INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.770

(01/2001)

SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

OSI management – Management functions and ODMA functions

ODMA notification dispatch function

ITU-T Recommendation X.770

(Formerly CCITT Recommendation)

ITU-T X-SERIES RECOMMENDATIONS

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

PUBLIC DATA NETWORKS	
Services and facilities	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalling and switching	X.50–X.89
Network aspects	X.90–X.149
Maintenance	X.150–X.179
Administrative arrangements	X.180–X.199
OPEN SYSTEMS INTERCONNECTION	
Model and notation	X.200–X.209
Service definitions	X.210–X.219
Connection-mode protocol specifications	X.220–X.229
Connectionless-mode protocol specifications	X.230–X.239
PICS proformas	X.240–X.259
Protocol Identification	X.260–X.269
Security Protocols	X.270–X.279
Layer Managed Objects	X.280–X.289
Conformance testing	X.290–X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300–X.349
Satellite data transmission systems	X.350–X.369
IP-based networks	X.370–X.399
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600–X.629
Efficiency	X.630–X.639
Quality of service	X.640–X.649
Naming, Addressing and Registration	X.650–X.679
Abstract Syntax Notation One (ASN.1)	X.680–X.699
OSI MANAGEMENT	
Systems Management framework and architecture	X.700–X.709
Management Communication Service and Protocol	X.710–X.719
Structure of Management Information	X.720–X.729
Management functions and ODMA functions	**X.730–X.799**
SECURITY	X.800–X.849
OSI APPLICATIONS	
Commitment, Concurrency and Recovery	X.850–X.859
Transaction processing	X.860–X.879
Remote operations	X.880–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation X.770

ODMA notification dispatch function

Summary

This Recommendation specifies a set of UML descriptions of the OMG notification service, thus placing it within the ODMA architecture (specified in ITU-T X.703) as the ODMA Notification Dispatch Function.

Source

ITU-T Recommendation X.770 was prepared by ITU-T Study Group 4 (2001-2004) and approved under the WTSA Resolution 1 procedure on 19 January 2001.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

ITU-T Recommendation X.770

ODMA notification dispatch function

1 Scope

This ODMA Notification Dispatch Function is part of the Open Distributed Management Architecture, as specified in ITU-T X.703. This Recommendation references the specification of the OMG notification service for the normative specification of the ODP IDL (ITU-T X.950) engineering viewpoint interface definitions.

This Recommendation provides a UML based set of enterprise, information, and computational viewpoint language description of the OMG notification service, thus placing it within ODMA as the Notification Dispatch Function.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

– ITU-T X.703 (1997) | ISO/IEC 13244:1998, *Information technology – Open Distributed Management Architecture*.

– OMG Document formal/2000-06-20, *Notification service specification*.

– OMG Document formal/1999-06-01, *OMG Modeling Book*.

3 Definitions

3.1 Definitions from notification service

The following terms are used as defined in OMG notification service:

– filter;

– mapping filter;

– structured notifications (content having a fixed header and a flexible body with names for each value element);

– typed notifications (content conforming to a statically defined syntax);

– untyped notifications (content determined at time of delivery).

3.2 Definitions from ITU-T X.703

The following terms are used as defined in ITU-T X.703:

– enterprise viewpoint;

– information viewpoint;

– computational viewpoint;

– engineering viewpoint;

– notification.

3.3 Definitions from UML

The following terms are used as defined in UML:

- association;
- association class;
- class diagram;
- include;
- use case diagram.

4 Abbreviations and conventions

The following abbreviations are used in this Recommendation:

CEC	Cos Event Comm
CN	Cos Notification
CNC	Cos Notification Comm
CNCA	Cos Notify Channel Admin
CNF	Cos Notify Filter
CTEC	Cos Typed Event Comm

The diagrams in this Recommendation use the Notation specified in the OMG UML version 1.3.

5 Enterprise viewpoint UML use-cases for OMG notification service

The actors are shown only at the first level of the use case diagram. Where the detailed decompositions are described for the use cases, the actors are not explicitly shown for readability.

5.1 Use-case overview

See Figure 5-1.

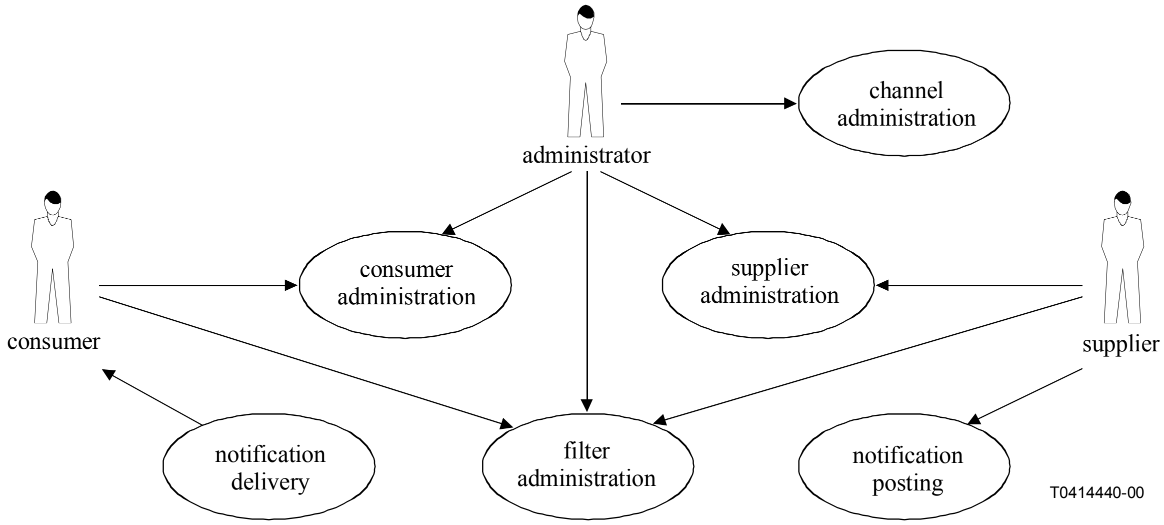


Figure 5-1/X.770 – High-level use case diagram for notification service

In this figure, all the use cases correspond to the various functions that are part of the notification service. These use cases include both administrative activities such as consumer administration, and the activities for dispatching the notifications.

An administrator is responsible for administering the overall properties of notification channels, which are used to receive notifications from suppliers and deliver the received notifications to consumers that have subscribed to the notifications.

Consumers and administrators are involved in the consumer specific administration for notification channels.

Suppliers and administrators are involved in the supplier specific administration for notification channels.

Administrators, suppliers and consumers are involved in the administration of filters, which control the receipt and delivery of notifications.

The supplier is involved in the posting of notifications. The supplier may not know the ultimate consumers of the notifications it supplies.

The consumer receives the delivered notifications. The consumer may not know the originating supplier of the notifications it consumes.

5.2 Notification delivery use case diagram

See Figure 5-2.

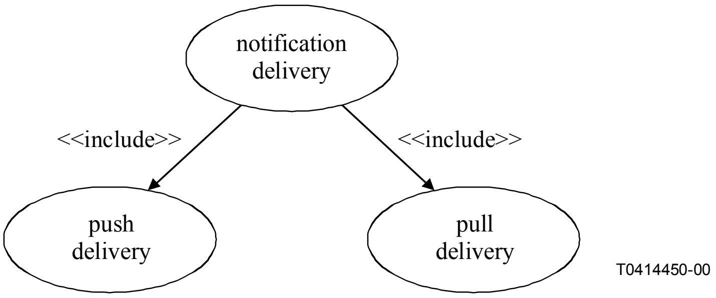


Figure 5-2/X.770 – Push and pull as specialization of notification delivery

Notifications may be delivered using either a push model (notification service invokes an operation on consumer in order to deliver a notification to the consumer) or a pull model (consumer invokes an operation on the notification service in order to have its notifications delivered to itself).

5.3 Notification posting use case diagram

See Figure 5-3.

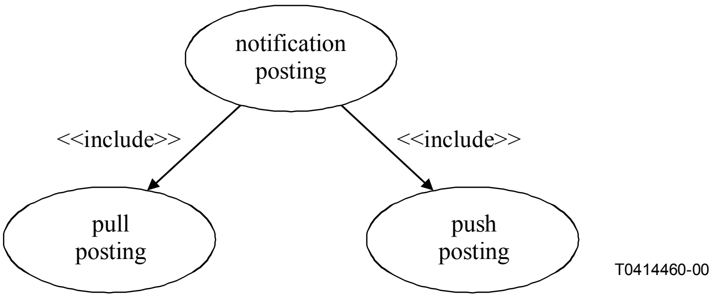


Figure 5-3/X.770 – Push and pull as specialization for notification posting

Notifications may be posted using either a push model (supplier invokes an operation on notification service in order to post a notification) or a pull model (notification service invokes an operation on the supplier to receive posted notifications from the supplier).

5.4 Filter administration use case diagram

See Figure 5-4.

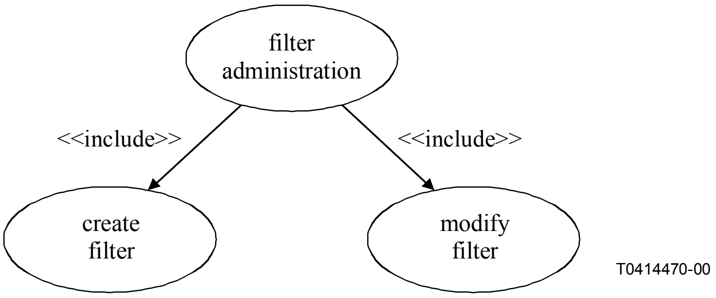


Figure 5-4/X.770 – Filter administration includes creating and modifying filters

There is a need for mechanisms to both create and modify the logical constructs of filters, which control the delivery of notification instances based on the actual values of the parameters of the notification, (e.g. do not deliver notifications which have a error count of zero).

5.5 Consumer administration use case diagram

Figure 5-5 shows how to use the proxy administration corresponding to the consumer. The administration of a consumer proxy includes the following functions: create a proxy, define the filter (associating filter) to determine what notifications are to be received, and attach or detach to/from the proxy. The notify subscribe use case supports the function for the consumer to subscribe to notifications resulting from changes to the notification types published by the supplier.

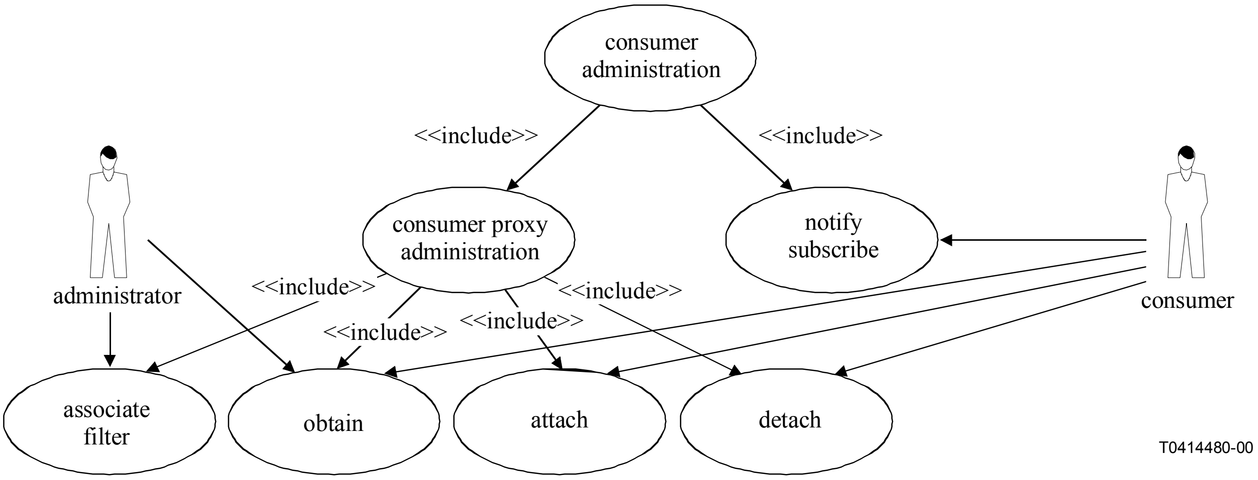


Figure 5-5/X.770 – Consumer administration

5.6 Supplier administration use case

Figure 5-6 shows how to use the proxy administration corresponding to the supplier. The administration of a supplier proxy includes the following functions: create a proxy, define the filter (associating filter) to determine what notifications are to be forwarded and attach or detach to/from the proxy. The notify publish use case supports the function for the supplier to publish changes (addition or deletion) to notification types it supplies.

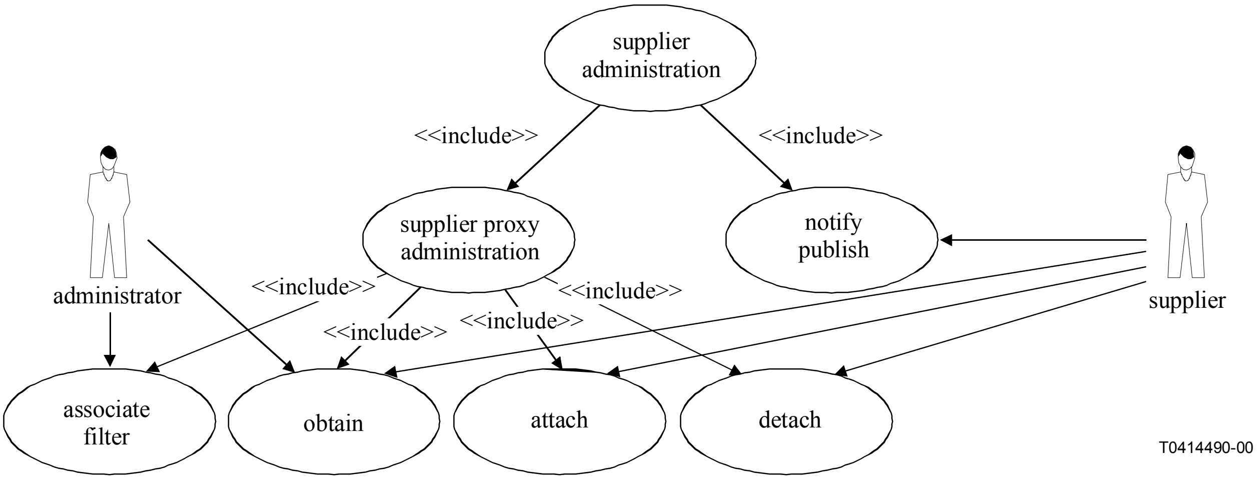


Figure 5-6/X.770 – Supplier administration

6 Information viewpoint model for OMG notification service

This clause shows information object classes and their relationships. Attributes for information object classes are not shown in this clause, for purposes of simplicity of exposition. Not all classes shown in this viewpoint correspond to IDL interfaces for the notification service. See Figure 6-1.

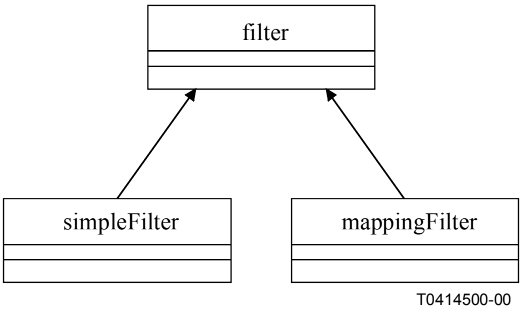


Figure 6-1/X.770 – Filter types

A filter has a logical construct, made up from assertions on the values of the various parameters of a notification instance along with logical operators, (i.e. and, or, not). The filter evaluates to either true or false.

