and everything else will be generated from the structure of the Logical Service Model.

For more control the user may override what Phase-1 does, by providing *explicit* annotations and *defaults*.

The user may create a corresponding explicit hierarchy of annotation instances: Port, Interface, Operation, Parameter, Message, where each annotation instance must connect via a Realization to its corresponding element in the LSM. In this case, Phase-1 will use the explicit annotations wherever they are provided.

Alternatively, the user may create *default* metadata for any of these elements. These are modeled using properties called PortDefault, InterfaceDefault, OperationDefault, ParameterDefault and MessageDefault, which can be set by the user on any higher level of the hierarchy – so for example an OperationDefault can be specified for a Port, but not vice-versa. If there is just one such default set on an element, and that default is not marked to realize anything, then that default metadata will be expanded for every corresponding subsidiary element in the intermediate file. A default may also be marked to realize a particular corresponding LSM element, in which case that default will be expanded as annotations for only that realized element - see 8.2.12.

None of these default properties appear in the intermediate annotations.xmi, since they would have all been expanded in Phase-1.

For Interface, Operation, Parameter and Message annotations, the annotation library contains specialized WSDLInterface, WSDLOperation, WSDLParameter and WSDLMessage options that may be used when it is necessary to provide explicit non-default values for the generated WSDL elements.

## Examples

The Pet example from Annex A has a default as follows:

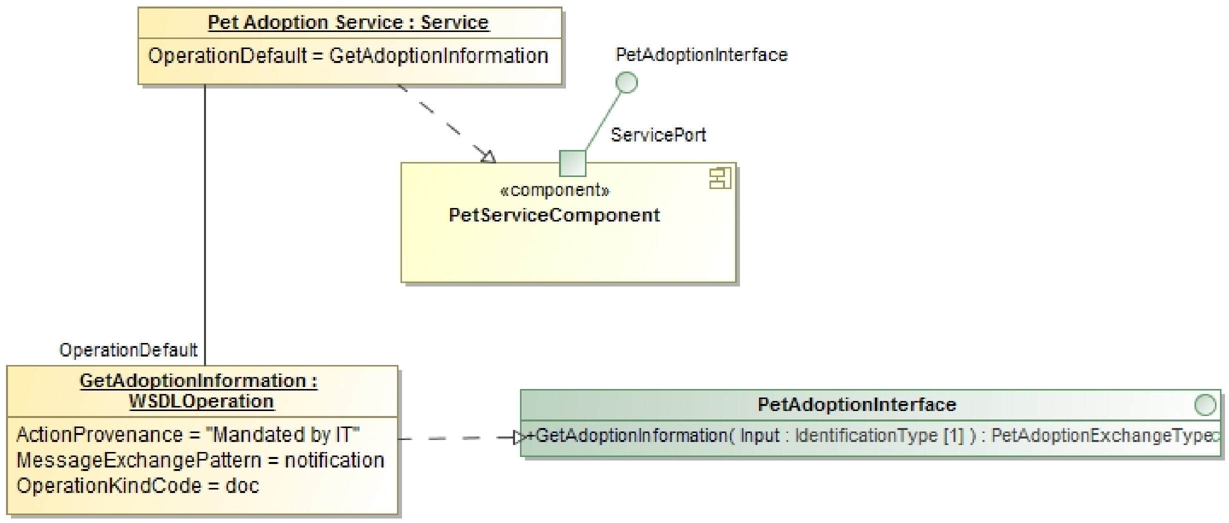

Pet Adoption Service : Service
OperationDefault = GetAdoptionInformation
PetAdoptionInterface
ServicePort
«component»
PetServiceComponent
OperationDefault
GetAdoptionInformation : WSDLOperation
ActionProvenance = "Mandated by IT"
MessageExchangePattern = notification
OperationKindCode = doc
PetAdoptionInterface
+GetAdoptionInformation( Input : IdentificationType [1] ) : PetAdoptionExchangeType

**Figure 10 Example operation default**

Figure 10 shows an operation default that realizes the operation GetAdoptionInformation. This specifies that the realized operation will have a MessageExchangePattern of “notification”, an OperationKindCode of “doc” and an ActionProvenance of “Mandated by IT”.

If there were no realization from this default, then it would apply to all operations within the hierarchical scope of Pet Adoption Service, the declarer of the default.

If necessary, the user may create an explicit hierarchy of Port, Interface, Operation, Parameter and Message annotations, explicitly realizing respectively their corresponding logical elements. An example is shown in Figure 11[1], which shows WSDL-specific options in an explicit hierarchy. Typically defining such a hierarchy is more verbose than using defaults.

[1] In this example the WSDLParameter annotation does in fact realize the Input parameter of the GetAdoptionInformation operation, but this is not shown on the diagram because the tool cannot display it.

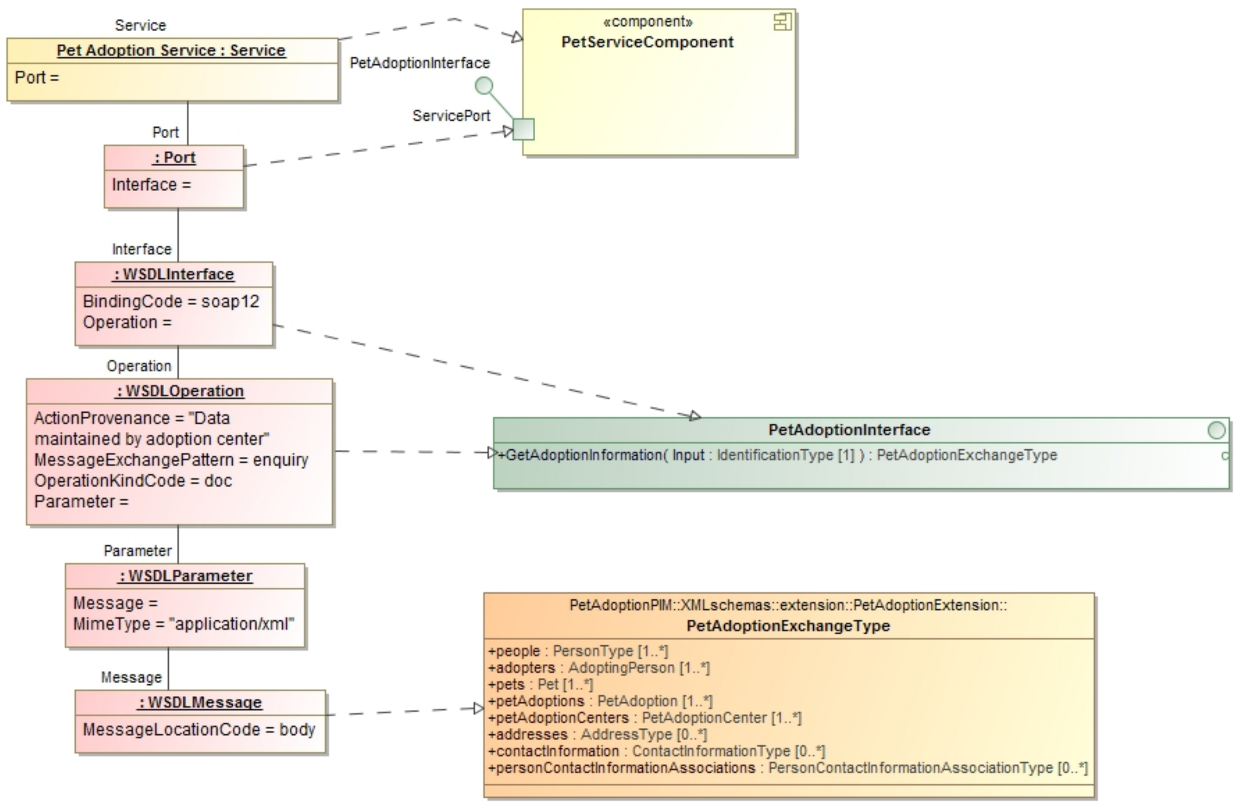

Service
Pet Adoption Service : Service
Port =
«component»
PetServiceComponent
PetAdoptionInterface
ServicePort
Port
: Port
Interface =
Interface
: WSDLInterface
BindingCode = soap12
Operation =
Operation
: WSDLOperation
ActionProvenance = "Data maintained by adoption center"
MessageExchangePattern = enquiry
OperationKindCode = doc
Parameter =
PetAdoptionInterface
+GetAdoptionInformation( Input : IdentificationType [1] ) : PetAdoptionExchangeType
Parameter
: WSDLParameter
Message =
MimeType = "application/xml"
Message
: WSDLMessage
MessageLocationCode = body
PetAdoptionPIM::XMLschemas::extension::PetAdoptionExtension::
PetAdoptionExchangeType
+people : PersonType [1..*]
+adopters : AdoptingPerson [1..*]
+pets : Pet [1..*]
+petAdoptions : PetAdoption [1..*]
+petAdoptionCenters : PetAdoptionCenter [1..*]
+addresses : AddressType [0..*]
+contactInformation : ContactInformationType [0..*]
+personContactInformationAssociations : PersonContactInformationAssociationType [0..*]

**Figure 11 Explicit annotation hierarchy**

### 7.4.5 Interaction requirements defaults

Interaction requirements specify the business requirements for a service interaction. Interaction requirements are specified using the following class:

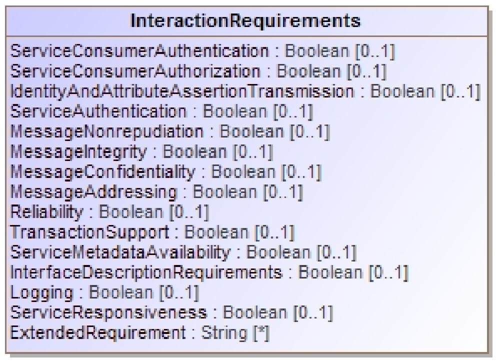

InteractionRequirements
ServiceConsumerAuthentication : Boolean [0..1]
ServiceConsumerAuthorization : Boolean [0..1]
IdentityAndAttributeAssertionTransmission : Boolean [0..1]
ServiceAuthentication : Boolean [0..1]
MessageNonrepudiation : Boolean [0..1]
MessageIntegrity : Boolean [0..1]
MessageConfidentiality : Boolean [0..1]
MessageAddressing : Boolean [0..1]
Reliability : Boolean [0..1]
TransactionSupport : Boolean [0..1]
ServiceMetadataAvailability : Boolean [0..1]
InterfaceDescriptionRequirements : Boolean [0..1]
Logging : Boolean [0..1]
ServiceResponsiveness : Boolean [0..1]
ExtendedRequirement : String [*]

**Figure 12 InteractionRequirements**

Interaction requirements are meaningful through the full hierarchy of an SSP, starting with the “Service Description” and proceeding down the service hierarchy described above.

The InteractionRequirements specified at the level of the SSP are considered business requirements and are included in the GRA SDD document. These requirements also serve as a default for all the lower levels of the hierarchy, but any level of the hierarchy may override the more general requirements and specify the interaction requirements for that level.

If the interaction requirements are the same for the entire SSP there is only the need for the one InteractionRequirement annotation element. However it is common for specific ports, interfaces or operations to have more specific needs. To define these more specific needs a “Requirement” is linked to the more specific element and the requirement properties filled in.

Any property not specifically set to true or false in any InteractionRequirement element will acquire its value from a higher level, or if there is no such value, a default value for the property as specified by Table 3.

Phase-1 will only output the InteractionRequirements specifically provided by the model; the processing of these defaults is handled by Phase-2.

### 7.4.6 Property value defaults

There are many elements that are required by the WSDL or GRA specification that are typically filled in with “boilerplate” or other typical values. To simplify the model and also to provide flexibility as to the contents of this boilerplate, certain required properties may be omitted from the service specification. Phase-1 will fill in values that can be determined from the PIM. Phase-2 will then fill in any missing values with “boilerplate”. Any value that is specifically provided in the GRA model will take precedence over the boilerplate default.

Table 2 details the property values that will be calculated by Phase-1. Table 3 details the property values that will be used by Phase-2, if no value is provided by Phase-1. The end-result should be a valid GRA SSP with all mandatory values included.

### Example

The GRA specification requires a “Security” section. Many SSPs use a standard text of the form:

> The service must adhere to the rules of the CJIS Security Policies regarding the Advanced Encryption Standard (AES) level of encryption. The use of Virtual Private Networks (VPNs) or Secure Sockets Layer (SSL) for transport-level security in addition to these requirements is optional.

If the “Security” property of a ServiceDescription has no value provided, Phase-2 will insert the above text.

### 7.4.7 Hierarchical Structure

XSLT transformations work best on hierarchically-structured documents, and the default Phase-2 uses XSLTs that consume the output of Phase-1. So the intermediate form is designed to be hierarchical, and this is represented in the Intermediate Model by numerous compositions (also known as composite aggregations, signified in UML by black diamond adornments). From the modeler’s perspective, however, it may be inconvenient to structure the annotations hierarchically: the same annotation may be shared between several elements, and this is best represented simply by linking them together. So in the Annotations there are no compositions; one of the tasks of Phase-1 is to duplicate annotation elements shared in the user’s model so that they are hierarchically represented in the intermediate model.

An example of this approach is the property GRAServiceAnnotationBase::InteractionRequirements. This allows the attaching of an instance of InteractionRequirements at any level of the model from the complete ServiceDescription down to the individual Message. This instance provides a bundle of metadata specifying interaction requirements for the element to which it is attached – see the description in 7.4.5, above. The user might share a single instance of InteractionRequirements amongst, say, a set of Operations; Phase-1 processing would replicate that instance so that a copy is hierarchically contained with the metadata for all of those operations in the intermediate annotations.xmi.

Clause 9 specifies where these compositions differ between the libraries.

# 8 GRA-UML Logical Services Profile

## 8.1 UML subset

GRA-UML Logical Service Models are normal UML models. Any UML construct can be used within a GRA LSM. However, only certain elements within the LSM are interpreted by the SSP provisioning process. Clause 8.2 documents the UML classes which are used to construct the LSM. Other elements may be used for documentation purposes, or as input to other provisioning processes outside the scope of this specification. The complete Logical Service Model is itself part of the provisioned SSP, appearing in the artifacts/service model folder in subfolders named by the format of the model file, e.g. OMG, Eclipse, MagicDraw.

## 8.2 Logical service classes

This section specifies the UML classes whose instances affect Phase-1 of SSP provisioning. These are classes from the complete merged L3 version of UML 2.4.1, in which each is fully identified by their name, so for example the xmi definition of Actor may be found at http://www.omg.org/spec/UML/20110701/UML.xmi#Actor.

Elements in the model that do not have an effect documented here or in [NIEM-UML] are ignored by Phase-1.

Examples of the use of these elements may be found in clause 7.3.

### 8.2.1 Actor

A UML Actor represents a participant in the SSP subject community. Actors are used by Phase-1 to populate the property ServiceDescription::ExchangePartner. Actors may be directly realized by Participants in the annotation model, or may be contained in realized Packages (see 8.2.8). Also where an Actor plays a role in a Collaboration realized by the ServiceDescription, a corresponding ExchangePartner is generated, if it is not already present.

### 8.2.2 Association

A UML Association between an Actor and a UseCase stereotyped as «Provider» or «Consumer» (see 8.3) specifies whether the ExchangePartner represented by the Actor is a provider or consumer of the RealWorldEffect represented by the UseCase. Such an Association is used by Phase-1 to populate the annotation properties UseCase::Provider and UseCase::Consumer.

### 8.2.3 Collaboration

A UML Collaboration represents the community involved in the SSP. It must be realized by the ServiceDescription that represents the SSP. Phase-1 generates a ModelReference from this realization, and ServiceInteractions that correspond to the Interactions owned by the Collaboration. Where an Actor plays a role, or is realized by a Component that plays a role, in a Collaboration realized by the ServiceDescription, a corresponding ExchangePartner is generated, if it is not already present.

### 8.2.4 Component

A UML Component represents a service.

A Component may realize an Actor, signifying that it represents a service provided by the Participant represented by that Actor. Such a Realization is used by Phase-1 to populate the property Service::ServiceProvider.

A Component may be realized by a Service annotation, providing the foundation for a service interface description. Each service interface description must have a Service realizing a Component. Phase-1 uses this Realization to generate the service hierarchy (see 7.4.4).

### 8.2.5 Interaction

A UML Interaction represents a service interaction. Each Interaction owned by the top-level Collaboration will cause the generation of a ServiceInteraction whose Participants realize Actors that participate in the Interaction.

### 8.2.6 Interface

A UML Interface represents an interface on the port that provides the interface. Phase-1 uses Port::provided to generate the service hierarchy (see 7.4.4).

### 8.2.7 Operation

A UML Operation represents a synchronous operation on the interface that owns it. Phase-1 uses Interface::ownedOperation to generate the service hierarchy (see 7.4.4), and for each Operation generates an operation annotation with IsAsynchronous = false. If the Operation has any raisedExceptions, these cause the generation of parameter annotations with Use = ParameterUse::exception.

### 8.2.8 Package

A UML Package may be used to collect together Actors or UseCases. Where Actors are contained by Packages, those Packages may be realized by Participants in the annotation model, in which case Phase-1 generates an ExchangePartner for each Actor contained in the realized Package. Where UseCases are contained by Packages, those Packages may be realized by UseCases in the annotation model, in which case Phase-1 generates a RealWorldEffect for each UseCase contained in the realized Package.

### 8.2.9 Parameter

A UML Parameter represents a parameter to the operation that owns it. Phase-1 uses Operation::ownedParameter and Reception::ownedParameter (or, equivalently, Reception::signal.ownedAttribute) to generate the service hierarchy (see 7.4.4), and for each Parameter generates a parameter annotation. If the Operation or Reception has any raisedExceptions, these cause the generation of parameter annotations with Use = ParameterUse::exception, and whose message refers to the raisedException type.

### 8.2.10 Port

A UML Port represents a port on the Service represented by the Component that owns the Port. Phase-1 uses Component::ownedPort to generate the service hierarchy (see 7.4.4).

### 8.2.11 Reception

A UML Reception represents an asynchronous operation on the interface that owns it. Phase-1 uses Interface::ownedReception to generate the service hierarchy (see 7.4.4), and for each Reception generates an operation annotation with IsAsynchronous = true. If the Reception has any raisedExceptions, these cause the generation of parameter annotations with Use = ParameterUse::exception.

### 8.2.12 Realization

UML Realizations are primarily used to relate annotations to the Logical Service Model elements that they annotate. The following table specifies which Realizations are processed by Phase-1. A name preceded by a colon (e.g. :Participant) signifies that the Realization is connected to an InstanceSpecification classified by the named annotation type. A name without a colon (e.g. Actor) signifies that the Realization is connected to a UML model element of the named type. These definitions apply equally to subtypes, e.g. a :WSDLOperation annotation follows the same rules as an :Operation annotation.

Any Realizations not in this table have no effect.

**Table 1 Realizations**

| Source | Target | Phase-1 result |
|---|---|---|
| :ServiceDescription | Collaboration | A ServiceDescription instance is generated containing ServiceInteractions for every Interaction owned by the Collaboration, and an exchange partner for every Actor that plays a role in any of those Interactions. |
| :Participant | Actor | A single Participant instance is generated corresponding to the Actor. |
| :Participant | Package | A Participant instance is generated corresponding to every Actor in the Package. |
| :UseCase | UseCase | A single UseCase instance is generated representing the real world effect. |
| :UseCase | Package | A UseCase instance is generated corresponding to every UseCase in the Package. |
| :Service | Component | A Service instance is generated whose hierarchical contents are systematically derived from the hierarchical contents of the Component. |
| Component | Actor | A :Service that realizes the Component shall contain a serviceProvider element corresponding to the Actor. |
| :Port | Port | A Port instance generated in the service hierarchy corresponding to the target Port is explicitly specified by the source :Port annotation. |
| :Interface | Interface | A Interface instance generated in the service hierarchy corresponding to the target Interface is explicitly specified by the source :Interface annotation. |
| :Operation | Operation | An Operation instance generated in the service hierarchy corresponding to the target Operation is explicitly specified by the source :Operation annotation. |
| :Operation | Reception | An Operation instance generated in the service hierarchy corresponding to the target Reception is explicitly specified by the source :Operation annotation. |
| :Parameter | Parameter | A Parameter instance generated in the service hierarchy corresponding to the target Parameter is explicitly specified by the source :Parameter annotation. |
| :Message | Type | If the source :Message is a MessageDefault for an element E, then within the scope of E, Message instances generated in the service hierarchy that correspond to the target type shall contain the properties specified by the source :Message annotation. |

### 8.2.13 Signal

A UML Signal is associated with each Reception that represents an asynchronous operation. The attributes of the Signal match the parameters of the Reception according to UML rules.

### 8.2.14 Type

UML Types represent the information model according to the principles defined by [NIEM-UML].

### 8.2.15 Usage

A UML Usage is used to relate a ServiceDescription to each IEPD that it uses.  Each Usage is used by Phase-1 to generate an IEPDReference in the ServiceDescription.

### 8.2.16 UseCase

A UML UseCase represents a “real world effect” – the user’s understanding of a service. UseCases are used by Phase-1 to populate the property ServiceDescription::RealWorldEffect. UseCase annotations may be directly realized by UML UseCases, or may be contained in realized Packages (see 8.2.8).

## 8.3 Stereotypes

GRA-UML defines two stereotypes, shown in Figure 13. «Provider» and «Consumer» both extend the UML metaclass Association, and are used in use case models such as that shown in Figure 4.

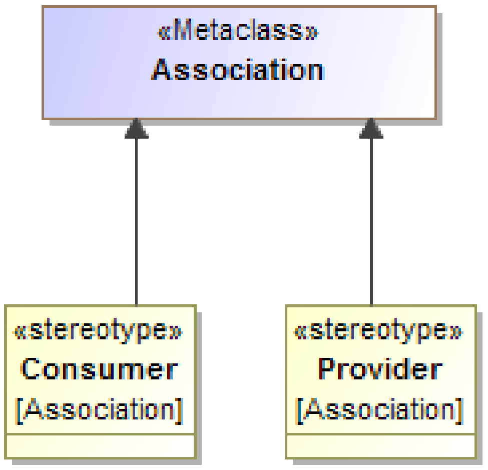


**Figure 13 GRA-UML stereotypes**

The UML profile for NIEM [NIEM-UML], is used to define the information models referenced by GRA-UML service models.

Stereotypes from other UML profiles, including [SoaML], are legal in GRA-UML models but are ignored by SSP provisioning.

### 8.3.1 «Provider»

The Provider stereotype may be applied to a UML Association between an Actor and UseCase. It signifies that the ExchangePartner represented by the Actor is a provider of the RealWorldEffect represented by the UseCase.

### 8.3.2 «Consumer»

The Consumer stereotype may be applied to a UML Association between an Actor and UseCase. It signifies that the ExchangePartner represented by the Actor is a consumer of the RealWorldEffect represented by the UseCase.

# 9 GRA Annotations

This clause documents the Annotation and Intermediate models. As explained in 7.4.1, these two models are very similar, and in this clause they are documented together, with the differences marked where they occur. Both models are divided into two packages, called GRAAnnotationModel and GRAWsdl. The latter package contains just the classes for WSDL-specific annotations, and is identical in both of its forms: Phase-1 does no processing on it and simply passes the input values through to the intermediate model.

## 9.1 Package GRAUML::GRAAnnotationModel

For this package Figure 14 shows the Annotations form, and Figure 15 the Intermediate model form. In summary the models differ in the following ways:

- The Annotations model enables the modeling of service hierarchy defaults; these properties do not appear in the Intermediate model as these defaults are fully expanded in phase-1.
- The Annotation model contains constraints which apply to the user's model. These constraints do not apply to the intermediate model.
- The Annotation model is used as a UML Library for the GRA architect's modeling.
- The Intermediate model contains properties that are calculated by Phase-1, as specified in 9.3[2].
- The Intermediate model is constructed using compositions (black diamonds), while the Annotations model does not contain compositions except for properties with primitive types and enumerations.
- The Intermediate model is used as an EMOF metamodel which forms the target of Phase-1 QVT processing, and which corresponds to XML Schemas which are accessed by XSLTs that execute Phase-2 processing.

