XML Schema Documentation

Table of Contents

- Schema Document Properties
- Global Declarations
  - Element: **event**
  - Element: **strategy**
- Global Definitions
  - Attribute Group: **VersionAttributes.atts**
  - Complex Type: **Allocation**
  - Complex Type: **AllocationTradeIdentifier**
  - Complex Type: **Allocations**
  - Complex Type: **Amendment**
  - Complex Type: **Approval**
  - Complex Type: **Approvals**
  - Complex Type: **BestFitTrade**
  - Complex Type: **BlockTradeIdentifier**
  - Complex Type: **ChangeContract**
  - Complex Type: **ChangeContractSize**
  - Complex Type: **Collateral**
  - Complex Type: **Contract**
  - Complex Type: **ContractHeader**
  - Complex Type: **ContractId**
  - Complex Type: **ContractIdentifier**
  - Complex Type: **ContractInformation**
  - Complex Type: **ContractNovation**
  - Complex Type: **ContractReference**
  - Complex Type: **ContractTermination**
  - Complex Type: **CreditDerivativesNotices**
  - Complex Type: **DataDocument**
  - Complex Type: **Document**
  - Complex Type: **Event**
  - Complex Type: **EventId**
  - Complex Type: **ExecutionDateTime**
  - Complex Type: **FirstPeriodStartDate**
  - Complex Type: **Increase**
  - Complex Type: **IndependentAmount**
  - Complex Type: **LinkId**
  - Complex Type: **PartyPortfolioName**
  - Complex Type: **PartyRole**
  - Complex Type: **PartyTradeIdentifier**
  - Complex Type: **PartyTradeIdentifiers**
  - Complex Type: **PartyTradeInformation**
  - Complex Type: **PaymentDetail**
  - Complex Type: **PaymentRule**
  - Complex Type: **PercentageRule**
  - Complex Type: **Portfolio**
  - Complex Type: **PortfolioName**
  - Complex Type: **QueryParameter**
  - Complex Type: **QueryParameterId**
  - Complex Type: **QueryParameterOperator**
  - Complex Type: **QueryPortfolio**
  - Complex Type: **Strategy**
  - Complex Type: **Trade**
  - Complex Type: **TradeDifference**
  - Complex Type: **TradeHeader**
  - Complex Type: **TradeId**
  - Complex Type: **TradeIdentifier**
  - Complex Type: **TradeSide**

- Complex Type: **Trader**
- Complex Type: **Validation**
- Complex Type: **VersionedContractId**
- Complex Type: **VersionedTradeId**
- Model Group: **AccountReferenceOrPartyReference.model**
- Model Group: **AllocationContent.model**
- Model Group: **AmendmentDetails.model**
- Model Group: **CalculationAgent.model**
- Model Group: **ContractNovationDetails.model**
- Model Group: **ContractOrContractReference.model**
- Model Group: **IncreaseDetails.model**
- Model Group: **TradeOrTradeReference.model**
- Model Group: **Validation.model**
- Simple Type: **QueryParameterValue**

- Legend
- Glossary

top

Schema Document Properties

Target Namespace	**http://www.fpml.org/2007/FpML-4-4**
Version	$Revision: 2674 $
Element and Attribute Namespaces	• Global element and attribute declarations belong to this schema's target namespace. • By default, local element declarations belong to this schema's target namespace. • By default, local attribute declarations have no namespace.
Schema Composition	• This schema includes components from the following schema document(s): ○ fpml-shared-4-4.xsd

Declared Namespaces

Prefix	Namespace
Default namespace	**http://www.fpml.org/2007/FpML-4-4**
ecore	http://www.eclipse.org/emf/2002/Ecore
xml	http://www.w3.org/XML/1998/namespace
fpml-annotation	http://www.fpml.org/annotation
xsd	http://www.w3.org/2001/XMLSchema
fpml	**http://www.fpml.org/2007/FpML-4-4**

Schema Component Representation

```
<xsd:schema nsPrefix="fpml" package="org.fpml" documentRoot="FpML" targetNamespace="http://
www.fpml.org/2007/FpML-4-4" version="$Revision: 2674 $"
attributeFormDefault="unqualified" elementFormDefault="qualified">
   <xsd:include schemaLocation="fpml-shared-4-4.xsd"/>
   ...
</xsd:schema>
```

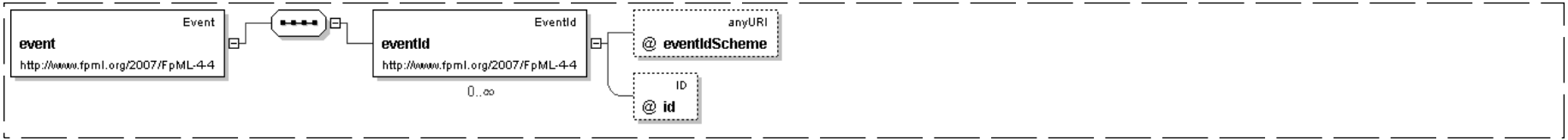
top

Global Declarations

Element: event

Name	event
Used by (from the same schema document)	Complex Type **DataDocument**
Type	Event
Nillable	no
Abstract	yes
Documentation	An abstract global element used as a basis for substition of event types

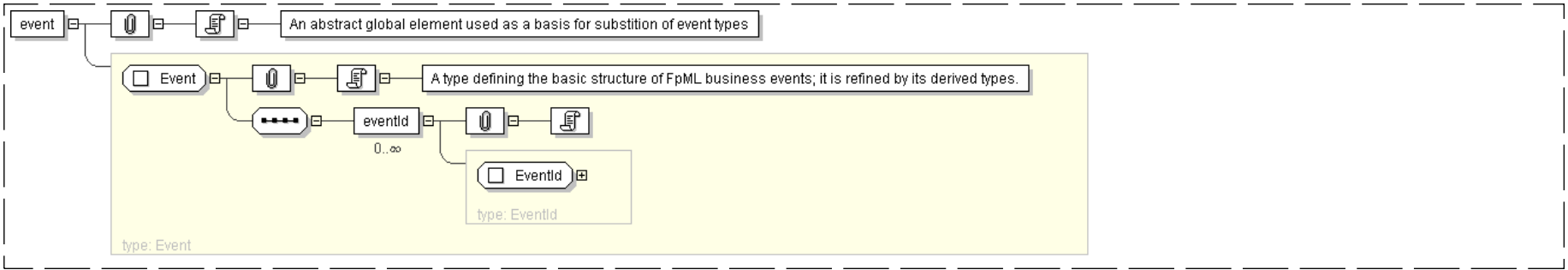
Logical Diagram



XML Instance Representation

```
<event>
   <eventId> EventId </eventId> [0..*]
   ''

</event>
```

Diagram



Schema Component Representation

```
<xsd:element name="event" type=" Event " abstract="true"/>
```

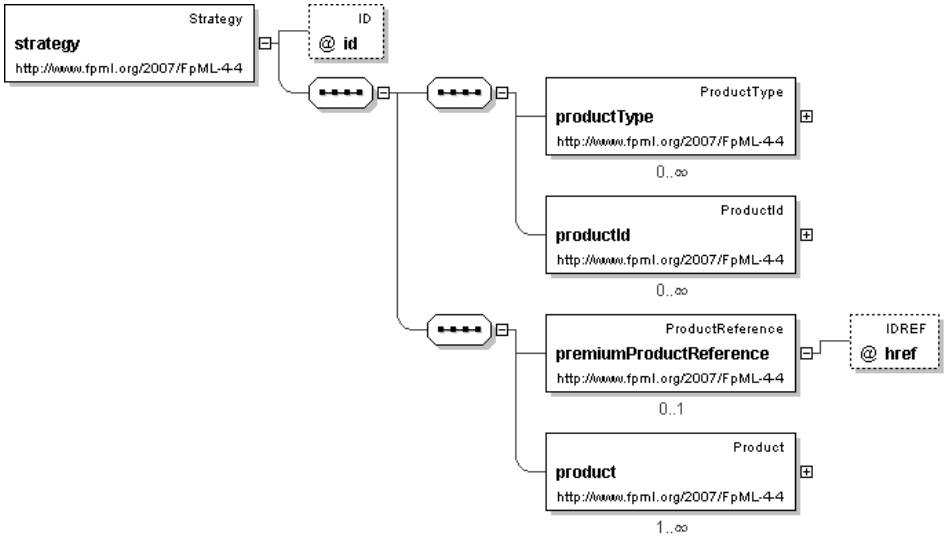
top

Element: strategy

- *This element can be used wherever the following element is referenced:*
  - product

Name	strategy
Type	Strategy
Nillable	no
Abstract	no
Documentation	A strategy product.

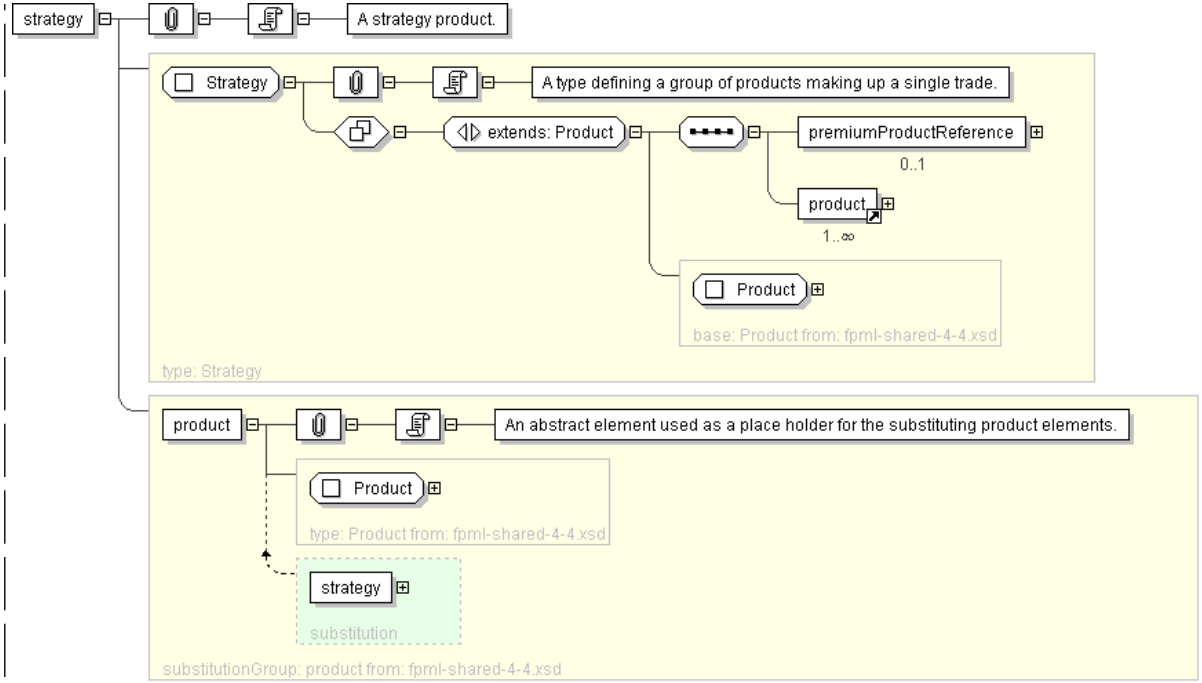
Logical Diagram



XML Instance Representation

```
<strategy
id=" xsd:ID [0..1]">
   <productType> ProductType </productType> [0..*]
   'A classification of the type of product. FpML defines a simple product categorization using
   a coding scheme.'

   <productId> ProductId </productId> [0..*]
   'A product reference identifier allocated by a party. FpML does not define the domain
   values associated with this element. Note that the domain values for this element are
   not strictly an enumerated list.'

   <premiumProductReference> ProductReference </premiumProductReference> [0..1]
   'Indicates which product within a strategy represents the premium payment.'

   <product> ... </product> [1..*]
</strategy>
```

Diagram



Schema Component Representation

```
<xsd:element name="strategy" type=" Strategy " substitutionGroup="product"/>
```

top

Global Definitions

Attribute Group: VersionAttributes.atts

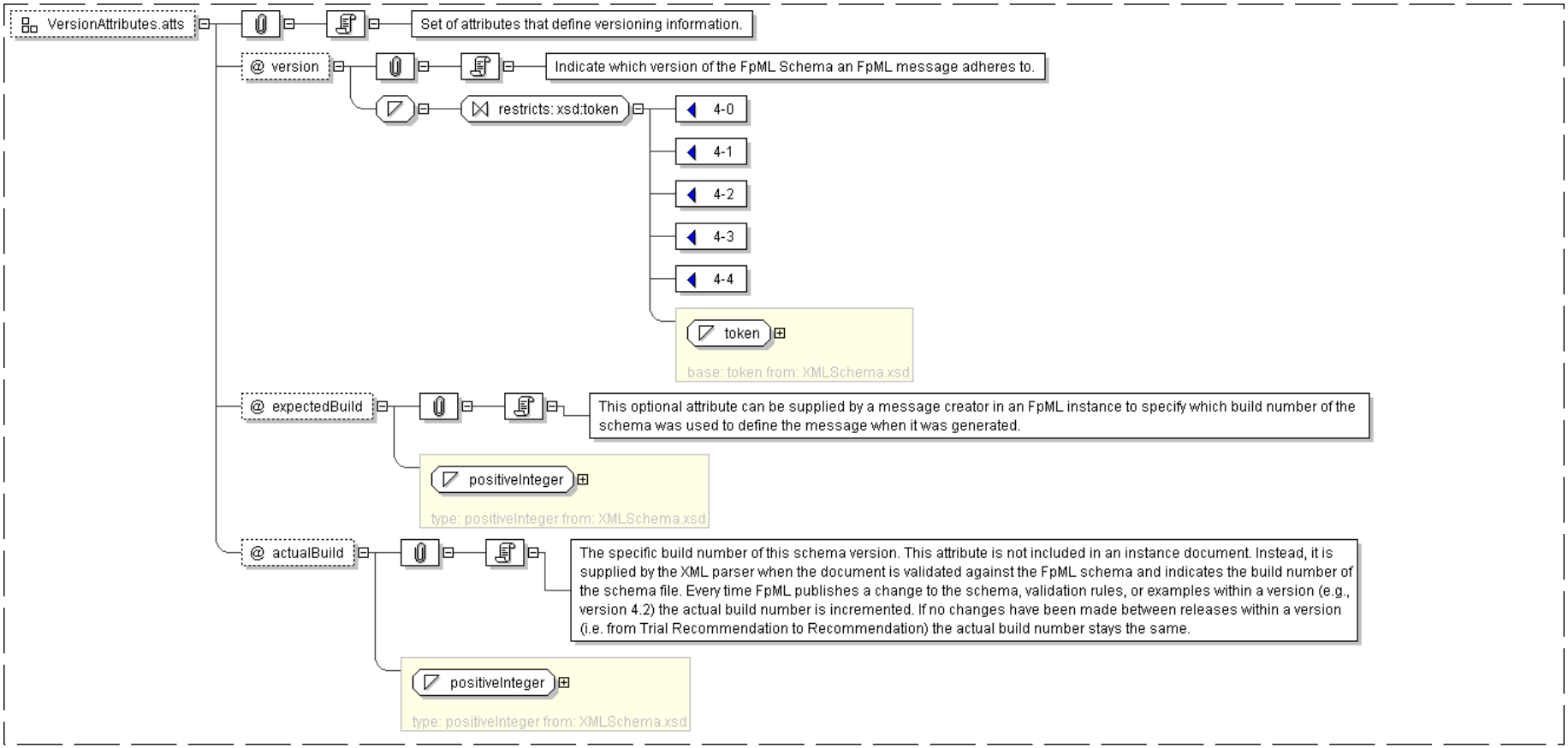
Name	VersionAttributes.atts
Used by (from the same schema document)	Complex Type **Document**
Documentation	Set of attributes that define versioning information.