A simple filter is used to determine delivery of a notification either into or out of a channel. If the filter construct evaluates to true the notification instance is delivered.

A mapping filter is used to associate a particular quality of service value with a notification instance. If the filter construct is true, the associated QOS value is assigned to that notification instance. See Figure 6-2.

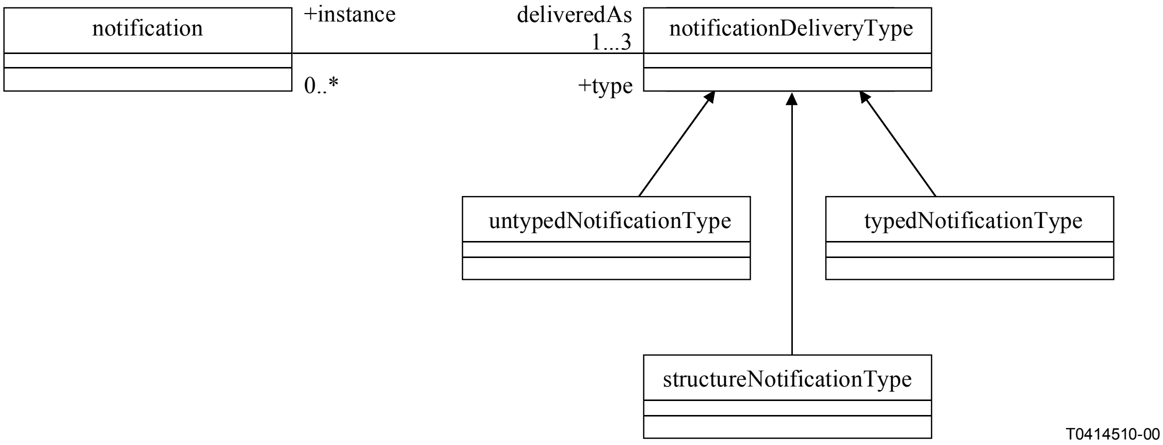


Figure 6-2/X.770 – Notification delivery types

Each notification instance is delivered using a notification delivery type, which may be either an untyped notification type (encoded as a single value of type "any"), a structure notification type (encoded as a structure containing a fixed header followed by elements which are name value pairs), or a typed notification type (encoded as a strongly typed operation). Each notification instance (i.e. the specification of the semantic contents of the notification) may be delivered using one or more of these notification delivery types.

The notification service provider needs to provide translation services allowing consumers of any notification delivery type to receive notifications posted by providers using any of the three notification delivery types. See Figure 6-3.

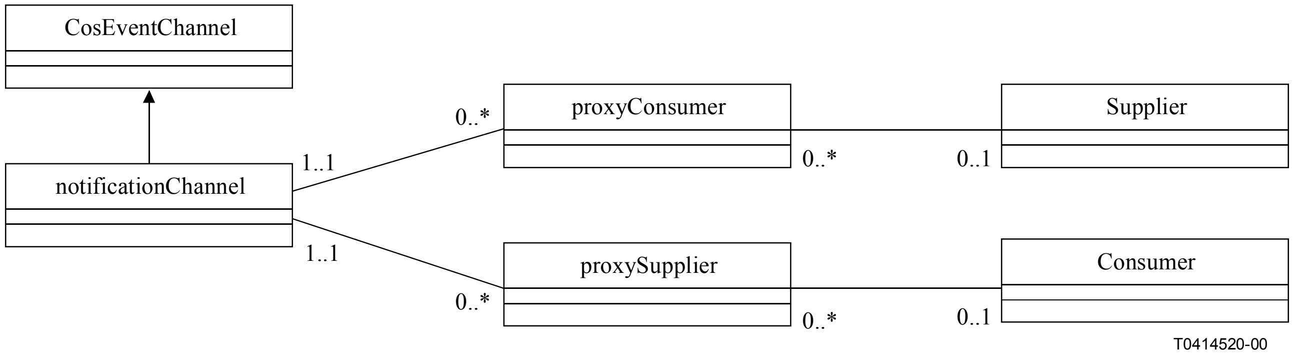


Figure 6-3/X.770 – Channel types

A notification channel is a specialization of a COS event channel and, as such, has zero or more proxy consumers and/or proxy suppliers. Each proxy consumer is associated with at most one supplier. Each proxy supplier is associated with at most one consumer. Each supplier may be associated with zero or more proxy consumers, while each consumer may be associated with zero or more proxy suppliers.

There are three subtypes of proxy consumers, those which consume untyped notifications, those which consume typed notifications, and those which consume structured notifications. See Figure 6-4.

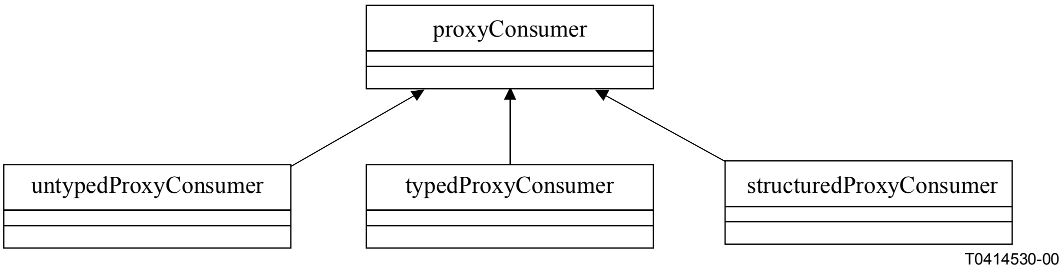


Figure 6-4/X.770 – Proxy consumers

There are three subtypes of proxy supplier, those which supply untyped notifications, those which supply typed notifications, and those which supply structured notification types. See Figure 6-5.

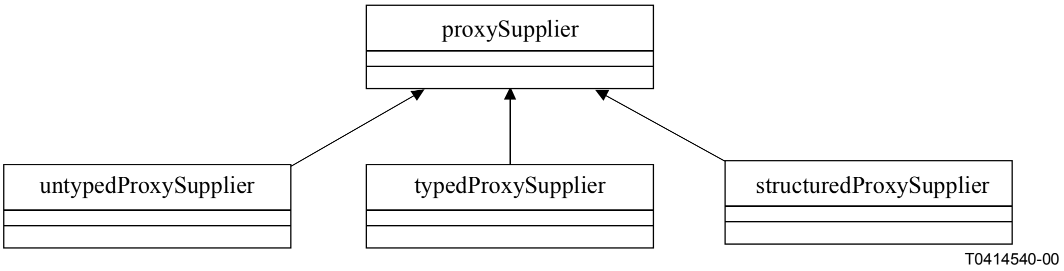
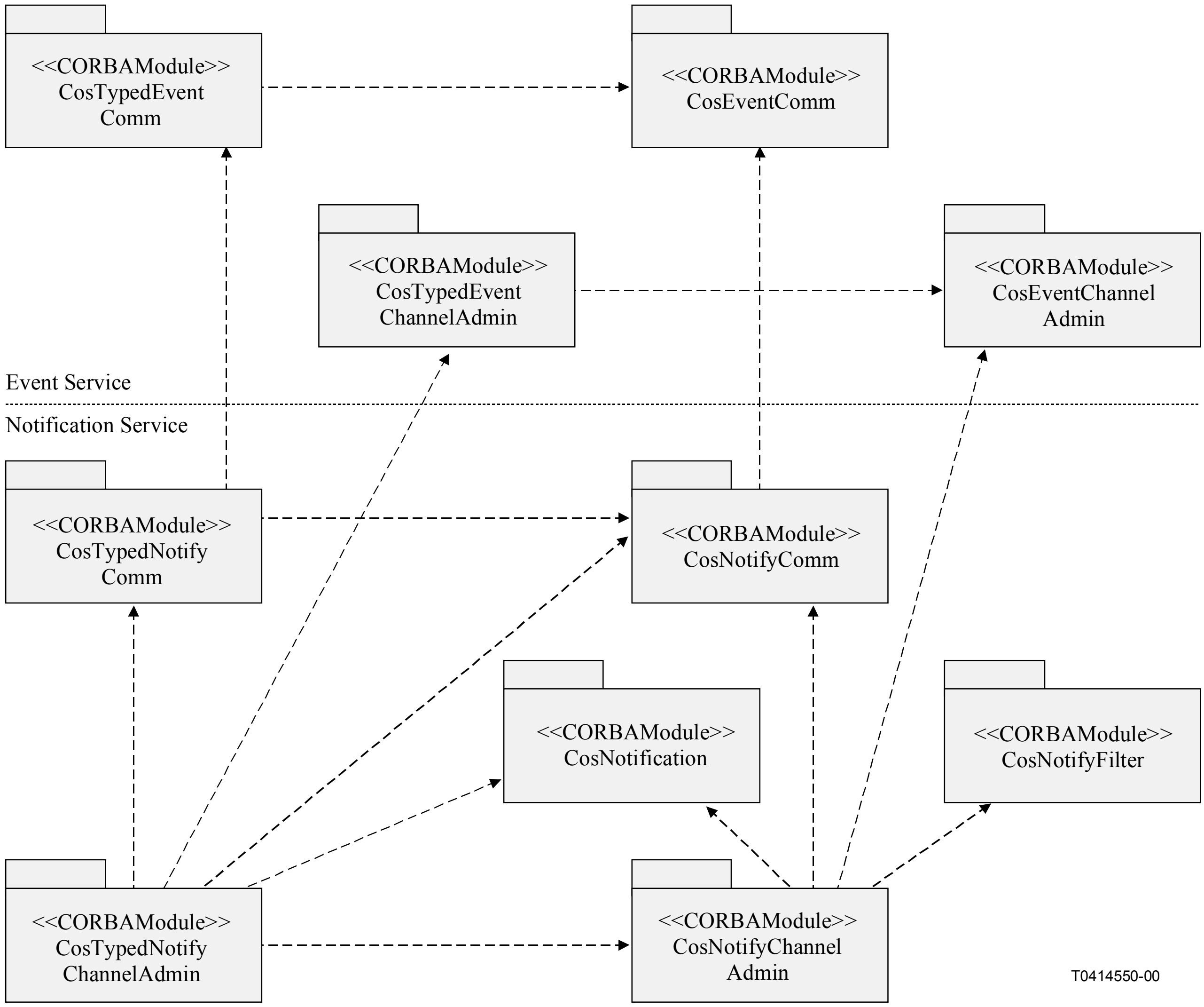


Figure 6-5/X.770 – Proxy suppliers

7 Computational interfaces for notification service

The detailed specifications of the computational interfaces described in this clause are contained in the OMG Notification service specification (OMG Document telecom/1999-07-01). See Figure 7-1.



<<CORBAModule>> CosTypedEvent Comm
<<CORBAModule>> CosEventComm
<<CORBAModule>> CosTypedEvent ChannelAdmin
<<CORBAModule>> CosEventChannel Admin
Event Service
Notification Service
<<CORBAModule>> CosTypedNotify Comm
<<CORBAModule>> CosNotifyComm
<<CORBAModule>> CosNotification
<<CORBAModule>> CosNotifyFilter
<<CORBAModule>> CosTypedNotify ChannelAdmin
<<CORBAModule>> CosNotifyChannel Admin
T0414550-00

Figure 7-1/X.770 – Module include dependencies for OMG notification service

The dashed lines in the figure show how each module in the notification service includes other modules either defined in notification service or COS event service.

7.1 CosEventComm module

See Figure 7-2.

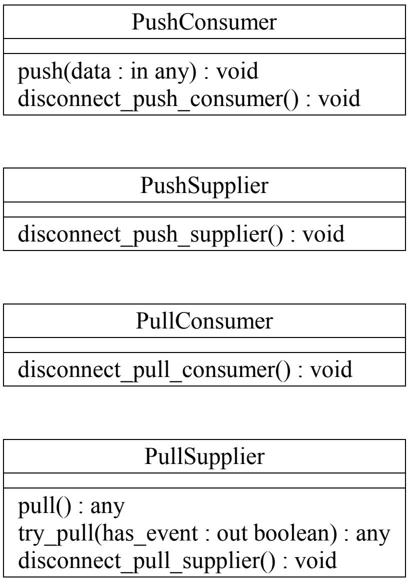

PushConsumer
push(data : in any) : void
disconnect_push_consumer() : void
PushSupplier
disconnect_push_supplier() : void
PullConsumer
disconnect_pull_consumer() : void
PullSupplier
pull() : any
try_pull(has_event : out boolean) : any
disconnect_pull_supplier() : void

Figure 7-2/X.770 – CosEventComm module interfaces

7.2 CosEventChannelAdmin module

See Figure 7-3.

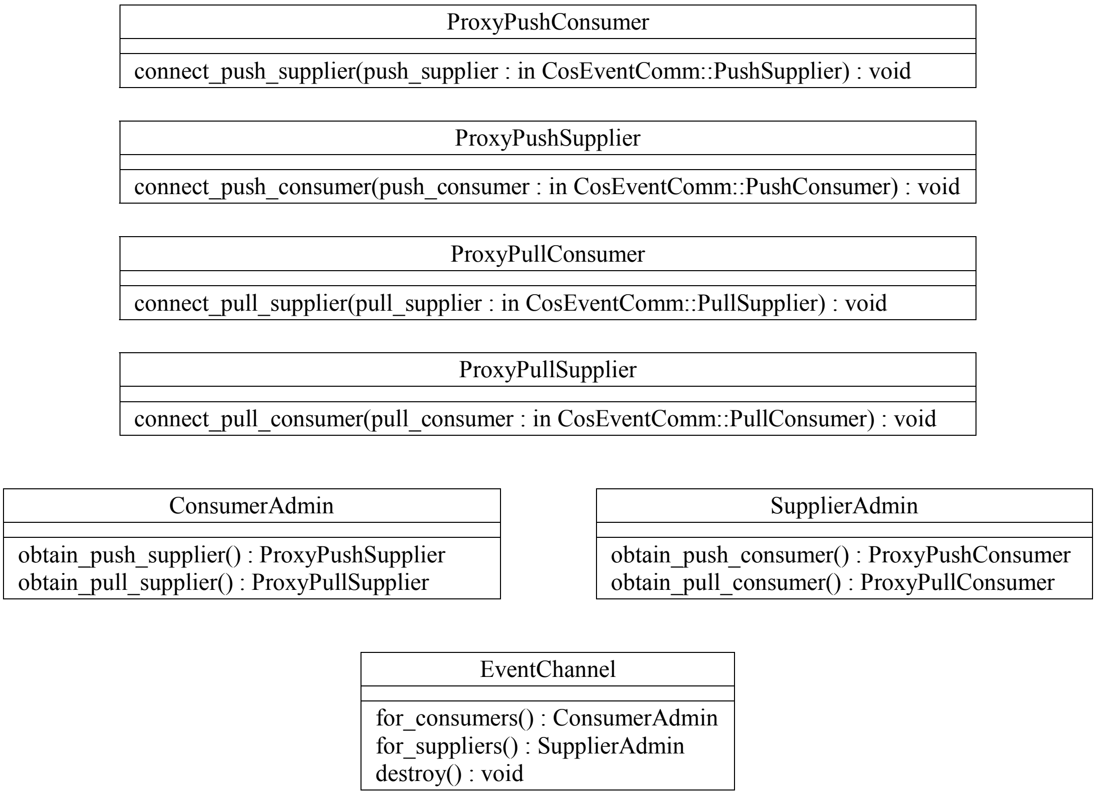

ProxyPushConsumer
connect_push_supplier(push_supplier : in CosEventComm::PushSupplier) : void
ProxyPushSupplier
connect_push_consumer(push_consumer : in CosEventComm::PushConsumer) : void
ProxyPullConsumer
connect_pull_supplier(pull_supplier : in CosEventComm::PullSupplier) : void
ProxyPullSupplier
connect_pull_consumer(pull_consumer : in CosEventComm::PullConsumer) : void
ConsumerAdmin
obtain_push_supplier() : ProxyPushSupplier
obtain_pull_supplier() : ProxyPullSupplier
SupplierAdmin
obtain_push_consumer() : ProxyPushConsumer
obtain_pull_consumer() : ProxyPullConsumer
EventChannel
for_consumers() : ConsumerAdmin
for_suppliers() : SupplierAdmin
destroy() : void

Figure 7-3/X.770 – CosEventChannelAdmin module interfaces

7.3 CosTypedEventComm module

See Figure 7-4.

