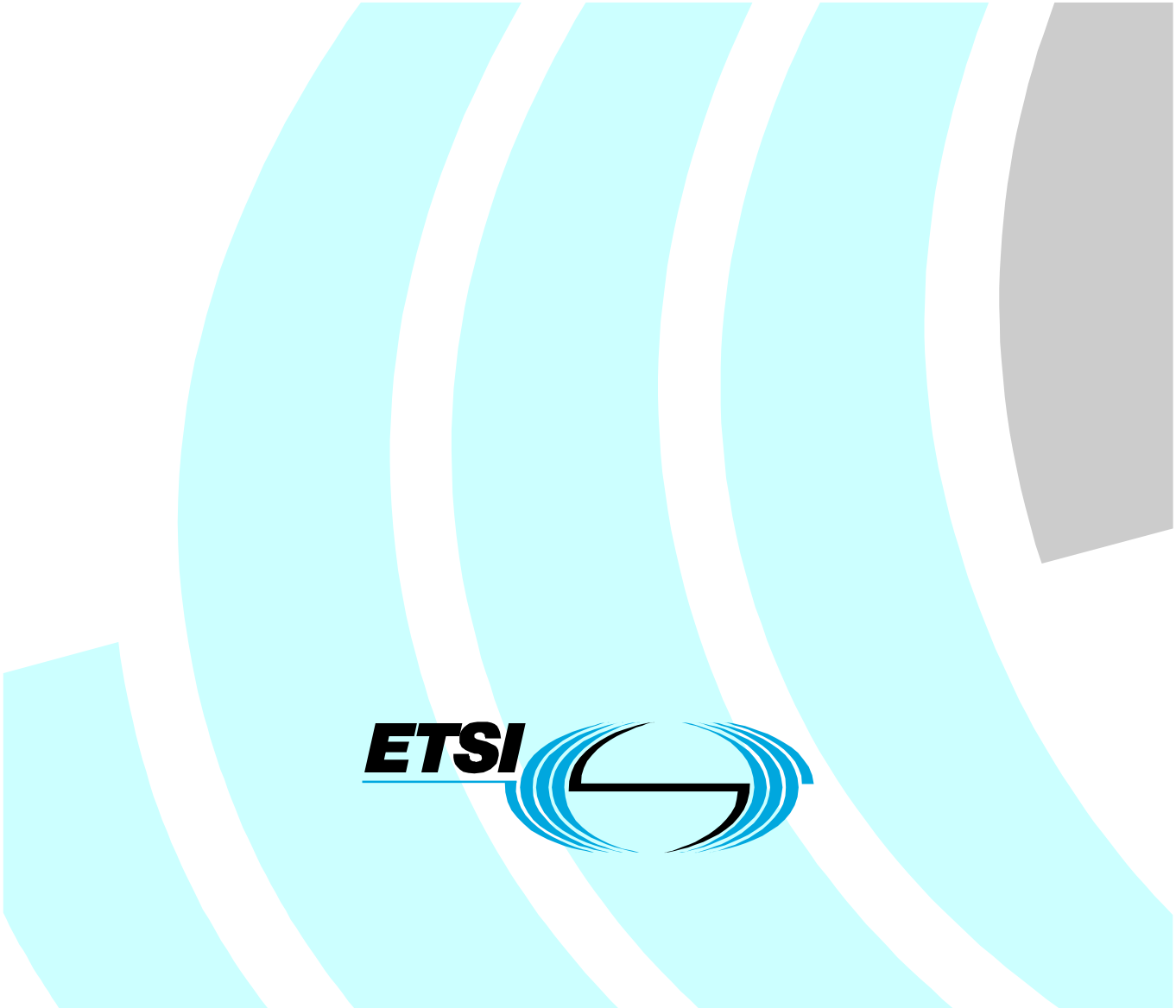
ETSI TR 101 882 V5.1.1 (2004-05)

Technical Report

Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 5; Protocol Framework Definition and Interface Requirement Definition; General



Reference
RTR/TISPAN-02014-TIPHON_R5

Keywords
Interface, IP, Protocol

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, send your comment to:
editor@etsi.org

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN).

Introduction

This version of the document differs from earlier versions in the following areas:

- The document status is changed from Technical Specification to Technical Report.
- It is presented as a single part document with one clause describing the behaviour of each group of service capabilities.
- The definition language is UML as opposed to SDL.
- Data is defined in UML terms and not using ASN.1 (the data can be translated to ASN.1 or any other format as appropriate).

In this version of the document each group of service capabilities identified in TR 101 878 [1] is presented as follows:

- static data model;
- service capabilities as UML operations within the class and identification of active interfaces to the class;
- state-chart diagrams of the operation of each service capability identifying signals that invoke the operation and the data each operation affects.

NOTE: Some service capabilities defined in TR 101 878 [1] have been identified to be services rather than service capabilities, i.e. they can be built from combination of the specified service capabilities. In Annex A it is defined for each of these services how they can be built from the specified service capabilities.

1 Scope

The present document specifies behavioural models in UML for each of the service capabilities defined in TR 101 878 [1].

The requirements expressed in the present document applies to all TIPHON compliant products.

2 References

For the purposes of this Technical Report (TR), the following references apply:

[1] ETSI TR 101 878: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 5; Service Capability Definition; Service Capabilities for a Multi Media Call".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

service capability: specified function that is used either alone or in combination with other service capabilities to realize a complete service application

NOTE: A single service capability may be used in more than one service application.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASN.1	Abstract Syntax Notation 1
NNI	Network Network Interface
QoS	Quality of Service
SDL	Specification and Description Language
UML	Unified Modelling Language
UNI	User Network Interface

4 Profile group

4.1 Introduction

The profile group of service capabilities is defined in TR 101 878 [1] and contains the service capabilities necessary for ser profile control, including user registration and authentication.

The profile is described in the present document as an active UML class shown in figure 1, identifying the ports and interfaces. The signals used to invoke the service capabilities are identified in figure 2.

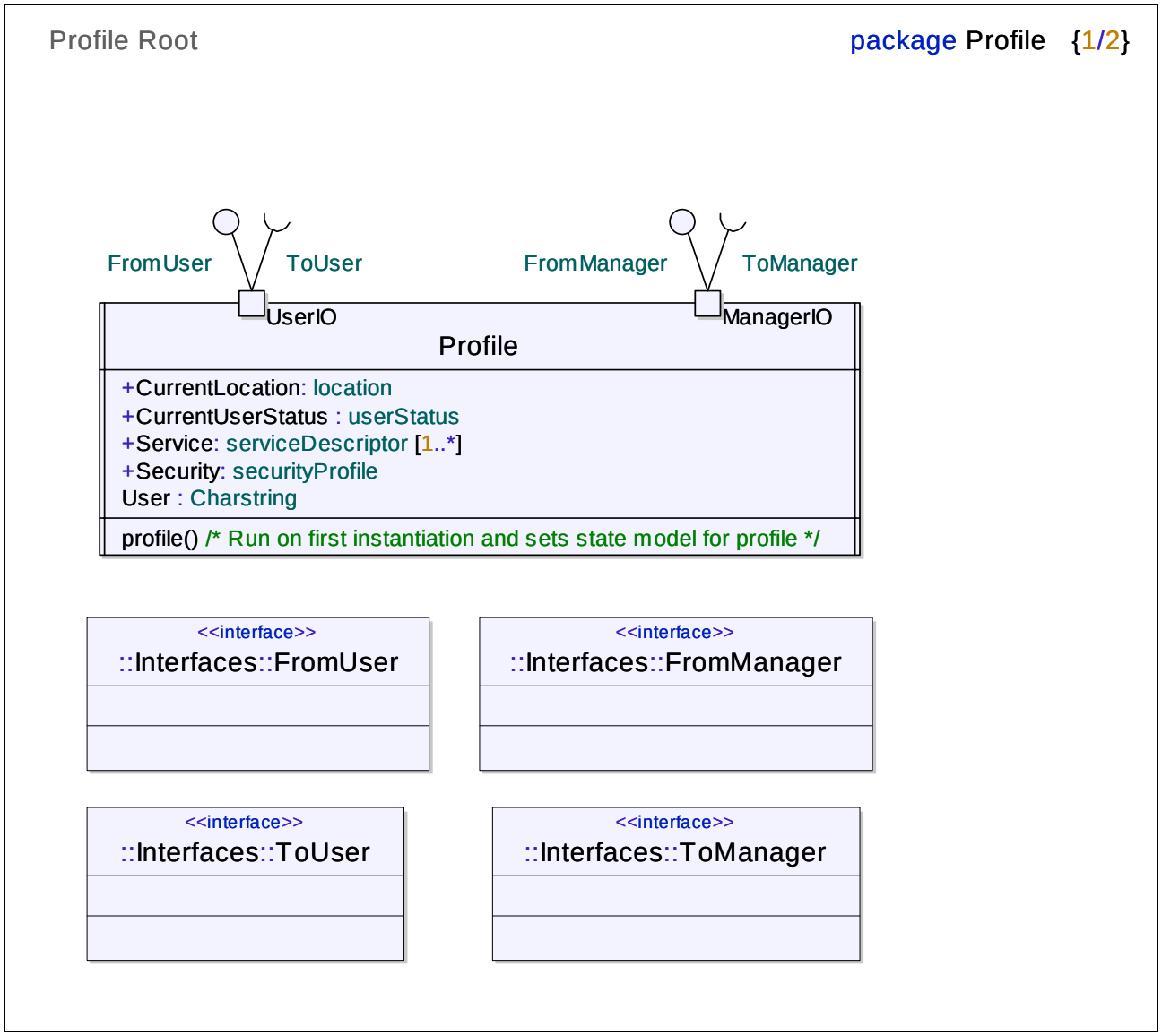


Figure 1: Class diagram showing the root class for profile, ports and interfaces

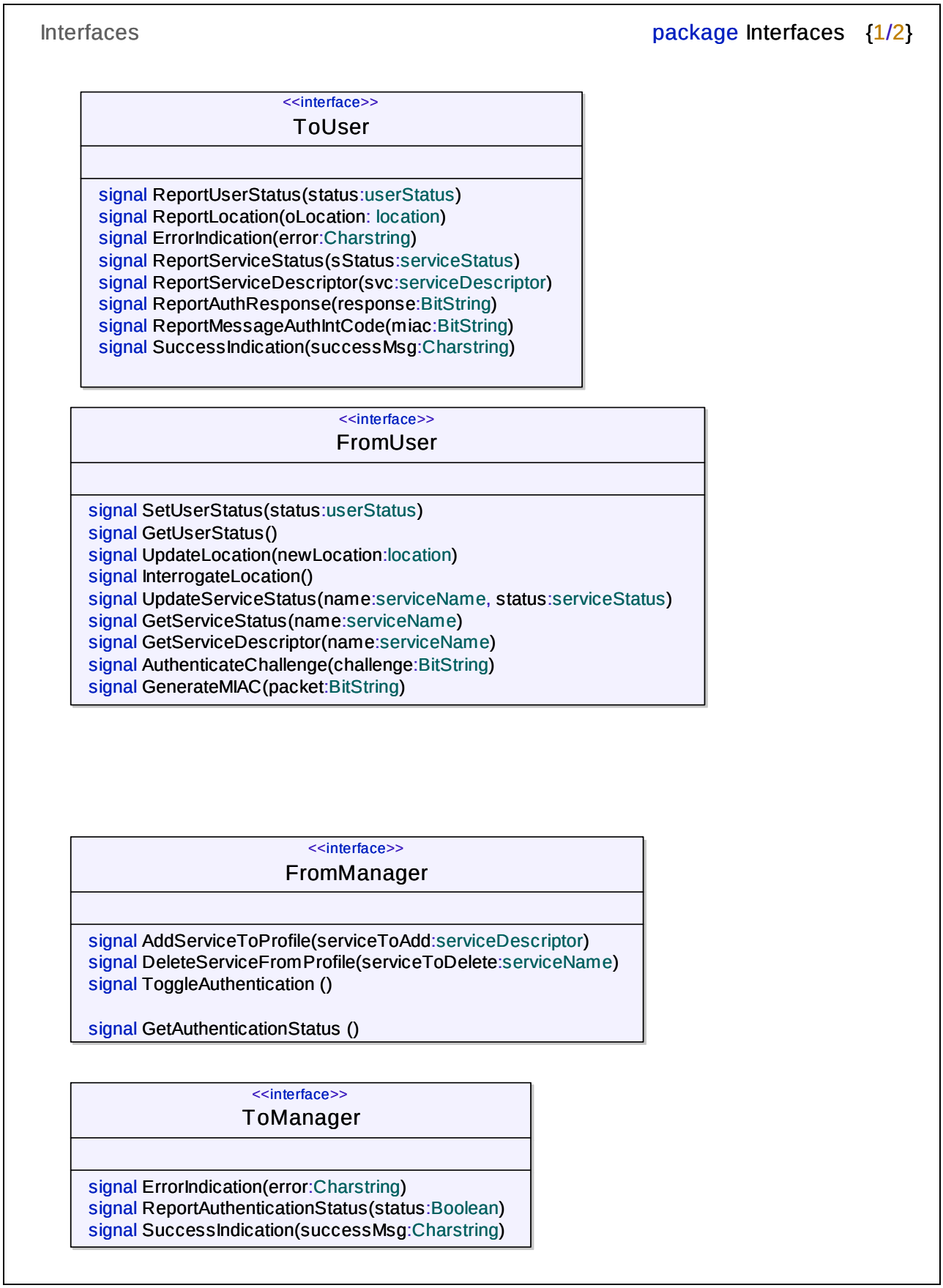


Figure 2: Class diagram showing the signals belonging to each interface

4.2 Data model

The data model derived from TR 101 878 [1] is shown in figures 3 and 4.

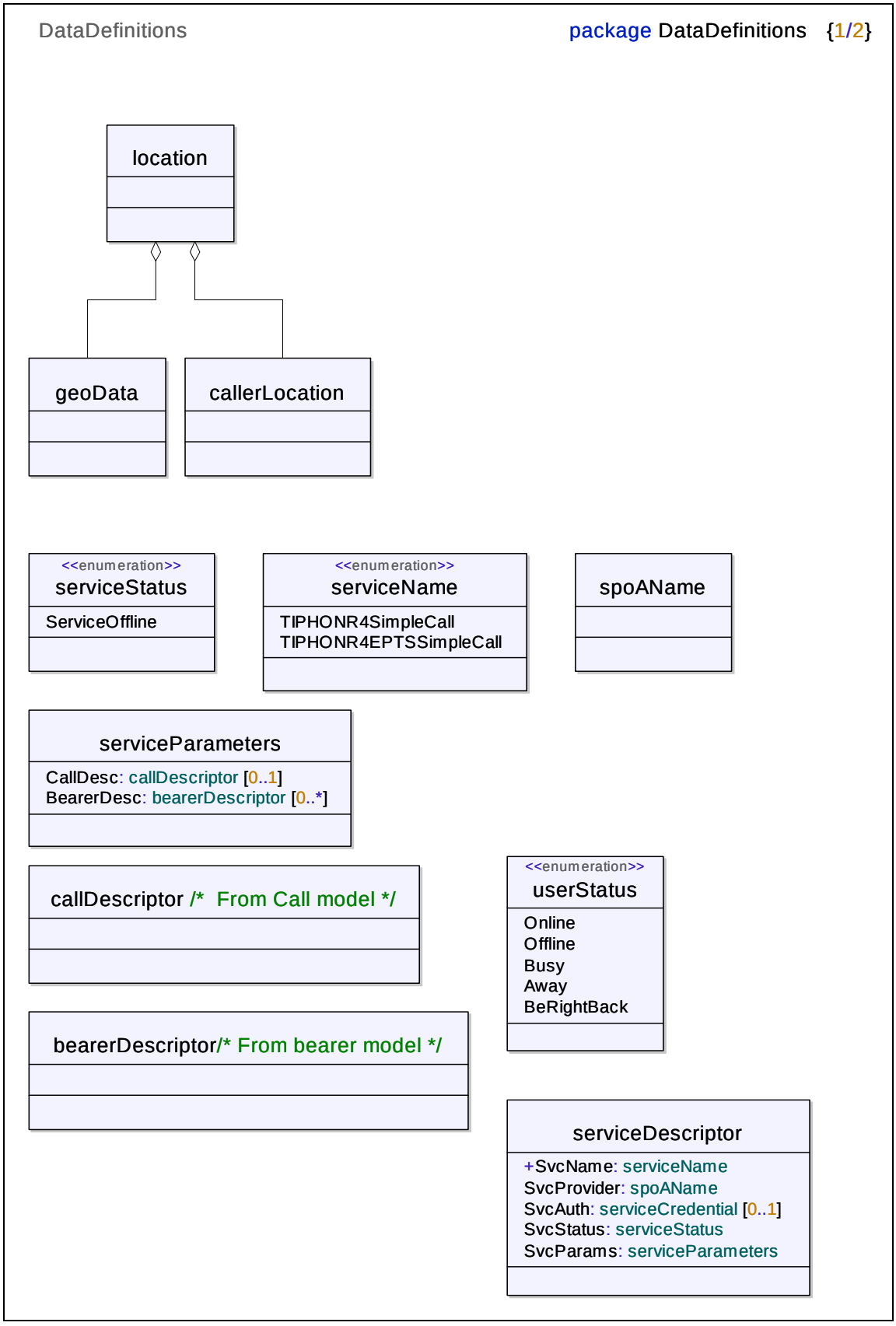


Figure 3: Data model for profile

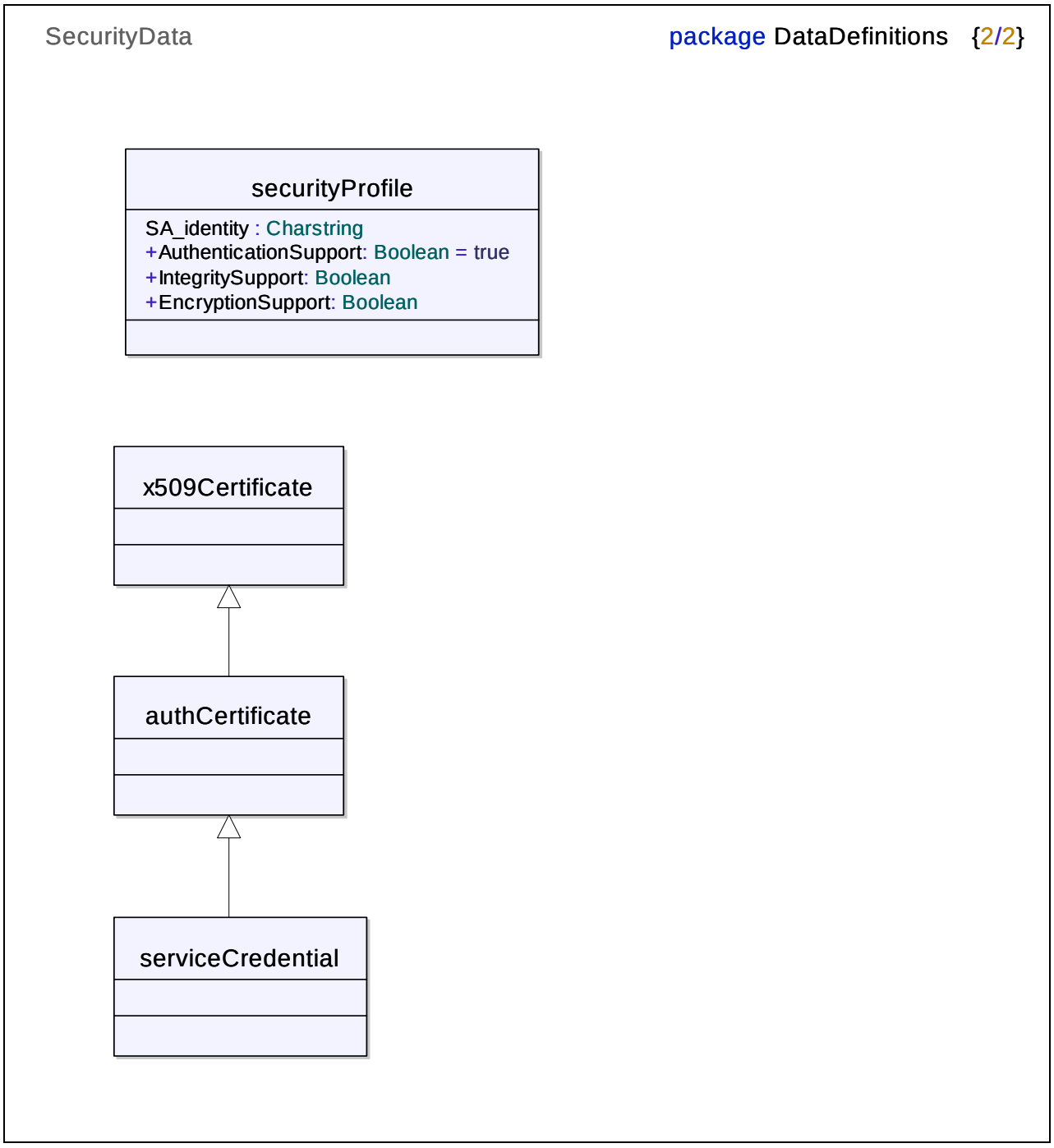


Figure 4: Data model for security elements in profile

4.3 Profile group service capabilities

The service capabilities act on the overall data model as defined in clause 4.2 and manipulate the data within the user profile. Some of the service capabilities defined in TR 101 878 [1] may be generated as applications of other root capabilities and where this is the case the invocation is shown as "pseudo-code" in Annex A.

4.3.1 Register

The *register* service capability can be generated from application of other profile group service capabilities. Invocation of the *register* service capability is shown as "pseudo-code" in Annex A.

4.3.2 Attach

The *attach* service capability can be generated from application of other profile group service capabilities. Invocation of the *attach* service capability is shown as "pseudo-code" in Annex A.

4.3.3 Authenticate

The *authenticate* service capability is described by a set of capabilities allowing Challenge-Response authentication and Message Authentication Integrity Code authentication forms. These are shown in figures 5, 6, 7, and 8.

The service capability supports symmetric and asymmetric keying methods, single and multi-pass protocols, and both unilateral and mutual authentication.

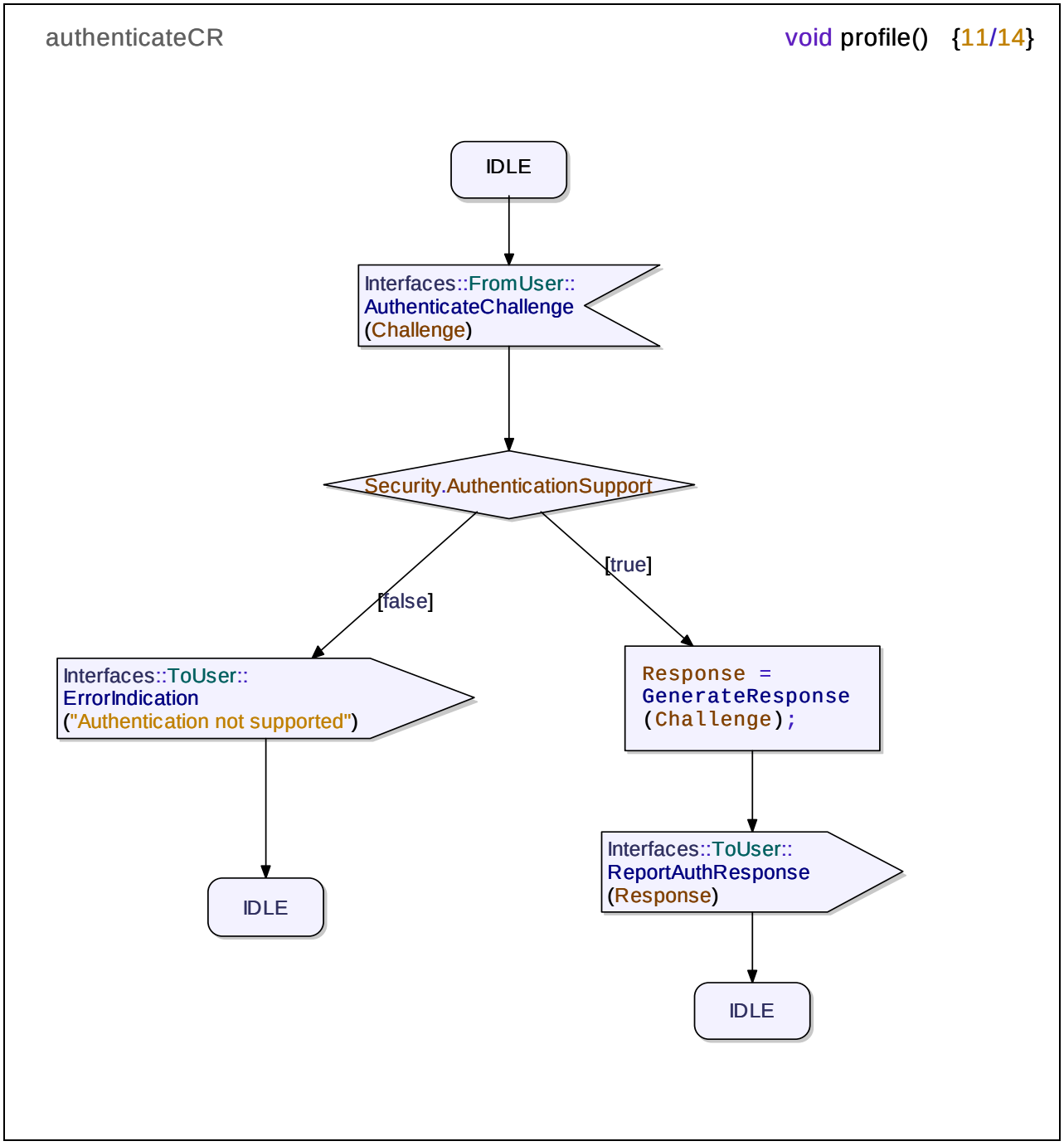


Figure 5: State chart diagram for authenticate challenge response service capability

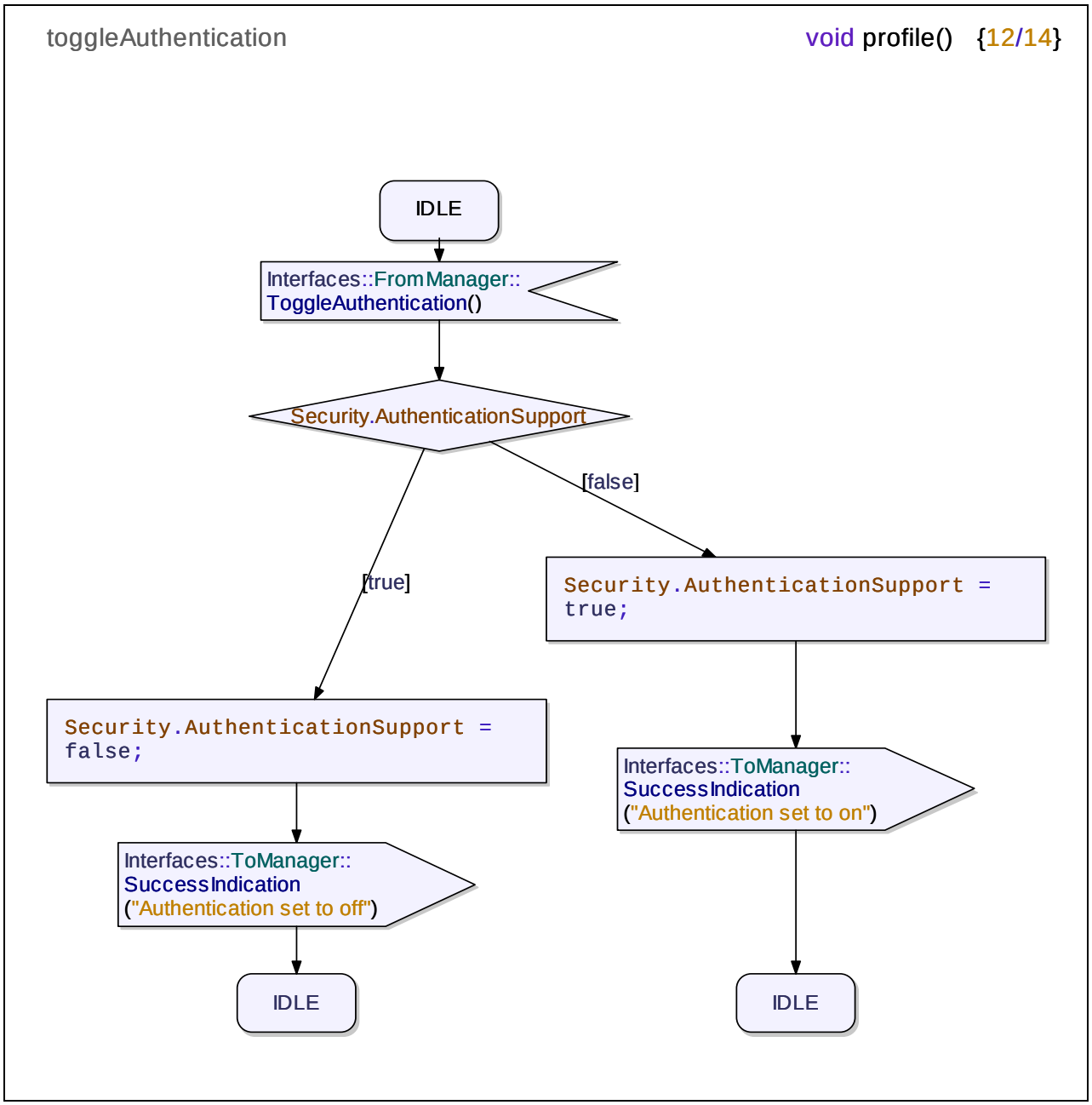


Figure 6: State chart diagram for authenticate toggle service capability

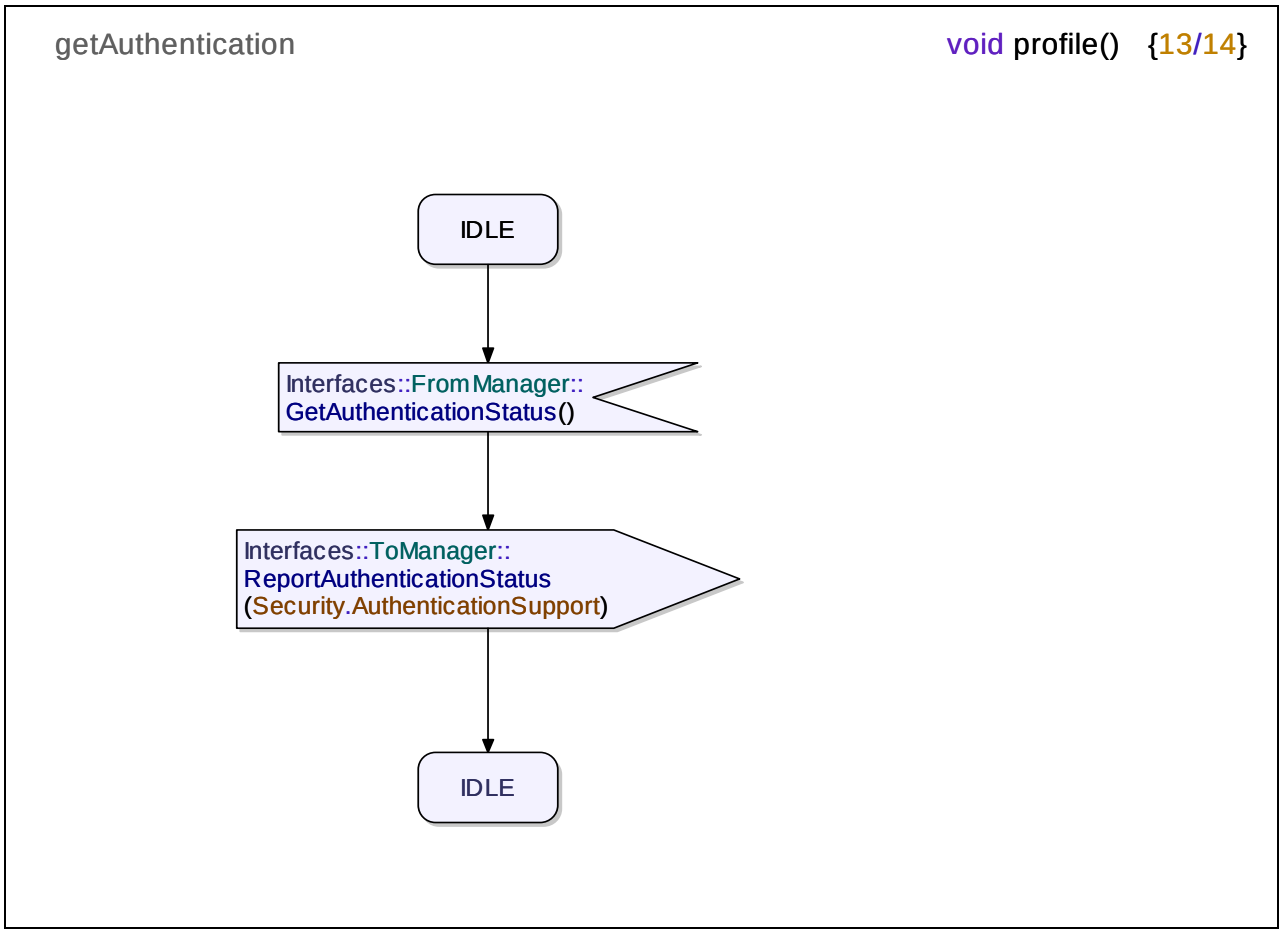


Figure 7: State chart diagram for get authenticate status service capability

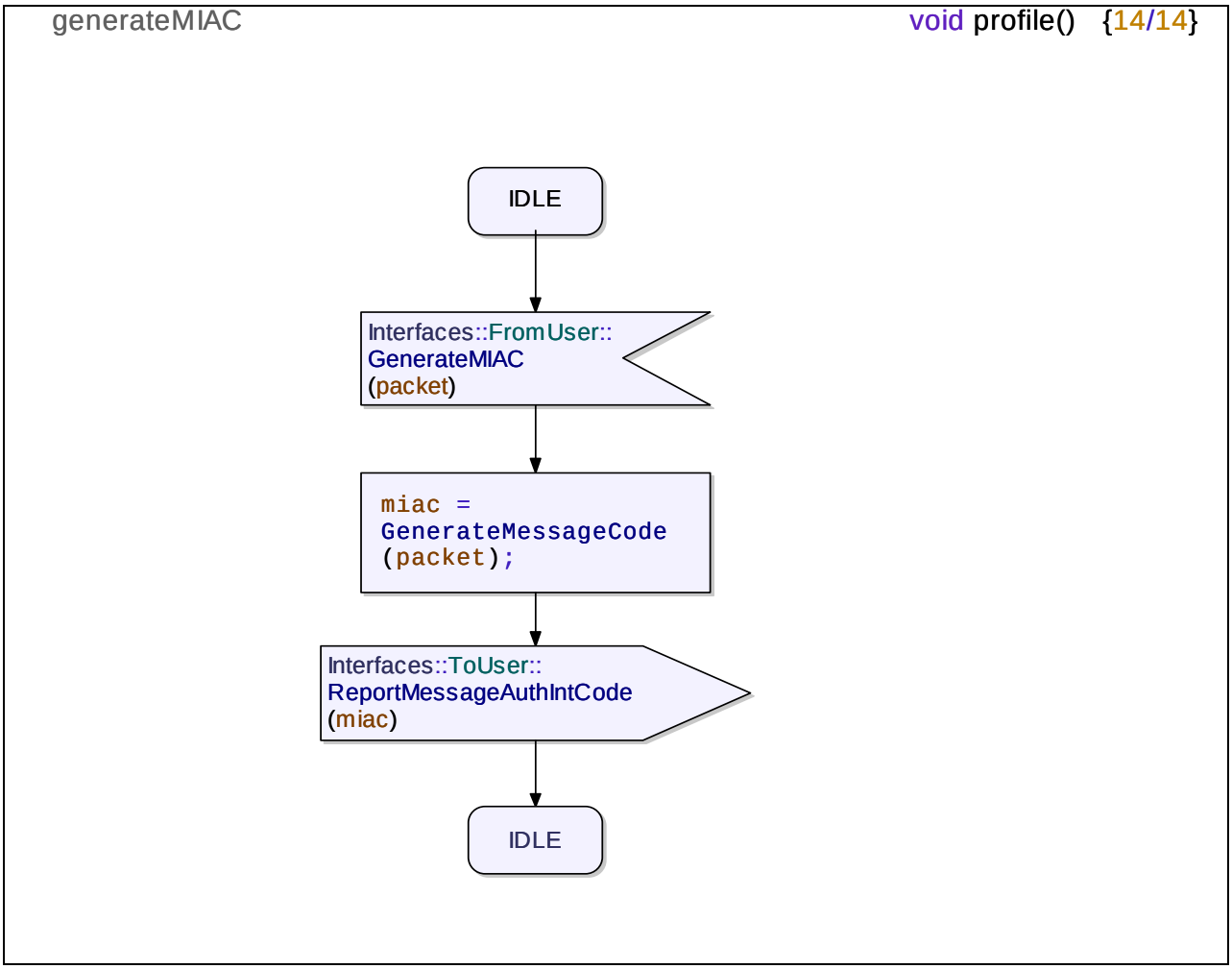


Figure 8: State chart diagram for generate message integrity and authentication code service capability

4.3.4 Get user status

The *get user status* service capability allows an authorized user to query the current status of a user (the requesting user or another). The service capability requirements are specified in figure 9.

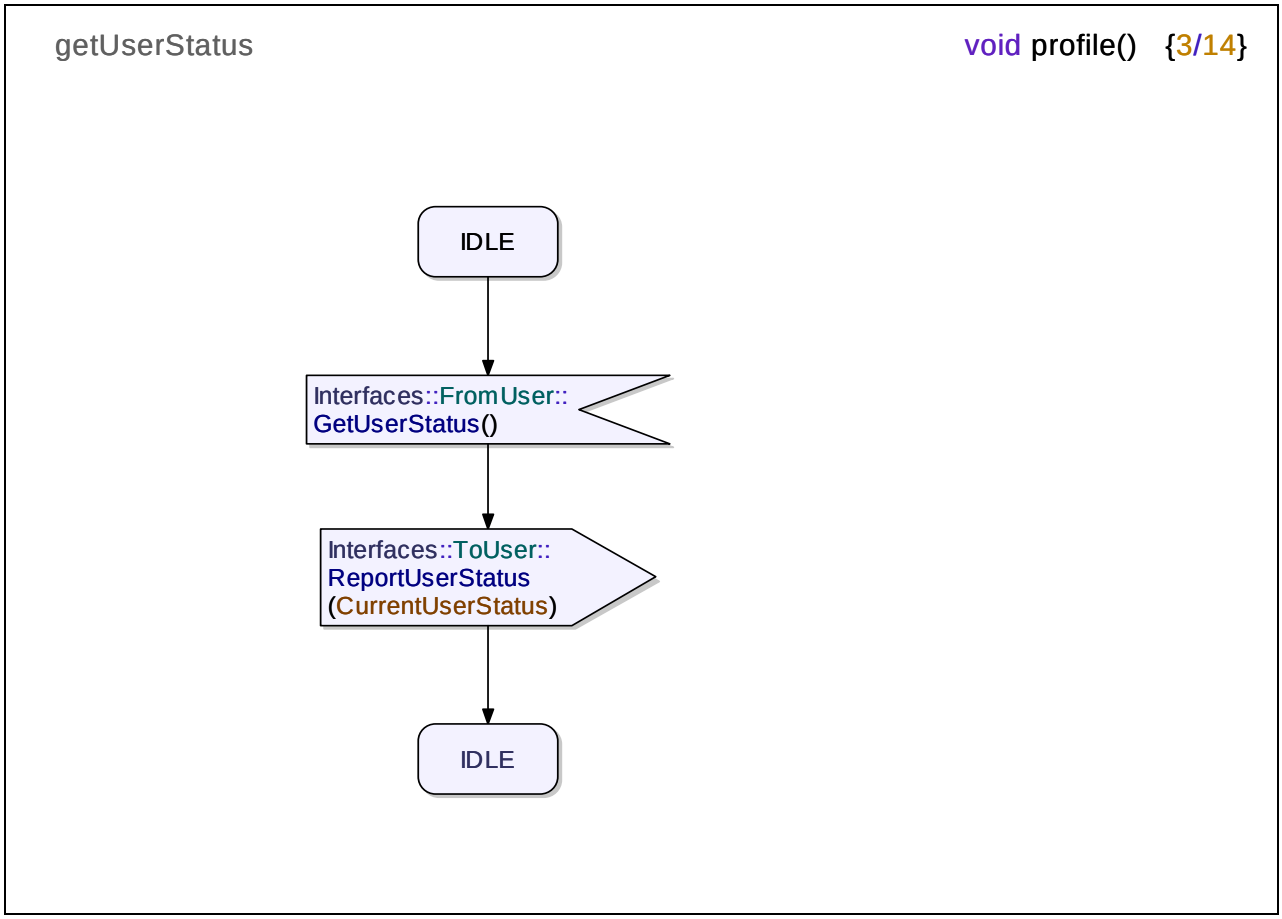


Figure 9: State chart diagram for get user status service capability

4.3.5 Deregister

The *deregister* service capability can be generated from application of other profile group service capabilities. Invocation of the *deregister* service capability is shown as "pseudo-code" in Annex A.

4.3.6 Transfer

The *transfer* service is external to the profile capability and is not defined further in the present document.

4.3.7 Authorize

The *authorize* service capability is achieved by checking that the service exists in the profile. As such there is no need for an explicit authorize service capability.

4.3.8 Set user status

The *set user status* service capability allows an authorized user to set the current status of a user. The service capability requirements are specified in figure 10.

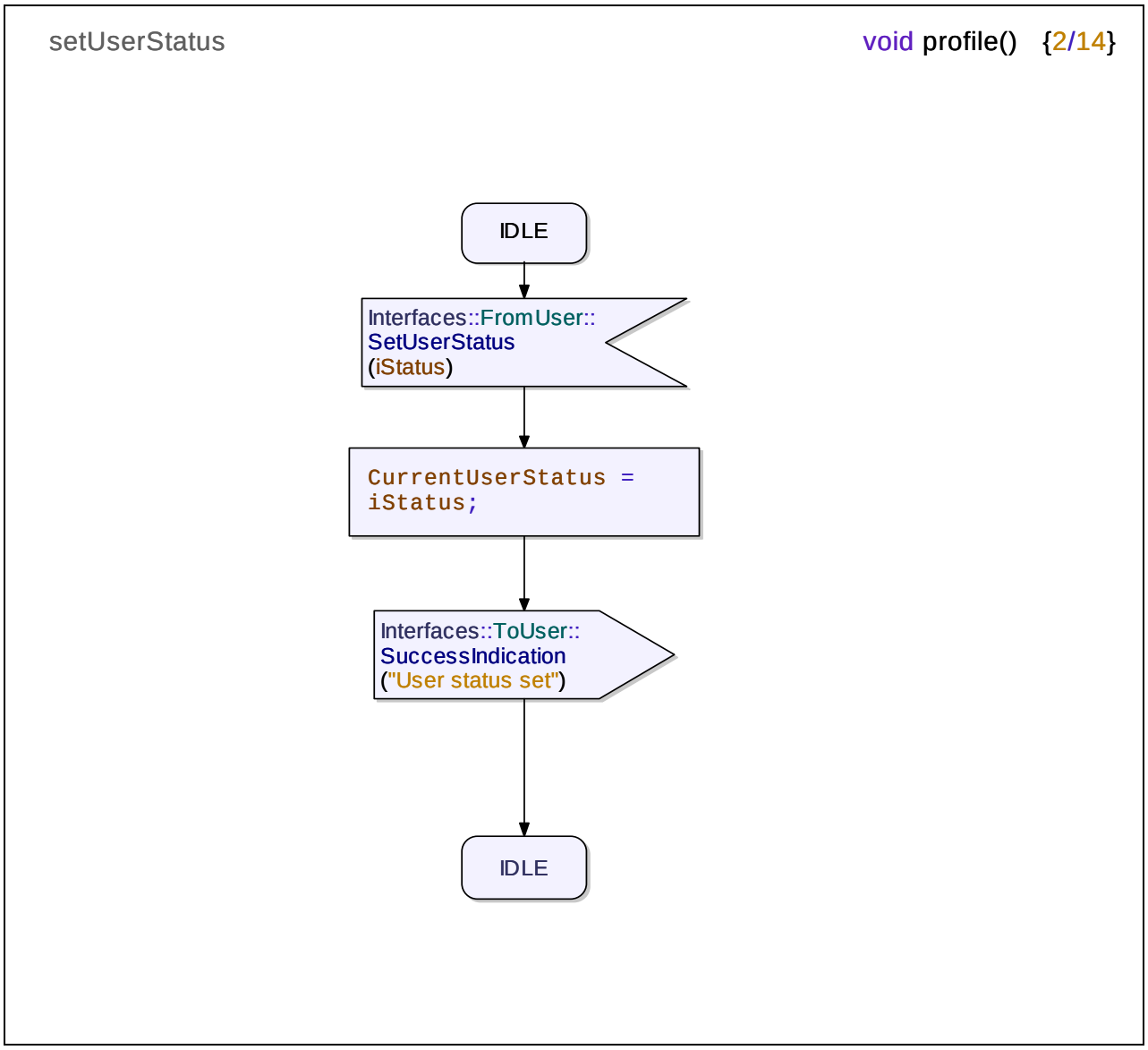


Figure 10: State chart diagram for set user status service capability

4.3.9 Interrogate location

The *interrogate location* service capability allows an authorized user to query the location of a user. The behaviour of the service capability is shown in figure 11.

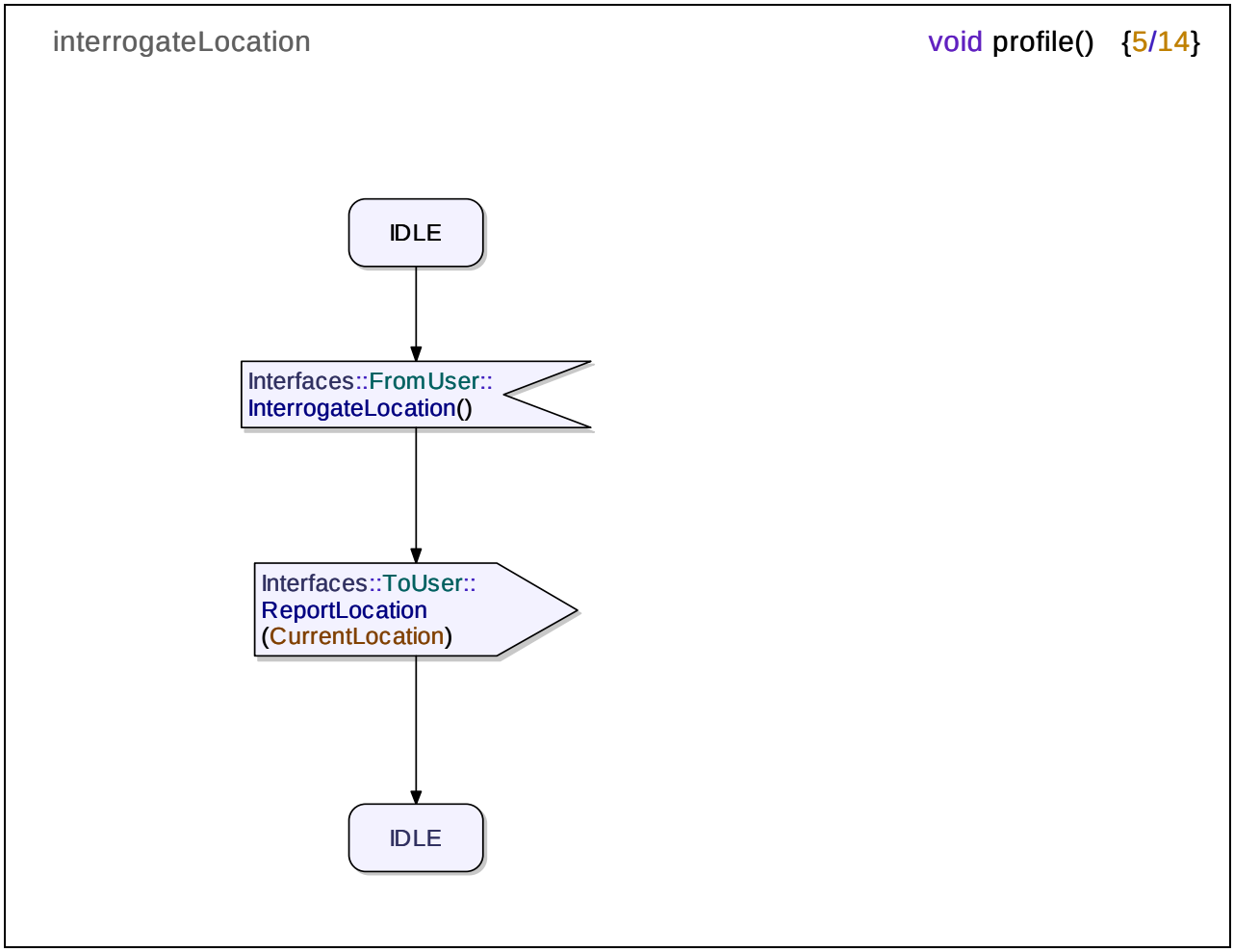


Figure 11: State chart diagram for interrogate location service capability

4.3.10 Update location

The *update location* service capability allows an authorized user to set the location of a user.figure 12 shows the behaviour of this service capability.