XML Instance Representation

```
version=" xsd:token (value comes from list: {'4-0'|'4-1'|'4-2'|'4-3'|'4-4'}) [1]
 'Indicate which version of the FpML Schema an FpML message adheres to.'

"
expectedBuild=" xsd:positiveInteger [0..1]
 'This optional attribute can be supplied by a message creator in an FpML instance to
 specify which build number of the schema was used to define the message when it was generated.'

"
actualBuild="2 [0..1]
 'The specific build number of this schema version. This attribute is not included in
 an instance document. Instead, it is supplied by the XML parser when the document is
 validated against the FpML schema and indicates the build number of the schema file. Every
 time FpML publishes a change to the schema, validation rules, or examples within a version
 (e.g., version 4.2) the actual build number is incremented. If no changes have been
 made between releases within a version (i.e. from Trial Recommendation to Recommendation)
 the actual build number stays the same.'
```

"

Diagram



Schema Component Representation

```
<xsd:attributeGroup name="VersionAttributes.atts">
   <xsd:attribute name="version" use="required">
      <xsd:simpleType>
         <xsd:restriction base=" xsd:token ">
            <xsd:enumeration value="4-0"/>
            <xsd:enumeration value="4-1"/>
            <xsd:enumeration value="4-2"/>
            <xsd:enumeration value="4-3"/>
            <xsd:enumeration value="4-4"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:attribute>
   <xsd:attribute name="expectedBuild" type=" xsd:positiveInteger "/>
   <xsd:attribute name="actualBuild" type=" xsd:positiveInteger " fixed="2"/>
</xsd:attributeGroup>
```

top

Complex Type: Allocation

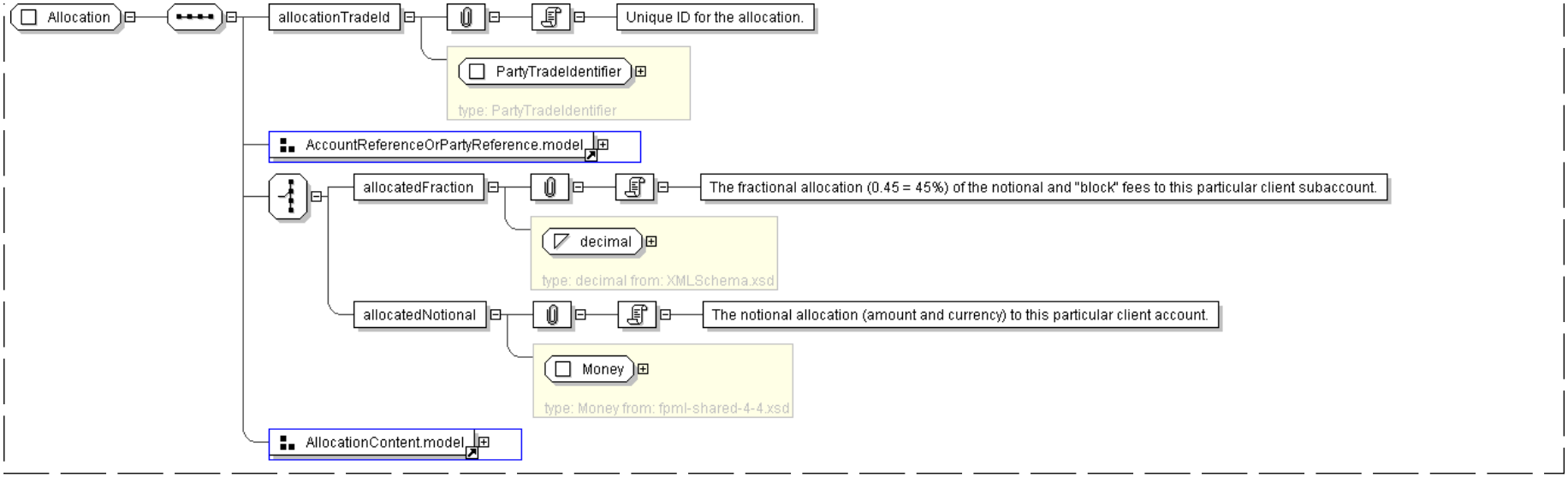
Super-types:	None
Sub-types:	None

Name	Allocation
Used by (from the same schema document)	Complex Type **Allocations**
Abstract	no

XML Instance Representation

```
<...>
   <allocationTradeId> PartyTradeIdentifier </allocationTradeId> [1]
   'Unique ID for the allocation.'

Start Choice [1]
      <accountReference> AccountReference </accountReference> [1]
      'Reference to the subaccount definition in the Party list.'

      <partyReference> PartyReference </partyReference> [1]
      'Reference to the party definition.'

End Choice
Start Choice [1]
      <allocatedFraction> xsd:decimal </allocatedFraction> [1]
      'The fractional allocation (0.45 = 45%) of the notional and \"block\" fees to this
      particular client subaccount.'

      <allocatedNotional> Money </allocatedNotional> [1]
      'The notional allocation (amount and currency) to this particular client account.'

End Choice
   <collateral> Collateral </collateral> [0..1]
   'The sum that must be posted upfront to collateralize against counterparty credit risk.'

   <creditChargeAmount> Money </creditChargeAmount> [0..1]
   'Special credit fee assessed to certain institutions.'

   <approvals> Approvals </approvals> [0..1]
   'A container for approval states in the workflow.'

   <masterConfirmationDate> xsd:date </masterConfirmationDate> [0..1]
   'The date of the confirmation executed between the parties and intended to govern the
   allocated trade between those parties.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Allocation">
   <xsd:sequence>
      <xsd:element name="allocationTradeId" type=" PartyTradeIdentifier "/>
      <xsd:group ref=" AccountReferenceOrPartyReference.model "/>
      <xsd:choice>
         <xsd:element name="allocatedFraction" type=" xsd:decimal "/>
         <xsd:element name="allocatedNotional" type=" Money "/>
      </xsd:choice>
      <xsd:group ref=" AllocationContent.model "/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: AllocationTradeIdentifier

Super-types:	TradeIdentifier < PartyTradeIdentifier (by extension) < **AllocationTradeIdentifier** (by extension)
Sub-types:	None

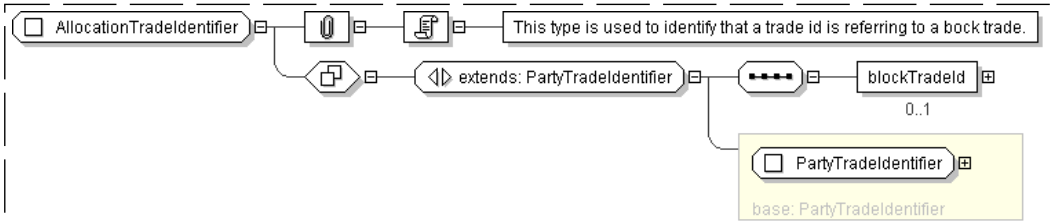
Name	AllocationTradeIdentifier
Abstract	no
Documentation	This type is used to identify that a trade id is referring to a bock trade.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]">
   <partyReference> PartyReference </partyReference> [1]
   'A pointer style reference to a party identifier defined elsewhere in the document. The
   party referenced has allocated the trade identifier.'

Start Choice [1..*]
      <tradeId> TradeId </tradeId> [1]
      <versionedTradeId> VersionedTradeId </versionedTradeId> [1]
End Choice
   <linkId> LinkId </linkId> [0..*]
   'A link identifier allowing the trade to be associated with other related trades, e.g.
   the linkId may contain a tradeId for an associated trade or several related trades may be
   given the same linkId. FpML does not define the domain values associated with this
```

```
element. Note that the domain values for this element are not strictly an enumerated list.'

<blockTradeId> PartyTradeIdentifier </blockTradeId> [0..1]
'The trade id of the block trade. This is used by each one of the allocated trades to
reference the block trade.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="AllocationTradeIdentifier">
   <xsd:complexContent>
      <xsd:extension base=" PartyTradeIdentifier ">
         <xsd:sequence>
            <xsd:element name="blockTradeId" type=" PartyTradeIdentifier " minOccurs="0"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: Allocations

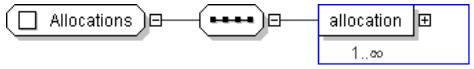
Super-types:	None
Sub-types:	None

Name	Allocations
Used by (from the same schema document)	Complex Type **Trade**
Abstract	no

XML Instance Representation

```
<...>
   <allocation> Allocation </allocation> [1..*]
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Allocations">
   <xsd:sequence>
      <xsd:element name="allocation" type=" Allocation " maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: Amendment

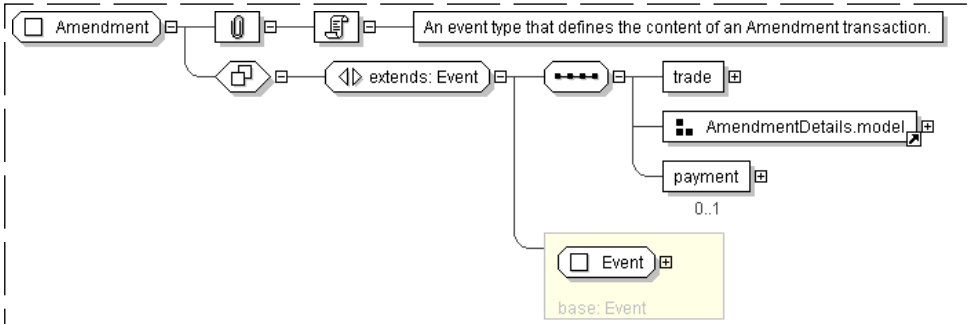
Super-types:	Event < **Amendment** (by extension)
Sub-types:	None

Name	Amendment
Abstract	no
Documentation	An event type that defines the content of an Amendment transaction.

XML Instance Representation

```
<...>
   <eventId> EventId </eventId> [0..*]
   ''

   <trade> Trade </trade> [1]
   <amendmentTradeDate> xsd:date </amendmentTradeDate> [1]
   'The date on which the the parties enter into the Amendment transaction'

   <amendmentEffectiveDate> xsd:date </amendmentEffectiveDate> [1]
   'The date on which the Amendment becomes effective'

   <payment> Payment </payment> [0..1]
   'A payment for the right to amend the trade.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Amendment">
   <xsd:complexContent>
      <xsd:extension base=" Event ">
         <xsd:sequence>
            <xsd:element name="trade" type=" Trade "/>
            <xsd:group ref=" AmendmentDetails.model "/>
            <xsd:element name="payment" type=" Payment " minOccurs="0"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: Approval

Super-types:	None

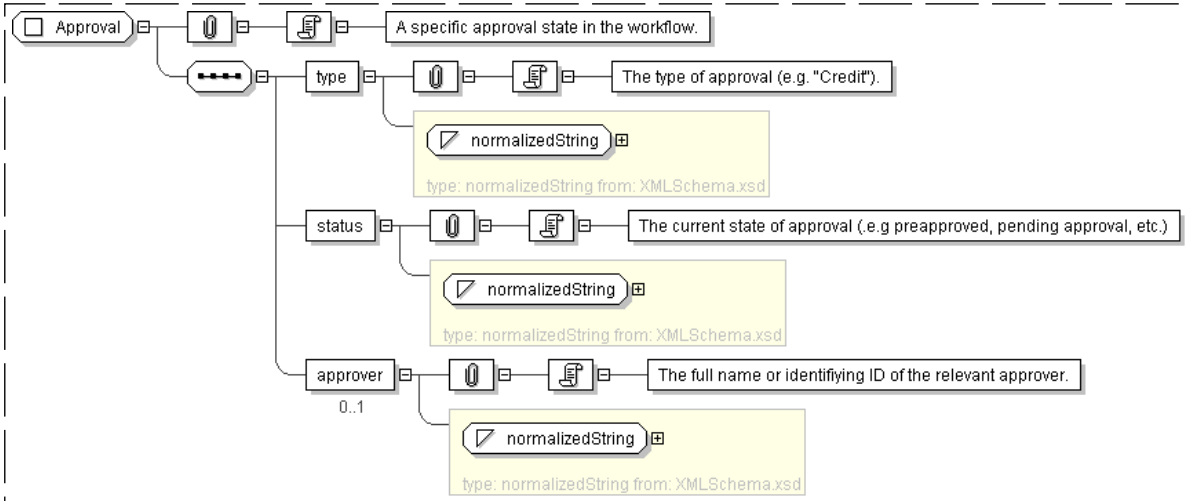
Sub-types:	None

Name	Approval
Used by (from the same schema document)	Complex Type **Approvals**
Abstract	no
Documentation	A specific approval state in the workflow.

XML Instance Representation

```
<...>
   <type> xsd:normalizedString </type> [1]
   'The type of approval (e.g. \"Credit\").'

   <status> xsd:normalizedString </status> [1]
   'The current state of approval (.e.g preapproved, pending approval, etc.)'

   <approver> xsd:normalizedString </approver> [0..1]
   'The full name or identifiying ID of the relevant approver.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Approval">
   <xsd:sequence>
      <xsd:element name="type" type=" xsd:normalizedString "/>
      <xsd:element name="status" type=" xsd:normalizedString "/>
      <xsd:element name="approver" type=" xsd:normalizedString " minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: Approvals

Super-types:	None
Sub-types:	None

Name	Approvals

Used by (from the same schema document)	Model Group **AllocationContent.model**
Abstract	no

XML Instance Representation

```
<...>
   <approval> Approval </approval> [1..*]
</...>
```

Diagram

Schema Component Representation

```
<xsd:complexType name="Approvals">
   <xsd:sequence>
      <xsd:element name="approval" type=" Approval " maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: BestFitTrade

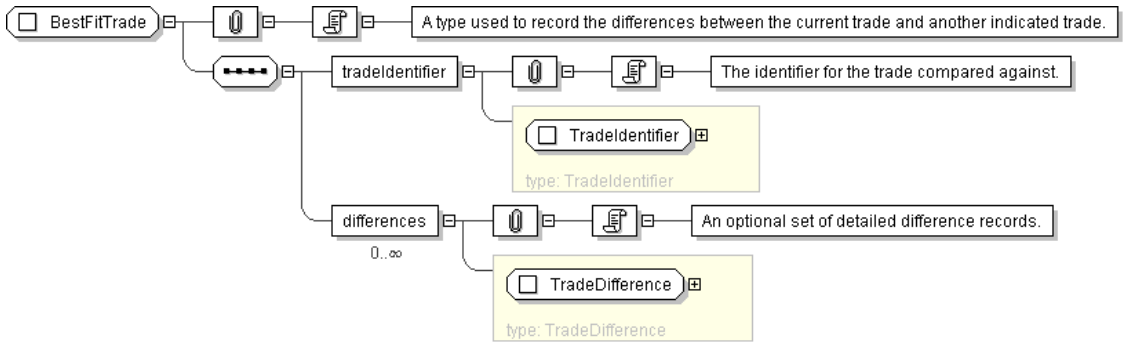
Super-types:	None
Sub-types:	None

Name	BestFitTrade
Abstract	no
Documentation	A type used to record the differences between the current trade and another indicated trade.

XML Instance Representation

```
<...>
   <tradeIdentifier> TradeIdentifier </tradeIdentifier> [1]
   'The identifier for the trade compared against.'

   <differences> TradeDifference </differences> [0..*]
   'An optional set of detailed difference records.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="BestFitTrade">
```

```
    <xsd:sequence>
       <xsd:element name="tradeIdentifier" type=" TradeIdentifier "/>
       <xsd:element name="differences" type=" TradeDifference " minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: BlockTradeIdentifier

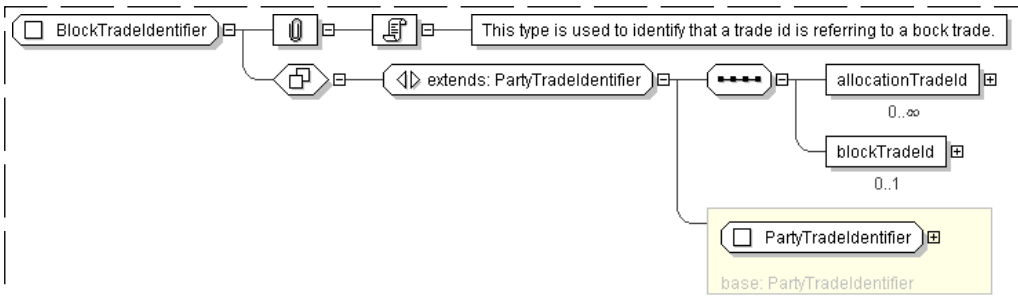
Super-types:	TradeIdentifier < PartyTradeIdentifier (by extension) < **BlockTradeIdentifier** (by extension)
Sub-types:	None

Name	BlockTradeIdentifier
Abstract	no
Documentation	This type is used to identify that a trade id is referring to a bock trade.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]">
   <partyReference> PartyReference </partyReference> [1]
   'A pointer style reference to a party identifier defined elsewhere in the document. The
   party referenced has allocated the trade identifier.'

Start Choice [1..*]
      <tradeId> TradeId </tradeId> [1]
      <versionedTradeId> VersionedTradeId </versionedTradeId> [1]
End Choice
   <linkId> LinkId </linkId> [0..*]
   'A link identifier allowing the trade to be associated with other related trades, e.g.
   the linkId may contain a tradeId for an associated trade or several related trades may be
   given the same linkId. FpML does not define the domain values associated with this
   element. Note that the domain values for this element are not strictly an enumerated list.'

   <allocationTradeId> PartyTradeIdentifier </allocationTradeId> [0..*]
   'The trade id of the allocated trade. This is used by the block trade to reference
   the allocated trade.'

   <blockTradeId> PartyTradeIdentifier </blockTradeId> [0..1]
   'The trade id of the parent trade for N-level allocations. This element is only used to model
   N-level allocations in which the trade acts as block and allocated trade at the same time.
   This basically means the ability to allocate a block trade to multiple allocation trades,
   and then allocate these in turn to other allocation trades (and so on if desired).'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="BlockTradeIdentifier">
   <xsd:complexContent>
      <xsd:extension base=" PartyTradeIdentifier ">
         <xsd:sequence>
            <xsd:element name="allocationTradeId" type=" PartyTradeIdentifier "
            minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="blockTradeId" type=" PartyTradeIdentifier " minOccurs="0"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: ChangeContract

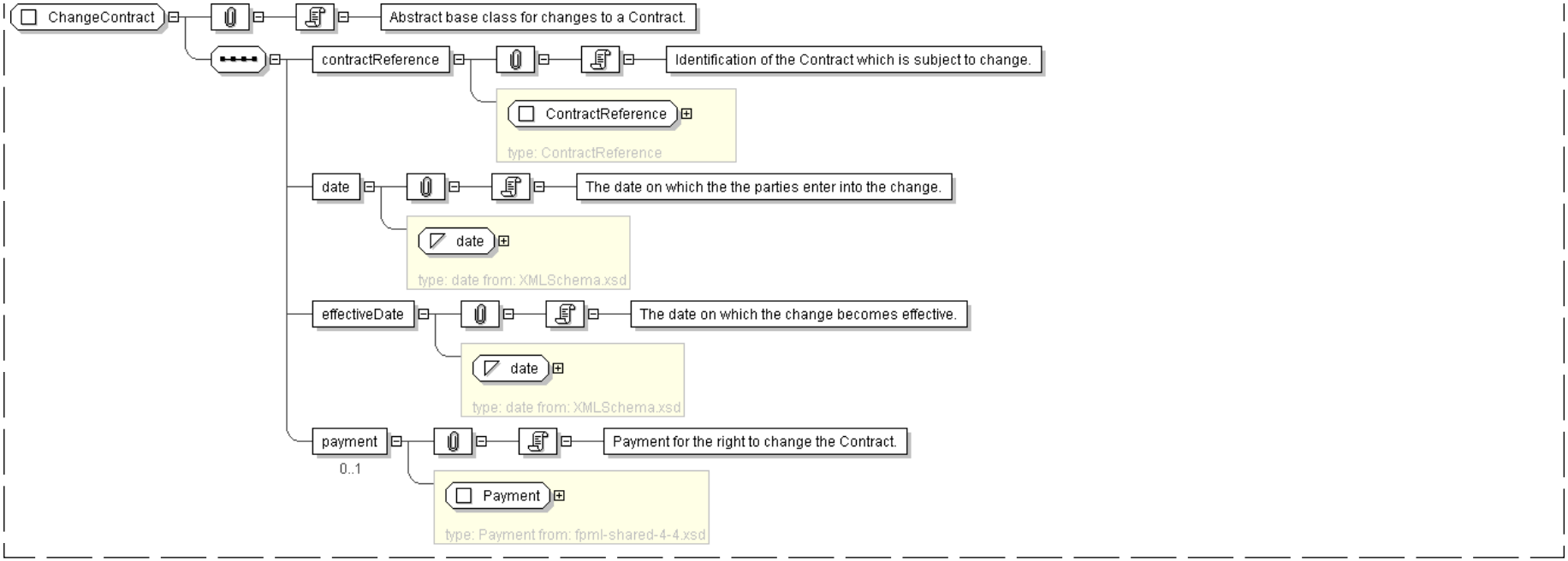
Super-types:	None
Sub-types:	• ChangeContractSize (by extension) • ContractTermination (by extension)

Name	ChangeContract
Abstract	yes
Documentation	Abstract base class for changes to a Contract.