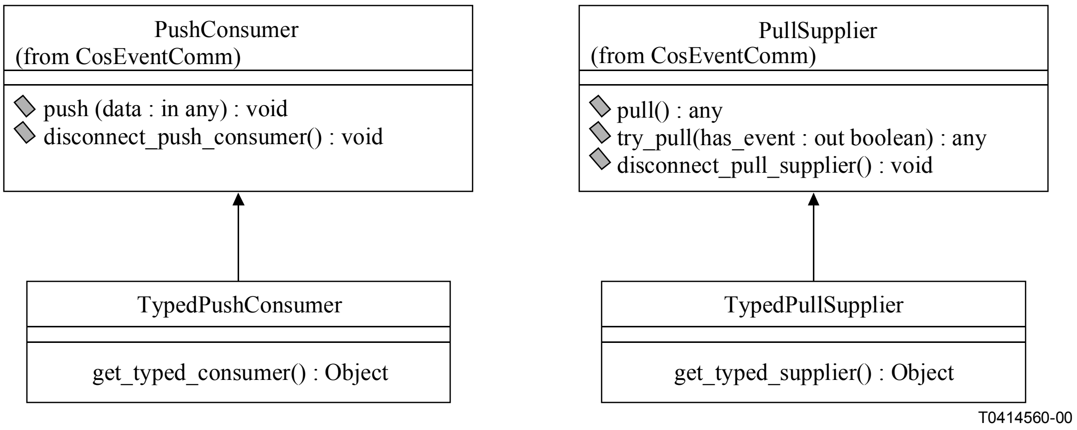

PushConsumer
(from CosEventComm)
push (data : in any) : void
disconnect_push_consumer() : void
PullSupplier
(from CosEventComm)
pull() : any
try_pull(has_event : out boolean) : any
disconnect_pull_supplier() : void
TypedPushConsumer
get_typed_consumer() : Object
TypedPullSupplier
get_typed_supplier() : Object
T0414560-00

Figure 7-4/X.770 – CosTypedEventComm module interfaces

7.4 CosTypedEventChannelAdmin module

See Figures 7-5 to 7-9.

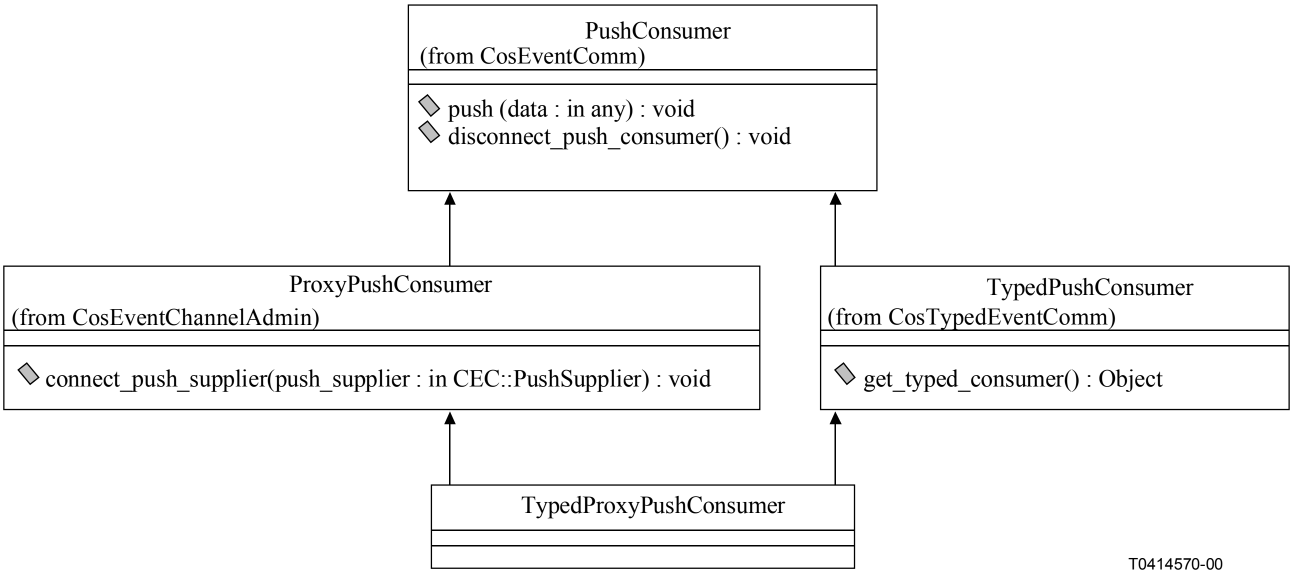
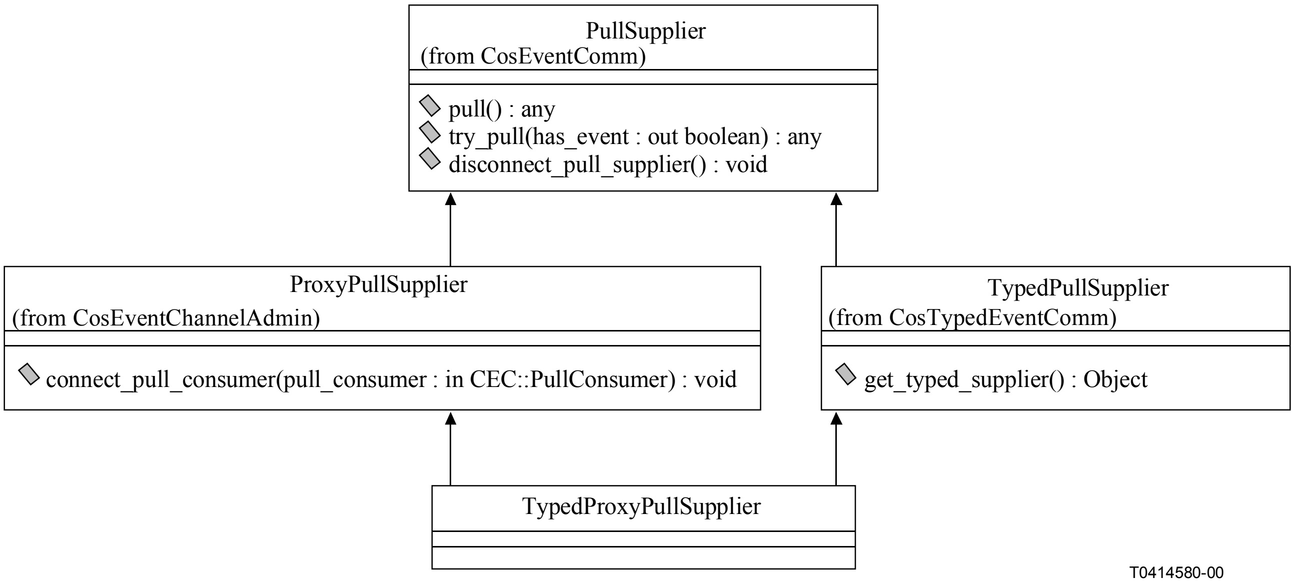
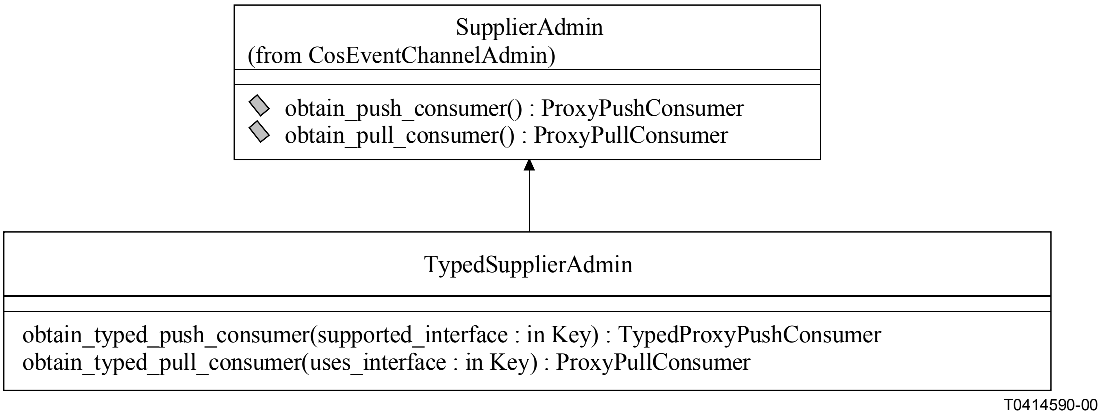

PushConsumer
(from CosEventComm)
push (data : in any) : void
disconnect_push_consumer() : void
ProxyPushConsumer
(from CosEventChannelAdmin)
connect_push_supplier(push_supplier : in CEC::PushSupplier) : void
TypedPushConsumer
(from CosTypedEventComm)
get_typed_consumer() : Object
TypedProxyPushConsumer
T0414570-00

Figure 7-5/X.770 – TypedProxyPushConsumer interface



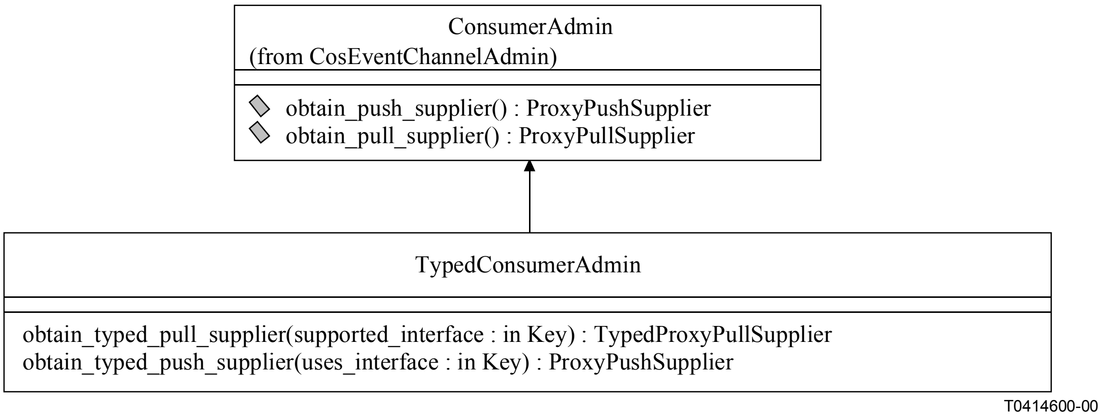
PullSupplier
(from CosEventComm)
pull() : any
try_pull(has_event : out boolean) : any
disconnect_pull_supplier() : void
ProxyPullSupplier
(from CosEventChannelAdmin)
connect_pull_consumer(pull_consumer : in CEC::PullConsumer) : void
TypedPullSupplier
(from CosTypedEventComm)
get_typed_supplier() : Object
TypedProxyPullSupplier
T0414580-00

Figure 7-6/X.770 – TypedProxyPullSupplier interfaces



SupplierAdmin
(from CosEventChannelAdmin)
obtain_push_consumer() : ProxyPushConsumer
obtain_pull_consumer() : ProxyPullConsumer
TypedSupplierAdmin
obtain_typed_push_consumer(supported_interface : in Key) : TypedProxyPushConsumer
obtain_typed_pull_consumer(uses_interface : in Key) : ProxyPullConsumer
T0414590-00

Figure 7-7/X.770 – TypedSupplierAdmin Interface



ConsumerAdmin
(from CosEventChannelAdmin)
obtain_push_supplier() : ProxyPushSupplier
obtain_pull_supplier() : ProxyPullSupplier
TypedConsumerAdmin
obtain_typed_pull_supplier(supported_interface : in Key) : TypedProxyPullSupplier
obtain_typed_push_supplier(uses_interface : in Key) : ProxyPushSupplier
T0414600-00

Figure 7-8/X.770 – TypedConsumerAdmin interface

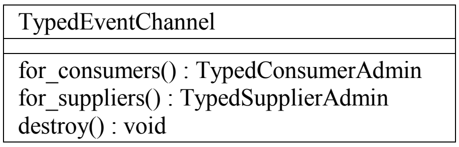

TypedEventChannel
for_consumers() : TypedConsumerAdmin
for_suppliers() : TypedSupplierAdmin
destroy() : void

Figure 7-9/X.770 – TypedEventChannel interface

7.5 CosNotifyComm module

See Figures 7-10 and 7-11.

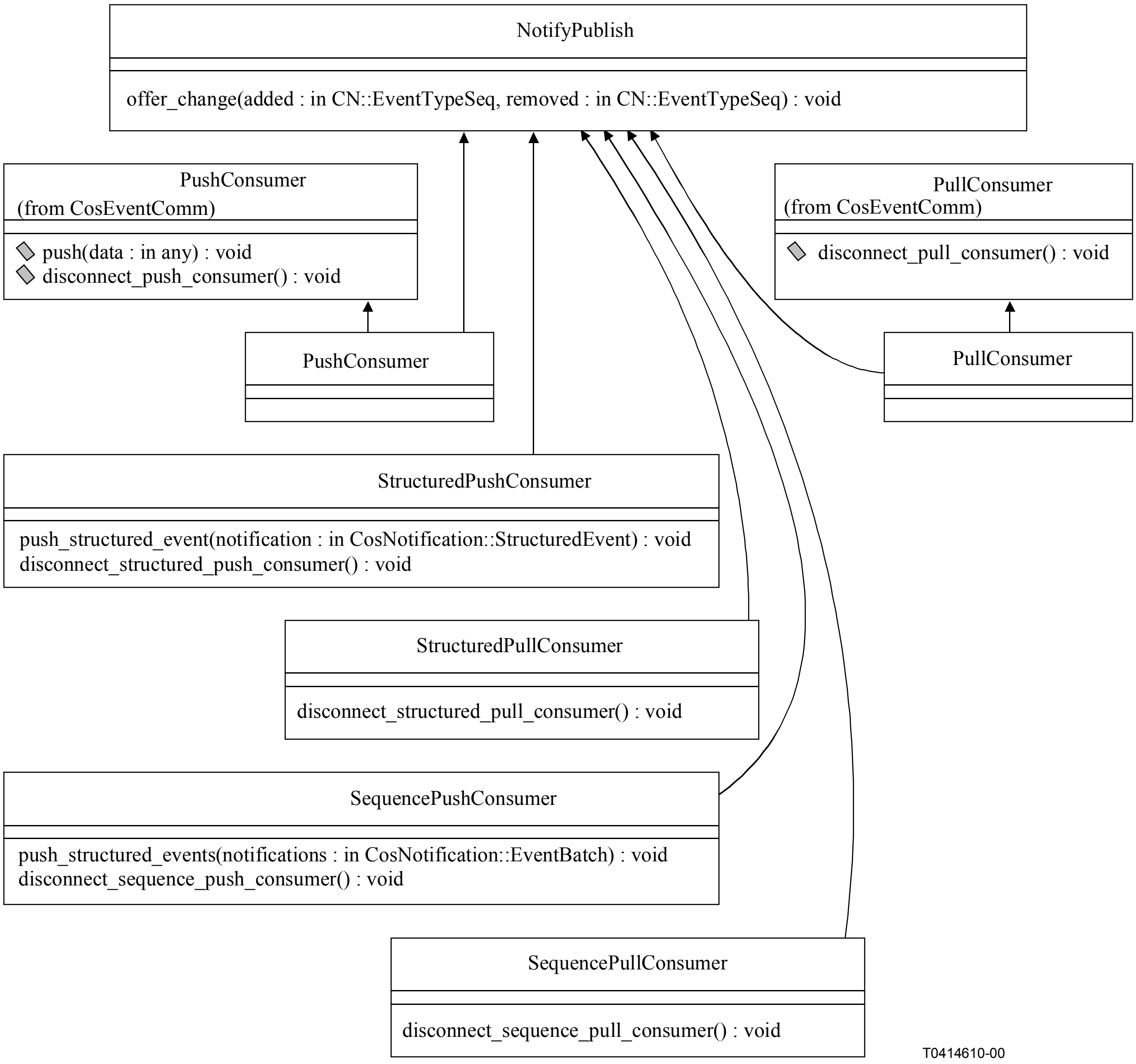

NotifyPublish
offer_change(added : in CN::EventTypeSeq, removed : in CN::EventTypeSeq) : void
PushConsumer
(from CosEventComm)
push(data : in any) : void
disconnect_push_consumer() : void
PullConsumer
(from CosEventComm)
disconnect_pull_consumer() : void
PushConsumer
PullConsumer
StructuredPushConsumer
push_structured_event(notification : in CosNotification::StructuredEvent) : void
disconnect_structured_push_consumer() : void
StructuredPullConsumer
disconnect_structured_pull_consumer() : void
SequencePushConsumer
push_structured_events(notifications : in CosNotification::EventBatch) : void
disconnect_sequence_push_consumer() : void
SequencePullConsumer
disconnect_sequence_pull_consumer() : void
T0414610-00