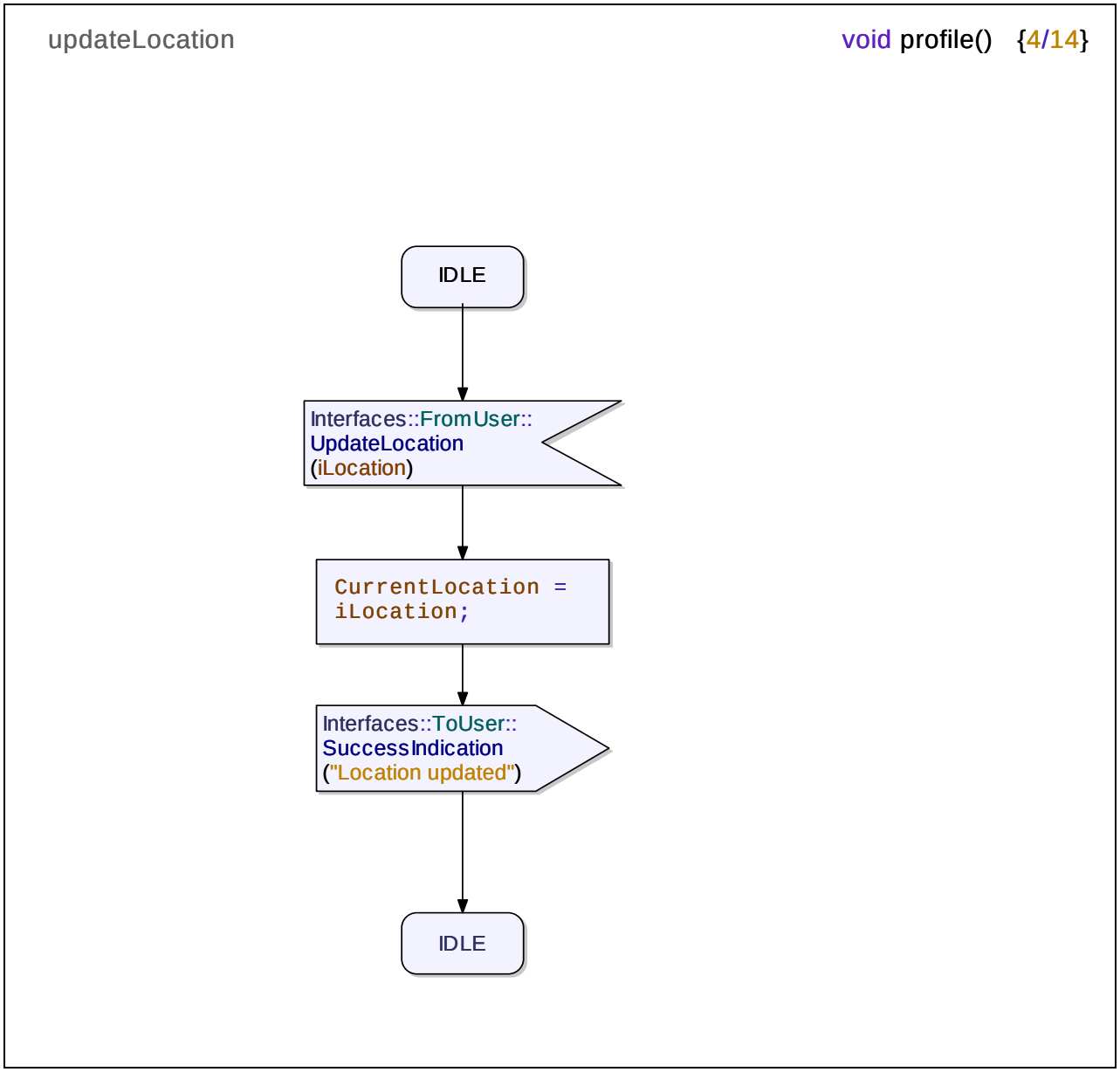


Figure 12: State chart diagram for update location service capability

4.3.11 Update service status

The *update service status* service capability modifies the service status where the service status may take values including available and unavailable. In figure 13 the *update service status* service capability behaviour is specified.

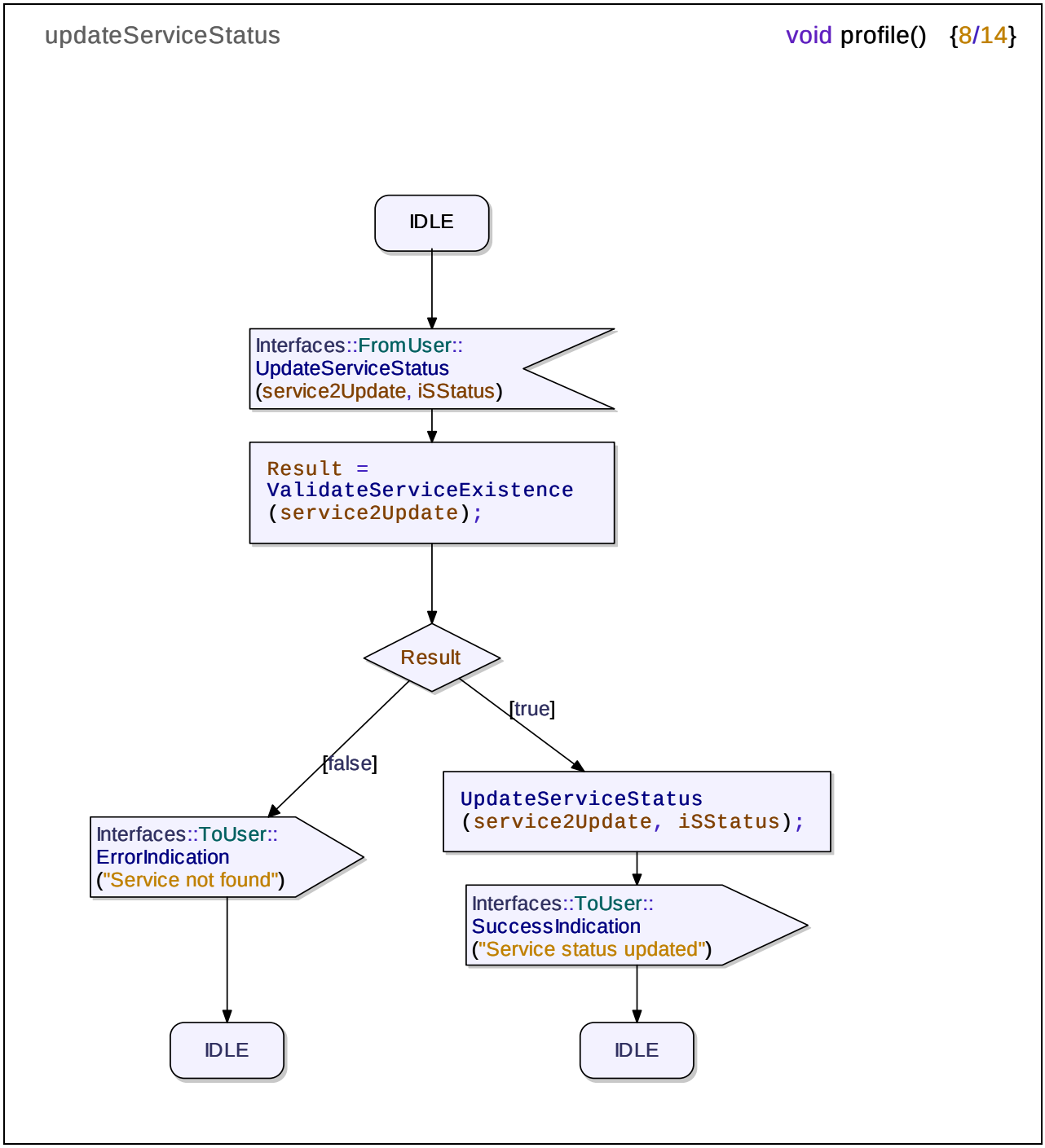


Figure 13: State chart diagram for update service status service capability

4.3.12 Add service to profile

The *add service to profile* service capability adds a service to the profile of the user. The behaviour of the service capability is shown in figure 14.

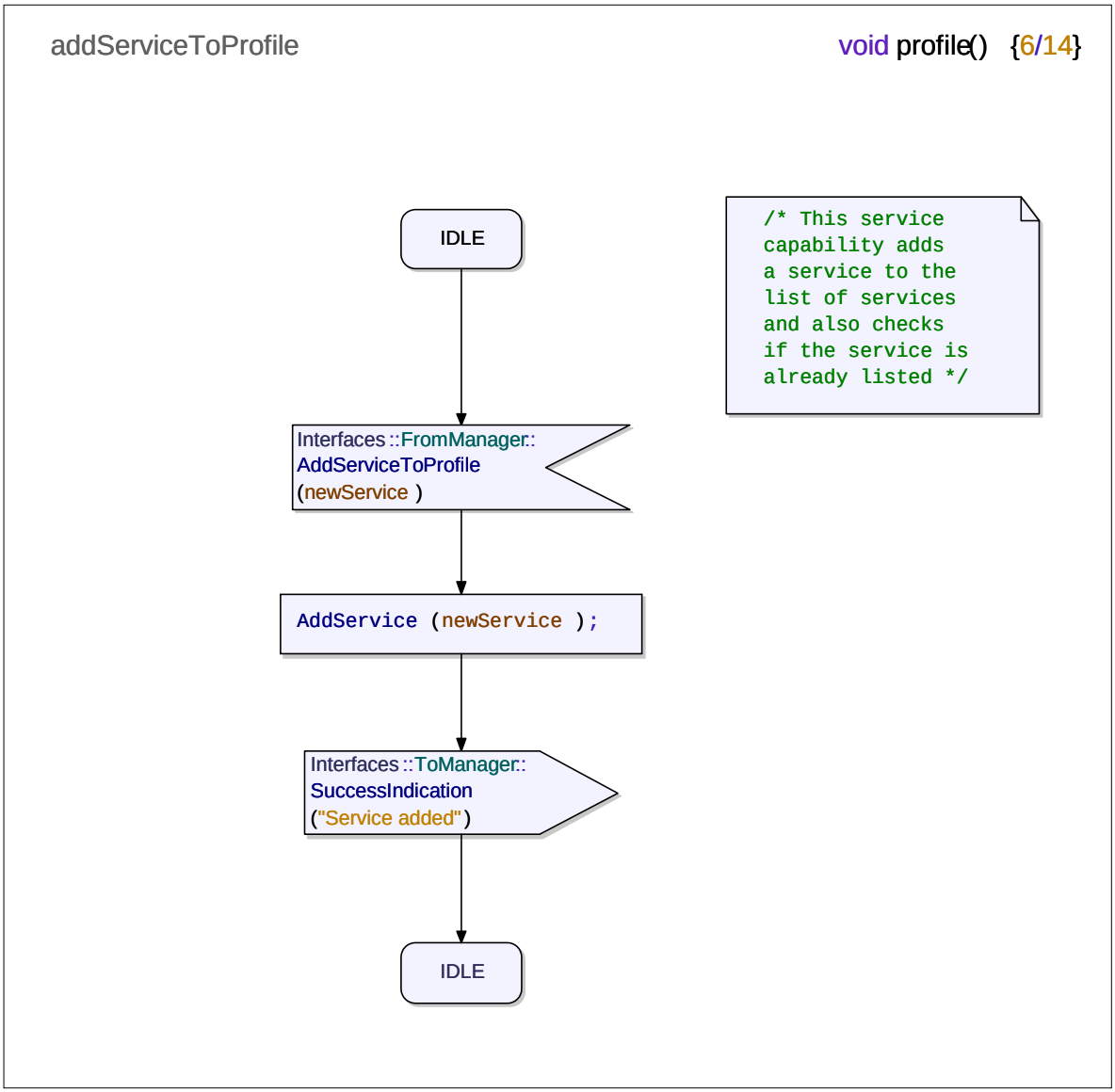


Figure 14: State chart diagram for add service to profile service capability

4.3.13 Remove service from profile

The *remove service from profile* service capability removes a service from the profile of the user. A specification of the behaviour of this service capability is shown in figure 15.

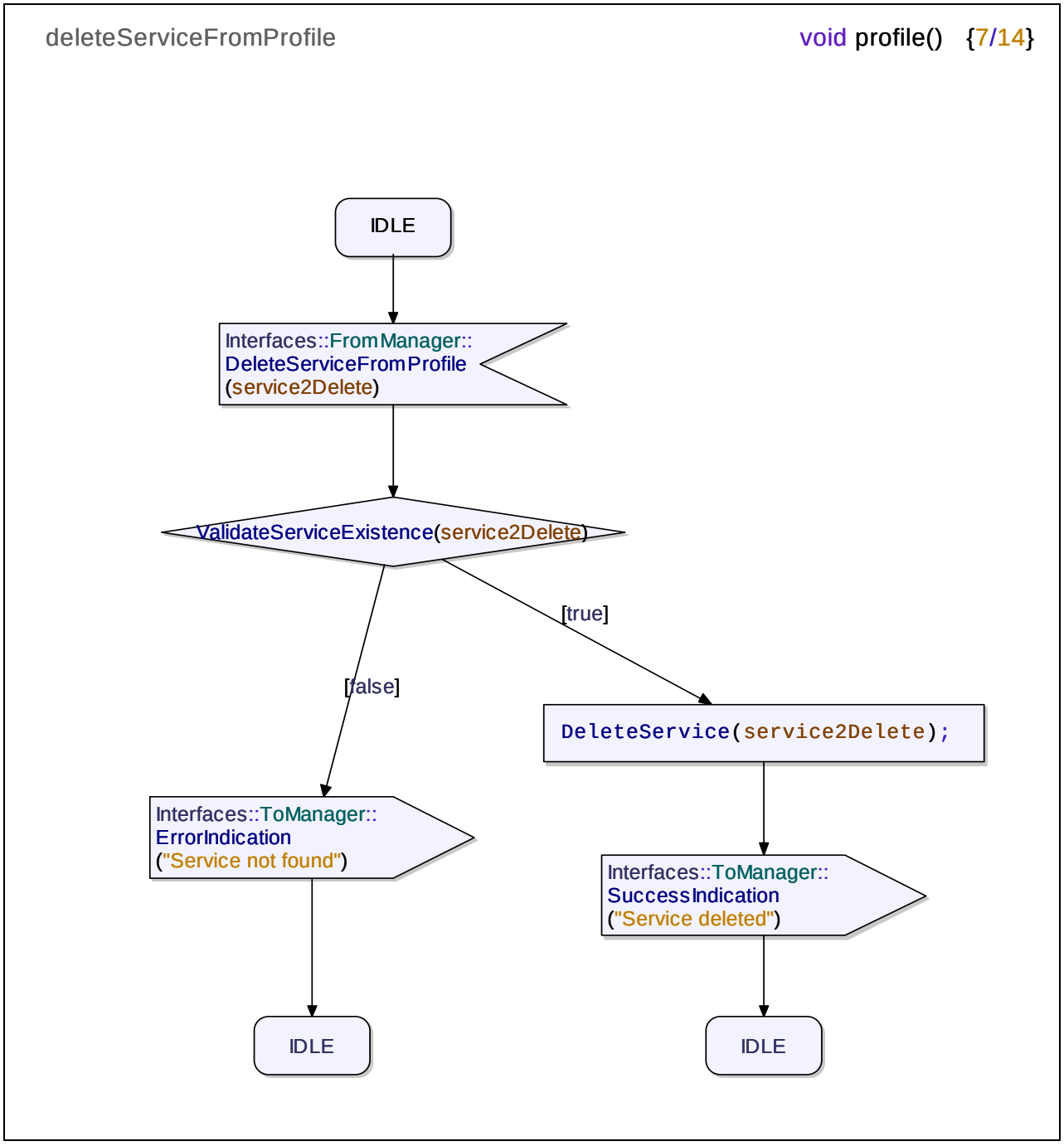


Figure 15: State chart diagram for delete service from profile service capability

4.3.14 Get service status

The *get service status* service capability allows a user to query the current status of a service. The behaviour of this service capability is shown in figure 16.

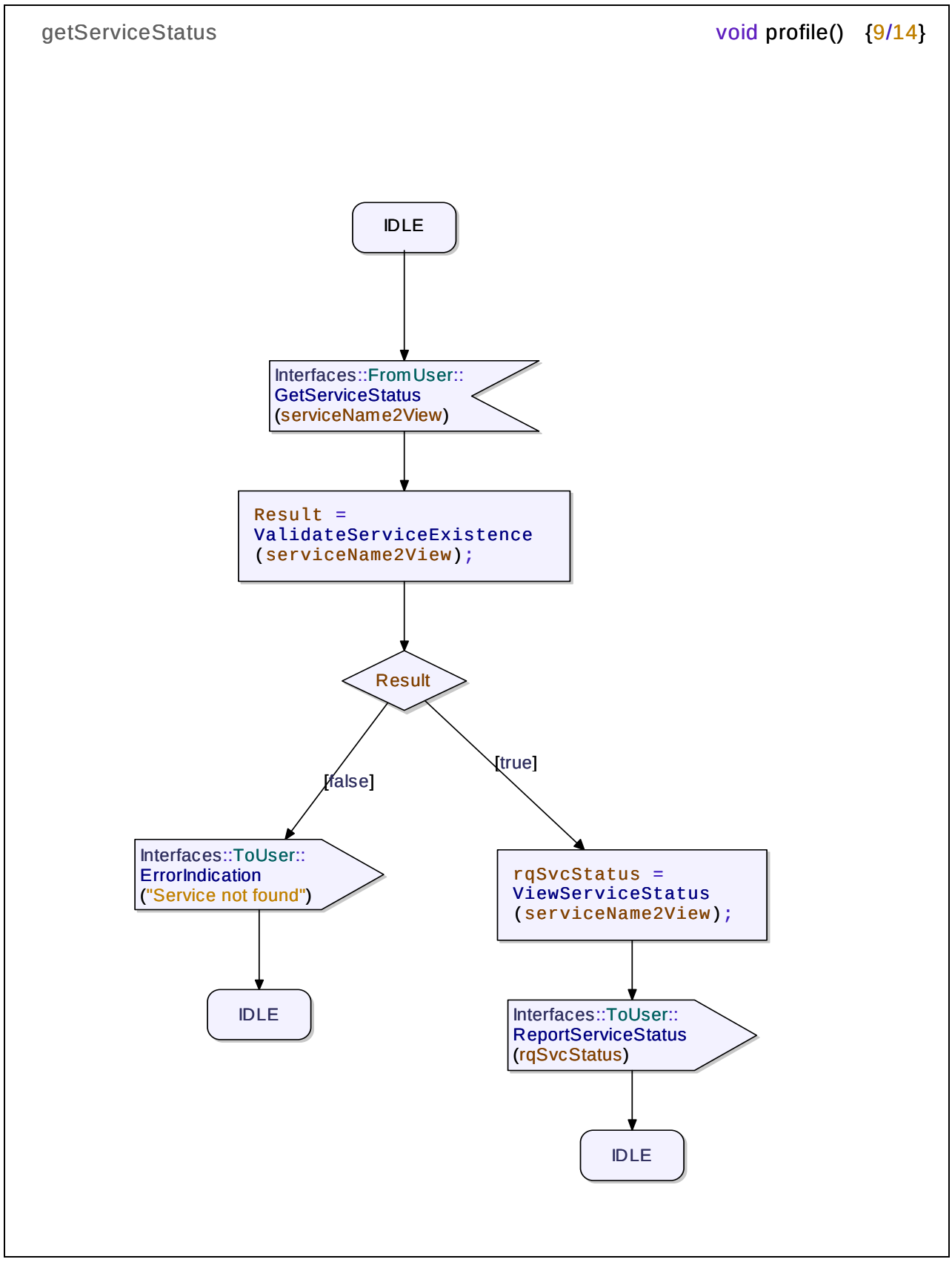


Figure 16: State chart diagram for get service status service capability

4.3.15 Get service descriptor

The *get service status* service capability allows a user to retrieve service descriptor information. Specification of the behaviour of this service capability is shown in figure 17.

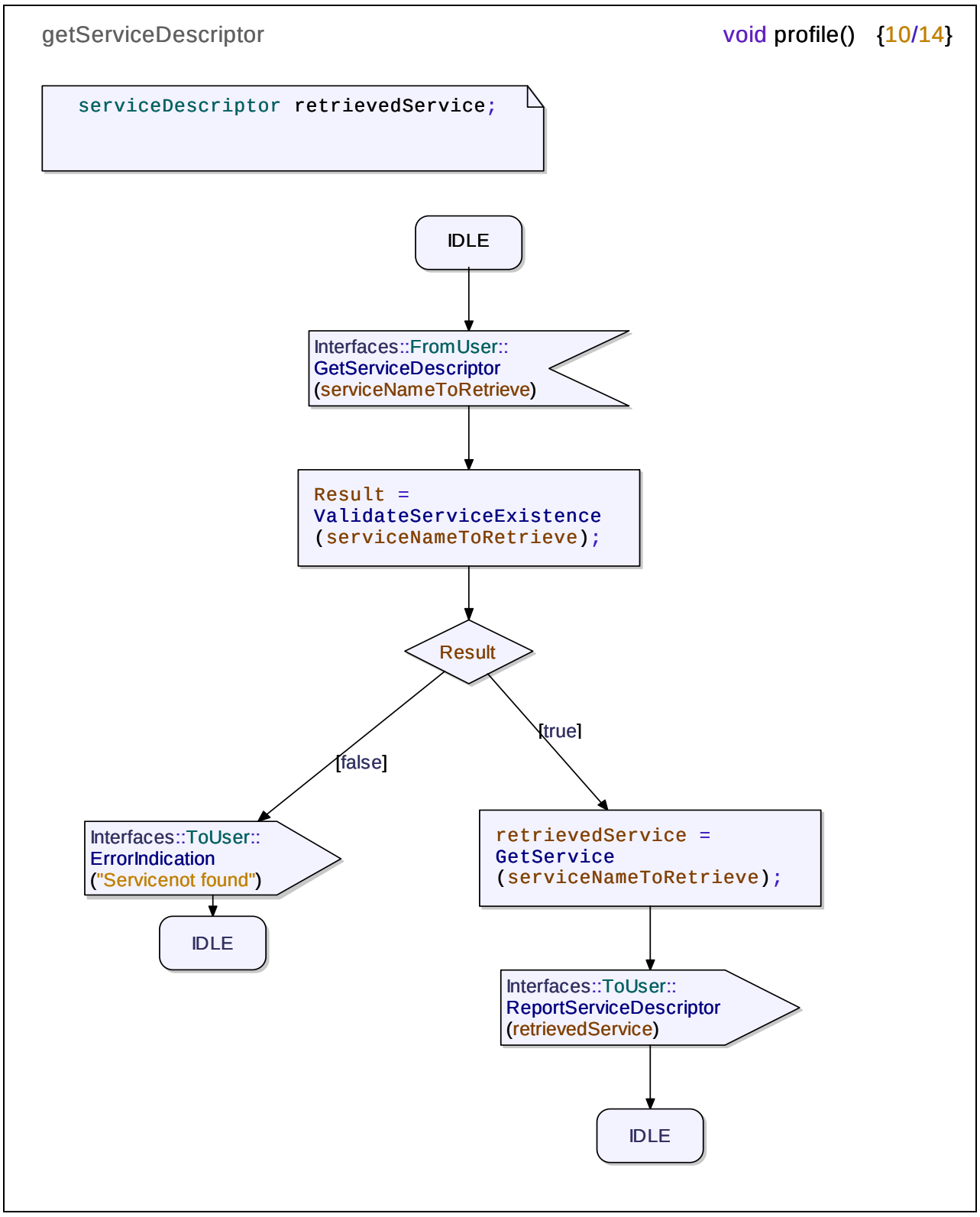


Figure 17: State chart diagram for get service descriptor service capability

4.4 Typical architecture

Figure 18 shows a typical architectural arrangement for an object based on the Profile class.

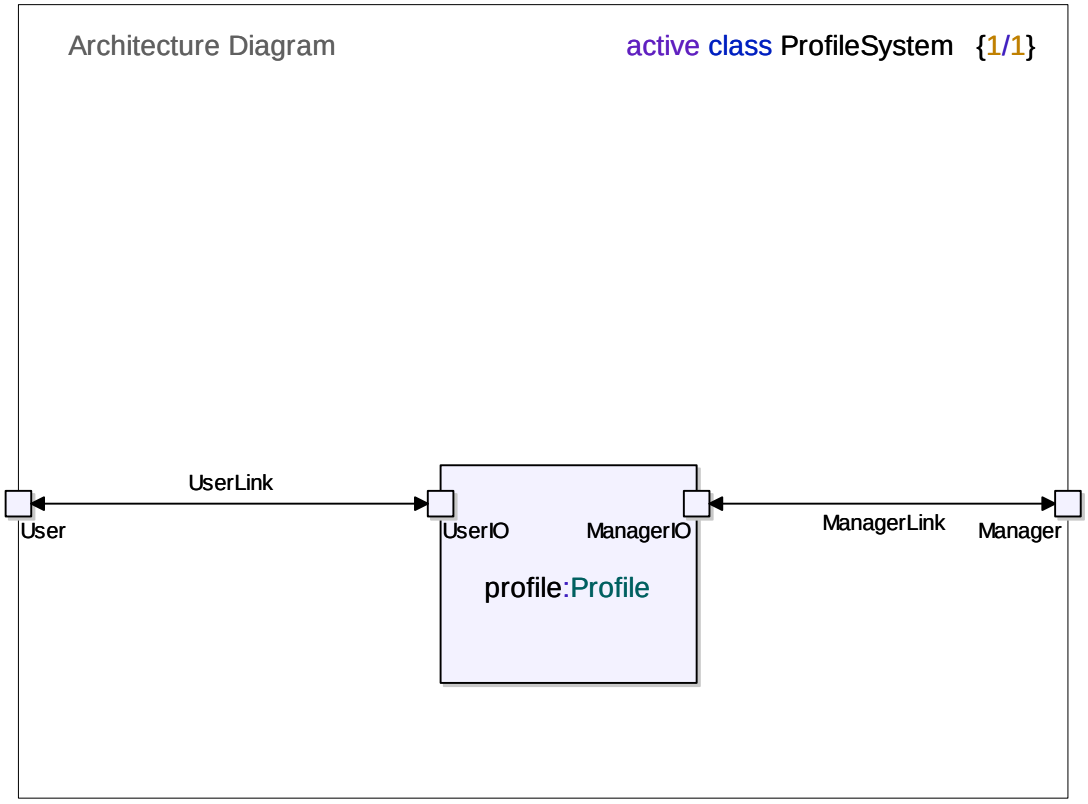


Figure 18: Typical profile service architecture

5 Call group

5.1 Introduction

The Call group of service capabilities is defined in TR 101 878 [1].

5.2 Call group

The TIPHON call class is specialized in three call classes defining the call group service capabilities in an originating, intermediate or destination domain of a call. The three call classes are shown in figures 19, 20, and 21.

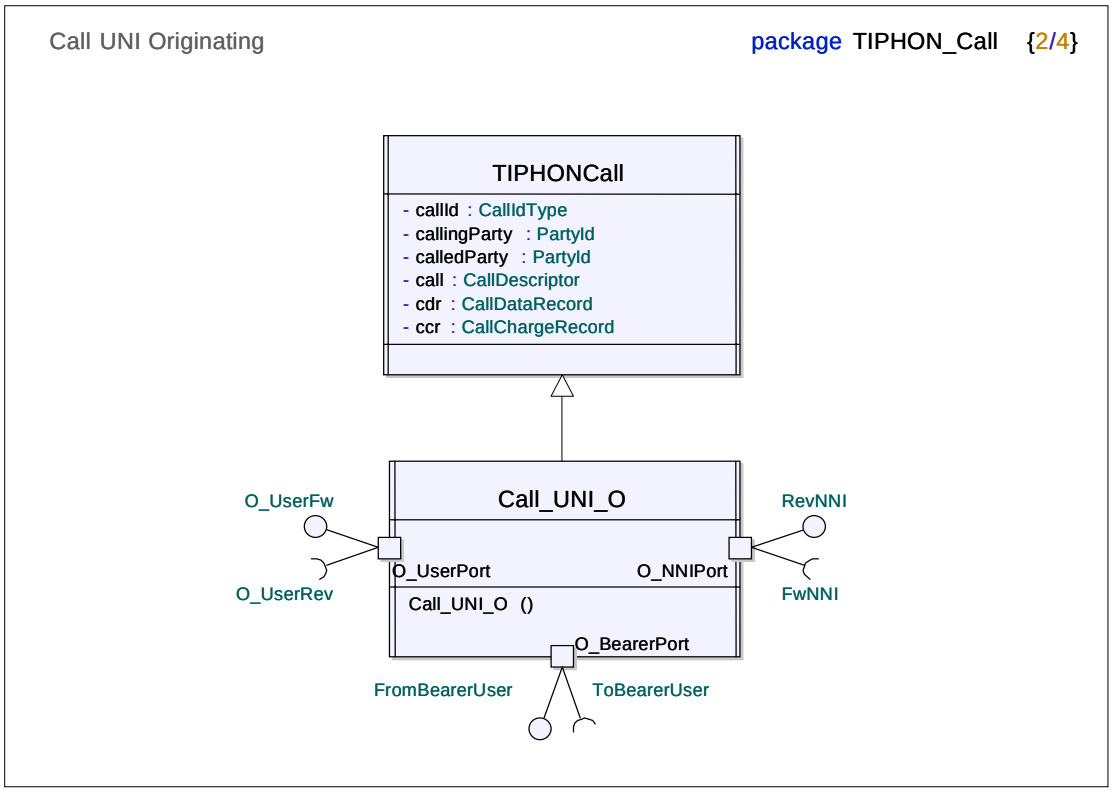


Figure 19: Originating domain call class with interfaces

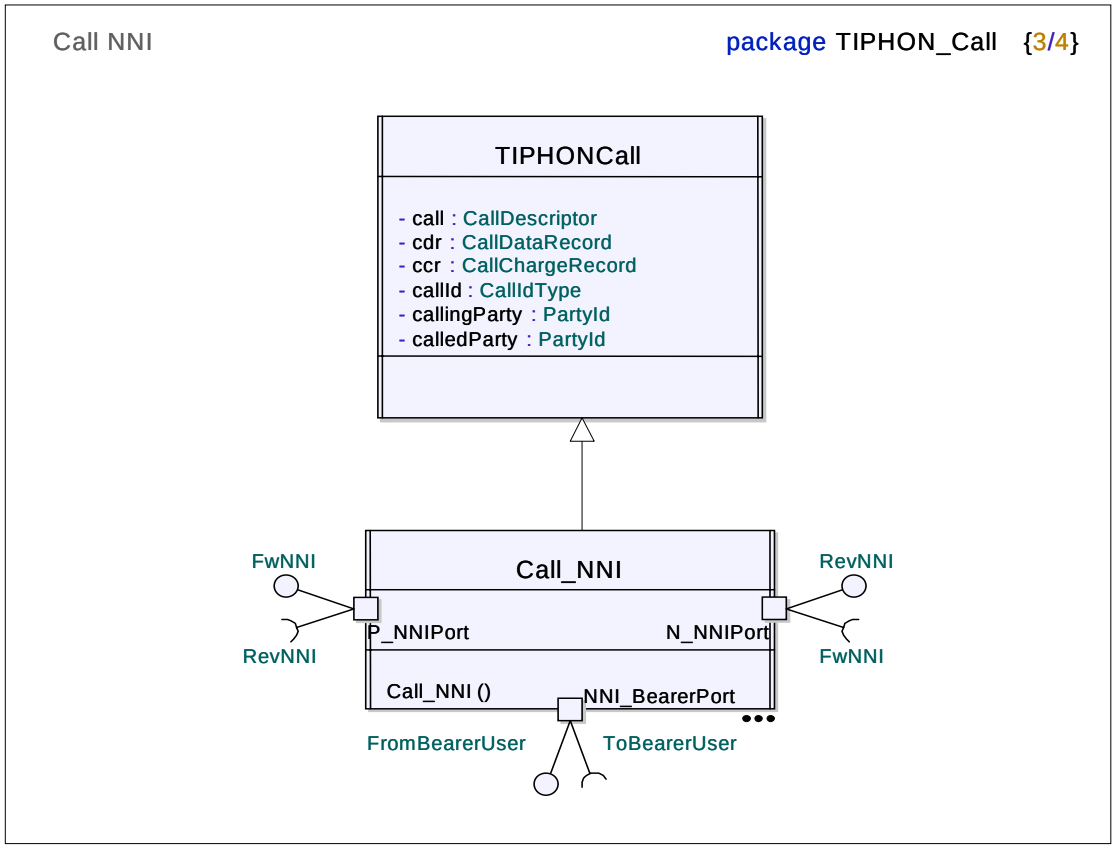


Figure 20: Intermediate domain call class with interfaces

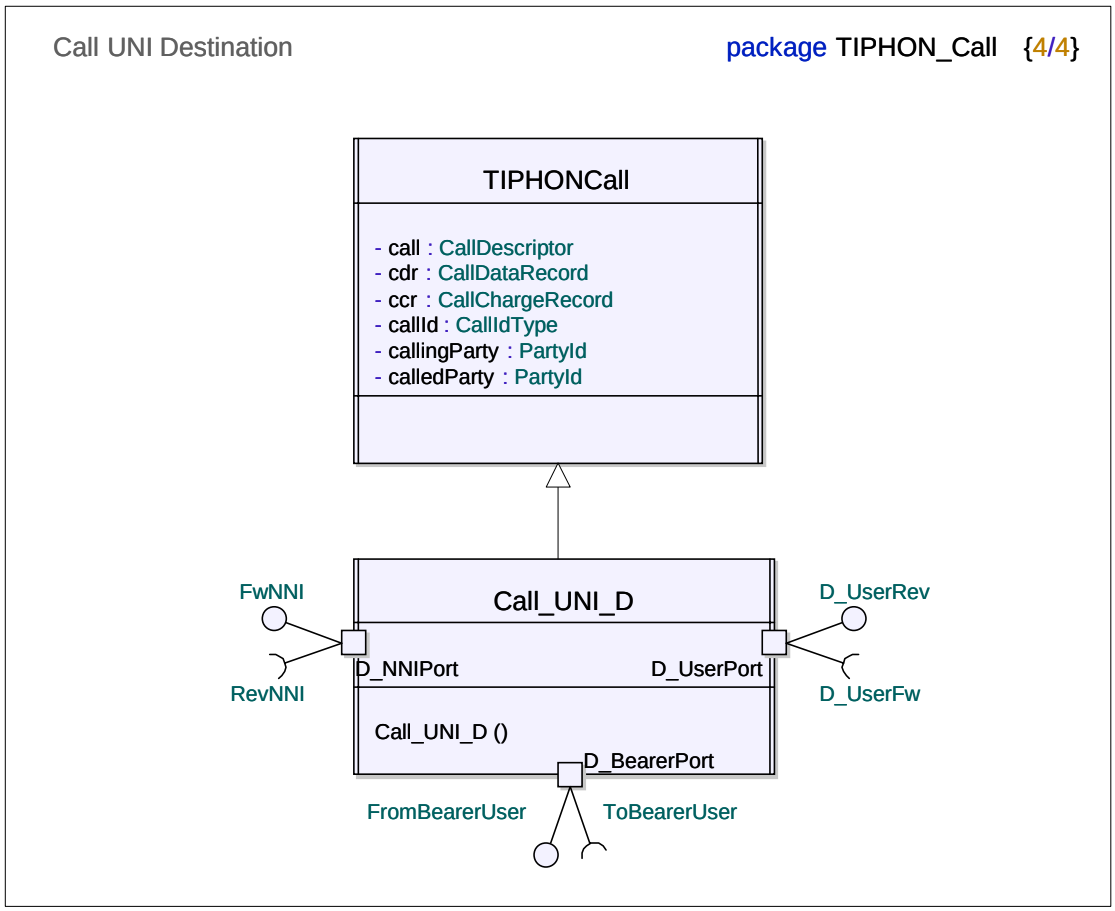


Figure 21: Destination domain call class with interfaces

The service capabilities available at the defined interfaces in the specialized call classes are shown as signals in figures 22 and 23. The interface definition for the interfaces at bearer ports are shown in the Bearer section.

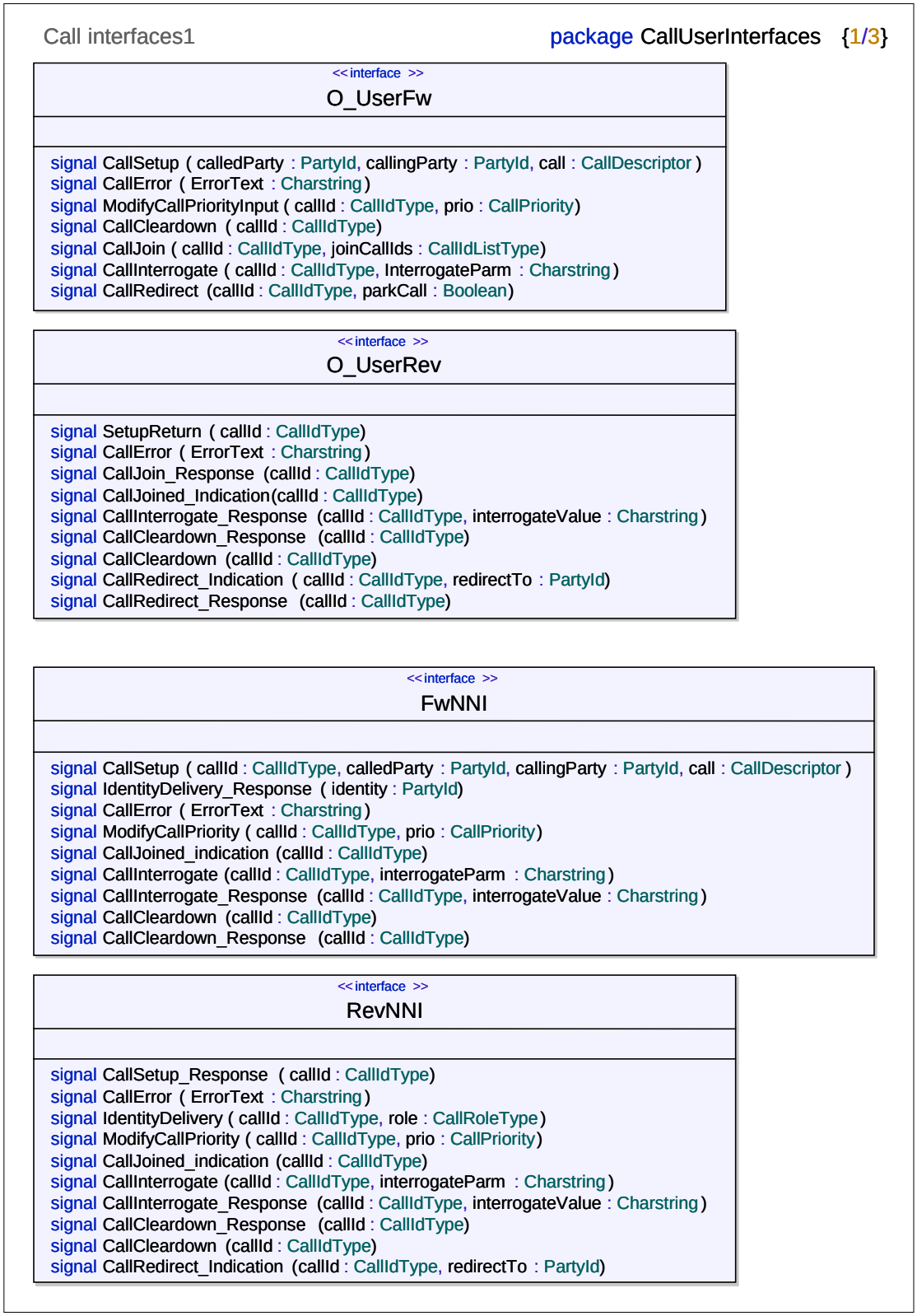


Figure 22: Interfaces of the call classes with call group service capabilities as signals

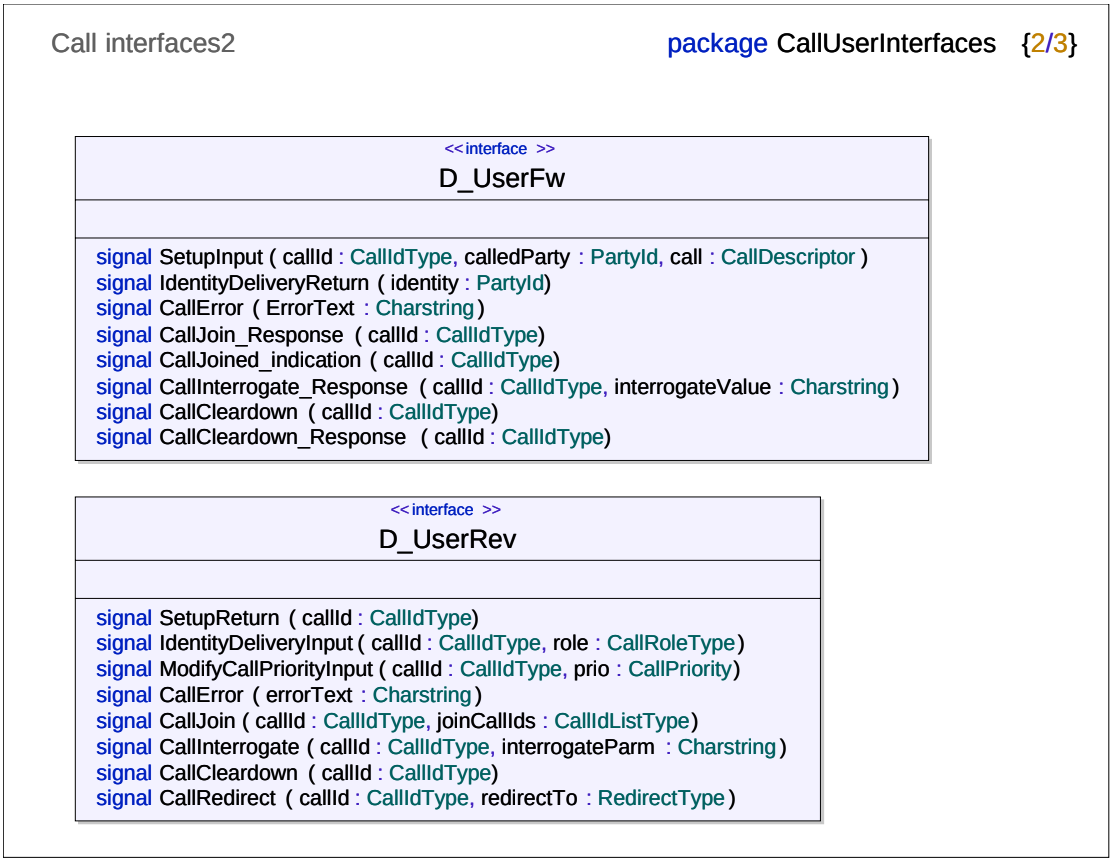


Figure 23: Interfaces of the destination call class with service capabilities as signals

5.3 Data definitions

The data model derived from TR 101 878 [1] is shown in figures 24 and 25.

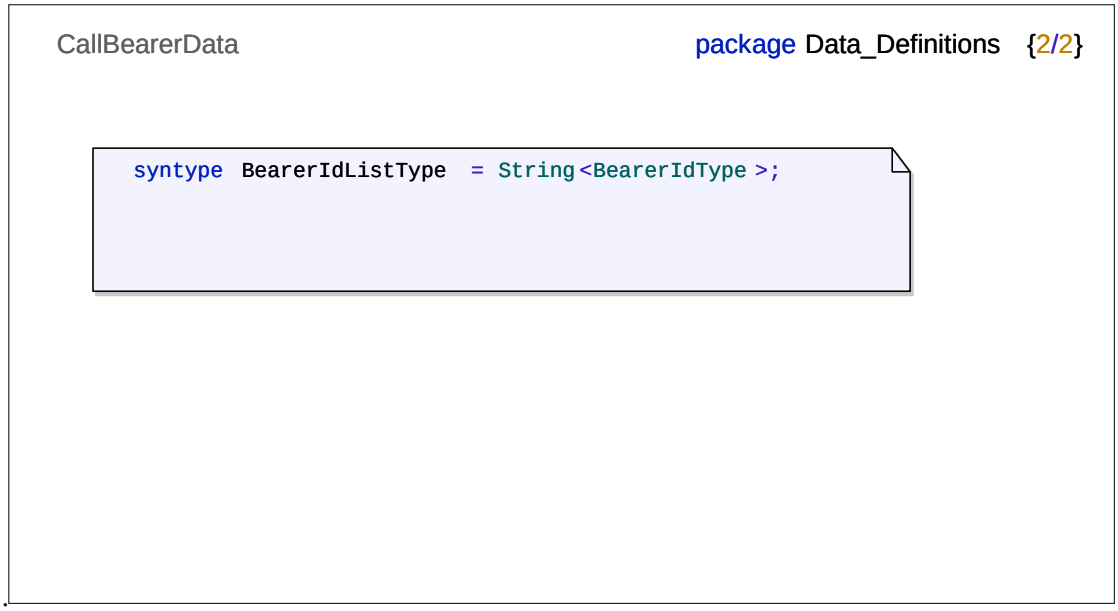


Figure 24: Call group data definitions 1

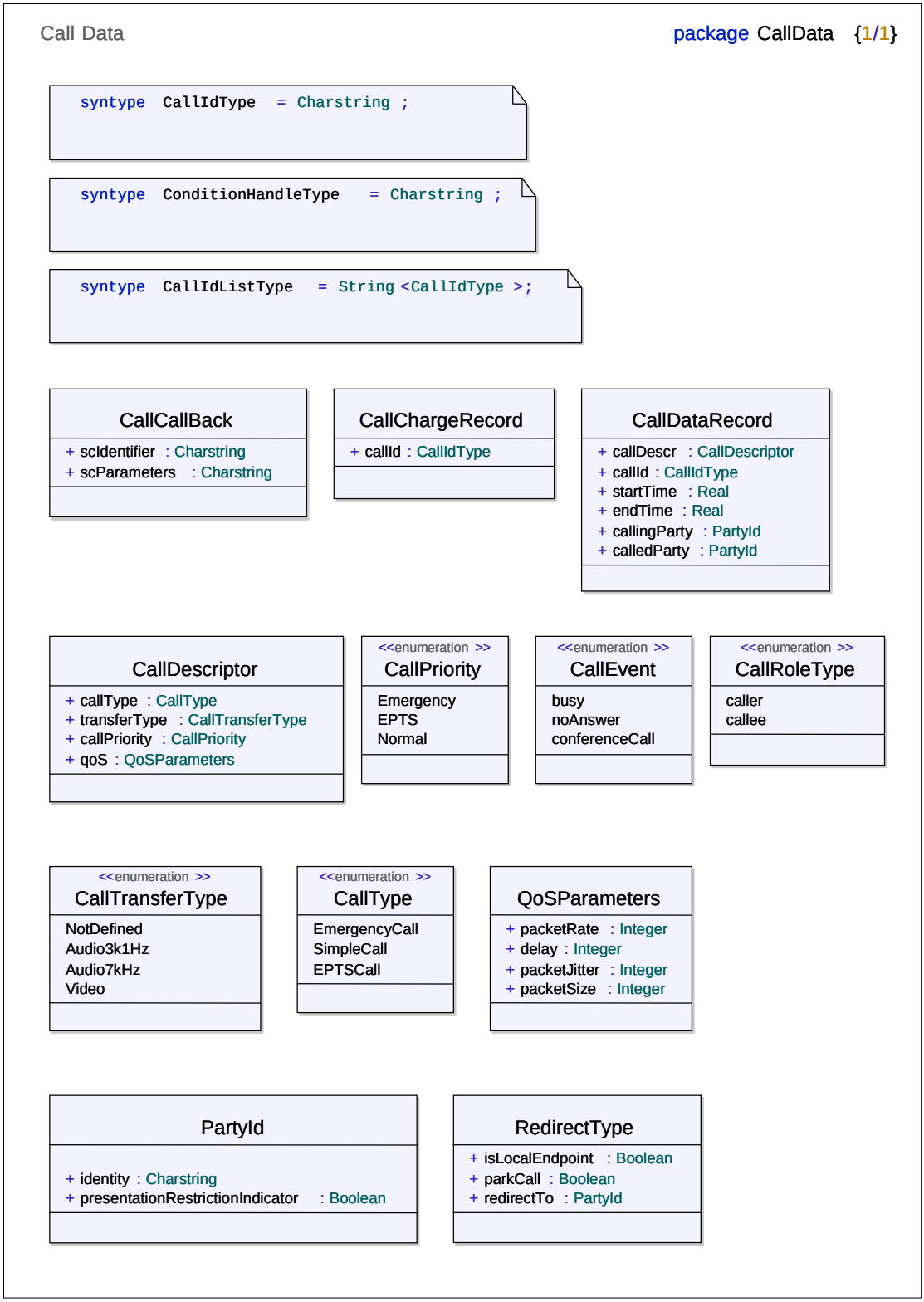


Figure 25: Call group data definitions 2

5.4 Call group service capabilities

The following service capabilities belongs to the specialized call classes:

- setup a call;
- clear down a call;
- deliver calling party identity information;
- redirect a call;
- set the call priority;
- join a call;
- interrogate a call.

The call group the service capabilities definitions are specific to the domain they belong. The call group service capabilities in an originating, intermediate, and destination domain are defined in the following clauses.

5.4.1 Originating domain call group service capabilities

Figure 26 shows the initialization of a call object in the originating domain of a call.