XML Instance Representation

```
<...>
   <contractReference> ContractReference </contractReference> [1]
   'Identification of the Contract which is subject to change.'

   <date> xsd:date </date> [1]
   'The date on which the the parties enter into the change.'

   <effectiveDate> xsd:date </effectiveDate> [1]
   'The date on which the change becomes effective.'

   <payment> Payment </payment> [0..1]
   'Payment for the right to change the Contract.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="ChangeContract" abstract="true">
   <xsd:sequence>
      <xsd:element name="contractReference" type=" ContractReference "/>
      <xsd:element name="date" type=" xsd:date "/>
      <xsd:element name="effectiveDate" type=" xsd:date "/>
      <xsd:element name="payment" type=" Payment " minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: ChangeContractSize

Super-types:	ChangeContract < **ChangeContractSize** (by extension)
Sub-types:	None

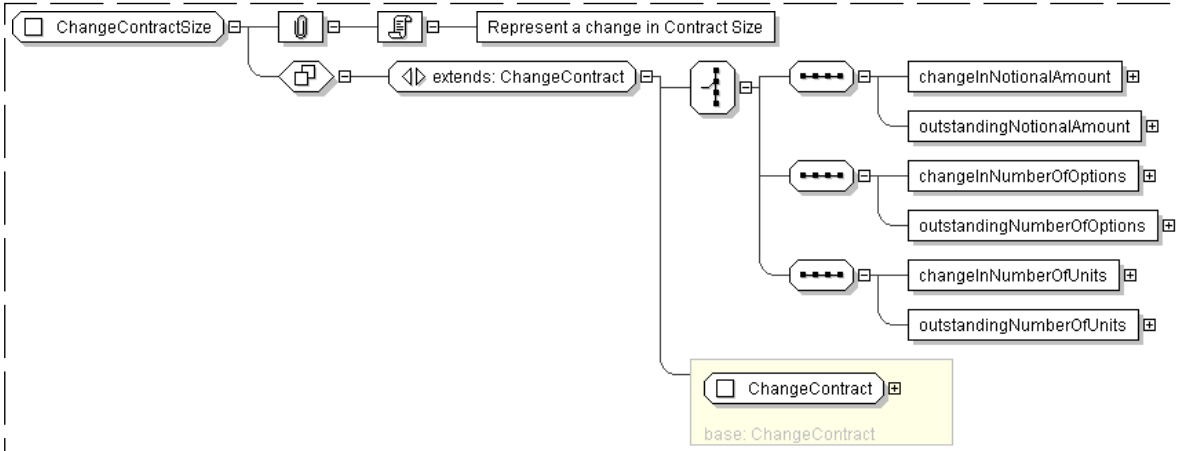
Name	ChangeContractSize
Abstract	no
Documentation	Represent a change in Contract Size

XML Instance Representation

```
<...>
   <contractReference> ContractReference </contractReference> [1]
   'Identification of the Contract which is subject to change.'

   <date> xsd:date </date> [1]
   'The date on which the the parties enter into the change.'

   <effectiveDate> xsd:date </effectiveDate> [1]
   'The date on which the change becomes effective.'

   <payment> Payment </payment> [0..1]
```

```
    'Payment for the right to change the Contract.'

Start Choice [1]
      <changeInNotionalAmount> Money </changeInNotionalAmount> [1]
      'Specifies the fixed amount by which the Notional Amount changes'

      <outstandingNotionalAmount> Money </outstandingNotionalAmount> [1]
      'Specifies the Notional amount after the Change'

      <changeInNumberOfOptions> xsd:decimal </changeInNumberOfOptions> [1]
      'Specifies the fixed amount by which the Number of Options changes'

      <outstandingNumberOfOptions> xsd:decimal </outstandingNumberOfOptions> [1]
      'Specifies the Number of Options after the Change.'

      <changeInNumberOfUnits> xsd:decimal </changeInNumberOfUnits> [1]
      'Specifies the fixed amount by which the Number of Units changes'

      <outstandingNumberOfUnits> xsd:decimal </outstandingNumberOfUnits> [1]
      'Specifies the Number of Units'

End Choice
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="ChangeContractSize">
   <xsd:complexContent>
      <xsd:extension base=" ChangeContract ">
         <xsd:choice>
            <xsd:sequence>
               <xsd:element name="changeInNotionalAmount" type=" Money "/>
               <xsd:element name="outstandingNotionalAmount" type=" Money "/>
            </xsd:sequence>
            <xsd:sequence>
               <xsd:element name="changeInNumberOfOptions" type=" xsd:decimal "/>
               <xsd:element name="outstandingNumberOfOptions" type=" xsd:decimal "/>
            </xsd:sequence>
            <xsd:sequence>
               <xsd:element name="changeInNumberOfUnits" type=" xsd:decimal "/>
               <xsd:element name="outstandingNumberOfUnits" type=" xsd:decimal "/>
            </xsd:sequence>
         </xsd:choice>
      </xsd:extension>
```

```
   </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: Collateral

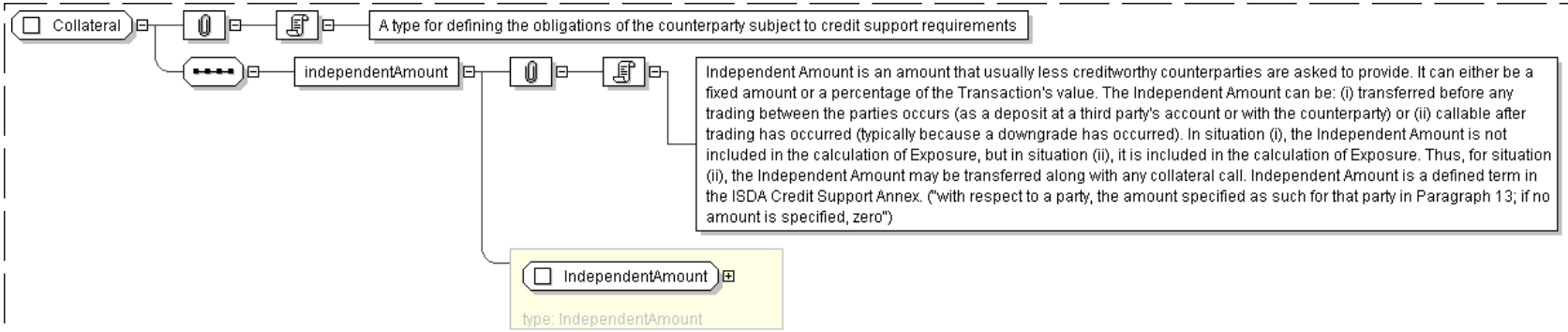
Super-types:	None
Sub-types:	None

Name	Collateral
Used by (from the same schema document)	Complex Type **Contract** , Complex Type **Trade** , Model Group **AllocationContent.model**
Abstract	no
Documentation	A type for defining the obligations of the counterparty subject to credit support requirements

XML Instance Representation

```
<...>
   <independentAmount> IndependentAmount </independentAmount> [1]
   'Independent Amount is an amount that usually less creditworthy counterparties are asked
   to provide. It can either be a fixed amount or a percentage of the Transaction\'s value.
   The Independent Amount can be: (i) transferred before any trading between the parties
   occurs (as a deposit at a third party\'s account or with the counterparty) or (ii)
   callable after trading has occurred (typically because a downgrade has occurred). In
   situation (i), the Independent Amount is not included in the calculation of Exposure, but
   in situation (ii), it is included in the calculation of Exposure. Thus, for situation (ii),
   the Independent Amount may be transferred along with any collateral call. Independent Amount
   is a defined term in the ISDA Credit Support Annex. (\"with respect to a party, the
   amount specified as such for that party in Paragraph 13; if no amount is specified, zero\")'
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Collateral">
   <xsd:sequence>
      <xsd:element name="independentAmount" type=" IndependentAmount "/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: Contract

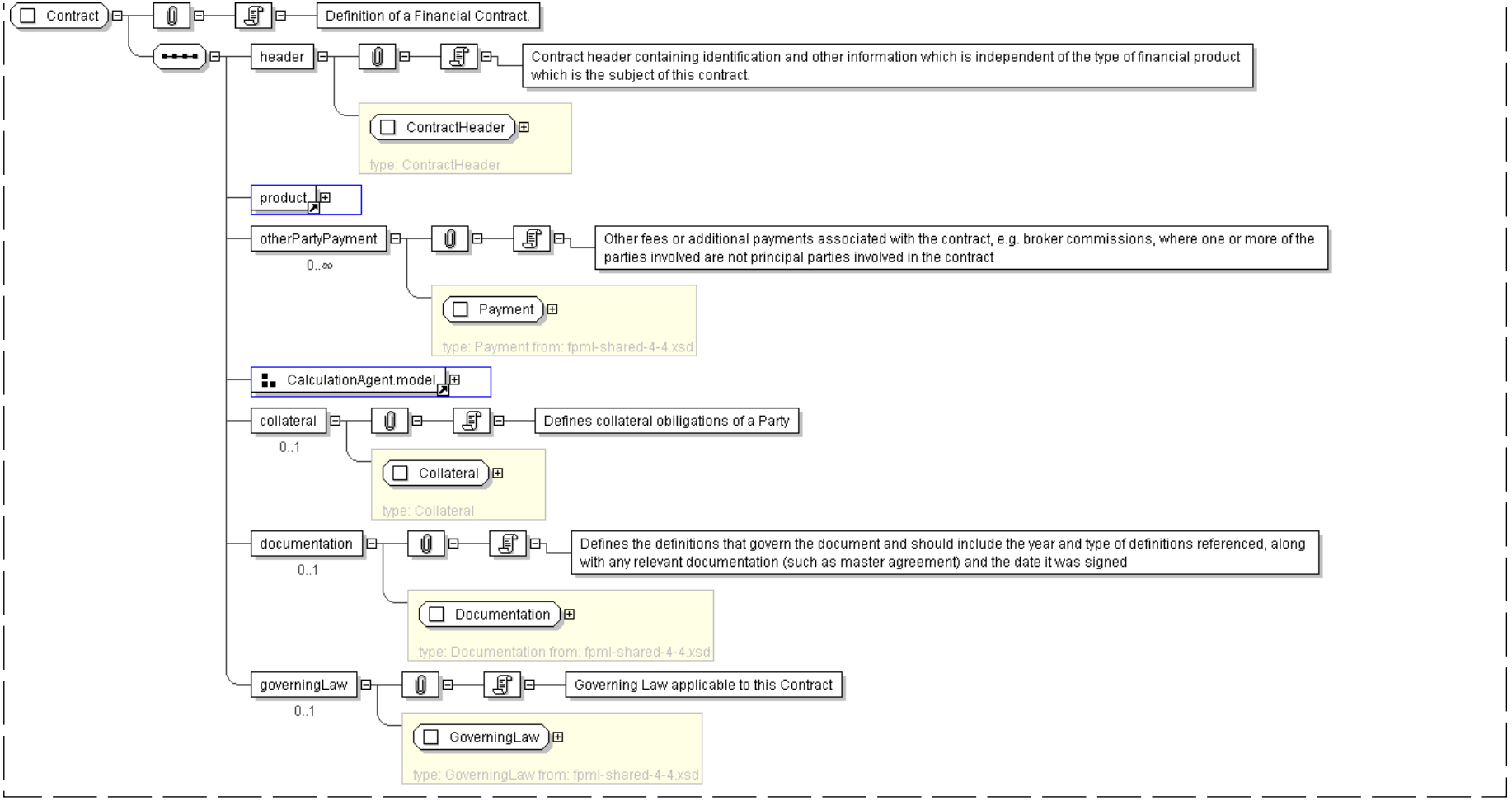
Super-types:	None
Sub-types:	None

Name	Contract
Used by (from the same schema document)	Model Group **ContractNovationDetails.model** , Model Group **ContractNovationDetails.model** , Model Group **ContractNovationDetails.model** , Model Group **ContractOrContractReference.model**
Abstract	no
Documentation	Definition of a Financial Contract.

XML Instance Representation

```
<...>
   <header> ContractHeader </header> [1]
   'Contract header containing identification and other information which is independent of
   the type of financial product which is the subject of this contract.'

   <product> ... </product> [1]
   <otherPartyPayment> Payment </otherPartyPayment> [0..*]
   'Other fees or additional payments associated with the contract, e.g. broker commissions,
   where one or more of the parties involved are not principal parties involved in the contract'

   <calculationAgent> CalculationAgent </calculationAgent> [0..1]
   'The ISDA Calculation Agent responsible for performing duties associated with an optional
   early termination'

   <calculationAgentBusinessCenter> BusinessCenter </calculationAgentBusinessCenter> [0..1]
   'The city in which the office through which ISDA Calculation Agent is acting for purposes
   of the transaction is located The short-form confirm for a trade that is executed under
   a Sovereign or Asia Pacific Master Confirmation Agreement ( MCA ), does not need to specify
   the Calculation Agent. However, the confirm does need to specify the Calculation Agent
   City. This is due to the fact that the MCA sets the value for Calculation Agent but does
   not set the value for Calculation Agent City.'

   <collateral> Collateral </collateral> [0..1]
   'Defines collateral obiligations of a Party'

   <documentation> Documentation </documentation> [0..1]
   'Defines the definitions that govern the document and should include the year and type
   of definitions referenced, along with any relevant documentation (such as master agreement)
   and the date it was signed'

   <governingLaw> GoverningLaw </governingLaw> [0..1]
   'Governing Law applicable to this Contract'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Contract">
   <xsd:sequence>
      <xsd:element name="header" type=" ContractHeader "/>
      <xsd:element ref=" product "/>
      <xsd:element name="otherPartyPayment" type=" Payment " minOccurs="0" maxOccurs="unbounded"/>
      <xsd:group ref=" CalculationAgent.model "/>
      <xsd:element name="collateral" type=" Collateral " minOccurs="0"/>
      <xsd:element name="documentation" type=" Documentation " minOccurs="0"/>
      <xsd:element name="governingLaw" type=" GoverningLaw " minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: ContractHeader

Super-types:	None
Sub-types:	None

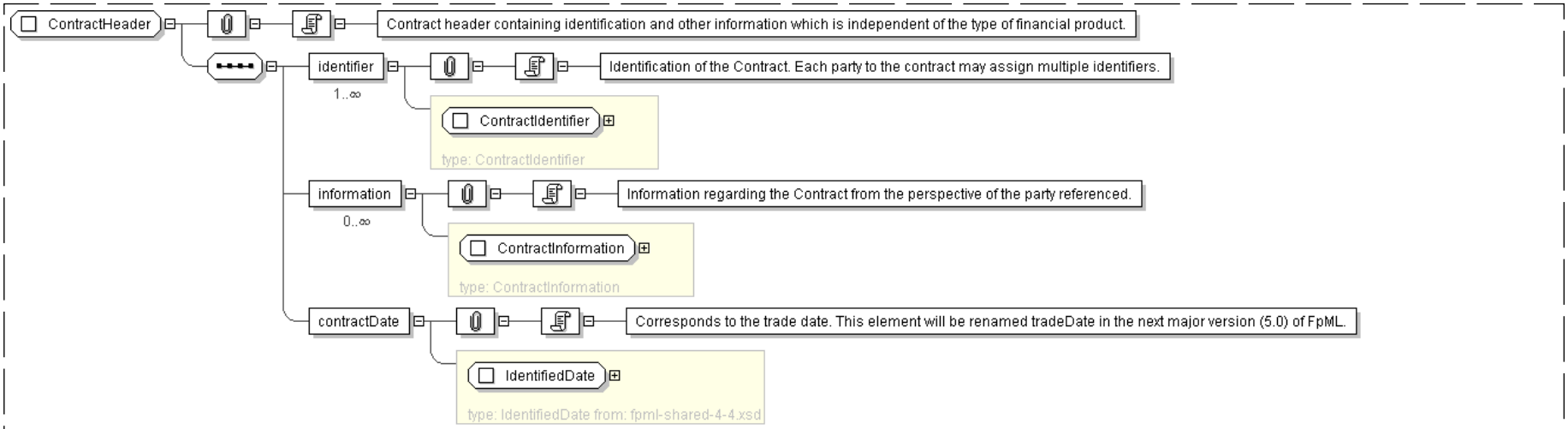
Name	ContractHeader

Used by (from the same schema document)	Complex Type **Contract**
Abstract	no
Documentation	Contract header containing identification and other information which is independent of the type of financial product.