Figure 7-10/X.770 – Structured and SequenceConsumer interfaces

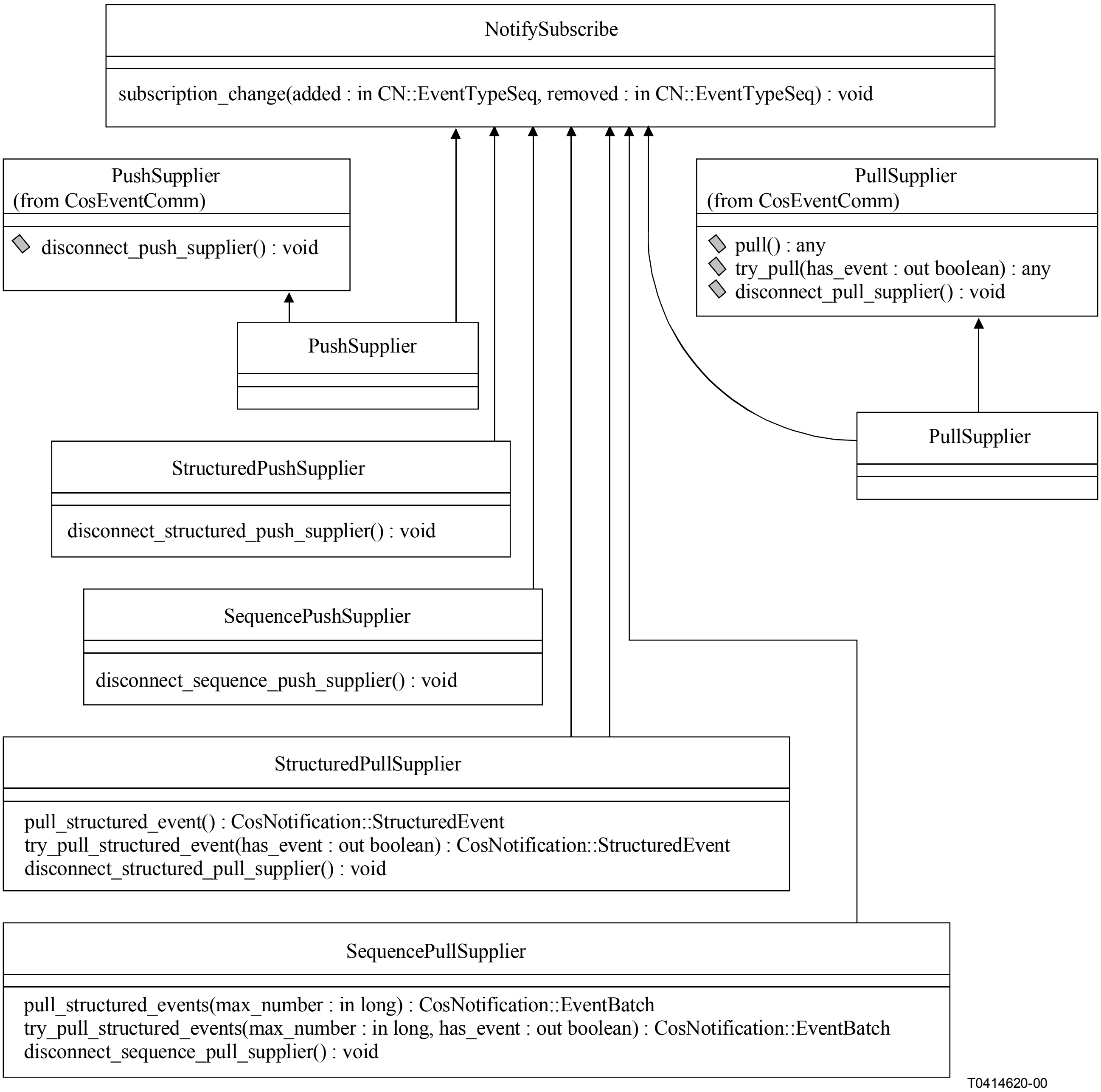
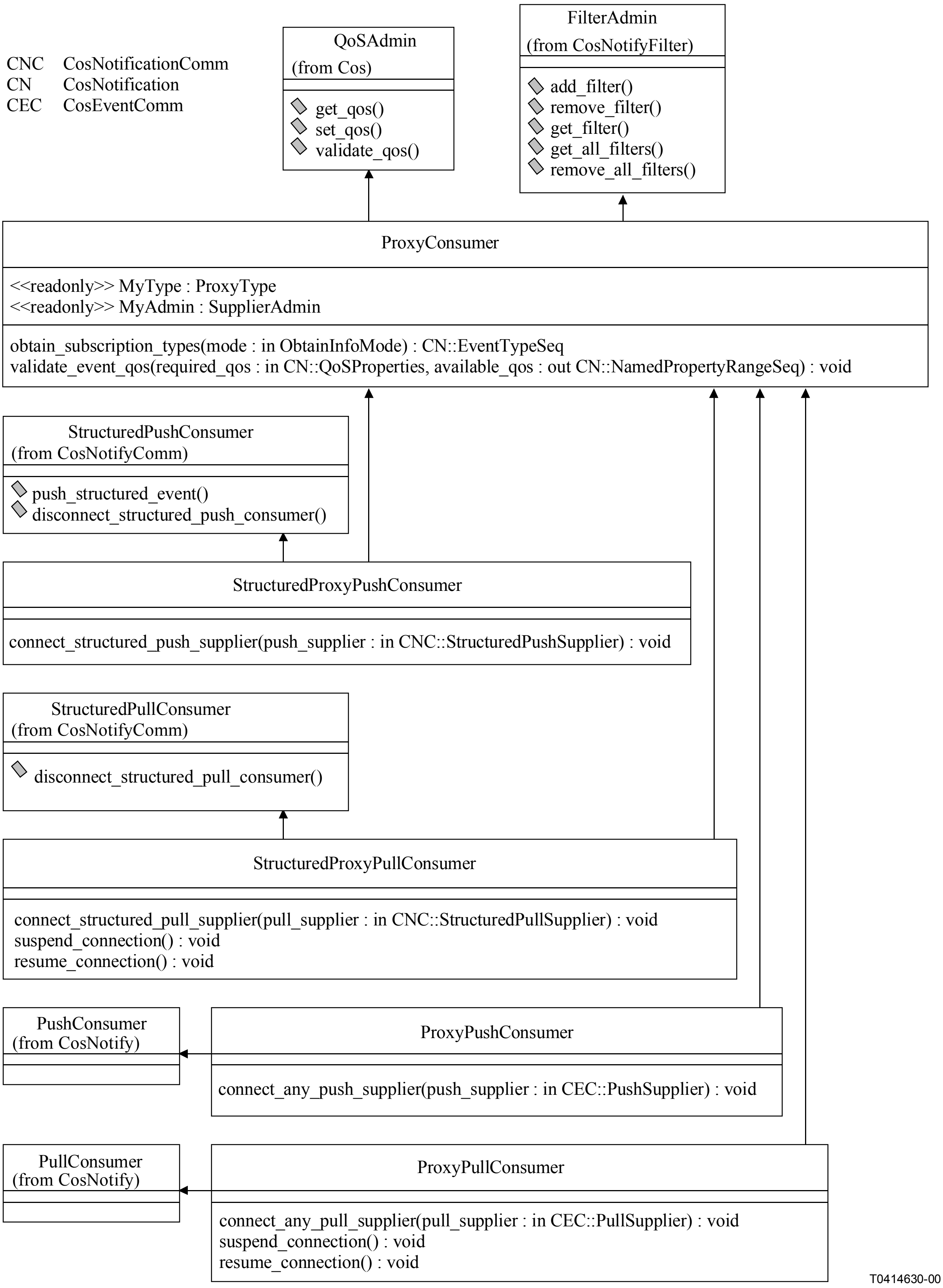

NotifySubscribe
subscription_change(added : in CN::EventTypeSeq, removed : in CN::EventTypeSeq) : void
PushSupplier
(from CosEventComm)
disconnect_push_supplier() : void
PullSupplier
(from CosEventComm)
pull() : any
try_pull(has_event : out boolean) : any
disconnect_pull_supplier() : void
PushSupplier
PullSupplier
StructuredPushSupplier
disconnect_structured_push_supplier() : void
SequencePushSupplier
disconnect_sequence_push_supplier() : void
StructuredPullSupplier
pull_structured_event() : CosNotification::StructuredEvent
try_pull_structured_event(has_event : out boolean) : CosNotification::StructuredEvent
disconnect_structured_pull_supplier() : void
SequencePullSupplier
pull_structured_events(max_number : in long) : CosNotification::EventBatch
try_pull_structured_events(max_number : in long, has_event : out boolean) : CosNotification::EventBatch
disconnect_sequence_pull_supplier() : void
T0414620-00

Figure 7-11/X.770 – Structured and Sequence Supplier interfaces

7.6 CosNotifyChannelAdmin module

See Figures 7-12 to 7-17.



CNC CosNotificationComm
CN CosNotification
CEC CosEventComm
QoSAdmin
(from Cos)
get_qos()
set_qos()
validate_qos()
FilterAdmin
(from CosNotifyFilter)
add_filter()
remove_filter()
get_filter()
get_all_filters()
remove_all_filters()
ProxyConsumer
<<readonly>> MyType : ProxyType
<<readonly>> MyAdmin : SupplierAdmin
obtain_subscription_types(mode : in ObtainInfoMode) : CN::EventTypeSeq
validate_event_qos(required_qos : in CN::QoSProperties, available_qos : out CN::NamedPropertyRangeSeq) : void
StructuredPushConsumer
(from CosNotifyComm)
push_structured_event()
disconnect_structured_push_consumer()
StructuredProxyPushConsumer
connect_structured_push_supplier(push_supplier : in CNC::StructuredPushSupplier) : void
StructuredPullConsumer
(from CosNotifyComm)
disconnect_structured_pull_consumer()
StructuredProxyPullConsumer
connect_structured_pull_supplier(pull_supplier : in CNC::StructuredPullSupplier) : void
suspend_connection() : void
resume_connection() : void
PushConsumer
(from CosNotify)
ProxyPushConsumer
connect_any_push_supplier(push_supplier : in CEC::PushSupplier) : void
PullConsumer
(from CosNotify)
ProxyPullConsumer
connect_any_pull_supplier(pull_supplier : in CEC::PullSupplier) : void
suspend_connection() : void
resume_connection() : void
T0414630-00

Figure 7-12/X.770 – StructuredProxy and Proxy Consumer interfaces

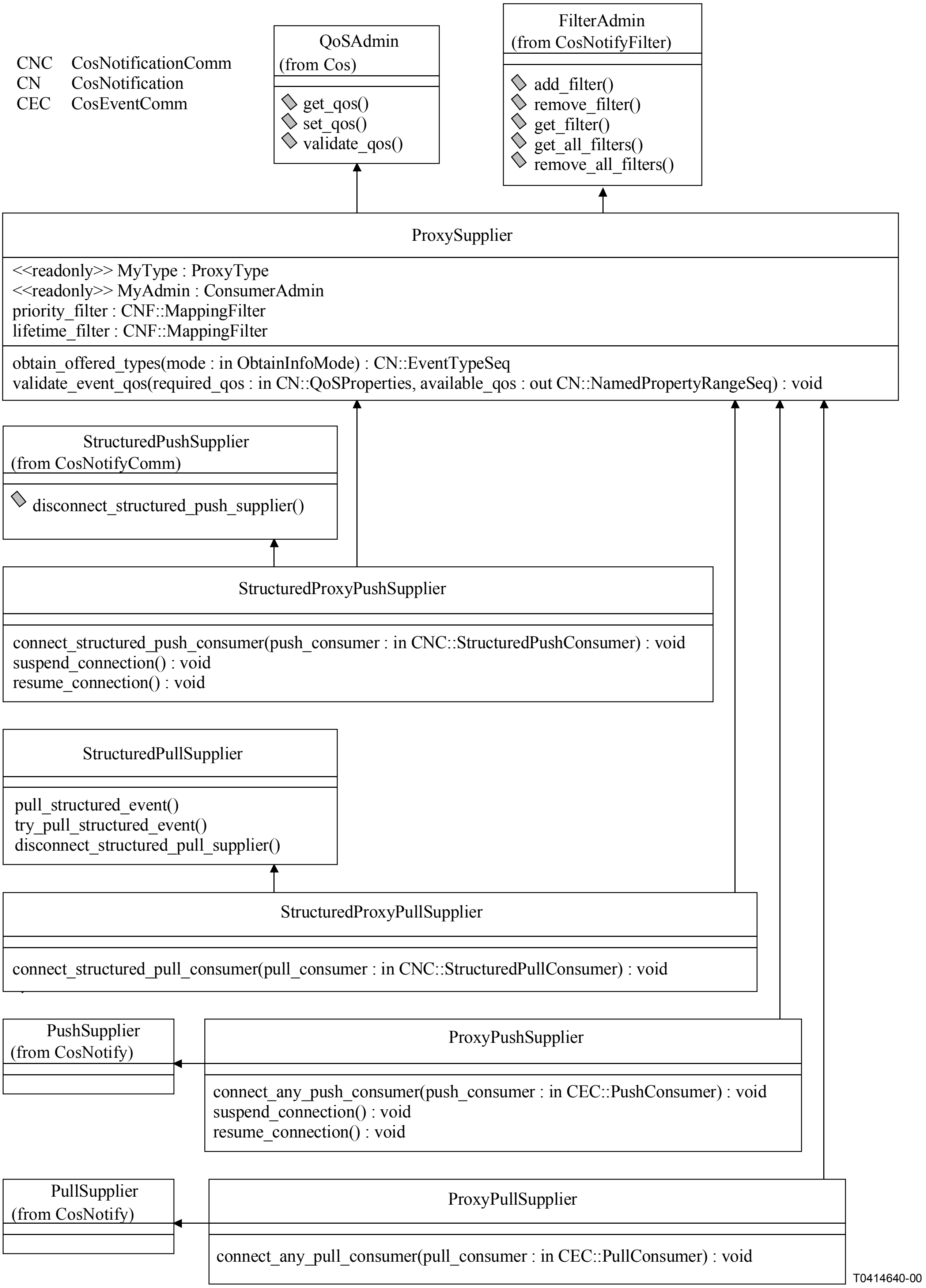

CNC CosNotificationComm
CN CosNotification
CEC CosEventComm
QoSAdmin
(from Cos)
get_qos()
set_qos()
validate_qos()
FilterAdmin
(from CosNotifyFilter)
add_filter()
remove_filter()
get_filter()
get_all_filters()
remove_all_filters()
ProxySupplier
<<readonly>> MyType : ProxyType
<<readonly>> MyAdmin : ConsumerAdmin
priority_filter : CNF::MappingFilter
lifetime_filter : CNF::MappingFilter
obtain_offered_types(mode : in ObtainInfoMode) : CN::EventTypeSeq
validate_event_qos(required_qos : in CN::QoSProperties, available_qos : out CN::NamedPropertyRangeSeq) : void
StructuredPushSupplier
(from CosNotifyComm)
disconnect_structured_push_supplier()
StructuredProxyPushSupplier
connect_structured_push_consumer(push_consumer : in CNC::StructuredPushConsumer) : void
suspend_connection() : void
resume_connection() : void
StructuredPullSupplier
pull_structured_event()
try_pull_structured_event()
disconnect_structured_pull_supplier()
StructuredProxyPullSupplier
connect_structured_pull_consumer(pull_consumer : in CNC::StructuredPullConsumer) : void
PushSupplier
(from CosNotify)
ProxyPushSupplier
connect_any_push_consumer(push_consumer : in CEC::PushConsumer) : void
suspend_connection() : void
resume_connection() : void
PullSupplier
(from CosNotify)
ProxyPullSupplier
connect_any_pull_consumer(pull_consumer : in CEC::PullConsumer) : void
T0414640-00