### *NOTE: Constraint evaluation*

*As noted above, the Annotations model contains constraints. These are formulated using [OCL] (Object Constraint Language). The observant reader may notice that these constraints are strictly-speaking invalid, because models created using the Annotations model consist primarily of InstanceSpecifications whose classifiers come from the Annotations model. OMG standard OCL does not recognize InstanceSpecifications, therefore the context for evaluating these constraints is formally incorrect.*

*The constraints in GRA-UML are written on the assumption that a constraint evaluator exists which can extend the context for an InstanceSpecification to behave also as though it were an instance of its classifier. So, for example, a constraint written on the Annotations class Message will be discovered by this evaluator when it executes against an InstanceSpecification whose classifier is Message, and the slots of the InstanceSpecification will be accessed as though they were properties of a Message instance. Inherent properties of InstanceSpecification (in particular clientDependency) are also assumed to be available in the extended context, and are accessed by coercing the type by means of the operation oclAsType(InstanceSpecification).*

[2] Note that these derived properties are marked with visibility Protected (#) in the diagram. This marking has no semantic consequence and is only used for model management.

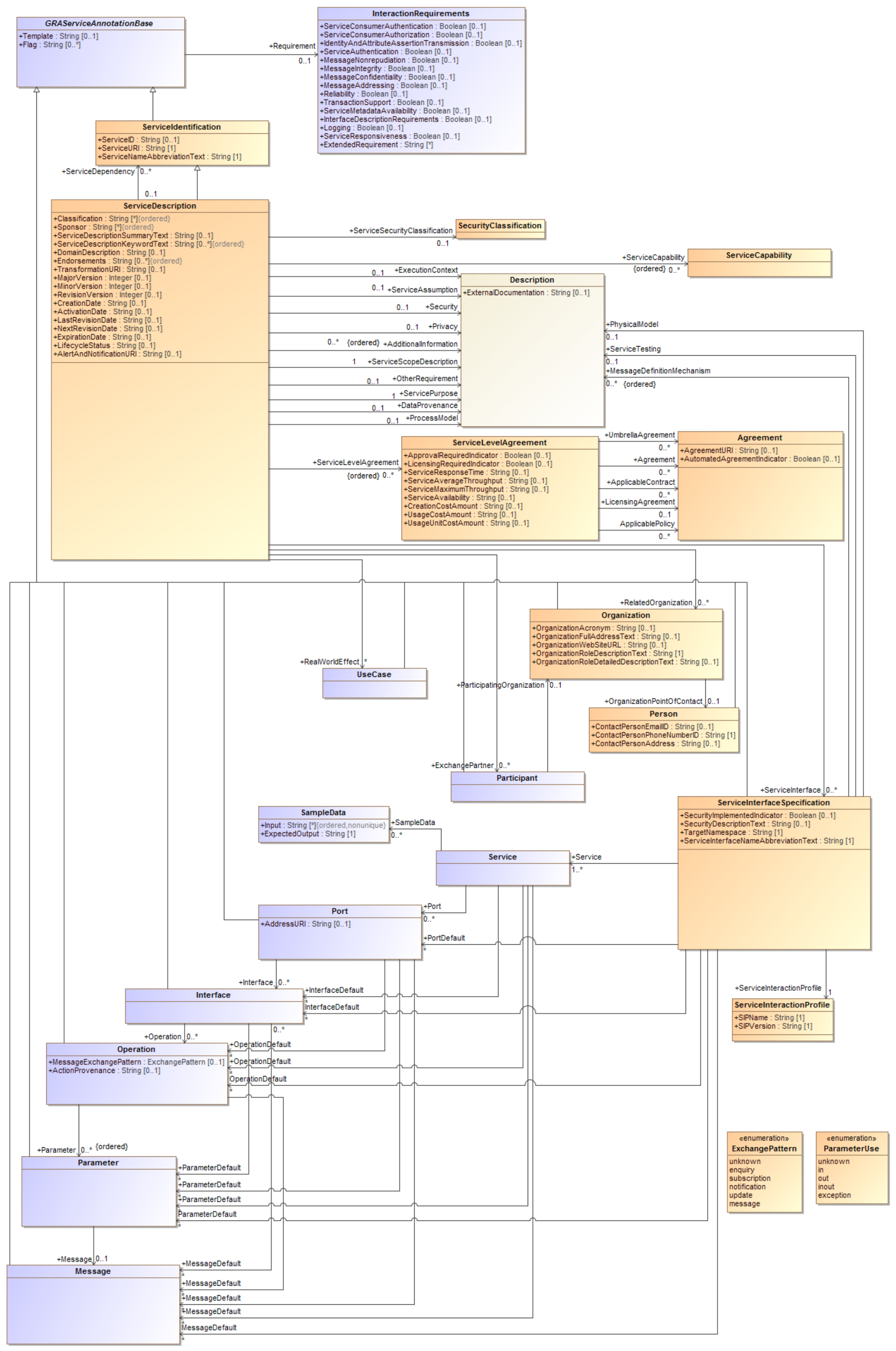

GRAServiceAnnotationBase
+Template : String [0..1]
+Flag : String [0..*]
+Requirement
0..1
InteractionRequirements
+ServiceConsumerAuthentication : Boolean [0..1]
+ServiceConsumerAuthorization : Boolean [0..1]
+IdentityAndAttributeAssertionTransmission : Boolean [0..1]
+ServiceAuthentication : Boolean [0..1]
+MessageNonrepudiation : Boolean [0..1]
+MessageIntegrity : Boolean [0..1]
+MessageConfidentiality : Boolean [0..1]
+MessageAddressing : Boolean [0..1]
+Reliability : Boolean [0..1]
+TransactionSupport : Boolean [0..1]
+ServiceMetadataAvailability : Boolean [0..1]
+InterfaceDescriptionRequirements : Boolean [0..1]
+Logging : Boolean [0..1]
+ServiceResponsiveness : Boolean [0..1]
+ExtendedRequirement : String [*]
ServiceIdentification
+ServiceID : String [0..1]
+ServiceURI : String [1]
+ServiceNameAbbreviationText : String [1]
+ServiceDependency
0..*
0..1
ServiceDescription
+Classification : String [*]{ordered}
+Sponsor : String [*]{ordered}
+ServiceDescriptionSummaryText : String [0..1]
+ServiceDescriptionKeywordText : String [0..*]{ordered}
+DomainDescription : String [0..1]
+Endorsements : String [0..*]{ordered}
+TransformationURI : String [0..1]
+MajorVersion : Integer [0..1]
+MinorVersion : Integer [0..1]
+RevisionVersion : Integer [0..1]
+CreationDate : String [0..1]
+ActivationDate : String [0..1]
+LastRevisionDate : String [0..1]
+NextRevisionDate : String [0..1]
+ExpirationDate : String [0..1]
+LifecycleStatus : String [0..1]
+AlertAndNotificationURI : String [0..1]
+ServiceSecurityClassification
0..1
SecurityClassification
+ServiceCapability
{ordered} 0..*
ServiceCapability
0..1 +ExecutionContext
0..1 +ServiceAssumption
0..1 +Security
0..1 +Privacy
0..* {ordered} +AdditionalInformation
1 +ServiceScopeDescription
0..1 +OtherRequirement
1 +ServicePurpose
0..1 +DataProvenance
0..1 +ProcessModel
Description
+ExternalDocumentation : String [0..1]
+PhysicalModel
0..1
+ServiceTesting
0..1
+MessageDefinitionMechanism
0..* {ordered}
+ServiceLevelAgreement
{ordered} 0..*
ServiceLevelAgreement
+ApprovalRequiredIndicator : Boolean [0..1]
+LicensingRequiredIndicator : Boolean [0..1]
+ServiceResponseTime : String [0..1]
+ServiceAverageThroughput : String [0..1]
+ServiceMaximumThroughput : String [0..1]
+ServiceAvailability : String [0..1]
+CreationCostAmount : String [0..1]
+UsageCostAmount : String [0..1]
+UsageUnitCostAmount : String [0..1]
+UmbrellaAgreement
0..*
+Agreement
0..*
+ApplicableContract
0..*
+LicensingAgreement
0..1
ApplicablePolicy
0..*
Agreement
+AgreementURI : String [0..1]
+AutomatedAgreementIndicator : Boolean [0..1]
+RelatedOrganization 0..*
Organization
+OrganizationAcronym : String [0..1]
+OrganizationFullAddressText : String [0..1]
+OrganizationWebSiteURL : String [0..1]
+OrganizationRoleDescriptionText : String [1]
+OrganizationRoleDetailedDescriptionText : String [0..1]
+RealWorldEffect 0..*
UseCase
+ParticipatingOrganization 0..1
+OrganizationPointOfContact 0..1
Person
+ContactPersonEmailID : String [0..1]
+ContactPersonPhoneNumberID : String [1]
+ContactPersonAddress : String [0..1]
+ExchangePartner 0..*
Participant
+ServiceInterface 0..*
ServiceInterfaceSpecification
+SecurityImplementedIndicator : Boolean [0..1]
+SecurityDescriptionText : String [0..1]
+TargetNamespace : String [1]
+ServiceInterfaceNameAbbreviationText : String [1]
SampleData
+Input : String [*]{ordered,nonunique}
+ExpectedOutput : String [1]
+SampleData
0..*
Service
+Service
1..*
Port
+AddressURI : String [0..1]
+Port
0..*
+PortDefault
*
+ServiceInteractionProfile
1
ServiceInteractionProfile
+SIPName : String [1]
+SIPVersion : String [1]
+Interface 0..*
Interface
+InterfaceDefault
*
InterfaceDefault
*
0..*
+Operation 0..*
Operation
+MessageExchangePattern : ExchangePattern [0..1]
+ActionProvenance : String [0..1]
+OperationDefault
*
+OperationDefault
*
OperationDefault
*
«enumeration»
ExchangePattern
unknown
enquiry
subscription
notification
update
message
«enumeration»
ParameterUse
unknown
in
out
inout
exception
+Parameter 0..* {ordered}
Parameter
+ParameterDefault
*
+ParameterDefault
*
+ParameterDefault
*
ParameterDefault
*
+Message 0..1
Message
+MessageDefault
*
+MessageDefault
*
+MessageDefault
*
+MessageDefault
*
MessageDefault

**Figure 14 Annotations model**

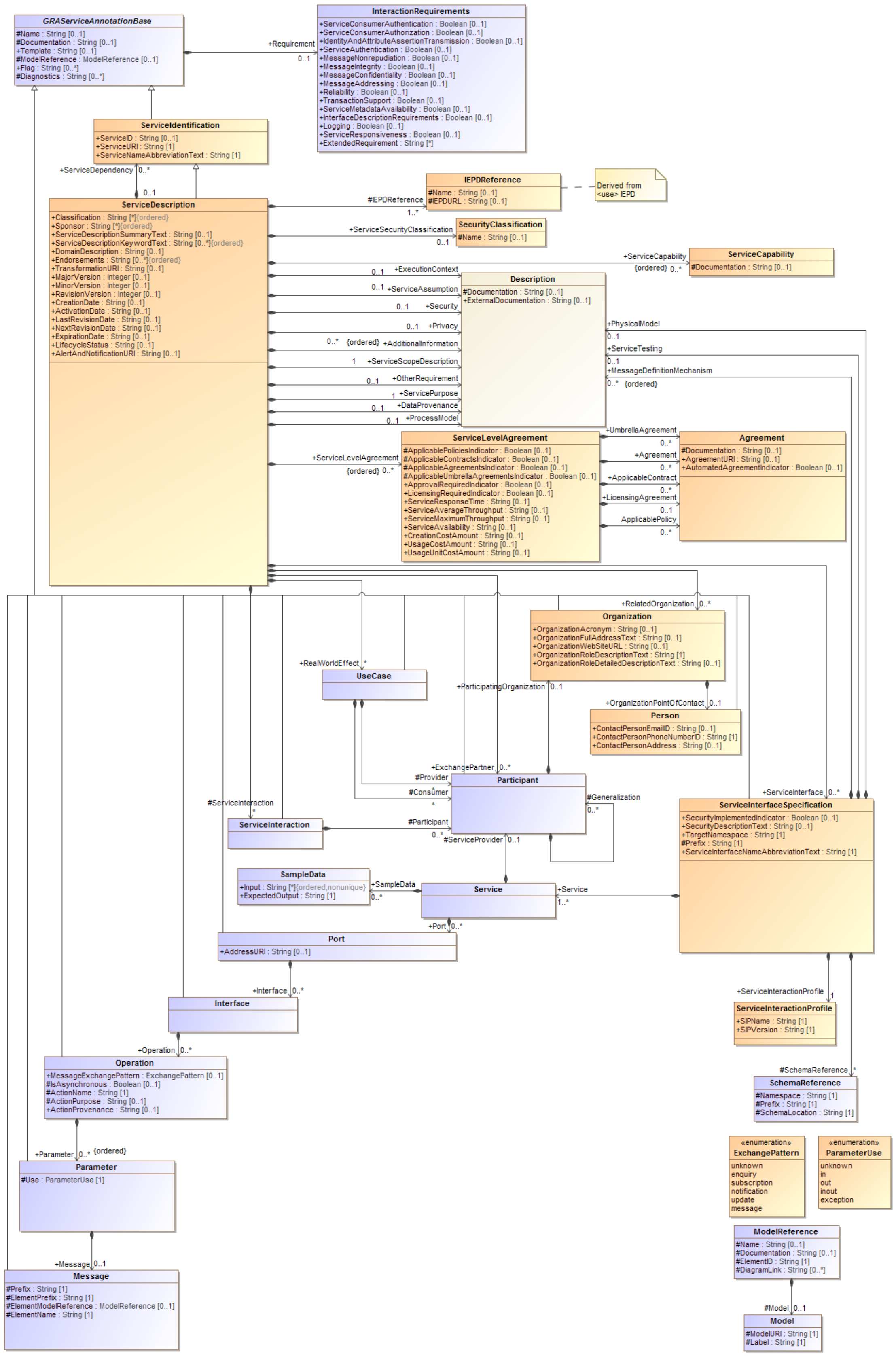

GRAServiceAnnotationBase
#Name : String [0..1]
#Documentation : String [0..1]
+Template : String [0..1]
#ModelReference : ModelReference [0..1]
+Flag : String [0..*]
#Diagnostics : String [0..*]
InteractionRequirements
+ServiceConsumerAuthentication : Boolean [0..1]
+ServiceConsumerAuthorization : Boolean [0..1]
+IdentityAndAttributeAssertionTransmission : Boolean [0..1]
+ServiceAuthentication : Boolean [0..1]
+MessageNonrepudiation : Boolean [0..1]
+MessageIntegrity : Boolean [0..1]
+MessageConfidentiality : Boolean [0..1]
+MessageAddressing : Boolean [0..1]
+Reliability : Boolean [0..1]
+TransactionSupport : Boolean [0..1]
+ServiceMetadataAvailability : Boolean [0..1]
+InterfaceDescriptionRequirements : Boolean [0..1]
+Logging : Boolean [0..1]
+ServiceResponsiveness : Boolean [0..1]
+ExtendedRequirement : String [*]
+Requirement
0..1
ServiceIdentification
+ServiceID : String [0..1]
+ServiceURI : String [1]
+ServiceNameAbbreviationText : String [1]
+ServiceDependency
0..*
0..1
IEPDReference
#Name : String [0..1]
#IEPDURL : String [0..1]
Derived from <use> IEPD
#IEPDReference
1..*
ServiceDescription
+Classification : String [*]{ordered}
+Sponsor : String [*]{ordered}
+ServiceDescriptionSummaryText : String [0..1]
+ServiceDescriptionKeywordText : String [0..*]{ordered}
+DomainDescription : String [0..1]
+Endorsements : String [0..*]{ordered}
+TransformationURI : String [0..1]
+MajorVersion : Integer [0..1]
+MinorVersion : Integer [0..1]
+RevisionVersion : Integer [0..1]
+CreationDate : String [0..1]
+ActivationDate : String [0..1]
+LastRevisionDate : String [0..1]
+NextRevisionDate : String [0..1]
+ExpirationDate : String [0..1]
+LifecycleStatus : String [0..1]
+AlertAndNotificationURI : String [0..1]
SecurityClassification
#Name : String [0..1]
+ServiceSecurityClassification
0..1
ServiceCapability
#Documentation : String [0..1]
+ServiceCapability
{ordered} 0..*
Description
#Documentation : String [0..1]
+ExternalDocumentation : String [0..1]
0..1 +ExecutionContext
0..1 +ServiceAssumption
0..1 +Security
0..1 +Privacy
0..* {ordered} +AdditionalInformation
1 +ServiceScopeDescription
0..1 +OtherRequirement
1 +ServicePurpose
0..1 +DataProvenance
0..1 +ProcessModel
+PhysicalModel
0..1
+ServiceTesting
0..1
+MessageDefinitionMechanism
0..* {ordered}
ServiceLevelAgreement
#ApplicablePoliciesIndicator : Boolean [0..1]
#ApplicableContractsIndicator : Boolean [0..1]
#ApplicableAgreementsIndicator : Boolean [0..1]
#ApplicableUmbrellaAgreementsIndicator : Boolean [0..1]
+ApprovalRequiredIndicator : Boolean [0..1]
+LicensingRequiredIndicator : Boolean [0..1]
+ServiceResponseTime : String [0..1]
+ServiceAverageThroughput : String [0..1]
+ServiceMaximumThroughput : String [0..1]
+ServiceAvailability : String [0..1]
+CreationCostAmount : String [0..1]
+UsageCostAmount : String [0..1]
+UsageUnitCostAmount : String [0..1]
+ServiceLevelAgreement
{ordered} 0..*
Agreement
#Documentation : String [0..1]
+AgreementURI : String [0..1]
+AutomatedAgreementIndicator : Boolean [0..1]
+UmbrellaAgreement
0..*
+Agreement
0..*
+ApplicableContract
0..*
+LicensingAgreement
0..1
ApplicablePolicy
0..*
+RelatedOrganization
0..*
Organization
+OrganizationAcronym : String [0..1]
+OrganizationFullAddressText : String [0..1]
+OrganizationWebSiteURL : String [0..1]
+OrganizationRoleDescriptionText : String [1]
+OrganizationRoleDetailedDescriptionText : String [0..1]
+RealWorldEffect
*
UseCase
+ParticipatingOrganization
0..1
+OrganizationPointOfContact
0..1
Person
+ContactPersonEmailID : String [0..1]
+ContactPersonPhoneNumberID : String [1]
+ContactPersonAddress : String [0..1]
+ExchangePartner
0..*
#Provider
#Consumer
*
Participant
#Generalization
0..*
#ServiceInteraction
*
ServiceInteraction
#Participant
0..*
#ServiceProvider
0..1
+ServiceInterface
0..*
ServiceInterfaceSpecification
+SecurityImplementedIndicator : Boolean [0..1]
+SecurityDescriptionText : String [0..1]
+TargetNamespace : String [1]
#Prefix : String [1]
+ServiceInterfaceNameAbbreviationText : String [1]
SampleData
+Input : String [*]{ordered,nonunique}
+ExpectedOutput : String [1]
+SampleData
0..*
Service
+Service
1..*
+Port
0..*
Port
+AddressURI : String [0..1]
+Interface
0..*
Interface
+ServiceInteractionProfile
1
ServiceInteractionProfile
+SIPName : String [1]
+SIPVersion : String [1]
+Operation
0..*
Operation
+MessageExchangePattern : ExchangePattern [0..1]
#IsAsynchronous : Boolean [0..1]
#ActionName : String [1]
#ActionPurpose : String [0..1]
+ActionProvenance : String [0..1]
#SchemaReference
*
SchemaReference
#Namespace : String [1]
#Prefix : String [1]
#SchemaLocation : String [1]
+Parameter
0..* {ordered}
Parameter
#Use : ParameterUse [1]
«enumeration»
ExchangePattern
unknown
enquiry
subscription
notification
update
message
«enumeration»
ParameterUse
unknown
in
out
inout
exception
ModelReference
#Name : String [0..1]
#Documentation : String [0..1]
#ElementID : String [1]
#DiagramLink : String [0..*]
+Message
0..1
Message
#Prefix : String [1]
#ElementPrefix : String [1]
#ElementModelReference : ModelReference [0..1]
#ElementName : String [1]
#Model
0..1
Model
#ModelURI : String [1]
#Label : String [1]

**Figure 15 Intermediate model**

### 9.1.1 Class Agreement

A data structure representing an agreement, formal or informal, associated with a service, application or thing, tangible or otherwise.

| | |
|---|---|
| **Name** | Agreement |
| **Abstract** | false |
| **Base Classifier** | |

## Properties

### AgreementURI

A locator referencing an agreement, formal or informal, associated with a service, application or thing, tangible or otherwise.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### AutomatedAgreementIndicator

Indicates whether the policy or contract represented by the Agreement has an automated implementation.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### [Intermediate Model Only] Documentation

A free text description of an agreement, formal or informal, associated with a service, application or thing, tangible or otherwise.  Derived from the instance documentation.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### 9.1.2 Class Description

A data structure used for the types of properties that are intended to document an element, with the possibility of a link to external documentation.

| | |
|---|---|
| **Name** | Description |
| **Abstract** | false |
| **Base Classifier** | |

### Properties

#### [Intermediate Model Only] Documentation

The documentation for the element. Derived from the instance documentation.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### ExternalDocumentation

Relative URL of an artifact which documents the element.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## 9.1.3 Class GRAServiceAnnotationBase

Abstract base class for GRA Service annotation metadata classes.

| | |
|---|---|
| **Name** | GRAServiceAnnotationBase |
| **Abstract** | true |
| **Base Classifier** | |

### Properties

#### [Intermediate Model Only] Diagnostics

Diagnostic information for this element.  Derived during phase-1 provisioning.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite |

#### [Intermediate Model Only] Documentation

The documentation for this element.  Derived from a realized or corresponding element's documentation or, if not provided, the instance's documentation.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## Flag

Flag provides an arbitrary extension mechanism whereby any business or technology choice may be notated on the containing element. The intent is that a user-supplied phase-2 template will then respect these flags and produce the desired result. Unknown flags are ignored.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite |

## [Intermediate Model Only] ModelReference

Derived reference to a model element realized by or corresponding to this element, used for traceability.

| | |
|---|---|
| **Type** | ModelReference |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## [Intermediate Model Only] Name

The element name. Derived from the instance's name, or if that is empty to the corresponding element's name.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## Requirement

The service interaction requirements.

| | |
|---|---|
| **Type** | InteractionRequirements |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## Template

Template is the name of a phase-2 template that will be called to process the containing element thus overriding the default mapping.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### 9.1.4 [Intermediate Model Only] Class IEPDReference

A data structure describing the name and location of an IEPD. An annotation element may be related to an IEPD by means of a Usage dependency, in which case the properties of this structure are derived from the used IEPD.

| | |
|---|---|
| **Name** | IEPDReference |
| **Abstract** | false |
| **Base Classifier** | |

#### Properties

#### IEPDURL

A URL where the IEPD is posted and available.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### Name

A human readable identification that is the name of the IEPD.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### 9.1.5 Class InteractionRequirements

The Service Interaction Requirements.  This may be specified at multiple levels and it assumed to default to the level above. If not specified in a ServiceDescription it defaults to GRA values (or false if not specified).

| | |
|---|---|
| **Name** | InteractionRequirements |
| **Abstract** | false |
| **Base Classifier** | |

#### Properties

#### ExtendedRequirement

ExtendedRequirement provides an arbitrary extension mechanism whereby InteractionRequirement may be notated on the containing element. The intent is that the phase-2 transform may then respect these flags and produce the desired result. Unknown extensions are ignored.