XML Instance Representation

```
<...>
   <identifier> ContractIdentifier </identifier> [1..*]
   'Identification of the Contract. Each party to the contract may assign multiple identifiers.'

   <information> ContractInformation </information> [0..*]
   'Information regarding the Contract from the perspective of the party referenced.'

   <contractDate> IdentifiedDate </contractDate> [1]
   'Corresponds to the trade date. This element will be renamed tradeDate in the next
   major version (5.0) of FpML.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="ContractHeader">
   <xsd:sequence>
      <xsd:element name="identifier" type=" ContractIdentifier " maxOccurs="unbounded"/>
      <xsd:element name="information" type=" ContractInformation "
      minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="contractDate" type=" IdentifiedDate "/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: ContractId

Super-types:	xsd:normalizedString < **ContractId** (by extension)
Sub-types:	None

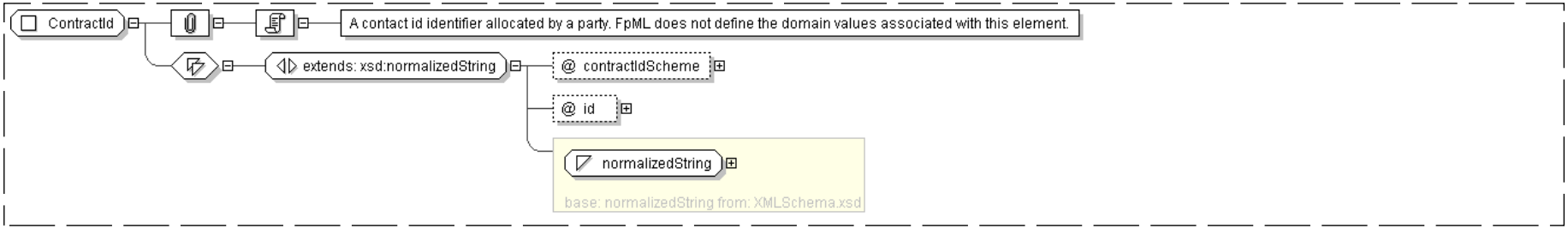
Name	ContractId
Used by (from the same schema document)	Complex Type **ContractIdentifier** , Complex Type **VersionedContractId**

Abstract	no
Documentation	A contact id identifier allocated by a party. FpML does not define the domain values associated with this element.

XML Instance Representation

```
<...
contractIdScheme=" xsd:anyURI [1]"
id=" xsd:ID [0..1]">
xsd:normalizedString
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="ContractId">
   <xsd:simpleContent>
      <xsd:extension base=" xsd:normalizedString ">
         <xsd:attribute name="contractIdScheme" type=" xsd:anyURI " use="required"/>
         <xsd:attribute name="id" type=" xsd:ID "/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: ContractIdentifier

Super-types:	None
Sub-types:	None

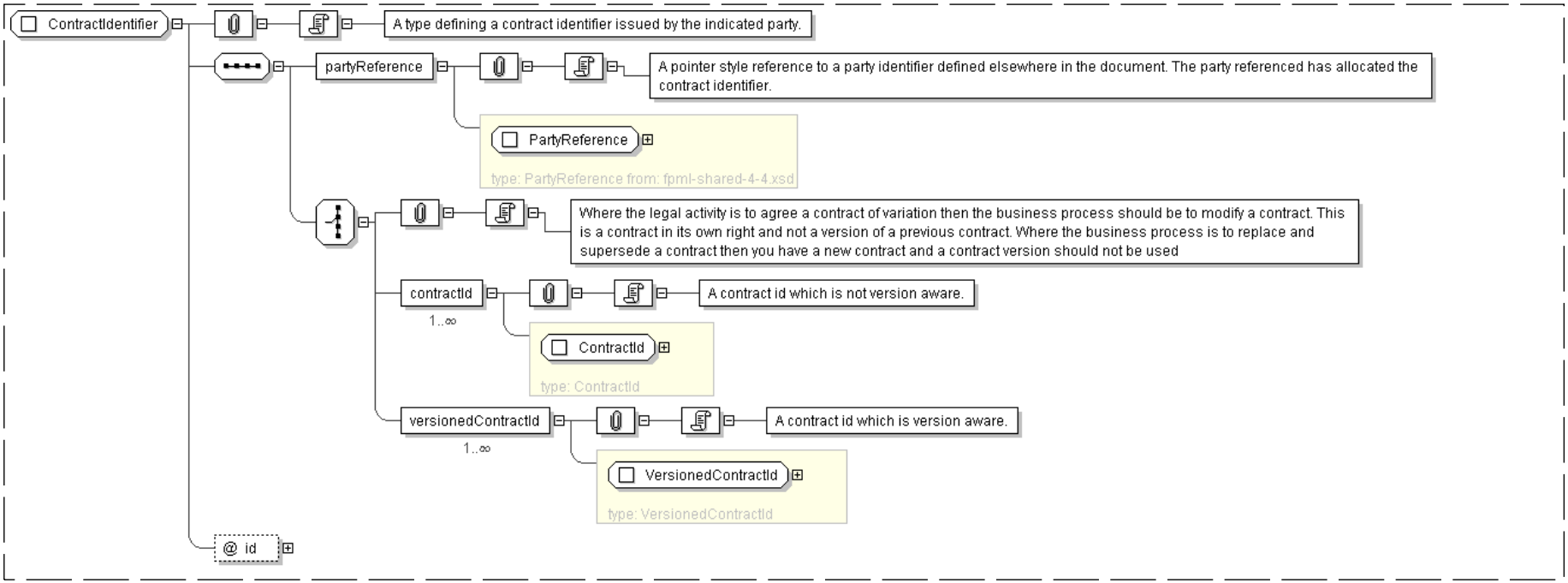
Name	ContractIdentifier
Used by (from the same schema document)	Complex Type **ContractHeader** , Complex Type **ContractReference**
Abstract	no
Documentation	A type defining a contract identifier issued by the indicated party.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]">
   <partyReference> PartyReference </partyReference> [1]
   'A pointer style reference to a party identifier defined elsewhere in the document. The
   party referenced has allocated the contract identifier.'

Start Choice [1]
'Where the legal activity is to agree a contract of variation then the business process
should be to modify a contract. This is a contract in its own right and not a version of
a previous contract. Where the business process is to replace and supersede a contract then
you have a new contract and a contract version should not be used'

      <contractId> ContractId </contractId> [1..*]
      'A contract id which is not version aware.'
```

```
      <versionedContractId> VersionedContractId </versionedContractId> [1..*]
      'A contract id which is version aware.'

End Choice
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="ContractIdentifier">
   <xsd:sequence>
      <xsd:element name="partyReference" type=" PartyReference "/>
      <xsd:choice>
         <xsd:element name="contractId" type=" ContractId " maxOccurs="unbounded"/>
         <xsd:element name="versionedContractId" type=" VersionedContractId " maxOccurs="unbounded"/>
      </xsd:choice>
   </xsd:sequence>
   <xsd:attribute name="id" type=" xsd:ID "/>
</xsd:complexType>
```

top

Complex Type: ContractInformation

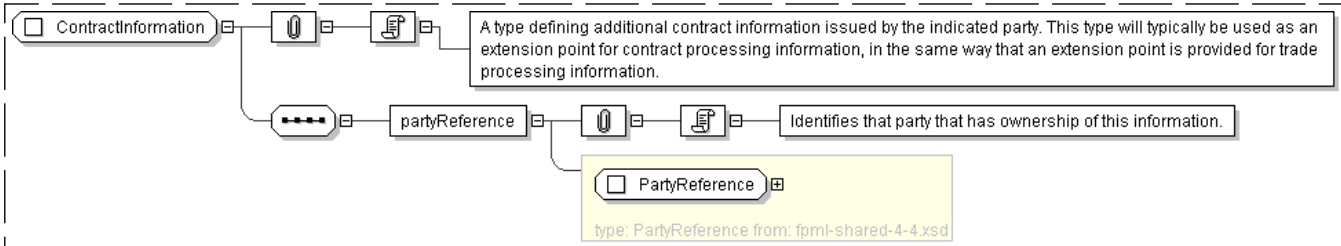
Super-types:	None
Sub-types:	None

Name	ContractInformation
Used by (from the same schema document)	Complex Type **ContractHeader**
Abstract	no
Documentation	A type defining additional contract information issued by the indicated party. This type will typically be used as an extension point for contract processing information, in the same way that an extension point is provided for trade processing information.

XML Instance Representation

```
<...>
   <partyReference> PartyReference </partyReference> [1]
   'Identifies that party that has ownership of this information.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="ContractInformation">
   <xsd:sequence>
      <xsd:element name="partyReference" type=" PartyReference "/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: ContractNovation

Super-types:	None
Sub-types:	None

Name	ContractNovation
Abstract	no
Documentation	Details of the Contract Novation

XML Instance Representation

```
<...>
Start Choice [1]
Start Choice [1]
'Choice between identification and representation of the new contract.'


         <newContractReference> ContractReference </newContractReference> [1]
         'Indicates a reference to the new Contract between the transferee and the remaining party.'

         <newContract> Contract </newContract> [1]
         'Indicates the new Contract between the transferee and the remaining party.'

End Choice
Start Choice [1]
         <oldContractReference> ContractReference </oldContractReference> [1]
         'Indicates a reference to the original contract between the transferor and the remaining party.'

         <oldContract> Contract </oldContract> [1]
         'Indicates the original Contract between the transferor and the remaining party.'

End Choice
Start Choice [0..1]
'Choice between identification and representation of the new contract.'
```

<newContractReference> ContractReference </newContractReference> [1]
<newContract> Contract </newContract> [1]
End Choice
End Choice
<transferor> PartyReference </transferor> [1]

'A pointer style reference to a party identifier defined elsewhere in the document. In a three-way novation the party referenced is the Transferor (outgoing party) in the novation. The Transferor means a party which transfers by novation to a Transferee all of its rights, liabilities, duties and obligations with respect to a Remaining Party. In a four-way novation the party referenced is Transferor 1 which transfers by novation to Transferee 1 all of its rights, liabilities, duties and obligations with respect to Transferor 2. ISDA 2004 Novation Term: Transferor (three-way novation) or Transferor 1 (four-way novation).'

<transferee> PartyReference </transferee> [1]

'A pointer style reference to a party identifier defined elsewhere in the document. In a three-way novation the party referenced is the Transferee (incoming party) in the novation. Transferee means a party which accepts by way of novation all rights, liabilities, duties and obligations of a Transferor with respect to a Remaining Party. In a four-way novation the party referenced is Transferee 1 which accepts by way of novation the rights, liabilities, duties and obligations of Transferor 1. ISDA 2004 Novation Term: Transferee (three-way novation) or Transferee 1 (four-way novation).'

<remainingParty> PartyReference </remainingParty> [1]

'A pointer style reference to a party identifier defined elsewhere in the document. In a three-way novation the party referenced is the Remaining Party in the novation. Remaining Party means a party which consents to a Transferor\'s transfer by novation and the acceptance thereof by the Transferee of all of the Transferor\'s rights, liabilities, duties and obligations with respect to such Remaining Party under and with respect of the Novated Amount of a transaction. In a four-way novation the party referenced is Transferor 2 per the ISDA definition and acts in the role of a Transferor. Transferor 2 transfers by novation to Transferee 2 all of its rights, liabilities, duties and obligations with respect to Transferor 1. ISDA 2004 Novation Term: Remaining Party (three-way novation) or Transferor 2 (four-way novation).'

<otherRemainingParty> PartyReference </otherRemainingParty> [0..1]

'A pointer style reference to a party identifier defined elsewhere in the document. This element is not applicable in a three-way novation and should be omitted. In a four-way novation the party referenced is Transferee 2. Transferee 2 means a party which accepts by way of novation the rights, liabilities, duties and obligations of Transferor 2. ISDA 2004 Novation Term: Transferee 2 (four-way novation).'

<novationDate> xsd:date </novationDate> [1]

'Specifies the date that one party\'s legal obligations with regard to a trade are transferred to another party. It corresponds to the Novation Date section of the 2004 ISDA Novation Definitions, section 1.16.'

<novationContractDate> xsd:date </novationContractDate> [0..1]

'Specifies the date the parties agree to assign or novate a Contract. If this element is not specified, the novationContractDate will be deemed to be the novationDate. It corresponds to the Novation Trade Date section of the 2004 ISDA Novation Definitions, section 1.17.'

Start Choice [1]

'Choice for expressing the novated amount as either a money amount, number of options, or number of units, according the the financial product which is being novated.'

<novatedAmount> Money </novatedAmount> [1]

'The amount which represents the portion of the Old Contract being novated.'

<novatedNumberOfOptions> xsd:decimal </novatedNumberOfOptions> [1]

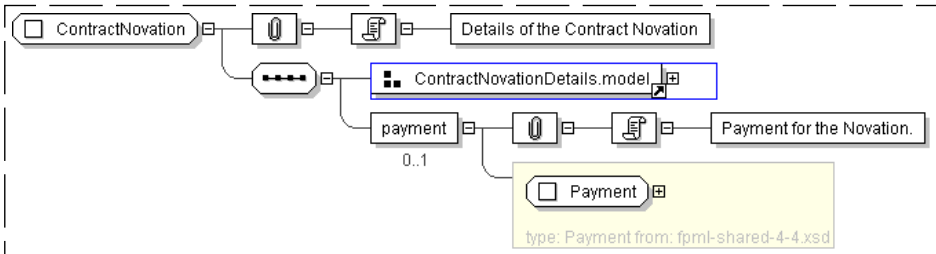
'The number of options which represent the portion of the Old Contract being novated.'

<novatedNumberOfUnits> xsd:decimal </novatedNumberOfUnits> [1]

'The number of options which represent the portion of the Old Contract being novated.'

```
'End Choice
    <fullFirstCalculationPeriod> xsd:boolean </fullFirstCalculationPeriod> [0..1]
    'This element corresponds to the applicability of the Full First Calculation Period as
    defined in the 2004 ISDA Novation Definitions, section 1.20.'

    <firstPeriodStartDate> FirstPeriodStartDate </firstPeriodStartDate> [0..2]
    'Element that is used to be able to make sense of the "new transaction" without
    requiring reference back to the "old transaction". In the case of interest rate products
    there are potentially 2 "first period start dates" to reference – one with respect to
    each party to the new transaction. For Credit Default Swaps there is just the one with
    respect to the party that is the fixed rate payer.'

    <nonReliance> Empty </nonReliance> [0..1]
    'This element corresponds to the non-Reliance section in the 2004 ISDA Novation
    Definitions, section 2.1 (c) (i). The element appears in the instance document when
    non-Reliance is applicable.'

    <creditDerivativesNotices> CreditDerivativesNotices </creditDerivativesNotices> [0..1]
    'This element should be specified if one or more of either a Credit Event Notice, Notice
    of Publicly Available Information, Notice of Physical Settlement or Notice of Intended
    Physical Settlement, as applicable, has been delivered by or to the Transferor or the
    Remaining Party. The type of notice or notices that have been delivered should be indicated
    by setting the relevant boolean element value(s) to true. The absence of the element means
    that no Credit Event Notice, Notice of Publicly Available Information, Notice of
    Physical Settlement or Notice of Intended Physical Settlement, as applicable, has
    been delivered by or to the Transferor or the Remaining Party.'

    <contractualDefinitions> ContractualDefinitions </contractualDefinitions> [0..*]
    'The definitions (such as those published by ISDA) that will define the terms of the
    novation transaction.'

    <contractualTermsSupplement> ContractualTermsSupplement </contractualTermsSupplement> [0..*]
    'A contractual supplement (such as those published by ISDA) that will apply to the trade.'

    <payment> Payment </payment> [0..1]
    'Payment for the Novation.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="ContractNovation">
   <xsd:sequence>
      <xsd:group ref=" ContractNovationDetails.model "/>
      <xsd:element name="payment" type=" Payment " minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: ContractReference

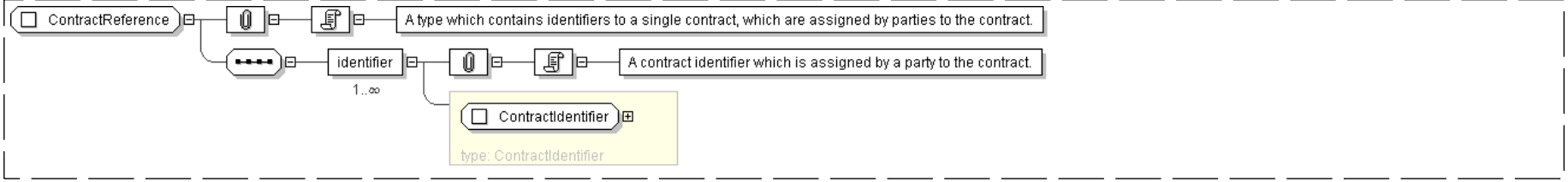
Super-types:	None
Sub-types:	None

Name	ContractReference
Used by (from the same schema document)	Complex Type **ChangeContract** , Model Group **ContractNovationDetails.model** , Model Group **ContractNovationDetails.model** , Model Group **ContractNovationDetails.model** , Model Group **ContractOrContractReference.model**
Abstract	no
Documentation	A type which contains identifiers to a single contract, which are assigned by parties to the contract.

XML Instance Representation

```
<...>
   <identifier> ContractIdentifier </identifier> [1..*]
   'A contract identifier which is assigned by a party to the contract.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="ContractReference">
   <xsd:sequence>
      <xsd:element name="identifier" type=" ContractIdentifier " maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: ContractTermination

Super-types:	ChangeContract < **ContractTermination** (by extension)
Sub-types:	None

Name	ContractTermination
Abstract	no
Documentation	Contract Termination Details.

XML Instance Representation

```
<...>
   <contractReference> ContractReference </contractReference> [1]
   'Identification of the Contract which is subject to change.'

   <date> xsd:date </date> [1]
   'The date on which the the parties enter into the change.'

   <effectiveDate> xsd:date </effectiveDate> [1]
   'The date on which the change becomes effective.'

   <payment> Payment </payment> [0..1]
   'Payment for the right to change the Contract.'

</...>
```

Diagram

Schema Component Representation

```
<xsd:complexType name="ContractTermination">
   <xsd:complexContent>
      <xsd:extension base=" ChangeContract "/>
   </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: CreditDerivativesNotices

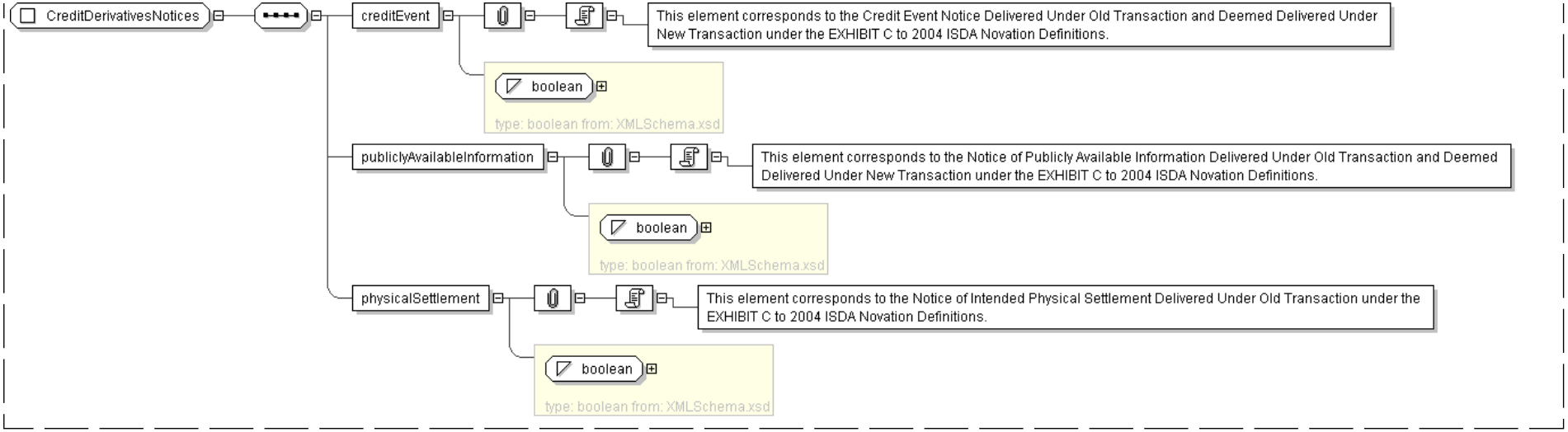
Super-types:	None
Sub-types:	None

Name	CreditDerivativesNotices
Used by (from the same schema document)	Model Group **ContractNovationDetails.model**
Abstract	no

XML Instance Representation

```
<...>
   <creditEvent> xsd:boolean </creditEvent> [1]
   'This element corresponds to the Credit Event Notice Delivered Under Old Transaction and
   Deemed Delivered Under New Transaction under the EXHIBIT C to 2004 ISDA Novation Definitions.'