Figure 7-13/X.770 – StructuredProxy and Proxy Supplier interfaces

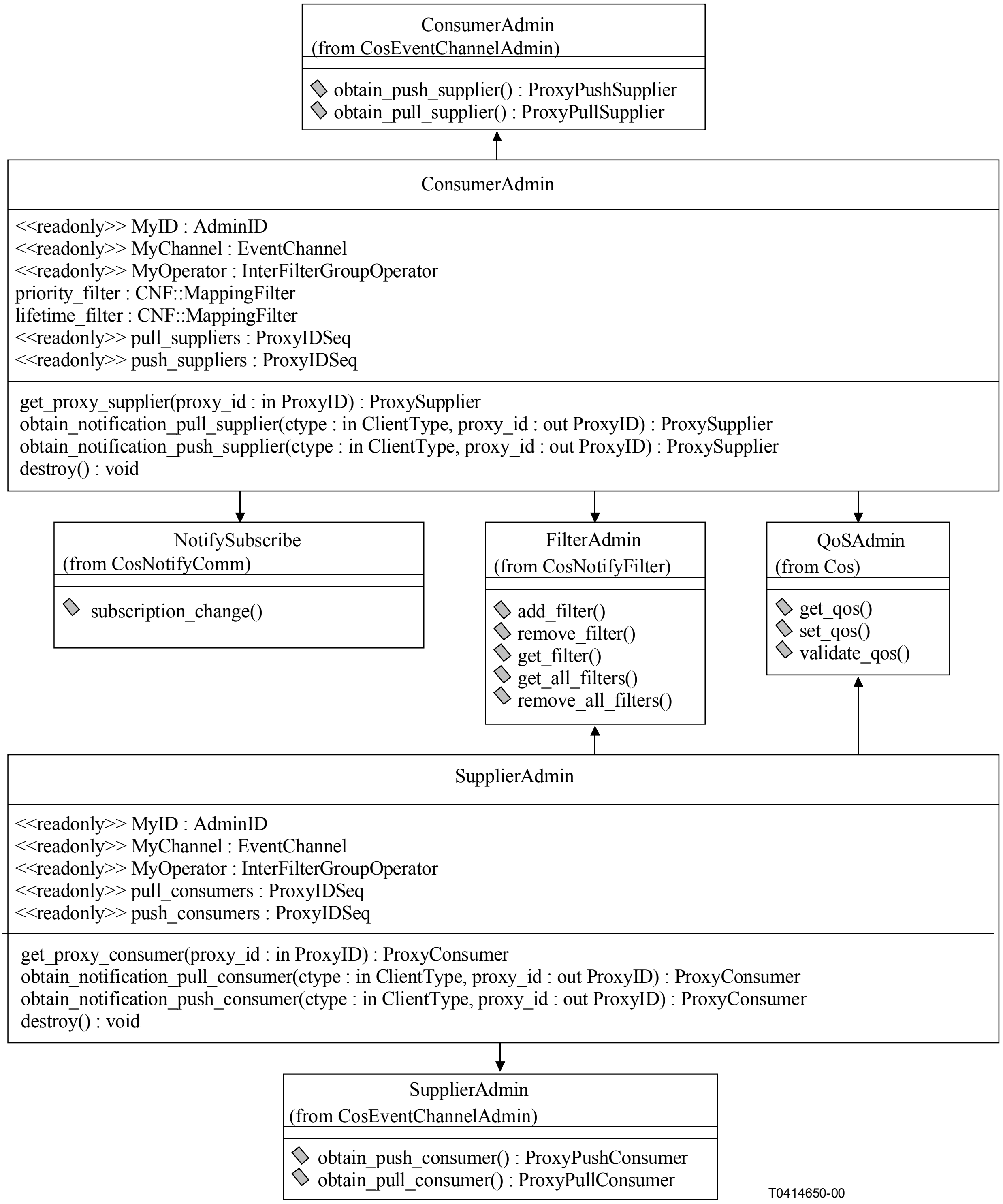

ConsumerAdmin
(from CosEventChannelAdmin)
obtain_push_supplier() : ProxyPushSupplier
obtain_pull_supplier() : ProxyPullSupplier
ConsumerAdmin
<<readonly>> MyID : AdminID
<<readonly>> MyChannel : EventChannel
<<readonly>> MyOperator : InterFilterGroupOperator
priority_filter : CNF::MappingFilter
lifetime_filter : CNF::MappingFilter
<<readonly>> pull_suppliers : ProxyIDSeq
<<readonly>> push_suppliers : ProxyIDSeq
get_proxy_supplier(proxy_id : in ProxyID) : ProxySupplier
obtain_notification_pull_supplier(ctype : in ClientType, proxy_id : out ProxyID) : ProxySupplier
obtain_notification_push_supplier(ctype : in ClientType, proxy_id : out ProxyID) : ProxySupplier
destroy() : void
NotifySubscribe
(from CosNotifyComm)
subscription_change()
FilterAdmin
(from CosNotifyFilter)
add_filter()
remove_filter()
get_filter()
get_all_filters()
remove_all_filters()
QoSAdmin
(from Cos)
get_qos()
set_qos()
validate_qos()
SupplierAdmin
<<readonly>> MyID : AdminID
<<readonly>> MyChannel : EventChannel
<<readonly>> MyOperator : InterFilterGroupOperator
<<readonly>> pull_consumers : ProxyIDSeq
<<readonly>> push_consumers : ProxyIDSeq
get_proxy_consumer(proxy_id : in ProxyID) : ProxyConsumer
obtain_notification_pull_consumer(ctype : in ClientType, proxy_id : out ProxyID) : ProxyConsumer
obtain_notification_push_consumer(ctype : in ClientType, proxy_id : out ProxyID) : ProxyConsumer
destroy() : void
SupplierAdmin
(from CosEventChannelAdmin)
obtain_push_consumer() : ProxyPushConsumer
obtain_pull_consumer() : ProxyPullConsumer
T0414650-00

Figure 7-14/X.770 – Consumer and Supplier Admin interfaces

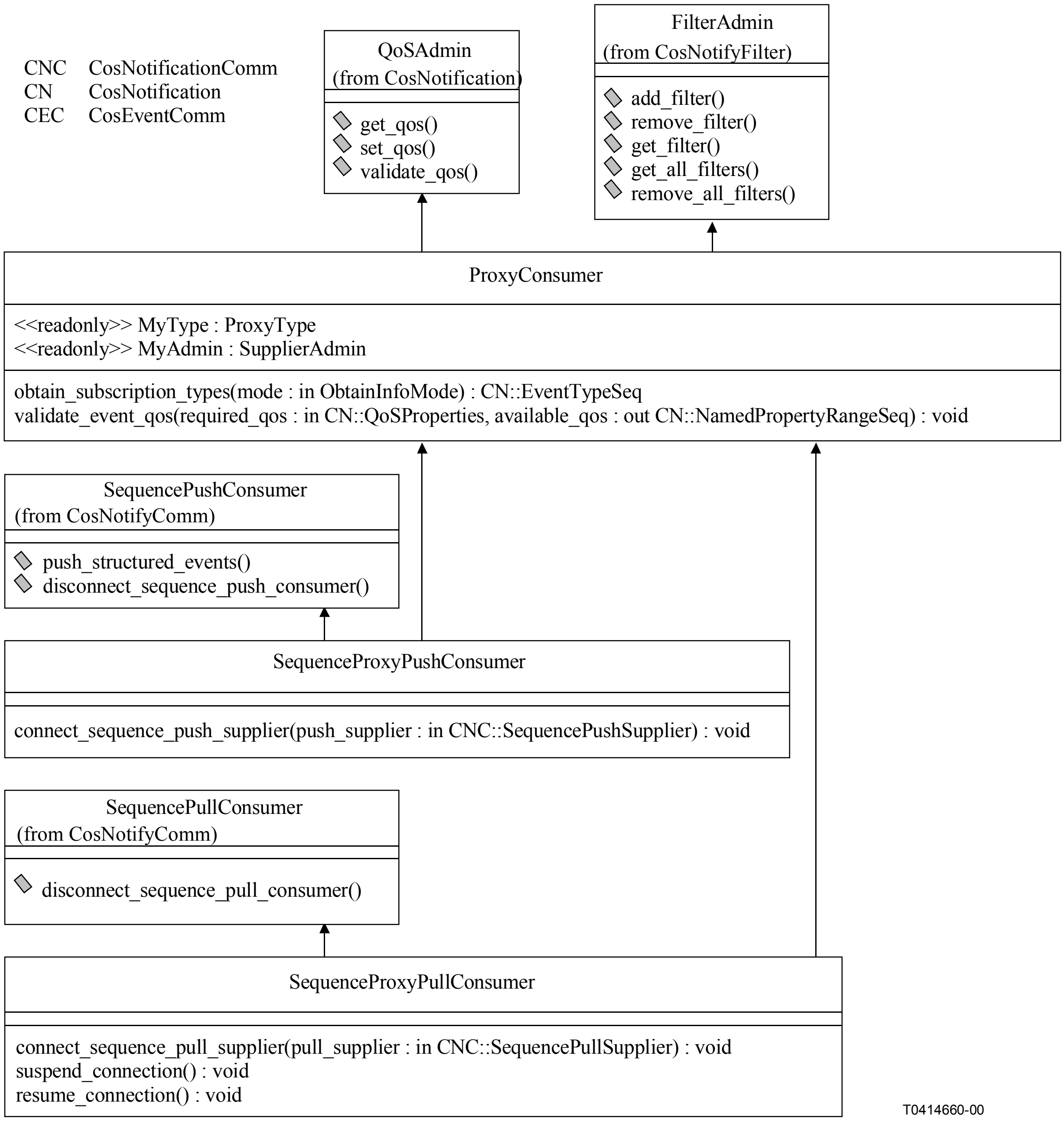

CNC CosNotificationComm
CN CosNotification
CEC CosEventComm
QoSAdmin
(from CosNotification)
get_qos()
set_qos()
validate_qos()
FilterAdmin
(from CosNotifyFilter)
add_filter()
remove_filter()
get_filter()
get_all_filters()
remove_all_filters()
ProxyConsumer
<<readonly>> MyType : ProxyType
<<readonly>> MyAdmin : SupplierAdmin
obtain_subscription_types(mode : in ObtainInfoMode) : CN::EventTypeSeq
validate_event_qos(required_qos : in CN::QoSProperties, available_qos : out CN::NamedPropertyRangeSeq) : void
SequencePushConsumer
(from CosNotifyComm)
push_structured_events()
disconnect_sequence_push_consumer()
SequenceProxyPushConsumer
connect_sequence_push_supplier(push_supplier : in CNC::SequencePushSupplier) : void
SequencePullConsumer
(from CosNotifyComm)
disconnect_sequence_pull_consumer()
SequenceProxyPullConsumer
connect_sequence_pull_supplier(pull_supplier : in CNC::SequencePullSupplier) : void
suspend_connection() : void
resume_connection() : void
T0414660-00

Figure 7-15/X.770 – SequenceProxy Consumer interfaces

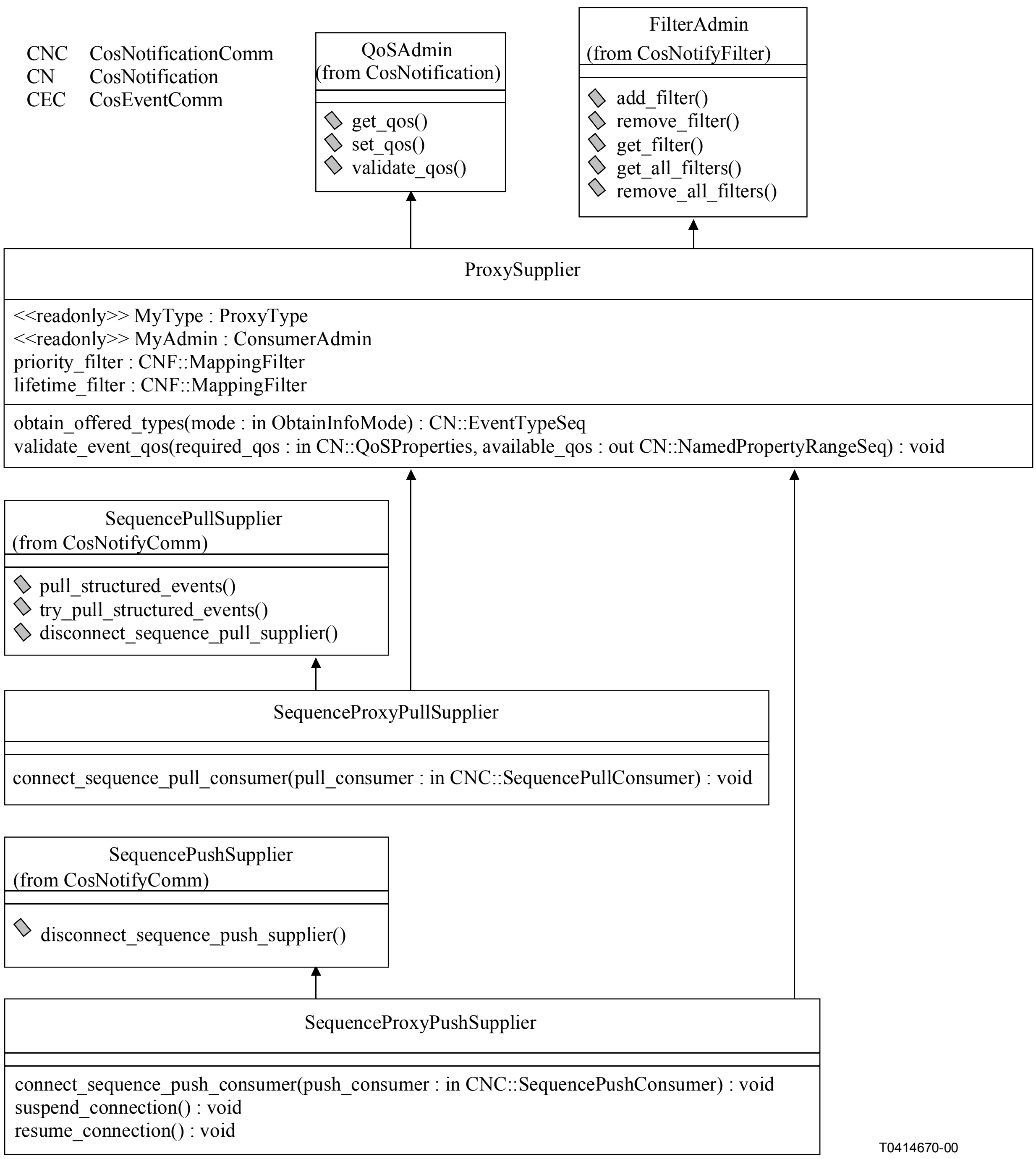

CNC CosNotificationComm
CN CosNotification
CEC CosEventComm
QoSAdmin
(from CosNotification)
get_qos()
set_qos()
validate_qos()
FilterAdmin
(from CosNotifyFilter)
add_filter()
remove_filter()
get_filter()
get_all_filters()
remove_all_filters()
ProxySupplier
<<readonly>> MyType : ProxyType
<<readonly>> MyAdmin : ConsumerAdmin
priority_filter : CNF::MappingFilter
lifetime_filter : CNF::MappingFilter
obtain_offered_types(mode : in ObtainInfoMode) : CN::EventTypeSeq
validate_event_qos(required_qos : in CN::QoSProperties, available_qos : out CN::NamedPropertyRangeSeq) : void
SequencePullSupplier
(from CosNotifyComm)
pull_structured_events()
try_pull_structured_events()
disconnect_sequence_pull_supplier()
SequenceProxyPullSupplier
connect_sequence_pull_consumer(pull_consumer : in CNC::SequencePullConsumer) : void
SequencePushSupplier
(from CosNotifyComm)
disconnect_sequence_push_supplier()
SequenceProxyPushSupplier
connect_sequence_push_consumer(push_consumer : in CNC::SequencePushConsumer) : void
suspend_connection() : void
resume_connection() : void
T0414670-00