| | |
|---|---|
| **Type** | String |

| | |
|---|---|
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | composite |

### IdentityAndAttributeAssertionTransmission

True if the service has identity and attribute assertion transmissions, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### InterfaceDescriptionRequirements

True if the service has interface description requirements, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### Logging

True if the service has logging, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### MessageAddressing

True if the service has message addressing, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### MessageConfidentiality

True if the service has message confidentiality, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |

| | |
|---|---|
| **Composition** | composite |

### MessageIntegrity

True if the service has message integrity, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### MessageNonrepudiation

True if the service has message non-repudiation, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### Reliability

True if the service has reliability, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### ServiceAuthentication

True if the service has service authentications, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### ServiceConsumerAuthentication

True if the service has service consumer authentications, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### ServiceConsumerAuthorization

True if the service has service consumer authorizations, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### ServiceMetadataAvailability

True if the service has metadata availability, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### ServiceResponsiveness

True if the service has service responsiveness, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### TransactionSupport

True if the service has transaction support, false otherwise.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### 9.1.6 Class Interface

Represents an abstract service interface.  If not declared as an annotation, derived during phase-1 provisioning for every UML Interface in the service hierarchy.

| | |
|---|---|
| **Name** | Interface |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

## Properties

### [Annotations Only] MessageDefault

A set of default message specifications. If there is one MessageDefault with no realizations it applies to all parameters throughout the hierarchical scope of the Interface. Otherwise there may be any number of MessageDefaults, each realizing a type in the information model, in which case each one applies to all parameters that have that type within the scope of the Interface.

| | |
|---|---|
| **Type** | Message |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | none |

### Operation

The operations contained by the interface. Either explicitly declared or derived during phase-1 provisioning from the Operations/Receptions in the service hierarchy.

| | |
|---|---|
| **Type** | Operation |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### [Annotations Only] ParameterDefault

A set of default parameter specifications. If there is one ParameterDefault with no realizations it applies to all parameters of all operations and receptions of provided interfaces throughout the hierarchical scope of the Interface. Otherwise there may be any number of ParameterDefaults, each realizing a Parameter within the scope of the Interface, in which case each one applies to its realized Parameter.

| | |
|---|---|
| **Type** | Parameter |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | none |

## [Annotations Only] Constraints

### OperationsInHierarchy

Explicit Operation annotations must realize Operations or Receptions of the realized Interface.

```
context Interface inv:
    let realizedInterfaces : Set(UML::Interface) =
        self.oclAsType(InstanceSpecification).clientDependency
            ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
            ->select(x|x.oclIsKindOf(UML::Interface)).oclAsType(UML::Interface)
            ->asSet()
    in
        self.Operation.oclAsType(InstanceSpecification)->forAll(o|o.clientDependency-
    >select(x|x.oclIsKindOf(Realization))->collect(supplier)
      ->one(element | realizedInterfaces.feature->select(x|x.oclIsKindOf(BehavioralFeature))->includes(element)))
```

### ParameterDefaultsInHierarchy

ParameterDefault annotations may only realize Parameters in the hierarchy of the Component realized by the related Service.

```
context Interface inv:
    let realizedComponents : Set(UML::Component) =
      self.Port.Service.oclAsType(InstanceSpecification).clientDependency
      ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
      ->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
    let realizedComponentsHierarchy : Set(UML::Parameter) =
      realizedComponents.ownedPort.provided.feature-
    >select(x|x.oclIsKindOf(BehavioralFeature)).oclAsType(BehavioralFeature)->collect(ownedParameter)->asSet() in
    let  realizations : Set(Realization) = ParameterDefault.oclAsType(InstanceSpecification).clientDependency
      ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->asSet()
    in
       realizations->notEmpty() implies  realizations->one(supplier->one( element | realizedComponentsHierarchy-
    >includes(element)))
```

### RealizesInterface

An Interface annotation may only realize an Interface.

```
context Interface inv:
self.oclAsType(InstanceSpecification).clientDependency->exists(x|x.oclIsKindOf(Realization)) implies
   self.oclAsType(InstanceSpecification). clientDependency
   ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->collect(supplier)
   ->one(x|x.oclIsKindOf(UML::Interface))
```

## 9.1.7 Class Message

Represents the data of an operation parameter or a signal. If not declared as an annotation, derived during phase-1 provisioning for the data type of every UML Parameter in the service hierarchy. For each Message there must be a property of its corresponding type in the NIEM-UML model.

| | |
|---|---|
| **Name** | Message |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

### Properties

### [Intermediate Model Only] ElementModelReference

Reference to a model element annotated by this element. More specifically, it is the ModelReference associated with a message part element. This may be derived from the NIEM type by selecting a NIEM property (i.e., xml element) whose type is the message part type.

| | |
|---|---|
| **Type** | ModelReference |

| | |
|---|---|
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### [Intermediate Model Only] ElementName

The NIEM name used to reference Schema Name of an element reference.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

#### [Intermediate Model Only] ElementPrefix

Prefix used to reference Schema Namespace of an element reference. Derived from the defaultPrefix tag on the NIEM information model containing the element relating to this Message.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

#### [Intermediate Model Only] Prefix

Prefix used to reference Schema Namespace. Derived from the defaultPrefix tag on the NIEM information model containing the type relating to this Message.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

### [Annotations Only] Constraints

#### MustRealizeType

A Message must realize a Type.

**context** Message **inv:**
    self.oclAsType(InstanceSpecification).clientDependency->select(x|x.oclIsKindOf(Realization))->collect(supplier)->one(oclIsKindOf(Type))

## 9.1.8 [Intermediate Model Only] Class Model

A representation of the model resource associated with a ModelReference and generated as part of the generation of the ModelReference.

| | |
|---|---|
| **Name** | Model |
| **Abstract** | false |
| **Base Classifier** | |

### Properties

#### Label

The name of the model.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

#### ModelURI

URI for the model resource.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

## 9.1.9 [Intermediate Model Only] Class ModelReference

Model references are generated from Realizations and service hierarchy correspondences during phase-1.

| | |
|---|---|
| **Name** | ModelReference |
| **Abstract** | false |
| **Base Classifier** | |

### Properties

#### DiagramLink

URIs of diagrams associated with the corresponding element.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite |

#### Documentation

The documentation of the corresponding element.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### ElementID

The id of the corresponding element.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

#### Model

The model resource containing the corresponding element.

| | |
|---|---|
| **Type** | Model |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### Name

The name of the corresponding element.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## 9.1.10 Class Operation

A data structure representing a service action.  If not declared as an annotation, derived during phase-1 provisioning for every UML Operation in the service hierarchy.

| | |
|---|---|
| **Name** | Operation |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

### Properties

#### [Intermediate Model Only] ActionName

The name of the action. Derived from the corresponding element's name.

| | |
|---|---|
| **Type** | String |

| | |
|---|---|
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

### ActionProvenance

A description of provenance information for this action. Used to populate the Action Model section of SDD document.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### [Annotations Only] MessageDefault

A set of default message specifications. If there is one MessageDefault with no realizations it applies to all parameters throughout the hierarchical scope of the Operation. Otherwise there may be any number of MessageDefaults, each realizing a type in the information model, in which case each one applies to all parameters that have that type within the scope of the Operation.

| | |
|---|---|
| **Type** | Message |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | none |

### [Intermediate Model Only] ActionPurpose

A description of the purpose performed by this service action. Derived from the corresponding element's documentation.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### [Intermediate Model Only] IsAsynchronous

True if the operation is asynchronous, false otherwise. Value is based on the corresponding element being an operation or signal reception.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### MessageExchangePattern

Represents the GRA message exchange pattern associated with the operation.

| | |
|---|---|
| **Type** | ExchangePattern |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### Parameter

The parameters of an operation.  Either explicitly declared or derived during phase-1 provisioning from the parameters of Operations/Receptions in the service hierarchy.

| | |
|---|---|
| **Type** | Parameter |
| **Multiplicity** | 0..* |
| **Ordering** | ordered |
| **Composition** | composite [Intermediate Model Only] |

## [Annotations Only] Constraints

### RealizesOperationOrReception

An Operation annotation may only realize an Operation or Reception.

```
context Operation inv:
    self.oclAsType(InstanceSpecification).clientDependency->exists(oclIsKindOf(Realization)) implies
        self.oclAsType(InstanceSpecification).clientDependency-
    >select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->collect(supplier)
        ->one(oclIsKindOf(UML::Operation) or oclIsKindOf(Reception))
```

## 9.1.11 Class Organization

A data structure describing information about an organization.

| | |
|---|---|
| **Name** | Organization |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

## Properties

### OrganizationAcronym

An acronym for the organization.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### OrganizationFullAddressText

A physical address of an organization in full text form.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### OrganizationPointOfContact

A person designated as the point of contact for an organization.

| | |
|---|---|
| **Type** | Person |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### OrganizationRoleDescriptionText

An organization's role defined in free form text. Examples could be creator, provider, owner, maintainer, authority’s source, etc.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

### OrganizationRoleDetailedDescriptionText

A very detailed textual explanation of the role and responsibilities of an organization.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### OrganizationWebSiteURL

An internet address of the organization's web site.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### 9.1.12 Class Parameter

Represents the information content of parameters defined in an operation.  If not declared as an annotation, derived during phase-1 provisioning for every UML Parameter in the service hierarchy.

| | |
|---|---|
| **Name** | Parameter |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

#### Properties

##### Message

The information transferred in a parameter. Either explicitly declared or derived from the parameter's type.

| | |
|---|---|
| **Type** | Message |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite  [Intermediate Model Only] |

##### [Intermediate Model Only] Use

The "direction" of the message. Derived from the parameter's direction.

| | |
|---|---|
| **Type** | ParameterUse |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

#### [Annotations Only] Constraints

##### RealizesParameter

A Parameter annotation may only realize a Parameter.

```
context Parameter inv:
    self.oclAsType(InstanceSpecification).clientDependency->exists(oclIsKindOf(Realization)) implies
        self.oclAsType(InstanceSpecification).clientDependency-
    >select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->collect(supplier)
        ->one(oclIsKindOf(UML::Parameter))
```

### 9.1.13 Class Participant

Each Participant represents an ExchangePartner of the ServiceDescription. Participant annotations are generated by phase-1 provisioning.  A Participant may be modeled directly realizing an Actor in the logical model, in which case a single Participant corresponding to that Actor is generated. A Participant may be modeled realizing a Package in the logical model, in which case a Participant is generated corresponding to every Actor in the package.  A Participant is also generated for every Actor that plays a role in any Interaction owned by the Collaboration realized by the ServiceDescription. If Actors are related in the logical model by inheritance, then a Participant generated from a specializing Actor will contain a generalization element that identifies the Participant corresponding to the generalizing Actor.

| | |
|---|---|
| **Name** | Participant |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

## Properties

### [Intermediate Model Only] Generalization

Represents the fact that this Participant is a "kind of" the generalizing Participant. Derived from Actor generalization.

| | |
|---|---|
| **Type** | Participant |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite |

### ParticipatingOrganization

An organization acting as a participant.

| | |
|---|---|
| **Type** | Organization |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

## [Annotations Only] Constraints

### RealizesPackageOrActor

A Participant annotation may realize either an Actor or a Package.

```
context Participant inv:
    self.oclAsType(InstanceSpecification).clientDependency->exists(oclIsKindOf(Realization)) implies
        self.oclAsType(InstanceSpecification).clientDependency-
    >select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->collect(supplier)
        ->one(oclIsKindOf(Package) or oclIsKindOf(Actor))
```

## 9.1.14 Class Person

A data structure describing a person's name and the means to contact that person.

| | |
|---|---|
| **Name** | Person |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

### Properties

#### ContactPersonAddress

A physical address of a person in full text form.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### ContactPersonEmailID

An email address of a person.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

#### ContactPersonPhoneNumberID

A phone number of the person.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

## 9.1.15 Class Port

Represents a service port. If not declared as an annotation, derived during phase-1 provisioning for every UML Port in the service hierarchy.

| | |
|---|---|
| **Name** | Port |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

### Properties

#### AddressURI

The <soap:address> location of a <wsdl:port>. Derived by default in phase-2 from ServiceURI+"/"+"Name" but MAY be changed in the model.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |

|  |  |
|---|---|
| **Composition** | composite |

### Interface

Interfaces provided by the Port.  Either explicitly declared or derived during phase-1 provisioning from the interfaces of Ports in the service hierarchy.

|  |  |
|---|---|
| **Type** | Interface |
| **Multiplicity** | 0..* |
| **Ordering** |  |
| **Composition** | composite [Intermediate Model Only] |

### [Annotations Only] MessageDefault

A set of default message specifications. If there is one MessageDefault with no realizations it applies to all parameters throughout the hierarchical scope of the Port. Otherwise there may be any number of MessageDefaults, each realizing a type in the information model, in which case each one applies to all parameters that have that type within the scope of the Port.

|  |  |
|---|---|
| **Type** | Message |
| **Multiplicity** | * |
| **Ordering** |  |
| **Composition** | none |

### [Annotations Only] OperationDefault

A set of default operation specifications. If there is one OperationDefault with no realizations it applies to all operations and receptions of provided interfaces throughout the hierarchical scope of the Port. Otherwise there may be any number of OperationDefaults, each realizing an Operation or Reception within the scope of the Port, in which case each one applies to its realized element.

|  |  |
|---|---|
| **Type** | Operation |
| **Multiplicity** | * |
| **Ordering** |  |
| **Composition** | none |

### [Annotations Only] ParameterDefault

A set of default parameter specifications. If there is one ParameterDefault with no realizations it applies to all parameters of all operations and receptions of provided interfaces throughout the hierarchical scope of the Port. Otherwise there may be any number of ParameterDefaults, each realizing a Parameter within the scope of the Port, in which case each one applies to its realized Parameter.

|  |  |
|---|---|
| **Type** | Parameter |
| **Multiplicity** | * |
| **Ordering** |  |
| **Composition** | none |

## [Annotations Only] Constraints

### InterfacesInHierarchy

Explicit Interface annotations must realize provided Interfaces of the realized Port.

```
context Port inv:
    let realizedPorts : Set(UML::Port) = self.oclAsType(InstanceSpecification).clientDependency
        ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->collect(supplier)
        ->select(x|x.oclIsKindOf(UML::Port)).oclAsType(UML::Port)->asSet()
        in
            not(self.Interface.oclIsUndefined())  implies
            self.Interface.oclAsType(InstanceSpecification)
            ->forAll(clientDependency->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->collect(supplier)-
    >one(element | realizedPorts.provided->includes(element)))
```

### OperationDefaultsInHierarchy

OperationDefault annotations may only realize BehavioralFeatures in the hierarchy of the Component realized by the related Service.

```
context Port inv:
    let realizedComponents : Set(UML::Component) =
        self.Service.oclAsType(InstanceSpecification).clientDependency
        ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)
        ->collect(supplier)->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
    let realizedComponentsHierarchy : Set(BehavioralFeature) =
        realizedComponents.ownedPort.provided.feature->select(x|x.oclIsKindOf(BehavioralFeature))->asSet() in
    let  realizations : Set(Realization) = self.OperationDefault.oclAsType(InstanceSpecification).clientDependency
            ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->asSet() in
                realizations->notEmpty() implies  realizations->one(supplier->one( element |
    realizedComponentsHierarchy->includes(element)))
```

### ParameterDefaultsInHierarchy

ParameterDefault annotations may only realize Parameters in the hierarchy of the Component realized by the related Service.

```
context Port inv:
    let realizedComponents : Set(UML::Component) =
        self.Service.oclAsType(InstanceSpecification).clientDependency
        ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
        ->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
    let realizedComponentsHierarchy : Set(UML::Parameter) =
        realizedComponents.ownedPort.provided.feature-
    >select(x|x.oclIsKindOf(BehavioralFeature)).oclAsType(BehavioralFeature)->collect(ownedParameter)->asSet() in
    let  realizations : Set(Realization) = self.ParameterDefault.oclAsType(InstanceSpecification).clientDependency
        ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->asSet() in
            realizations->notEmpty() implies  realizations->one(supplier->one( element | realizedComponentsHierarchy-
    >includes(element)))
```

#### RealizesPort

A Port annotation may only realize a Port.

```
context Port inv:
self.oclAsType(InstanceSpecification).clientDependency->exists(oclIsKindOf(Realization)) implies
    self.oclAsType(InstanceSpecification).clientDependency-
>select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)
    ->collect(supplier)->one(oclIsKindOf(UML::Port))
```

## 9.1.16 Class SampleData

Represents the information associated with a service test: the input and expected output associated with an operation under test.

| | |
|---|---|
| **Name** | SampleData |
| **Abstract** | false |
| **Base Classifier** | |

### Properties

#### ExpectedOutput

The expected output of a service operation test.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

#### Input

The value of input parameters for a service operation test. Input must be in the same order as parameters.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | * |
| **Ordering** | ordered |
| **Composition** | composite |

## 9.1.17 [Intermediate Model Only] Class SchemaReference

Elements describing a schema used by this service interface. Derived from those parts of the information model that are referenced from the service interface.

| | |
|---|---|
| **Name** | SchemaReference |
| **Abstract** | false |
| **Base Classifier** | |

### Properties

#### Namespace

Namespace of schema.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

#### Prefix

Prefix used to reference Schema Namespace.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

#### SchemaLocation

URI of Schema, relative to SSP Catalog.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

## 9.1.18 Class SecurityClassification

Any applicable classification of the security level of the information exchanged by the service, such as SBU, Secret, etc. If there is no strict classification this field can contain a brief statement regarding the security of the data.

| | |
|---|---|
| **Name** | SecurityClassification |
| **Abstract** | false |
| **Base Classifier** | |

### Properties

#### [Intermediate Model Only] Name

The name, derived from the instance name.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### 9.1.19 Class Service

Describes a service. To be successfully processed by phase-1 provisioning a Service must realize a UML Component.

| | |
|---|---|
| **Name** | Service |
| **Abstract** | false |
| **Base Classifier** | |

## Properties

### [Annotations Only] InterfaceDefault

A set of default interface specifications. If there is one InterfaceDefault with no realizations it applies to all provided interfaces throughout the hierarchical scope of the Service. Otherwise there may be any number of InterfaceDefaults, each realizing an Interface within the scope of the Service, in which case each one applies to its realized Interface.

| | |
|---|---|
| **Type** | Interface |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | none |

### [Annotations Only] MessageDefault

A set of default message specifications. If there is one MessageDefault with no realizations it applies to all parameters throughout the hierarchical scope of the Service. Otherwise there may be any number of MessageDefaults, each realizing a type in the information model, in which case each one applies to all parameters that have that type within the scope of the Service.

| | |
|---|---|
| **Type** | Message |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | none |

### [Annotations Only] OperationDefault

A set of default operation specifications. If there is one OperationDefault with no realizations it applies to all operations and receptions of provided interfaces throughout the hierarchical scope of the Service. Otherwise there may be any number of OperationDefaults, each realizing an Operation or Reception within the scope of the Service, in which case each one applies to its realized element.

| | |
|---|---|
| **Type** | Operation |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | none |

### [Annotations Only] ParameterDefault

A set of default parameter specifications. If there is one ParameterDefault with no realizations it applies to all parameters of all operations and receptions of provided interfaces throughout the hierarchical scope of the Service. Otherwise there may be any number of ParameterDefaults, each realizing a Parameter within the scope of the Service, in which case each one applies to its realized Parameter.

| | |
|---|---|
| **Type** | Parameter |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | none |

### Port

The port specification for the ports owned by the service component realizing this Service.  Either explicitly declared or derived from Ports on the realized Component.

| | |
|---|---|
| **Type** | Port |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### SampleData

Sample data associated with service testing.

| | |
|---|---|
| **Type** | SampleData |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### [Intermediate Model Only] ServiceProvider

The provider of the service. Derived from a Realization between the service's realized Component and the participant's realized Actor.

| | |
|---|---|
| **Type** | Participant |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## [Annotations Only] Constraints

### InterfaceDefaultsInHierarchy

InterfaceDefault annotations may only realize Interfaces provided by ownedPorts of the realized Component.

**context** Service **inv:**

```
let realizedComponents : Set(UML::Component) =
  self.oclAsType(InstanceSpecification).clientDependency
  ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
  ->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
let realizedComponentsHierarchy : Set(UML::Interface) =
   realizedComponents.ownedPort.provided->asSet() in
let  realizations : Set(Realization) = self.InterfaceDefault.oclAsType(InstanceSpecification).clientDependency
   ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->asSet()
in
     realizations->notEmpty() implies  realizations->one(supplier->one( element | realizedComponentsHierarchy-
>includes(element)))
```

## MustRealizeComponent

A Service must realize one Component.

**context** Service **inv:**

```
self.oclAsType(InstanceSpecification).clientDependency
   ->select(x|x.oclIsKindOf(Realization)).supplier
   ->select(x|x.oclIsKindOf(UML::Component))->size()=1
```

## OperationDefaultsInHierarchy

OperationDefault annotations may only realize Operations or Receptions of Interfaces of ownedPorts of the realized Component.

**context** Service **inv:**

```
let realizedComponents : Set(UML::Component) =
   self.oclAsType(InstanceSpecification).clientDependency
   ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
   ->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
let realizedComponentsHierarchy : Set(BehavioralFeature) =
   realizedComponents.ownedPort.provided.feature->select(x|x.oclIsKindOf(BehavioralFeature))->asSet() in
let  realizations : Set(Realization) = self.OperationDefault.oclAsType(InstanceSpecification).clientDependency
   ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->asSet()
in
    realizations->notEmpty() implies  realizations->one(supplier->one( element | realizedComponentsHierarchy-
>includes(element)))
```

## ParameterDefaultsInHierarchy

ParameterDefault annotations may only realize Parameters in the hierarchy of the realized Component.

**context** Service **inv:**

```
let realizedComponents : Set(UML::Component) =
   self.oclAsType(InstanceSpecification).clientDependency
   ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
   ->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
let realizedComponentsHierarchy : Set(UML::Parameter) =
   realizedComponents.ownedPort.provided.feature-
>select(x|x.oclIsKindOf(BehavioralFeature)).oclAsType(BehavioralFeature)->collect(ownedParameter)->asSet() in
let  realizations : Set(Realization) = self.ParameterDefault.oclAsType(InstanceSpecification).clientDependency
   ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->asSet()
```

```
in
    realizations->notEmpty() implies  realizations->one(supplier->one( element | realizedComponentsHierarchy->includes(element)))
```

### PortsInHierarchy

Explicit Port annotations must realize ownedPorts of the realized Component.

**context** Service **inv:**

```
let realizedComponents : Set(UML::Component) =
   self.oclAsType(InstanceSpecification).clientDependency
   ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
   ->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
   not(Port.oclIsUndefined()) implies
   self.Port.oclAsType(InstanceSpecification)
   ->forAll(clientDependency->select(x|x.oclIsKindOf(Realization))->collect(supplier)->one(element | realizedComponents.ownedPort->includes(element)))
```

## 9.1.20 Class ServiceCapability

A free text format description of the capability provided by a service. Per GRA, a capability represents the provider's view of a service.

| | |
|---|---|
| **Name** | ServiceCapability |
| **Abstract** | false |
| **Base Classifier** | |

### Properties

#### [Intermediate Model Only] Documentation

A free text format description of the capability provided by a service.   Derived from the instance documentation.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## 9.1.21 Class ServiceDescription

A data structure representing the details describing a service. To be successfully processed by phase-1 provisioning, a ServiceDescription must realize a UML Collaboration and must use at least one IEPD.

| | |
|---|---|
| **Name** | ServiceDescription |
| **Abstract** | false |
| **Base Classifier** | ServiceIdentification |

## Properties

### ActivationDate

A date when a service was or will be first available in production. Not to be confused with the date this service was submitted to a registry. The format is YYYY-MM.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### AdditionalInformation

This property contains any additional information pertinent to the service which should be included in this description but does not belong elsewhere.  This could be information about future capabilities the service could provide, information regarding specific conditions which govern the use of the service, information regarding specific domain capabilities the service fulfills, etc.  If required, subsections can be created to further structure the information provided in this section.