   <publiclyAvailableInformation> xsd:boolean </publiclyAvailableInformation> [1]
   'This element corresponds to the Notice of Publicly Available Information Delivered Under
   Old Transaction and Deemed Delivered Under New Transaction under the EXHIBIT C to 2004
   ISDA Novation Definitions.'

   <physicalSettlement> xsd:boolean </physicalSettlement> [1]
   'This element corresponds to the Notice of Intended Physical Settlement Delivered Under
   Old Transaction under the EXHIBIT C to 2004 ISDA Novation Definitions.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="CreditDerivativesNotices">
   <xsd:sequence>
      <xsd:element name="creditEvent" type=" xsd:boolean "/>
      <xsd:element name="publiclyAvailableInformation" type=" xsd:boolean "/>
      <xsd:element name="physicalSettlement" type=" xsd:boolean "/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: DataDocument

Super-types:	Document < **DataDocument** (by extension)
Sub-types:	None

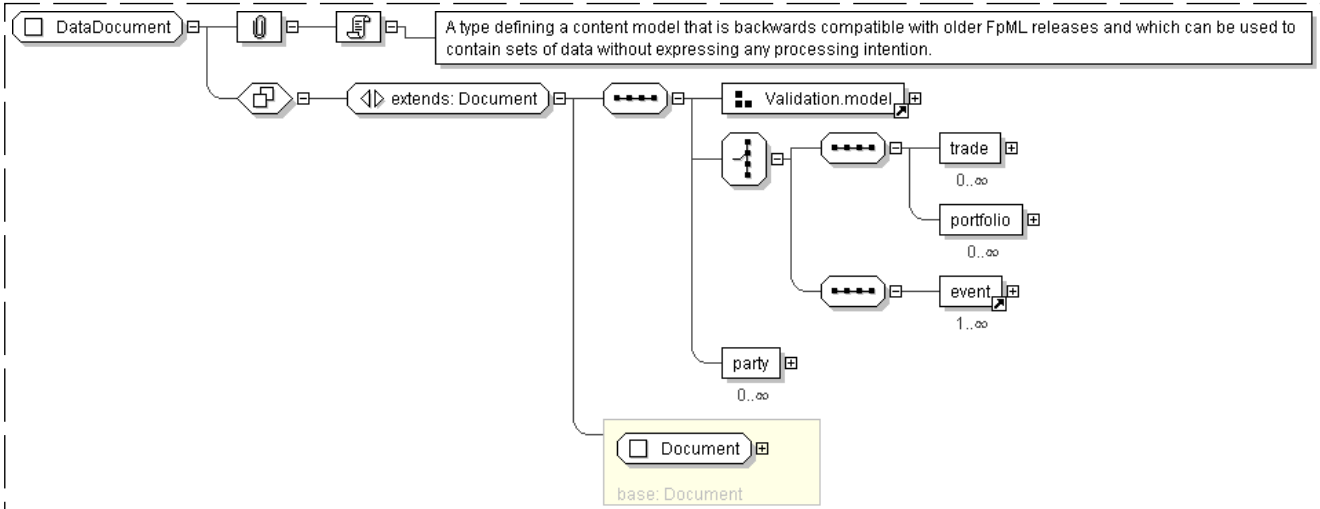
Name	DataDocument
Abstract	no
Documentation	A type defining a content model that is backwards compatible with older FpML releases and which can be used to contain sets of data without expressing any processing intention.

XML Instance Representation

```
<...
version=" xsd:token (value comes from list: {'4-0'|'4-1'|'4-2'|'4-3'|'4-4'}) [1]
'Indicate which version of the FpML Schema an FpML message adheres to.'
"
expectedBuild=" xsd:positiveInteger [0..1]
'This optional attribute can be supplied by a message creator in an FpML instance to
specify which build number of the schema was used to define the message when it was generated.'
"
actualBuild="2 [0..1]
'The specific build number of this schema version. This attribute is not included in
an instance document. Instead, it is supplied by the XML parser when the document is
validated against the FpML schema and indicates the build number of the schema file. Every
time FpML publishes a change to the schema, validation rules, or examples within a version
(e.g., version 4.2) the actual build number is incremented. If no changes have been
made between releases within a version (i.e. from Trial Recommendation to Recommendation)
the actual build number stays the same.'
```

```
">
   <validation> Validation </validation> [0..*]
Start Choice [1]
      <trade> Trade </trade> [0..*]
      'The root element in an FpML trade document.'

      <portfolio> Portfolio </portfolio> [0..*]
      'An arbitary grouping of trade references (and possibly other portfolios).'

      <event> ... </event> [1..*]
      'A business event.'

End Choice
   <party> Party </party> [0..*]
   'A legal entity or a subdivision of a legal entity.','Parties can perform multiple roles in
   a trade lifecycle. For example, the principal parties obligated to make payments from time
   to time during the term of the trade, but may include other parties involved in, or
   incidental to, the trade, such as parties acting in the role of novation transferor/
   transferee, broker, calculation agent, etc. In FpML roles are defined in multiple places
   within a document.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="DataDocument">
   <xsd:complexContent>
      <xsd:extension base=" Document ">
         <xsd:sequence>
            <xsd:group ref=" Validation.model "/>
            <xsd:choice>
               <xsd:sequence>
                  <xsd:element name="trade" type=" Trade " minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="portfolio" type=" Portfolio " minOccurs="0" maxOccurs="unbounded"/>
               </xsd:sequence>
               <xsd:sequence>
                  <xsd:element ref=" event " maxOccurs="unbounded"/>
               </xsd:sequence>
            </xsd:choice>
            <xsd:element name="party" type=" Party " minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:extension>
```

```
    </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: Document

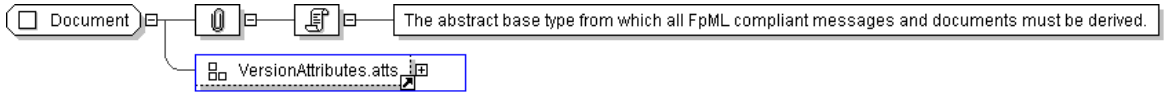
Super-types:	None
Sub-types:	• DataDocument (by extension)

Name	Document
Abstract	yes
Documentation	The abstract base type from which all FpML compliant messages and documents must be derived.

XML Instance Representation

```
<...
version=" xsd:token (value comes from list: {'4-0'|'4-1'|'4-2'|'4-3'|'4-4'}) [1]
 'Indicate which version of the FpML Schema an FpML message adheres to.'

"
expectedBuild=" xsd:positiveInteger [0..1]
 'This optional attribute can be supplied by a message creator in an FpML instance to
 specify which build number of the schema was used to define the message when it was generated.'

"
actualBuild="2 [0..1]
 'The specific build number of this schema version. This attribute is not included in
 an instance document. Instead, it is supplied by the XML parser when the document is
 validated against the FpML schema and indicates the build number of the schema file. Every
 time FpML publishes a change to the schema, validation rules, or examples within a version
 (e.g., version 4.2) the actual build number is incremented. If no changes have been
 made between releases within a version (i.e. from Trial Recommendation to Recommendation)
 the actual build number stays the same.'

"/>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Document" abstract="true">
   <xsd:attributeGroup ref=" VersionAttributes.atts "/>
</xsd:complexType>
```

top

Complex Type: Event

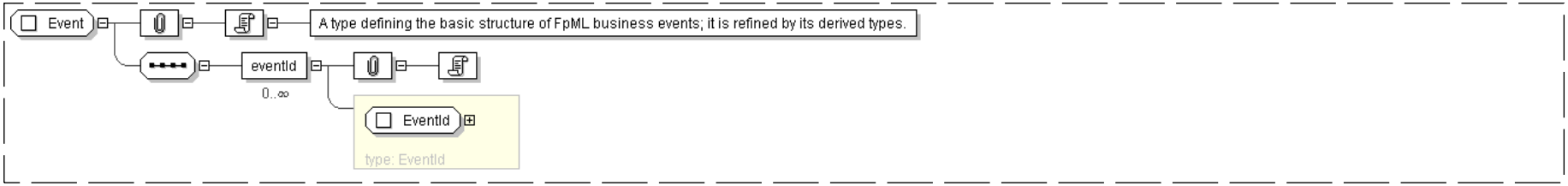
Super-types:	None
Sub-types:	• Amendment (by extension) • Increase (by extension)

Name	Event
Used by (from the same schema document)	Element **event**
Abstract	yes
Documentation	A type defining the basic structure of FpML business events; it is refined by its derived types.

XML Instance Representation

```
<...>
   <eventId> EventId </eventId> [0..*]
   ''

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Event" abstract="true">
   <xsd:sequence>
      <xsd:element name="eventId" type=" EventId " minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: EventId

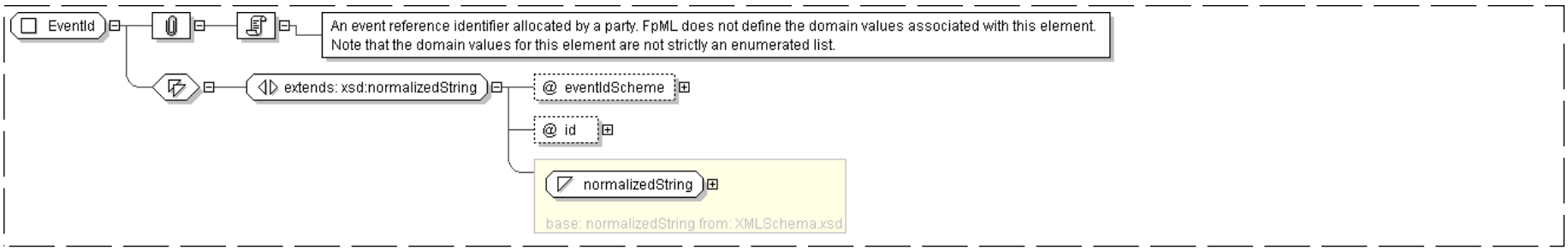
Super-types:	xsd:normalizedString < **EventId** (by extension)
Sub-types:	None

Name	EventId
Used by (from the same schema document)	Complex Type **Event**
Abstract	no
Documentation	An event reference identifier allocated by a party. FpML does not define the domain values associated with this element. Note that the domain values for this element are not strictly an enumerated list.

XML Instance Representation

```
<...
eventIdScheme=" xsd:anyURI [1]"
id=" xsd:ID [0..1]">
xsd:normalizedString
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="EventId">
```

```
   <xsd:simpleContent>
      <xsd:extension base=" xsd:normalizedString ">
         <xsd:attribute name="eventIdScheme" type=" xsd:anyURI " use="required"/>
         <xsd:attribute name="id" type=" xsd:ID "/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: ExecutionDateTime

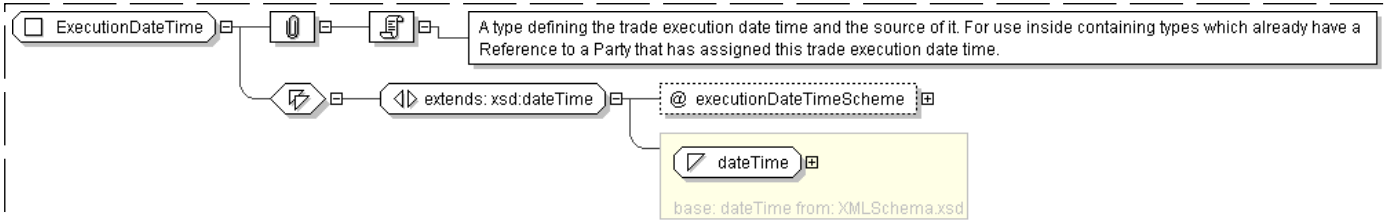
Super-types:	xsd:dateTime < **ExecutionDateTime** (by extension)
Sub-types:	None

Name	ExecutionDateTime
Used by (from the same schema document)	Complex Type **PartyTradeInformation**
Abstract	no
Documentation	A type defining the trade execution date time and the source of it. For use inside containing types which already have a Reference to a Party that has assigned this trade execution date time.

XML Instance Representation

```
<...
executionDateTimeScheme=" xsd:anyURI [0..1]
'Identification of the source (e.g. clock id) generating the execution date time.'

">
xsd:dateTime
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="ExecutionDateTime">
   <xsd:simpleContent>
      <xsd:extension base=" xsd:dateTime ">
         <xsd:attribute name="executionDateTimeScheme" type=" xsd:anyURI "/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: FirstPeriodStartDate

Super-types:	xsd:date < **FirstPeriodStartDate** (by extension)
Sub-types:	None

Name	FirstPeriodStartDate
Used by (from the same schema document)	Model Group **ContractNovationDetails.model**

Abstract	no

XML Instance Representation

```
<...
href=" xsd:IDREF [1]">
xsd:date
</...>
```

Diagram

Schema Component Representation

```
<xsd:complexType name="FirstPeriodStartDate">
   <xsd:simpleContent>
      <xsd:extension base=" xsd:date ">
         <xsd:attribute name="href" type=" xsd:IDREF " use="required" reference="Party"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: Increase

Super-types:	Event < **Increase** (by extension)
Sub-types:	None

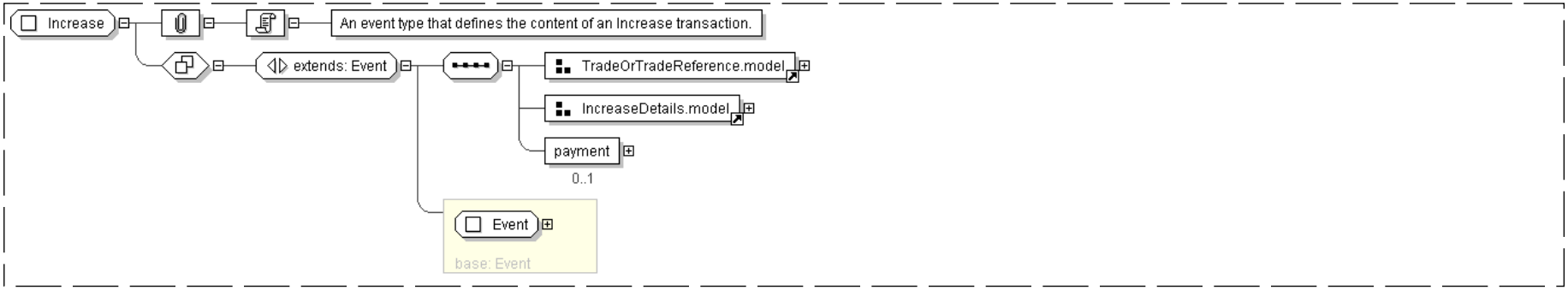
Name	Increase
Abstract	no
Documentation	An event type that defines the content of an Increase transaction.

XML Instance Representation

```
<...>
   <eventId> EventId </eventId> [0..*]
   ''

Start Choice [1]
      <trade> Trade </trade> [1]
      'An element that allows the full details of the trade to be used as a mechanism for
      identifying the trade for which the post-trade event pertains'

      <tradeReference> PartyTradeIdentifiers </tradeReference> [1]
      'A container since an individual trade can be referenced by two or more
      different partyTradeIdentifier elements - each allocated by a different party.'

End Choice
   <increaseTradeDate> xsd:date </increaseTradeDate> [1]
   'The date on which the the parties enter into the Increase transaction'

   <increaseEffectiveDate> xsd:date </increaseEffectiveDate> [1]
   'The date on which the Increase becomes effective'

Start Choice [1]
      <increaseInNotionalAmount> Money </increaseInNotionalAmount> [1]
      'Specifies the fixed amount by which the Notional increases due to the Increase transaction.'
```

```
      <outstandingNotionalAmount> Money </outstandingNotionalAmount> [1]
      'Specifies the Notional amount after the Increase.'

      <increaseInNumberOfOptions> xsd:decimal </increaseInNumberOfOptions> [1]
      'Specifies the fixed amount by which the Number of Options increases due to the
      Increase transaction.'

      <outstandingNumberOfOptions> xsd:decimal </outstandingNumberOfOptions> [1]
      'Specifies the Number of Options after the Increase.'