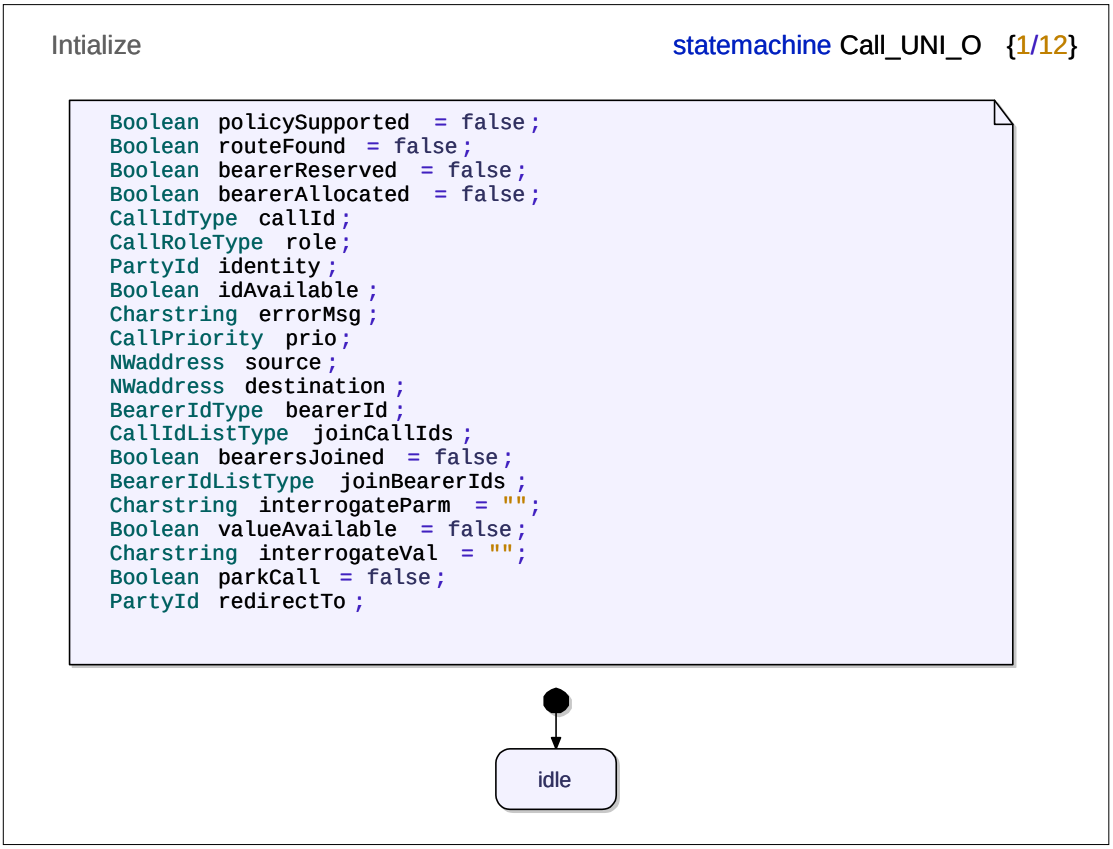


Figure 26: Initialization of originating domain call objects

5.4.1.1 Call setup

The *call setup* service capability establishes a call between two end points. The established call shall be characterized by the information elements in the supplied call descriptor. The originating domain *call setup* service capability is shown in figures 27 and 28.

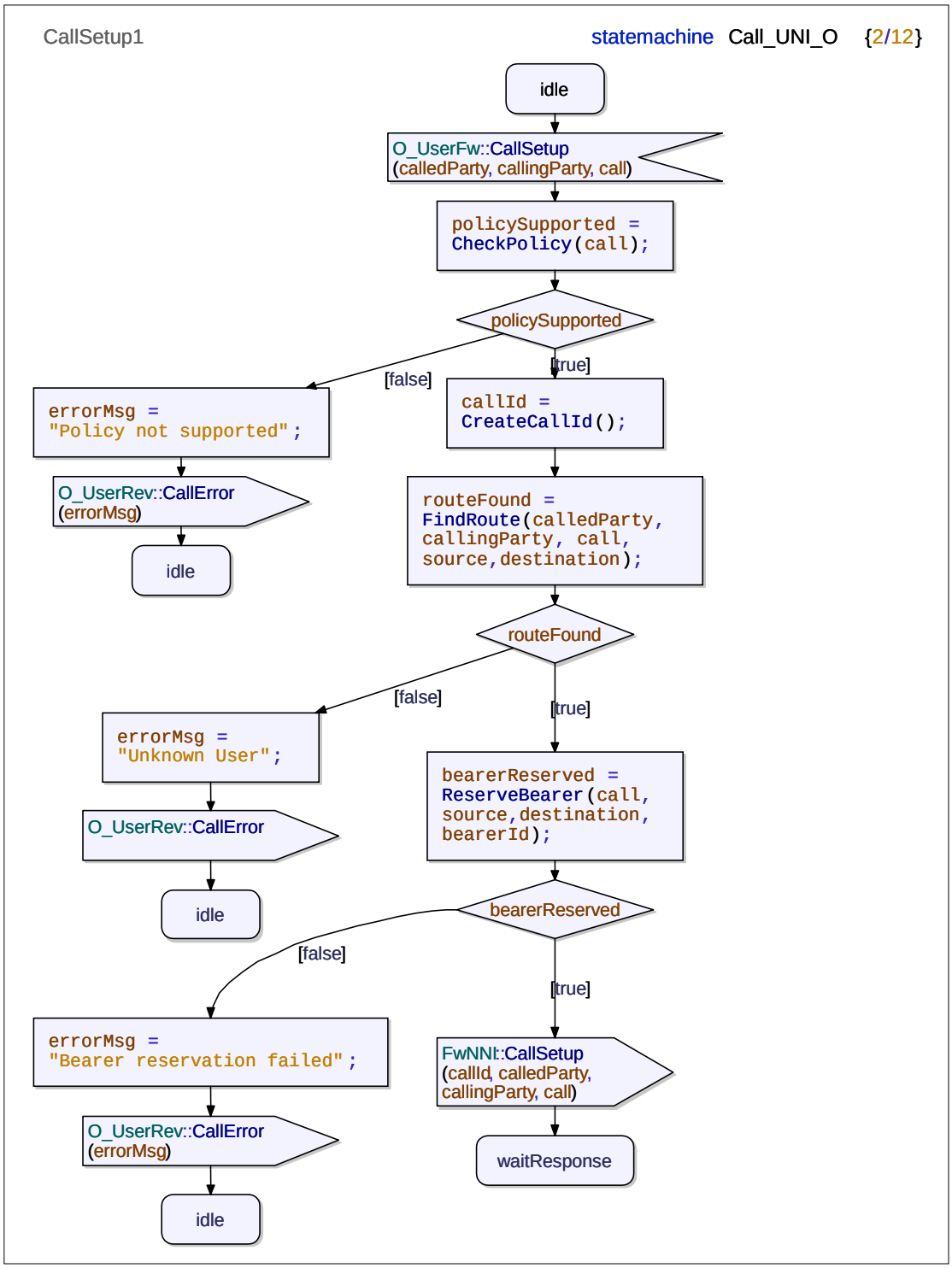


Figure 27: The originating domain call setup service capability (1 of 2)

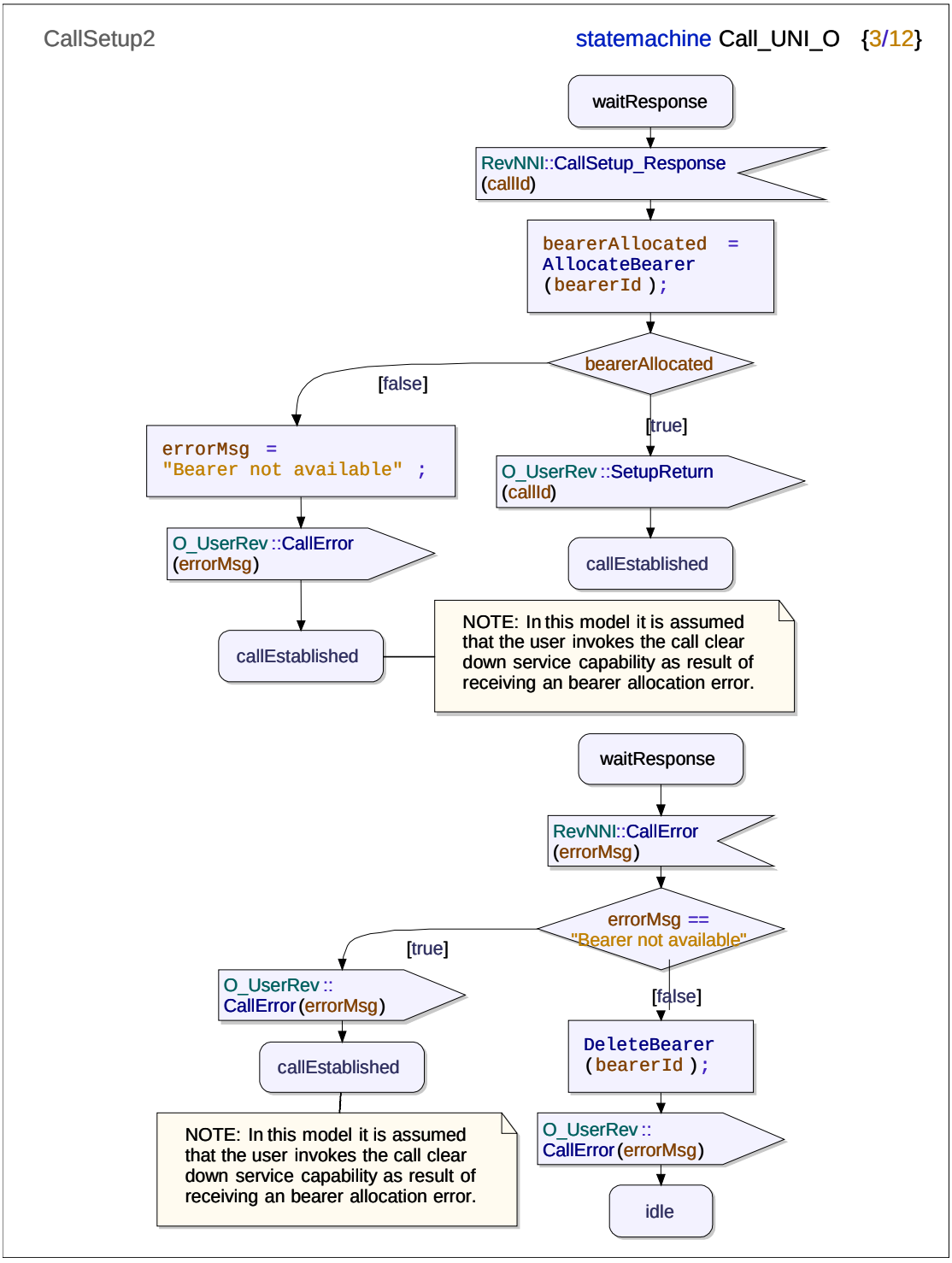


Figure 28: The originating domain call setup service capability (2 of 2)

5.4.1.2 Call identity delivery

The *call identity delivery* service capability delivers to an authorized user the identity of a party involved in an establishing or established call. The service capability behaviour is shown in figure 29.

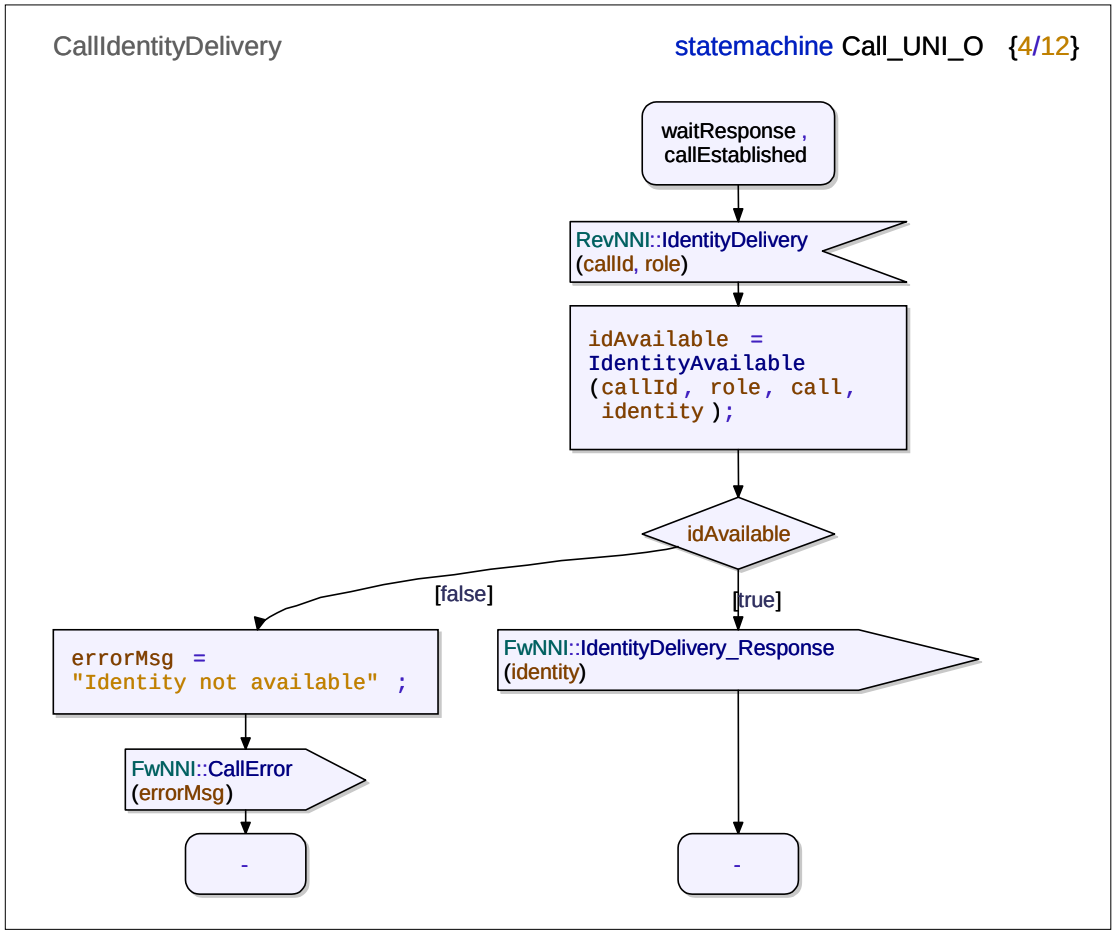


Figure 29: The originating domain call identity delivery service capability

5.4.1.3 Call redirect

The *call redirect* service capability changes one of the end-points of a call to another called user address based upon an event (for example to change called party when called party is busy, to perform park and retrieve operations). The behaviour of the service capability is shown in figure 30.

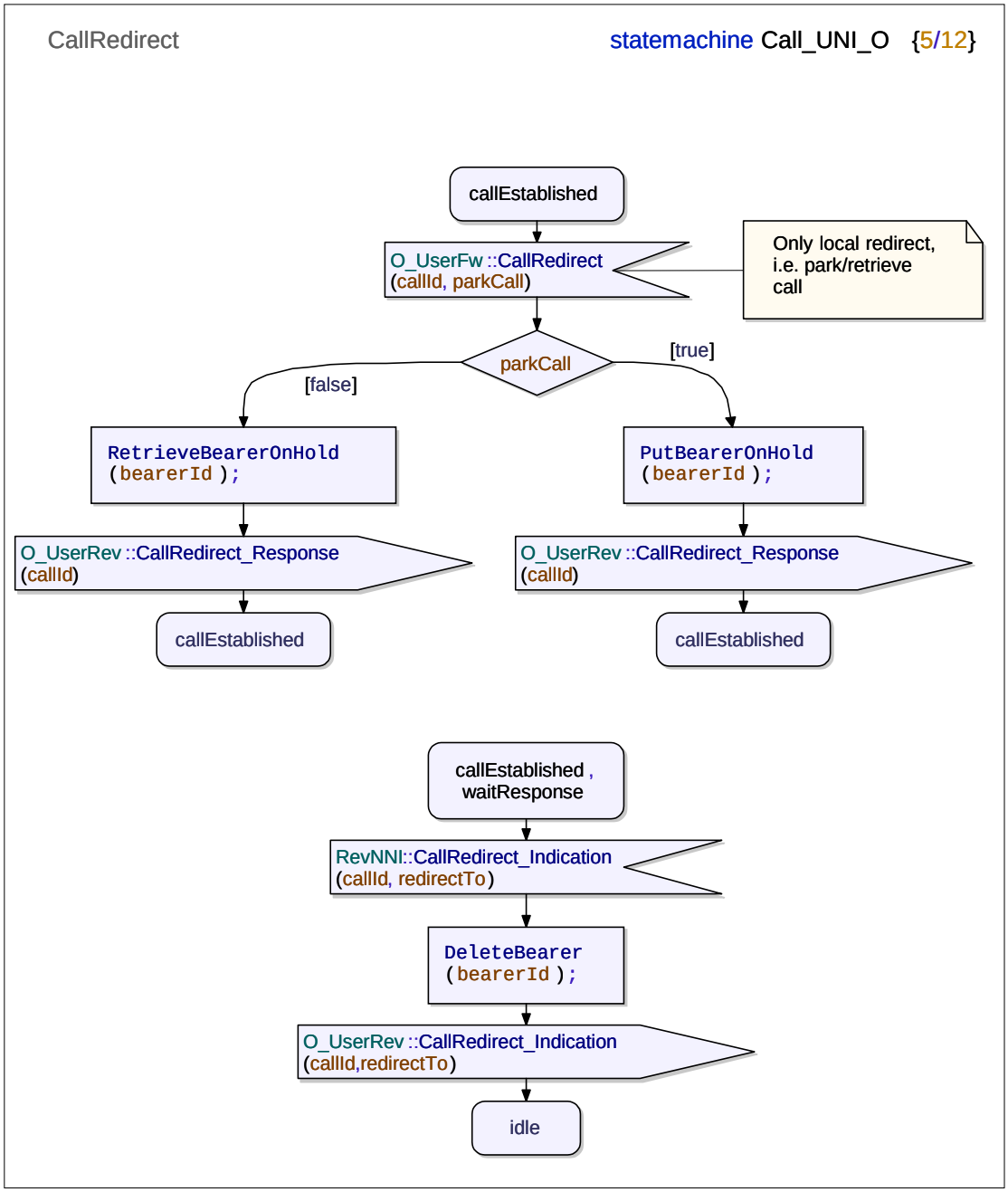


Figure 30: The originating domain call redirect service capability

5.4.1.4 Modify call priority

The *modify call priority* service capability modifies the priority assigned to a call (may be used to set Emergency priority on a dialled call). In figure 31 the behaviour of the *modify call priority* service capability is defined.

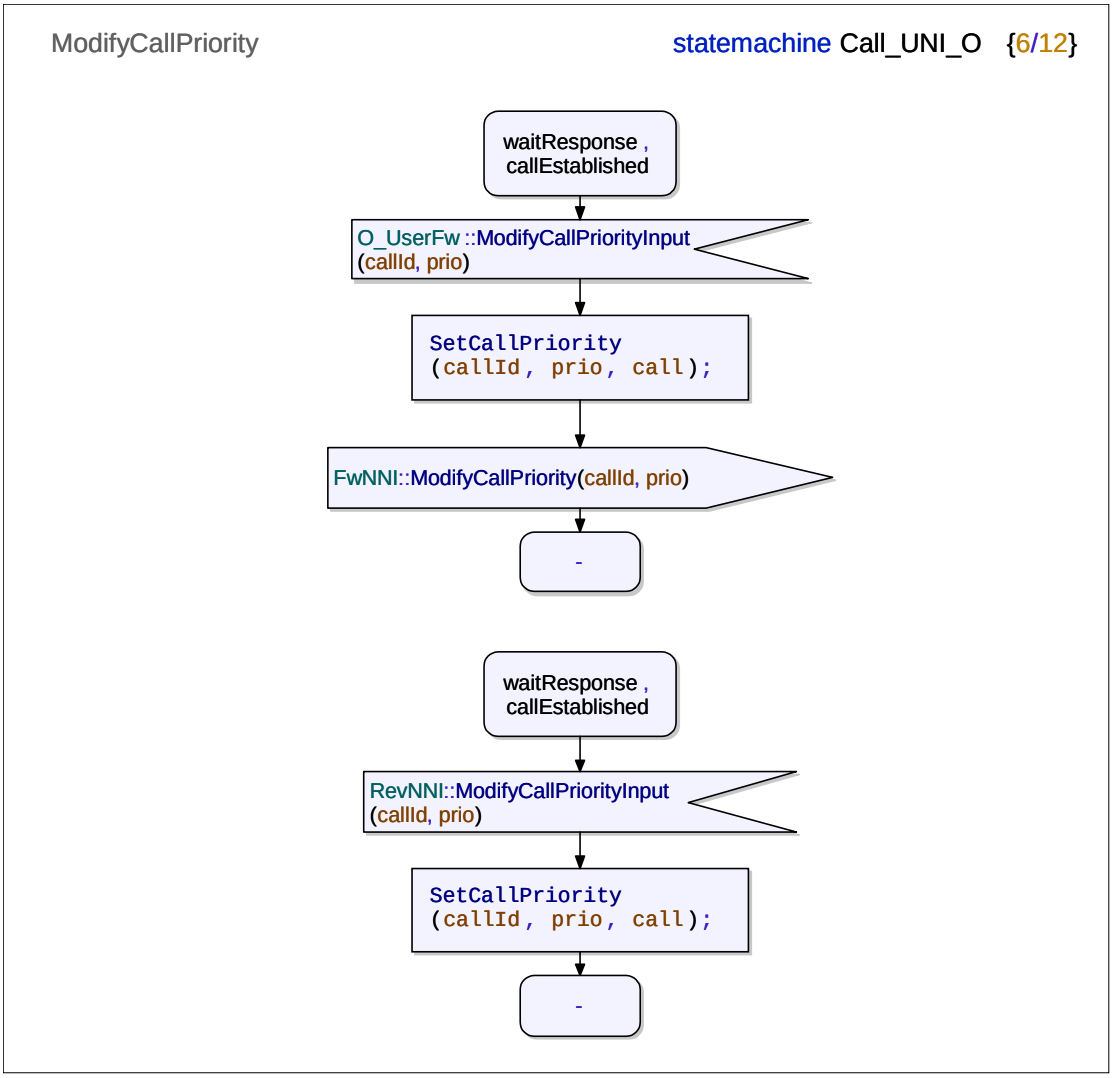


Figure 31: The originating domain modify call priority service capability

5.4.1.5 Call cleardown

The *call cleardown* service capability closes the call with the specified identity by removing the end-to-end connection. The behaviour is shown in figure 32.

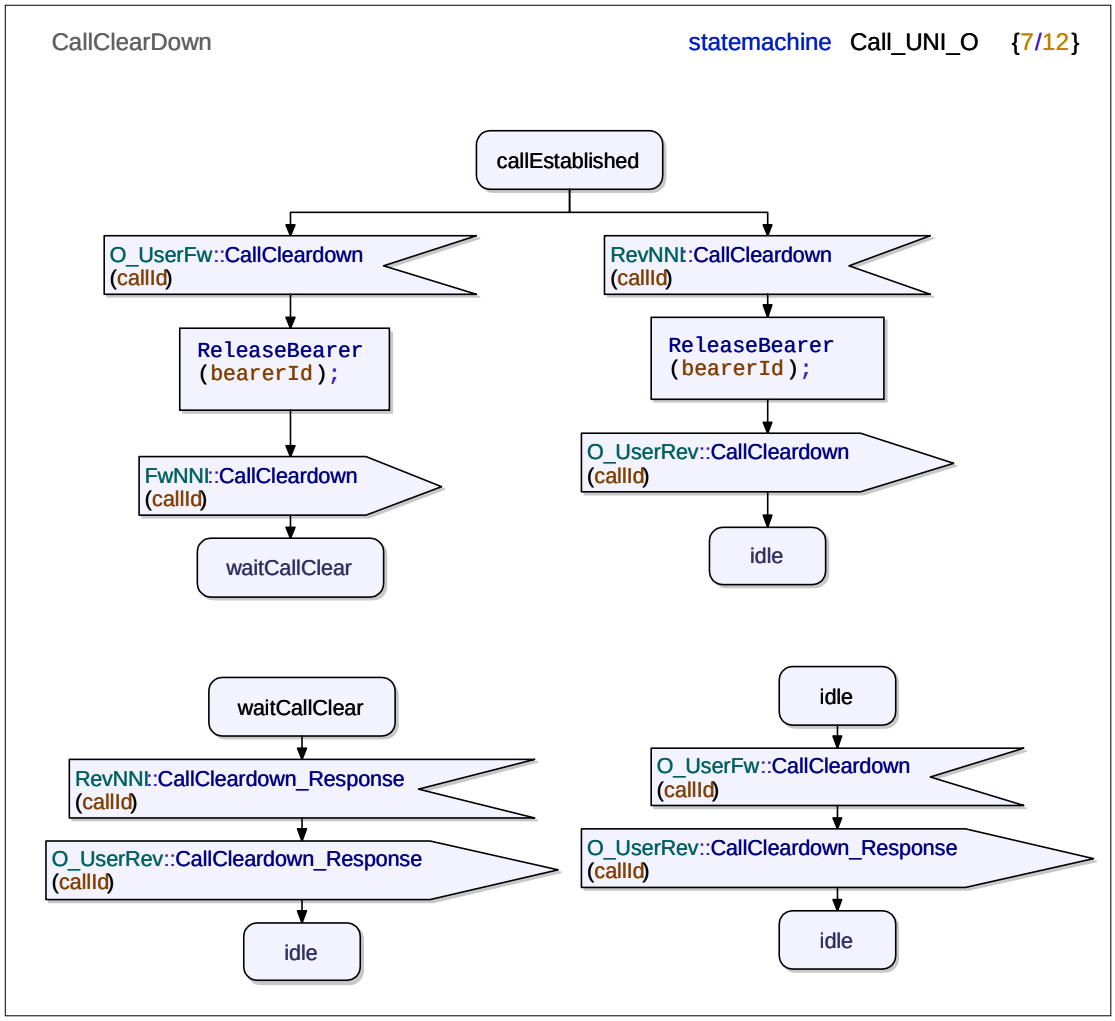


Figure 32: The originating domain call cleardown service capability

5.4.1.6 Call join

The *call join* service capability joins two or more calls sharing a common end-point. In figure 33 the behaviour of the *call join* service capability is defined.

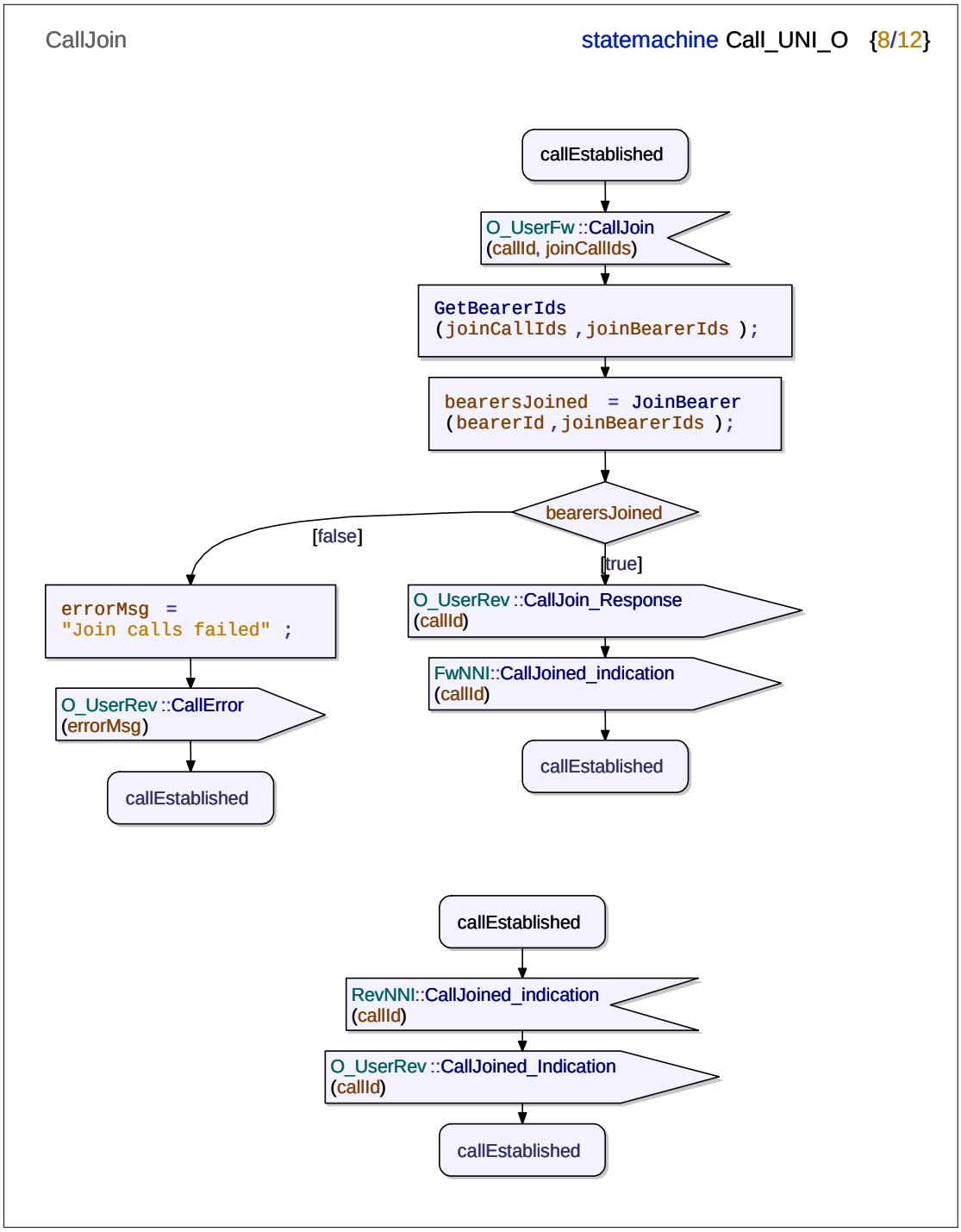


Figure 33: The originating domain call join service capability

5.4.1.7 Interrogate call

The *interrogate call* service capability returns the value of a user-specific attribute such as the contents of the call charge record to the invoking user or application. The service capability is shown in figure 34.

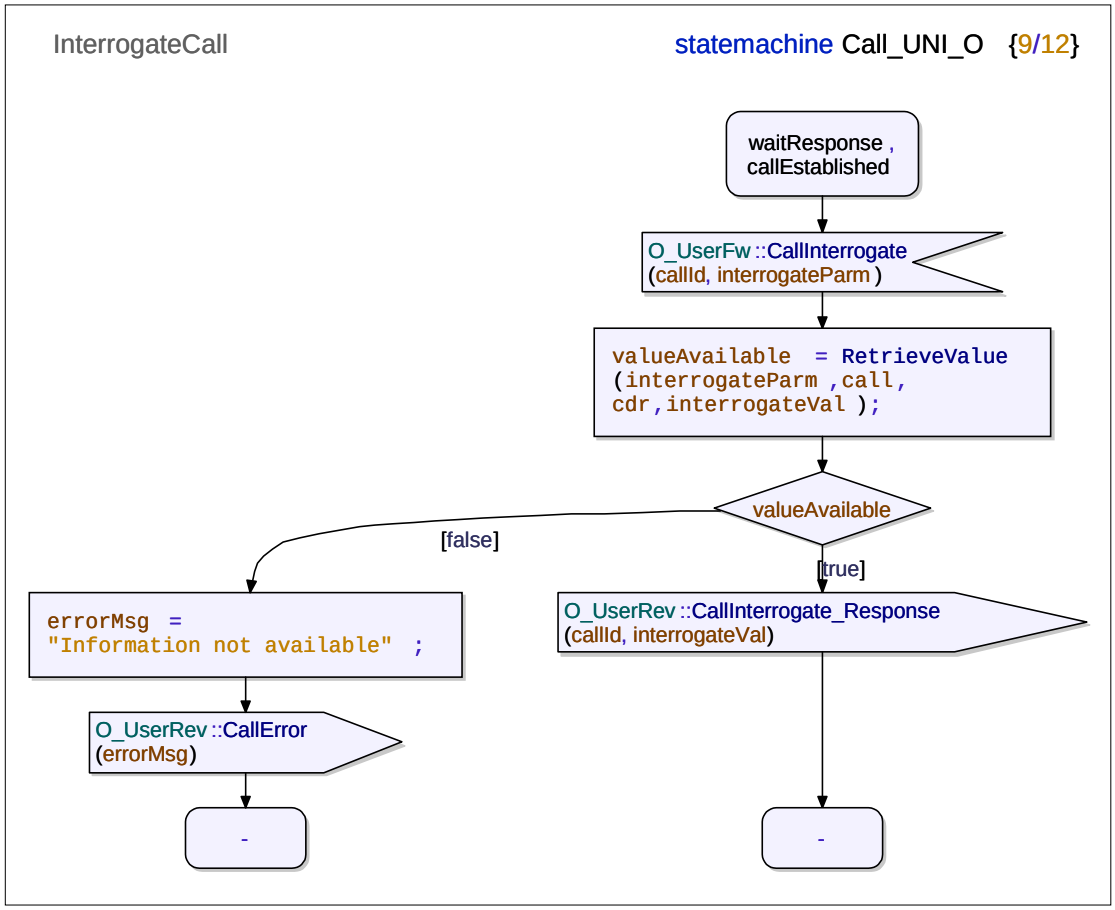


Figure 34: The originating domain interrogate call service capability

5.4.1.8 Operation signatures

The signatures of the operations used in the definition of the originating domain call group service capabilities are shown in figures 35, 36, and 37.

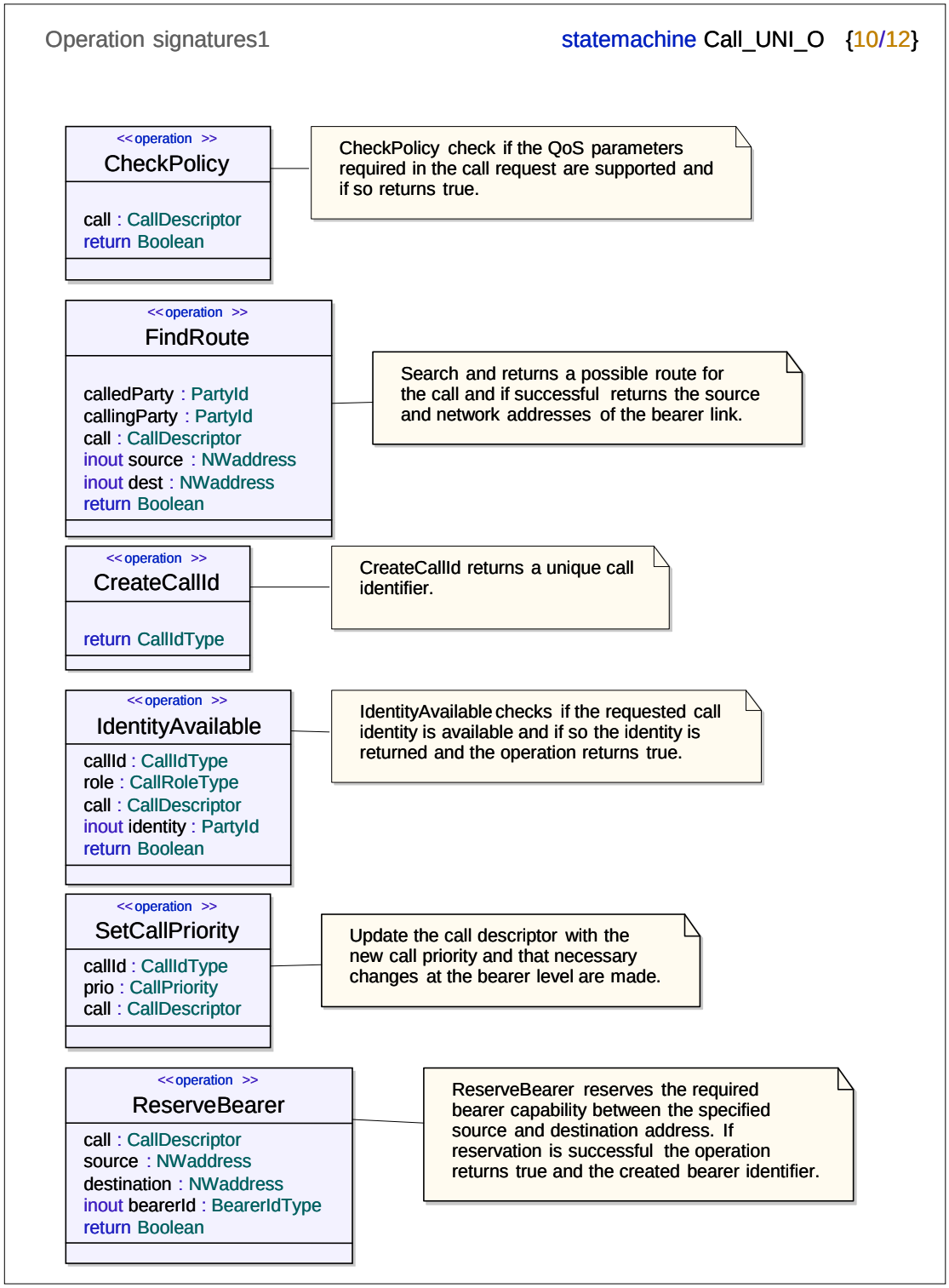


Figure 35: Originating domain operation signatures (1 of 3)

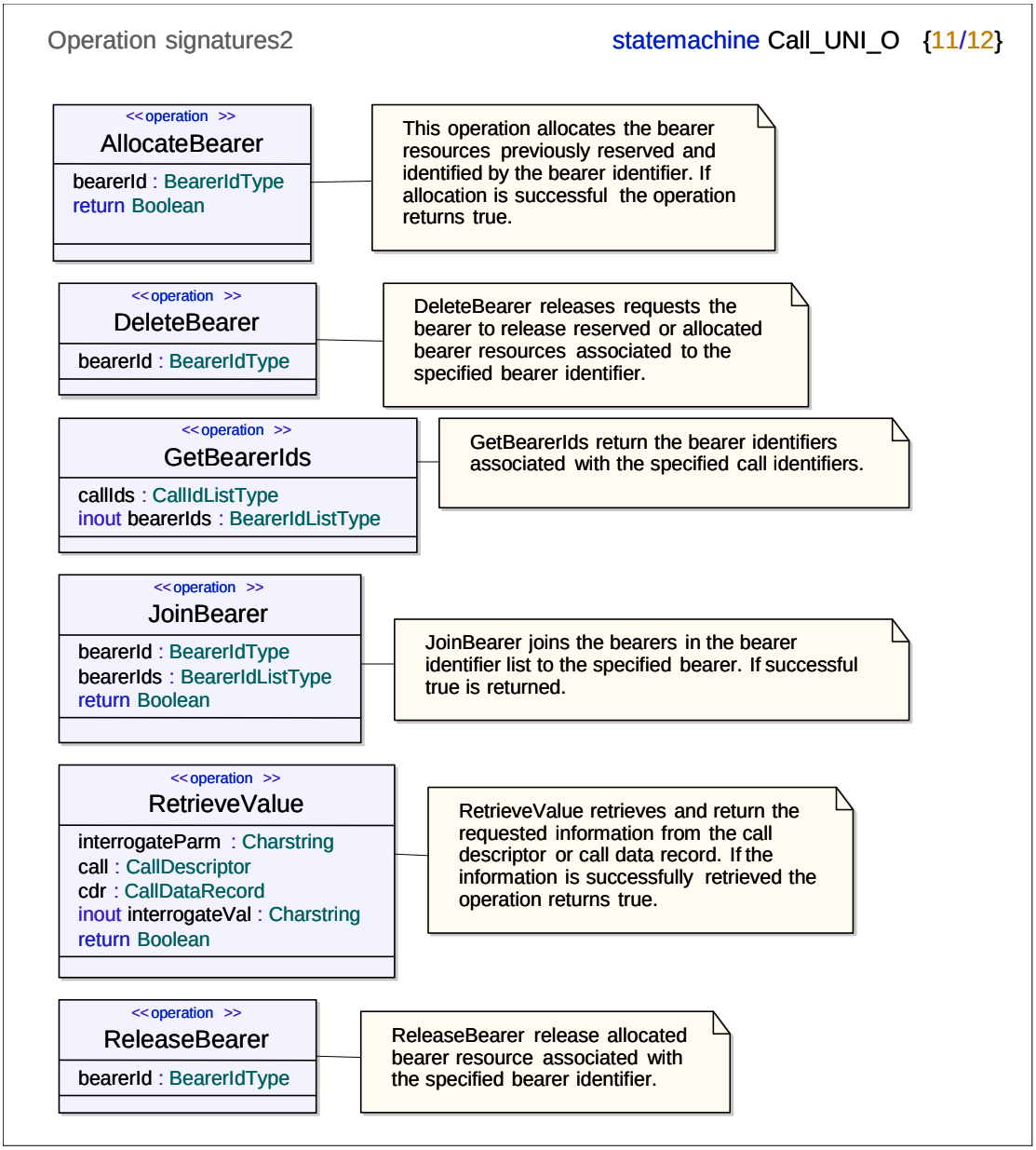


Figure 36: Originating domain operation signatures (2 of 3)

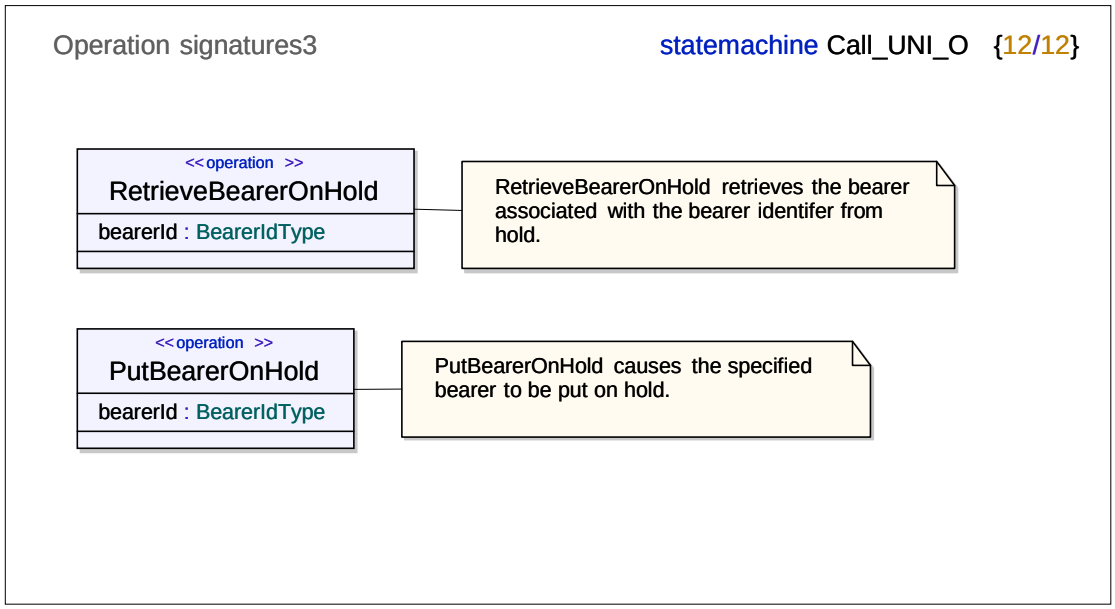


Figure 37: Originating domain operation signatures (3 of 3)

5.4.2 Intermediate domain call group service capabilities

Figure 38 shows the initialization of a call object in the intermediate domain of a call.

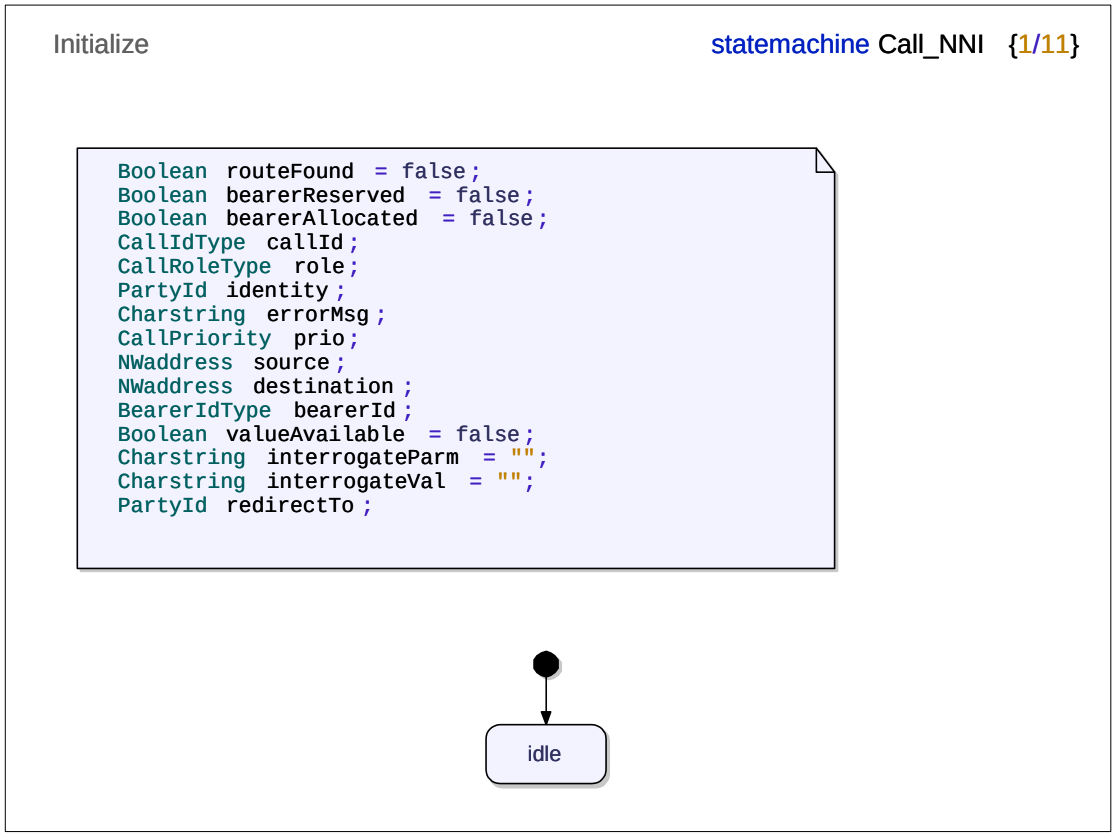


Figure 38: Initialization of intermediate domain call objects

5.4.2.1 Call setup

The *call setup* service capability establishes a call between two end points. The established call shall be characterized by the information elements in the supplied call descriptor. An intermediate domain *call setup* service capability is shown in figures 39 and 40.