Additional artifacts related to this section's content can be provided in the artifacts folder of the service package.

[Service Abbreviation] SSP [Service Version]\artifacts

If such artifacts are provided, they should be referenced here.  A description of the artifact and a link to it should be provided as part of the reference.

This property is used to populate the Additional Information section of the SDD document.

| | |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..* |
| **Ordering** | ordered |
| **Composition** | composite [Intermediate Model Only] |

### AlertAndNotificationURI

A URL to sign up for alerts and notifications for a specific service.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### Classification

A collection of classifications defining the relationship between a service and an applicable enterprise architecture and business reference model.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | * |
| **Ordering** | ordered |

| | |
|---|---|
| **Composition** | composite |

### CreationDate

A date designation when a service was first created. Not to be confused with the date a service is submitted to a registry. The format is YYYY-MM. Set by default to today by Phase-2.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### DataProvenance

Provenance is defined as the agency, office, or person of origin of records, i.e., the entity that created, received, or accumulated and used the records in the conduct of their business activities. Any applicable provenance information or restrictions should be provided in this section.

| | |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### DomainDescription

A primary domain or line of business (LoB) that a service covers.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### Endorsements

A collection of names and acronyms of professional or governmental organizations or people that endorse this service as an official business exchange.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..* |
| **Ordering** | ordered |
| **Composition** | composite |

### ExchangePartner

The Participants in a service. May be derived in Phase-1 - see 9.1.13 Participant.

| | |
|---|---|
| **Type** | Participant |
| **Multiplicity** | 0..* |

| | |
|---|---|
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### ExecutionContext

Service descriptions should include all information pertinent to the production or consumption of the service, including expected infrastructure functions and other dependencies. No information directly pertaining to implementation platform or technology should be included in the service description. Conversely, platform capabilities which are technology-independent should be included. For example, stating in the service description that services that are encrypted are being provided by the infrastructure is preferred compared to stating that the Public Key Infrastructure (PKI) infrastructure is expected. It is expected that the services defined using this document will minimize the dependence on specific technical infrastructure to provide the greatest flexibility and interoperability for service providers and service consumers.

It is also important to note that the information in this property will be applicable to more than one service, and these required capabilities will be provisioned as part of the infrastructure layer of the architecture. For instance, if information is to be exchanged securely within the execution context, enabling this functionality at the infrastructure level and not per a specific SSP or SIP is the strongly preferred direction for enabling the GRA. In other words, these commonly used technical functions would be most effectively achieved by an infrastructure solution which supports the GRA.

| | |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### ExpirationDate

A year and month (YYYY-MM) this service is expected to be no longer available (if applicable).

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### [Intermediate Model Only] IEPDReference

Reference information identifying an Information Exchange Package Document which the service uses in its data model. Derived from Usage dependencies to IEPD.

| | |
|---|---|
| **Type** | IEPDReference |
| **Multiplicity** | 1..* |
| **Ordering** | |
| **Composition** | composite |

### LastRevisionDate

A date with the year and month specifying when this service information was last revised. The format is YYYY-MM. Set by default by Phase-2.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### LifecycleStatus

A word indicating the current stage of the service within the development lifecycle.  Valid values are: In Design, In Development, Release Candidate, Operational/Production, and Deprecated.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### MajorVersion

A value identifying the primary version number. Defaults to 1.

| | |
|---|---|
| **Type** | Integer |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### MinorVersion

A value designating a minor version number. Defaults to 0.

| | |
|---|---|
| **Type** | Integer |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### NextRevisionDate

A year and month (YYYY-MM) a service is expected to be revised.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### OtherRequirement

This section lists any other requirements which have to be met and on which the service depends to deliver its capabilities.

Additional artifacts related to specific subsections of this section's content can be provided under the artifacts folder of the service package. These artifacts are included as documents or folders under the various subfolders of the artifacts folder.

| | |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### Privacy

While service descriptions are technology-agnostic and do not detail the physical model of a service, certain privacy requirements are applicable to the service and need to be carried through all its implementations.  This property outlines those requirements.

Value is used to populate the Privacy section of the SDD Document.

| | |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### ProcessModel

Process models describe processing rules and fault handling and should provide a step-by-step description of the logic carried out by the service actions including any pre- or post-conditions.  This should be supplemented with algorithms, workflow diagrams, or even entire business process definitions.

The process model could be described utilizing a variety of business process modeling notations and languages which are further described in business process modeling profiles.

| | |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### RealWorldEffect

A collection of the Real World Effects caused by the Service.

| | |
|---|---|
| **Type** | UseCase |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### RelatedOrganization

A collection of organizations that are somehow related to the service.

|  |  |
|---|---|
| **Type** | Organization |
| **Multiplicity** | 0..* |
| **Ordering** |  |
| **Composition** | composite [Intermediate Model Only] |

### RevisionVersion

A value designating a minor version revision number.

|  |  |
|---|---|
| **Type** | Integer |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | composite |

### Security

While service descriptions are technology-agnostic and do not detail the physical model of a service, certain security requirements are applicable to the service and need to be carried through all its implementations. This property outlines those requirements.

The value is used to populate the Security section of the SDD document.

|  |  |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | composite [Intermediate Model Only] |

### ServiceAssumption

This property lists all assumptions on which the service depends to deliver its real-world effects.

|  |  |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | composite [Intermediate Model Only] |

### ServiceCapability

An enumeration of the capabilities provided by a service.

|  |  |
|---|---|
| **Type** | ServiceCapability |
| **Multiplicity** | 0..* |
| **Ordering** | ordered |
| **Composition** | composite [Intermediate Model Only] |

## ServiceDependency

Services upon which a service directly depends to deliver its real world effects.

| | |
|---|---|
| **Type** | ServiceIdentification |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

## ServiceDescriptionKeywordText

Search terms that would not otherwise be in other metadata attributes (e.g., Child Support Warrant, Domestic Relations Warrant, Domestics).

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..* |
| **Ordering** | ordered |
| **Composition** | composite |

## ServiceDescriptionSummaryText

A brief summary of this service for short display purposes.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## [Intermediate Model Only] ServiceInteraction

The exchanges between participants described by the service. Maps to the GRA business scenarios. Derived from the Interactions owned by the realized Collaboration.

| | |
|---|---|
| **Type** | ServiceInteraction |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | composite |

## ServiceInterface

A set of service interface specifications.

| | |
|---|---|
| **Type** | ServiceInterfaceSpecification |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### ServiceLevelAgreement

A set of service level agreements.

| | |
|---|---|
| **Type** | ServiceLevelAgreement |
| **Multiplicity** | 0..* |
| **Ordering** | ordered |
| **Composition** | composite [Intermediate Model Only] |

### ServicePurpose

A purpose which the service intends or resolves to perform or accomplish.

| | |
|---|---|
| **Type** | Description |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### ServiceScopeDescription

A description of the scope of a service.

| | |
|---|---|
| **Type** | Description |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### ServiceSecurityClassification

The service's security classification.

| | |
|---|---|
| **Type** | SecurityClassification |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### Sponsor

A collection of professional or governmental organization(s) or person that sponsored, contributed, or participated in the development of a service.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | * |
| **Ordering** | ordered |
| **Composition** | composite |

### TransformationURI

URI of a template file structure that includes artifacts to transform the intermediate SSP into a final one as the "phase-2" transform.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## [Annotations Only] Constraints

### MustRealizeCollaboration

A ServiceDescription must realize a Collaboration.

**context** ServiceDescription **inv:**

```
    self.oclAsType(InstanceSpecification).clientDependency
        ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->collect(supplier)
        ->one(oclIsKindOf(UML::Collaboration))
```

### MustUseIEPD

A ServiceDescription must use one or more IEPDs.

**context** ServiceDescription **inv:**

```
    self.oclAsType(InstanceSpecification).clientDependency
    ->exists(oclIsKindOf(Usage)) and self.oclAsType(InstanceSpecification).clientDependency-
    >select(x|x.oclIsKindOf(Usage))->collect(supplier)->forAll(oclIsKindOf(Type))
```

## 9.1.22 Class ServiceIdentification

A data structure representing the means of uniquely identifying a service.

| | |
|---|---|
| **Name** | ServiceIdentification |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

## Properties

### ServiceID

An identification of the service in a service registry and/or repository.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### ServiceNameAbbreviationText

A human readable abbreviation of the Service Name.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

### ServiceURI

A fully qualified locator of the service interface potentially including version and environment.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

## 9.1.23 [Intermediate Model Only] Class ServiceInteraction

A ServiceInteraction represents an exchange between participants. ServiceInteractions are generated by Phase-1 processing for each Interaction owned by the Collaboration realized by the ServiceDescription, with the Participants property derived from the Actors that play a role in the Interaction.

| | |
|---|---|
| **Name** | ServiceInteraction |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

### Properties

### Participant

Derived from the Actors that play a role in, or are realized by Components that play a role in, the Interaction corresponding to the ServiceInteraction.

| | |
|---|---|
| **Type** | Participant |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite |

## 9.1.24 Class ServiceInteractionProfile

A data structure containing information about a Service Interaction Profile.

| | |
|---|---|
| **Name** | ServiceInteractionProfile |
| **Abstract** | false |
| **Base Classifier** | |

### Properties

#### SIPName

The name of the Service Interaction Profile.
[Derived from] SIP Name

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

#### SIPVersion

Version of the Service Interaction Profile.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

## 9.1.25 Class ServiceInterfaceSpecification

A data type representing details relating to a service interface.

| | |
|---|---|
| **Name** | ServiceInterfaceSpecification |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

### Properties

#### [Annotations Only] InterfaceDefault

A set of default interface specifications. If there is one InterfaceDefault with no realizations it applies to all provided interfaces throughout the hierarchical scope of the ServiceInterfaceSpecification. Otherwise there may be any number of InterfaceDefaults, each realizing an Interface within the scope of the ServiceInterfaceSpecification, in which case each one applies to its realized Interface.

| | |
|---|---|
| **Type** | Interface |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | none |

#### [Annotations Only] MessageDefault

A set of default message specifications. If there is one MessageDefault with no realizations it applies to all parameters throughout the hierarchical scope of the ServiceInterfaceSpecification. Otherwise there may be any number of MessageDefaults, each realizing a type in the information model, in which case each one applies to all parameters that have that type within the scope of the ServiceInterfaceSpecification.

|  |  |
|---|---|
| **Type** | Message |
| **Multiplicity** | * |
| **Ordering** |  |
| **Composition** | none |

### MessageDefinitionMechanism

This property includes information about the message definition mechanism utilized by the service actions.

Per the GRA, message definition mechanisms are closely related to the interface description requirements. Unlike interface description requirements, message definition mechanisms establish a standard way of defining the structure and contents of a message.

Additional artifacts related to this section's content can be provided in the artifacts folder of the service package.

[Service Abbreviation] SSP [Service Version]\artifacts

If such artifacts are provided, they should be referenced here. A description of the artifact and a link to it should be provided as part of the reference.

|  |  |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..* |
| **Ordering** | ordered |
| **Composition** | composite [Intermediate Model Only] |

### [Annotations Only] OperationDefault

A set of default operation specifications. If there is one OperationDefault with no realizations it applies to all operations and receptions of provided interfaces throughout the hierarchical scope of the ServiceInterfaceSpecification. Otherwise there may be any number of OperationDefaults, each realizing an Operation or Reception within the scope of the ServiceInterfaceSpecification, in which case each one applies to its realized element.

|  |  |
|---|---|
| **Type** | Operation |
| **Multiplicity** | * |
| **Ordering** |  |
| **Composition** | none |

### [Annotations Only] ParameterDefault

A set of default parameter specifications. If there is one ParameterDefault with no realizations it applies to all parameters of all operations and receptions of provided interfaces throughout the hierarchical scope of the ServiceInterfaceSpecification. Otherwise there may be any number of ParameterDefaults, each realizing a Parameter within the scope of the ServiceInterfaceSpecification, in which case each one applies to its realized Parameter.

|  |  |
|---|---|
| **Type** | Parameter |
| **Multiplicity** | * |
| **Ordering** |  |
| **Composition** | none |

## PhysicalModel

The physical model should sufficiently describe the set of actions implemented by the service interface and the physical endpoint(s) for accessing these actions. This section will also include any relevant details of the Service Interaction Profile (SIP) that will govern how the service interaction requirements of the service will be met. The physical model described in this document will also provide details regarding the message schema(s) for the information model of the service.

The above information can be made part of this document or included by reference

| | |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

## [Annotations Only] PortDefault

A set of default port specifications. If there is one PortDefault with no realizations it applies to all ports throughout the hierarchical scope of the ServiceInterfaceSpecification. Otherwise there may be any number of PortDefaults, each realizing a Port within the scope of the ServiceInterfaceSpecification, in which case each one applies to its realized Port.

| | |
|---|---|
| **Type** | Port |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | none |

## [Intermediate Model Only] Prefix

Prefix to be used for target namespace of (WSDL) Service Specification. Derived from (the same as) ServiceInterfaceNameAbbeviationText.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

## [Intermediate Model Only] SchemaReference

References to used Schema. Derived from those parts of the information model that are referenced from the service interface.

| | |
|---|---|
| **Type** | SchemaReference |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | composite |

### SecurityDescriptionText

A text description that identifies the security which was implemented to protect this service interface (GFIPM, Trusted Broker, etc)

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### SecurityImplementedIndicator

True when security has been implemented to access this service; otherwise false.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### Service

The services of the service interface. Note that in GRA it is best practice to have exactly one service per service interface specification.

| | |
|---|---|
| **Type** | Service |
| **Multiplicity** | 1..* |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### ServiceInteractionProfile

Information about the implemented Service Interaction Profile.

| | |
|---|---|
| **Type** | ServiceInteractionProfile |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### ServiceInterfaceNameAbbreviationText

A human readable abbreviation of the Service Name that is also appropriate as a WSDL prefix.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

### ServiceTesting

Service Providers may deploy testing facilities and specific testing environments for their services. Use of these testing facilities and environments may be required or optional. As consumers implement service interfaces, there will be a need to test those implementations. Service providers should document in this section testing options, testing prerequisites, test endpoints, environmental requirements, test schedules, and control procedures and sample data (inputs and expected outputs) for each supported action.

| | |
|---|---|
| **Type** | Description |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### TargetNamespace

Target namespace of the (WSDL) Service Interface.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | |
| **Composition** | composite |

## [Annotations Only] Constraints

### InterfaceDefaultsInHierarchy

InterfaceDefault annotations may only realize Interfaces in the hierarchy of the realized Component of a related Service.

```
context ServiceInterfaceSpecification inv:
    let realizedComponents : Set(UML::Component) =
       self.Service.oclAsType(InstanceSpecification).clientDependency
       ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
       ->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
    let realizedComponentsHierarchy : Set(UML::Interface) =
       realizedComponents.ownedPort.provided->asSet() in
    let  interfaceDefaultRealizations : Set(NamedElement) =
       self.InterfaceDefault.oclAsType(InstanceSpecification).clientDependency-
    >select(x|x.oclIsKindOf(Realization)).oclAsType(Realization).supplier->asSet()
    in
        interfaceDefaultRealizations->forAll(element | realizedComponentsHierarchy->includes(element))
```

### OperationDefaultsInHierarchy

OperationDefault annotations may only realize BehavioralFeatures in the hierarchy of the realized Component of a related Service.

```
context ServiceInterfaceSpecification inv:
    let realizedComponents : Set(UML::Component) =
       self.Service.oclAsType(InstanceSpecification).clientDependency
       ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
       ->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
```

```
let realizedComponentsHierarchy : Set(BehavioralFeature) =
   realizedComponents.ownedPort.provided.feature->select(x|x.oclIsKindOf(BehavioralFeature))->asSet() in
let  realizations : Set(Realization) = self.OperationDefault.oclAsType(InstanceSpecification).clientDependency
   ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->asSet() in
     realizations->notEmpty() implies  realizations->one(supplier->one( element | realizedComponentsHierarchy-
>includes(element)))
```

### ParameterDefaultsInHierarchy

ParameterDefault annotations may only realize Parameters in the hierarchy of the realized Component of a related Service.

```
context ServiceInterfaceSpecification inv:
     let realizedComponents : Set(UML::Component) =
        self.Service.oclAsType(InstanceSpecification).clientDependency
        ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
        ->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
     let realizedComponentsHierarchy : Set(UML::Parameter) =
        realizedComponents.ownedPort.provided.feature-
     >select(x|x.oclIsKindOf(BehavioralFeature)).oclAsType(BehavioralFeature)->collect(ownedParameter)->asSet() in
     let  realizations : Set(Realization) = self.OperationDefault.oclAsType(InstanceSpecification).clientDependency
        ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->asSet()
     in
          realizations->notEmpty() implies  realizations->one(supplier->one( element | realizedComponentsHierarchy-
     >includes(element)))
```

### PortDefaultsInHierarchy

PortDefault annotations may only realize Ports in the hierarchy of the realized Component of a related Service.

```
context ServiceInterfaceSpecification inv:
     let realizedComponents : Set(UML::Component) =
        self.Service.oclAsType(InstanceSpecification).clientDependency
        ->select(x|x.oclIsKindOf(Realization))->collect(supplier)
        ->select(x|x.oclIsKindOf(UML::Component)).oclAsType(UML::Component)->asSet() in
     let realizedComponentsHierarchy : Set(UML::Port) = realizedComponents.ownedPort->asSet() in
     let  realizations : Set(Realization) = self.PortDefault.oclAsType(InstanceSpecification).clientDependency
        ->select(x|x.oclIsKindOf(Realization)).oclAsType(Realization)->asSet()
     in
          realizations->notEmpty() implies  realizations->one(supplier->one( element | realizedComponentsHierarchy-
     >includes(element)))
```

## 9.1.26 Class ServiceLevelAgreement

A collection of policies, agreements, licensing and any other governance or performance documentation specifying constraints and any other details regarding the realization of a service.

| | |
|---|---|
| **Name** | ServiceLevelAgreement |
| **Abstract** | false |
| **Base Classifier** | |

## Properties

### Agreement

Applicable Agreements or MOUs governing the use, administration, or implementation of a service.

| | |
|---|---|
| **Type** | Agreement |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### [Intermediate Model Only] ApplicableAgreementsIndicator

True when there are any applicable agreements or Memoranda Of Understanding (MOUs) relating to the use, administration, or implementation of a service; otherwise false.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### ApplicableContract

A set of formal contract associated with a service, application, process, transaction, or thing, tangible or otherwise.

| | |
|---|---|
| **Type** | Agreement |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### [Intermediate Model Only] ApplicableContractsIndicator

True when there are any applicable contracts relating to the use, administration, or implementation of a service; otherwise false.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### [Intermediate Model Only] ApplicablePoliciesIndicator

True when there are any applicable policies governing the use, administration, or implementation of a service; otherwise false.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |

|  |  |
|---|---|
| **Ordering** | |
| **Composition** | composite |

### ApplicablePolicy

A collection of all policies that in some way constrain, govern, or control the usage of a service, application, process, etc.

|  |  |
|---|---|
| **Type** | Agreement |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### [Intermediate Model Only] ApplicableUmbrellaAgreementsIndicator

True when there are any applicable umbrella agreements relating to the use, administration, or implementation of a service; otherwise false.

|  |  |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### ApprovalRequiredIndicator

True when a permission must first be obtained prior to using a service or performing some action in a business process; otherwise false.

|  |  |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### CreationCostAmount

Cost to create a thing, such as an application or service. This includes the full cost to design, manage, develop, test, implement and maintain. Currency text may precede or follow amount.

|  |  |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### LicensingAgreement

Agreement licensing a service or application. Descriptive values could be In House, No License, Open Source, Purchase License, etc.

|  |  |
|---|---|
| **Type** | Agreement |

| | |
|---|---|
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### LicensingRequiredIndicator

True when a license is required to use a service or application; otherwise false.

| | |
|---|---|
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### ServiceAvailability

A description or measurement of the expected availability that a service is usable.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### ServiceAverageThroughput

A description of how often a service is expected to be, or actually used, averaged over a period of time.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### ServiceMaximumThroughput

A description of the limit of how often a service is able to be accessed or used at, over a period of time during peak capacity.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### ServiceResponseTime

A description of the average response time for a service.  The response time is calculated as the time input is provided to the service until the service completes its process or provides output for the consumer.

| | |
|---|---|
| **Type** | String |

| | |
|---|---|
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### UmbrellaAgreement

A set of umbrella agreements that in some way constrain, govern, or control the usage of a service, application, process etc.

| | |
|---|---|
| **Type** | Agreement |
| **Multiplicity** | 0..* |
| **Ordering** | |
| **Composition** | composite [Intermediate Model Only] |

### UsageCostAmount

A total cost to use something, such as a service, etc. Currency text may precede or follow amount.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### UsageUnitCostAmount

A cost associated with a service (e.g. transaction, unlimited transactions, minutes of use). Currency text may precede or follow amount.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

## 9.1.27 Class UseCase

A UseCase annotation represents one or more "real world effects" of a ServiceDescription.  In the annotation model, a UseCase instance may realize a UseCase or a Package in the logical model.  In the former case, Phase-1 will generate a single UseCase instance. In the latter case, Phase-1 will generate a UseCase instance for each UML UseCase contained in the realized Package.  In all cases the generated UseCases' Provider and Consumer properties are derived from Associations stereotyped «Provider» and «Consumer» between the logical UseCases and the logical Actors that correspond to Participants in the generated annotation model.

| | |
|---|---|
| **Name** | UseCase |
| **Abstract** | false |
| **Base Classifier** | GRAServiceAnnotationBase |

## Properties

### [Intermediate Model Only] Consumer

Participants that consume services. Derived from associations stereotyped Consumer.

| | |
|---|---|
| **Type** | Participant |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | composite |

### [Intermediate Model Only] Provider

Participants that provide services. Derived from associations stereotyped Provider.

| | |
|---|---|
| **Type** | Participant |
| **Multiplicity** | * |
| **Ordering** | |
| **Composition** | composite |

## [Annotations Only] Constraints

### RealizesPackageOrUseCase

A UseCase annotation may realize either a UseCase or a Package.

**context** UseCase **inv:**
```
    self.oclAsType(InstanceSpecification).clientDependency->exists(oclIsKindOf(Realization)) implies
      self.oclAsType(InstanceSpecification).clientDependency->select(x|x.oclIsKindOf(Realization))->collect(supplier)-
    >one(oclIsKindOf(Package) or oclIsKindOf(UML::UseCase))
```

## 9.1.28 Enumeration ExchangePattern

Represents a GRA Exchange Pattern.  This type should define the message exchange patterns leveraged by the service actions.  The GRA recognizes the following message exchange patterns:

- The FIRE-AND-FORGET pattern calls for the sender of a message (which could be the service consumer or service) to send the message and not expect a reply message back from the recipient. This pattern is useful for one-way transmission of information, such as notification that an event has occurred.
- The REQUEST-REPLY pattern calls for the sender of a message to send the message and expect a reply from the recipient.

These two patterns are considered “primitive” patterns, in that they are the fundamental building blocks of more complex information exchange scenarios. For instance, the complex PUBLISH-SUBSCRIBE pattern involves an initial request-reply exchange in which the subscriber subscribes to a service, followed by the service using the fire-and-forget pattern to notify subscribers of an event.

Within the service interface description, the behavioral model should describe message exchanges in terms of these two primitive exchange patterns. Each action can specify a different message exchange pattern.

### Literals

**unknown**
: This is a temporary workaround for some EMF issues.

**enquiry**
: A request for information with the subsequent return of that information, if available and authorized. A request-reply.

**subscription**
: A request for changes in information with the later notification of such changes, if available and authorized. A request-reply.