End Choice
   <payment> Payment </payment> [0..1]
   'A payment for the right to increase the trade.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Increase">
   <xsd:complexContent>
      <xsd:extension base=" Event ">
         <xsd:sequence>
            <xsd:group ref=" TradeOrTradeReference.model "/>
            <xsd:group ref=" IncreaseDetails.model "/>
            <xsd:element name="payment" type=" Payment " minOccurs="0"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: IndependentAmount

Super-types:	None
Sub-types:	None

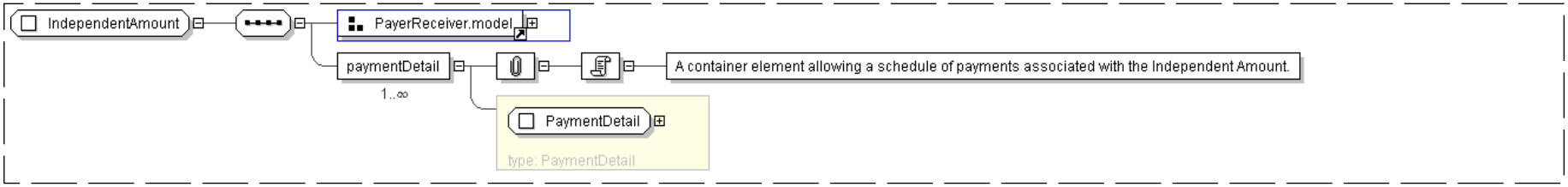
Name	IndependentAmount
Used by (from the same schema document)	Complex Type **Collateral**
Abstract	no

XML Instance Representation

```
<...>
   <payerPartyReference> PartyOrAccountReference </payerPartyReference> [1]
   'A reference to the party responsible for making the payments defined by this structure.'
```

```
<receiverPartyReference> PartyOrAccountReference </receiverPartyReference> [1]
'A reference to the party that receives the payments corresponding to this structure.'

<paymentDetail> PaymentDetail </paymentDetail> [1..*]
'A container element allowing a schedule of payments associated with the Independent Amount.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="IndependentAmount">
   <xsd:sequence>
      <xsd:group ref=" PayerReceiver.model "/>
      <xsd:element name="paymentDetail" type=" PaymentDetail " maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: LinkId

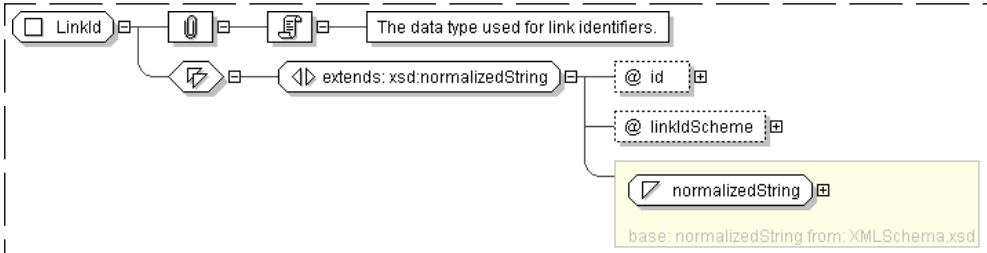
Super-types:	xsd:normalizedString < **LinkId** (by extension)
Sub-types:	None

Name	LinkId
Used by (from the same schema document)	Complex Type **PartyTradeIdentifier**
Abstract	no
Documentation	The data type used for link identifiers.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]"
linkIdScheme=" xsd:anyURI [1]">
xsd:normalizedString
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="LinkId">
```

```
   <xsd:simpleContent>
      <xsd:extension base=" xsd:normalizedString ">
         <xsd:attribute name="id" type=" xsd:ID "/>
         <xsd:attribute name="linkIdScheme" type=" xsd:anyURI " use="required"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: PartyPortfolioName

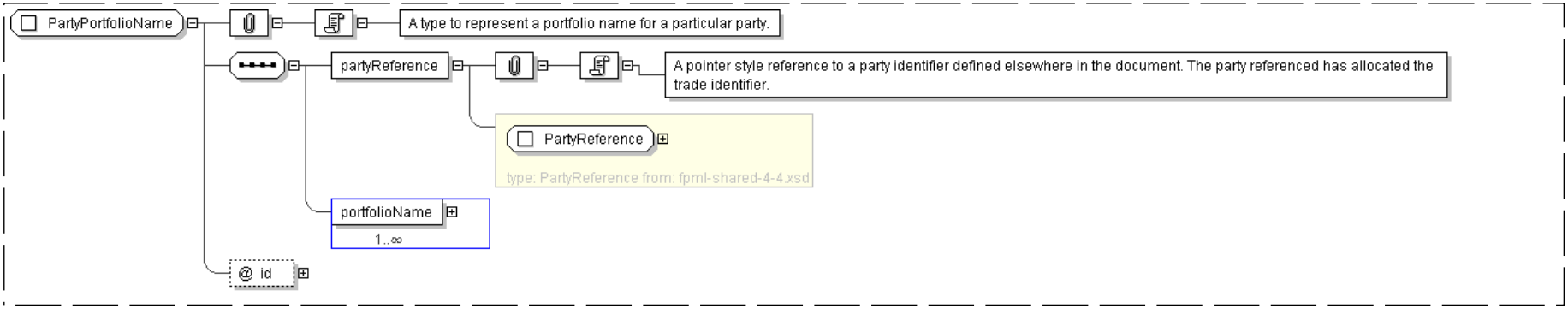
Super-types:	None
Sub-types:	None

Name	PartyPortfolioName
Used by (from the same schema document)	Complex Type **Portfolio**
Abstract	no
Documentation	A type to represent a portfolio name for a particular party.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]">
   <partyReference> PartyReference </partyReference> [1]
   'A pointer style reference to a party identifier defined elsewhere in the document. The
   party referenced has allocated the trade identifier.'

   <portfolioName> PortfolioName </portfolioName> [1..*]
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="PartyPortfolioName">
   <xsd:sequence>
      <xsd:element name="partyReference" type=" PartyReference "/>
      <xsd:element name="portfolioName" type=" PortfolioName " maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="id" type=" xsd:ID "/>
</xsd:complexType>
```

top

Complex Type: PartyRole

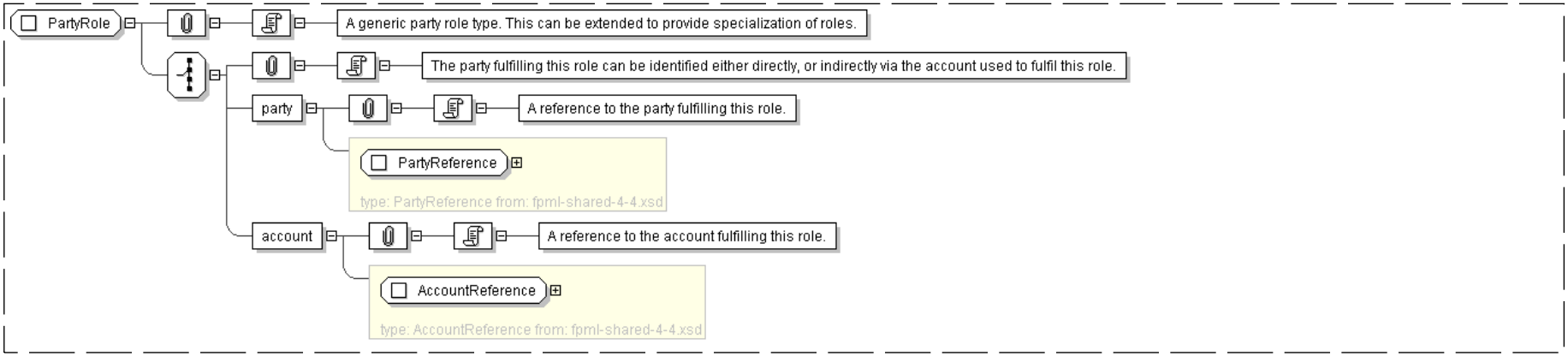
Super-types:	None
Sub-types:	None

Name	PartyRole
Used by (from the same schema document)	Complex Type **TradeSide** , Complex Type **TradeSide** , Complex Type **TradeSide** , Complex Type **TradeSide** , Complex Type **TradeSide** , Complex Type **TradeSide** , Complex Type **TradeSide** , Complex Type **TradeSide**
Abstract	no
Documentation	A generic party role type. This can be extended to provide specialization of roles.

XML Instance Representation

```
<...>
Start Choice [1]
  'The party fulfilling this role can be identified either directly, or indirectly via
  the account used to fulfil this role.'

      <party> PartyReference </party> [1]
      'A reference to the party fulfilling this role.'

      <account> AccountReference </account> [1]
      'A reference to the account fulfilling this role.'

End Choice
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="PartyRole">
   <xsd:choice>
      <xsd:element name="party" type=" PartyReference "/>
      <xsd:element name="account" type=" AccountReference "/>
   </xsd:choice>
</xsd:complexType>
```

top

Complex Type: PartyTradeIdentifier

Super-types:	TradeIdentifier < **PartyTradeIdentifier** (by extension)
Sub-types:	• AllocationTradeIdentifier (by extension) • BlockTradeIdentifier (by extension)

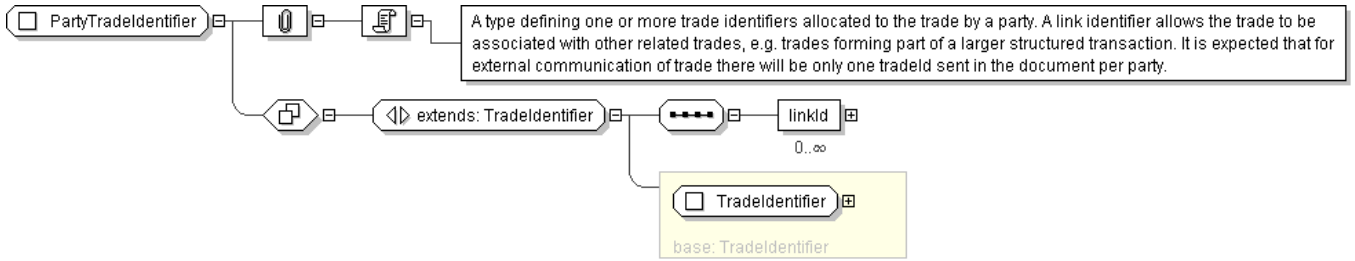
Name	PartyTradeIdentifier

Used by (from the same schema document)	Complex Type **Allocation** , Complex Type **AllocationTradeIdentifier** , Complex Type **BlockTradeIdentifier** , Complex Type **BlockTradeIdentifier** , Complex Type **PartyTradeIdentifiers** , Complex Type **TradeHeader**
Abstract	no
Documentation	A type defining one or more trade identifiers allocated to the trade by a party. A link identifier allows the trade to be associated with other related trades, e. g. trades forming part of a larger structured transaction. It is expected that for external communication of trade there will be only one tradeId sent in the document per party.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]">
   <partyReference> PartyReference </partyReference> [1]
   'A pointer style reference to a party identifier defined elsewhere in the document. The
   party referenced has allocated the trade identifier.'

Start Choice [1..*]
      <tradeId> TradeId </tradeId> [1]
      <versionedTradeId> VersionedTradeId </versionedTradeId> [1]
End Choice
   <linkId> LinkId </linkId> [0..*]
   'A link identifier allowing the trade to be associated with other related trades, e.g.
   the linkId may contain a tradeId for an associated trade or several related trades may be
   given the same linkId. FpML does not define the domain values associated with this
   element. Note that the domain values for this element are not strictly an enumerated list.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="PartyTradeIdentifier">
   <xsd:complexContent>
      <xsd:extension base=" TradeIdentifier ">
         <xsd:sequence>
            <xsd:element name="linkId" type=" LinkId " minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: PartyTradeIdentifiers

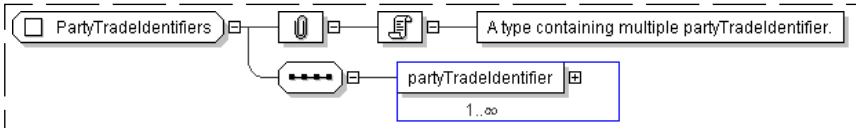
Super-types:	None
Sub-types:	None

Name	PartyTradeIdentifiers
Used by (from the same schema document)	Model Group **TradeOrTradeReference.model**
Abstract	no
Documentation	A type containing multiple partyTradeIdentifier.

XML Instance Representation

```
<...>
   <partyTradeIdentifier> PartyTradeIdentifier </partyTradeIdentifier> [1..*]
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="PartyTradeIdentifiers">
   <xsd:sequence>
      <xsd:element name="partyTradeIdentifier" type=" PartyTradeIdentifier " maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: PartyTradeInformation

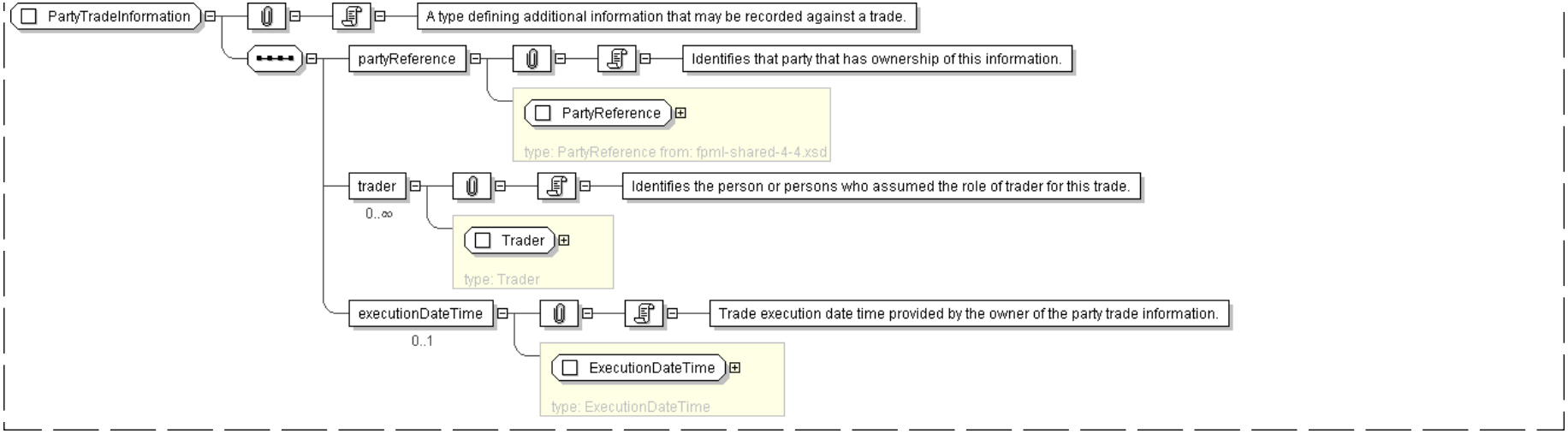
Super-types:	None
Sub-types:	None

Name	PartyTradeInformation
Used by (from the same schema document)	Complex Type **TradeHeader**
Abstract	no
Documentation	A type defining additional information that may be recorded against a trade.

XML Instance Representation

```
<...>
   <partyReference> PartyReference </partyReference> [1]
   'Identifies that party that has ownership of this information.'

   <trader> Trader </trader> [0..*]
   'Identifies the person or persons who assumed the role of trader for this trade.'

   <executionDateTime> ExecutionDateTime </executionDateTime> [0..1]
   'Trade execution date time provided by the owner of the party trade information.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="PartyTradeInformation">
   <xsd:sequence>
      <xsd:element name="partyReference" type=" PartyReference "/>
      <xsd:element name="trader" type=" Trader " minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="executionDateTime" type=" ExecutionDateTime " minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: PaymentDetail

Super-types:	None
Sub-types:	None

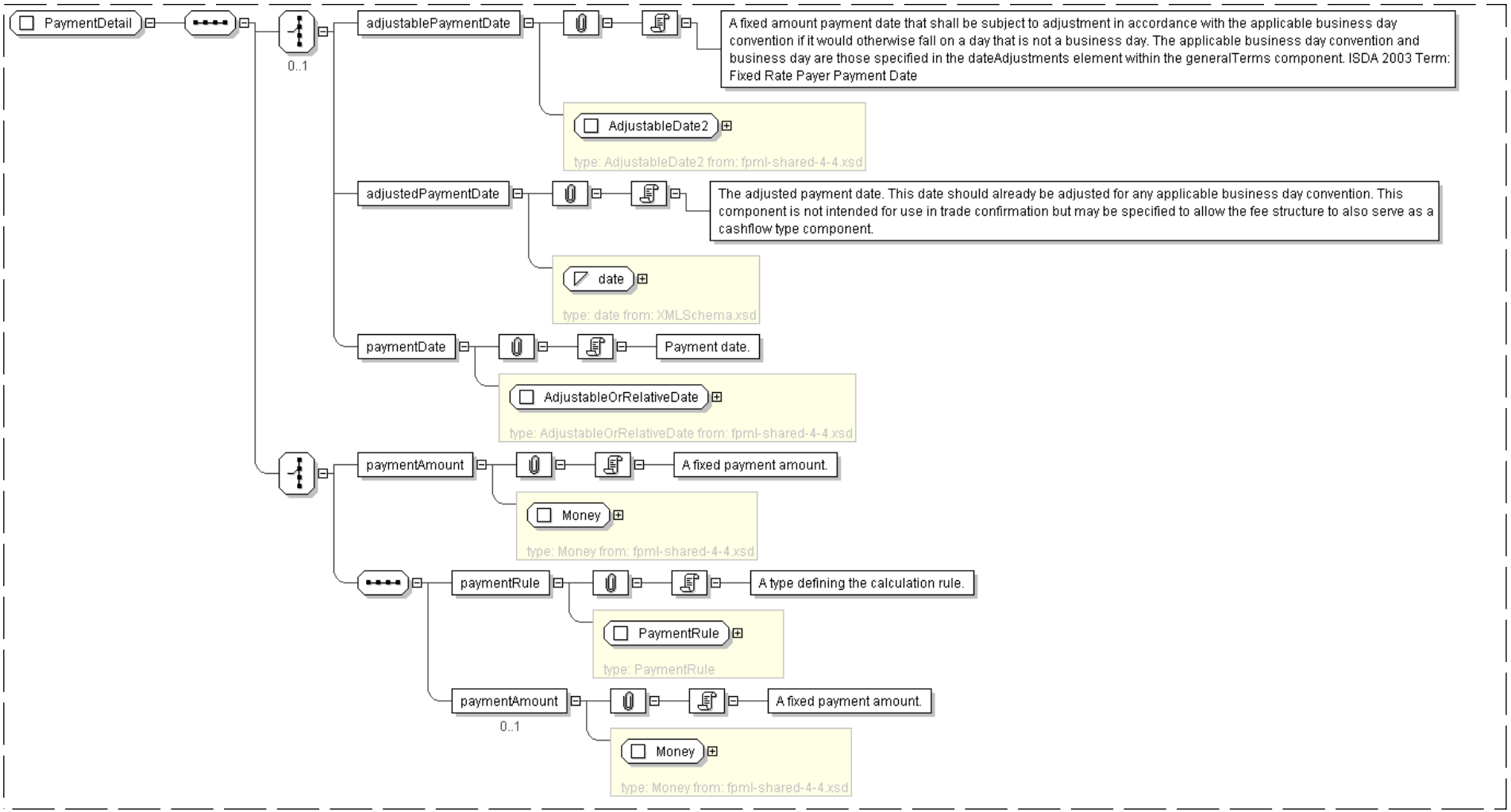
Name	PaymentDetail
Used by (from the same schema document)	Complex Type **IndependentAmount**
Abstract	no

XML Instance Representation

```
<...>
Start Choice [0..1]
      <adjustablePaymentDate> AdjustableDate2 </adjustablePaymentDate> [1]
      'A fixed amount payment date that shall be subject to adjustment in accordance with
      the applicable business day convention if it would otherwise fall on a day that is not
      a business day. The applicable business day convention and business day are those specified
      in the dateAdjustments element within the generalTerms component. ISDA 2003 Term: Fixed
      Rate Payer Payment Date'

      <adjustedPaymentDate> xsd:date </adjustedPaymentDate> [1]
      'The adjusted payment date. This date should already be adjusted for any applicable
      business day convention. This component is not intended for use in trade confirmation but
      may be specified to allow the fee structure to also serve as a cashflow type component.'

      <paymentDate> AdjustableOrRelativeDate </paymentDate> [1]
      'Payment date.'

End Choice
Start Choice [1]
```

```
      <paymentAmount> Money </paymentAmount> [1]
      'A fixed payment amount.'

      <paymentRule> PaymentRule </paymentRule> [1]
      'A type defining the calculation rule.'

      <paymentAmount> Money </paymentAmount> [0..1]
      'A fixed payment amount.'

End Choice
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="PaymentDetail">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="adjustablePaymentDate" type=" AdjustableDate2 "
         deprecated="true" deprecatedReason="This structure doesn't provide the ability to provide
         a payment date relative to another date. The paymentDate element of
```

```
        type AdjustableOrRelativeDate should be used instead. In version 5.0 the date structures
        will be rationalized."/>
        <xsd:element name="adjustedPaymentDate" type=" xsd:date "
        deprecated="true" deprecatedReason="In version 5.0 the date structures will be rationalized."/>
        <xsd:element name="paymentDate" type=" AdjustableOrRelativeDate "/>
      </xsd:choice>
      <xsd:choice>
        <xsd:element name="paymentAmount" type=" Money "/>
        <xsd:sequence>
          <xsd:element name="paymentRule" type=" PaymentRule "/>
          <xsd:element name="paymentAmount" type=" Money " minOccurs="0"/>
        </xsd:sequence>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
```

top

Complex Type: PaymentRule

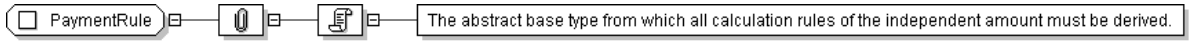
Super-types:	None
Sub-types:	• PercentageRule (by extension)

Name	PaymentRule
Used by (from the same schema document)	Complex Type **PaymentDetail**
Abstract	yes
Documentation	The abstract base type from which all calculation rules of the independent amount must be derived.