Figure 7-16/X.770 – SequenceProxy supplier interfaces

CNC	CosNotificationComm
CN	CosNotification
CEC	CosEventComm

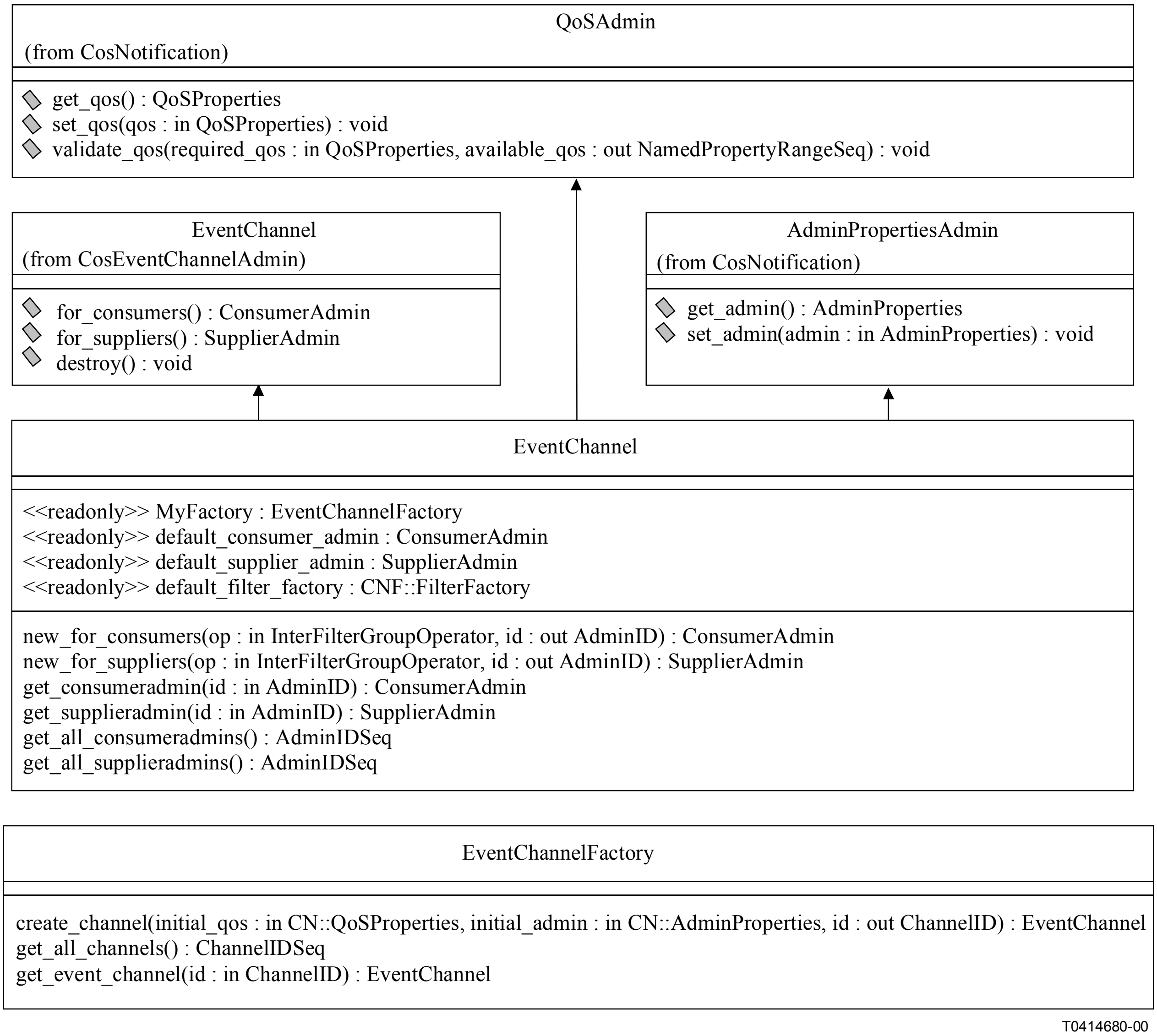

QoSAdmin
(from CosNotification)
get_qos() : QoSProperties
set_qos(qos : in QoSProperties) : void
validate_qos(required_qos : in QoSProperties, available_qos : out NamedPropertyRangeSeq) : void
EventChannel
(from CosEventChannelAdmin)
for_consumers() : ConsumerAdmin
for_suppliers() : SupplierAdmin
destroy() : void
AdminPropertiesAdmin
(from CosNotification)
get_admin() : AdminProperties
set_admin(admin : in AdminProperties) : void
EventChannel
<<readonly>> MyFactory : EventChannelFactory
<<readonly>> default_consumer_admin : ConsumerAdmin
<<readonly>> default_supplier_admin : SupplierAdmin
<<readonly>> default_filter_factory : CNF::FilterFactory
new_for_consumers(op : in InterFilterGroupOperator, id : out AdminID) : ConsumerAdmin
new_for_suppliers(op : in InterFilterGroupOperator, id : out AdminID) : SupplierAdmin
get_consumeradmin(id : in AdminID) : ConsumerAdmin
get_supplieradmin(id : in AdminID) : SupplierAdmin
get_all_consumeradmins() : AdminIDSeq
get_all_supplieradmins() : AdminIDSeq
EventChannelFactory
create_channel(initial_qos : in CN::QoSProperties, initial_admin : in CN::AdminProperties, id : out ChannelID) : EventChannel
get_all_channels() : ChannelIDSeq
get_event_channel(id : in ChannelID) : EventChannel
T0414680-00

Figure 7-17/X.770 – EventChannel and EventChannel Factory interfaces

7.7 CosNotification module

See Figures 7-18 and 7-19.

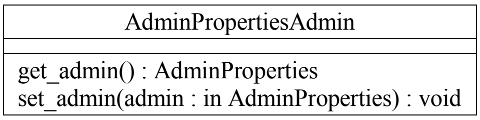

AdminPropertiesAdmin
get_admin() : AdminProperties
set_admin(admin : in AdminProperties) : void

Figure 7-18/X.770 – AdminProperties Admin interface

QoSAdmin
get_qos() : QoSProperties set_qos(qos : in QoSProperties) : void validate_qos(required_qos : in QoSProperties, available_qos : out NamePropertyRangeSeq) : void

Figure 7-19/X.770 – QoSAdmin interface

7.8 CosNotifyFilter module

See Figures 7-20 to 7-23.

Filter
<<readonly>> constraint_grammar : string
add_constraints(constraint_list : in ConstraintExpSeq) : ConstraintInfoSeq modify_constraints(del_list : in ConstraintIDSeq, modify_list : in ConstraintInfoSeq) : void get_constraints(id_list : in ConstraintIDSeq) : ConstraintInfoSeq get_all_constraints() : ConstraintInfoSeq remove_all_constraints() : void destroy() : void match(filterable_data : in any) : boolean match_structured(filterable_data : in CosNotification::StructuredEvent) : boolean match_typed(filterable_data : in CosNotification::PropertySeq) : boolean attach_callback(callback : in CosNotifyComm::NotifySubscribe) : CallbackID detach_callback(callback : in CallbackID) : void get_callbacks() : CallbackIDSeq

Figure 7-20/X.770 – Filter interface

MappingFilter
<<readonly>> constraint_grammar : string <<readonly>> value_type : CORBA : TypeCode <<readonly>> default_value : any
add_mapping_constraints(pair_list : in MappingConstraintPairSeq) : MappingConstraintInfoSeq modify_mapping_constraints(del_list : in ConstraintIDSeq, modify_list : in MappingConstraintInfoSeq) : void get_mapping_constraints(id_list : in ConstraintIDSeq) : MappingConstraintInfoSeq get_all_mapping_constraints() : MappingConstraintInfoSeq remove_all_mapping_constraints() : void destroy() : void match(filterable_data : in any, result_to_set : out any) : boolean match_structured(filterable_data : in CosNotification::StructuredEvent, result_to_set : out any) : boolean match_typed(filterable_data : in CosNotification::PropertySeq, result_to_set : out any) : boolean

Figure 7-21/X.770 – Mapping Filter interface

FilterAdmin
add_filter(new_filter : in Filter) : FilterID remove_filter(filter : in FilterID) : void get_filter(filter : in FilterID) : Filter get_all_filters() : FilterIDSeq remove_all_filters() : void

Figure 7-22/X.770 – Filter Admin interface

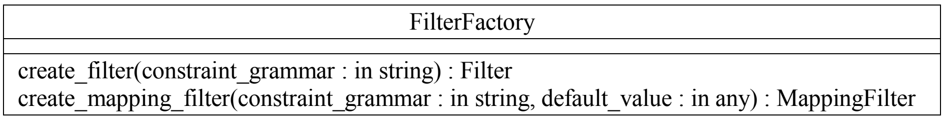

FilterFactory
create_filter(constraint_grammar : in string) : Filter
create_mapping_filter(constraint_grammar : in string, default_value : in any) : MappingFilter

Figure 7-23/X.770 – Filter Factory interface

7.9 CosTypedNotifyComm module

See Figures 7-24 and 7-25.