**notification**
: Notification of an event of change or information. A fire-and-forget.

**update**
: A request to change some information with a subsequent return of success of failure. Related information may also be returned. A request-reply.

**message**
: A message as a result of a subscription. A fire-and-forget.

### 9.1.29 Enumeration ParameterUse

Defines the direction of a message associated with the parameters and exceptions of an operation.

### Literals

**unknown**
: This is a temporary workaround for some EMF issues.

**in**
: Specifies that the corresponding operation parameter has an in direction.

**out**
: Specifies that the corresponding operation parameter has an out direction.

**inout**
: Specifies that the corresponding operation parameter has an in and out direction.

**exception**
: Specifies that the usage for the message is as a fault, or exception, on the operation.

## 9.2 Package GRAUML::GRAWsdl

This package, shown in Figure 16, is identical in the Annotations and Intermediate models. Classes from this package are only intended to be used when explicit non-default values are required for the generated WSDL.

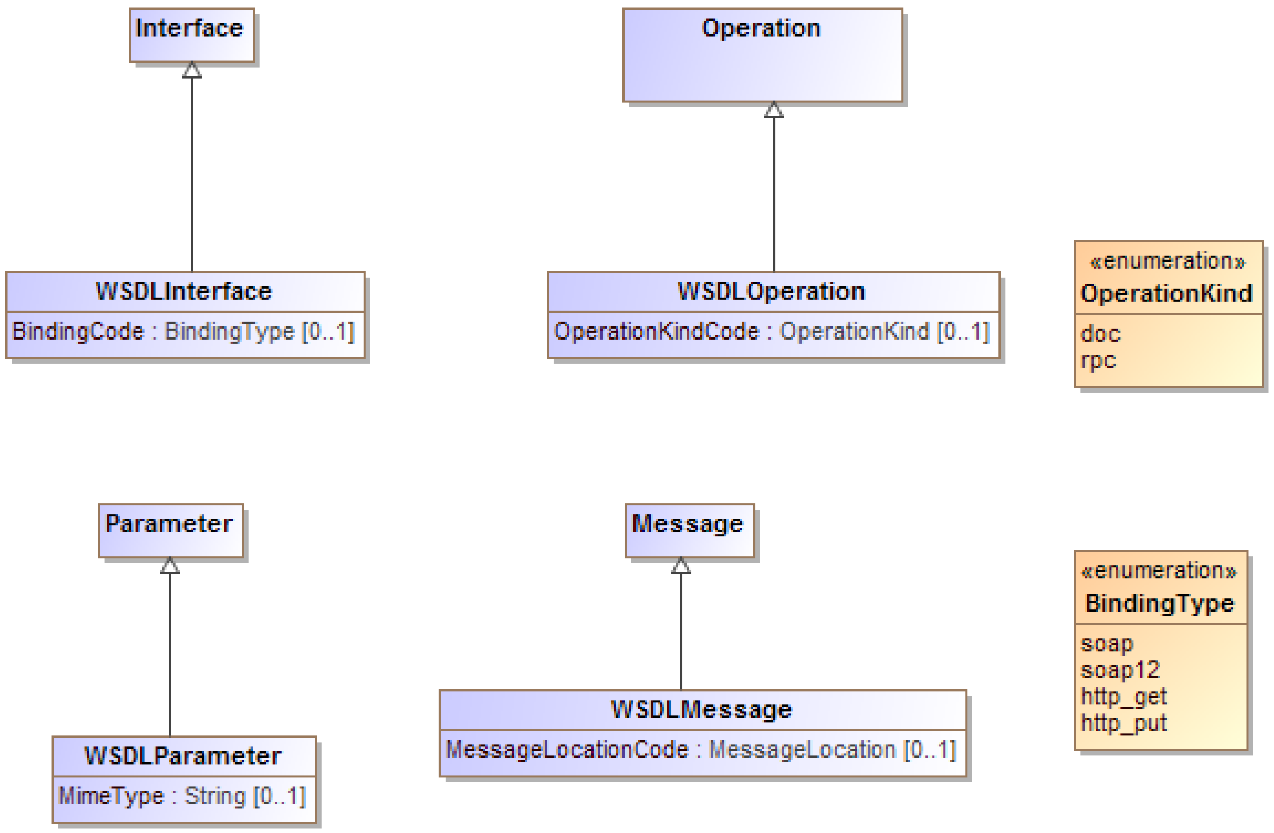


**Figure 16 WSDL annotations**

### 9.2.1 Class WSDLInterface

WSDL Extension of the technology-independent element. Represents a <wsdl:portType>.

| | |
|---|---|
| **Name** | WSDLInterface |
| **Abstract** | false |
| **Base Classifier** | Interface |

### Properties

#### BindingCode

Specifies technology implementation for a <wsdl:binding>: soap, soap12, http_get, http_put.

| | |
|---|---|
| **Type** | BindingType |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### 9.2.2 Class WSDLMessage

WSDL Extension of the technology-independent element. Represents a <wsdl:message>, its contained <part>, and its usage from a <wsdl:input>, <wsdl:output>, or a <wsdl:fault> within a <wsdl:portType> <wsdl:operation> .

| | |
|---|---|
| **Name** | WSDLMessage |
| **Abstract** | false |
| **Base Classifier** | Message |

#### Properties

##### MessageLocationCode

When used in conjunction with a soap binding, indicates the location of the part within the message: body, header, url.

| | |
|---|---|
| **Type** | MessageLocation |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### 9.2.3 Class WSDLOperation

WSDL Extension of the technology-independent element. Represents a <wsdl:Operation> within a <wsdl:binding> or <wsdl:portType>.

| | |
|---|---|
| **Name** | WSDLOperation |
| **Abstract** | false |
| **Base Classifier** | Operation |

#### Properties

##### OperationKindCode

Represents the style {doc, rpc} of a <soap:operation> within a <wsdl:binding><wsdl:operation> which has a <soap:binding>.  Doc is recommended by GRA.

| | |
|---|---|
| **Type** | OperationKind |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### 9.2.4 Class WSDLParameter

WSDL Extension of the technology-independent element.

| | |
|---|---|
| **Name** | WSDLParameter |
| **Abstract** | false |
| **Base Classifier** | Parameter |

### Properties

#### MimeType

Represents the <mime:content> type within a <wsdl:binding> <wsdl:operation> parameter.

| | |
|---|---|
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** | |
| **Composition** | composite |

### 9.2.5 Enumeration BindingType

Binding type as per WSD specification.

### Literals

**soap**
Represents a soap binding.

**soap12**
Represents a wsdl soap 1.2 binding.

**http_get**
Represents a wsdl http get binding.

**http_put**
Represents a wsdl http post binding.

### 9.2.6 Enumeration MessageLocationa

Location of message content as the body, header or URL

### Literals

**body**
For a soap binding, indicates that the part is in the body of the message.

**header**
For a soap binding, indicates that the part is in the header of the message.

**url**
For a soap binding, indicates that the part is in the URL of the message request.

### 9.2.7 Enumeration OperationKind

WSDL doc or rpc binding. Note for GRA "doc" should always be used.

### Literals

**doc**
Represents a soap:operation style of "document". Doc is recommended by GRA.

**rpc**
Represents a soap:operation style of "rpc".

## 9.3 Phase-1 Derived Properties

Table 2 below summarizes all of the properties in the intermediate model (serialized as annotations.xmi) whose values are calculated by Phase-1. Where the derivation uses the term “instance” it refers to the InstanceSpecification in the annotation model. Where the derivation uses the term “documentation” it means that the related UML model element should have exactly one ownedComment, and the documentation is the contents of the body of that comment. Where the derivation uses the term “corresponding” it means either the realized element, or an element generated by Phase-1 by following the service hierarchy as explained in 7.4.4.

**Table 2 Phase-1 Derived Properties**

| Class | Name | Type | Phase 1 Derivation |
|---|---|---|---|
| Agreement | Documentation | String | Instance documentation. |
| Description | Documentation | String | Instance documentation. |
| GRAServiceAnnotation Base | Diagnostics | String | Contains errors encountered during Phase-1 processing. Not used in Phase-2. |
| GRAServiceAnnotation Base | Documentation | String | Corresponding element's documentation or (if none) instance documentation. |
| GRAServiceAnnotation Base | ModelReference | ModelReference | Corresponding element. |
| GRAServiceAnnotation Base | Name | String | Instance name or (if none) corresponding element's name, or if the corresponding element is a Reception then the name of the Reception’s Signal. |
| IEPDReference | IEPDURL | String | Usage target. |
| IEPDReference | Name | String | Usage target's name. |
| Interface | Operation | Operation | Either explicitly declared or derived from the Operations/Receptions in the service hierarchy. |
| Message | ElementModel Reference | ModelReference | Derived from the NIEM type by selecting a NIEM property (i.e., xml element) whose type is the message part type. |
| Message | ElementName | String | The NIEM name of the referenced element. |
| Message | ElementPrefix | String | Derived from the defaultPrefix tag on the NIEM information model containing the element relating to this Message. |
| Message | Prefix | String | Derived from the defaultPrefix tag on the NIEM information model containing the type relating to this Message. |
| Model | Label | String | The derived target of a ModelReference or an ElementModelReference. The name of the target's model. |

| | | | |
|---|---|---|---|
| Model | ModelURI | String | The derived target of a ModelReference or an ElementModelReference. The URI of the target's model resource. |
| ModelReference | DiagramLink | String | Created from realization, service hierarchy or ElementModelReference. URIs of diagrams associated with the target element. |
| ModelReference | Documentation | String | Created from realization, service hierarchy or ElementModelReference. Documentation of the target element. |
| ModelReference | ElementID | String | Created from realization, service hierarchy or ElementModelReference. Id of the target element. |
| ModelReference | Model | Model | Created from realization, service hierarchy or ElementModelReference. Model containing the target element. |
| ModelReference | Name | String | Created from realization, service hierarchy or ElementModelReference. Name of the target element. |
| Operation | ActionName | String | Corresponding element's name. |
| Operation | ActionPurpose | String | Corresponding element's documentation. |
| Operation | IsAsynchronous | Boolean | True if corresponding element is Reception; false if corresponding element is Operation. |
| Operation | Parameter | Parameter | Either explicitly declared or derived from the parameters of Operations/Receptions in the service hierarchy. |
| Parameter | Message | Message | Either explicitly declared or derived from the parameter's type. |
| Parameter | Use | ParameterUse | Derived from parameter’s direction. |
| Participant | Generalization | Participant | Derived from Actor generalization. |
| Port | Interface | Interface | Either explicitly declared or derived from the parameters of Operations/Receptions in the service hierarchy. |
| SchemaReference | Namespace | String | Namepace of schema. Derived from TargetNamespace of an information model referenced from the service interface. |
| SchemaReference | Prefix | String | Prefix used to reference Schema Namespace. Derived from DefaultPrefix of an information model referenced from the service interface. |
| SchemaReference | SchemaLocation | String | The physical location of the generated schema, relative to SSP Catalog. |
| SecurityClassification | Name | String | Instance name. |

| | | | |
|---|---|---|---|
| Service | Port | Port | Either explicitly declared or derived from Ports on the corresponding Component. |
| Service | ServiceProvider | Participant | Derived from a Realization between the service's realized Component and the participant's realized Actor. |
| ServiceCapability | Documentation | String | Instance documentation. |
| ServiceDescription | Exchange Partner | Participant | Actors playing roles referenced by lifelines in Interactions in realized Collaboration, or where an actor or package containing actors is explicitly realized by an ExchangePartner annotation instance. |
| ServiceDescription | IEPDReference | IEPDReference | Derived from usage to IEPD. |
| ServiceDescription | RealWorld Effect | UseCase | If the modeled real world effect realizes a package, then Phase 1 expands it to give a real world effect for every use case in that package. |
| ServiceDescription | RevisionVersion | Integer | If nothing is modeled, 1 is inserted. |
| ServiceDescription | Service Interaction | ServiceInteraction | Derived from Interactions owned by the realized Collaboration. |
| ServiceInteraction | Participant | Participant | Derived from the Actors that play a role in the Interaction owned by the realized Collaboration. |
| ServiceInterface Specification | Prefix | String | Derived from (the same as) ServiceInterfaceNameAbbreviationText. |
| ServiceInterface Specification | Schema Reference | SchemaReference | Derived from those parts of the information model that are referenced from the service interface. |
| ServiceLevel Agreement | Applicable Agreements Indicator | Boolean | True if and only if there are Agreements. |
| ServiceLevel Agreement | Applicable Contracts Indicator | Boolean | True if and only if there are ApplicableContracts. |
| ServiceLevel Agreement | Applicable Policies Indicator | Boolean | True if and only if there are ApplicablePolicies. |
| ServiceLevel Agreement | Applicable Umbrella Agreements Indicator | Boolean | True if and only if there are UmbrellaAgreements. |

## 9.4 Phase-2 Default Values (Informative)

Table 3 below summarizes property values that are assumed by the baseline Phase-2 if no values are set in the intermediate file by Phase-1.

**Table 3 Phase-2 Default Property Values**

| Class | Name | Type | Default value |
|---|---|---|---|
| Agreement | Automated Agreement Indicator | Boolean | FALSE |
| GRAServiceAnnotation Base | Requirement | Interaction Requirements | Default requirements |
| InteractionRequirements | IdentityAnd Attribute Assertion Transmission | Boolean | FALSE |
| InteractionRequirements | Interface Description Requirements | Boolean | TRUE |
| InteractionRequirements | Logging | Boolean | FALSE |
| InteractionRequirements | Message Addressing | Boolean | TRUE |
| InteractionRequirements | Message Confidentiality | Boolean | FALSE |
| InteractionRequirements | Message Integrity | Boolean | FALSE |
| InteractionRequirements | Message Nonrepudiation | Boolean | FALSE |
| InteractionRequirements | Reliability | Boolean | FALSE |
| InteractionRequirements | Service Authentication | Boolean | FALSE |
| InteractionRequirements | Service Consumer Authentication | Boolean | FALSE |
| InteractionRequirements | Service Consumer Authorization | Boolean | FALSE |
| InteractionRequirements | Service Metadata Availability | Boolean | FALSE |
| InteractionRequirements | Service Responsiveness | Boolean | FALSE |
| InteractionRequirements | Transaction Support | Boolean | FALSE |
| Port | AddressURI | String | ServiceURI+"/"+"Name" |

| | | | |
|---|---|---|---|
| ServiceDescription | Additional Information | Description | "N/A" |
| ServiceDescription | CreationDate | String | [today] |
| ServiceDescription | DataProvenance | Description | "Provenance is defined as the agency, office, or person of origin of records, i.e., the entity that created, received, or accumulated and used the records in the conduct of their business activities. All applicable provenance information or restrictions are provided below. Also, additional artifacts related to this sections content can be provided in the service model folders of the service package. [Service Abbreviation] SSP [Service Version]\artifacts\service model\information model" |
| ServiceDescription | Domain Description | String | "N/A" |
| ServiceDescription | Execution Context | Description | "Service descriptions should include all information pertinent to the production or consumption of the service, including expected infrastructure functions and other dependencies. No information directly pertaining to implementation platform or technology should be included in the service description." |
| ServiceDescription | Other Requirement | Description | "N/A" |
| ServiceDescription | Privacy | Description | "The MOUs between participating entities will further define specific privacy requirements. Global has developed a document, "Implementing Privacy Policy in Justice Information Sharing: A Technical Framework" This document is intended to provide guidelines for supporting the electronic expression of privacy policy and how to convert privacy policy so that it is understandable to computers and software." |
| ServiceDescription | Security | Description | "The service must adhere to the rules of the CJIS Security Policies regarding the Advanced Encryption Standard (AES) level of encryption. The use of Virtual Private Networks (VPNs) or Secure Sockets Layer (SSL) for transport-level security in addition to these requirements is optional." |
| ServiceDescription | Service Assumption | Description | "N/A" |
| ServiceDescription | Service Dependency | Service Identification | "N/A" |
| ServiceDescription | ServiceLevel Agreement | ServiceLevel Agreement | None |
| ServiceDescription | ServiceSecurity Classification | Security Classification | "The highest level of security classification for the information exchanged by this service is Sensitive |

| | | | |
|---|---|---|---|
| | | | but Unclassified (SBU). As a result the service can be assigned a security classification of SBU." |
| ServiceInterface Specification | Message Definition Mechanism | Description | "The service will comply with the message definition mechanisms identified in the GRA Reliable Secure Web Services Service Interaction Profile, Version 1.2 (GRA RS WS-SIP 1.2)." |
| ServiceInterface Specification | Security Description Text | String | "The service must adhere to the rules of the CJIS Security Policies regarding the Advanced Encryption Standard (AES) level of encryption. The use of Virtual Private Networks (VPNs) or Secure Sockets Layer (SSL) for transport-level security in addition to these requirements is optional." |
| ServiceInterface Specification | Security Implemented Indicator | Boolean | TRUE if the technology implements security |
| ServiceInterface Specification | ServiceTesting | Description | "The service validation and testing will leverage the Springboard specification to validate the interoperable aspects of the service interface specification in order to assert that a participating system conforms to the underlying specification. The conformance specification and the associated test cases define a series of tests designed to exercise each interoperability aspect of the specification at least once. The validation testing is not intended to address functional aspects of the systems/services nor is the test suite designed to verify the robustness or performance of the interface software." |
| ServiceLevelAgreement | Approval Required Indicator | Boolean | FALSE |
| ServiceLevelAgreement | Licensing Required Indicator | Boolean | TRUE if licensing specified |
| WSDLInterface | BindingCode | BindingType | soap12 |
| WSDLMessage | Message LocationCode | Message Location | Body |
| WSDLOperation | Operation KindCode | OperationKind | doc |
| WSDLParameter | MimeType | String | text/xml |

## 9.5 Phase-2 Processing (Informative)

Table 4 below summarizes the handling for all of the property values that appear in the intermediate file when processed by the baseline Phase-2 transformations. In most cases the table states the destination for the property; otherwise the processing is signified by an asterisk.

**Table 4 Phase-2 processing**

| Class | Name | Type | Destination / processing |
|---|---|---|---|
| Agreement | AgreementURI | String | Metadata Instance (metadata.xml) |
| Agreement | AutomatedAgreementIndicator | Boolean | * Used to determine if the agreement is automated |
| Agreement | Documentation | String | SDD - Section 4.5: Policies and Contracts |
| Description | Documentation | String | * Used to contain documentation |
| Description | ExternalDocumentation | String | |
| GRAService AnnotationBase | Diagnostics | String | * Used as a supertype to provide standard attributes |
| GRAService AnnotationBase | Documentation | String | |
| GRAService AnnotationBase | Flag | String | |
| GRAService AnnotationBase | ModelReference | Model Reference | |
| GRAService AnnotationBase | Name | String | |
| GRAService AnnotationBase | Requirement | Interaction Requirements | |
| GRAService AnnotationBase | Template | String | |
| IEPDReference | IEPDURL | String | SDD - Section 6.1.1: IEPD Reference |
| IEPDReference | Name | String | |
| Interaction Requirements | ExtendedRequirement | String | SDD - Section 4.1: Service Interaction Requirements |
| Interaction Requirements | IdentityAndAttribute AssertionTransmission | Boolean | SDD - Section 4.1: Service Interaction Requirements<br>SIDD - Section 3: Service Interaction Requirements<br>WSDL - <wsp:Policy> requirements section |
| Interaction Requirements | InterfaceDescription Requirements | Boolean | |
| Interaction Requirements | Logging | Boolean | |
| Interaction Requirements | MessageAddressing | Boolean | |
| Interaction Requirements | MessageConfidentiality | Boolean | |

| | | | |
|---|---|---|---|
| Interaction Requirements | MessageIntegrity | Boolean | |
| Interaction Requirements | MessageNonrepudiation | Boolean | |
| Interaction Requirements | Reliability | Boolean | |
| Interaction Requirements | ServiceAuthentication | Boolean | |
| Interaction Requirements | ServiceConsumerAuthentication | Boolean | |
| Interaction Requirements | ServiceConsumerAuthorization | Boolean | |
| Interaction Requirements | ServiceMetadataAvailability | Boolean | |
| Interaction Requirements | ServiceResponsiveness | Boolean | |
| Interaction Requirements | TransactionSupport | Boolean | |
| Interface | Operation | Operation | SDD - Section 6.2.1 Action Model WSDL <PortType> & <Binding> sections |
| Message | ElementModelReference | Model Reference | WSDL <Part> section |
| Message | ElementName | String | |
| Message | ElementPrefix | String | |
| Message | Prefix | String | WSDL - <Message> & <Part> sections |
| Model | Label | String | * Traceability to the model. |
| Model | ModelURI | String | |
| Model Reference | DiagramLink | String | * Traceability to the model. |
| Model Reference | Documentation | String | |
| Model Reference | ElementID | String | |
| Model Reference | Model | Model | |
| Model Reference | Name | String | |
| Operation | ActionName | String | SDD - Section 6.2.1 Action Model, WSDL - <PortType> & <Binding> Sections |
| Operation | ActionProvenance | String | SDD - Section 6.2.1 Action Model |
| Operation | ActionPurpose | String | Metadata Instance (metadata.xml) |

| | | | |
|---|---|---|---|
| Operation | IsAsynchronous | Boolean | * Not used in the default template. Provided for possible use by customized phase-2 templates. |
| Operation | MessageExchangePattern | Exchange Pattern | SIDD - Section 5. Message Exchange Patterns |
| Operation | Parameter | Parameter | SDD - Section 6.2.1 Action Model, WSDL - <PortType> & <Binding> Sections |
| Organization | OrganizationAcronym | String | Metadata Instance (metadata.xml) |
| Organization | OrganizationFullAddressText | String | Metadata Instance (metadata.xml) |
| Organization | OrganizationPointOfContact | Person | Metadata Instance (metadata.xml) |
| Organization | OrganizationRole DescriptionText | String | Metadata Instance (metadata.xml) |
| Organization | OrganizationRoleDetailed DescriptionText | String | Metadata Instance (metadata.xml) |
| Organization | OrganizationWebSiteURL | String | Metadata Instance (metadata.xml) |
| Parameter | Message | Message | WSDL - <Message> & <Part> Sections |
| Parameter | Use | ParameterUse | WSDL - <PortType> & <Binding> Sections |
| Participant | Generalization | Participant | SDD - Section 2.4: Real World Effects |
| Participant | ParticipatingOrganization | Organization | SDD - Section 2.4: Real World Effects |
| Person | ContactPersonAddress | String | Metadata Instance (metadata.xml) |
| Person | ContactPersonEmailID | String | Metadata Instance (metadata.xml) |
| Person | ContactPersonPhoneNumberID | String | Metadata Instance (metadata.xml) |
| Port | AddressURI | String | WSDL - <Service> Section |
| Port | Interface | Interface | WSDL - <PortType> & <Binding> Sections |
| SampleData | ExpectedOutput | String | * Not used in the default template. Provided for possible use by customized phase-2 templates. |
| SampleData | Input | String | * Not used in the default template. Provided for possible use by customized phase-2 templates. |
| Schema Reference | Namespace | String | WSDL - <Types> Section - Schema Imports |
| Schema Reference | Prefix | String | |
| Schema Reference | SchemaLocation | String | |
| Security Classification | Name | String | SDD - Section2.7: Security Classification |
| Service | Port | Port | WSDL - <PortType> & <Binding> Sections |