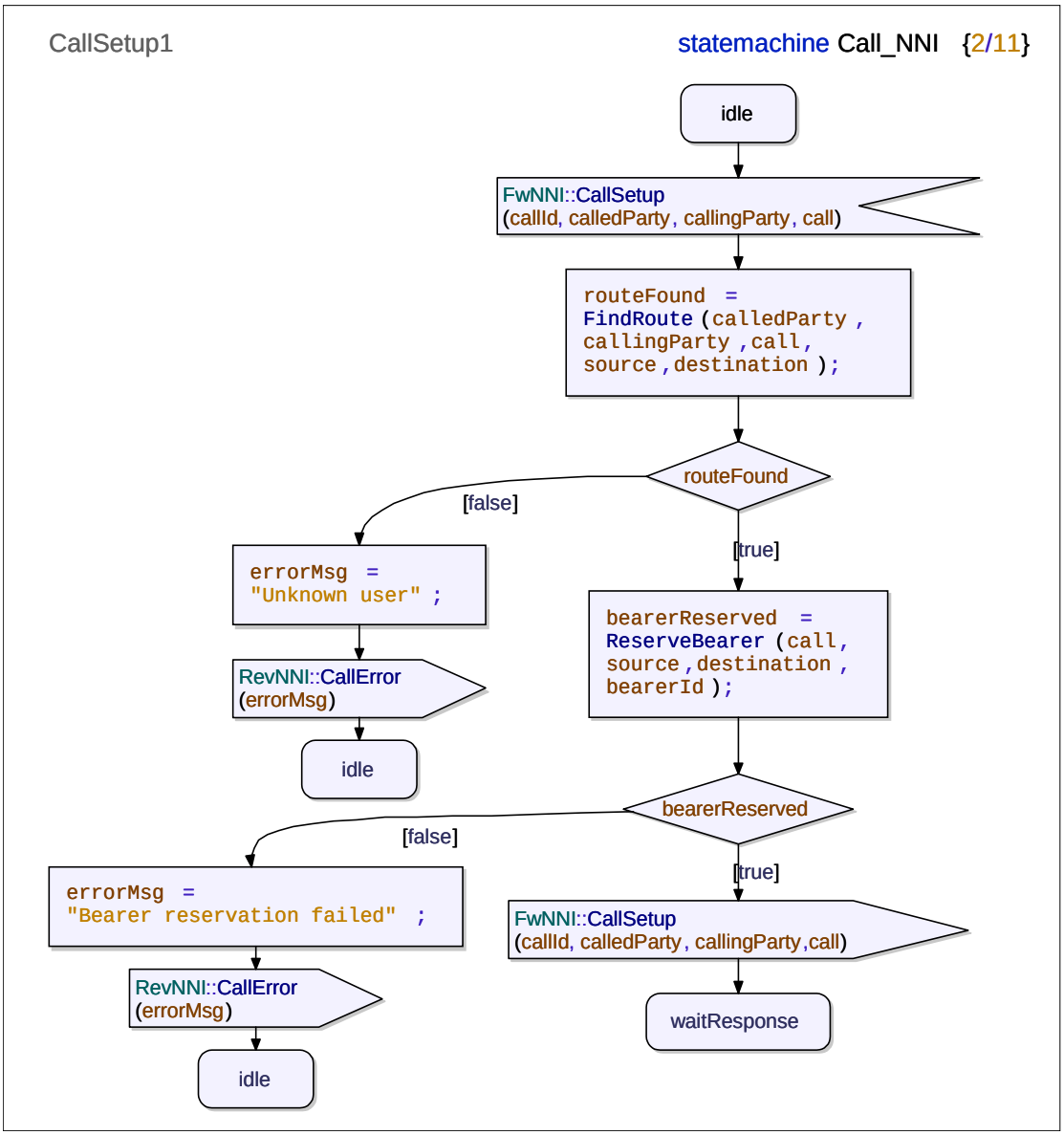


Figure 39: Intermediate domain call setup service capability (1 of 2)

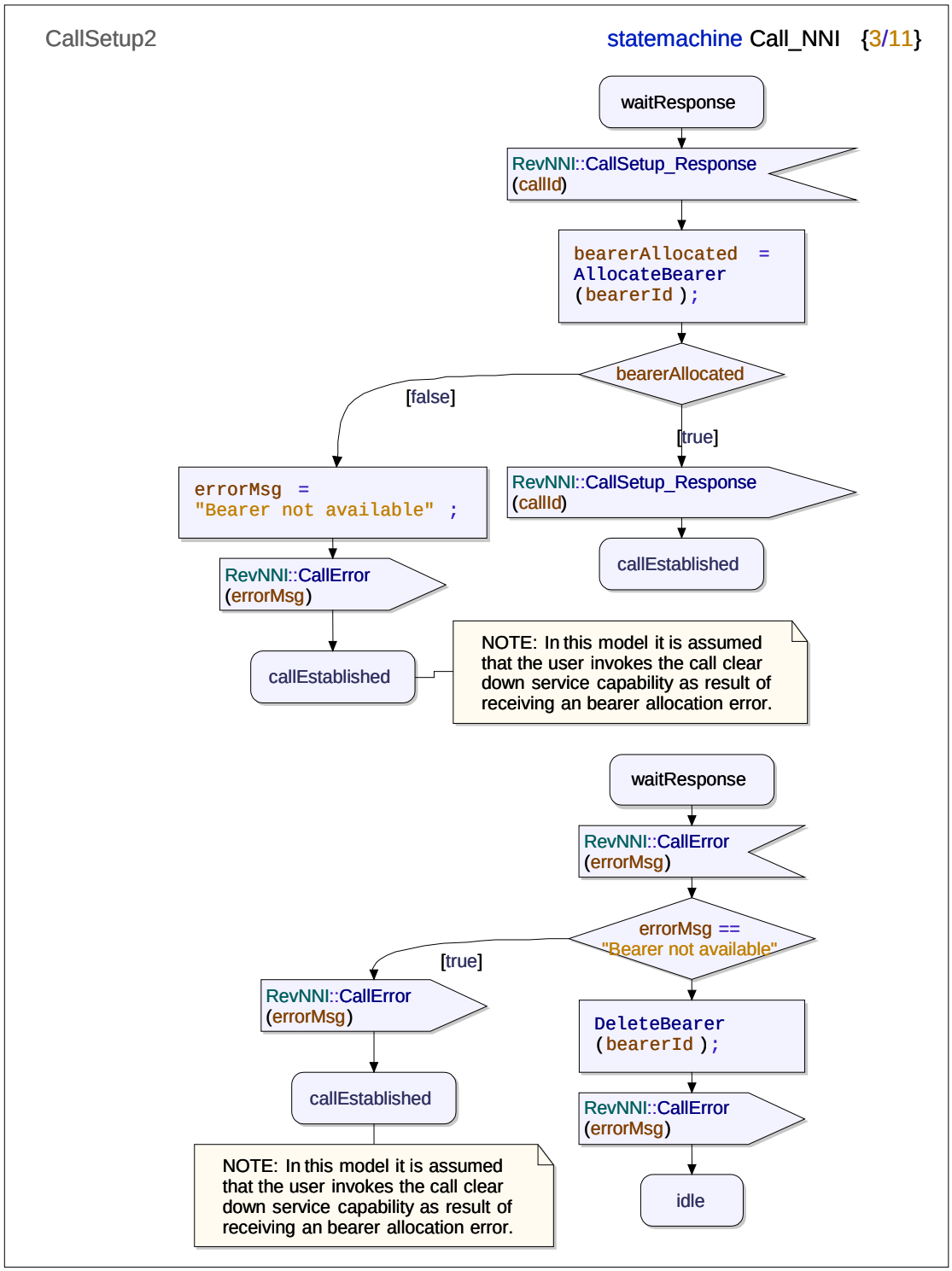


Figure 40: Intermediate domain call setup service capability (2 of 2)

5.4.2.2 Call identity delivery

The *call identity delivery* service capability delivers to an authorized user the identity of a party involved in an establishing or established call. The intermediate domain *call identity delivery* service capability is shown in figure 41.

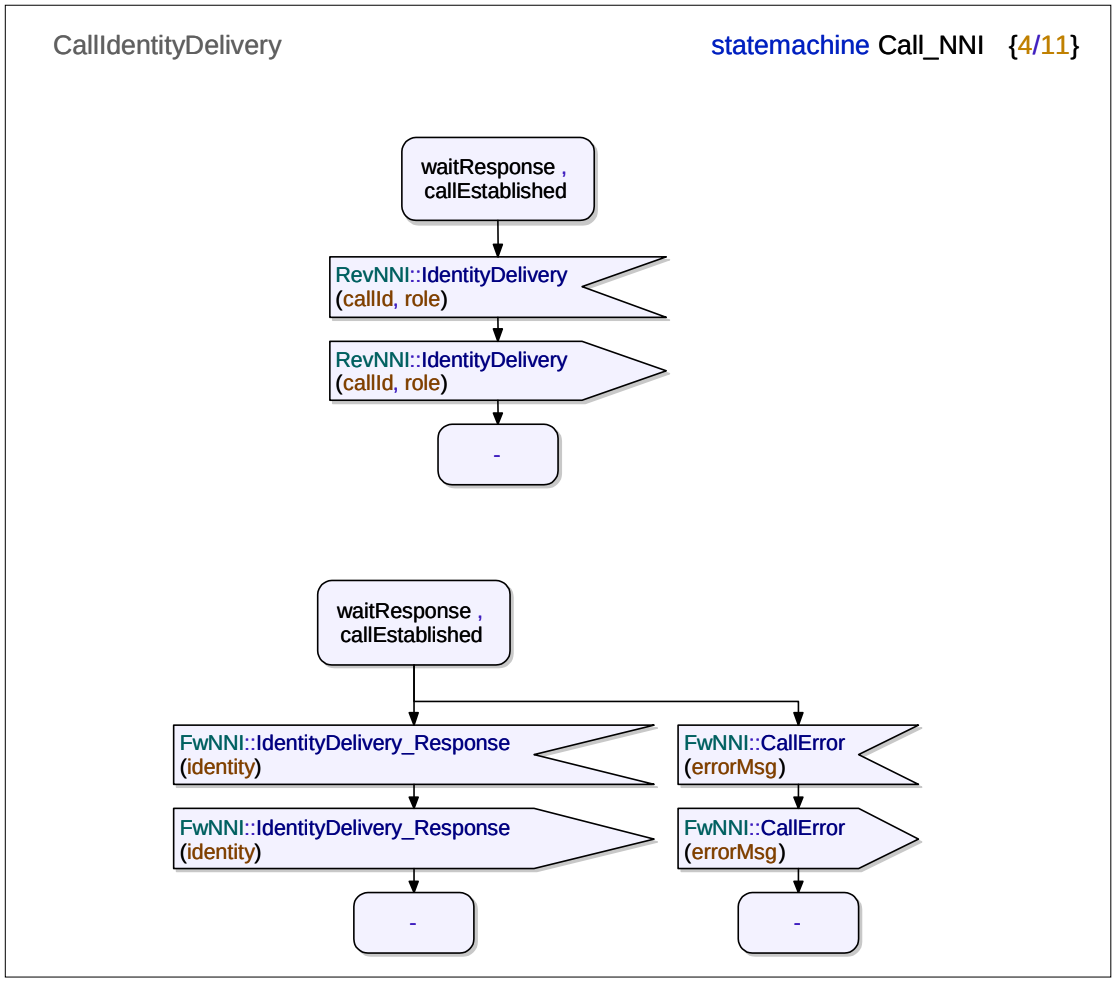


Figure 41: Intermediate domain call identity delivery service capability

5.4.2.3 Call redirect

The *call redirect* service capability changes one of the end-points of a call to another called user address based upon an event (for example to change called party when called party is busy, to perform park and retrieve operations). The behaviour of the service capability in an intermediate domain is shown in figure 42.

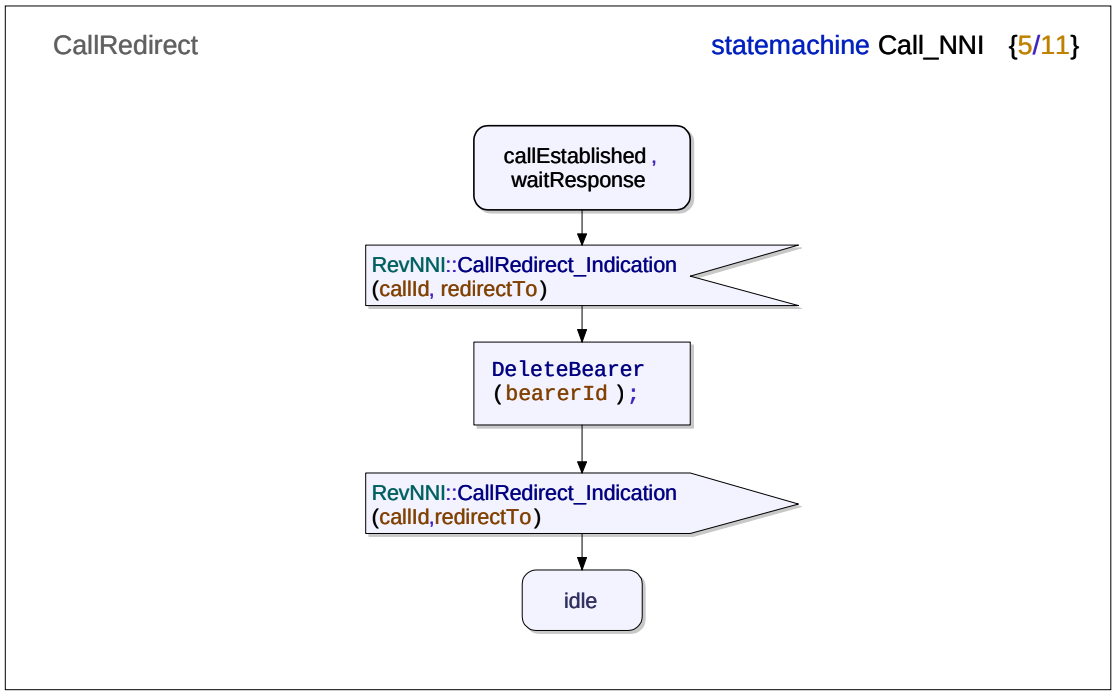


Figure 42: Intermediate domain call redirect service capability

5.4.2.4 Modify call priority

The *modify call priority* service capability modifies the priority assigned to a call (may be used to set Emergency priority on a dialled call). In figure 43 the behaviour of the intermediate domain *modify call priority* service capability is defined.

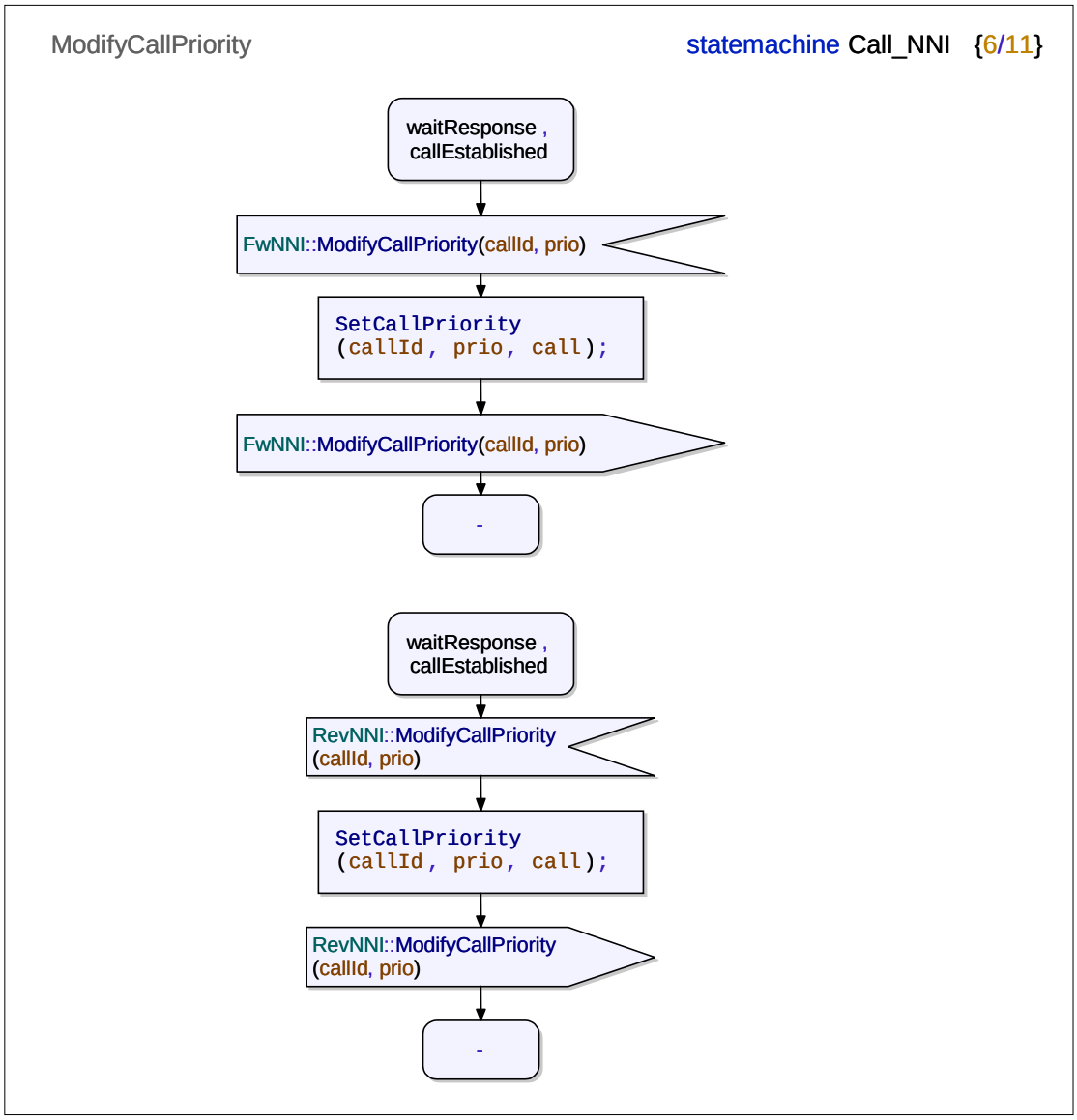


Figure 43: Intermediate domain modify call priority service capability

5.4.2.5 Call cleardown

The *call cleardown* service capability closes the call with the specified identity by removing the end-to-end connection. The behaviour in the intermediate domain is shown in figure 44.

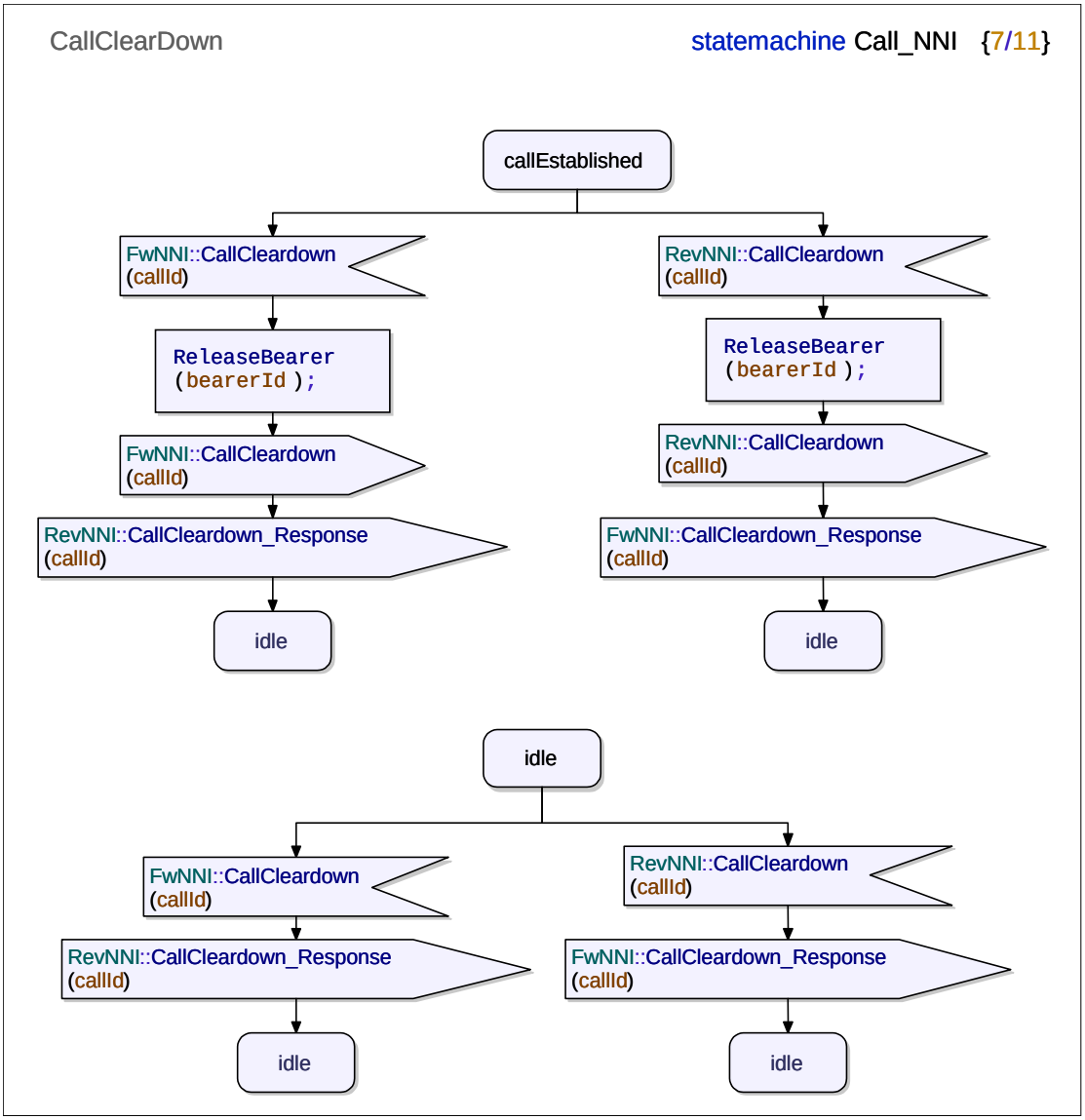


Figure 44: Intermediate domain call cleardown service capability

5.4.2.6 Call join

The *call join* service capability joins two or more calls sharing a common end-point. In figure 45 the behaviour of the *call join* service capability in the intermediate domain is defined.

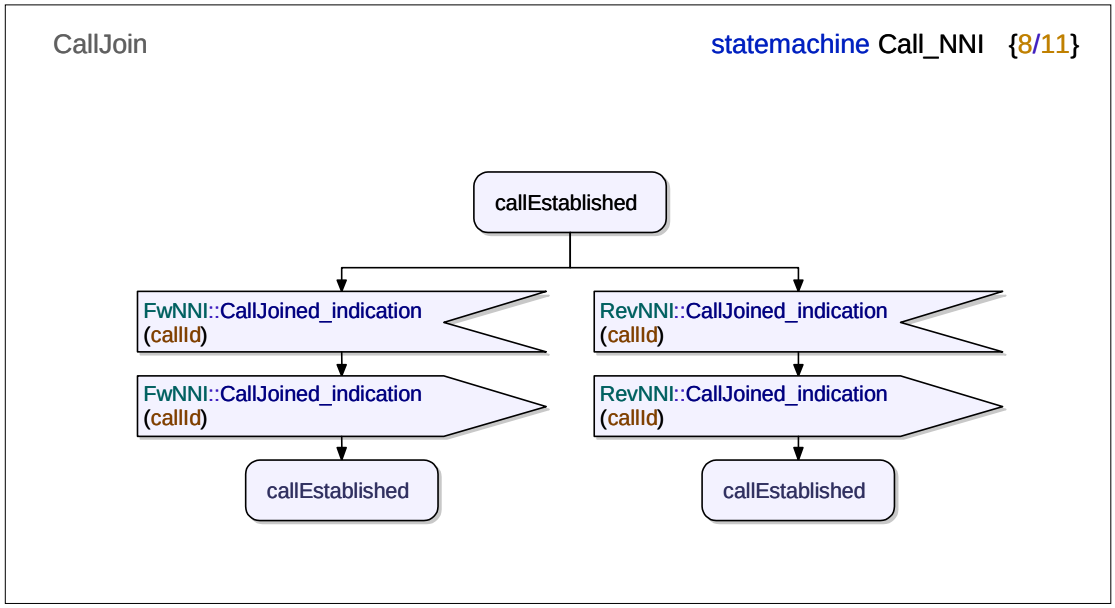


Figure 45: Intermediate domain call join service capability

5.4.2.7 Interrogate call

The *interrogate call* service capability returns the value of a user-specific attribute such as the contents of the call charge record to the invoking user or application. The intermediate domain *interrogate call* service capability is shown in figure 46.

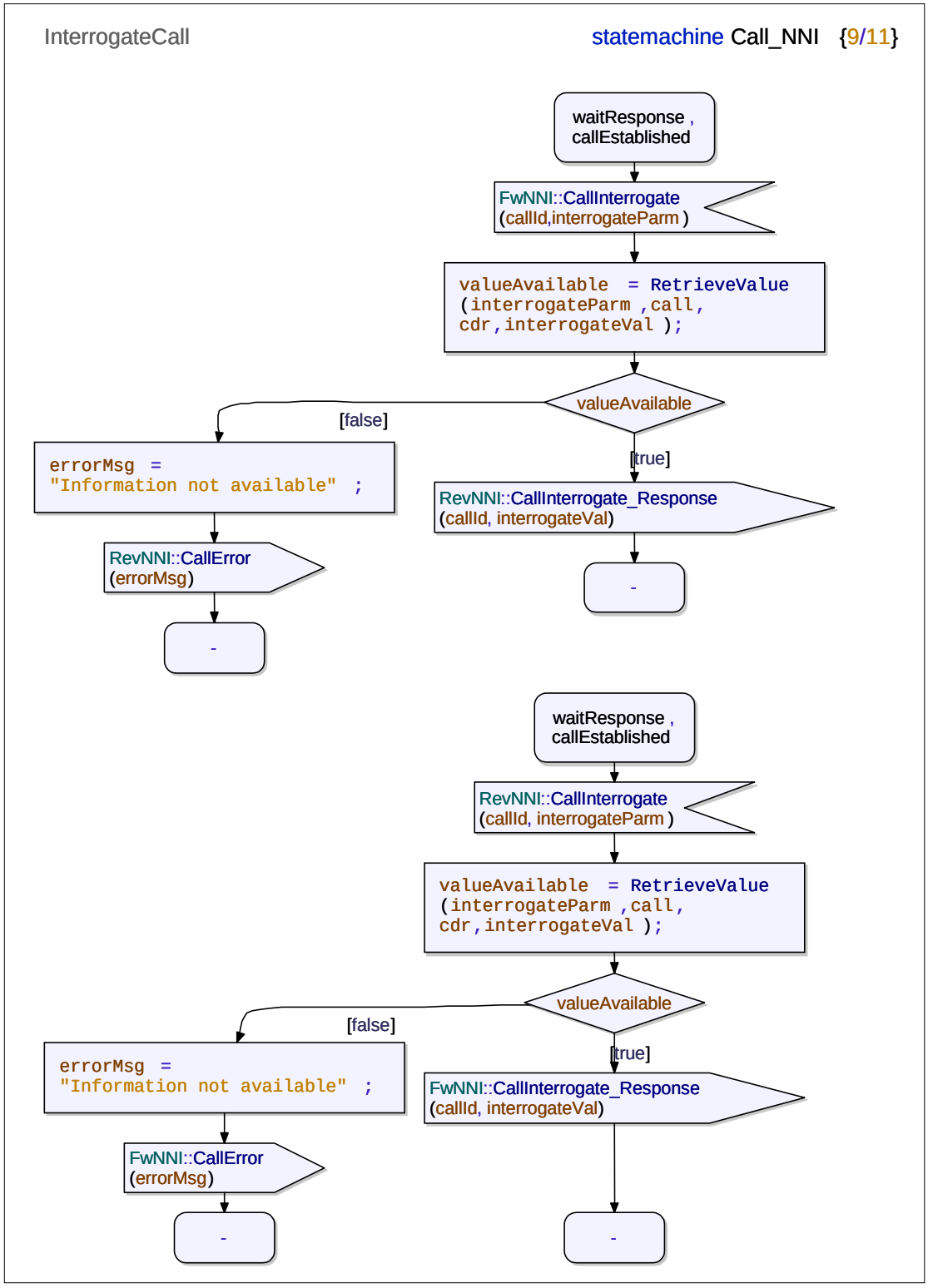


Figure 46: Intermediate domain interrogate call service capability

5.4.2.8 Operation signatures

The signatures of the operations used in the definition of the intermediate domain call group service capabilities are shown in figures 47 and 48.

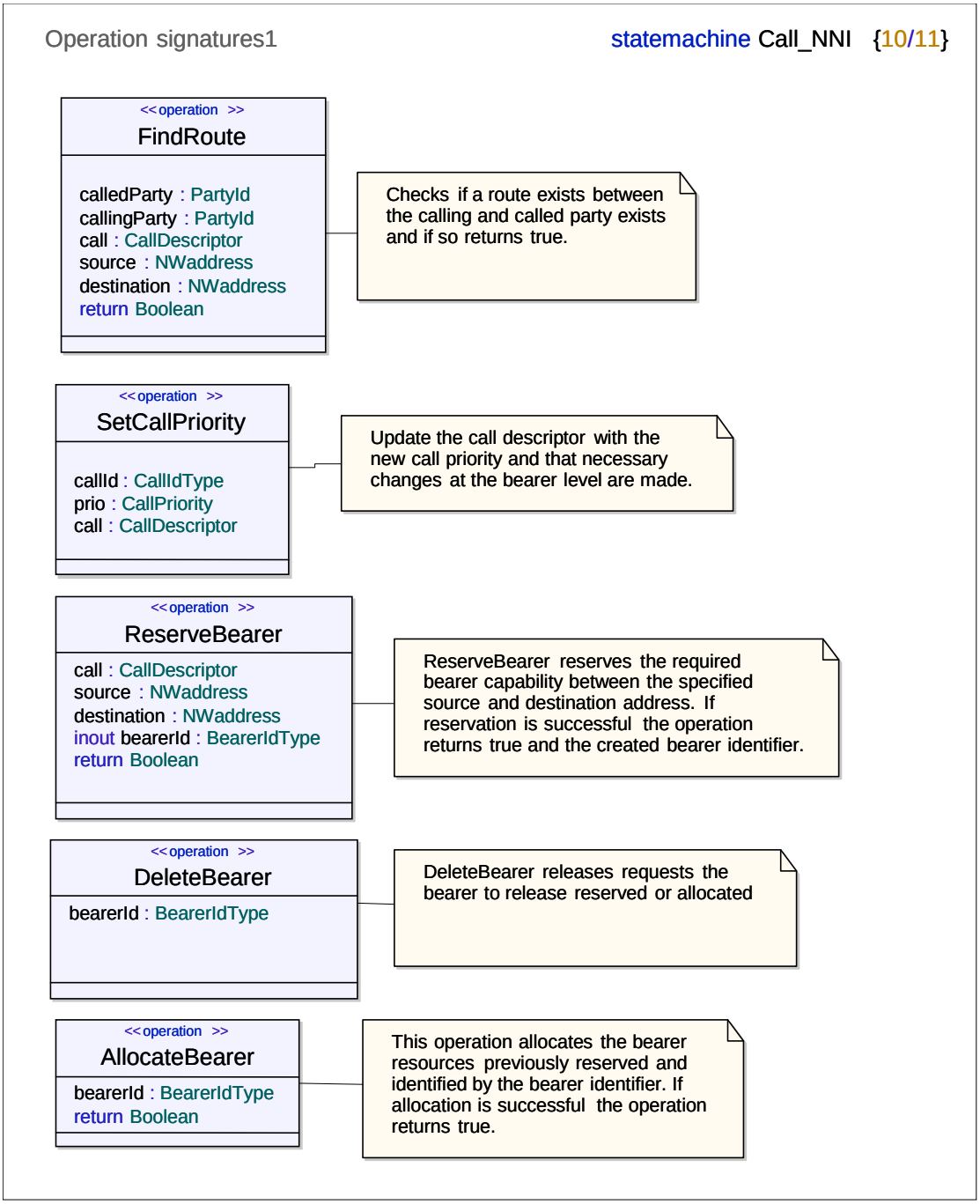


Figure 47: Intermediate domain operation signatures (1 of 2)

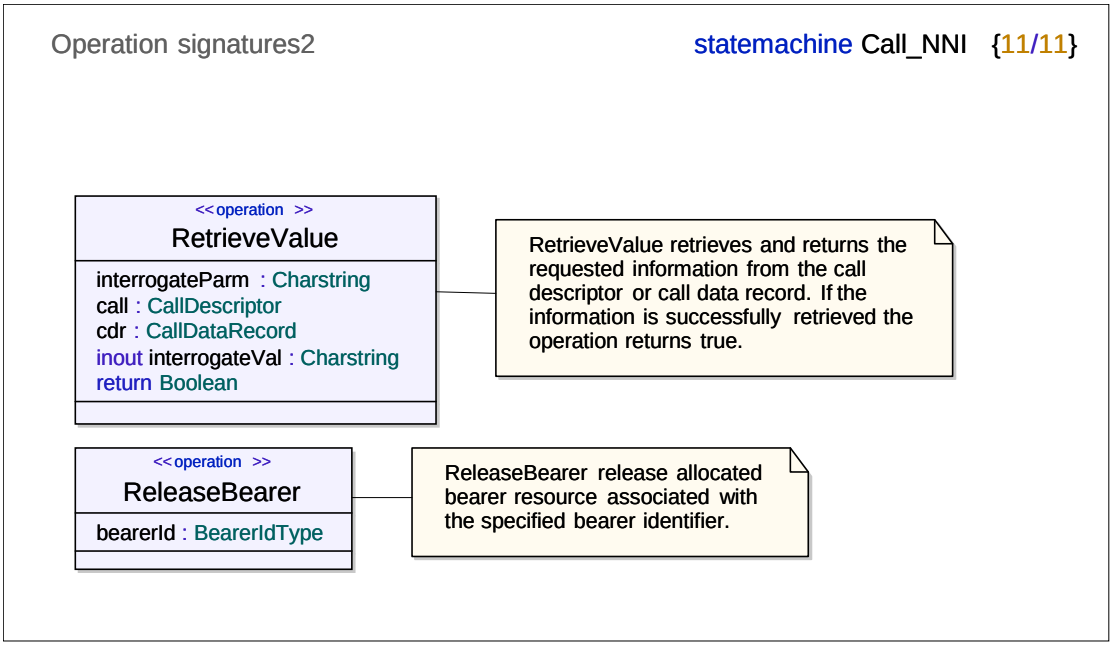


Figure 48: Intermediate domain operation signatures (2 of 2)

5.4.3 Destination domain call group service capabilities

Figure 49 shows the initialization of a call object in the destination domain of a call.

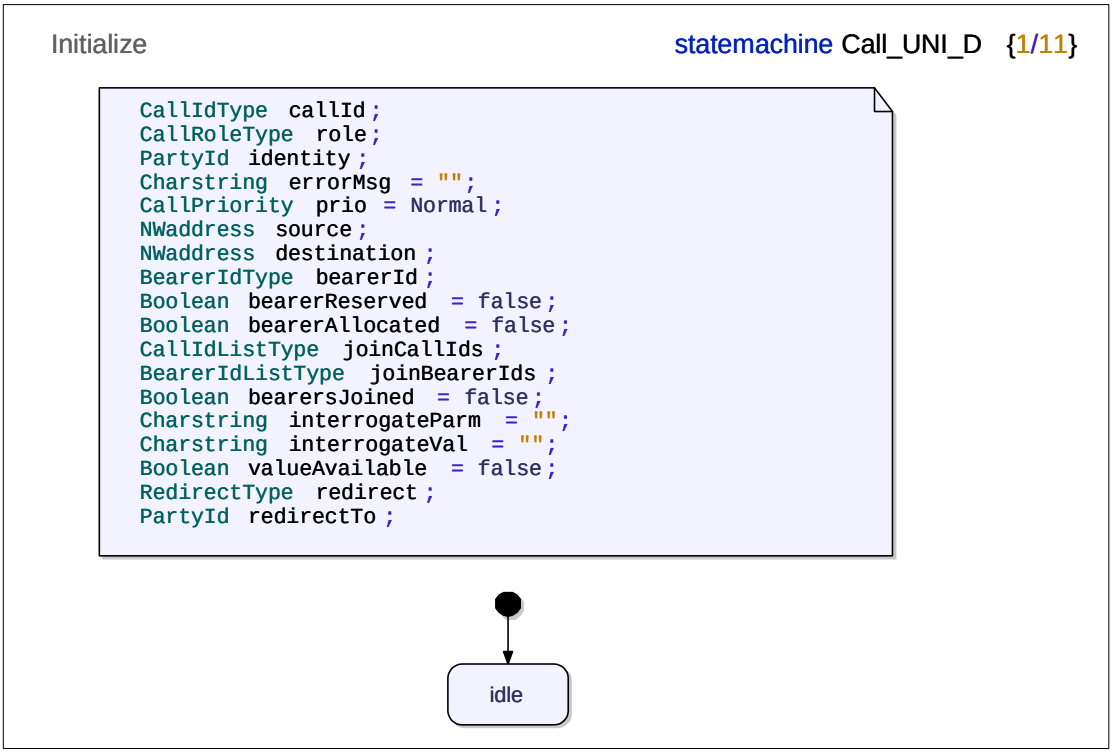


Figure 49: Initialization of destination domain call objects

5.4.3.1 Call setup

The *call setup* service capability establishes a call between two end points. The established call shall be characterized by the information elements in the supplied call descriptor. The destination domain *call setup* service capability is shown in figures 50 and 51.

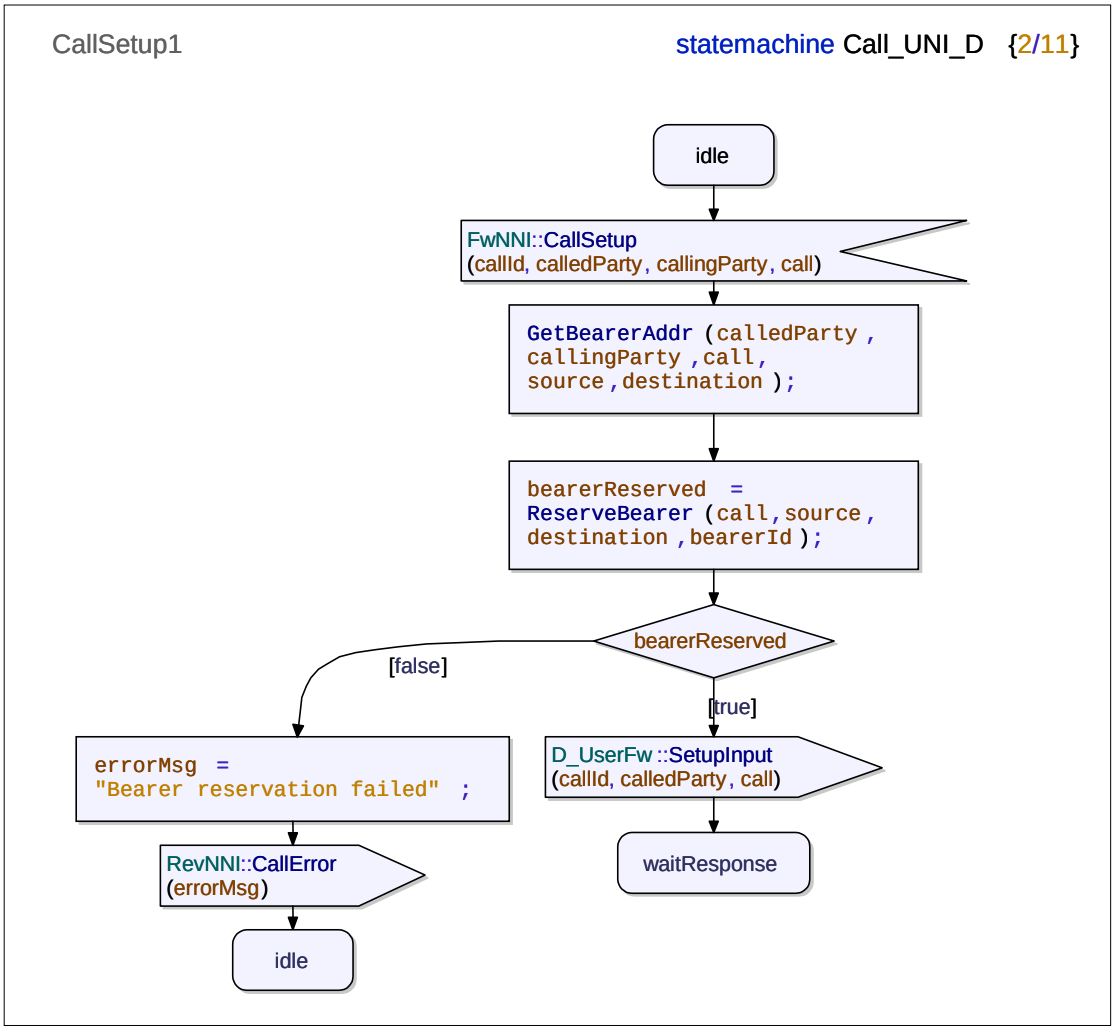


Figure 50: The destination domain call setup service capability (1 of 2)

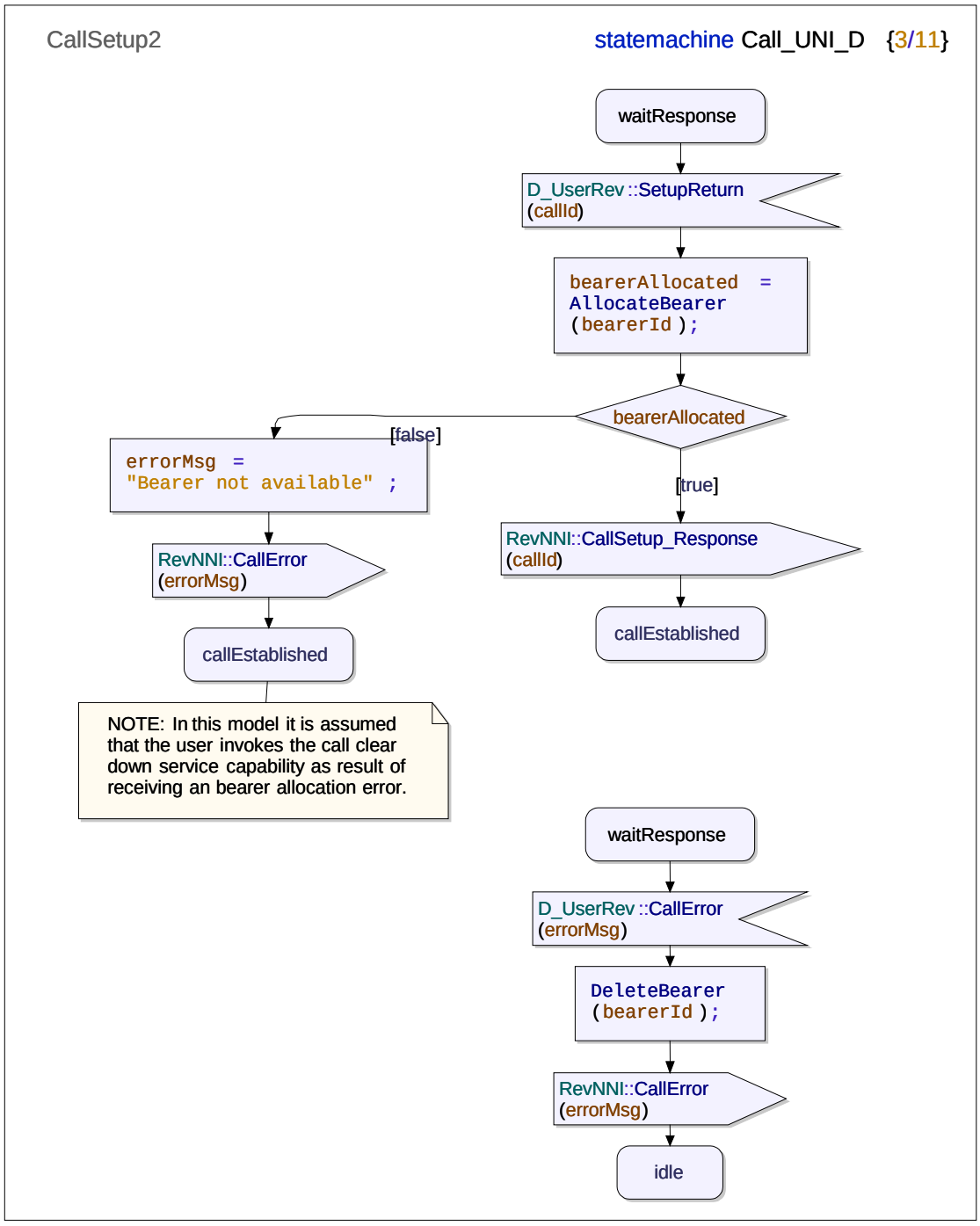


Figure 51: The destination domain call setup service capability (2 of 2)

5.4.3.2 Call identity delivery

The *call identity delivery* service capability delivers to an authorized user the identity of a party involved in an establishing or established call. The destination domain *call identity delivery* service capability is shown in figure 52.

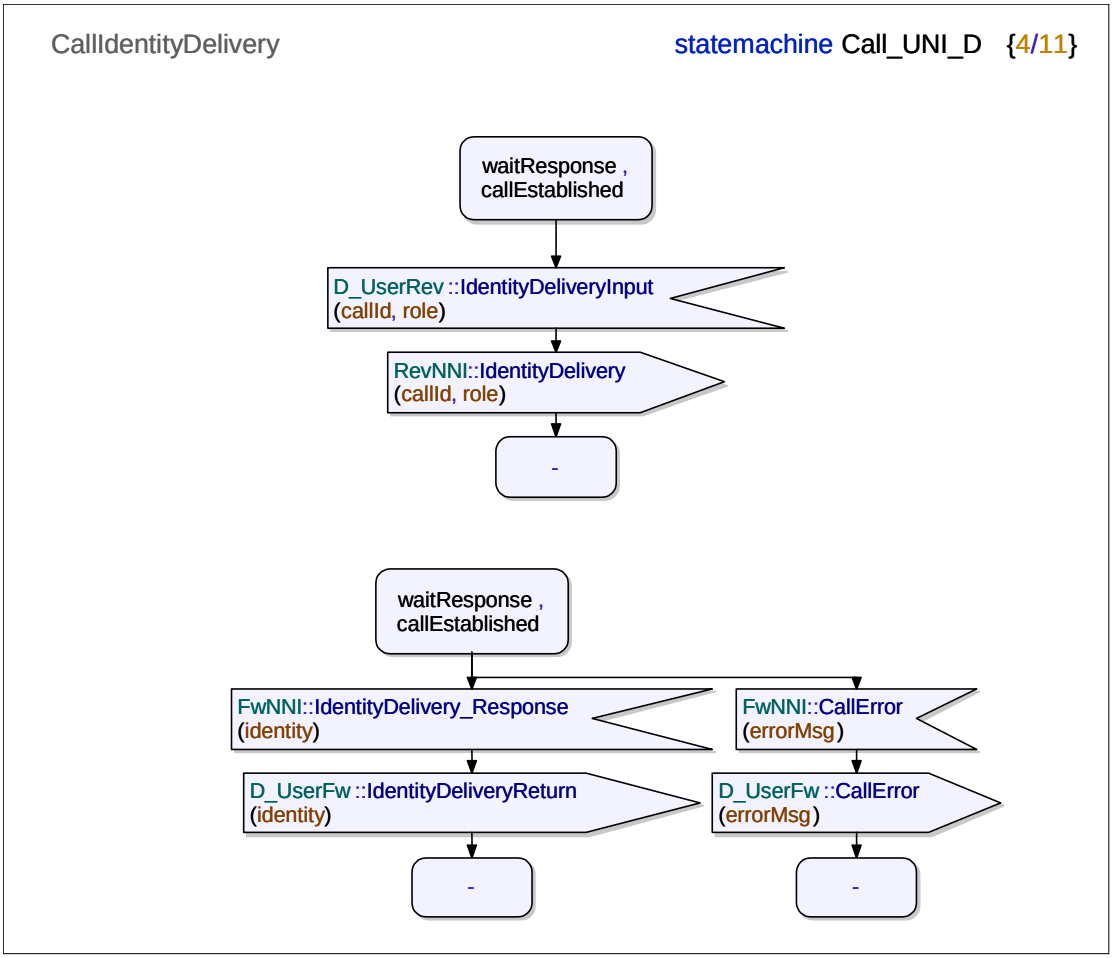


Figure 52: The destination domain call identity delivery service capability

5.4.3.3 Call redirect

The *call redirect* service capability changes one of the end-points of a call to another called user address based upon an event (for example to change called party when called party is busy, to perform park and retrieve operations). The behaviour of the service capability in the destination domain is shown in figure 53.

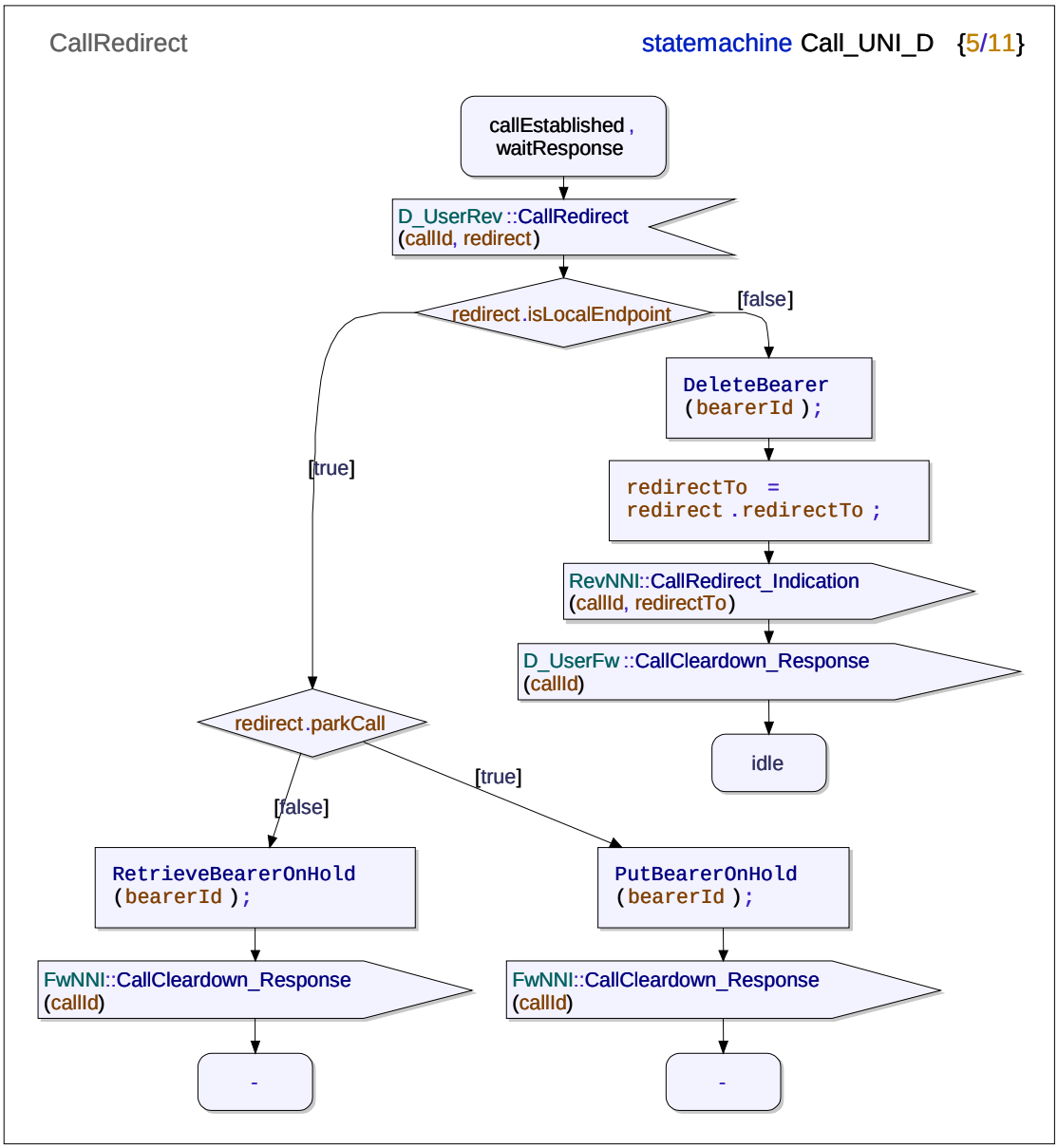


Figure 53: The destination domain call redirect service capability

5.4.3.4 Modify call priority

The *modify call priority* service capability modifies the priority assigned to a call (may be used to set Emergency priority on a dialled call). In figure 54 the behaviour of the destination domain *modify call priority* service capability is defined.