XML Instance Representation

```
<.../>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="PaymentRule" abstract="true"/>
```

top

Complex Type: PercentageRule

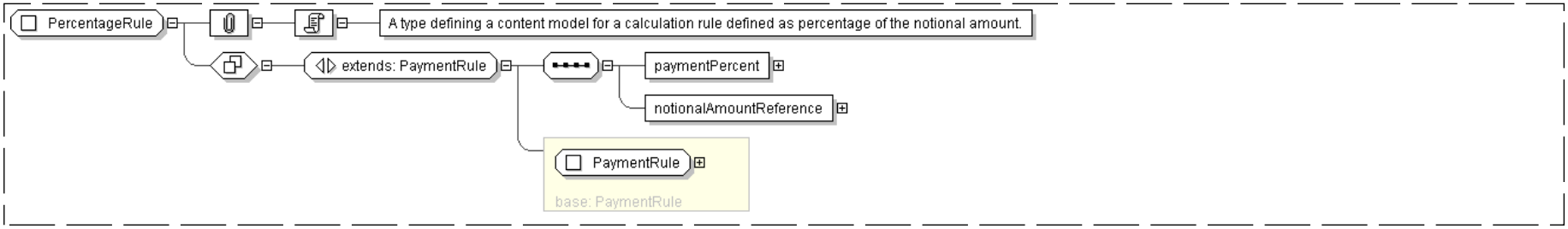
Super-types:	PaymentRule < **PercentageRule** (by extension)
Sub-types:	None

Name	PercentageRule
Abstract	no
Documentation	A type defining a content model for a calculation rule defined as percentage of the notional amount.

XML Instance Representation

```
<...>
  <paymentPercent> xsd:decimal </paymentPercent> [1]
  'A percentage of the notional amount.'

  <notionalAmountReference> NotionalAmountReference </notionalAmountReference> [1]
  'A reference to the notional amount.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="PercentageRule">
   <xsd:complexContent>
      <xsd:extension base=" PaymentRule ">
         <xsd:sequence>
            <xsd:element name="paymentPercent" type=" xsd:decimal "/>
            <xsd:element name="notionalAmountReference" type=" NotionalAmountReference "/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: Portfolio

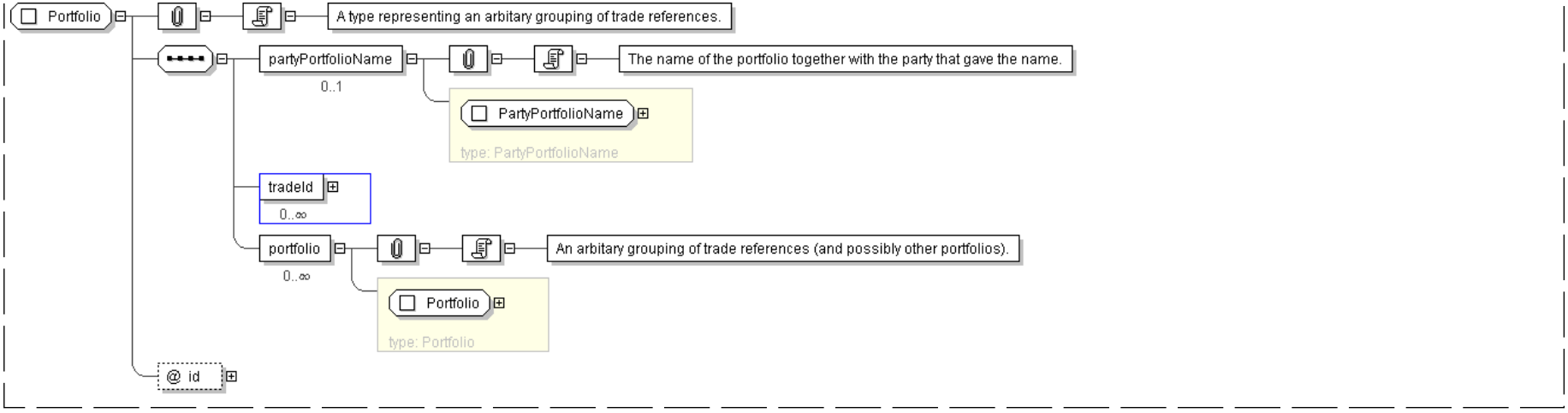
Super-types:	None
Sub-types:	• QueryPortfolio (by extension)

Name	Portfolio
Used by (from the same schema document)	Complex Type **DataDocument** , Complex Type **Portfolio**
Abstract	no
Documentation	A type representing an arbitary grouping of trade references.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]">
   <partyPortfolioName> PartyPortfolioName </partyPortfolioName> [0..1]
   'The name of the portfolio together with the party that gave the name.'

   <tradeId> TradeId </tradeId> [0..*]
   <portfolio> Portfolio </portfolio> [0..*]
   'An arbitary grouping of trade references (and possibly other portfolios).'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Portfolio">
   <xsd:sequence>
      <xsd:element name="partyPortfolioName" type=" PartyPortfolioName " minOccurs="0"/>
      <xsd:element name="tradeId" type=" TradeId " minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="portfolio" type=" Portfolio " minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="id" type=" xsd:ID "/>
</xsd:complexType>
```

top

Complex Type: PortfolioName

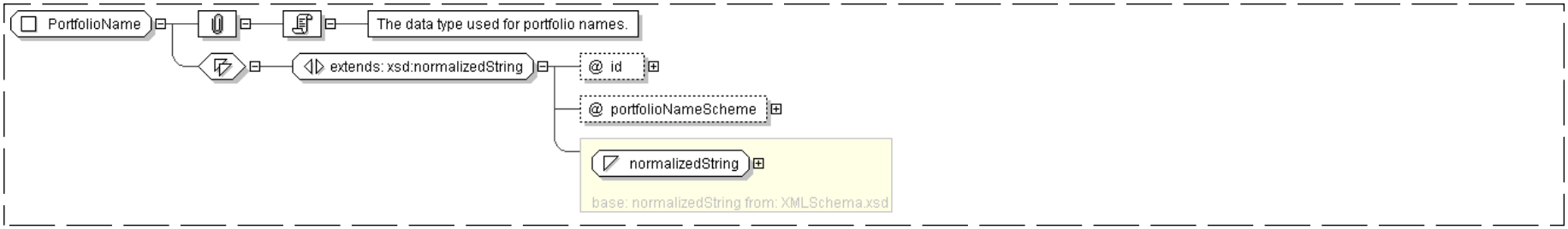
Super-types:	xsd:normalizedString < **PortfolioName** (by extension)
Sub-types:	None

Name	PortfolioName
Used by (from the same schema document)	Complex Type **PartyPortfolioName**
Abstract	no
Documentation	The data type used for portfolio names.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]"
portfolioNameScheme=" xsd:anyURI [0..1]">
xsd:normalizedString
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="PortfolioName">
   <xsd:simpleContent>
      <xsd:extension base=" xsd:normalizedString ">
         <xsd:attribute name="id" type=" xsd:ID "/>
         <xsd:attribute name="portfolioNameScheme" type=" xsd:anyURI "/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: QueryParameter

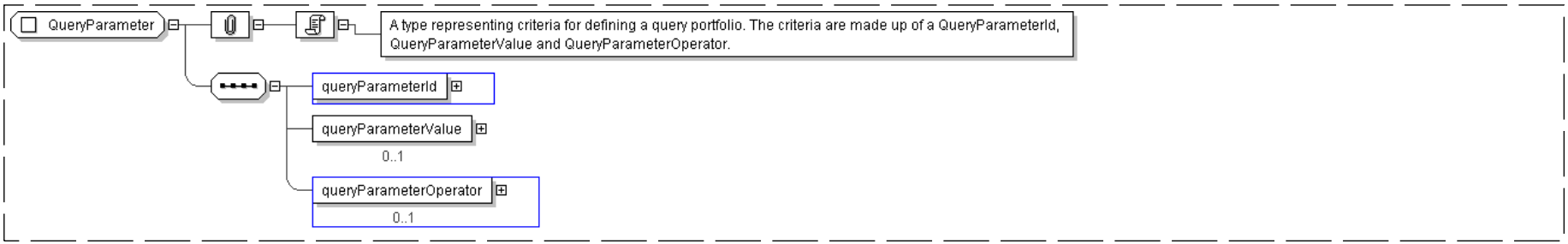
Super-types:	None
Sub-types:	None

Name	QueryParameter
Used by (from the same schema document)	Complex Type **QueryPortfolio**
Abstract	no
Documentation	A type representing criteria for defining a query portfolio. The criteria are made up of a QueryParameterId, QueryParameterValue and QueryParameterOperator.

XML Instance Representation

```
<...>
   <queryParameterId> QueryParameterId </queryParameterId> [1]
   <queryParameterValue> xsd:normalizedString </queryParameterValue> [0..1]
   <queryParameterOperator> QueryParameterOperator </queryParameterOperator> [0..1]
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="QueryParameter">
   <xsd:sequence>
      <xsd:element name="queryParameterId" type=" QueryParameterId "/>
      <xsd:element name="queryParameterValue" type=" xsd:normalizedString " minOccurs="0"/>
      <xsd:element name="queryParameterOperator" type=" QueryParameterOperator " minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: QueryParameterId

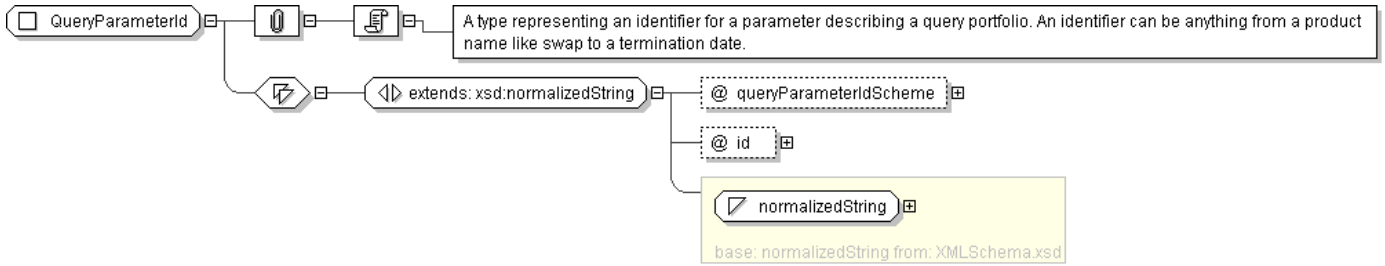
Super-types:	xsd:normalizedString < **QueryParameterId** (by extension)
Sub-types:	None

Name	QueryParameterId
Used by (from the same schema document)	Complex Type **QueryParameter**
Abstract	no
Documentation	A type representing an identifier for a parameter describing a query portfolio. An identifier can be anything from a product name like swap to a termination date.

XML Instance Representation

```
<...
queryParameterIdScheme=" xsd:anyURI [1]"
id=" xsd:ID [0..1]">
xsd:normalizedString
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="QueryParameterId">
   <xsd:simpleContent>
      <xsd:extension base=" xsd:normalizedString ">
         <xsd:attribute name="queryParameterIdScheme" type=" xsd:anyURI " use="required"/>
         <xsd:attribute name="id" type=" xsd:ID "/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: QueryParameterOperator

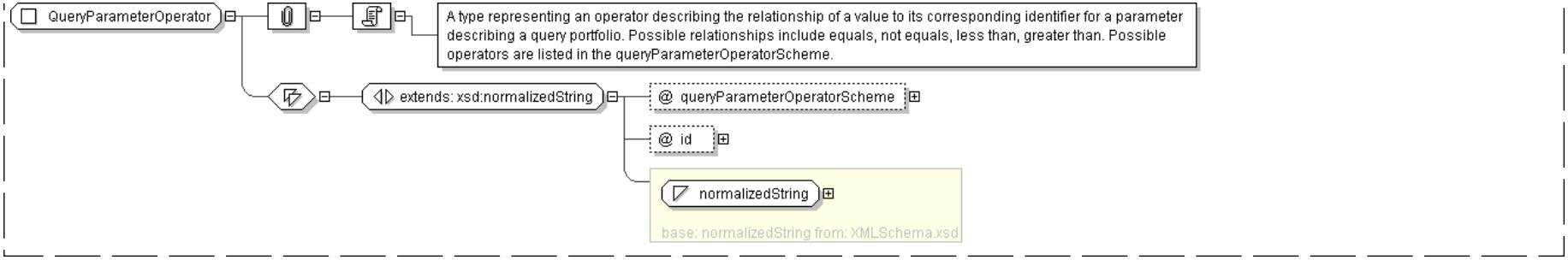
Super-types:	xsd:normalizedString < **QueryParameterOperator** (by extension)
Sub-types:	None

Name	QueryParameterOperator
Used by (from the same schema document)	Complex Type **QueryParameter**
Abstract	no
Documentation	A type representing an operator describing the relationship of a value to its corresponding identifier for a parameter describing a query portfolio. Possible relationships include equals, not equals, less than, greater than. Possible operators are listed in the queryParameterOperatorScheme.