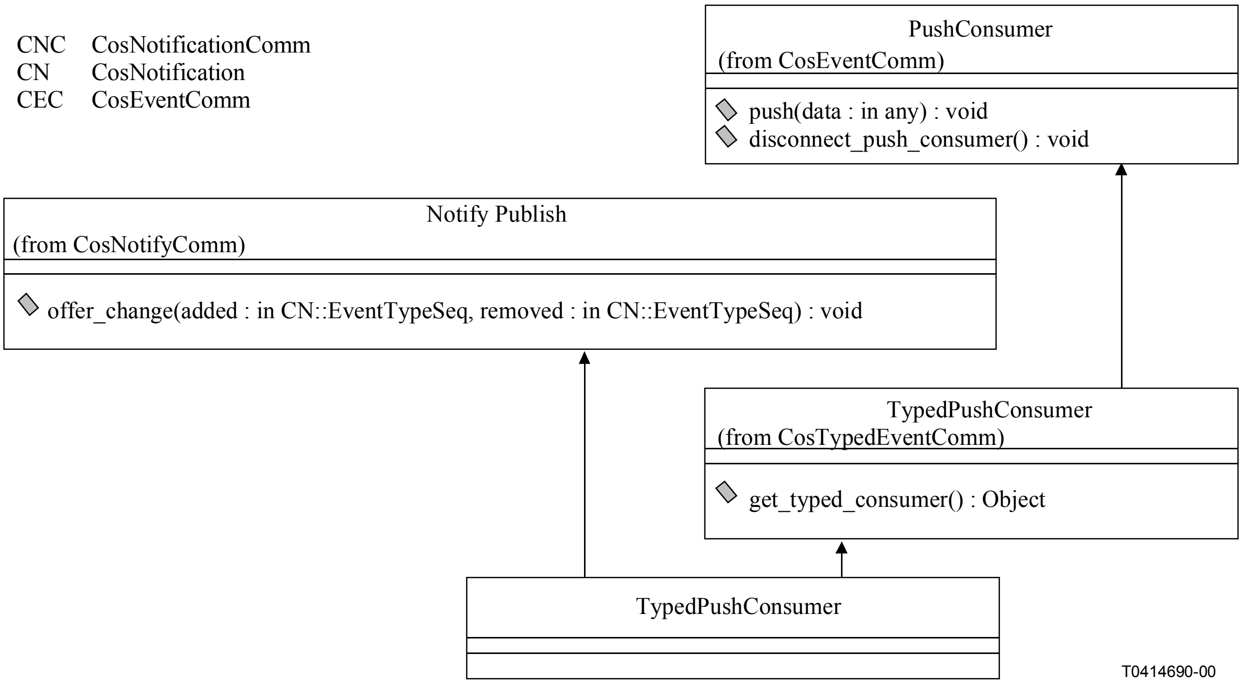


Figure 7-24/X.770 – TypedPush Consumer interface

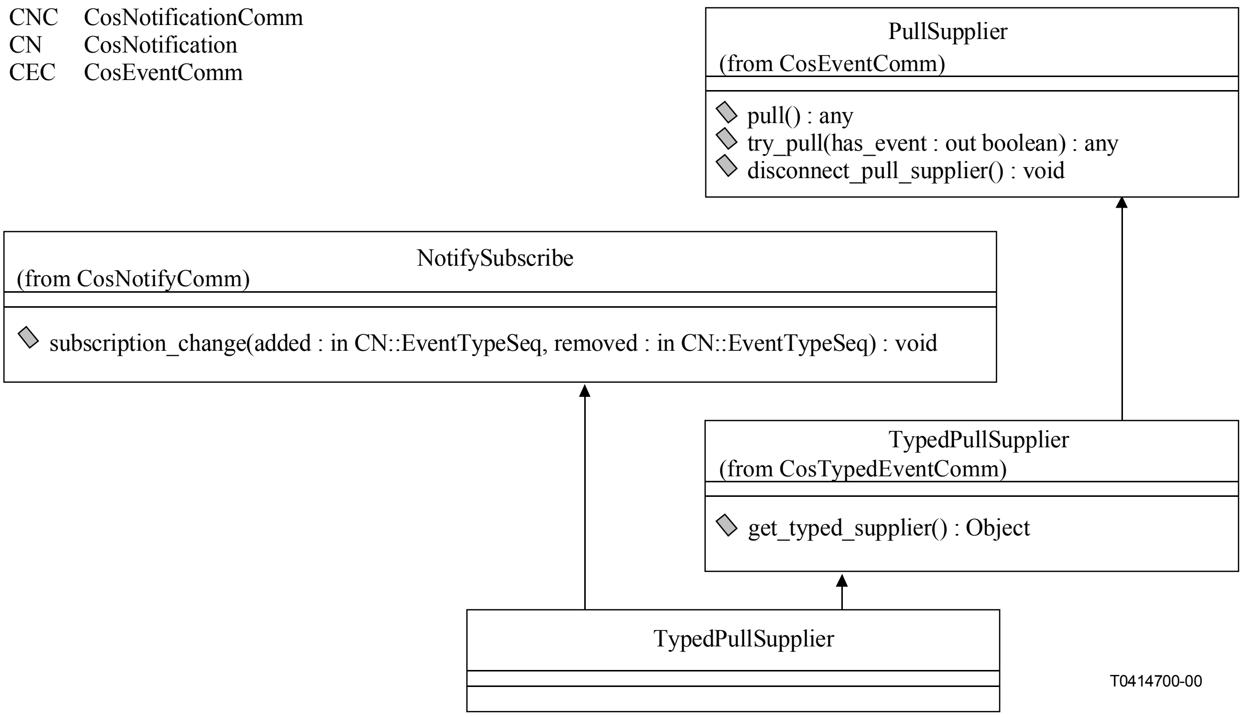
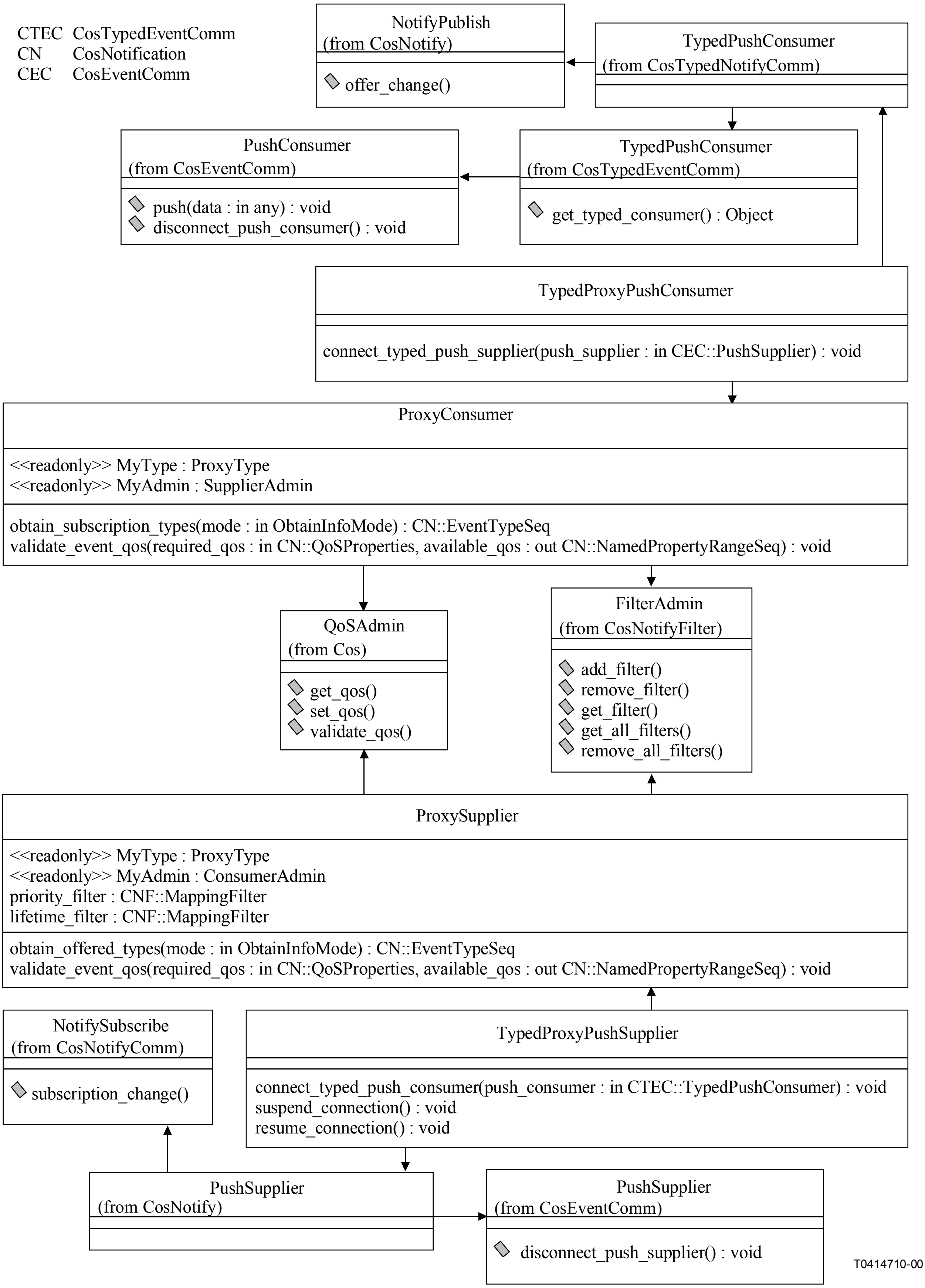


Figure 7-25/X.770 – TypedPull Supplier interface

7.10 CosTypedNotifyChannelAdmin module

See Figures 7-26 to 7-31.



CTEC CosTypedEventComm
CN CosNotification
CEC CosEventComm
NotifyPublish
(from CosNotify)
offer_change()
TypedPushConsumer
(from CosTypedNotifyComm)
PushConsumer
(from CosEventComm)
push(data : in any) : void
disconnect_push_consumer() : void
TypedPushConsumer
(from CosTypedEventComm)
get_typed_consumer() : Object
TypedProxyPushConsumer
connect_typed_push_supplier(push_supplier : in CEC::PushSupplier) : void
ProxyConsumer
<<readonly>> MyType : ProxyType
<<readonly>> MyAdmin : SupplierAdmin
obtain_subscription_types(mode : in ObtainInfoMode) : CN::EventTypeSeq
validate_event_qos(required_qos : in CN::QoSProperties, available_qos : out CN::NamedPropertyRangeSeq) : void
QoSAdmin
(from Cos)
get_qos()
set_qos()
validate_qos()
FilterAdmin
(from CosNotifyFilter)
add_filter()
remove_filter()
get_filter()
get_all_filters()
remove_all_filters()
ProxySupplier
<<readonly>> MyType : ProxyType
<<readonly>> MyAdmin : ConsumerAdmin
priority_filter : CNF::MappingFilter
lifetime_filter : CNF::MappingFilter
obtain_offered_types(mode : in ObtainInfoMode) : CN::EventTypeSeq
validate_event_qos(required_qos : in CN::QoSProperties, available_qos : out CN::NamedPropertyRangeSeq) : void
NotifySubscribe
(from CosNotifyComm)
subscription_change()
TypedProxyPushSupplier
connect_typed_push_consumer(push_consumer : in CTEC::TypedPushConsumer) : void
suspend_connection() : void
resume_connection() : void
PushSupplier
(from CosNotify)
PushSupplier
(from CosEventComm)
disconnect_push_supplier() : void
T0414710-00

Figure 7-26/X.770 – TypedProxyPush Consumer and Supplier interfaces

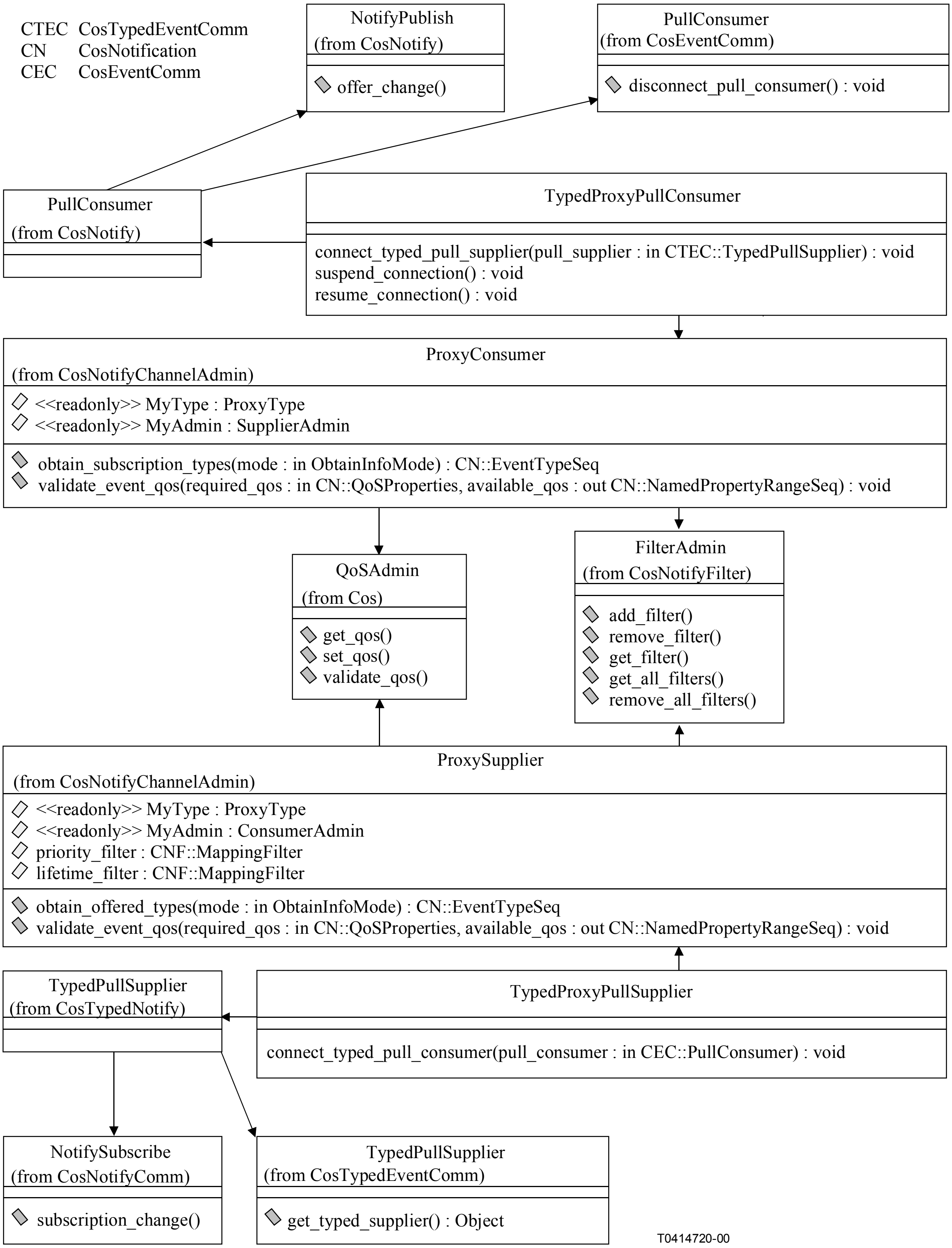

CTEC CosTypedEventComm
CN CosNotification
CEC CosEventComm
NotifyPublish
(from CosNotify)
offer_change()
PullConsumer
(from CosEventComm)
disconnect_pull_consumer() : void
PullConsumer
(from CosNotify)
TypedProxyPullConsumer
connect_typed_pull_supplier(pull_supplier : in CTEC::TypedPullSupplier) : void
suspend_connection() : void
resume_connection() : void
ProxyConsumer
(from CosNotifyChannelAdmin)
<<readonly>> MyType : ProxyType
<<readonly>> MyAdmin : SupplierAdmin
obtain_subscription_types(mode : in ObtainInfoMode) : CN::EventTypeSeq
validate_event_qos(required_qos : in CN::QoSProperties, available_qos : out CN::NamedPropertyRangeSeq) : void
QoSAdmin
(from Cos)
get_qos()
set_qos()
validate_qos()
FilterAdmin
(from CosNotifyFilter)
add_filter()
remove_filter()
get_filter()
get_all_filters()
remove_all_filters()
ProxySupplier
(from CosNotifyChannelAdmin)
<<readonly>> MyType : ProxyType
<<readonly>> MyAdmin : ConsumerAdmin
priority_filter : CNF::MappingFilter
lifetime_filter : CNF::MappingFilter
obtain_offered_types(mode : in ObtainInfoMode) : CN::EventTypeSeq
validate_event_qos(required_qos : in CN::QoSProperties, available_qos : out CN::NamedPropertyRangeSeq) : void
TypedPullSupplier
(from CosTypedNotify)
TypedProxyPullSupplier
connect_typed_pull_consumer(pull_consumer : in CEC::PullConsumer) : void
NotifySubscribe
(from CosNotifyComm)
subscription_change()
TypedPullSupplier
(from CosTypedEventComm)
get_typed_supplier() : Object
T0414720-00