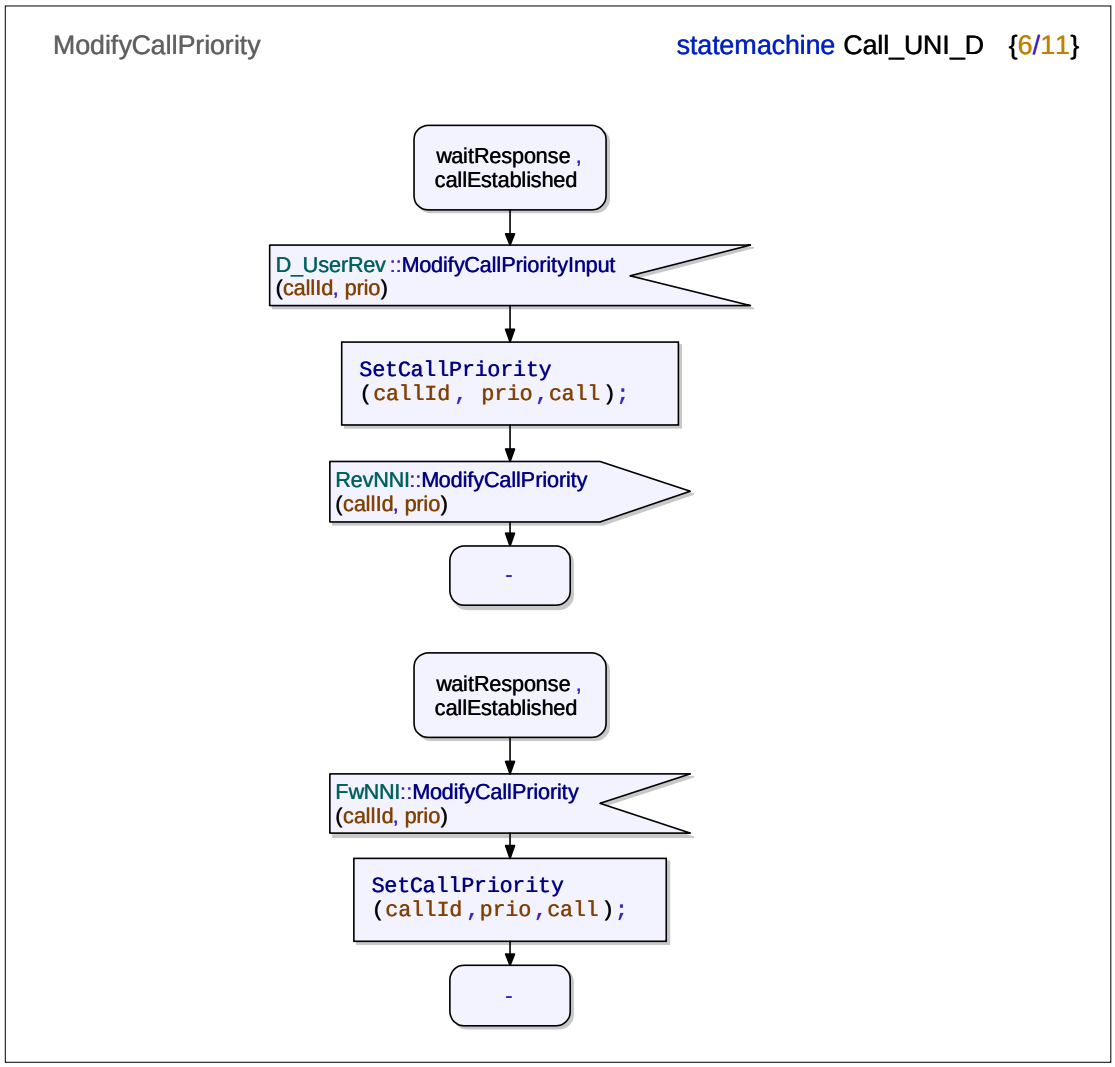


Figure 54: The destination domain modify call priority service capability

5.4.3.5 Call cleardown

The *call cleardown* service capability closes the call with the specified identity by removing the end-to-end connection. The behaviour in the intermediate domain is shown in figure 55.

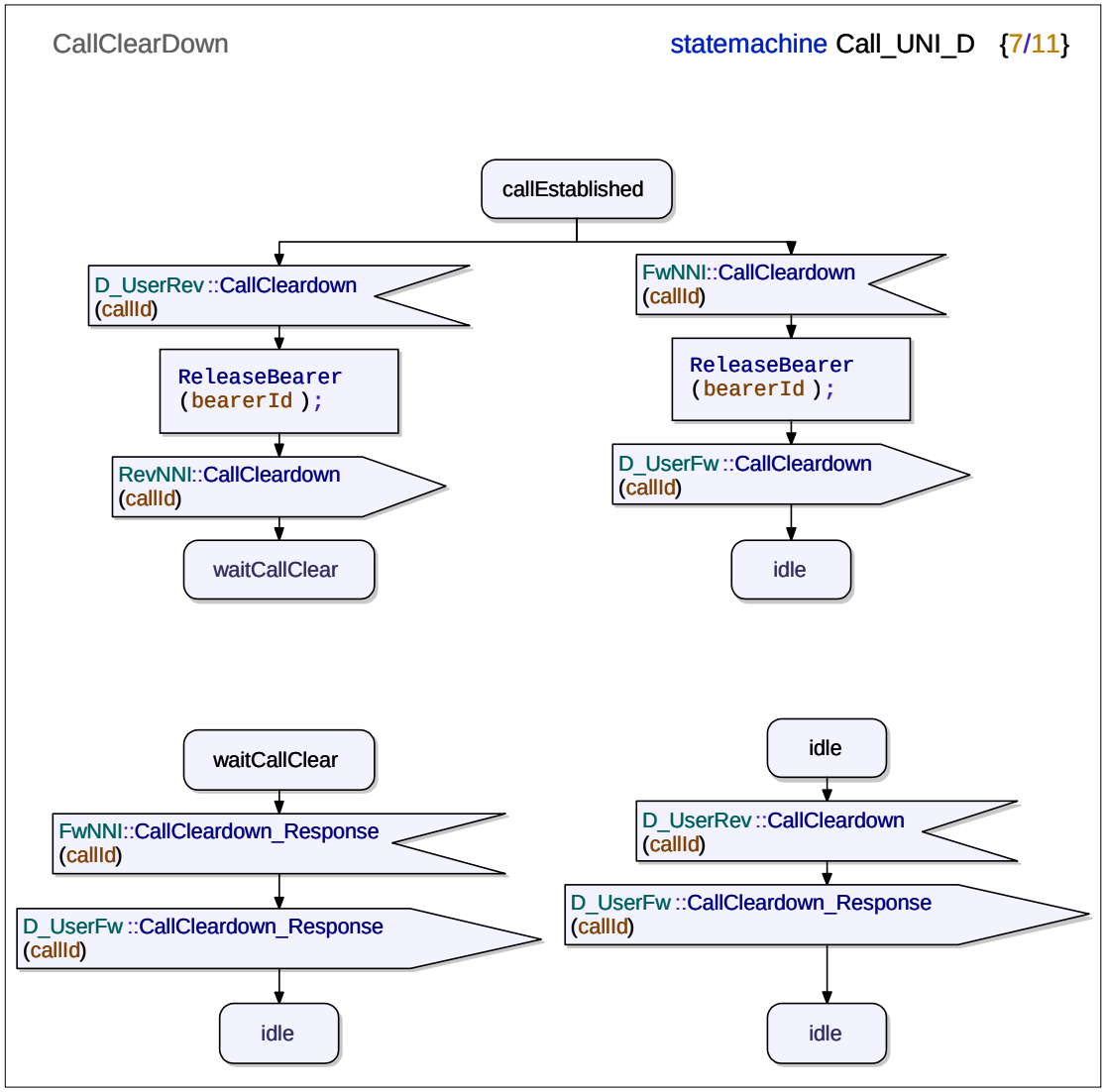


Figure 55: The destination domain call cleardown service capability

5.4.3.6 Call join

The *call join* service capability joins two or more calls sharing a common end-point. In figure 56 the behaviour of the *call join* service capability in the destination domain is defined.

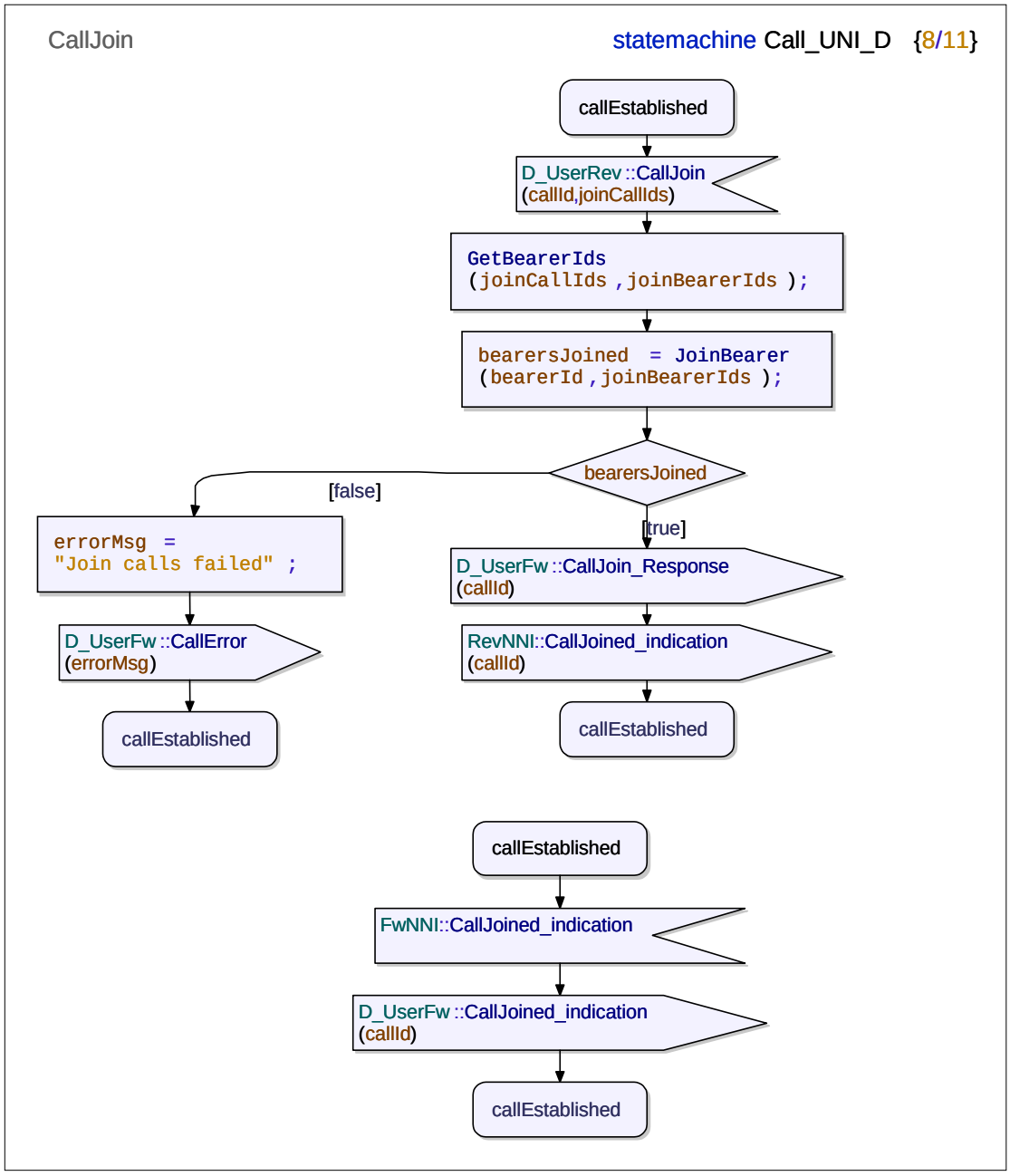


Figure 56: The destination domain call join service capability

5.4.3.7 Interrogate call

The *interrogate call* service capability returns the value of a user-specific attribute such as the contents of the call charge record to the invoking user or application. The destination domain *interrogate call* service capability is shown in figure 57.

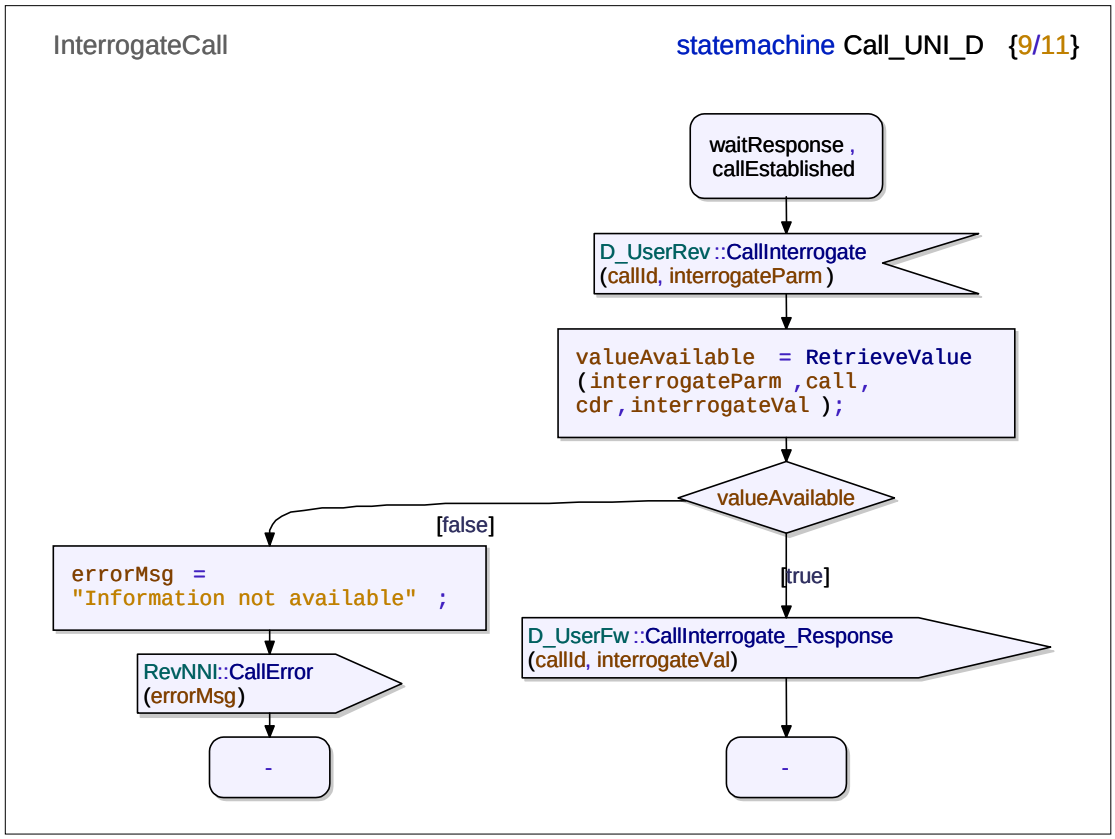


Figure 57: The destination domain interrogate call service capability

5.4.3.8 Operation signatures

The signatures of the operations used in the definition of the destination domain call group service capabilities are shown in figures 58 and 59.

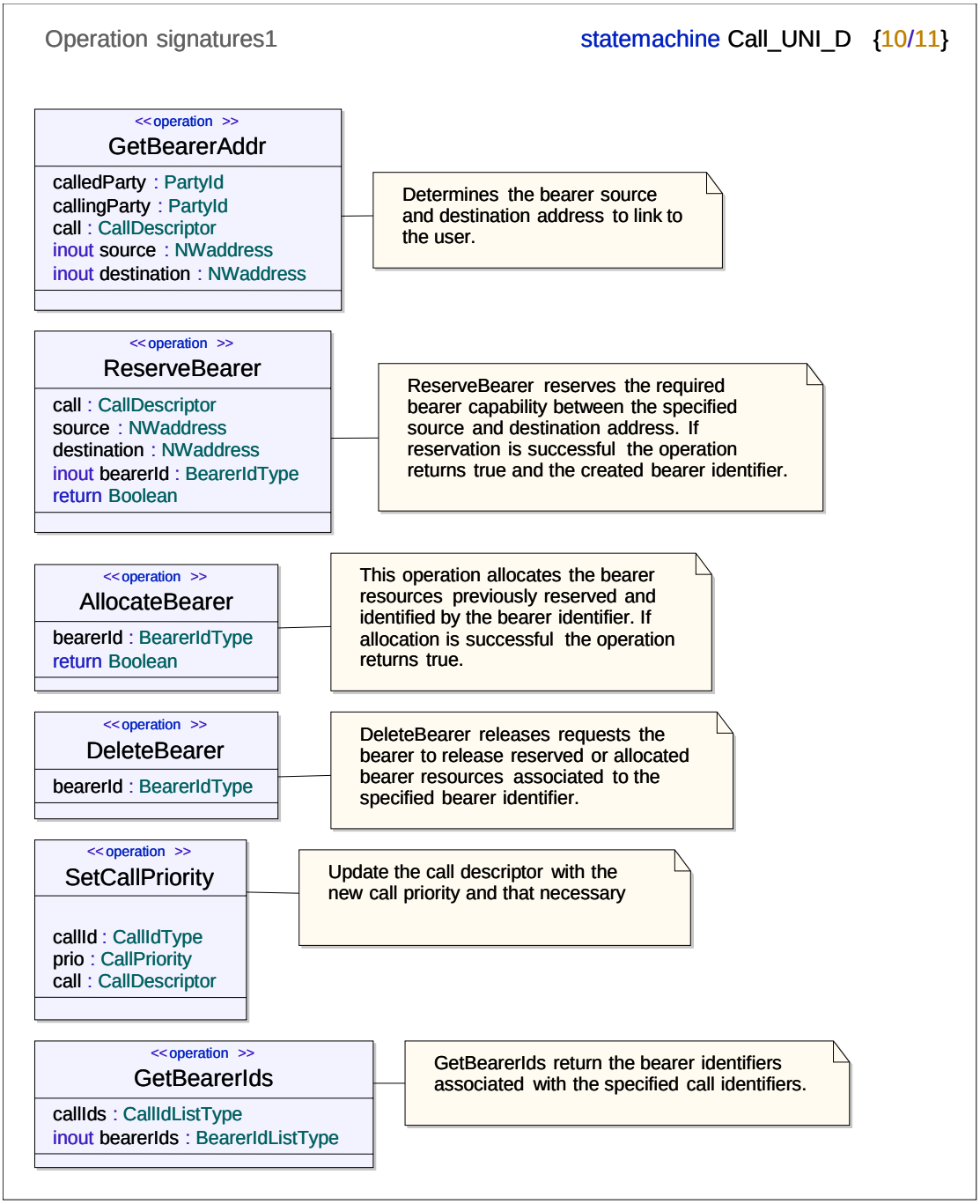


Figure 58: The destination domain operation signatures (1 of 2)

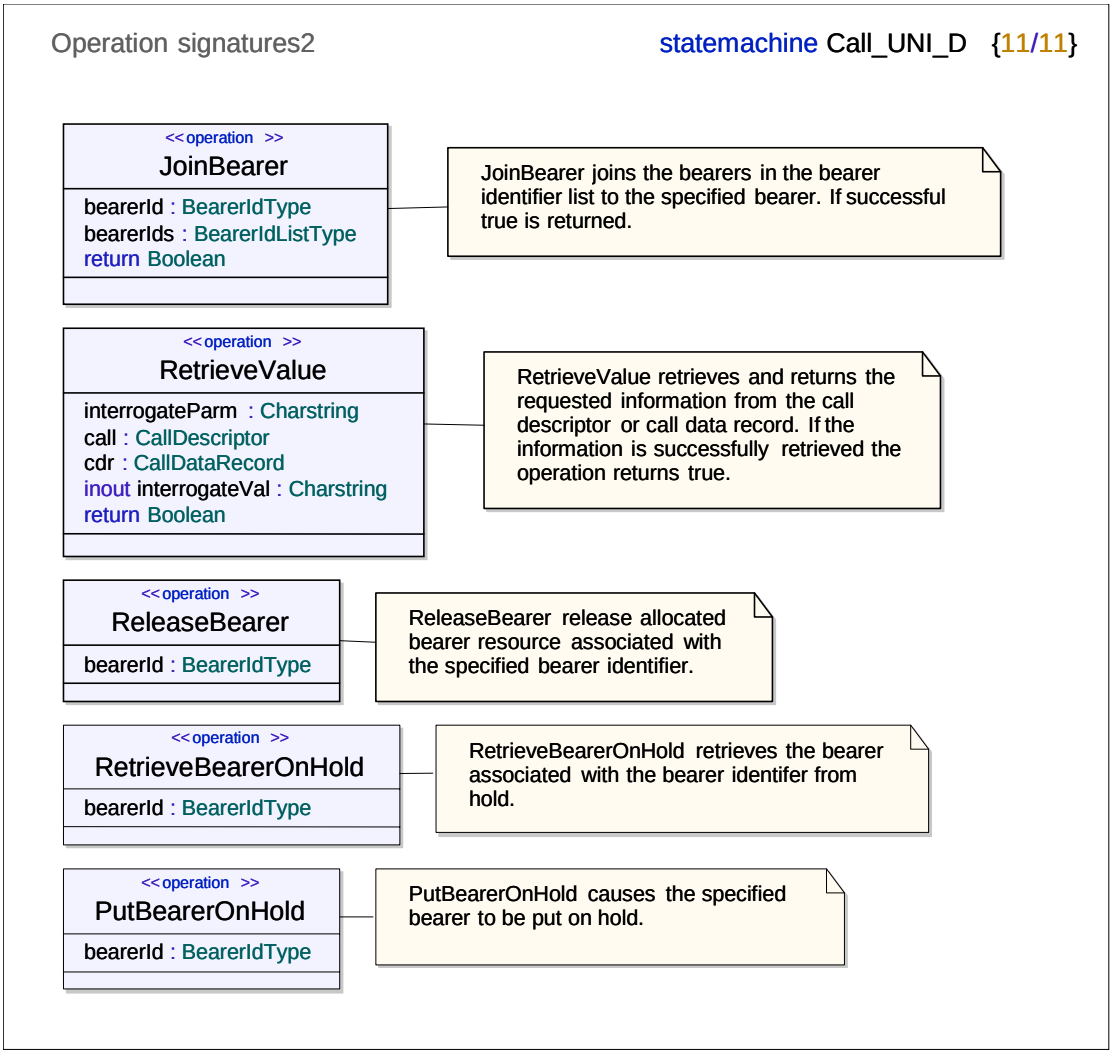


Figure 59: The destination domain operation signatures (2 of 2)

5.5 Typical architecture

Figure 60 shows a typical architectural arrangement for an object based on the Call class.

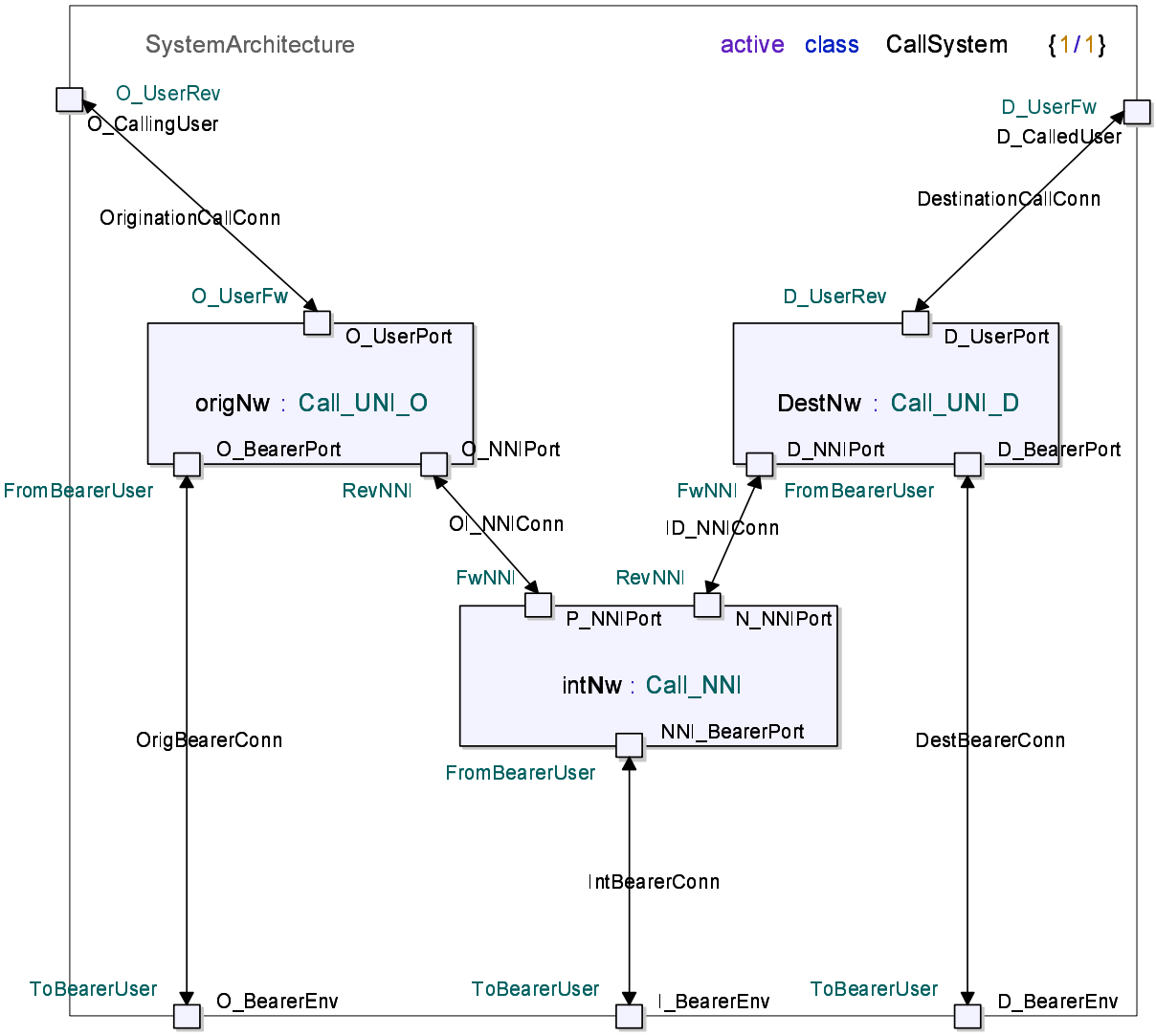


Figure 60: Typical call service architecture

6 Bearer group

6.1 Introduction

The bearer group contains those service capabilities required for the establishment, modification and release of bearer resources. The required function of each service capability is described in TR 101 878 [1].

6.2 Data model

The bearer data model derived from the description in TR 101 878 [1] is shown in figures 61 to 67.

Bearer Data_1 package BearerData {1/7}

```
public syntype UserInformationKindType = Charstring
                constants ("",
                           "speech",
                           "sound",
                           "text",
                           "facsimile",
                           "text_facsimile",
                           "videotex",
                           "video",
                           "text_interactive",
                           "sms");
```

```
public syntype SignalStructureType = Charstring
                constants ("",
                           "timeSlotSequenceIntegrity",
                           "serviceDataIntegrity",
                           "unstructured",
                           "dataSequenceIntegrity",
                           "integrity8kHZ");
```

```
public syntype TransferModeType = Charstring
                constants ("",
                           "circuit",
                           "packet",
                           "atm");
```

```
public syntype TransportParmQualifierType = Charstring
        constants ("",
                   "totalRemainingBudget",
                   "budgetAvailableForDomain");
```

```
public syntype ReservationResultType = Charstring
        constants ("",
                   "bandwidthReserved",
                   "bandwidthUnavailable",
                   "destinationUnknown");
```

```
public syntype TransportAssignResultType = Charstring
          constants ("",
                     "connectionMade",
                     "unableToConnect");
```

Figure 61: Bearer data model (1 of 7)

Bearer Data_2 package BearerData {2/7}

```
public syntype TransferCapabilityType = Charstring
                constants ("",
                           "unrestricted",
                           "digitalInformation",
                           "speech",
                           "audio3k1Hz",
                           "audio7kHz",
                           "audio15kHz",
                           "video");

public syntype CommunicationEstablishmentModeType = Charstring
                constants ("",
                           "onDemand",
                           "reserved",
                           "permanent");

public syntype BearerKindType = Charstring
                constants ("",
                           "unidirectional",
                           "bidirectionalSymmetric",
                           "bidirectionalAsymmetric");

public syntype CommunicationConfigurationType = Charstring
                 constants ("",
                            "pointToPoint",
                            "pointToMultipoint",
                            "broadcast");

public syntype DirectionType = Charstring
                 constants ("",
                            "bidirectional",
                            "rx_only",
                            "tx_only");

public choice NWaddress
              { part E164Number   e164;
                part E212Number   e212;
                part IpAddress    ip; }

public choice IpAddress
              { part IPv4Address ipv4Address;
                part IPv6Address ipv6Address; }
```

Figure 62: Bearer data model (2 of 7)

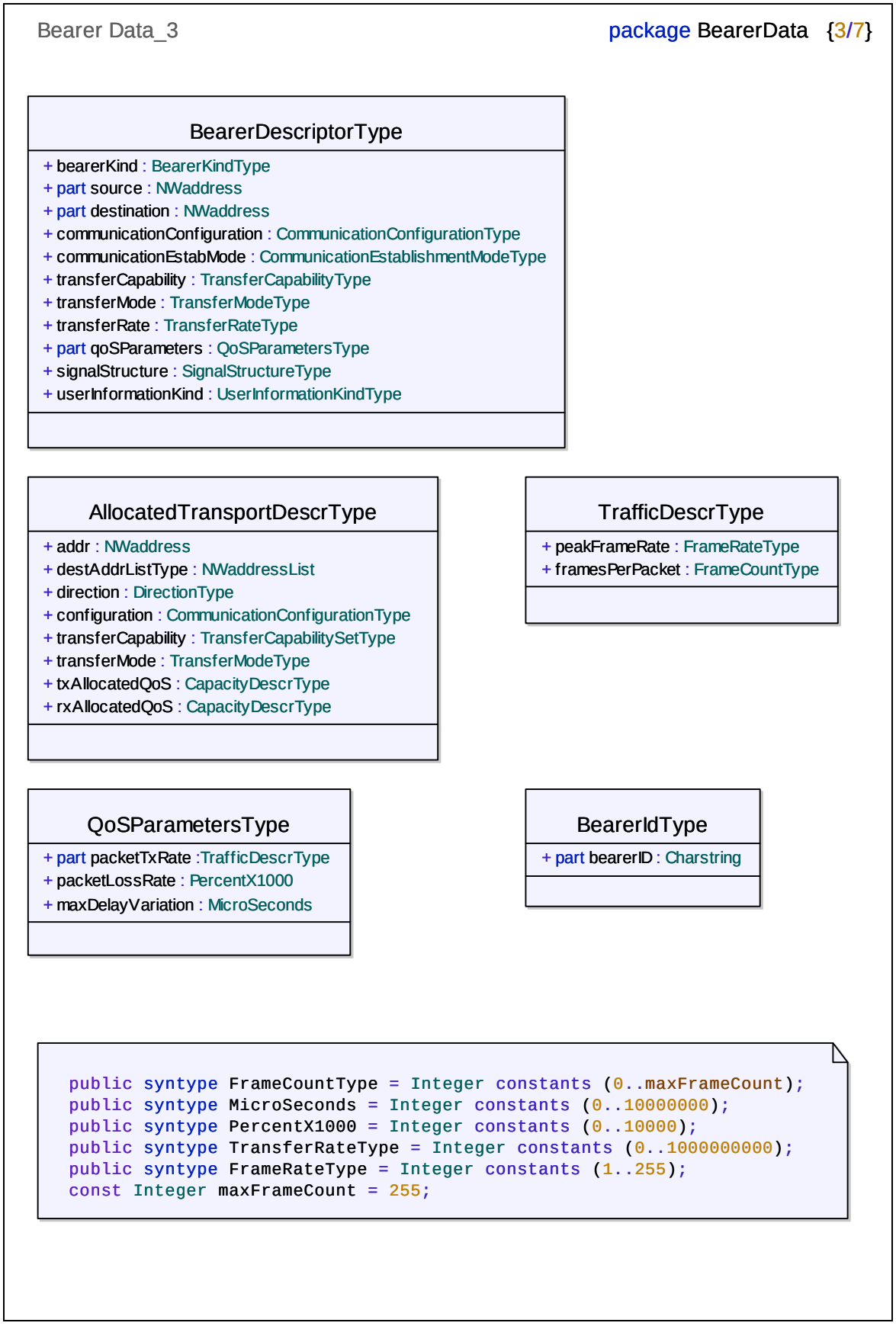


Figure 63: Bearer data model (3 of 7)

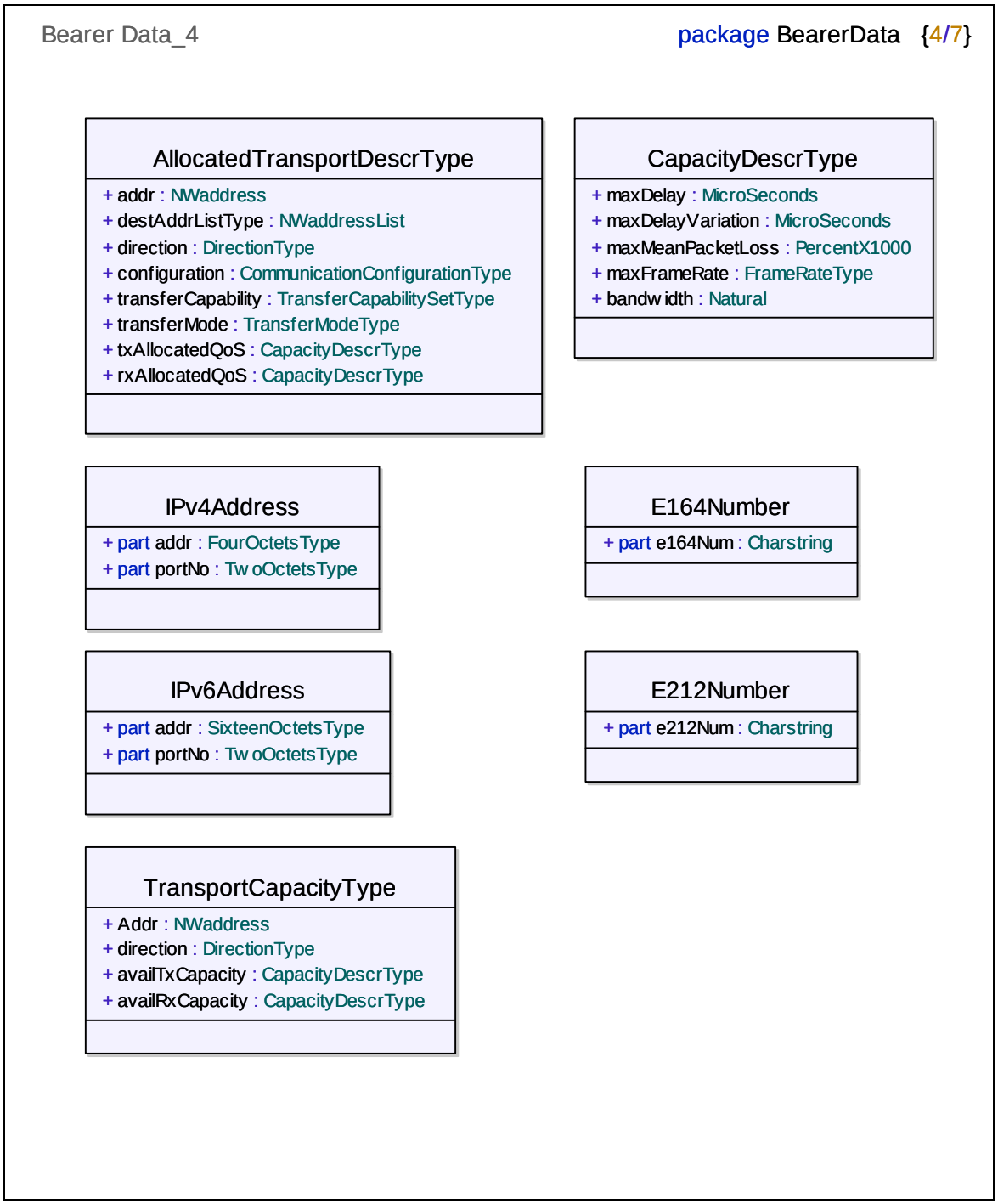


Figure 64: Bearer data model (4 of 7)

Bearer Data_5 package BearerData {5/7}

```
public syntype AllocatedTransportDescrListType
               = String<AllocatedTransportDescrType >;
public syntype FourOctetIndex
               = Natural constants (0..3);
public syntype FourOctetsType  = Array<FourOctetIndex , Octet>;
public syntype IpAddrListType  = String<IpAddress>;
public syntype NWaddressList  = String<NWaddress>;
public syntype Octet = Natural constants (0..255);
public syntype SixteenOctetIndex  = Natural constants (0..15);
public syntype SixteenOctetsType  = Array<SixteenOctetIndex , Octet>;
public syntype TransferCapabilitySetType =PowerSet<TransferCapabilityType >;
public syntype TransportCapacityListType  = String<TransportCapacityType >;
public syntype TwoOctetIndex  = Natural constants (0..1);
public syntype TwoOctetsType  = Array<TwoOctetIndex , Octet>;
```

Figure 65: Bearer data model (5 of 7)

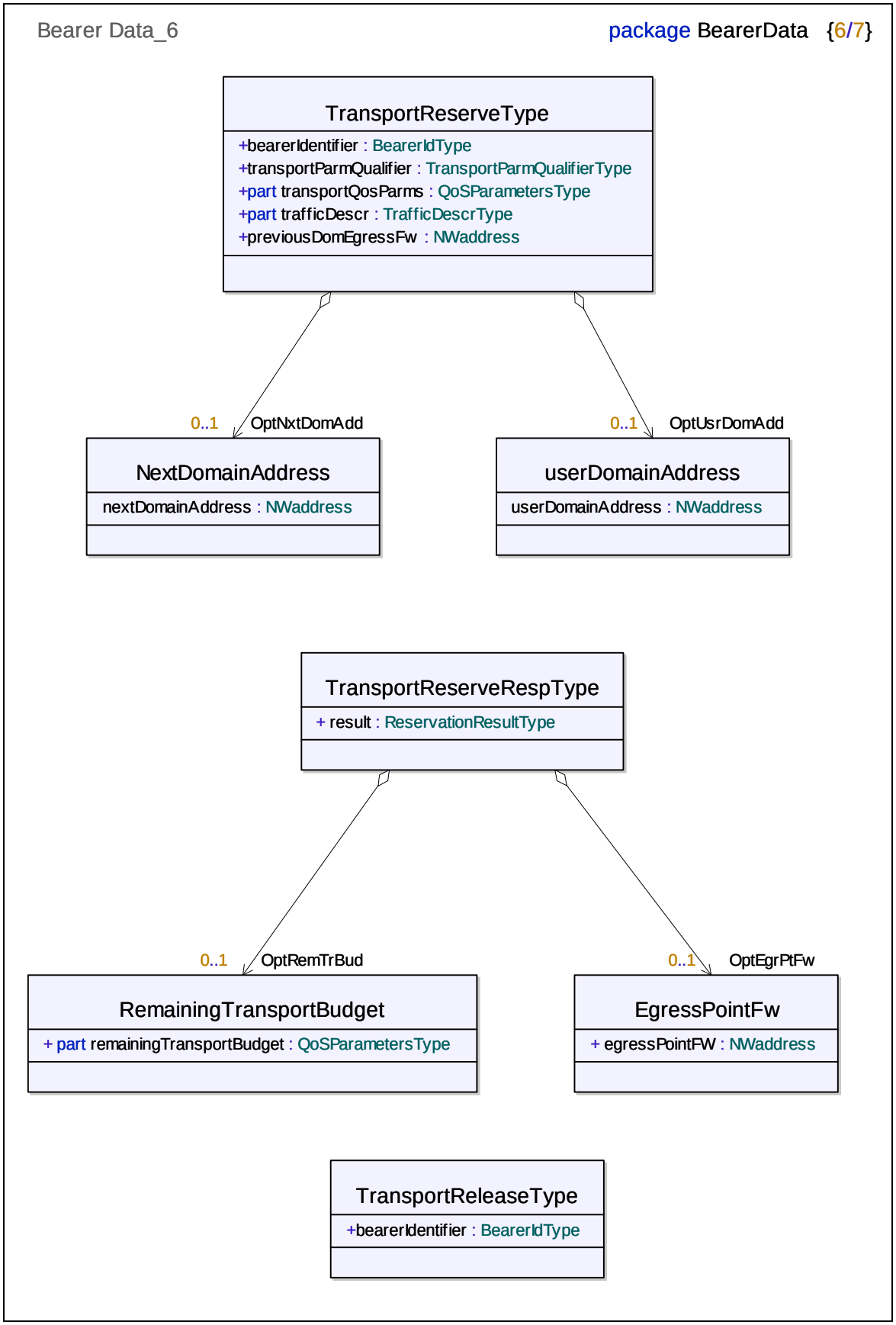


Figure 66: Bearer data model (6 of 7)

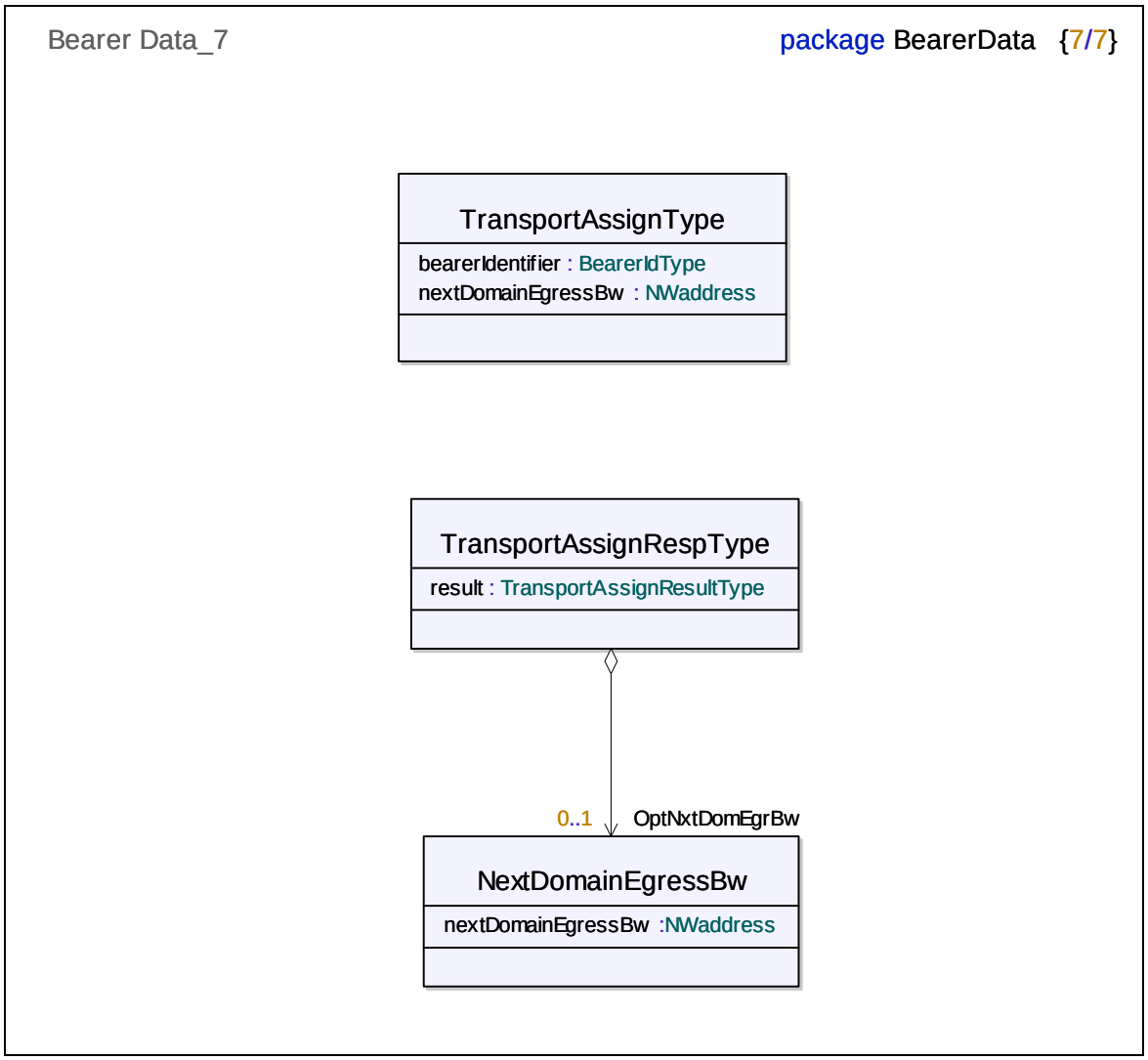


Figure 67: Bearer data model (7 of 7)

6.3 Bearer group service capabilities

The attributes and service capabilities associated with the Media group class are shown in figures 68 and 69.

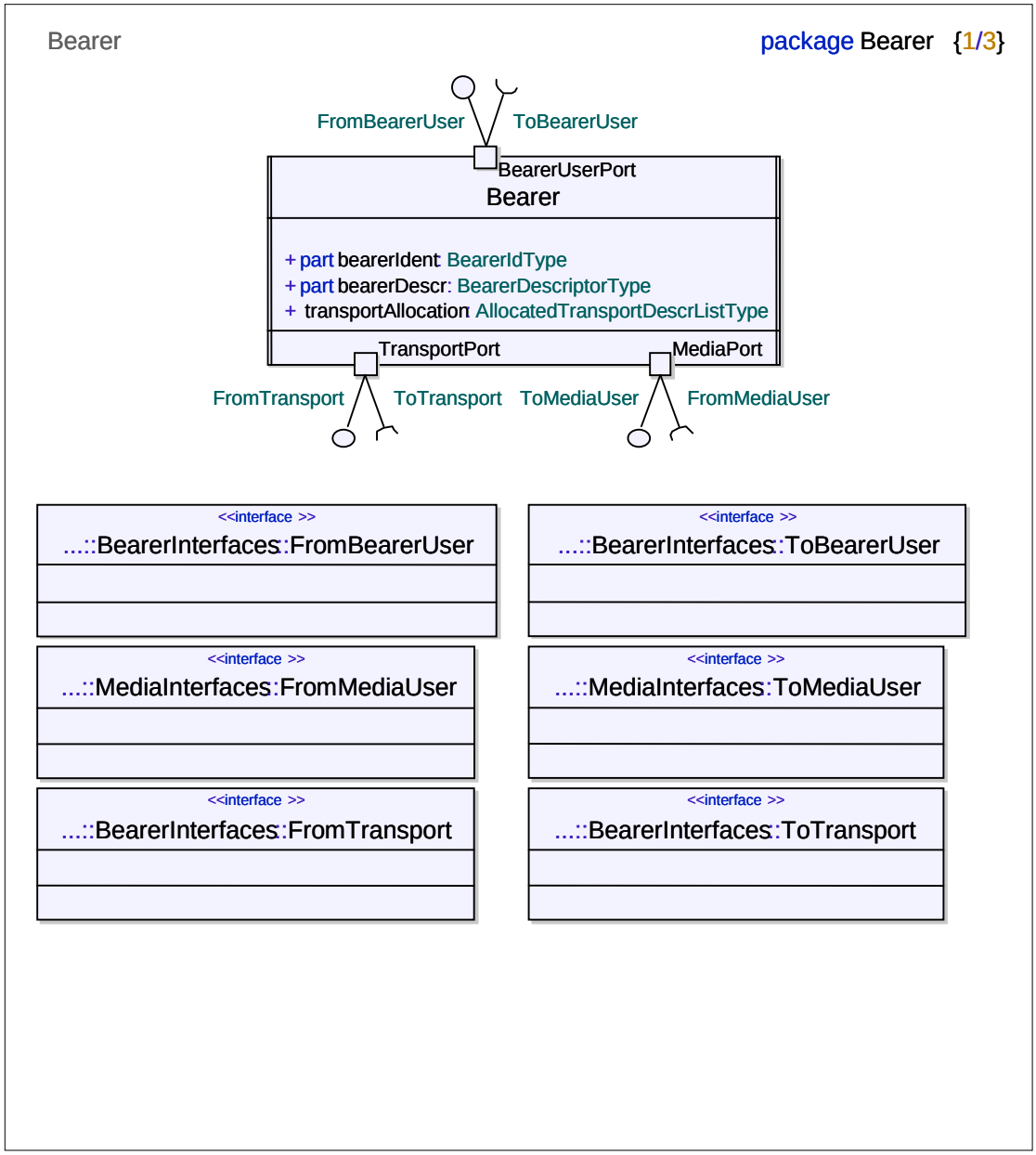


Figure 68: The bearer class showing attributes and interfaces

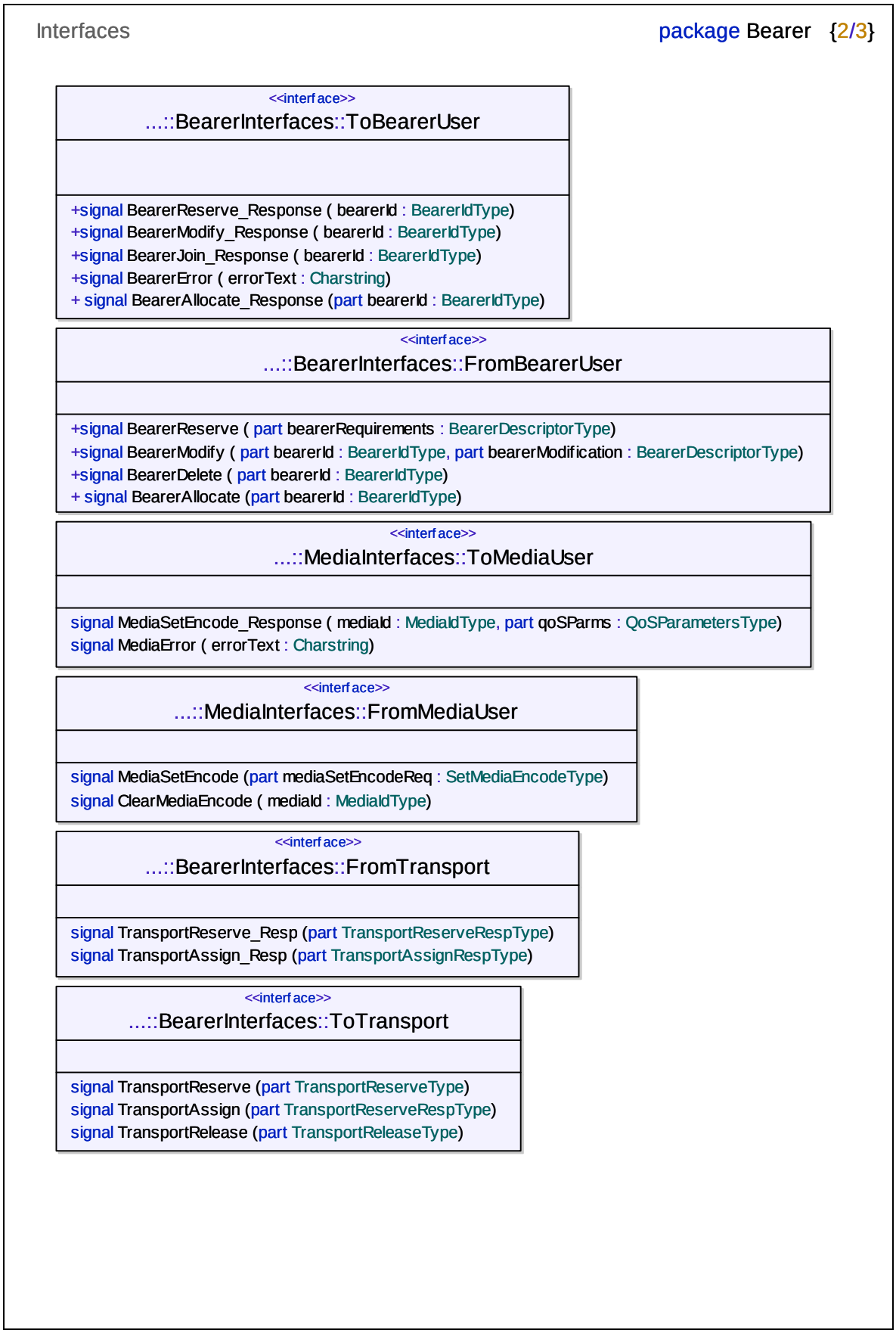


Figure 69: The interfaces of the bearer class showing details of the signals supported by each

The initialization of bearer processing is shown in figure 70.