| | | | |
|---|---|---|---|
| Service | SampleData | SampleData | * Not used in the default template. Provided for possible use by customized phase-2 templates. |
| Service | ServiceProvider | Participant | SIDD - Section 1: Introduction |
| Service Capability | Documentation | String | SDD - Section 2.3: Capabilities |
| Service Description | ActivationDate | String | Metadata Instance (metadata.xml) |
| Service Description | AdditionalInformation | Description | SDD - Section 5: Additional Information |
| Service Description | AlertAndNotificationURI | String | Metadata Instance (metadata.xml) |
| Service Description | Classification | String | Metadata Instance (metadata.xml) |
| Service Description | CreationDate | String | Metadata Instance (metadata.xml) |
| Service Description | DataProvenance | Description | SDD - Section 6.1.4: Data Provenance |
| Service Description | DomainDescription | String | Metadata Instance (metadata.xml) |
| Service Description | Endorsements | String | Metadata Instance (metadata.xml) |
| Service Description | ExchangePartner | Participant | Metadata Instance (metadata.xml) |
| Service Description | ExecutionContext | Description | SDD - Section 4.4: Execution Context |
| Service Description | ExpirationDate | String | Metadata Instance (metadata.xml) |
| Service Description | IEPDReference | IEPDReference | SDD - Section 6.1.1: IEPD Reference |
| Service Description | LastRevisionDate | String | SDD & SIDD - Cover Page |
| Service Description | LifecycleStatus | String | Metadata Instance (metadata.xml) |
| Service Description | MajorVersion | Integer | SDD - Section 2.8: Service Specification Package Version |
| Service Description | MinorVersion | Integer | SDD - Section 2.8: Service Specification Package Version |
| Service Description | NextRevisionDate | String | Metadata Instance (metadata.xml) |
| Service Description | OtherRequirement | Description | SDD - Section 4.8: Other Requirements |
| Service Description | Privacy | Description | SDD - Section 4.7: Privacy<br>SIDD - Section 10: Privacy |

| | | | |
|---|---|---|---|
| Service Description | ProcessModel | Description | SDD - Section 6.2.2: Process Model |
| Service Description | RealWorldEffect | UseCase | SDD - Section 2.4: Real-World Effects |
| Service Description | RelatedOrganization | Organization | SDD & SIDD - Cover Page |
| Service Description | RevisionVersion | Integer | Metadata Instance (metadata.xml) |
| Service Description | Security | Description | SDD - Section 4.6: Security<br>SIDD - Section 9: Security |
| Service Description | ServiceAssumption | Description | SDD - Section 4.2: Service Assumptions |
| Service Description | ServiceCapability | Service Capability | SDD - Section 2.3: Capabilities |
| Service Description | ServiceDependency | Service Identification | SDD - Section 4.3: Service Dependencies |
| Service Description | ServiceDescription KeywordText | String | Metadata Instance (metadata.xml) |
| Service Description | ServiceDescription SummaryText | String | SDD - Section 2.5: Summary |
| Service Description | ServiceInteraction | Service Interaction | SDD - Section 4.1: Service Interaction Requirements |
| Service Description | ServiceInterface | Service Interface Specification | SDD - Section 4.1: Service Interaction Requirements<br>SIDD - Section 3: Service Interaction Requirements<br>WSDL - <wsp:Policy> requirements section |
| Service Description | ServiceLevelAgreement | ServiceLevel Agreement | SDD - Section 4.5: Policies and Contracts |
| Service Description | ServicePurpose | Description | SDD - Section 2.6: Description |
| Service Description | ServiceScopeDescription | Description | SDD - Section 2.2: Scope |
| Service Description | ServiceSecurityClassification | SecurityClassification | SDD - Section2.7: Security Classification |
| Service Description | Sponsor | String | SDD & SIDD - Cover Page |
| Service Description | TransformationURI | String | N/A |
| Service Identification | ServiceID | String | Metadata Instance (metadata.xml) |
| Service Identification | ServiceNameAbbreviationText | String | SDD & SIDD - Cover Page, Section 1: Introduction |
| Service | ServiceURI | String | Metadata Instance (metadata.xml) |

| | | | |
|---|---|---|---|
| Identification | | | |
| Service Interaction | Participant | Participant | SDD - Section 2.4: Real World Effects |
| Service Interaction Profile | SIPName | String | * The SIP elements influence the InteractionRequirements. Refer to InteractionRequirements for details. |
| Service Interaction Profile | SIPVersion | String | |
| ServiceInterface Specification | MessageDefinitionMechanism | Description | SIDD - Section 6: Message Definition Mechanisms |
| ServiceInterface Specification | PhysicalModel | Description | SIDD - Section 2: Physical Model |
| ServiceInterface Specification | Prefix | String | WSDL - Target Namespace Prefix |
| ServiceInterface Specification | SchemaReference | Schema Reference | WSDL - <Types> Section - Schema Imports |
| ServiceInterface Specification | SecurityDescriptionText | String | Metadata Instance (metadata.xml) |
| ServiceInterface Specification | SecurityImplementedIndicator | Boolean | Metadata Instance (metadata.xml) |
| ServiceInterface Specification | Service | Service | Metadata Instance (metadata.xml) |
| ServiceInterface Specification | ServiceInteractionProfile | Service Interaction Profile | Metadata Instance (metadata.xml) |
| ServiceInterface Specification | ServiceInterfaceName AbbreviationText | String | Metadata Instance (metadata.xml) |
| ServiceInterface Specification | ServiceTesting | Description | SIDD - Section 11: Service Testing |
| ServiceInterface Specification | TargetNamespace | String | WSDL - Target Namespace |
| ServiceLevel Agreement | Agreement | Agreement | SDD - Section 4.5: Policies and Contracts |
| ServiceLevel Agreement | ApplicableAgreementsIndicator | Boolean | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | ApplicableContract | Agreement | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | ApplicableContractsIndicator | Boolean | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | ApplicablePoliciesIndicator | Boolean | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | ApplicablePolicy | Agreement | Metadata Instance (metadata.xml) |
| ServiceLevel | ApplicableUmbrellaAgreements | Boolean | Metadata Instance (metadata.xml) |

| | | | |
|---|---|---|---|
| Agreement | Indicator | | |
| ServiceLevel Agreement | ApprovalRequiredIndicator | Boolean | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | CreationCostAmount | String | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | LicensingAgreement | Agreement | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | LicensingRequiredIndicator | Boolean | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | ServiceAvailability | String | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | ServiceAverageThroughput | String | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | ServiceMaximumThroughput | String | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | ServiceResponseTime | String | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | UmbrellaAgreement | Agreement | SIDD - Section 8: Umbrella Agreements |
| ServiceLevel Agreement | UsageCostAmount | String | Metadata Instance (metadata.xml) |
| ServiceLevel Agreement | UsageUnitCostAmount | String | Metadata Instance (metadata.xml) |
| UseCase | Consumer | Participant | SDD - Section 2.4: Real World Effects |
| UseCase | Provider | Participant | SDD - Section 2.4: Real World Effects |
| WSDLInterface | BindingCode | BindingType | WSDL - Binding Section |
| WSDLMessage | MessageLocationCode | Message Location | WSDL - Binding Section |
| WSDLOperation | OperationKindCode | OperationKind | WSDL - Binding Section |
| WSDLParameter | MimeType | String | WSDL - Binding Section |

# 10 GRA-UML Workflow (Informative)

## 10.1 User's workflow

This clause explains the workflow that the GRA-UML architect uses to create a new GRA-UML model and provision an SSP. It identifies the process steps and the various normative and ancillary files that are used in the process. A summary of all of the files published by this specification appears in Annex C.

The starting point is a UML tool loaded with the GRA-UML profile and the GRA Annotations library.

To start work, the user may load into this tool the GRATemplate file. This is a starter file that contains the basic elements of a GRA-UML model, comprising a starter Logical Service Model and related annotations. These elements are organized into the package structure shown in Figure 17. This structure is purely for convenience: the user will most likely rename the packages to be more suitable for the domain being modeled. The MyGRAModel package applies the GRAProfile, enabling the GRA-UML stereotypes to be applied within the model.

The user should also load into the tool one or more [NIEM-UML] IEPDs that constitute information models for the domain being implemented. These IEPDs provide types which should be used as parameter and result types for operations within the GRA-UML interfaces.

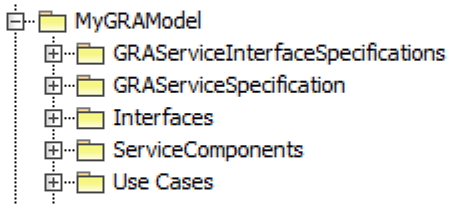


**Figure 17 GRA Template folder structure**

The user will then create a Logical Service Model. The starter LSM provides a single UseCase with two Actors, as shown in Figure 18; a single Component with a Port providing an Interface, as shown in Figure 19; a Collaboration with parts typed by the Component and Actors, as shown in Figure 20; and an Interaction in which the consumer Actor calls an operation on the service Component. This model is intended as a stylistic template for the GRA-UML architect to extend and enhance to represent the domain being implemented.

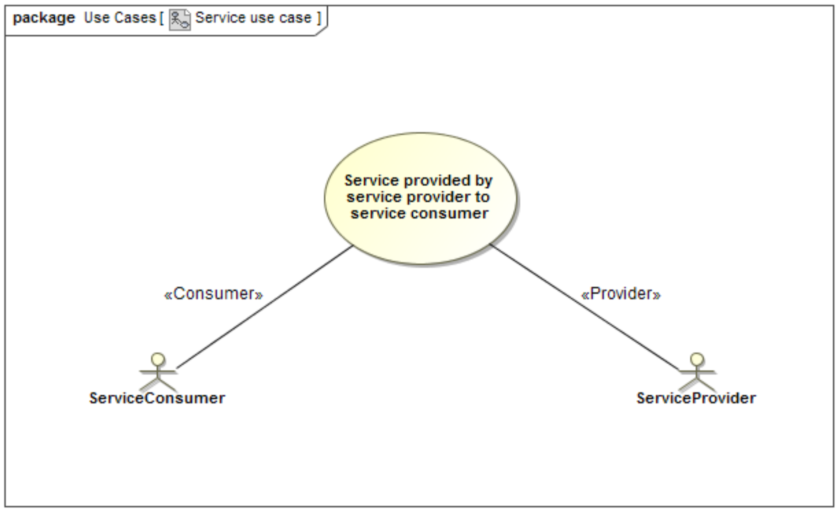

package Use Cases [ Service use case ]
Service provided by service provider to service consumer
«Consumer»
«Provider»
ServiceConsumer
ServiceProvider

**Figure 18 Starter UseCase and Actors**

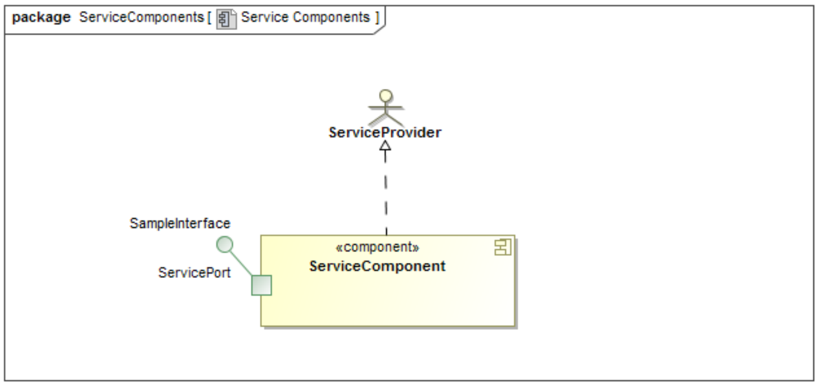


**Figure 19 Starter Component**

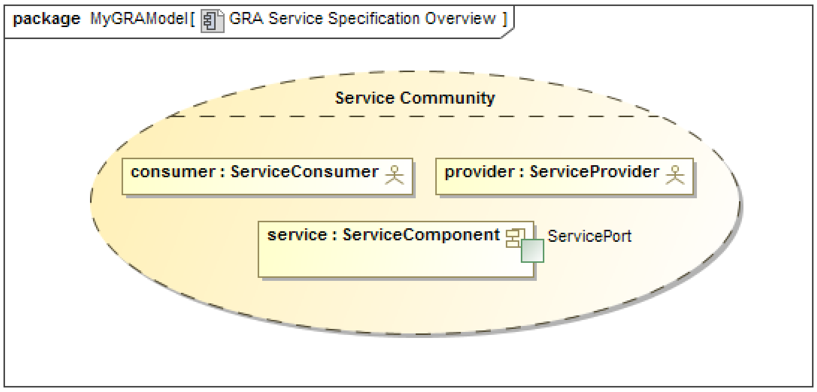

package MyGRAModel [ GRA Service Specification Overview ]
Service Community
consumer : ServiceConsumer
provider : ServiceProvider
service : ServiceComponent
ServicePort

**Figure 20 Starter Collaboration**

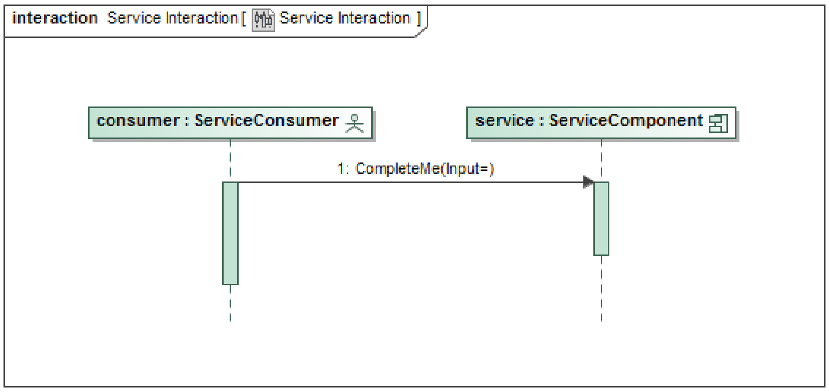
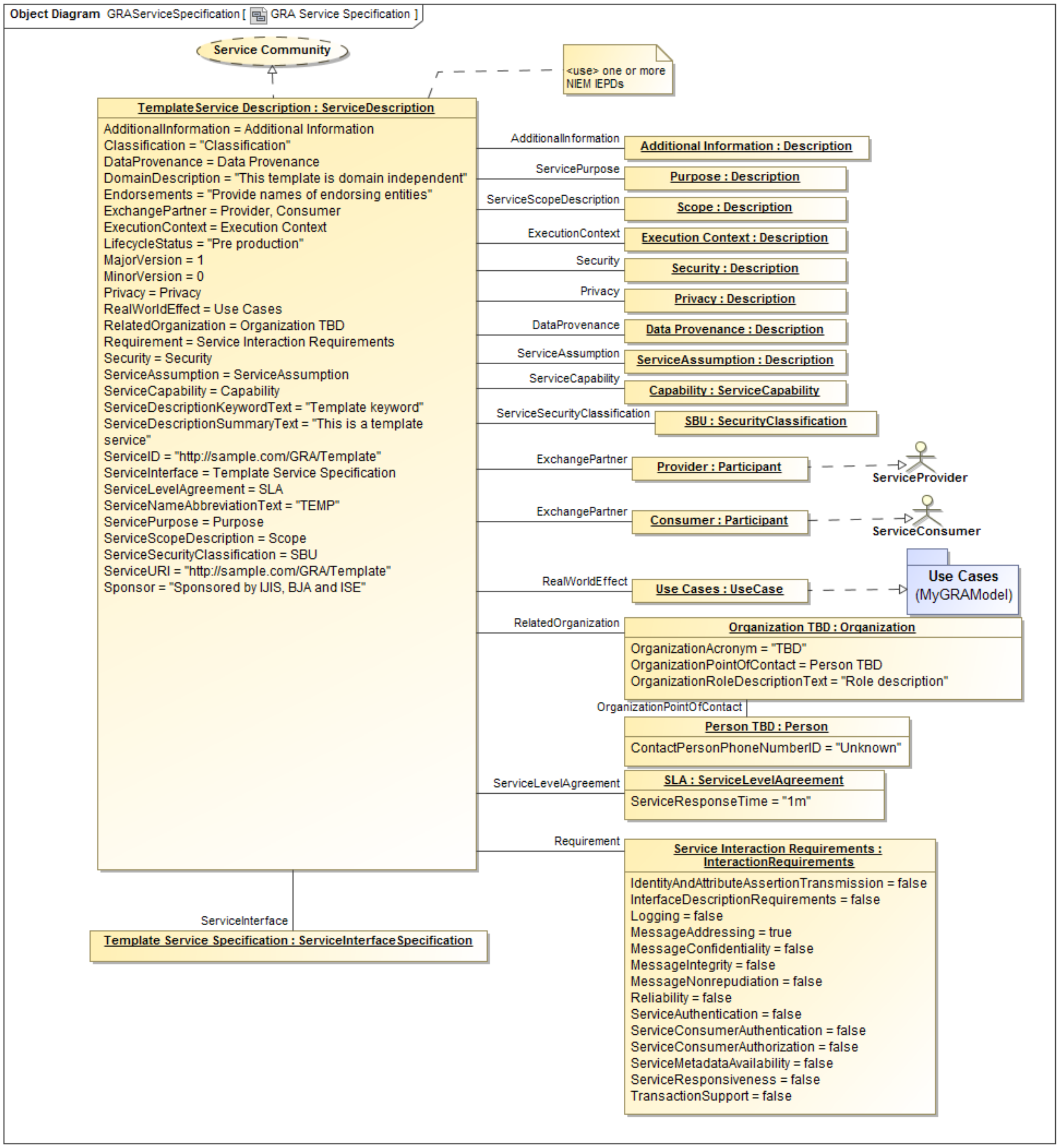

interaction Service Interaction [ Service Interaction ]
consumer : ServiceConsumer
service : ServiceComponent
1: CompleteMe(Input=)

**Figure 21 Starter Interaction**

Having made a Logical Service Model, the GRA-UML Architect should create annotations to drive the provisioning of the SSP. There are starter annotations in the template. Figure 22 shows the starter service annotations, and Figure 23 shows the starter service interface annotations. Note that the TemplateServiceDescription instance realizes the Service Community Collaboration; the Provider and Consumer instances realize the ServiceProvider and ServiceConsumer Actors; the Use Cases instance realizes the Use Cases Package; and the Template Service instance realizes the ServiceComponent Component. The architect should follow a similar pattern of realizations for all of the Actors, UseCases and Components in the Logical Service Model.

The architect should create Usages between the ServiceDescription and the [NIEM-UML] IEPDs that constitute the information model. Operations and Receptions in the model should use types within the information model for their parameters, results and exceptions.

The architect will also need to change the starter metadata values into values suitable for the SSP being implemented, and provide defaults and explicit annotations where non-default provisioning is required.



**TemplateService Description : ServiceDescription**

AdditionalInformation = Additional Information
Classification = "Classification"
DataProvenance = Data Provenance
DomainDescription = "This template is domain independent"
Endorsements = "Provide names of endorsing entities"
ExchangePartner = Provider, Consumer
ExecutionContext = Execution Context
LifecycleStatus = "Pre production"
MajorVersion = 1
MinorVersion = 0
Privacy = Privacy
RealWorldEffect = Use Cases
RelatedOrganization = Organization TBD
Requirement = Service Interaction Requirements
Security = Security
ServiceAssumption = ServiceAssumption
ServiceCapability = Capability
ServiceDescriptionKeywordText = "Template keyword"
ServiceDescriptionSummaryText = "This is a template service"
ServiceID = "http://sample.com/GRA/Template"
ServiceInterface = Template Service Specification
ServiceLevelAgreement = SLA
ServiceNameAbbreviationText = "TEMP"
ServicePurpose = Purpose
ServiceScopeDescription = Scope
ServiceSecurityClassification = SBU
ServiceURI = "http://sample.com/GRA/Template"
Sponsor = "Sponsored by IJIS, BJA and ISE"

**Organization TBD : Organization**

OrganizationAcronym = "TBD"
OrganizationPointOfContact = Person TBD
OrganizationRoleDescriptionText = "Role description"

**Person TBD : Person**

ContactPersonPhoneNumberID = "Unknown"

**SLA : ServiceLevelAgreement**

ServiceResponseTime = "1m"

**Service Interaction Requirements : InteractionRequirements**

IdentityAndAttributeAssertionTransmission = false
InterfaceDescriptionRequirements = false
Logging = false
MessageAddressing = true
MessageConfidentiality = false
MessageIntegrity = false
MessageNonrepudiation = false
Reliability = false
ServiceAuthentication = false
ServiceConsumerAuthentication = false
ServiceConsumerAuthorization = false
ServiceMetadataAvailability = false
ServiceResponsiveness = false
TransactionSupport = false

**Figure 22 Starter service annotations**

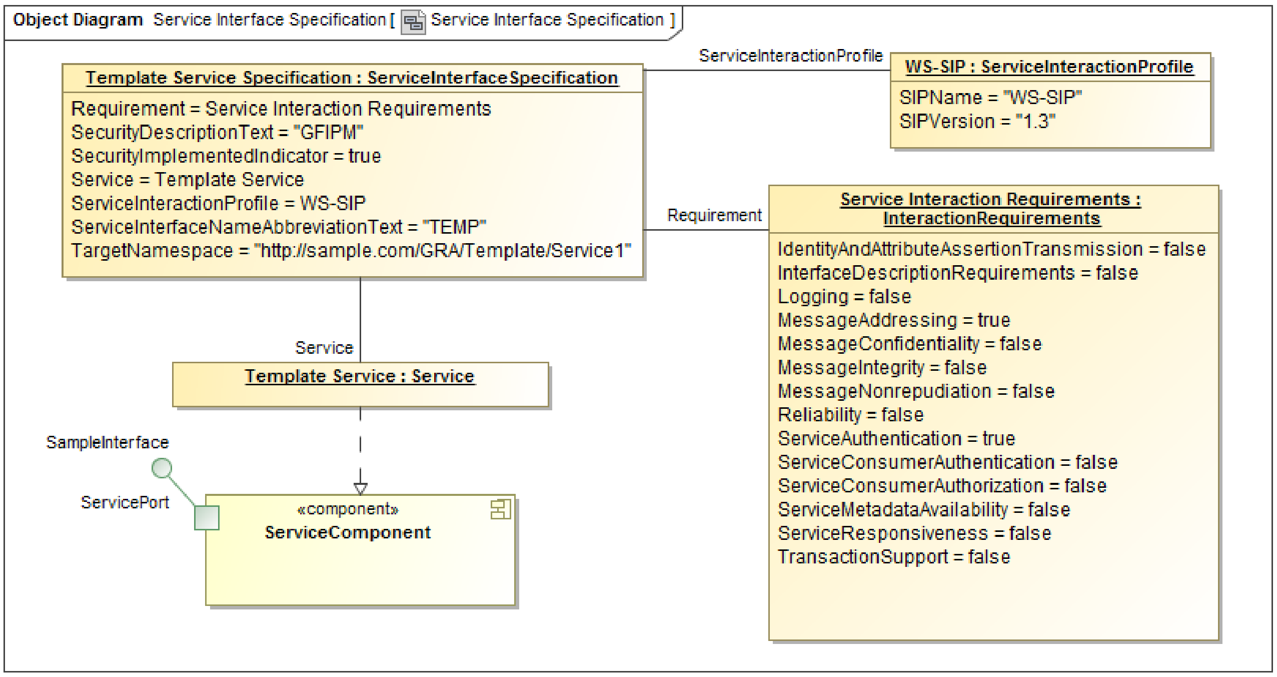

Object Diagram Service Interface Specification [ Service Interface Specification ]
Template Service Specification : ServiceInterfaceSpecification
Requirement = Service Interaction Requirements
SecurityDescriptionText = "GFIPM"
SecurityImplementedIndicator = true
Service = Template Service
ServiceInteractionProfile = WS-SIP
ServiceInterfaceNameAbbreviationText = "TEMP"
TargetNamespace = "http://sample.com/GRA/Template/Service1"
ServiceInteractionProfile
WS-SIP : ServiceInteractionProfile
SIPName = "WS-SIP"
SIPVersion = "1.3"
Requirement
Service Interaction Requirements : InteractionRequirements
IdentityAndAttributeAssertionTransmission = false
InterfaceDescriptionRequirements = false
Logging = false
MessageAddressing = true
MessageConfidentiality = false
MessageIntegrity = false
MessageNonrepudiation = false
Reliability = false
ServiceAuthentication = true
ServiceConsumerAuthentication = false
ServiceConsumerAuthorization = false
ServiceMetadataAvailability = false
ServiceResponsiveness = false
TransactionSupport = false
Service
Template Service : Service
SampleInterface
ServicePort
«component»
ServiceComponent

**Figure 23 Starter service interface annotations**

In a more complex case, the architect may need to create multiple service interface specifications based on the pattern shown in Figure 23, perhaps using different Service Interaction Profiles and different sets of requirements.