Figure 7-27/X.770 – TypedProxyPull Consumer and Supplier interfaces

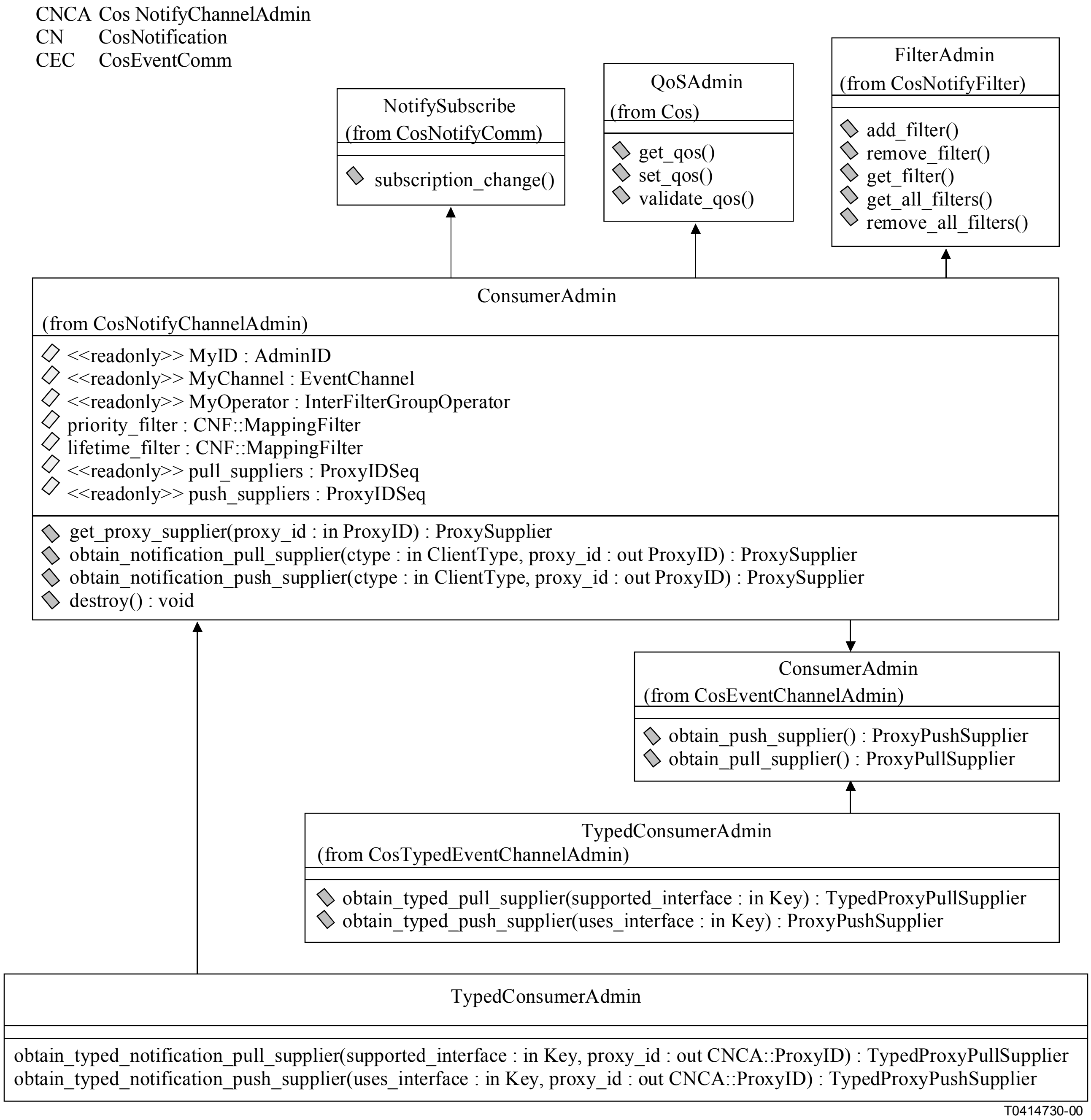
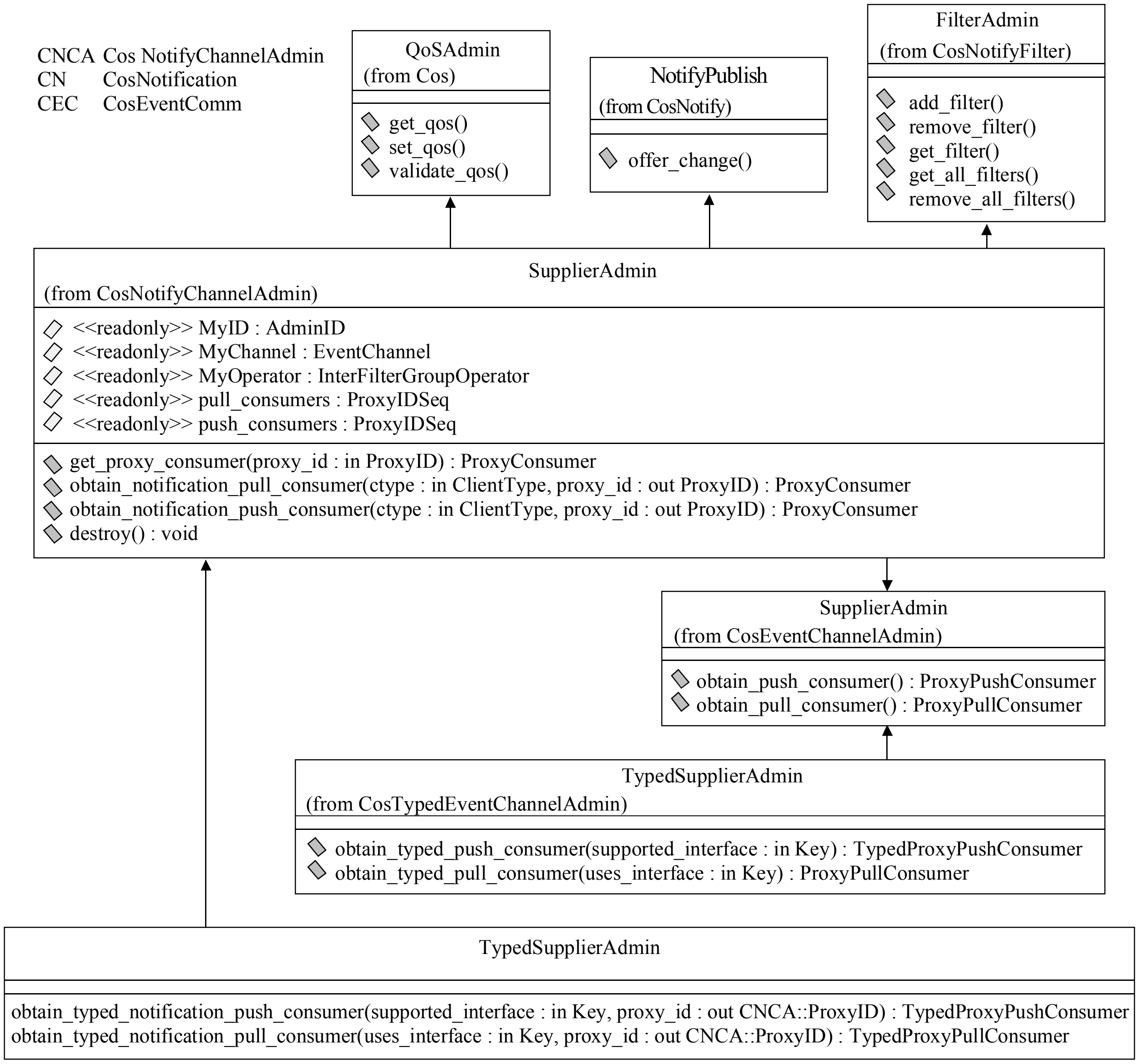

CNCA Cos NotifyChannelAdmin
CN CosNotification
CEC CosEventComm
NotifySubscribe
(from CosNotifyComm)
subscription_change()
QoSAdmin
(from Cos)
get_qos()
set_qos()
validate_qos()
FilterAdmin
(from CosNotifyFilter)
add_filter()
remove_filter()
get_filter()
get_all_filters()
remove_all_filters()
ConsumerAdmin
(from CosNotifyChannelAdmin)
<<readonly>> MyID : AdminID
<<readonly>> MyChannel : EventChannel
<<readonly>> MyOperator : InterFilterGroupOperator
priority_filter : CNF::MappingFilter
lifetime_filter : CNF::MappingFilter
<<readonly>> pull_suppliers : ProxyIDSeq
<<readonly>> push_suppliers : ProxyIDSeq
get_proxy_supplier(proxy_id : in ProxyID) : ProxySupplier
obtain_notification_pull_supplier(ctype : in ClientType, proxy_id : out ProxyID) : ProxySupplier
obtain_notification_push_supplier(ctype : in ClientType, proxy_id : out ProxyID) : ProxySupplier
destroy() : void
ConsumerAdmin
(from CosEventChannelAdmin)
obtain_push_supplier() : ProxyPushSupplier
obtain_pull_supplier() : ProxyPullSupplier
TypedConsumerAdmin
(from CosTypedEventChannelAdmin)
obtain_typed_pull_supplier(supported_interface : in Key) : TypedProxyPullSupplier
obtain_typed_push_supplier(uses_interface : in Key) : ProxyPushSupplier
TypedConsumerAdmin
obtain_typed_notification_pull_supplier(supported_interface : in Key, proxy_id : out CNCA::ProxyID) : TypedProxyPullSupplier
obtain_typed_notification_push_supplier(uses_interface : in Key, proxy_id : out CNCA::ProxyID) : TypedProxyPushSupplier
T0414730-00

Figure 7-28/X.770 – TypedConsumerAdmin interface


CNCA Cos NotifyChannelAdmin
CN CosNotification
CEC CosEventComm
QoSAdmin
(from Cos)
get_qos()
set_qos()
validate_qos()
NotifyPublish
(from CosNotify)
offer_change()
FilterAdmin
(from CosNotifyFilter)
add_filter()
remove_filter()
get_filter()
get_all_filters()
remove_all_filters()
SupplierAdmin
(from CosNotifyChannelAdmin)
<<readonly>> MyID : AdminID
<<readonly>> MyChannel : EventChannel
<<readonly>> MyOperator : InterFilterGroupOperator
<<readonly>> pull_consumers : ProxyIDSeq
<<readonly>> push_consumers : ProxyIDSeq
get_proxy_consumer(proxy_id : in ProxyID) : ProxyConsumer
obtain_notification_pull_consumer(ctype : in ClientType, proxy_id : out ProxyID) : ProxyConsumer
obtain_notification_push_consumer(ctype : in ClientType, proxy_id : out ProxyID) : ProxyConsumer
destroy() : void
SupplierAdmin
(from CosEventChannelAdmin)
obtain_push_consumer() : ProxyPushConsumer
obtain_pull_consumer() : ProxyPullConsumer
TypedSupplierAdmin
(from CosTypedEventChannelAdmin)
obtain_typed_push_consumer(supported_interface : in Key) : TypedProxyPushConsumer
obtain_typed_pull_consumer(uses_interface : in Key) : ProxyPullConsumer
TypedSupplierAdmin
obtain_typed_notification_push_consumer(supported_interface : in Key, proxy_id : out CNCA::ProxyID) : TypedProxyPushConsumer
obtain_typed_notification_pull_consumer(uses_interface : in Key, proxy_id : out CNCA::ProxyID) : TypedProxyPullConsumer

T0414740-00

Figure 7-29/X.770 – TypedSupplierAdmin interface

CNCA	Cos NotifyChannelAdmin
CN	CosNotification
CEC	CosEventComm

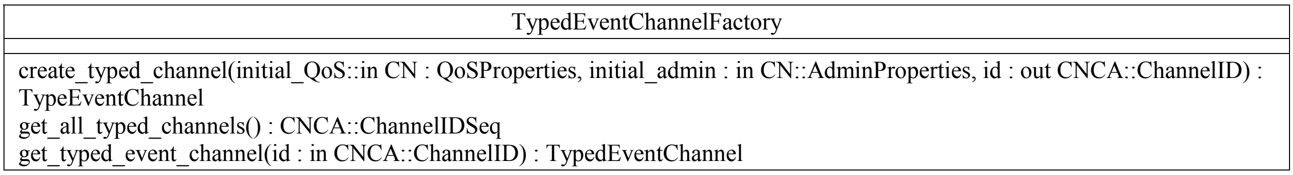

TypedEventChannelFactory
create_typed_channel(initial_QoS::in CN : QoSProperties, initial_admin : in CN::AdminProperties, id : out CNCA::ChannelID) : TypeEventChannel
get_all_typed_channels() : CNCA::ChannelIDSeq
get_typed_event_channel(id : in CNCA::ChannelID) : TypedEventChannel

Figure 7-30/X.770 – TypedEventChannel Factory interface

CNCA	Cos NotifyChannelAdmin
CN	CosNotification
CEC	CosEventComm
CNF	CosNotifyFilter

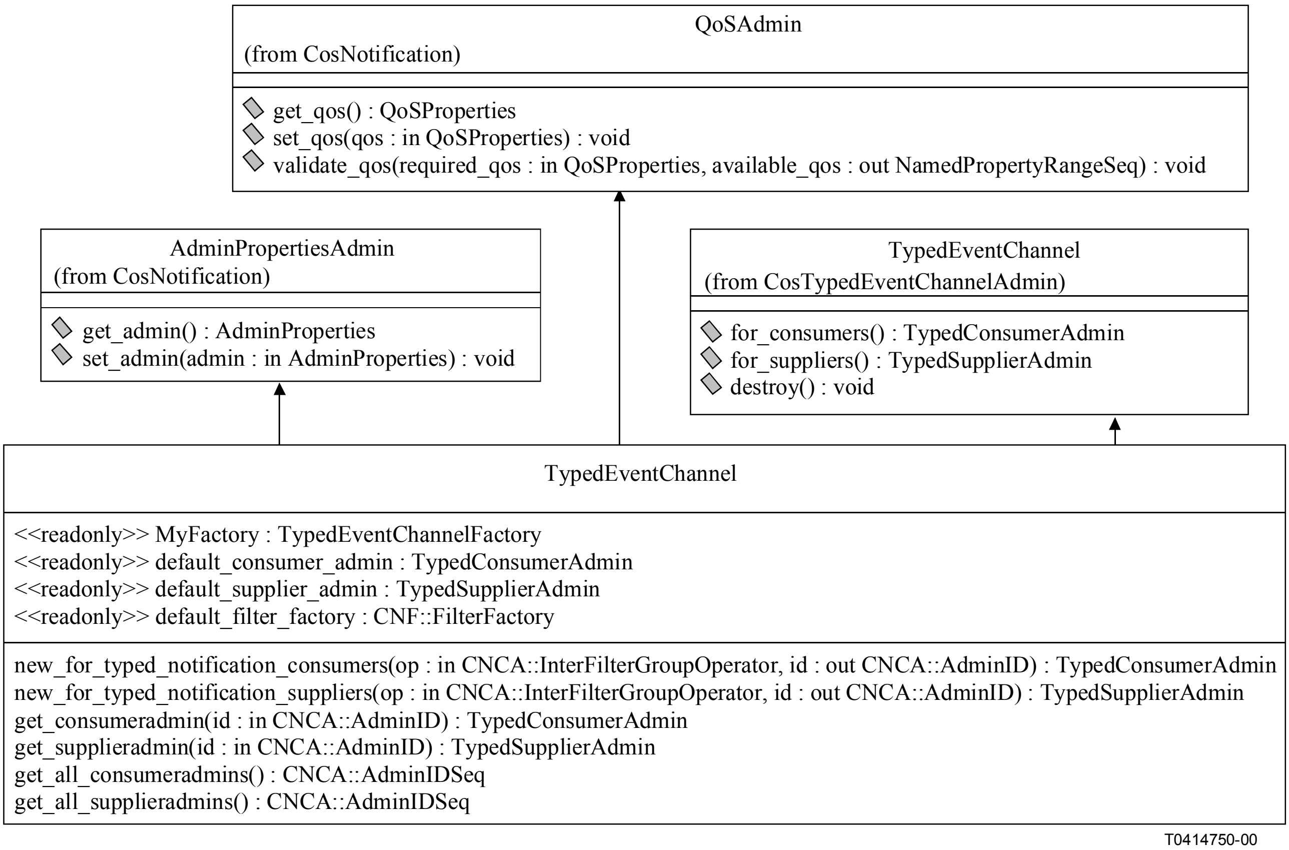

QoSAdmin
(from CosNotification)
get_qos() : QoSProperties
set_qos(qos : in QoSProperties) : void
validate_qos(required_qos : in QoSProperties, available_qos : out NamedPropertyRangeSeq) : void
AdminPropertiesAdmin
(from CosNotification)
get_admin() : AdminProperties
set_admin(admin : in AdminProperties) : void
TypedEventChannel
(from CosTypedEventChannelAdmin)
for_consumers() : TypedConsumerAdmin
for_suppliers() : TypedSupplierAdmin
destroy() : void
TypedEventChannel
<<readonly>> MyFactory : TypedEventChannelFactory
<<readonly>> default_consumer_admin : TypedConsumerAdmin
<<readonly>> default_supplier_admin : TypedSupplierAdmin
<<readonly>> default_filter_factory : CNF::FilterFactory
new_for_typed_notification_consumers(op : in CNCA::InterFilterGroupOperator, id : out CNCA::AdminID) : TypedConsumerAdmin
new_for_typed_notification_suppliers(op : in CNCA::InterFilterGroupOperator, id : out CNCA::AdminID) : TypedSupplierAdmin
get_consumeradmin(id : in CNCA::AdminID) : TypedConsumerAdmin
get_supplieradmin(id : in CNCA::AdminID) : TypedSupplierAdmin
get_all_consumeradmins() : CNCA::AdminIDSeq
get_all_supplieradmins() : CNCA::AdminIDSeq
T0414750-00

Figure 7-31/X.770 – TypedEventChannel interface

8 Engineering interface definitions

This Recommendation incorporates, by reference, all of the ODP IDL Interface definitions and associated operation description and behaviour text specified in the OMG Notification Service.

The interface definitions, supporting data type definitions, and conformance requirements for this Recommendation are fully specified in OMG Notification Service.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	**Data networks and open system communications**
Series Y	Global information infrastructure and Internet protocol aspects
Series Z	Languages and general software aspects for telecommunication systems

Geneva, 2001