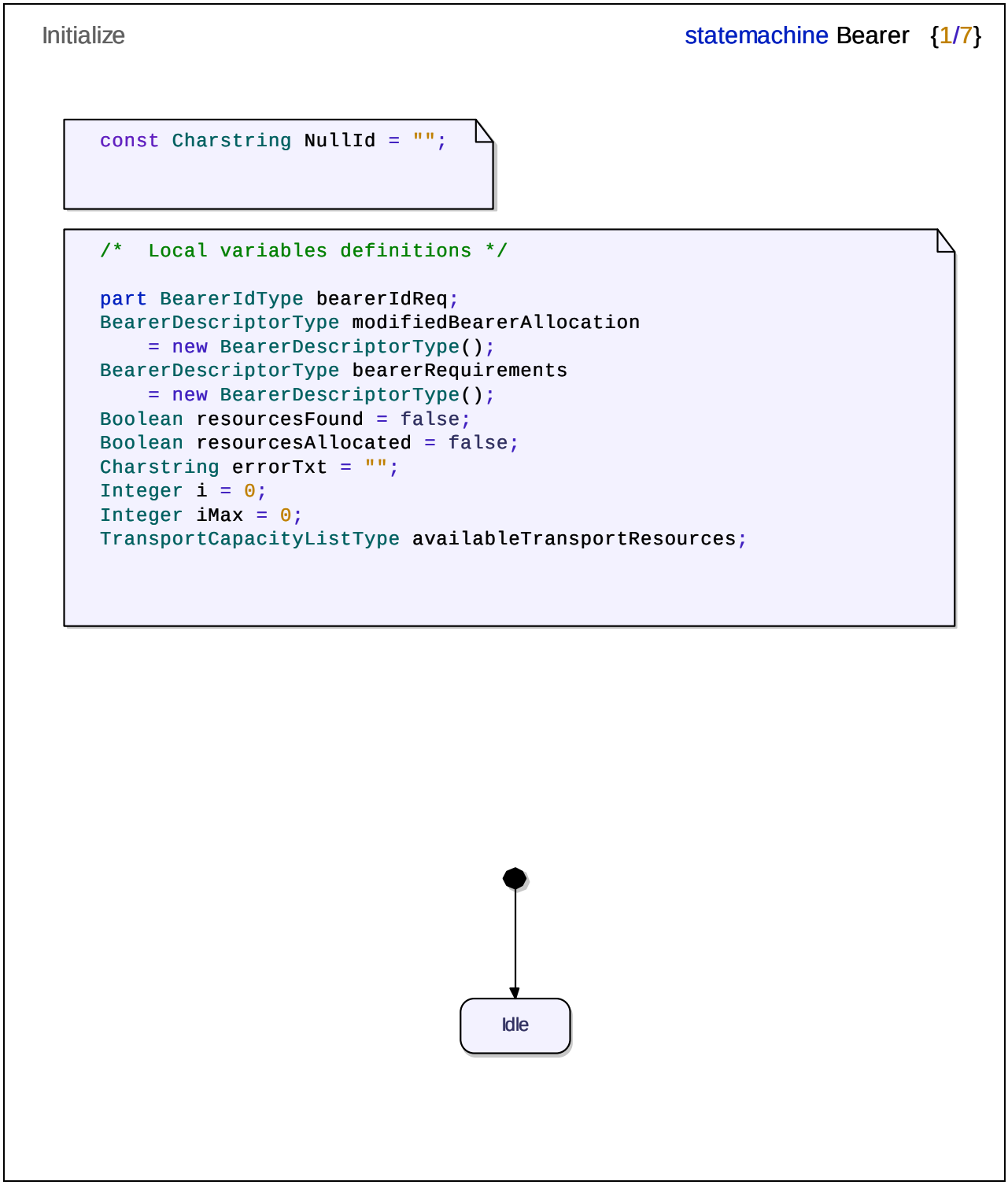


Figure 70: Bearer initialization

6.3.1 Create

The *Bearer Create* service capability can be further sub-divided into two capabilities, *Reserve Bearer* and *Allocate Bearer* .

6.3.1.1 Reserve Bearer

The *Reserve Bearer* service capability assigns the necessary bearer resources, if available, to a call but does not complete the connection. The service capability is specified in figure 71.

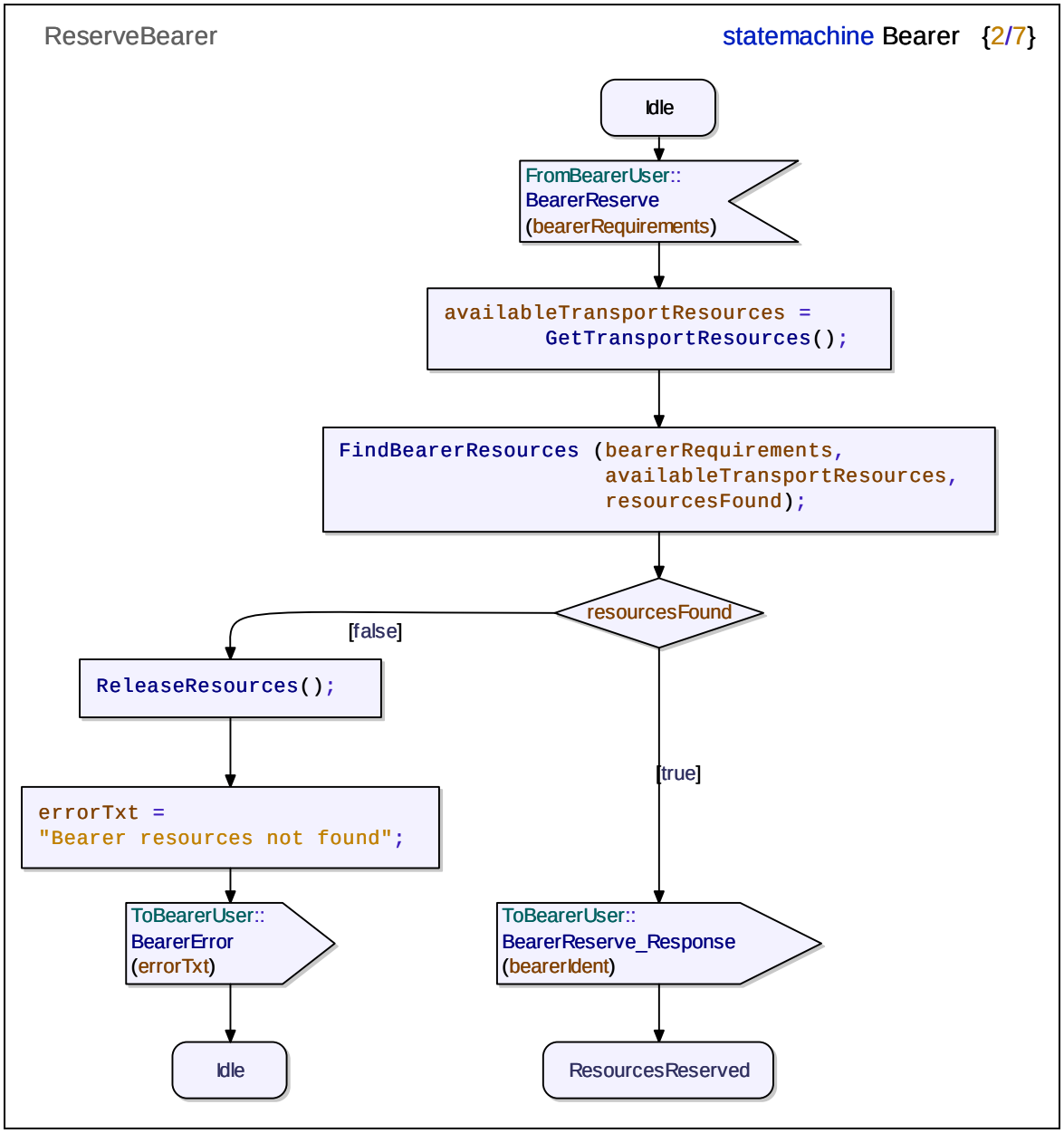


Figure 71: The reserve bearer service capability

6.3.1.2 Allocate Bearer

The *Allocate Bearer* service capability completes the connection of previously reserved bearer resources. The service capability is specified in figure 72.

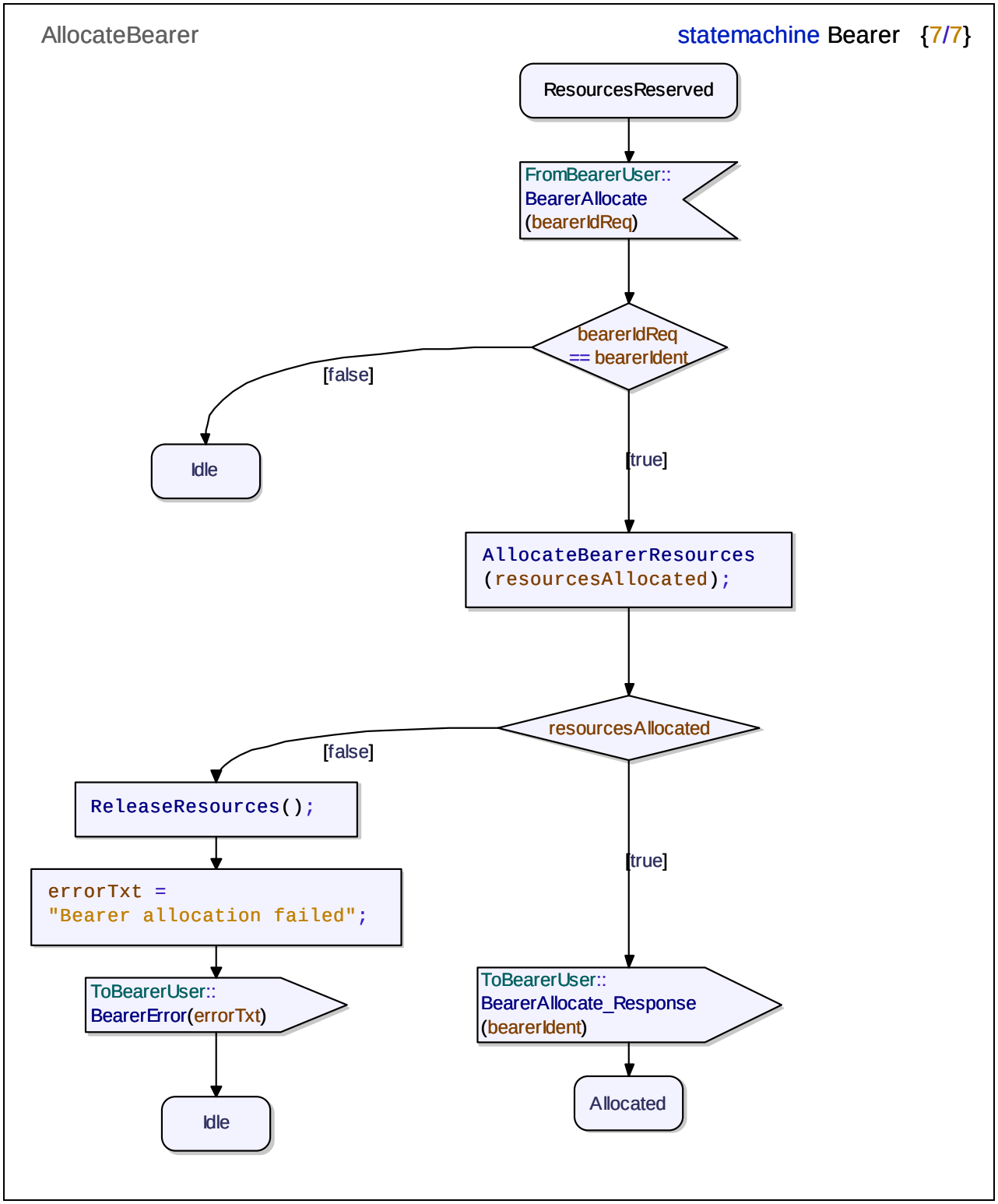


Figure 72: The allocate bearer service capability

6.3.2 Modify Bearer

The *Modify Bearer* service capability assigns the bearer resources, if available, required to alter the bearer capabilities of an established call.

NOTE: This service capability does not complete the connection of the modified bearer resources. This can be achieved by invocation of the *Allocate Bearer* service capability.

The service capability is specified in figures 73 and 74.

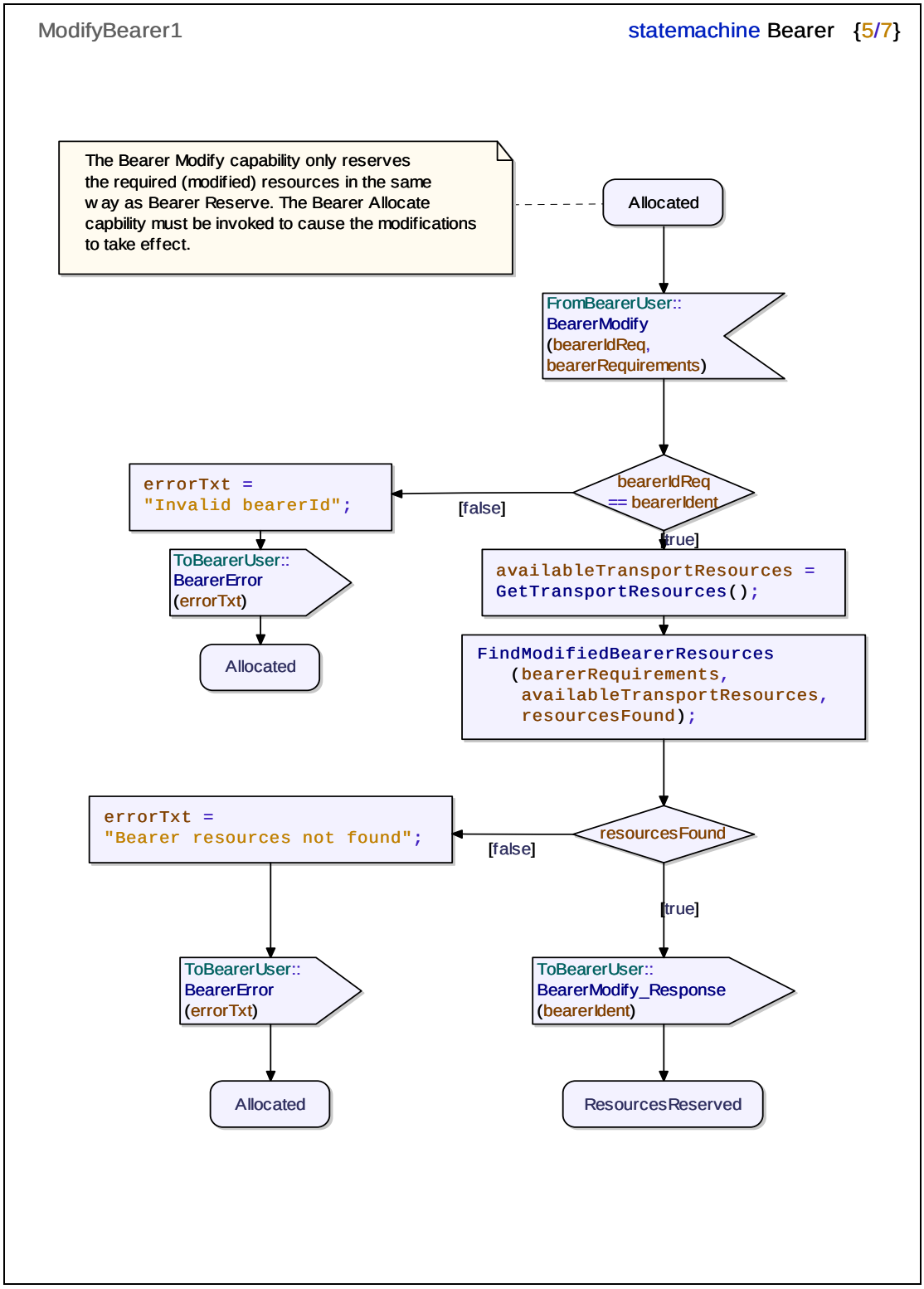


Figure 73: The modify bearer service capability (1 of 2)

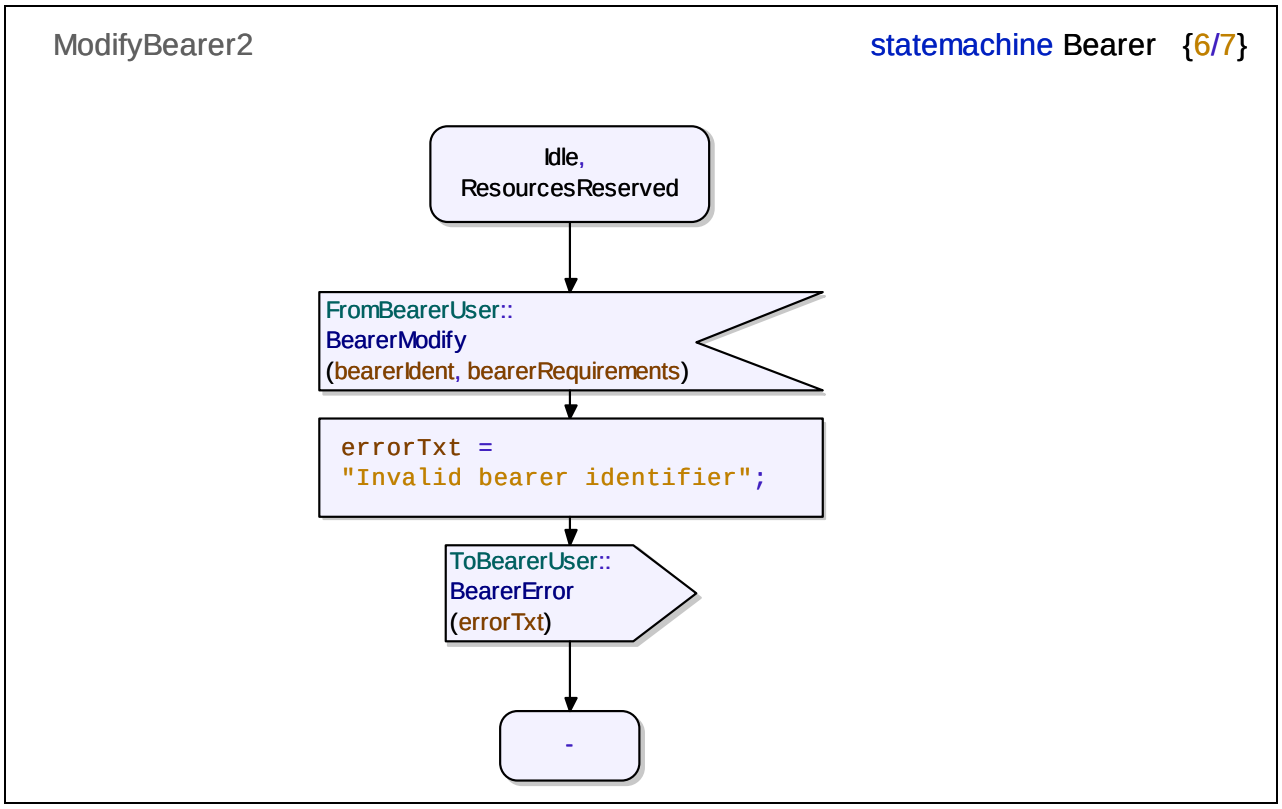


Figure 74: The Modify bearer service capability (2 of 2)

6.3.3 Delete Bearer

The *Delete Bearer* service capability releases all previously reserved and allocated bearer resources associated with a particular call. The service capability is specified in figure 75.

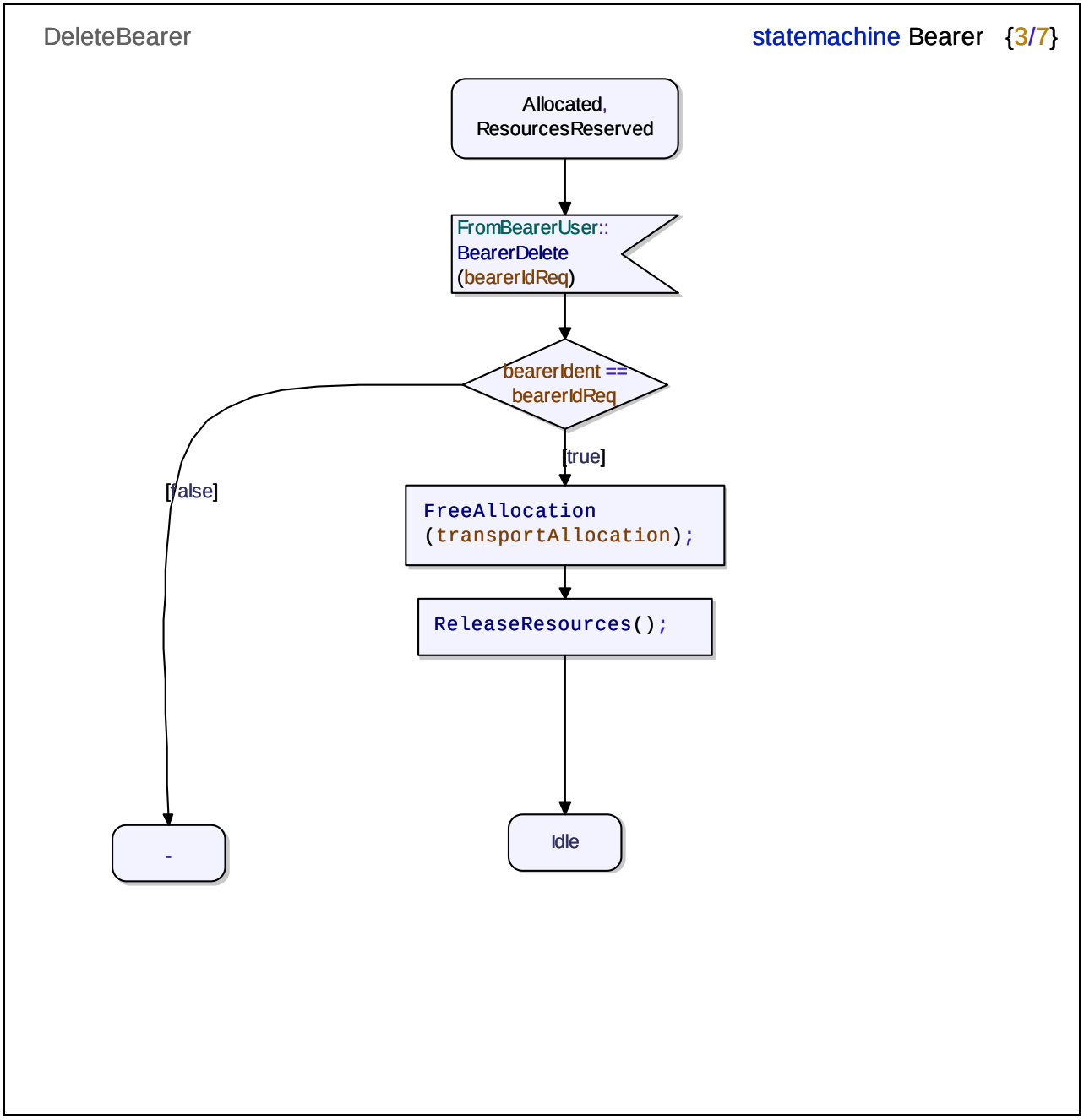


Figure 75: The delete bearer service capability

6.3.4 Operation interfaces

Figure 76 shows the parameter interfaces to the operations of the bearer service capabilities.

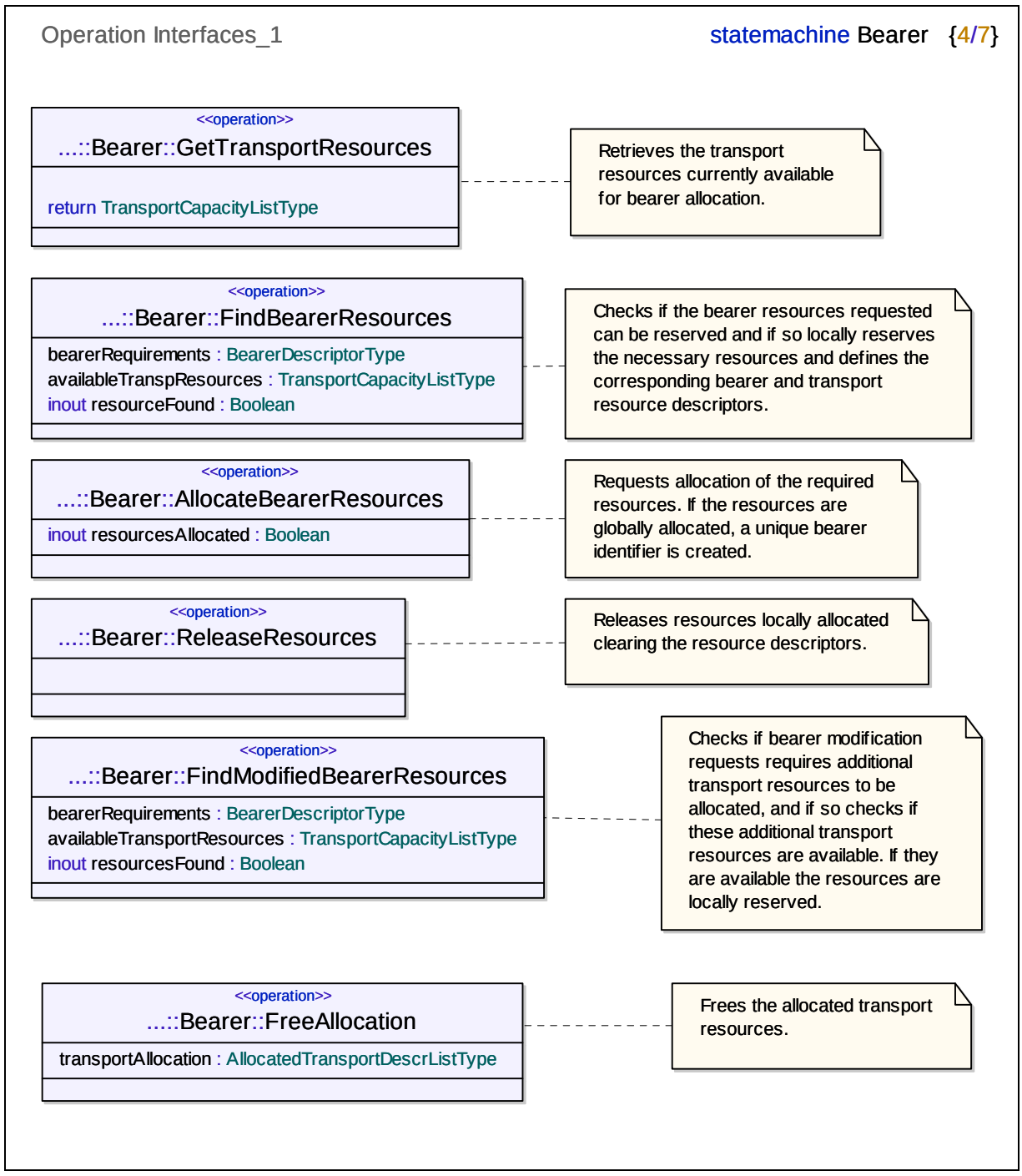


Figure 76: Bearer operation interfaces

6.4 Typical architecture

Figure 77 shows a typical architectural arrangement for an object based on the Bearer class.

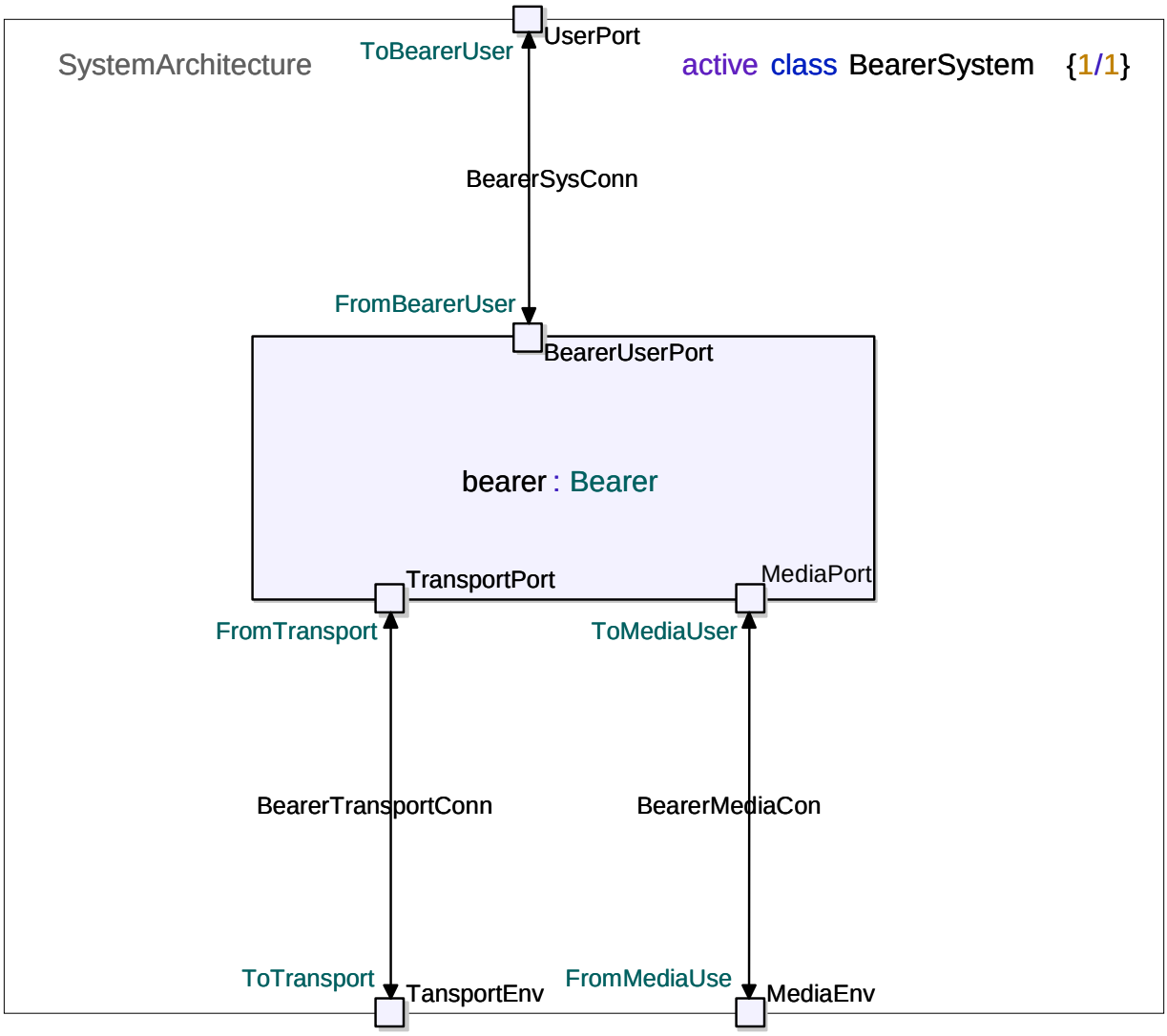


Figure 77: Typical bearer service architecture

7 Media group

7.1 Introduction

The media group contains those service capabilities required to establish and teardown media encoding facilities.

7.2 Data model

The media data model derived from the description in TR 101 878 [1] is shown in figures 78, 79, and 80.

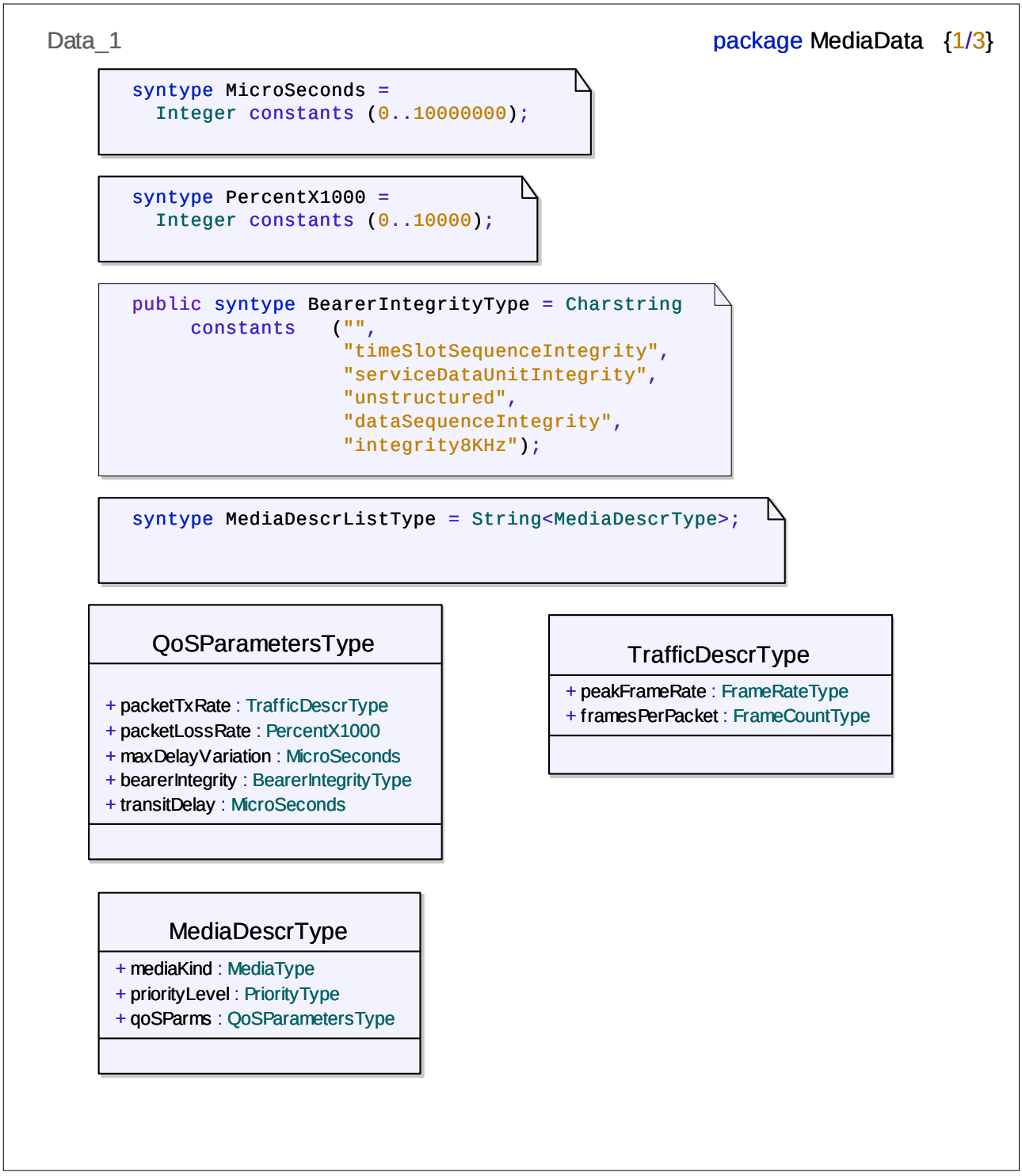


Figure 78: Media data model (1 of 3)

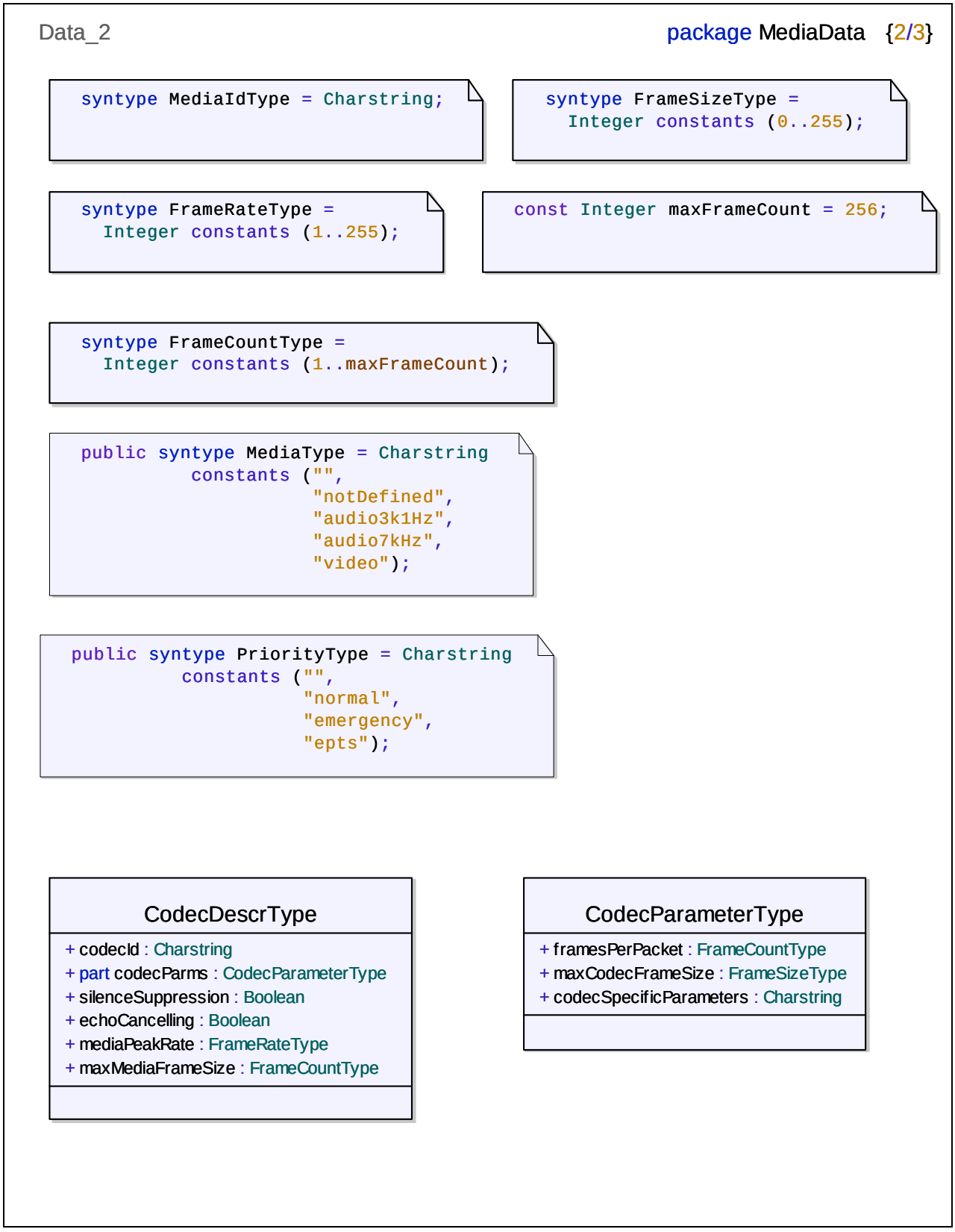


Figure 79: Media data model (2 of 3)

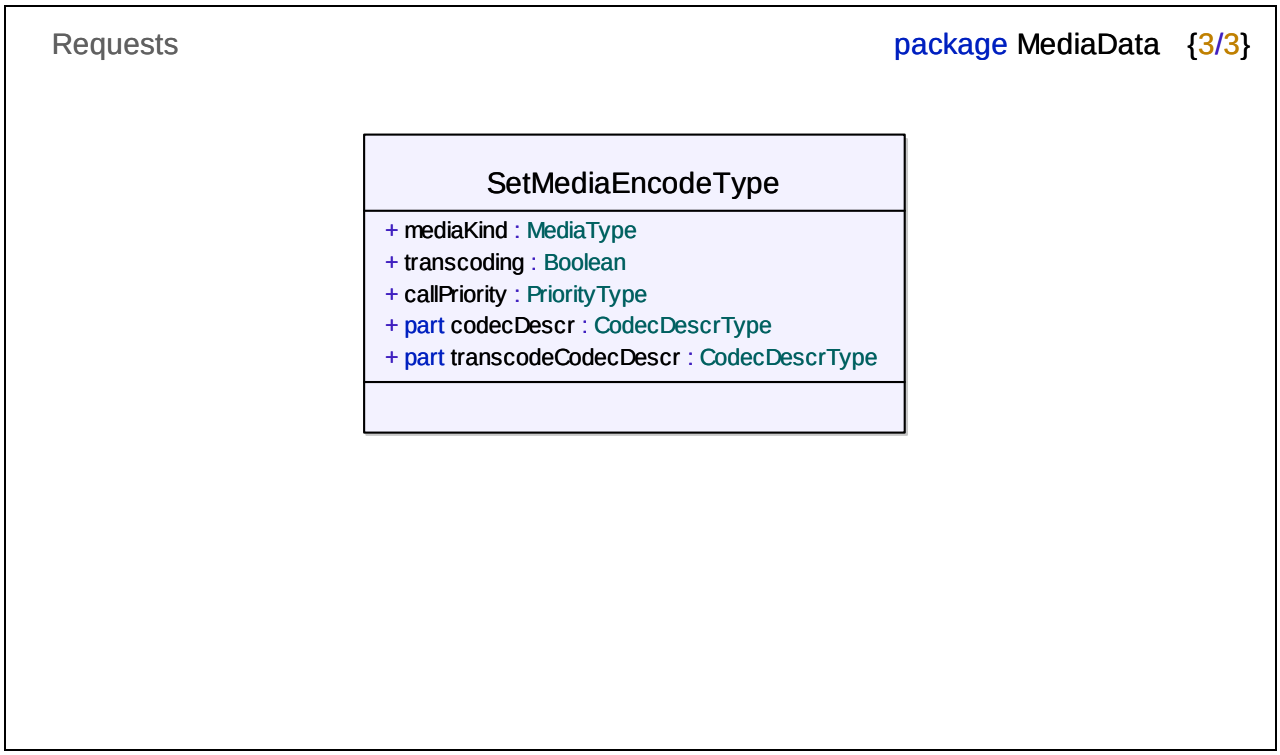


Figure 80: Media data model (3 of 3)

7.3 Media group service capabilities

The attributes and service capabilities associated with the media group class are shown in figures 81 and 82.