Once the architect has completed the GRA-UML model, the model should be saved. Then Phase-1 and Phase-2 processing need to be applied, as described in the next section. How this processing is actually invoked and coordinated is tool-specific.

## 10.2 Example Automated workflow

This section describes a particular Eclipse-based implementation of the GRA-UML standard which uses specific tools to co-ordinate the execution of Phase-1 and Phase-2 provisioning. Alternative implementations may use different tools and intermediate artifacts. The remainder of this description assumes that Eclipse plugin is installed.

The GRA-UML model created as described in 10.1 should be saved in a format consumable by the tool that executes Phase-1. In the Eclipse implementation, this is the Eclipse UML2 v4.x format, and the files that are saved include the GRA-UML model itself and all of its dependencies, including the NIEM IEPD, and GRA and NIEM profiles. These files end with the .uml suffix. These files should be moved to a folder contained in an Eclipse project.

Once the files are in the Eclipse workspace, select the primary GRA-UML model, right-click, and select Generate GRA SSP. Both phases of the provisioning process will run automatically, and place all of the generated files in the correct places in the SSP folder structure. The first phase will generate annotations.xmi, and will also place in the root folder an Apache Ant script (http://ant.apache.org/) called graPhase2.ant.xml. The execution of this script constitutes Phase-2. It has four stages:

1. Download from the StandardTemplate (see Annex D) the xslt processor and a copy of GRA's metadata.xsd.
2. Create the SSP folder structure.
3. Execute the XSLT transformations: either those located in the StandardTemplate or another location specified in the model. These produce:
    a. OMG-compliant XMI2.4 rendering of the model in the artifacts\service model\OMG folder.

b. Other technology-specific renderings of the model in the artifacts\service model\Eclipse and artifacts\service model\MD folders.

c. Catalog.xml, catalog.html, metadata.xml and metadata.xsd in the root folder.

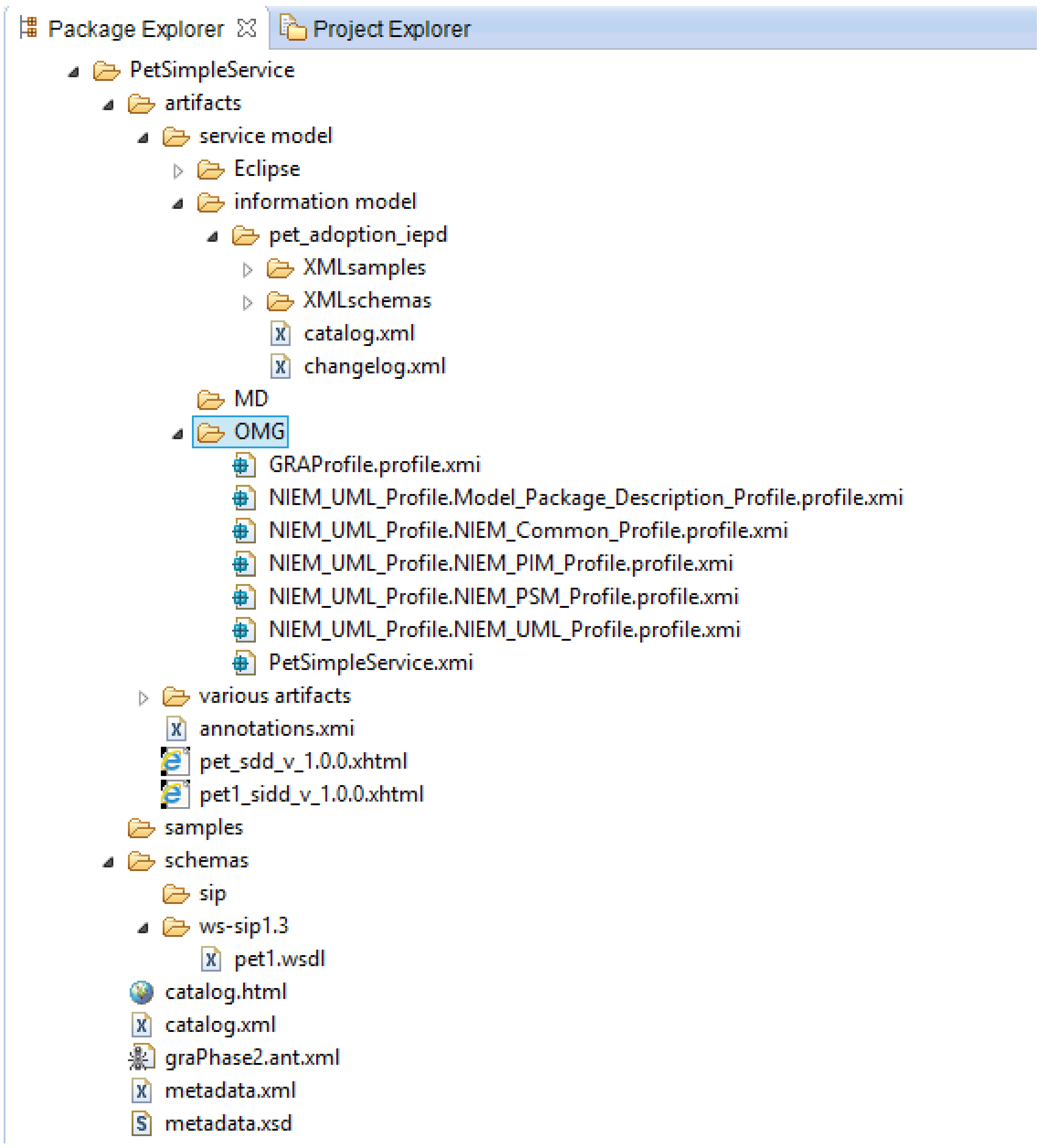
d. The SDD and SIDD XHTML files located in the artifacts folder.

e. WSDL files in the schemas\[SIP] folders.

4. Delete temporary files.

Finally the GRA plugin invokes a [NIEM-UML] transformation to create the NIEM IEPD artifacts in the artifacts\service model\information model folder.

The end result in the Eclipse package explorer should look something like Figure 24.



```
Package Explorer    Project Explorer
PetSimpleService
    artifacts
        service model
            Eclipse
            information model
                pet_adoption_iepd
                    XMLsamples
                    XMLschemas
                    catalog.xml
                    changelog.xml
            MD
            OMG
                GRAProfile.profile.xmi
                NIEM_UML_Profile.Model_Package_Description_Profile.profile.xmi
                NIEM_UML_Profile.NIEM_Common_Profile.profile.xmi
                NIEM_UML_Profile.NIEM_PIM_Profile.profile.xmi
                NIEM_UML_Profile.NIEM_PSM_Profile.profile.xmi
                NIEM_UML_Profile.NIEM_UML_Profile.profile.xmi
                PetSimpleService.xmi
        various artifacts
        annotations.xmi
        pet_sdd_v_1.0.0.xhtml
        pet1_sidd_v_1.0.0.xhtml
    samples
    schemas
        sip
        ws-sip1.3
            pet1.wsdl
    catalog.html
    catalog.xml
    graPhase2.ant.xml
    metadata.xml
    metadata.xsd
```

**Figure 24 GRA files provisioned in an Eclipse project**

# Annex A GRA-UML Example

(informative)

This annex contains a simple example to illustrate the GRA-UML modeling concepts. The example describes a Pet Adoption Center: a fictitious agency whose business is the adoption of (presumably unwanted) pets. This is an extension of the example elaborated in the [NIEM-UML] profile, emphasizing that the information transferred by GRA services is specified using NIEM. The aim is to generate a Service Specification Package (SSP) that defines the services provided and consumed in the pet adoption community.

We start with the Platform Independent Model (PIM). This is a UML model, built using the Logical Service Profile under GRA-UML conventions, which describes services provided by the Pet Adoption Centre.

The PIM contains Actors, which depict the entities that interact when services are offered or consumed. Actors are often people or classes of people, organizations or classes of organizations, or even machines or classes of machines. In this example there are two Actors: the PetAdoptionCenter, and Anyone.

In the PIM, a UseCase represents the user's view of a service, which is known in GRA as the "real world effect". In this example there is one service: "Provide information on a particular pet adoption based on the ID of the adopting individual". The Actors and the UseCase are shown together on a single diagram in Figure 25, which also shows by means of associations stereotyped «Provider» and «Consumer» that the PetAdoptionCenter provides this service, and Anyone may consume it. These Actors and UseCase are contained in a Package called Use Cases: this fact will be important later.

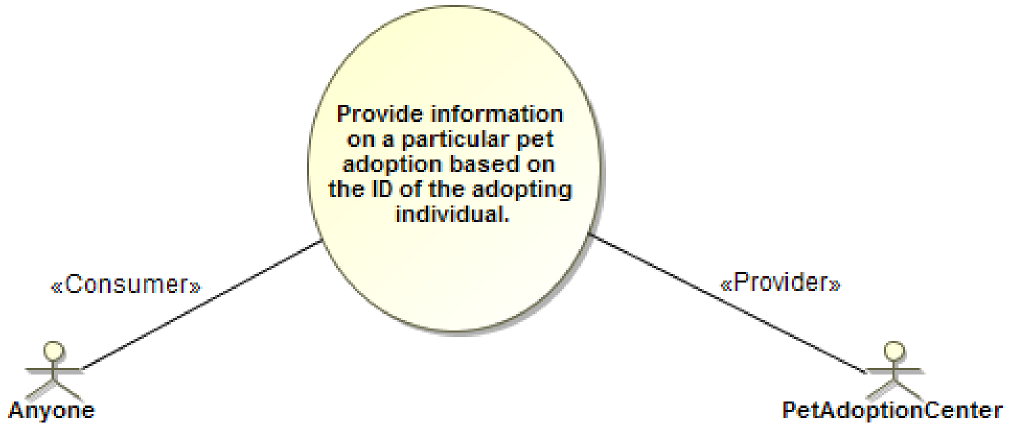

Provide information on a particular pet adoption based on the ID of the adopting individual.
«Consumer»
«Provider»
Anyone
PetAdoptionCenter

**Figure 25 Pet Adoption Actors and UseCase**

Interfaces in the PIM define the operations and signal receptions that service providers implement to offer a service. For the PetAdoptionCenter there is a single Interface, shown in Figure 26, with a single Operation.

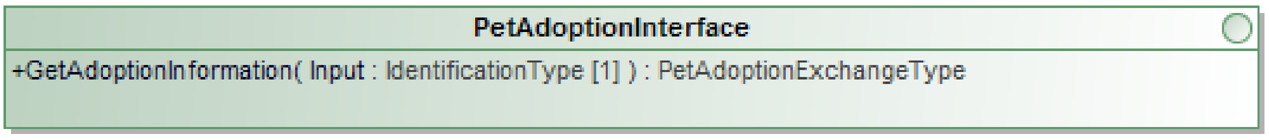

PetAdoptionInterface
+GetAdoptionInformation( Input : IdentificationType [1] ) : PetAdoptionExchangeType

**Figure 26 PetAdoptionInterface**

The argument and result of the operation directly reference [NIEM-UML] message types IdentificationType and PetAdoptionExchangeType. IdentificationType comes from PetAdoptionNIEMCoreSubset, and PetAdoptionExchangeType from PetAdoptionExtension, both of which are InformationModels in the NIEM IEPD

(Information Exchange Package Documentation) depicted in Figure 27, which is constructed according to the rules and conventions of [NIEM-UML]. Full explanation of this IEPD can be found in the [NIEM-UML] specification.

The GetAdoptionInformation operation also has a raisedException of PetAdoptionExtension::Exception, which is not shown on the diagram, but flows through the SSP provisioning process into the eventual WSDL.

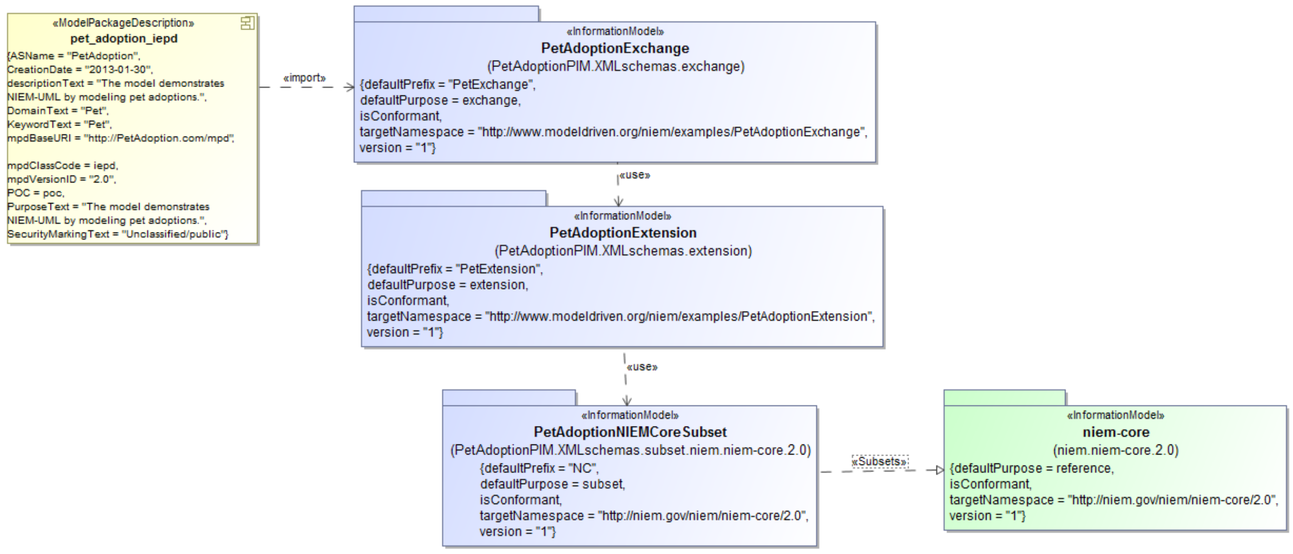

«ModelPackageDescription»
pet_adoption_iepd
{ASName = "PetAdoption",
CreationDate = "2013-01-30",
descriptionText = "The model demonstrates NIEM-UML by modeling pet adoptions.",
DomainText = "Pet",
KeywordText = "Pet",
mpdBaseURI = "http://PetAdoption.com/mpd",
mpdClassCode = iepd,
mpdVersionID = "2.0",
POC = poc,
PurposeText = "The model demonstrates NIEM-UML by modeling pet adoptions.",
SecurityMarkingText = "Unclassified/public"}
«import»
«InformationModel»
PetAdoptionExchange
(PetAdoptionPIM.XMLschemas.exchange)
{defaultPrefix = "PetExchange",
defaultPurpose = exchange,
isConformant,
targetNamespace = "http://www.modeldriven.org/niem/examples/PetAdoptionExchange",
version = "1"}
«use»
«InformationModel»
PetAdoptionExtension
(PetAdoptionPIM.XMLschemas.extension)
{defaultPrefix = "PetExtension",
defaultPurpose = extension,
isConformant,
targetNamespace = "http://www.modeldriven.org/niem/examples/PetAdoptionExtension",
version = "1"}
«use»
«InformationModel»
PetAdoptionNIEMCoreSubset
(PetAdoptionPIM.XMLschemas.subset.niem.niem-core.2.0)
{defaultPrefix = "NC",
defaultPurpose = subset,
isConformant,
targetNamespace = "http://niem.gov/niem/niem-core/2.0",
version = "1"}
«Subsets»
«InformationModel»
niem-core
(niem.niem-core.2.0)
{defaultPurpose = reference,
isConformant,
targetNamespace = "http://niem.gov/niem/niem-core/2.0",
version = "1"}

**Figure 27 Pet Adoption IEPD**

In a GRA-UML PIM, Interfaces are provided by Ports on Components. In this example there is a single Component with a single Port – called ServicePort - which provides the PetAdoptionInterface, as shown in Figure 28.

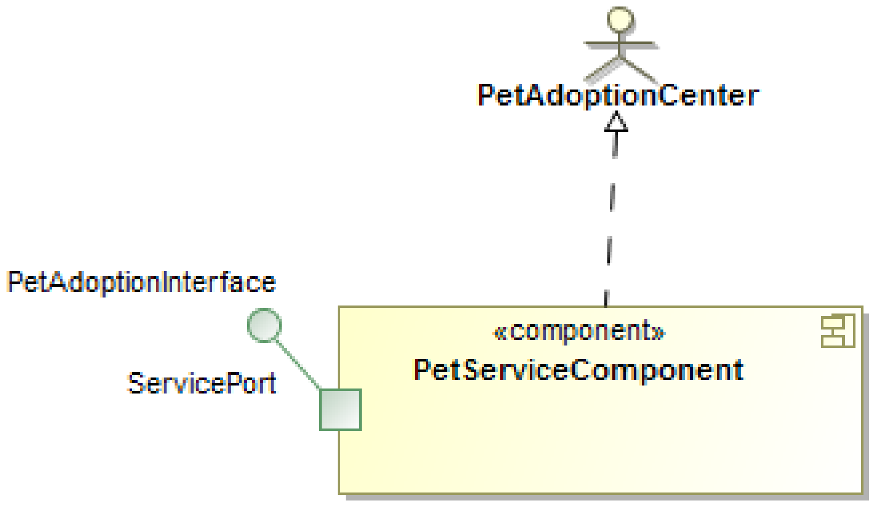


**Figure 28 PetServiceComponent**

Figure 28 also shows PetServiceComponent realizing the PetAdoptionCenter actor: the dashed arrow with the open triangle arrowhead signifies a UML Realization. This means that the Component represents a service offered by that Actor.

The PetServiceComponent realizing the PetAdoptionCenter may also be shown on the UseCase diagram, as illustrated by Figure 29.

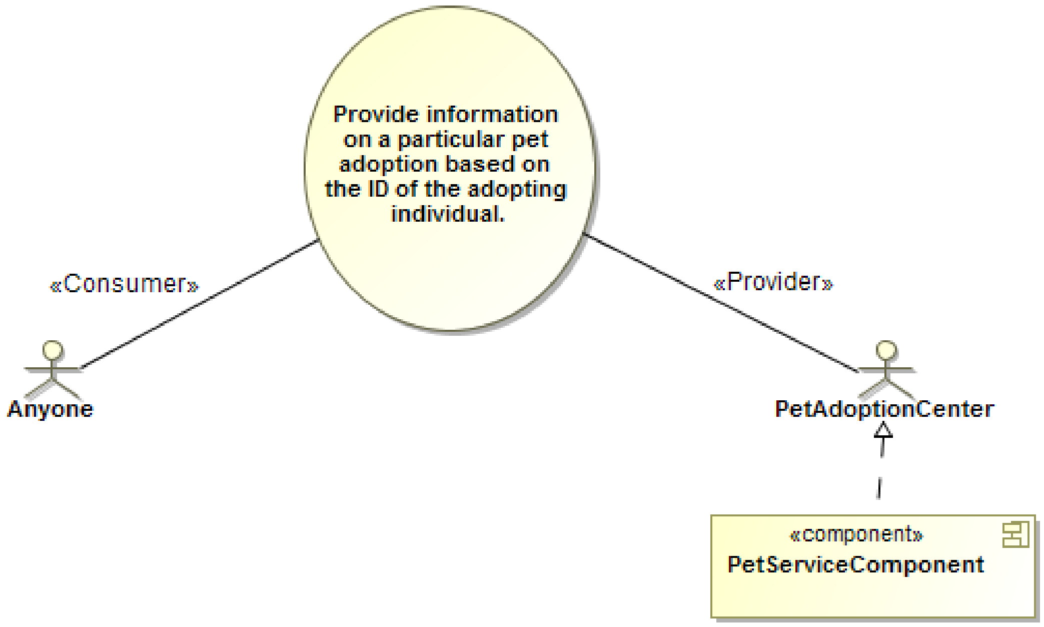

Provide information on a particular pet adoption based on the ID of the adopting individual.
«Consumer»
«Provider»
Anyone
PetAdoptionCenter
«component»
PetServiceComponent

**Figure 29 UseCase diagram showing Component realization**

The central element of the PIM is a single UML Collaboration that represents the community involved in the provision and consumption of pet adoption services, and pulls together the participants by representing them as roles in the Collaboration.

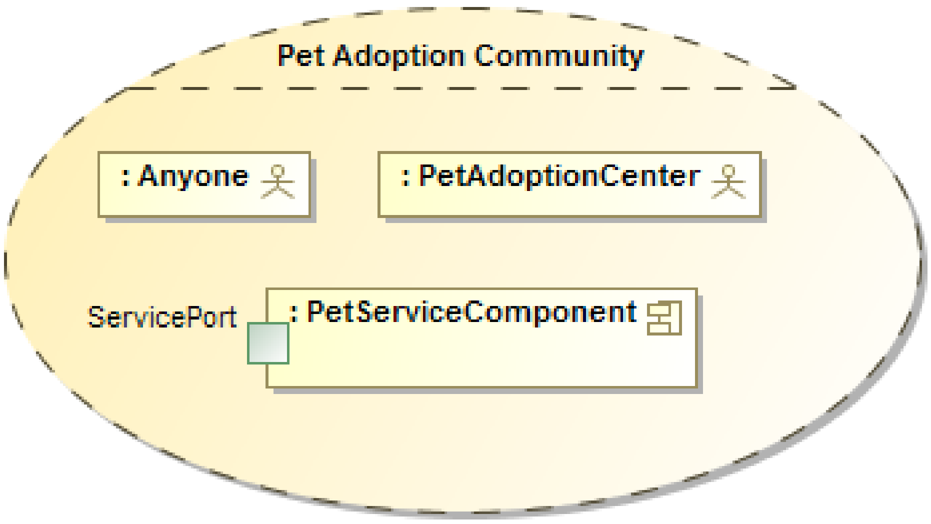


**Figure 30 Pet Adoption Community Collaboration**

The final aspect of the Pet Adoption PIM is shown in the Interaction in Figure 31, in which Anyone synchronously invokes the GetAdoptionInformation operation on the PetAdoptionInterface offered by a PetServiceComponent.

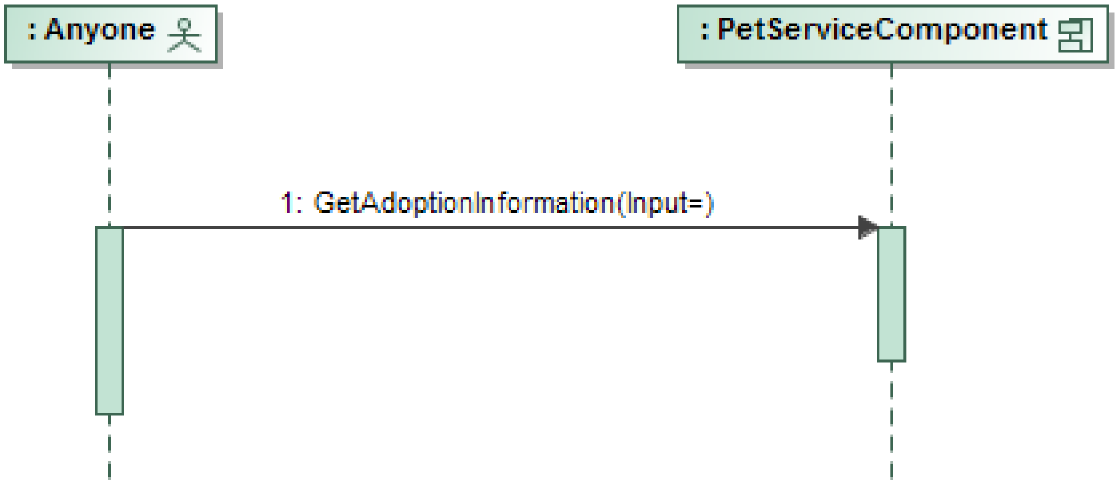


**Figure 31 Pet Adoption Interaction**

In the UML model, such Interactions are owned by the Collaboration that represents the subject community, and their Lifelines represent roles that the actors and components play in that Collaboration.