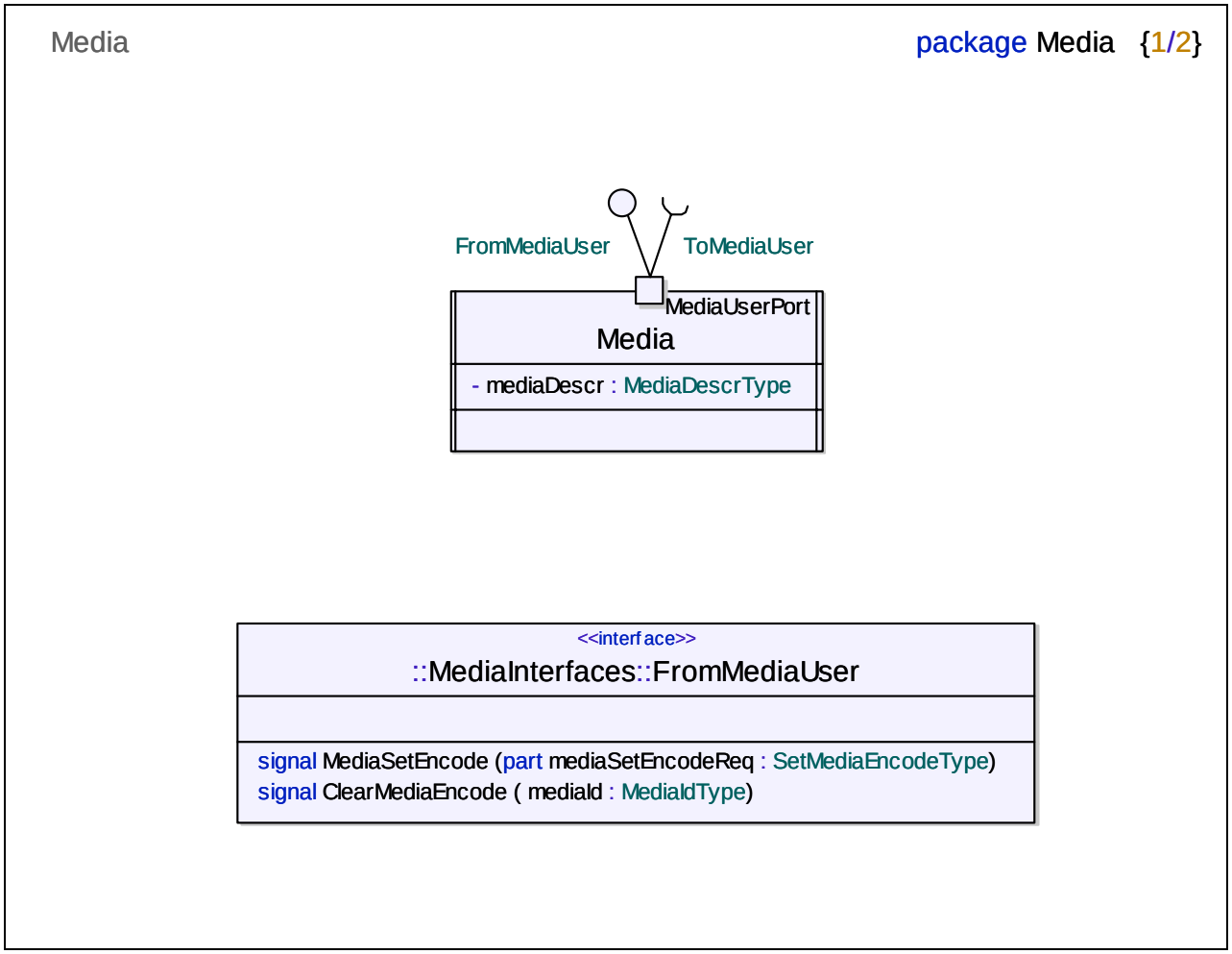


Figure 81: Media class with attributes and service capabilities as signals in the interface from media user

The initialization of the media group service capabilities is shown in figure 82

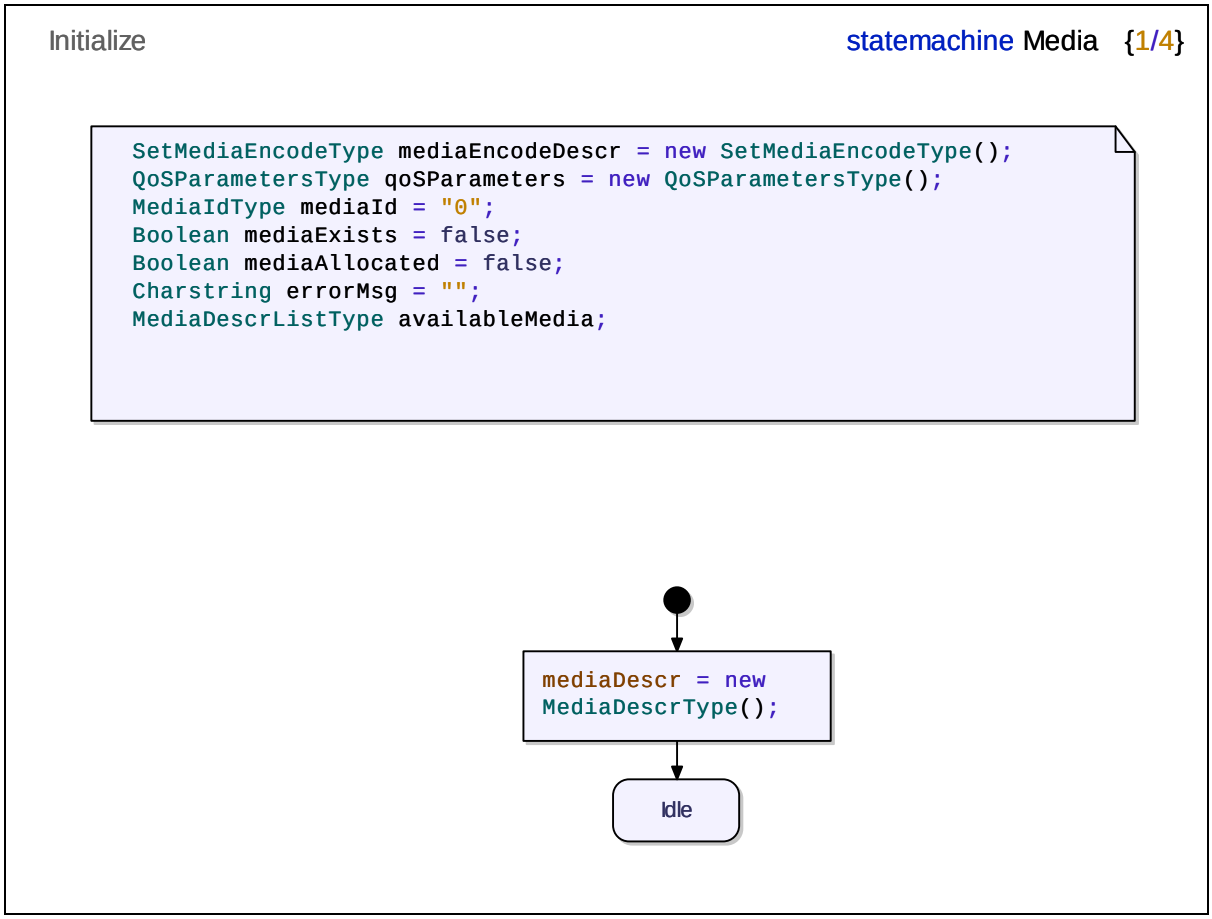


Figure 82: The Media group service capabilities common initialization

7.3.1 Set media encode

The *set media encode* service capability establishes the media encoding and decoding requirements for a particular media type. These requirements are characterized by the information elements in the supplied media attributes. The service capability is specified in figure 83.

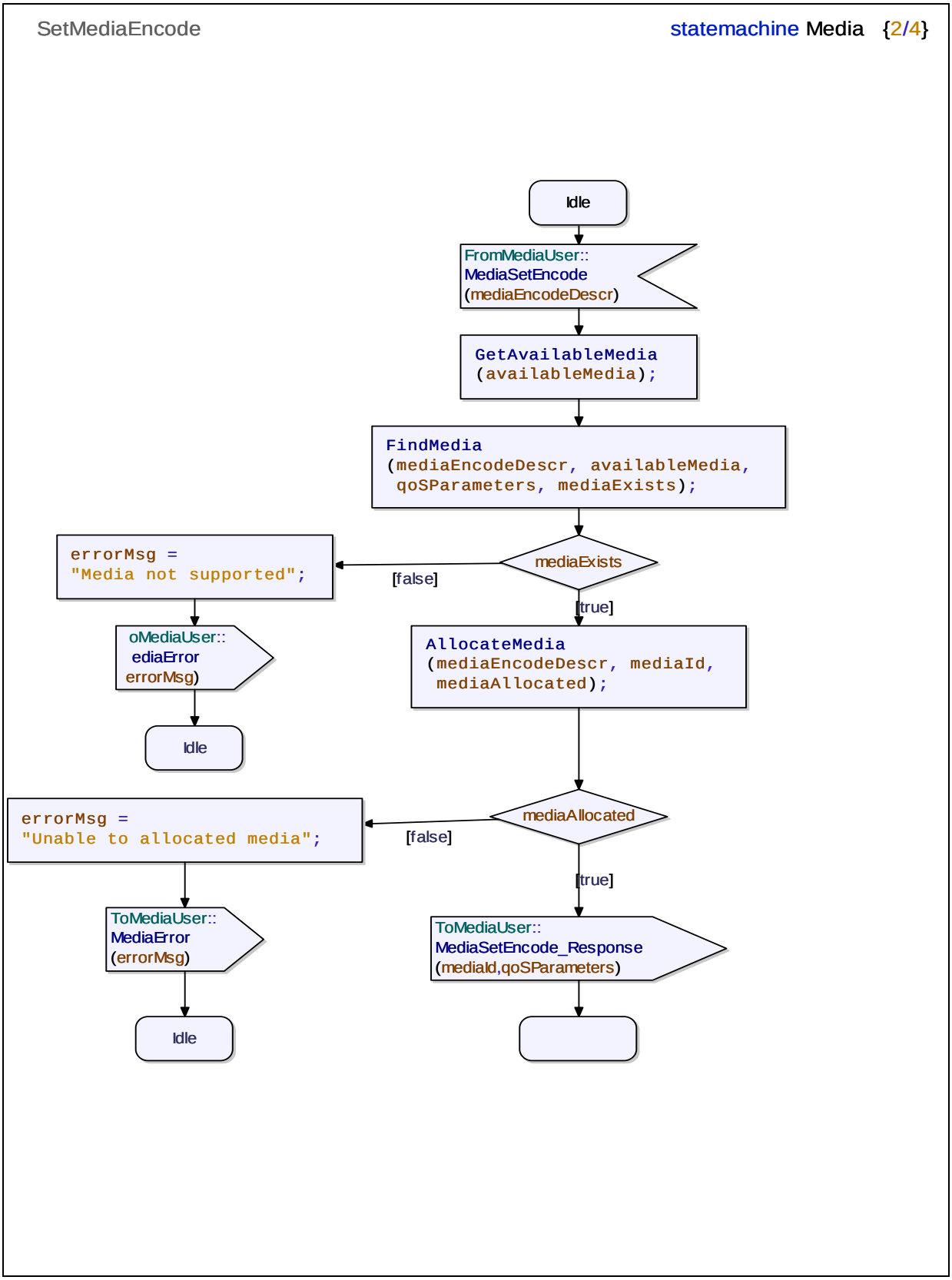


Figure 83: The set media encode service capability

7.3.2 Clear media encode

The *clear media encode* service capability releases any media encoding and decoding resources allocated by the *set media encode* service capability. The *clear media encode* service capability requirements are specified in figure 84.

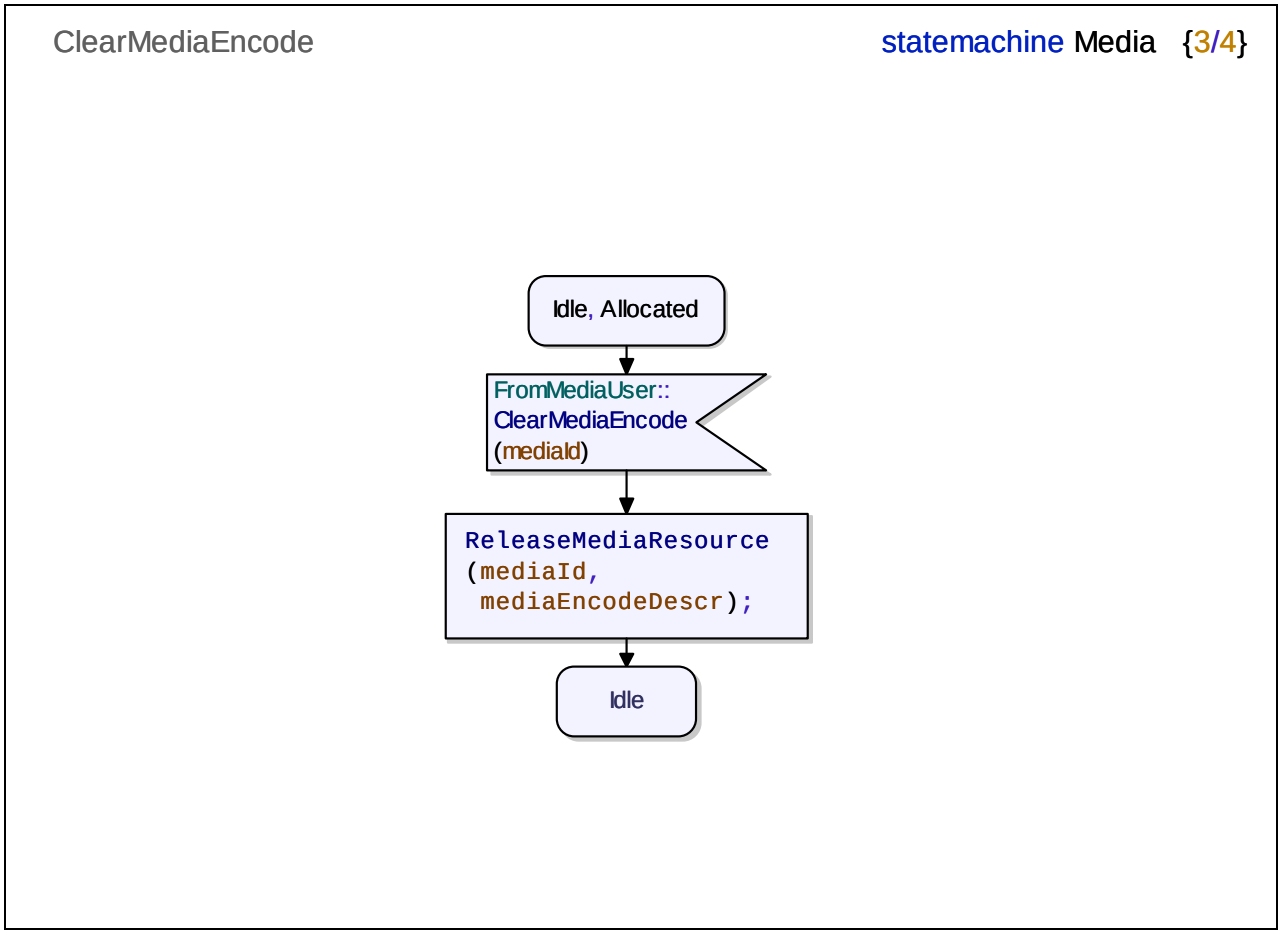


Figure 84: The set media encode service capability

7.3.3 Operation interfaces

Figure 85 shows the parameter interfaces to the operations of the media service capabilities.

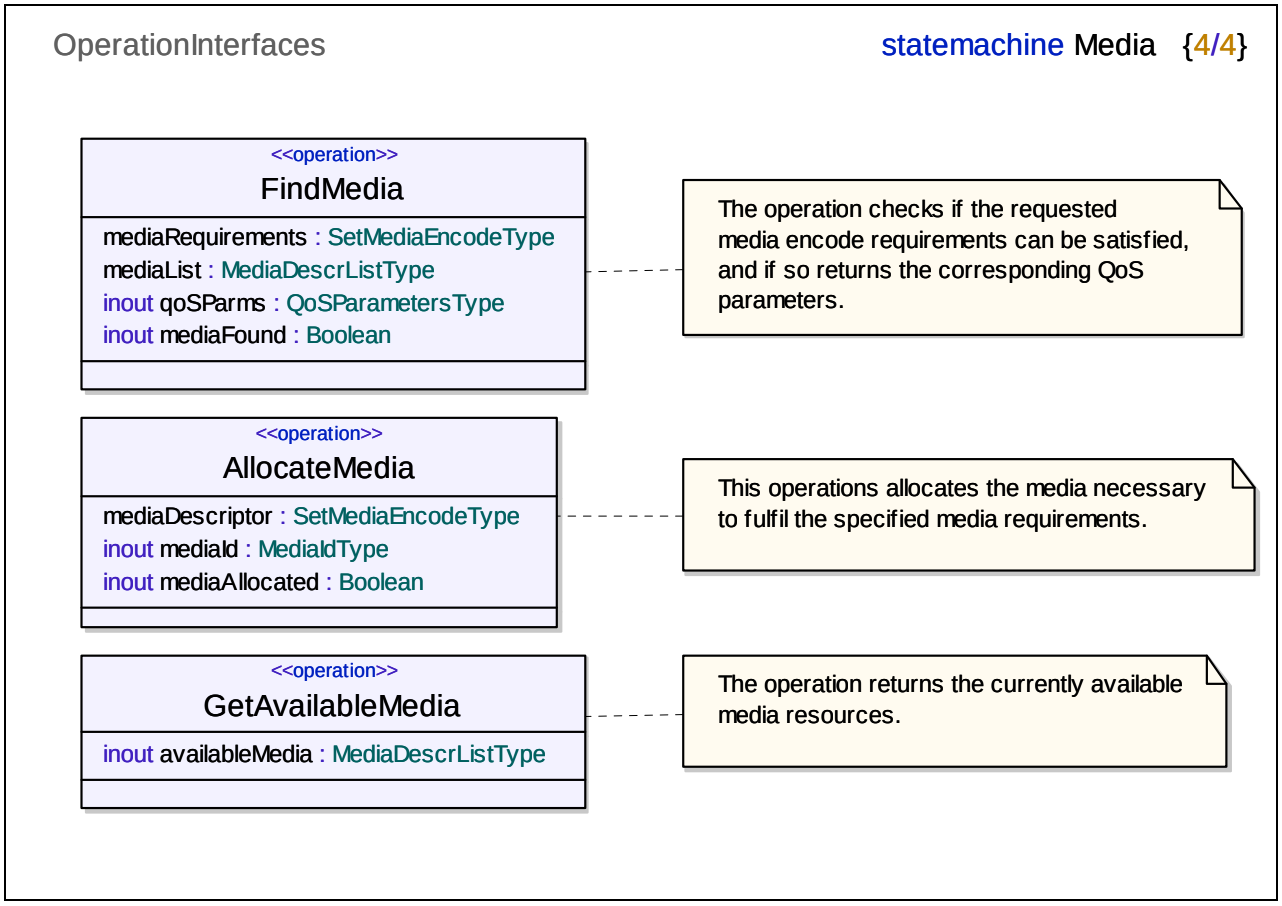


Figure 85: Media operation interfaces

7.4 Typical architecture

Figure 86 shows a typical architectural arrangement for an object based on the media class.

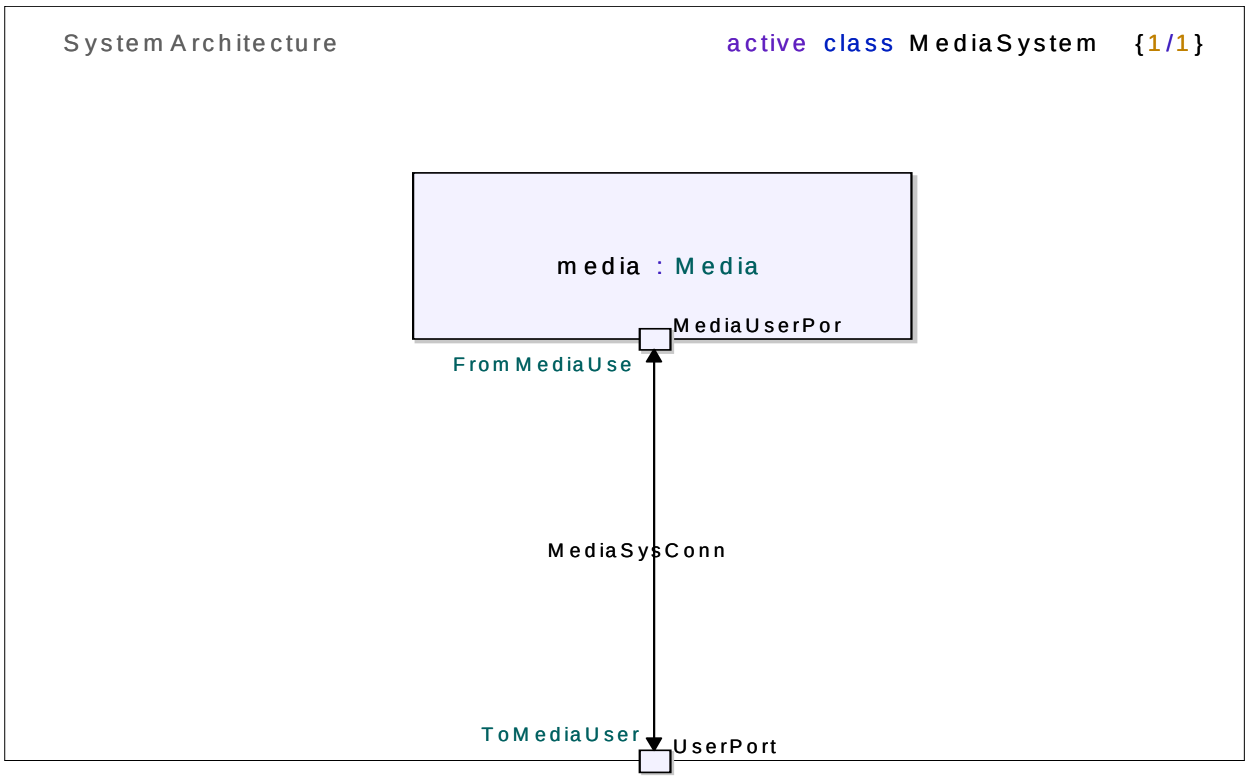


Figure 86: Typical media service architecture

8 Message group

8.1 Introduction

The message group contains those service capabilities required for control of message handling including storage, retrieval and maintenance. The required function of each service capability is described in TR 101 878 [1].

8.2 Data model

The data model derived from the description in TR 101 878 [1] is shown in figures 87, 88, and 89.

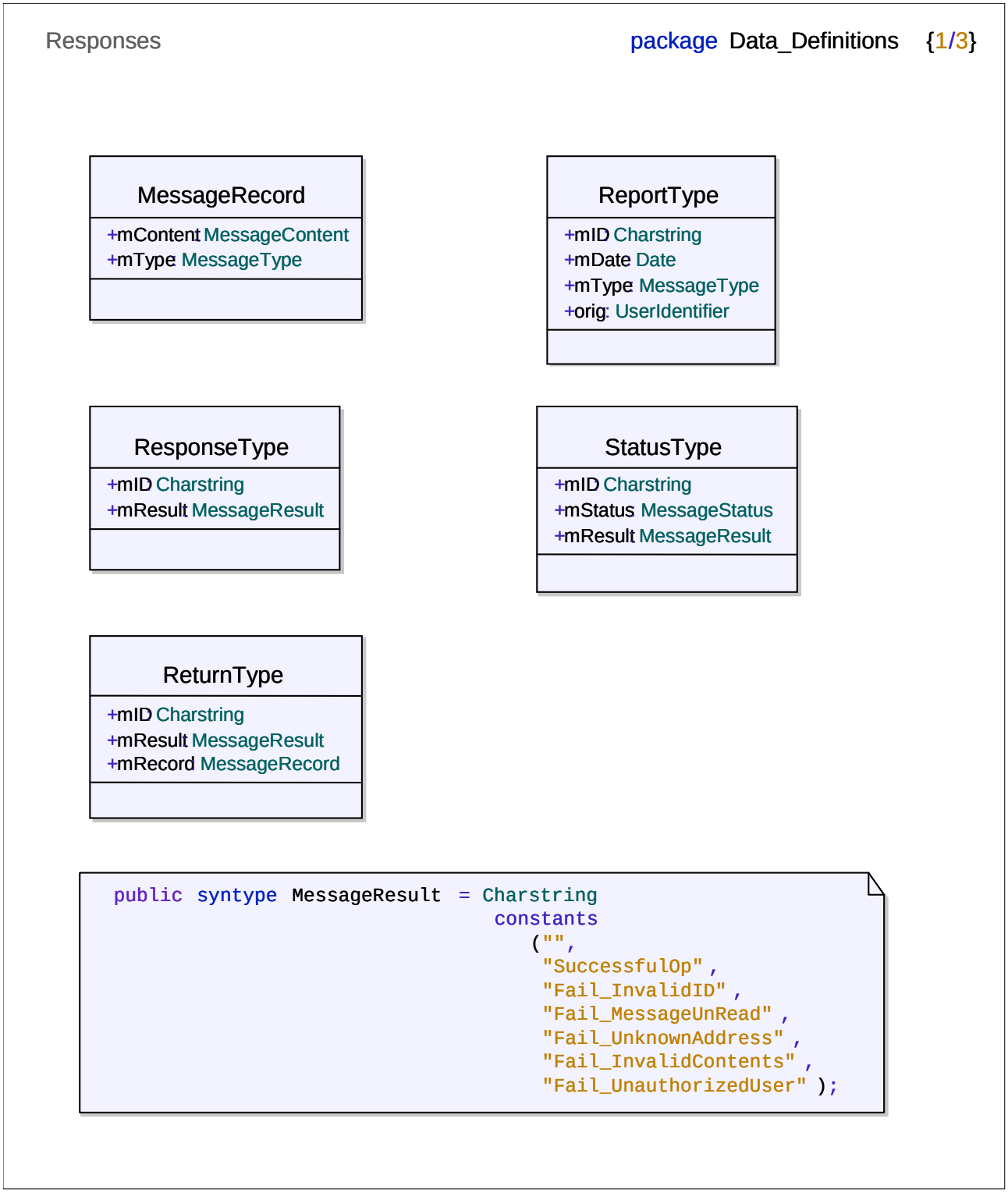


Figure 87: Message data model, response types

General package Data_Definitions {2/3}

```
public syntype UserIdentifier = Charstring;
public syntype Date = Integer;
public syntype MessageContent = Charstring;
public syntype MessageStatus = Charstring
                               constants
                                 ("",
                                  "Read",
                                  "UnRead");
public syntype MessageType   = Charstring
                               constants
                                 ("",
                                  "DialledMessage",
                                  "EmptyMessage",
                                  "TextMessage",
                                  "MediaMessage");
```

Figure 88: Message data model, general types

Requests package Data_Definitions {3/3}

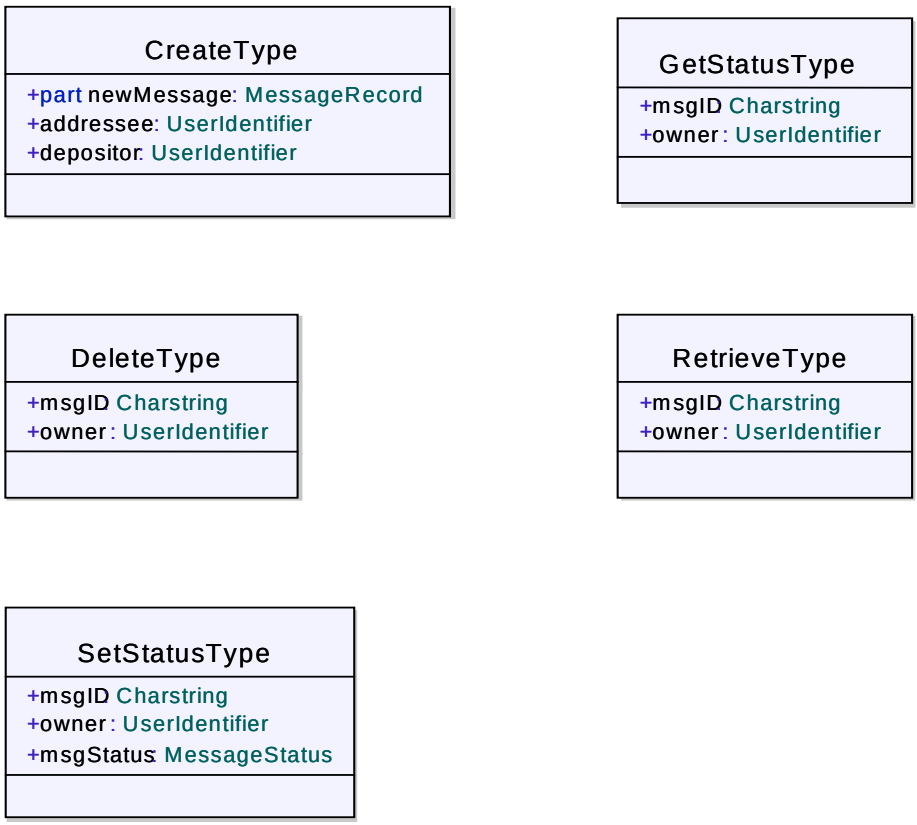


Figure 89: Message data model, request types

8.3 Message group service capabilities

The attributes and service capabilities associated with the message group class are shown in figure 90.

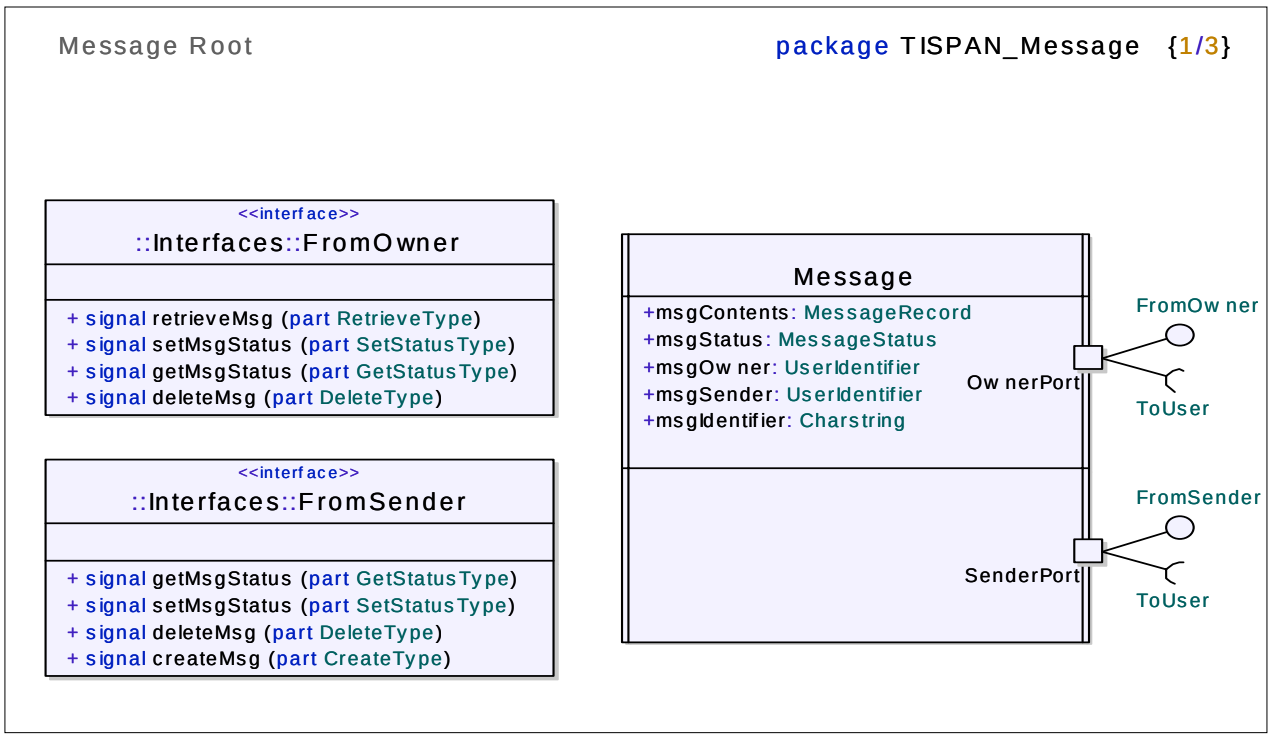


Figure 90: Message class with attributes and service capabilities as signals

Figure 90 also shows the four interfaces to the message class, FromOwner, FromSender, ToOwner and ToSender which carry signals as defined in figure 91.

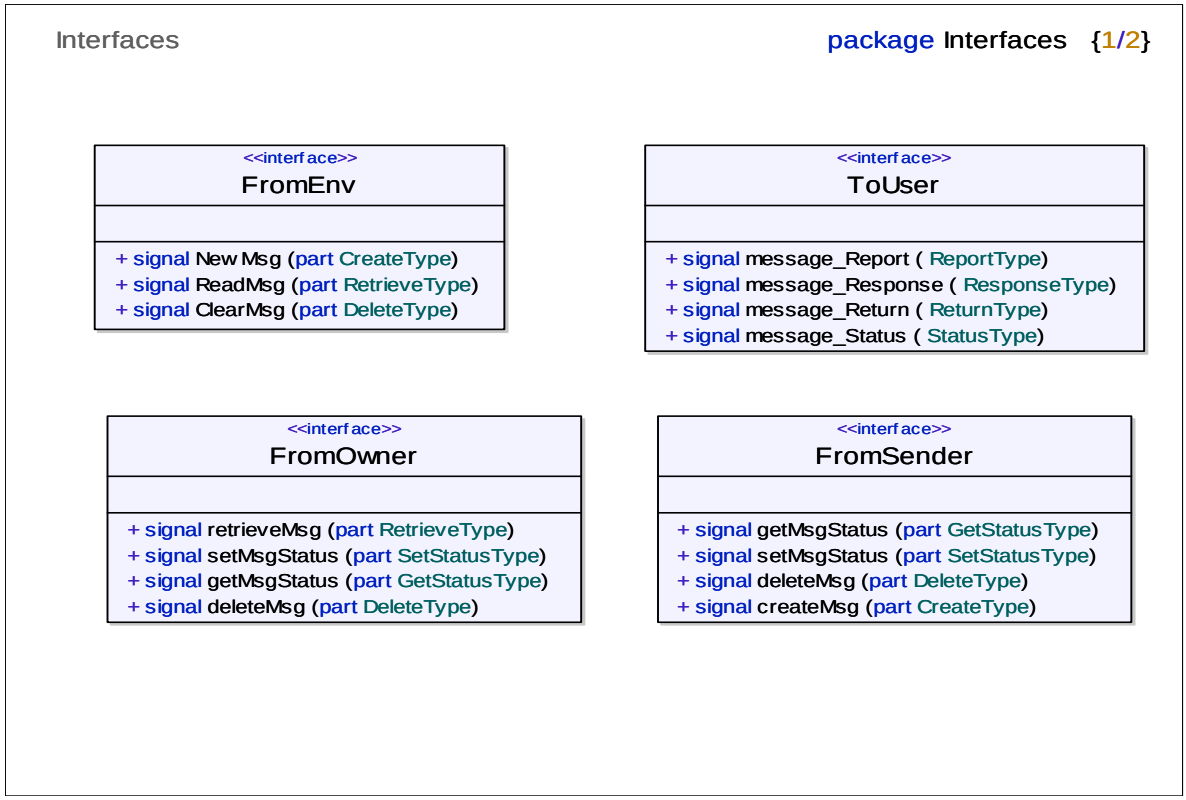


Figure 91: Message interfaces

Figure 90 identifies the following service capabilities which can operate on objects in the Message class:

- create a message;
- retrieve a message;
- delete a message;
- set the status of a message;
- get the status of a message.

The initialization and local variables of the Message group service capabilities are shown in figure 92.

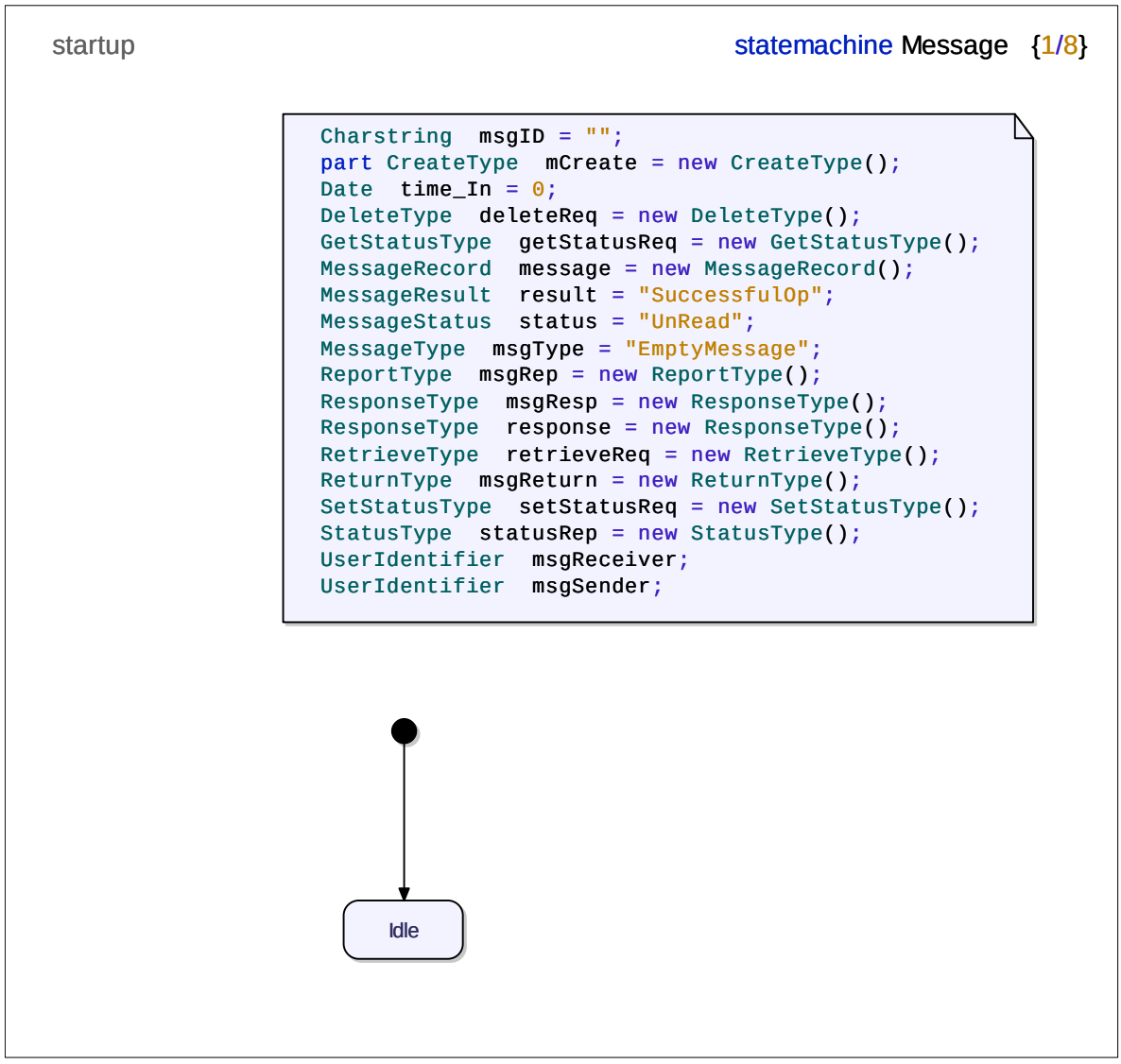


Figure 92: The Message group service capabilities common initialization

8.3.1 Create message

The *create message* service capability creates a new message on request from a suitably authorized user or application. A specification of the behaviour of this service capability is shown in figure 93.

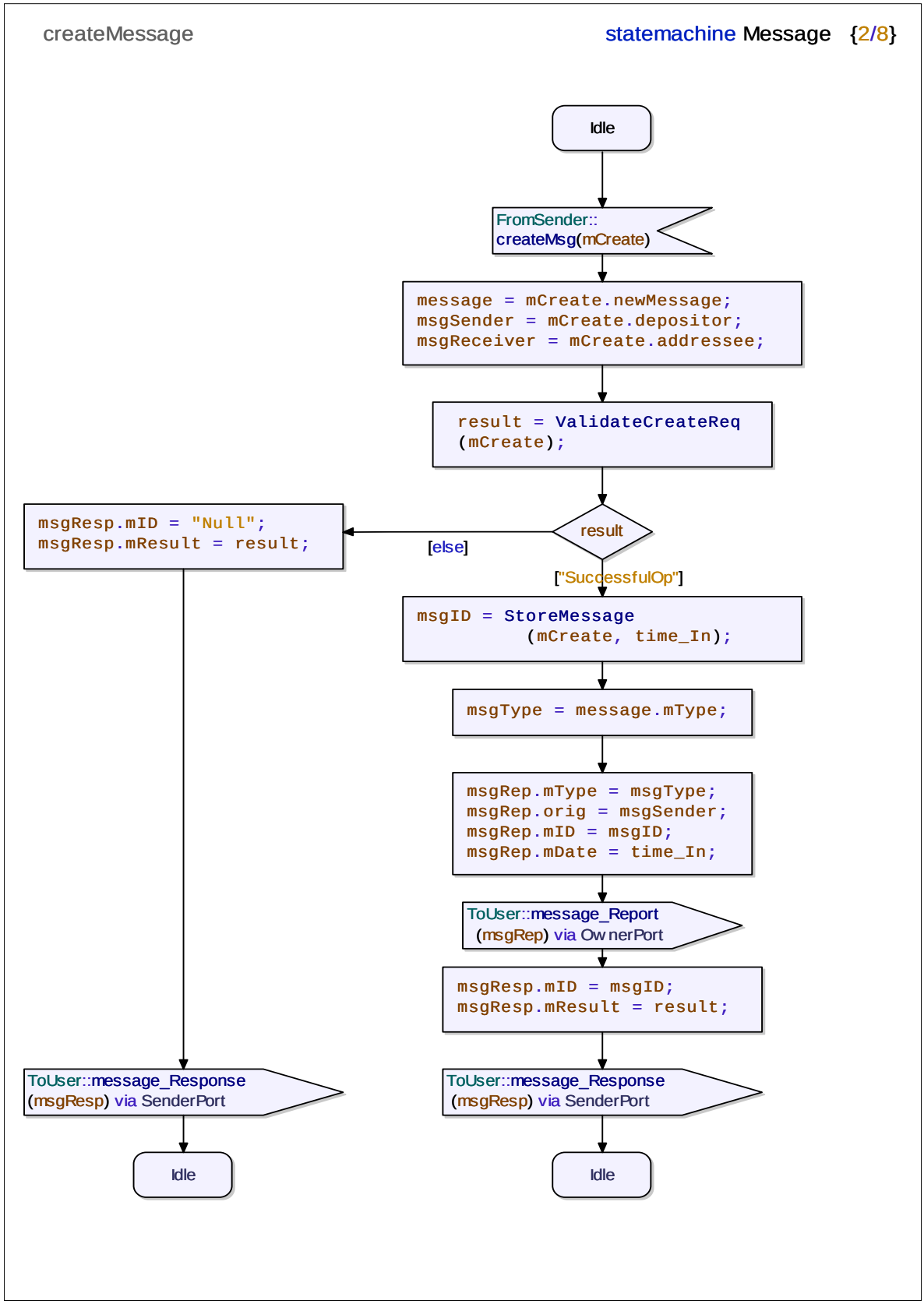


Figure 93: The create message service capability

8.3.2 Retrieve message

The *message retrieve* service capability delivers the contents of an existing message to a suitable authorized user or application (normally the message recipient). A specification of the behaviour of this service capability is shown in figure 94.

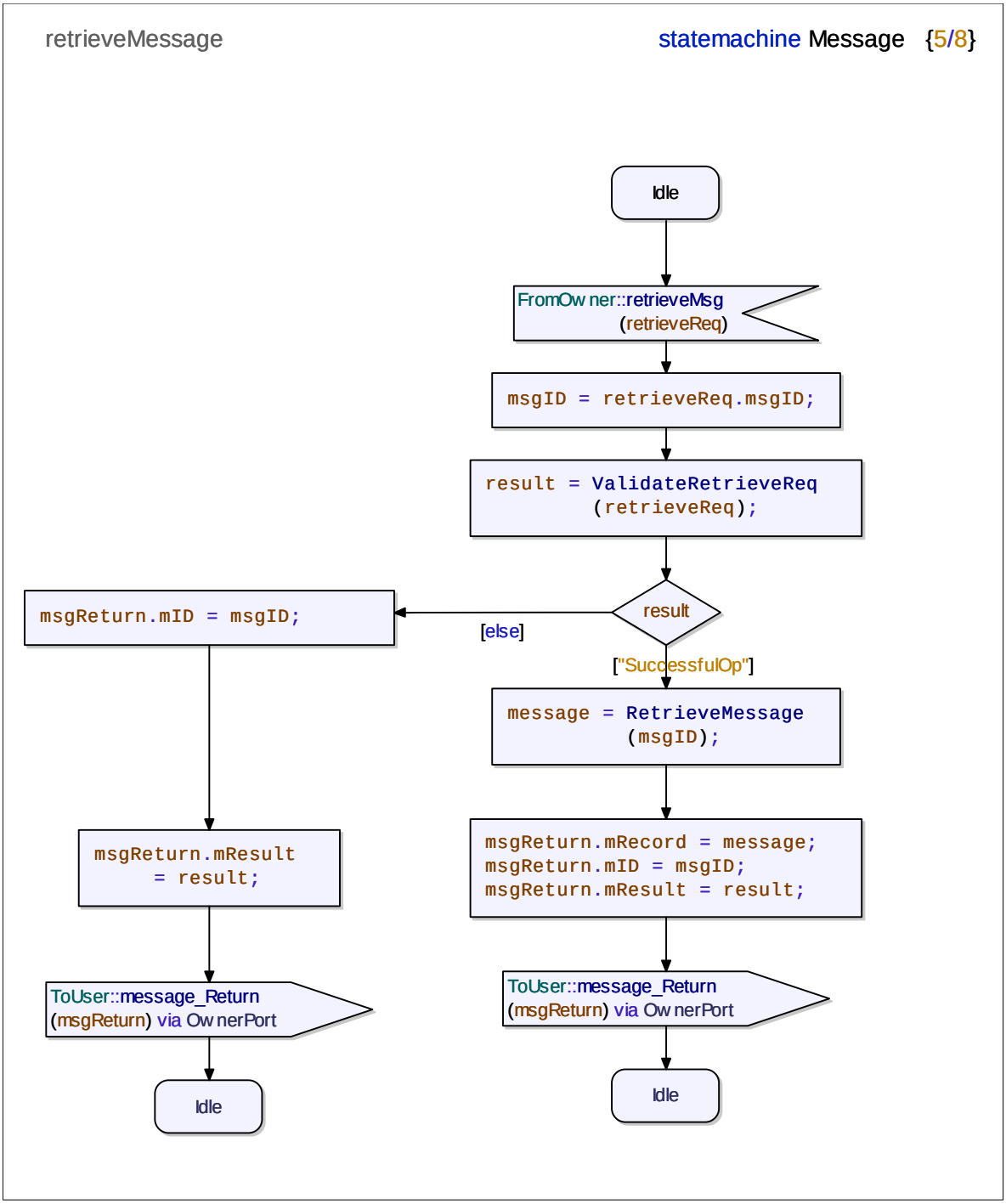


Figure 94: The retrieve message service capability

8.3.3 Set message status

The *set message status* service capability modifies the current status of an existing message. The only valid values of message status shall be "Read" and "Unread". A specification of the behaviour of this service capability is shown in figures 95 and 96. Two diagrams are necessary here as the message status can be set by both the sender and the owner so processing common to both routes has been taken out into a separate operation, *ProcessSetStatusRequest*.

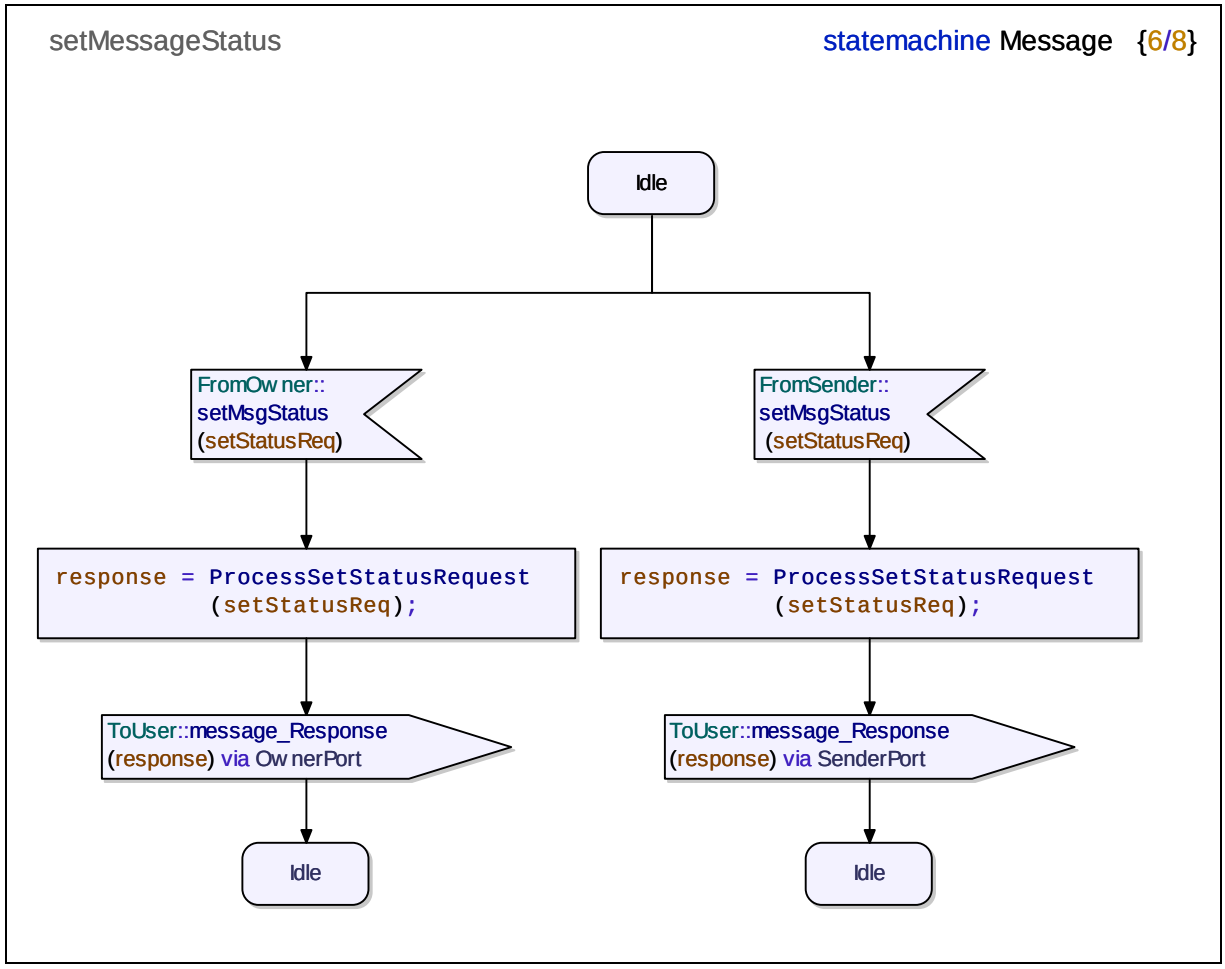


Figure 95: The set message status service capability

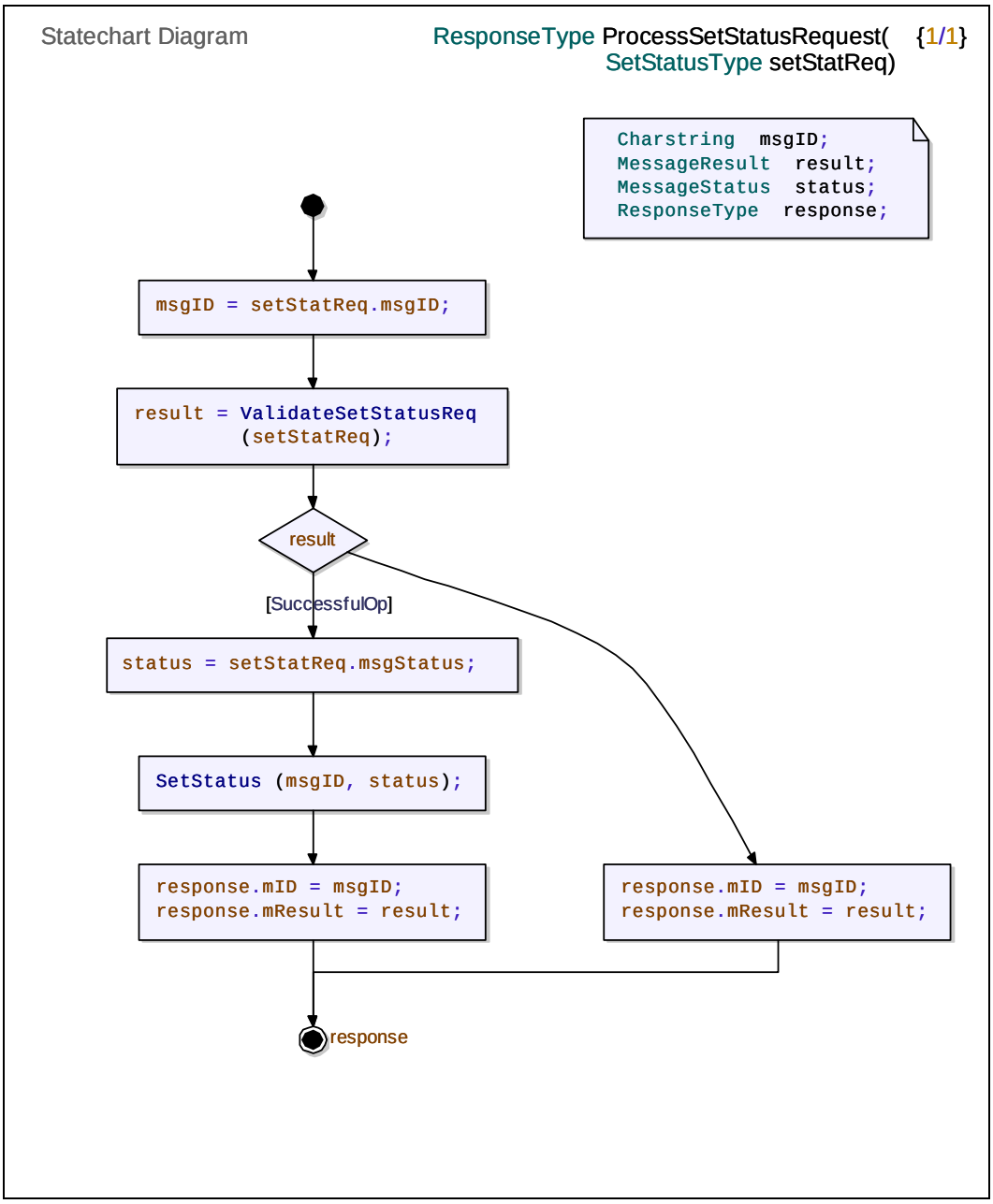


Figure 96: The process set status request operation

8.3.4 Get message status

The *get message status* service capability returns the current status of an existing message to a suitably authorized user or application. A specification of the behaviour of this service capability is shown in figure 98.

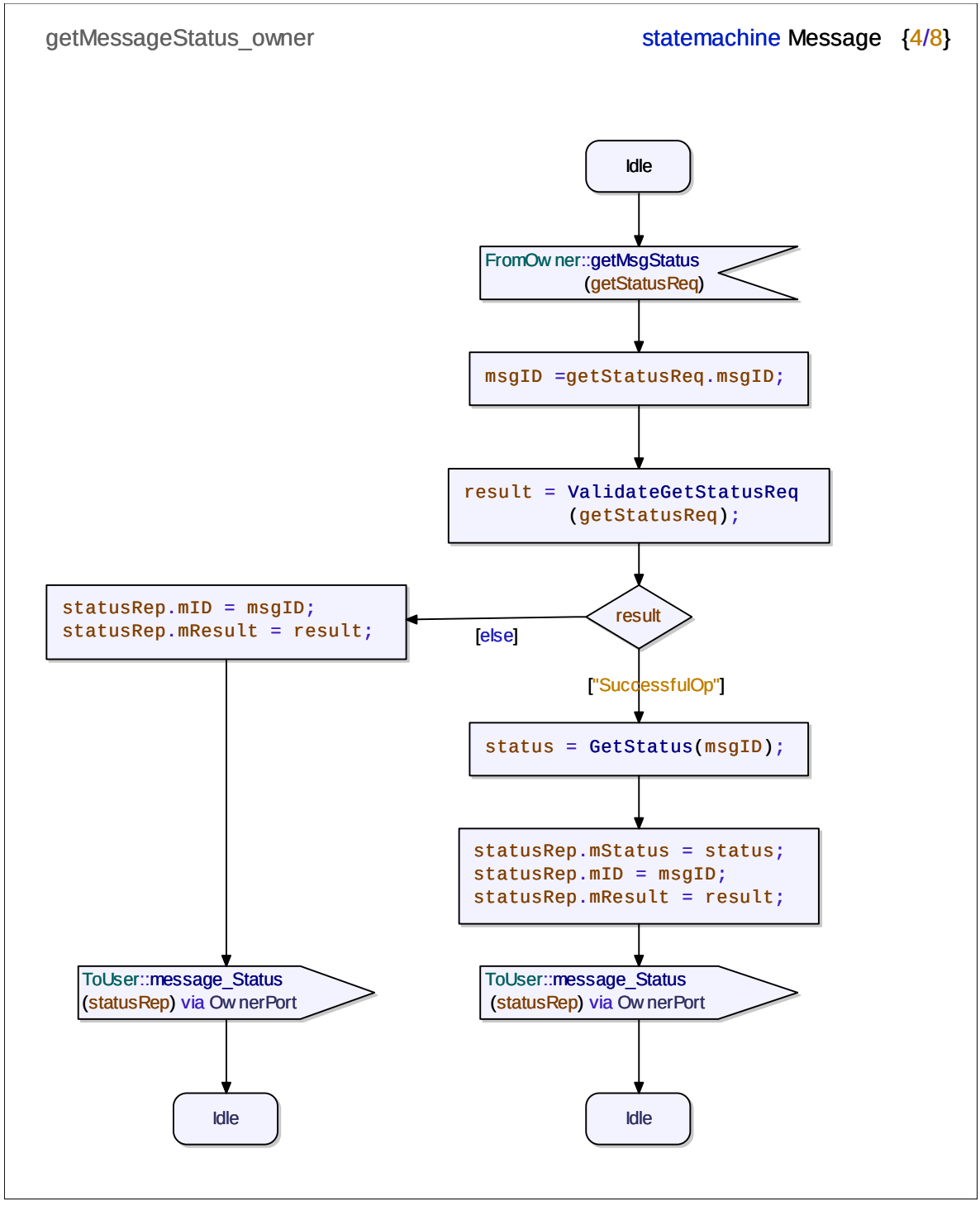


Figure 97: The get message status service capability (owner part)

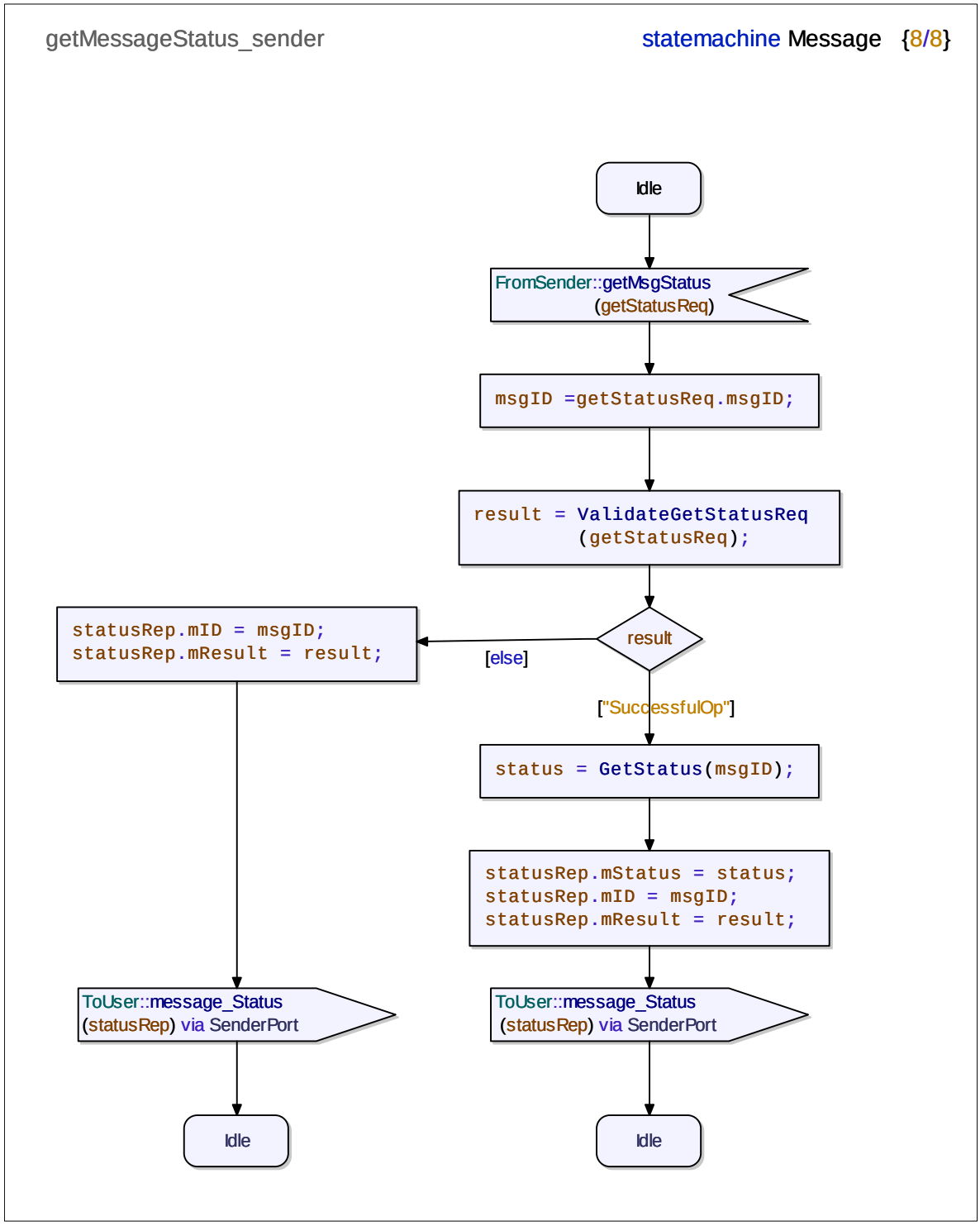


Figure 98: The get message status service capability (sender part)

8.3.5 Delete message

The *delete message* service capability removes an existing message on request from a suitably authorized user or application. A specification of the behaviour of this service capability is shown in figure 99.

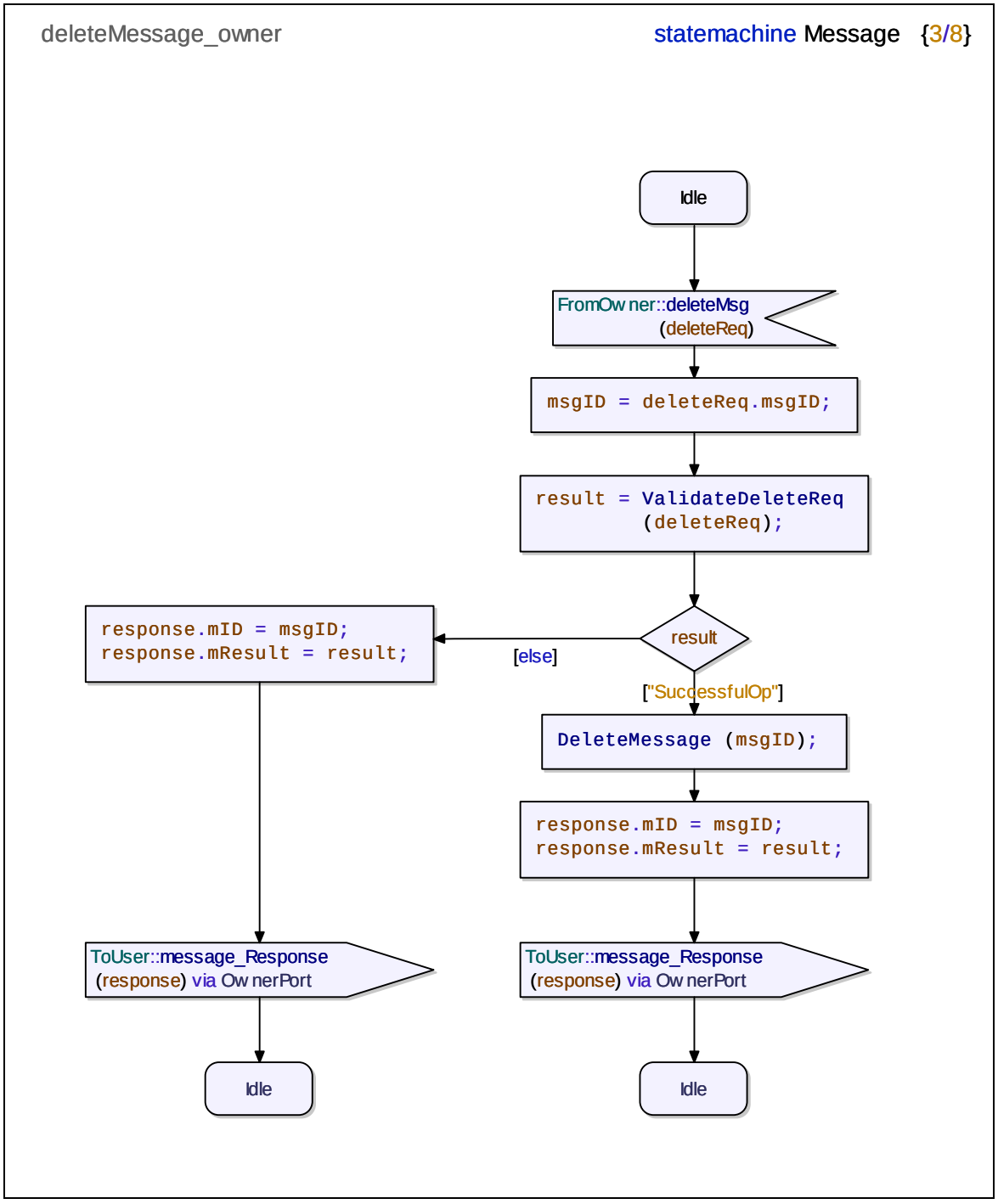


Figure 99: The delete message service capability (owner part)

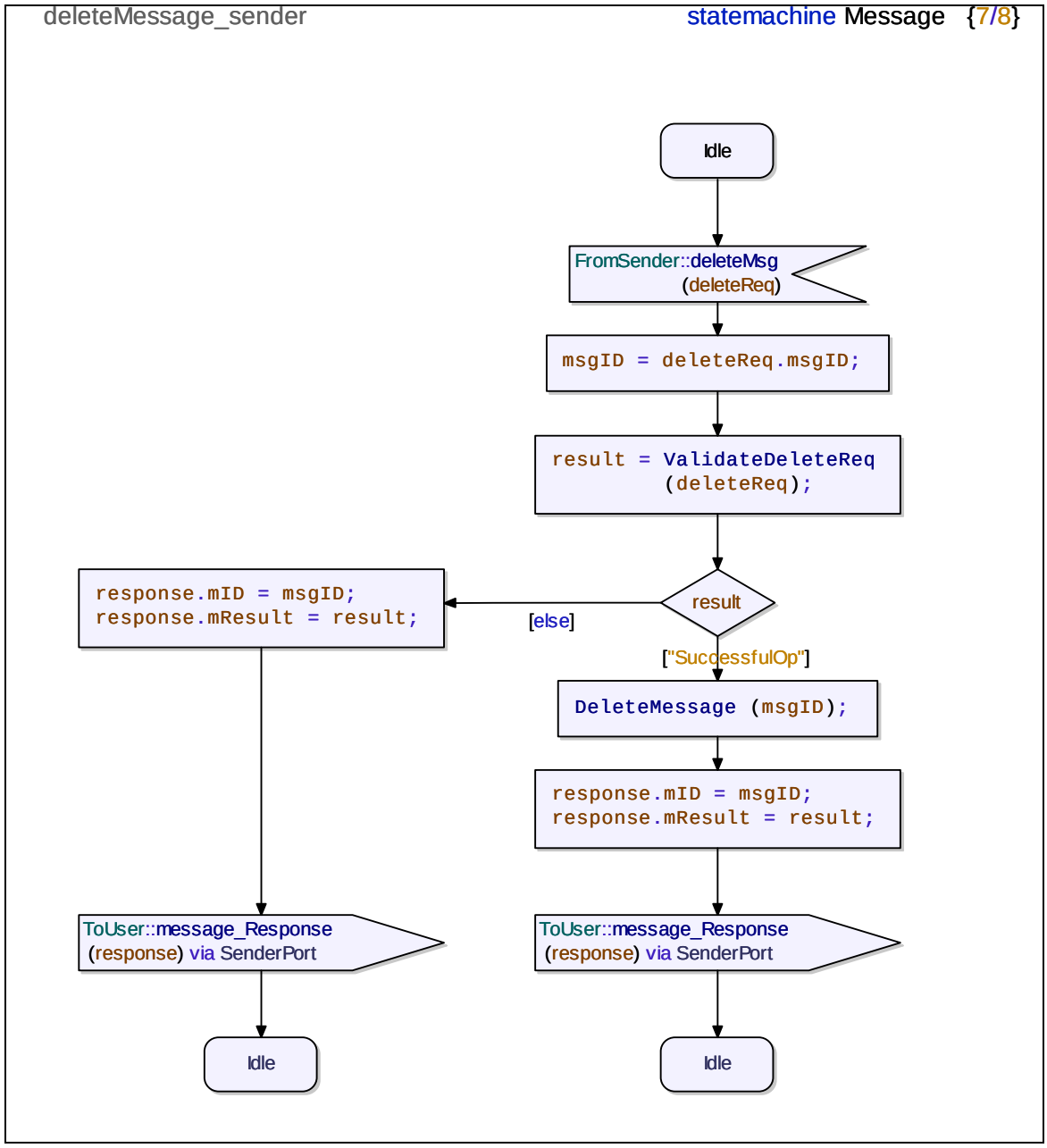


Figure 100: The delete message service capability (sender part)

8.4 Typical architecture

Figure 101 shows a typical architectural arrangement for an object based on the Message class.

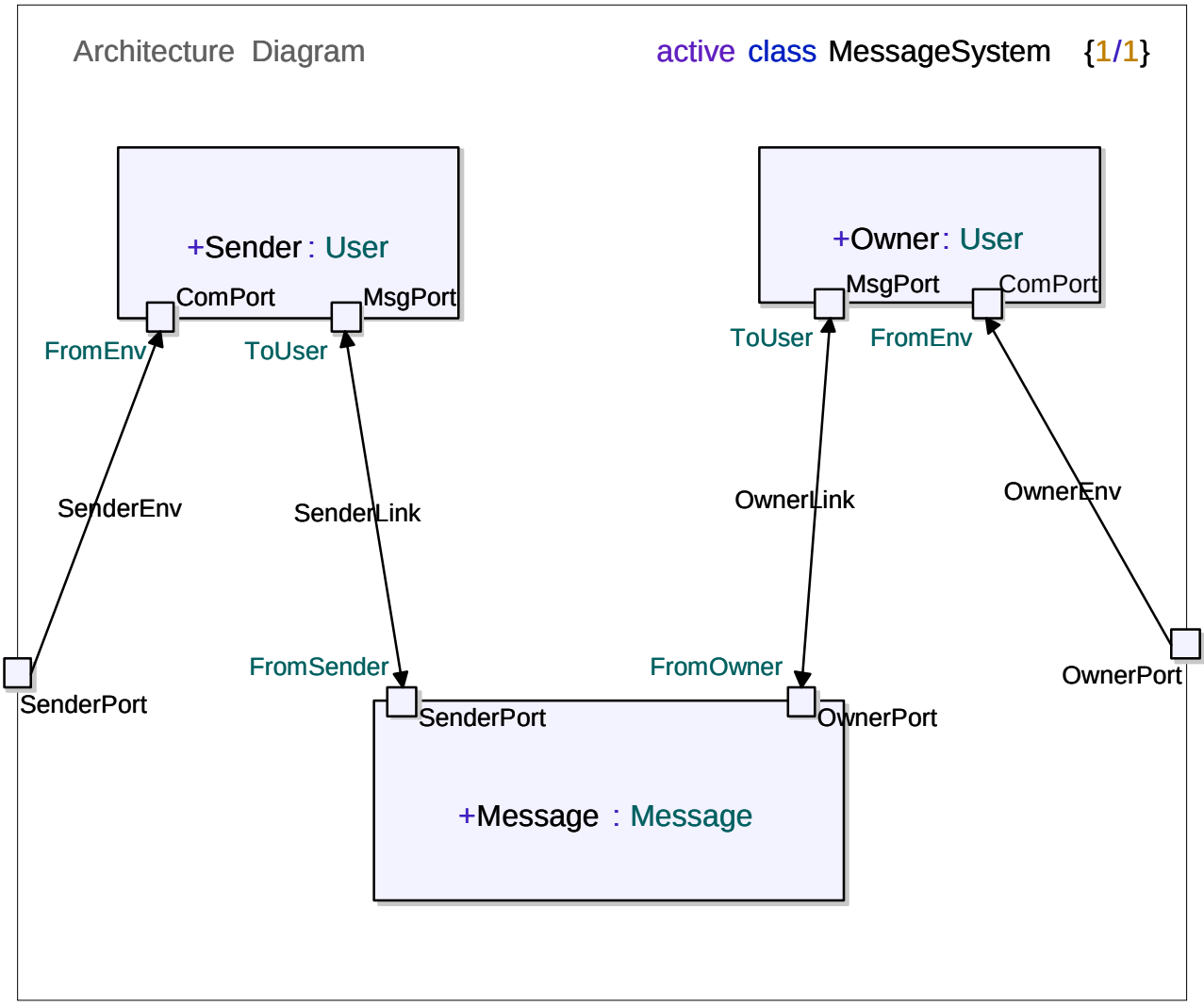


Figure 101: Typical message service architecture

NOTE: The User class has been added to this model purely for the purpose of completeness and to enable a clear distinction between the sender and the owner of a message. It has only simple behaviour defined for it and should not be considered to be normative.

9 Event handler group

9.1 Introduction

The event handler group contains those service capabilities required to track status and events occurring in different groups and to bind actions based upon these events.

9.2 Event handler group

The service capabilities and attributes of the event handler group are shown in figure 102.

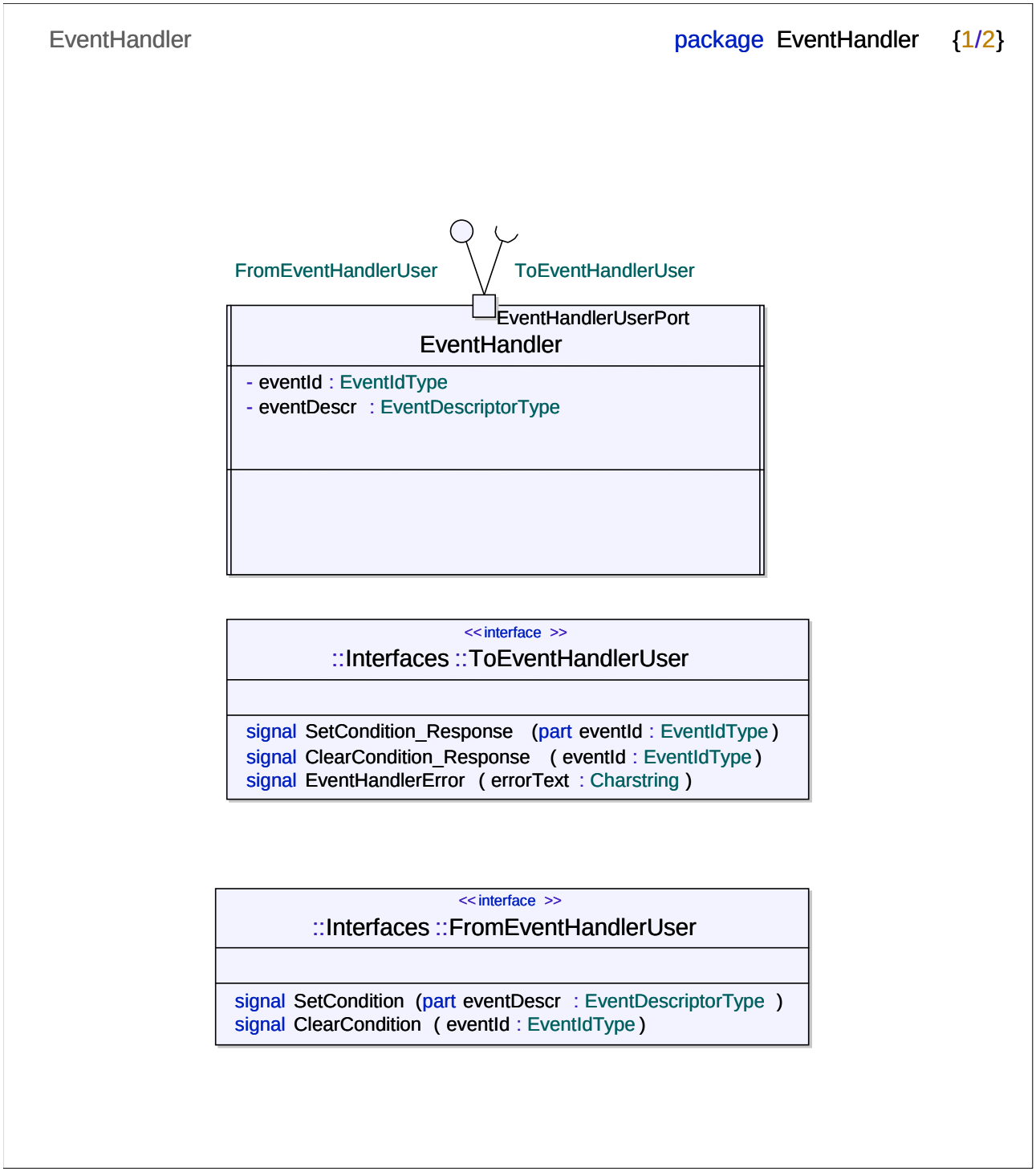


Figure 102: Event handler class with service capabilities as signals

9.3 Data definitions

The data definition model of the Event handler group is shown in figure 103.

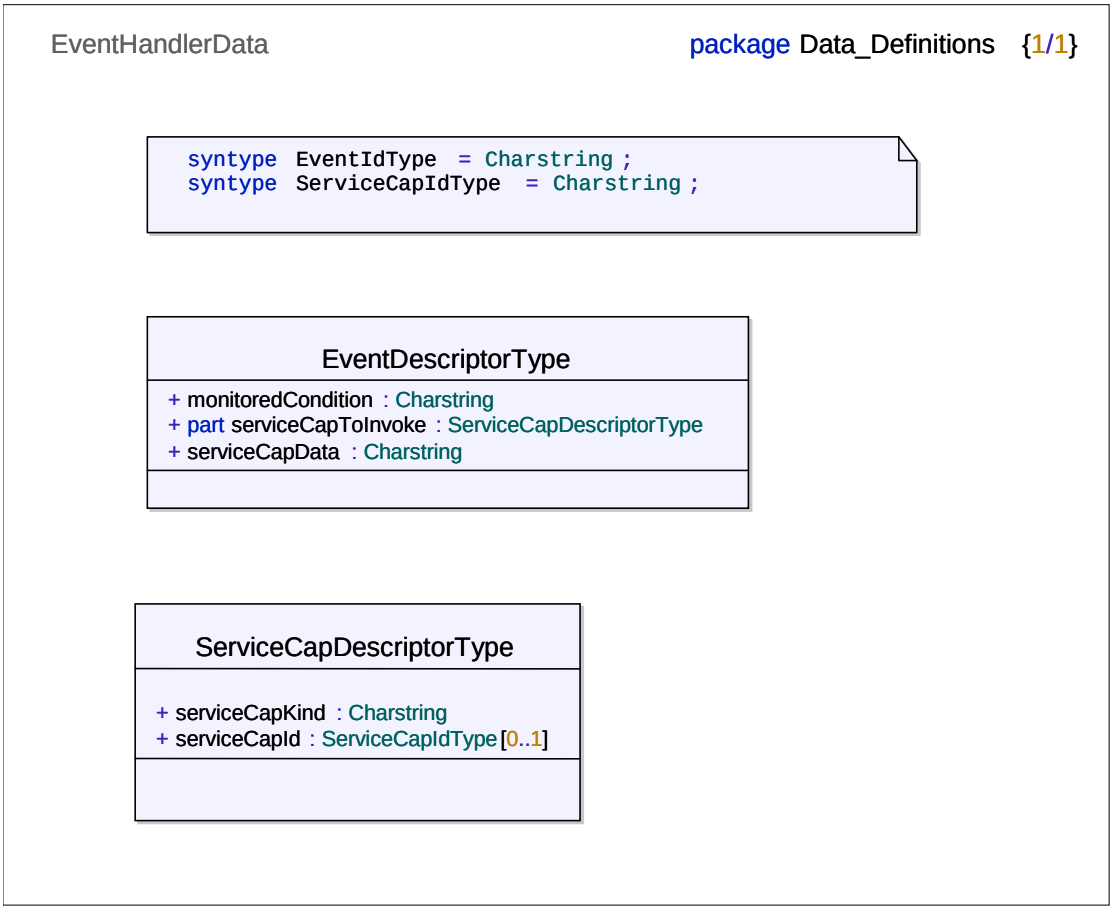


Figure 103: Data definitions for the event handler group service capabilities

9.4 Service capability models

The following service capabilities belongs to the Event handler class:

- set a condition;
- clear a condition.

Figure 104 shows the initialization of instances of the Event handler class.

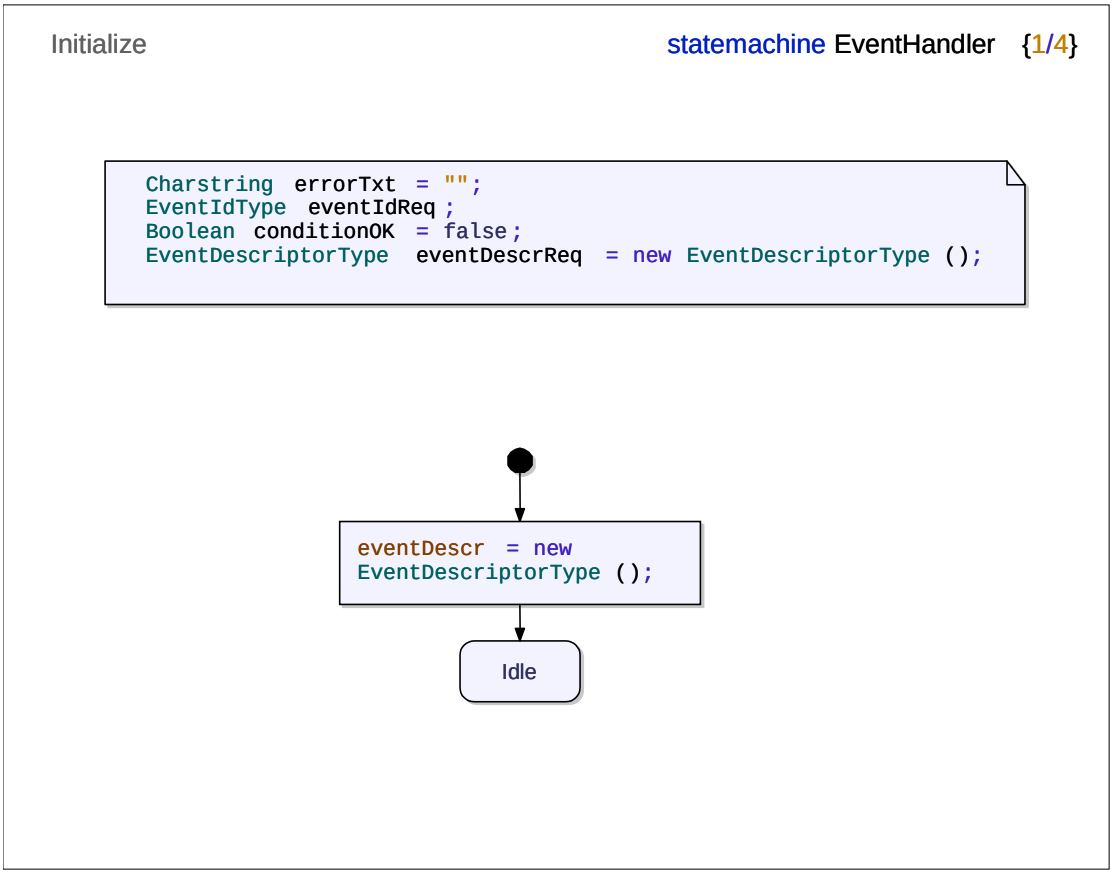


Figure 104: Initialization of event handler objects

9.4.1 Set condition

The *set condition* service capability sets a trigger based upon a condition related to the monitored group. The supplied event descriptor specifies the service capability to be invoked and the parameters to use when the condition is met. In figure 105 the behaviour of the set condition service capability is defined.

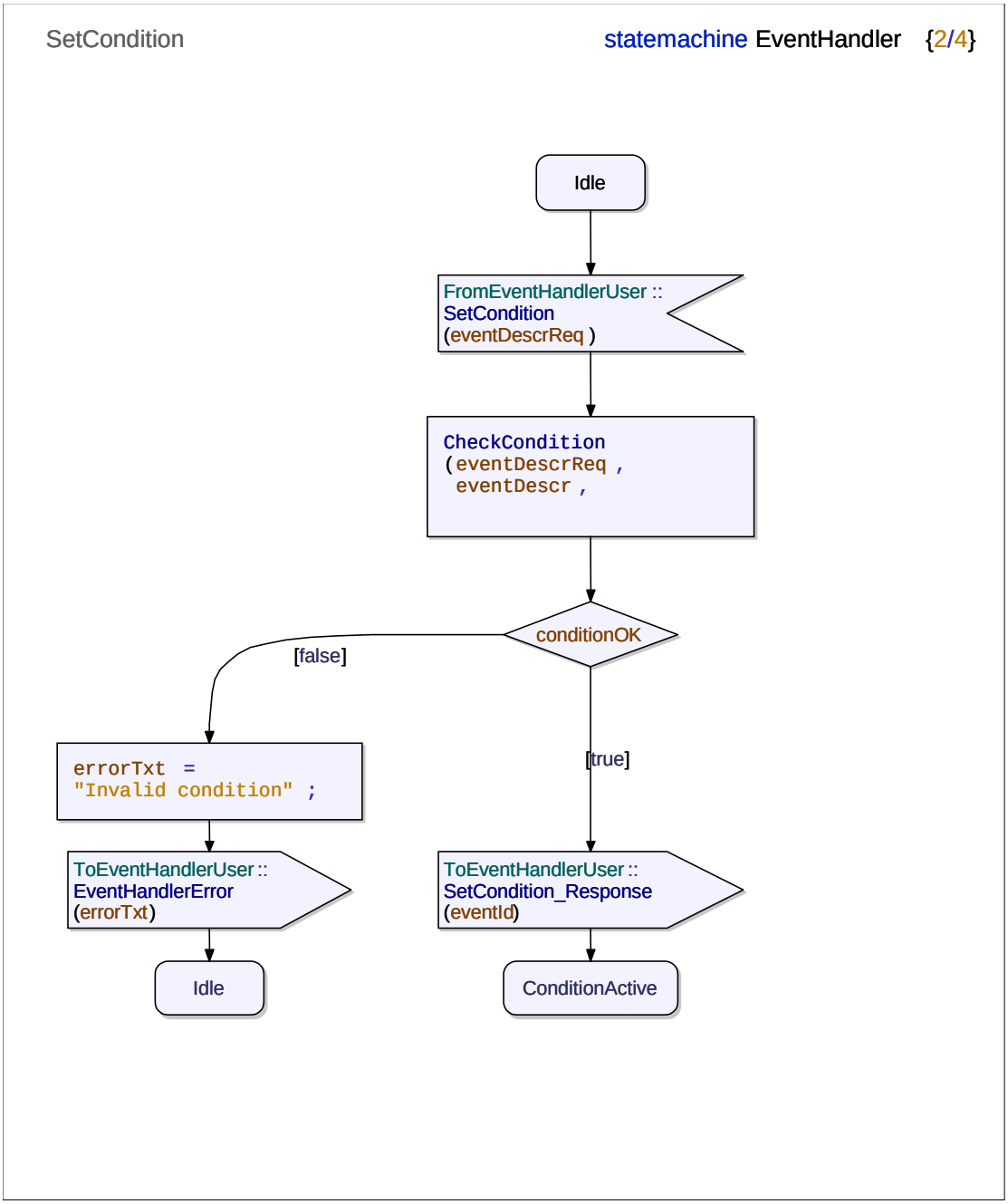


Figure 105: The set condition service capability

9.4.2 Clear condition

The *clear condition* service capability clears a previously set condition, identified by the supplied event identity. The service capability behaviour is shown in figure 106.

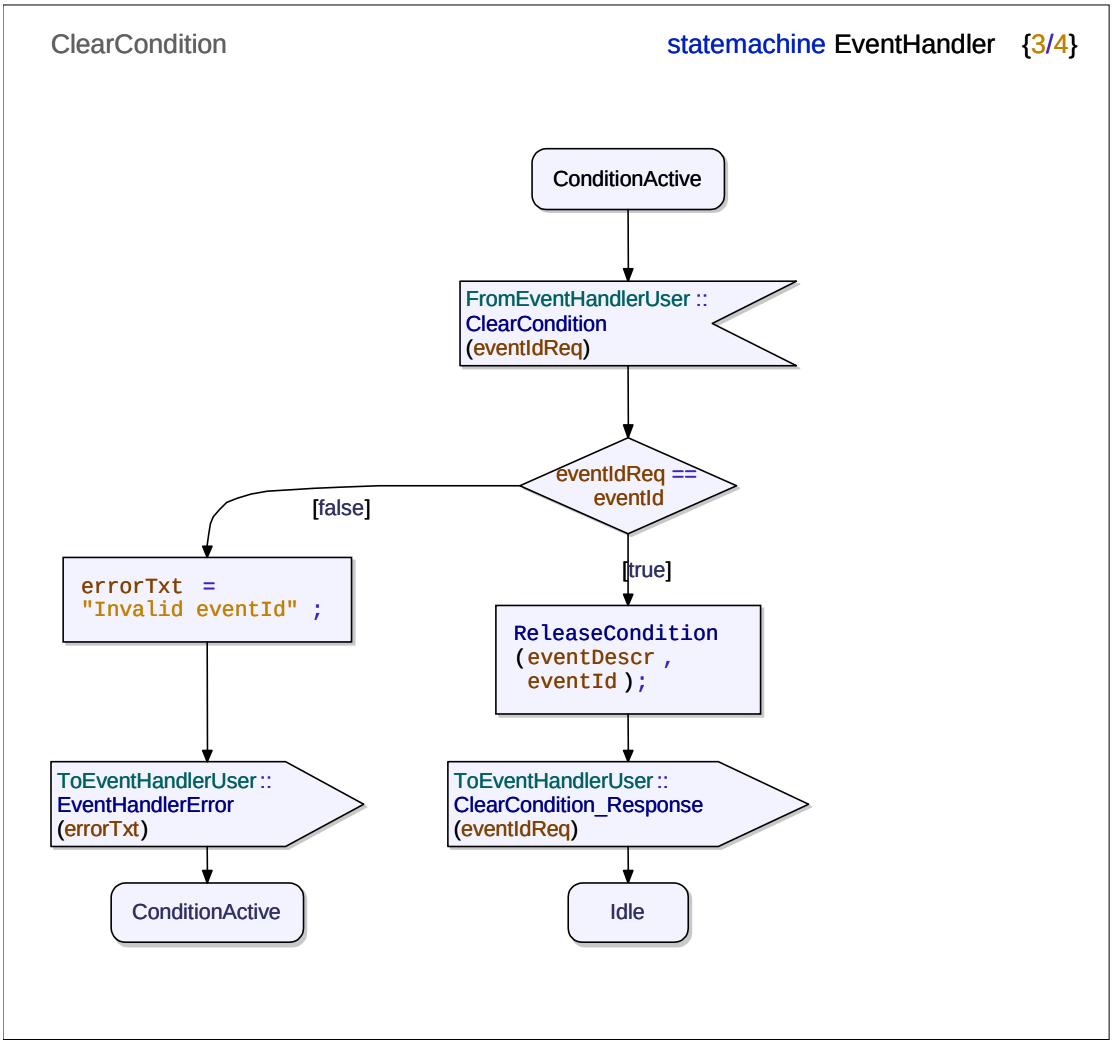


Figure 106: The clear condition service capability

9.4.3 Operation signatures

The operation signatures used in the state machine of Event Handler class are shown in figure 107.

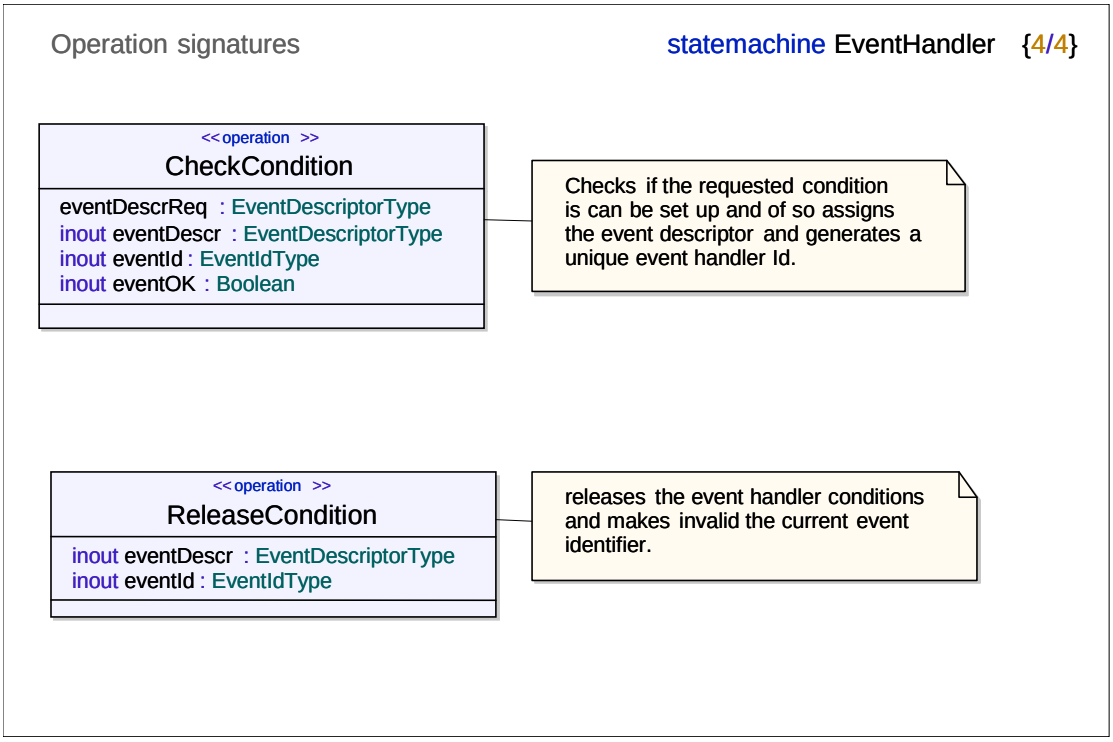


Figure 107: The operation signatures of the event handler state machine

9.5 Typical architecture

Figure 108 shows a typical architectural arrangement for an object based on the Event class.

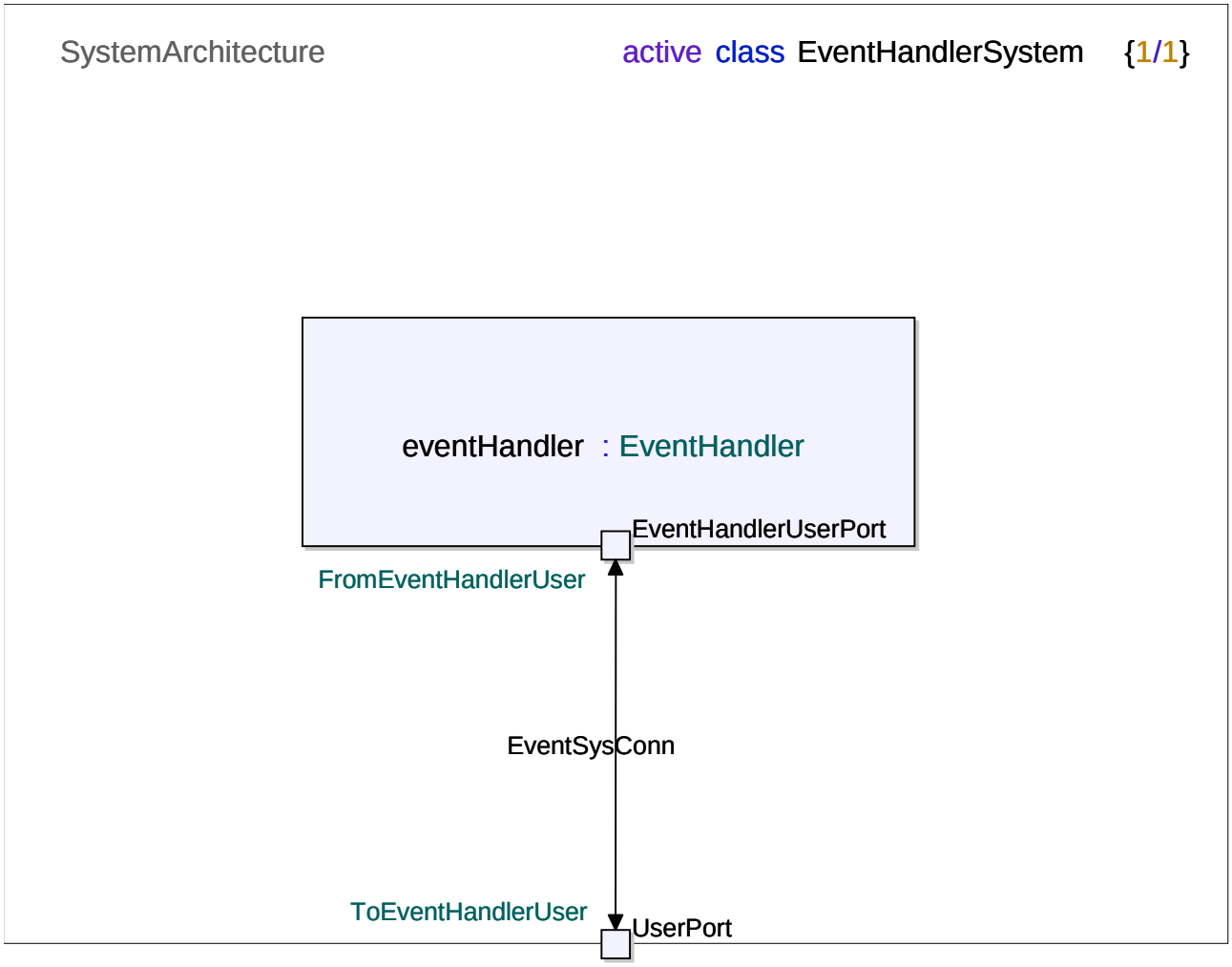


Figure 108: Typical event service architecture

Annex A:
Derived services and capabilities

A.1 Derived capabilities from Profile group

A.1.1 Register

The *register* service capability may be built by invocation of the *Set User Status*, *Update Location* and *Update Service Status* capabilities defined in clauses 4.3.8, 4.3.10, and 4.3.11. A pseudo-code illustration of the *register* service capability is given below:

```
Register()
{
    setUserStatus(registering-user, Online)
    updateLocation(registering-user, CurrentLocation)
    for 1 to NumberOfServicesToBeOpened
    {
        updateServiceStatus(registering-user, service-name, Online)
        getServiceDescriptor(registering-user, service-name)
    }
}
```

NOTE: The default number of services to be opened in a *register* service capability invocation is one.

A.1.2 Deregister

The *deregister* service capability may be built by invocation of the setUserStatus, updateLocation and updateServiceStatus capabilities defined in clauses 4.3.8, 4.3.10, and 4.3.11. A pseudo-code illustration of the *register* service capability is given below:

```
Deregister()
{
    setUserStatus(registering-user, Offline)
    for 1 to NumberOfServicesOpened
    {
        updateServiceStatus(registering-user, service-name, Offline)
    }
}
```

A.1.3 Attach

The *attach* service capability may be built by invocation of the *Update Service Status* capability defined in clause 4.3.11. A pseudo-code illustration of the *attach* service capability is given below:

```
Attach()
{
    updateServiceStatus(registering-user, service-name, Offline)
}
```

A.1.4 Detach

The *detach* service capability may be built by invocation of the *Update Service Status*, capability defined in clause 4.3.11. A pseudo-code illustration of the *detach* service capability is given below:

```
Detach()
{
    updateServiceStatus(registering-user, service-name, Offline)
}
```

A.2 Derived capabilities from call group

A.2.1 Call join

The *Call join* service may be specified using the *Add call descriptor* and *Retrieve call descriptor* service capabilities. A pseudo-code example of how to build the *Call join* service is shown below:

```
Calljoin( callA, callB )
{
    callA.retrieveCallDescriptor(callDescr);
    callB.addCallDescriptor(callDescr);
}
```

Annex B:
UML model source files

The UML models of the service capabilities were developed using the Telelogic TAU G2 tool, ver. 2.2. Electronic version of the UML models are contained in archive TR_101882v050101 which accompanies the present document:

Table 1: UML model archive files

UML model group	Files
Profile group	ProfileGroup.zip
Call group	CallGroup
Bearer group	BearerGroup.zip
Media group	MediaGroup.zip
Message group	MessageGroup.zip
Event handler group	EventHandlerGroup.zip

History

Document history		
V1.1.1	May 2002	Publication as TS 101 882
V4.1.1	September 2003	Publication as TS 101 882-1
V5.1.1	May 2004	Publication