Together, these elements – the Collaboration, UseCases and Actors, Components, Ports, Interfaces and Interactions comprise the complete PIM for the example. This, on its own, is insufficient to generate the complete SSP. To complete the story it is necessary to provide annotations that add metadata sufficient for the generation task. These annotations are carried by UML InstanceSpecifications (which for readability we will call instances) whose types are classes in the GRA Annotations model specified in clause 9.

Figure 32 shows the annotations that primarily relate to the generation of the Service Description Document (SDD). The focus of this model is the Pet Adoption Service Description instance. The UML notation for an instance is a rectangle whose label contains the name of the instance, followed by a colon, followed by the type of the instance. In this case the label is Pet Adoption Service Description : ServiceDescription, indicating that the name is Per Adoption Service Description and the type is ServiceDescription.

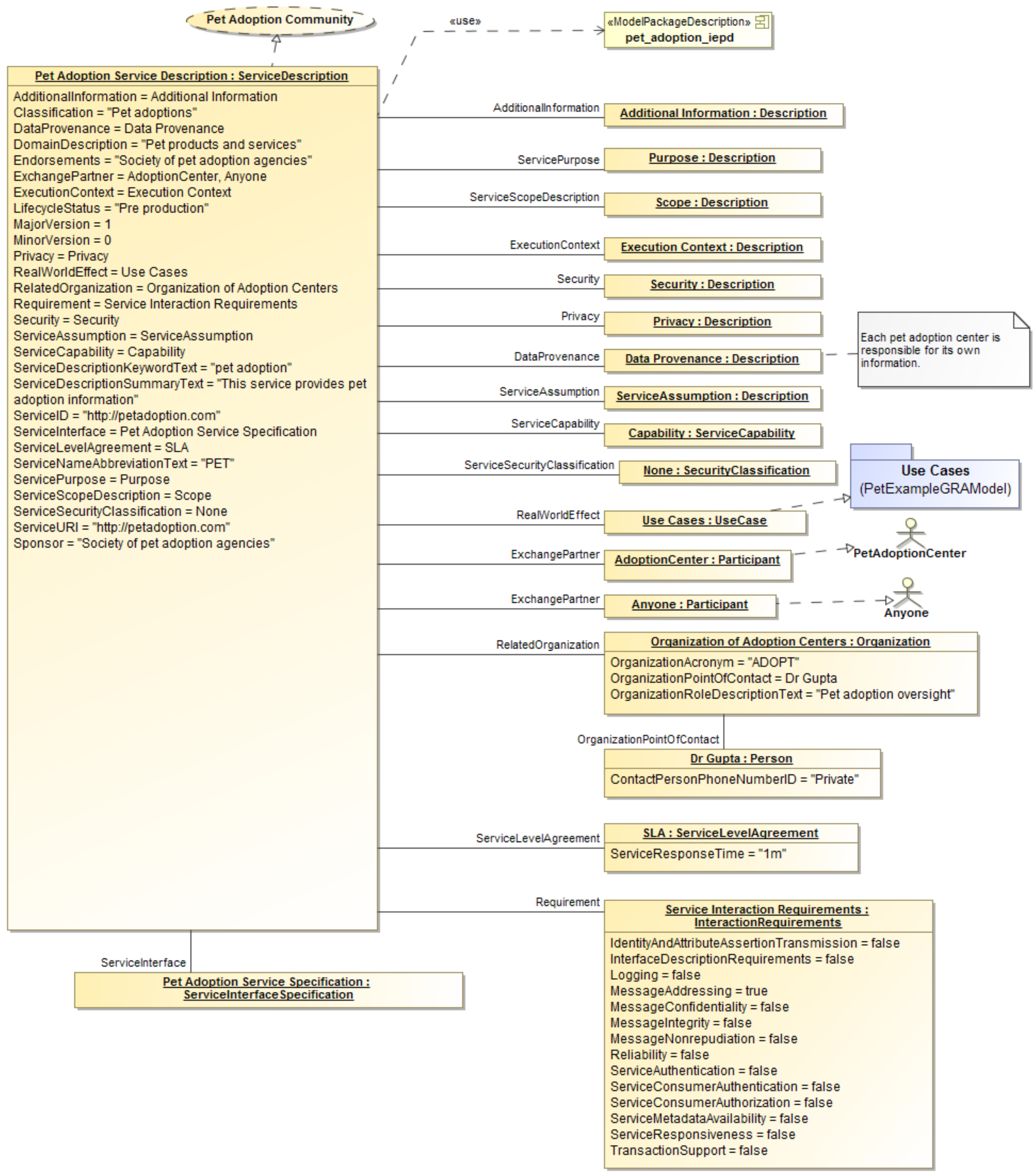
The Pet Adoption Service Description realizes the Pet Adoption Community collaboration and uses the pet_adoption_iepd. In any GRA-UML PIM it is required that each ServiceDescription instance realizes a collaboration that represents its community, and uses one or more IEPDs that encapsulate its information model. The two-phase generation process uses the ServiceDescription instance to initiate the generation of all of the artifacts that will ultimately constitute the SSP.

In the main body of the Pet Adoption Service Description are a number of properties, in alphabetical order, together with their values in the form propertyname = value. Many of these properties have simple types: String, Integer, Boolean. For example, MajorVersion = 1 and ServiceNameAbbreviationText = “PET”. These property values flow through the generation process and find their place (or places) in the provisioned artifacts.

Linked to Pet Adoption Service Description are several other instances. In most cases, the link is also redundantly represented in the main body of Pet Adoption Service Description. For example the third property in the body appears as DataProvenance = Data Provenance. This is a redundant representation of the link depicted as a line between Pet Adoption Service Description and the Data Provenance instance, which also carries the property name DataProvenance.

*Note: Depending on the tool being used, depicting such properties redundantly in the body of the instance may be a display option, which a user might choose differently.*

Eight of these linked instances have the type Description: Additional Information, Purpose, Scope, Execution Context, Security, Privacy, Data Provenance and ServiceAssumption.



Pet Adoption Community
«use»
«ModelPackageDescription»
pet_adoption_iepd
Pet Adoption Service Description : ServiceDescription
AdditionalInformation = Additional Information
Classification = "Pet adoptions"
DataProvenance = Data Provenance
DomainDescription = "Pet products and services"
Endorsements = "Society of pet adoption agencies"
ExchangePartner = AdoptionCenter, Anyone
ExecutionContext = Execution Context
LifecycleStatus = "Pre production"
MajorVersion = 1
MinorVersion = 0
Privacy = Privacy
RealWorldEffect = Use Cases
RelatedOrganization = Organization of Adoption Centers
Requirement = Service Interaction Requirements
Security = Security
ServiceAssumption = ServiceAssumption
ServiceCapability = Capability
ServiceDescriptionKeywordText = "pet adoption"
ServiceDescriptionSummaryText = "This service provides pet adoption information"
ServiceID = "http://petadoption.com"
ServiceInterface = Pet Adoption Service Specification
ServiceLevelAgreement = SLA
ServiceNameAbbreviationText = "PET"
ServicePurpose = Purpose
ServiceScopeDescription = Scope
ServiceSecurityClassification = None
ServiceURI = "http://petadoption.com"
Sponsor = "Society of pet adoption agencies"
AdditionalInformation
Additional Information : Description
ServicePurpose
Purpose : Description
ServiceScopeDescription
Scope : Description
ExecutionContext
Execution Context : Description
Security
Security : Description
Privacy
Privacy : Description
DataProvenance
Data Provenance : Description
Each pet adoption center is responsible for its own information.
ServiceAssumption
ServiceAssumption : Description
ServiceCapability
Capability : ServiceCapability
ServiceSecurityClassification
None : SecurityClassification
Use Cases
(PetExampleGRAModel)
RealWorldEffect
Use Cases : UseCase
ExchangePartner
AdoptionCenter : Participant
PetAdoptionCenter
ExchangePartner
Anyone : Participant
Anyone
RelatedOrganization
Organization of Adoption Centers : Organization
OrganizationAcronym = "ADOPT"
OrganizationPointOfContact = Dr Gupta
OrganizationRoleDescriptionText = "Pet adoption oversight"
OrganizationPointOfContact
Dr Gupta : Person
ContactPersonPhoneNumberID = "Private"
ServiceLevelAgreement
SLA : ServiceLevelAgreement
ServiceResponseTime = "1m"
Requirement
Service Interaction Requirements : InteractionRequirements
IdentityAndAttributeAssertionTransmission = false
InterfaceDescriptionRequirements = false
Logging = false
MessageAddressing = true
MessageConfidentiality = false
MessageIntegrity = false
MessageNonrepudiation = false
Reliability = false
ServiceAuthentication = false
ServiceConsumerAuthentication = false
ServiceConsumerAuthorization = false
ServiceMetadataAvailability = false
ServiceResponsiveness = false
TransactionSupport = false
ServiceInterface
Pet Adoption Service Specification : ServiceInterfaceSpecification

**Figure 32 Service Specification annotation model**

Each instance carries its own documentation. In UML terms, this is a single comment owned by the instance. Many UML tools have a special user interface to display this comment outside of the diagram. In the case of Data Provenance, the documentation comment is shown both on the diagram and in Figure 33: this will flow through the generation process and ends up in the SDD's Data Provenance section.

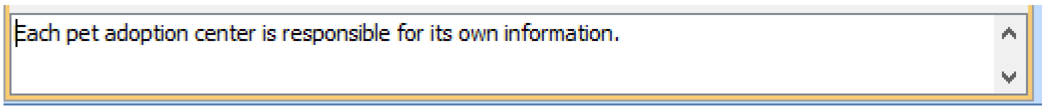

Each pet adoption center is responsible for its own information.

**Figure 33 Data Provenance documentation**

The class Description also provides the possibility for a property containing a URL of external documentation, which would be incorporated into the generated SDD.

Other instances in the diagram have different types: Capability : ServiceCapability; None : SecurityClassification; Organization of Adoption Centers : Organization; Dr. Gupta : Person; SLA : ServiceLevelAgreement. Each of these provides metadata that flows through the provisioning process into the SDD documentation.

Some of the instances play especially important roles in the provisioning process. Service Interaction Requirements : InteractionRequirements provide top-level defaults for the interaction requirements, as described in section 7.4.5. They may be – and in this example will be – overridden at low levels of the hierarchy.

The RealWorldEffect property refers to the instance UseCases : UseCase. Notice that UseCases realizes the Use Cases Package in the PIM. The effect of this is that the Phase-1 generation will create a RealWorldEffect for every UseCase in that package. In fact there is only one - “Provide information on a particular pet adoption based on the ID of the adopting individual” – but if there were several UseCases in the package, then a RealWorldEffect would be generated for each of them. Don’t confuse the GRA Annotation class UseCase with the UML metaclass with the same name: the GRA Annotation UseCase instance is used to tell the generation process which UML UseCases in the PIM should be included in the output. For example, a model containing one annotation UseCase element that realizes a package containing three UML UseCases will cause the generation of three annotation UseCase elements in the intermediate annotations.xmi file, each referring to the corresponding PIM UseCase.

The ExchangePartner property refers to two instances of Participant called AdoptionCenter and Anyone, each of which realizes an Actor: AdoptionCenter realizes PetAdoptionCenter, and Anyone realizes Anyone. These ExchangePartners will appear in the intermediate annotations.xmi file. Just as for UseCases, an equivalent effect could have been produced by a single ExchangePartner realizing the Use Cases package. In that case, the Phase-1 generation would have looked for Actors in the package and generated an ExchangePartner for each Actor.

The final instance shown in Figure 32 is Pet Adoption Service Interface Specification : ServiceInterfaceSpecification. This is an elided representation of the ServiceInterfaceSpecification instance that is shown more completely in Figure 34. Remembering from section 7.1 that an SSP contains one SDD and one or more SIDDs, the annotation model must similarly contain one ServiceDescription and one or more ServiceInterfaceSpecifications.

Pet Adoption Service Interface Specification contains some properties with simple types, such as SecurityDescriptionText = “NONE”. It is linked to instances representing the ServiceInteractionProfile, the Service, and an overriding InteractionRequirements instance, in which the property ServiceAuthentication = true instead of the false value which would otherwise have been inherited from the top-level requirements.

Service instances are required to realize Components in the PIM, in order that the generation process can create the corresponding service hierarchy described in section 7.4.4. In this case, the single Service instance called Pet Adoption Service realizes the PetServiceComponent.

The Pet Adoption Service also has an OperationDefault, which is a WSDLOperation instance that realizes GetAdoptionInformation. This ensures that this operation will have a MessageExchangePattern of “notification”, an OperationKindCode of “doc” and an ActionProvenance of “Mandated by IT”.

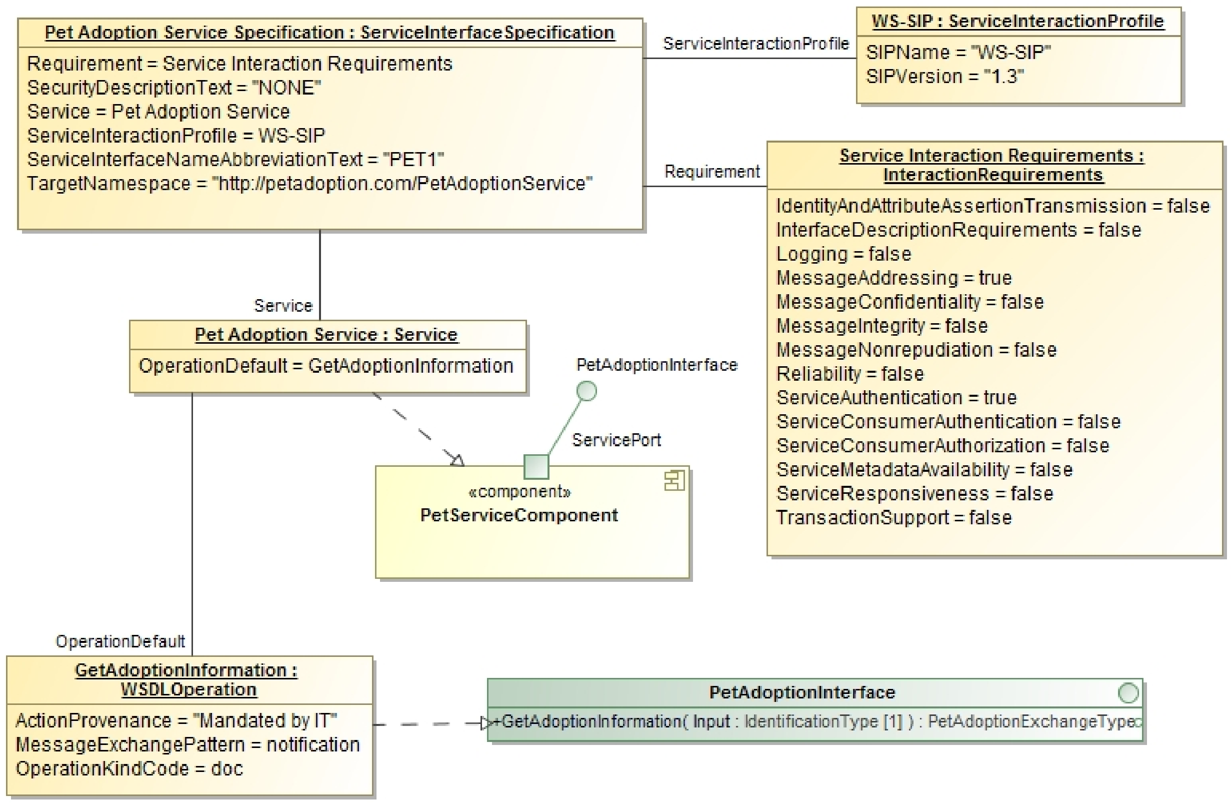

Pet Adoption Service Specification : ServiceInterfaceSpecification
Requirement = Service Interaction Requirements
SecurityDescriptionText = "NONE"
Service = Pet Adoption Service
ServiceInteractionProfile = WS-SIP
ServiceInterfaceNameAbbreviationText = "PET1"
TargetNamespace = "http://petadoption.com/PetAdoptionService"
ServiceInteractionProfile
WS-SIP : ServiceInteractionProfile
SIPName = "WS-SIP"
SIPVersion = "1.3"
Requirement
Service Interaction Requirements : InteractionRequirements
IdentityAndAttributeAssertionTransmission = false
InterfaceDescriptionRequirements = false
Logging = false
MessageAddressing = true
MessageConfidentiality = false
MessageIntegrity = false
MessageNonrepudiation = false
Reliability = false
ServiceAuthentication = true
ServiceConsumerAuthentication = false
ServiceConsumerAuthorization = false
ServiceMetadataAvailability = false
ServiceResponsiveness = false
TransactionSupport = false
Service
Pet Adoption Service : Service
OperationDefault = GetAdoptionInformation
PetAdoptionInterface
ServicePort
«component»
PetServiceComponent
OperationDefault
GetAdoptionInformation : WSDLOperation
ActionProvenance = "Mandated by IT"
MessageExchangePattern = notification
OperationKindCode = doc
PetAdoptionInterface
+GetAdoptionInformation( Input : IdentificationType [1] ) : PetAdoptionExchangeType

**Figure 34 Service Interface Specification annotation model**

# Annex B Mappings to other specifications

(informative)

The section details the relationship between GRA-UML concepts and the concepts defined by Oasis in the SOA reference model and SOA ontology. Because the GRA is based on the SOA reference model the correspondence is very close and most of the terms match exactly. The following table maps Oasis reference model and ontology concepts to GRA-UML and [NIEM-UML] concepts.

| **Oasis Reference Model** | **Oasis Ontology** | **GRA-UML / NIEM-UML** |
|---|---|---|
| Attributes | Attributes | UML Attributes and relations in NIEM-UML model |
| | Concept | NIEM-UML Class |
| Service | | Service realizing UML Component |
| Execution Context | | ExecutionContext |
| Contract and policy | | ServiceLevelAgreement |
| Visibility | | N/A |
| Service Description | ServiceDescription | ServiceDescription |
| Interaction | | ServiceInteraction realizing UML Interaction |
| Information Model | | NIEM-UML Model |
| Behavior Model | | GRA-UML Model |
| Action Model | | Set of operations in service |
| Process model | | ProcessModel |
| RealWorldEffect | CapabilityDescription pre- and post-conditions | RealWorldEffect |
| Mediation | | N/A |
| Capability | CapabilityDescription | ServiceCapability |
| Goal Description | GoalDescription | ServicePurpose |
| | Ontology | Model |
| | Relation | UML Association |
| | Instance | UML Instance Specification |
| | Axiom | UML Constraints |
| Interface | Interface | Interface realizing UML Interface |
| Functionality | CapabilityDescription pre- and post-conditions | ServiceDescription documentation |
| Choreography | Choreography | ServiceInteraction realizing UML Interaction |
| Orchestration | Orchestration | N/A (Optional UML activity diagram) |

# Annex C Machine Readable Files

## Normative files

GRA Profile in UML 2.4.1 serialized in XMI 2.4.2
http://www.omg.org/spec/GRA-UML/20141101/GRAProfile.xmi

GRA Annotation Model in UML 2.4.1 serialized in XMI 2.4.2
http://www.omg.org/spec/GRA-UML/20141101/GRAAnnotations.xmi

GRA Intermediate MetaModel in EMOF 2.4.1 serialized in XMI 2.4.2
http://www.omg.org/spec/GRA-UML/20141101/GRAAnnotationModel.xmi

GRA Intermediate XML Schema, with target namespace http://ijis.org/GRA/Annotations
http://www.omg.org/spec/GRA-UML/20141101/graAnnotationModelXMI.xsd

GRA Intermediate WSDL-specific XML Schema, with target namespace http://ijis.org/GRA/WSDLAnnotations
http://www.omg.org/spec/GRA-UML/20141101/graWsdlXMI.xsd

GRA model to intermediate file QVT transformation
http://www.omg.org/spec/GRA-UML/20141101/GraSspModel2template.qvto

GRA QVT transformation helper
http://www.omg.org/spec/GRA-UML/20141101/GraCommon.qvto

## Informative files

GRA starter template in UML 2.4.1 serialized in XMI 2.4.2
http://www.omg.org/spec/GRA-UML/20141101/Informative/GRATemplate.xmi

GRA Standard Template for Phase-2 processing. The StandardTemplate comprises a folder structure containing numerous xslt, xml and other files at standard locations: see Annex D.
http://www.omg.org/spec/GRA-UML/20141101/Informative/StandardTemplate.zip

GRA-UML Examples in MagicDraw format, including profile and annotations model
http://www.omg.org/spec/GRA-UML/20141101/Informative/PetSimpleService/PetSimpleService.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/PetSimpleService/PetIEPD.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/SIRS/SIRS-Service.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/SIRS/SIRS-Information.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/Corrections/Corrections_SSP_v_1.0.0.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/Corrections/Corrections_IEPD_v_1.0.0.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/Corrections/MessageMetadata.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/Corrections/MessageMetadataReference.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/PetTest/PetTestService.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/PetTest/PetIEPD.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/GRAAnnotations.mdzip
http://www.omg.org/spec/GRA-UML/20141101/Informative/GRAProfile.mdzip

## Ancillary files

*MagicDraw source files:*

GRA Profile: GRAProfile.mdzip

Starter annotation model template: GRATemplate.mdzip

GRA Annotation Model: GRAAnnotations.zip

GRA Combined Annotation Model: GRAAnnotationModelCombined.mdzip

Updated NIEM profile to work with Eclipse: NIEM-UML-Profile.mdzip

# Annex D Standard Template (Informative)

The standard template is an arrangement of folders and files structured as shown by the nested structure below. This structure mirrors the structure of a provisioned SSP, with the XSLT templates positioned in the corresponding place in the structure to the files they will generate.  Folders are shown in *italics* and files in **bold**.

- *SSP*
  - *artifacts*
    - *service model*
      - *behavior model*
      - *information model*
        - *IEPD*
    - *various artifacts*
      - *assumptions*
      - *dependencies*
      - *execution context*
      - *monitoring*
      - *other*
        - *wsdl parts*
          - **wsa.xml**
          - **wsrm.xml**
          - **wss.xml**
          - **wss-arg-s.xml**
          - **wss-arg-se.xml**
          - **wss-wsa-arg-s.xml**
          - **wss-wsa-arg-se.xml**
        - *xslt utilities*
          - **gra_ssp_css.xslt**
          - **gra_ssp_util_return.xslt**
          - **gra_ssp_util_sir.xslt**
          - **gra_ssp_util_sir_table.xslt**
          - **gra_ssp_wsdl_binding.xslt**
          - **gra_ssp_wsdl_policy.xslt**
          - **gra_ssp_wsdl_sir.xslt**
          - **gra_ssp_wsdl_util.xslt**
      - *policies and contracts*
      - *privacy*
      - *security*
      - *testing*
    - **gra_eclipse_uml_2_omg.xslt**
    - **gra_ssp_sdd.xslt**
    - **gra_ssp_sidd.xslt**
  - *samples*
    - *information*
    - *sip*
  - *schemas*
    - *sip*
      - **gra_ssp_wsdl.xslt**
  - **gra_ssp_catalog_html.xslt**
  - **gra_ssp_catalog_html.xslt**
  - **gra_ssp_metadata.xslt**
  - **graPhase2.ant.xml    // Ant script to drive phase 2**
  - **Metadata.xsd    // GRA-provided schema for metadata file**
  - **saxon9he.jar    // Open-source Saxon XSLT processor**