XML Instance Representation

```
<...
queryParameterOperatorScheme=" xsd:anyURI [0..1]"
id=" xsd:ID [0..1]">
xsd:normalizedString
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="QueryParameterOperator">
   <xsd:simpleContent>
      <xsd:extension base=" xsd:normalizedString ">
         <xsd:attribute name="queryParameterOperatorScheme" type=" xsd:anyURI " default="http://www.
         fpml.org/coding-scheme/query-parameter-operator-1-0"/>
         <xsd:attribute name="id" type=" xsd:ID "/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: QueryPortfolio

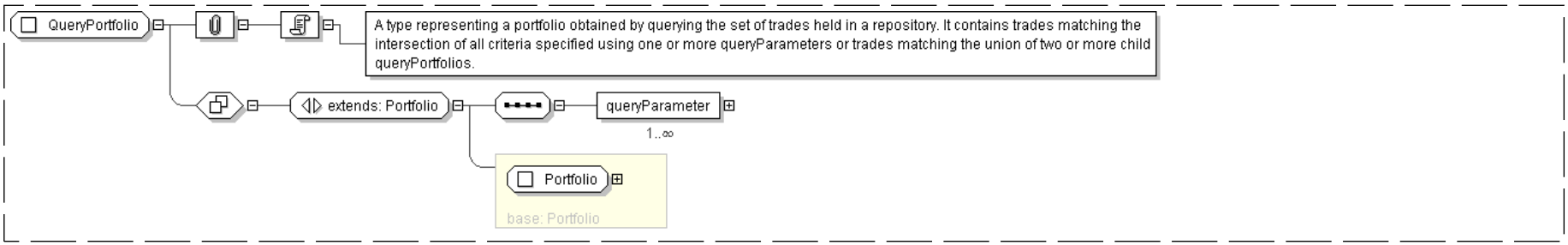
Super-types:	Portfolio < **QueryPortfolio** (by extension)
Sub-types:	None

Name	QueryPortfolio
Abstract	no
Documentation	A type representing a portfolio obtained by querying the set of trades held in a repository. It contains trades matching the intersection of all criteria specified using one or more queryParameters or trades matching the union of two or more child queryPortfolios.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]">
   <partyPortfolioName> PartyPortfolioName </partyPortfolioName> [0..1]
   'The name of the portfolio together with the party that gave the name.'

   <tradeId> TradeId </tradeId> [0..*]
   <portfolio> Portfolio </portfolio> [0..*]
   'An arbitary grouping of trade references (and possibly other portfolios).'

   <queryParameter> QueryParameter </queryParameter> [1..*]
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="QueryPortfolio">
   <xsd:complexContent>
      <xsd:extension base=" Portfolio ">
         <xsd:sequence>
            <xsd:element name="queryParameter" type=" QueryParameter " maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: Strategy

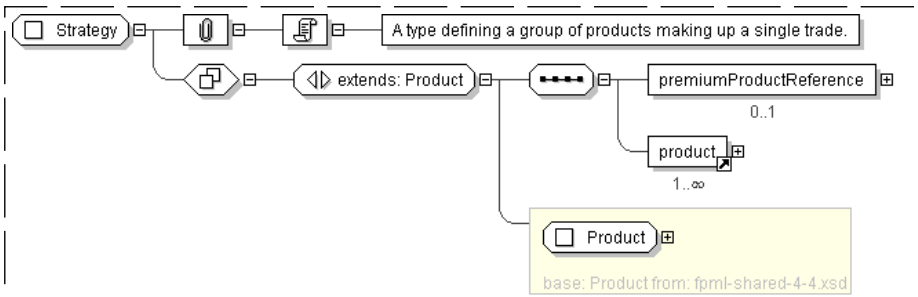
Super-types:	Product < **Strategy** (by extension)
Sub-types:	None

Name	Strategy
Used by (from the same schema document)	Element **strategy**
Abstract	no
Documentation	A type defining a group of products making up a single trade.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]">
   <productType> ProductType </productType> [0..*]
   'A classification of the type of product. FpML defines a simple product categorization using
   a coding scheme.'

   <productId> ProductId </productId> [0..*]
   'A product reference identifier allocated by a party. FpML does not define the domain
   values associated with this element. Note that the domain values for this element are
   not strictly an enumerated list.'

   <premiumProductReference> ProductReference </premiumProductReference> [0..1]
   'Indicates which product within a strategy represents the premium payment.'

   <product> ... </product> [1..*]
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Strategy">
   <xsd:complexContent>
      <xsd:extension base=" Product ">
         <xsd:sequence>
```

```
            <xsd:element name="premiumProductReference" type=" ProductReference " minOccurs="0"/>
            <xsd:element ref=" product " maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

top

Complex Type: Trade

Super-types:	None
Sub-types:	None

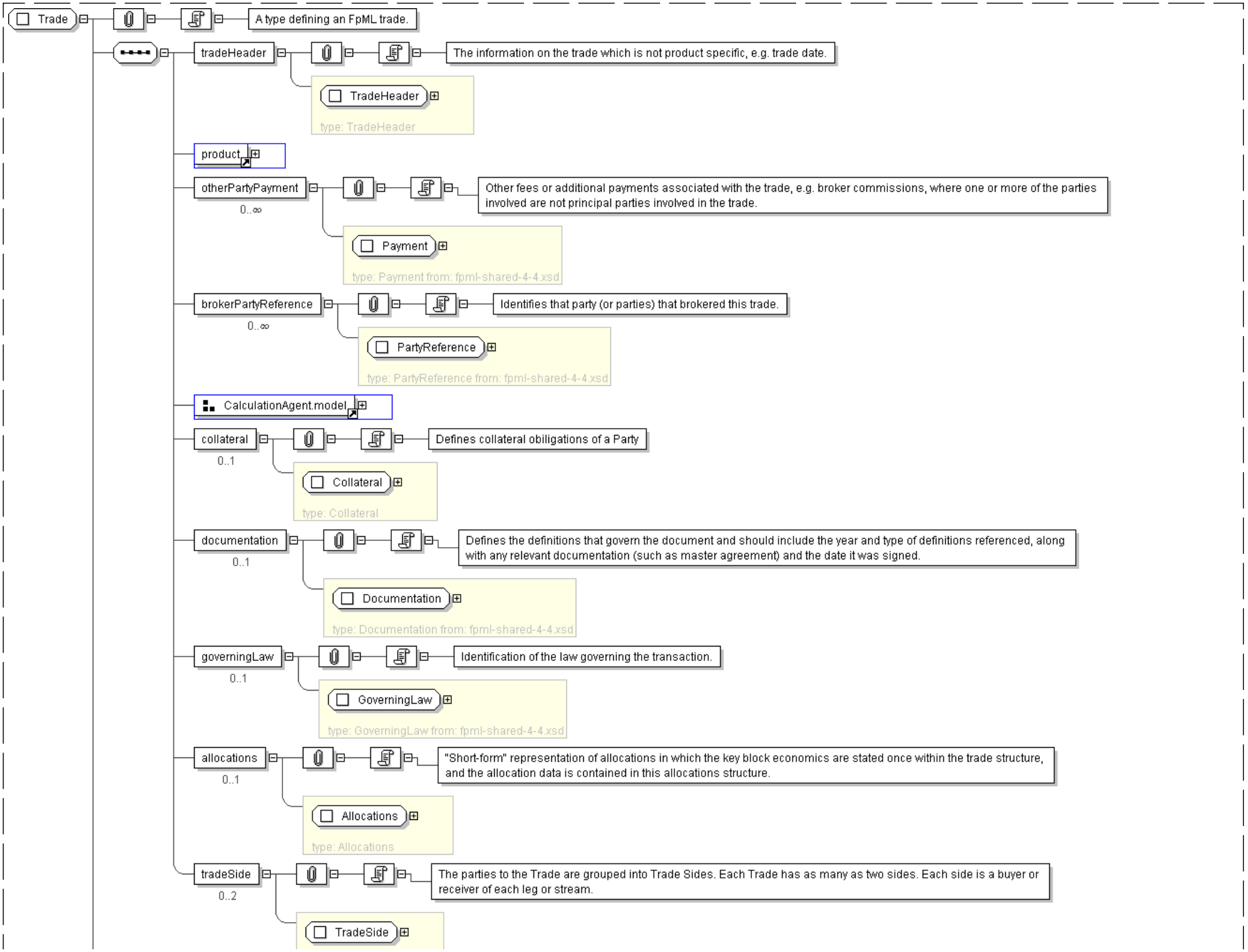
Name	Trade
Used by (from the same schema document)	Complex Type **Amendment** , Complex Type **DataDocument** , Model Group **TradeOrTradeReference.model**
Abstract	no
Documentation	A type defining an FpML trade.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]">
   <tradeHeader> TradeHeader </tradeHeader> [1]
   'The information on the trade which is not product specific, e.g. trade date.'

   <product> ... </product> [1]
   <otherPartyPayment> Payment </otherPartyPayment> [0..*]
   'Other fees or additional payments associated with the trade, e.g. broker commissions,
   where one or more of the parties involved are not principal parties involved in the trade.'

   <brokerPartyReference> PartyReference </brokerPartyReference> [0..*]
   'Identifies that party (or parties) that brokered this trade.'

   <calculationAgent> CalculationAgent </calculationAgent> [0..1]
   'The ISDA Calculation Agent responsible for performing duties associated with an optional
   early termination'

   <calculationAgentBusinessCenter> BusinessCenter </calculationAgentBusinessCenter> [0..1]
   'The city in which the office through which ISDA Calculation Agent is acting for purposes
   of the transaction is located The short-form confirm for a trade that is executed under
   a Sovereign or Asia Pacific Master Confirmation Agreement ( MCA ), does not need to specify
   the Calculation Agent. However, the confirm does need to specify the Calculation Agent
   City. This is due to the fact that the MCA sets the value for Calculation Agent but does
   not set the value for Calculation Agent City.'

   <collateral> Collateral </collateral> [0..1]
   'Defines collateral obiligations of a Party'

   <documentation> Documentation </documentation> [0..1]
   'Defines the definitions that govern the document and should include the year and type
   of definitions referenced, along with any relevant documentation (such as master agreement)
   and the date it was signed.'

   <governingLaw> GoverningLaw </governingLaw> [0..1]
   'Identification of the law governing the transaction.'

   <allocations> Allocations </allocations> [0..1]
   '"Short-form" representation of allocations in which the key block economics are stated
   once within the trade structure, and the allocation data is contained in this
   allocations structure.'

   <tradeSide> TradeSide </tradeSide> [0..2]
```

```
'The parties to the Trade are grouped into Trade Sides. Each Trade has as many as two
sides. Each side is a buyer or receiver of each leg or stream.'

</...>
```

Diagram



type: TradeSide

@ id

Schema Component Representation

```
<xsd:complexType name="Trade">
   <xsd:sequence>
      <xsd:element name="tradeHeader" type=" TradeHeader "/>
      <xsd:element ref=" product "/>
      <xsd:element name="otherPartyPayment" type=" Payment " minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="brokerPartyReference" type=" PartyReference "
      minOccurs="0" maxOccurs="unbounded"/>
      <xsd:group ref=" CalculationAgent.model "/>
      <xsd:element name="collateral" type=" Collateral " minOccurs="0"/>
      <xsd:element name="documentation" type=" Documentation " minOccurs="0"/>
      <xsd:element name="governingLaw" type=" GoverningLaw " minOccurs="0"/>
      <xsd:element name="allocations" type=" Allocations " minOccurs="0"/>
      <xsd:element name="tradeSide" type=" TradeSide " minOccurs="0" maxOccurs="2"/>
   </xsd:sequence>
   <xsd:attribute name="id" type=" xsd:ID "/>
</xsd:complexType>
```

top

Complex Type: TradeDifference

Super-types:	None
Sub-types:	None

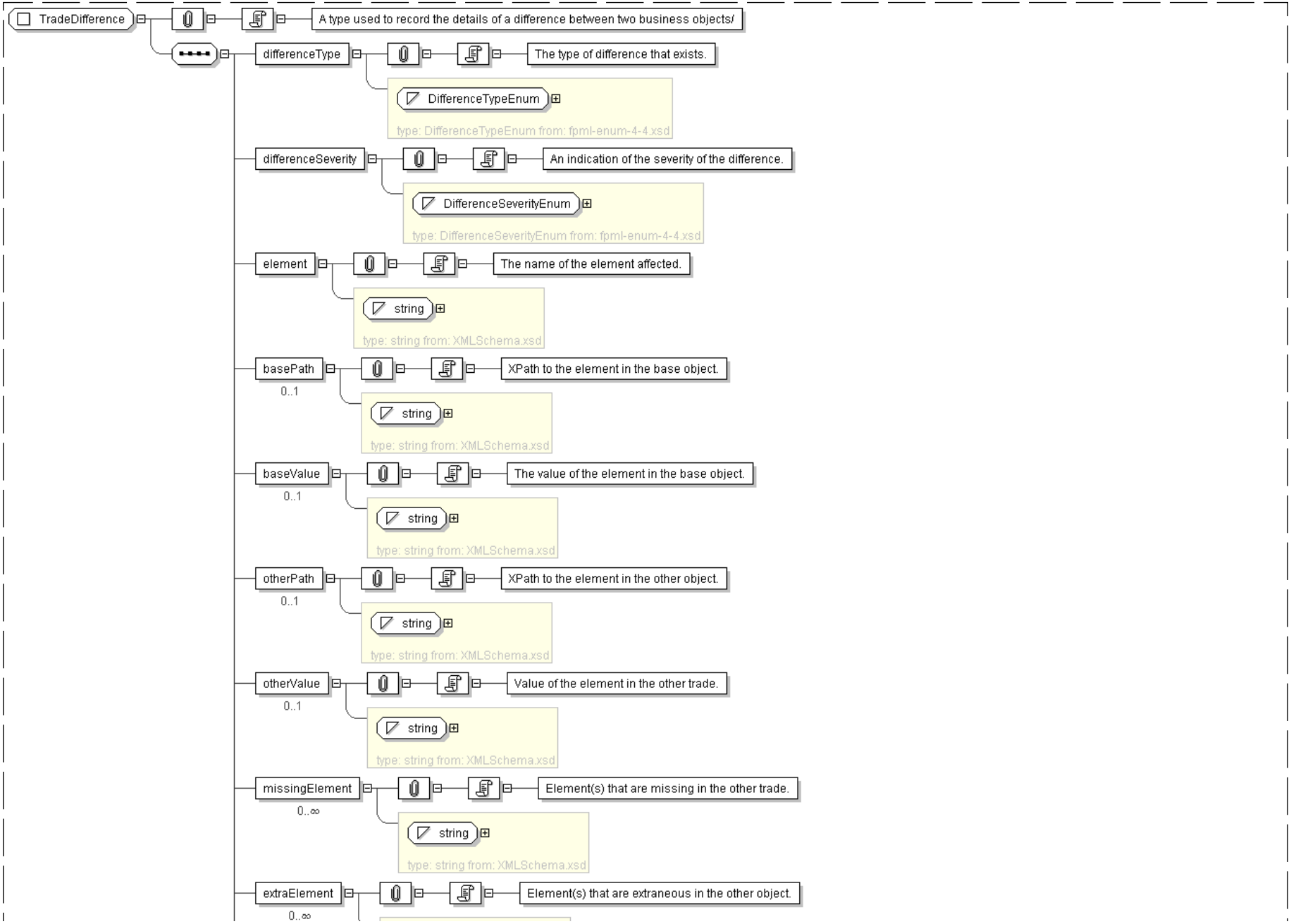
Name	TradeDifference
Used by (from the same schema document)	Complex Type **BestFitTrade**
Abstract	no
Documentation	A type used to record the details of a difference between two business objects/

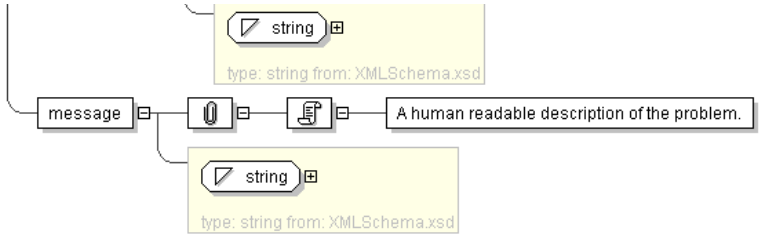
XML Instance Representation

```
<...>
   <differenceType> DifferenceTypeEnum </differenceType> [1]
   'The type of difference that exists.'

   <differenceSeverity> DifferenceSeverityEnum </differenceSeverity> [1]
   'An indication of the severity of the difference.'

   <element> xsd:string </element> [1]
   'The name of the element affected.'

   <basePath> xsd:string </basePath> [0..1]
   'XPath to the element in the base object.'

   <baseValue> xsd:string </baseValue> [0..1]
   'The value of the element in the base object.'

   <otherPath> xsd:string </otherPath> [0..1]
   'XPath to the element in the other object.'

   <otherValue> xsd:string </otherValue> [0..1]
   'Value of the element in the other trade.'

   <missingElement> xsd:string </missingElement> [0..*]
   'Element(s) that are missing in the other trade.'
```

```
<extraElement> xsd:string </extraElement> [0..*]
'Element(s) that are extraneous in the other object.'

<message> xsd:string </message> [1]
'A human readable description of the problem.'

</...>
```

Diagram





Schema Component Representation

```
<xsd:complexType name="TradeDifference">
   <xsd:sequence>
      <xsd:element name="differenceType" type=" DifferenceTypeEnum "/>
      <xsd:element name="differenceSeverity" type=" DifferenceSeverityEnum "/>
      <xsd:element name="element" type=" xsd:string "/>
      <xsd:element name="basePath" type=" xsd:string " minOccurs="0"/>
      <xsd:element name="baseValue" type=" xsd:string " minOccurs="0"/>
      <xsd:element name="otherPath" type=" xsd:string " minOccurs="0"/>
      <xsd:element name="otherValue" type=" xsd:string " minOccurs="0"/>
      <xsd:element name="missingElement" type=" xsd:string " minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="extraElement" type=" xsd:string " minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="message" type=" xsd:string "/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: TradeHeader

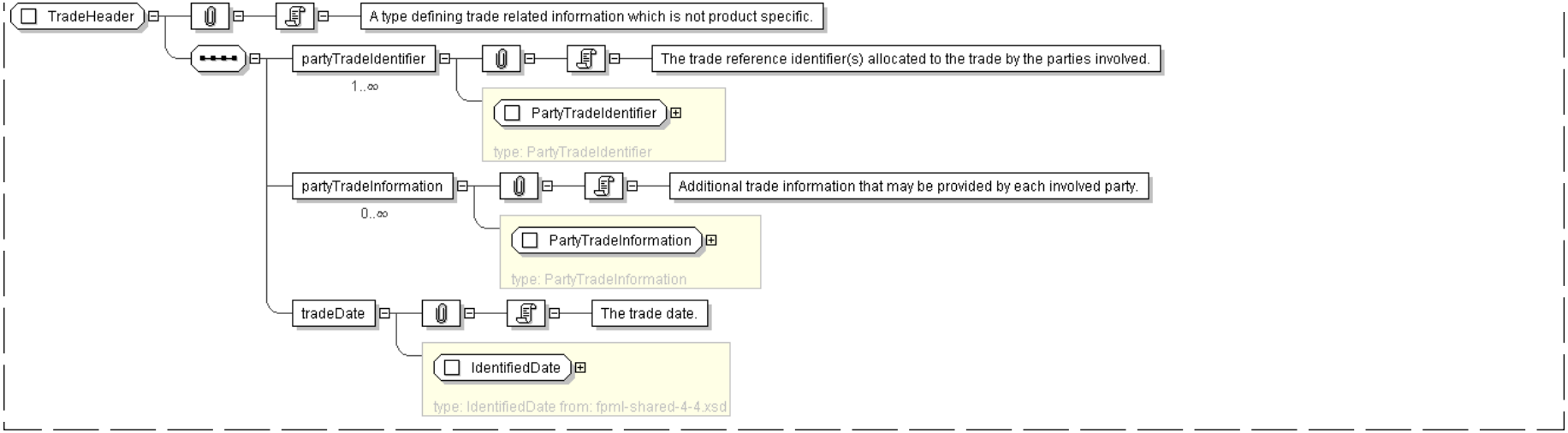
Super-types:	None
Sub-types:	None

Name	TradeHeader
Used by (from the same schema document)	Complex Type **Trade**
Abstract	no
Documentation	A type defining trade related information which is not product specific.

XML Instance Representation

```
<...>
   <partyTradeIdentifier> PartyTradeIdentifier </partyTradeIdentifier> [1..*]
   'The trade reference identifier(s) allocated to the trade by the parties involved.'

   <partyTradeInformation> PartyTradeInformation </partyTradeInformation> [0..*]
   'Additional trade information that may be provided by each involved party.'

   <tradeDate> IdentifiedDate </tradeDate> [1]
   'The trade date.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="TradeHeader">
   <xsd:sequence>
      <xsd:element name="partyTradeIdentifier" type=" PartyTradeIdentifier " maxOccurs="unbounded"/>
      <xsd:element name="partyTradeInformation" type=" PartyTradeInformation "
      minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="tradeDate" type=" IdentifiedDate "/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: TradeId

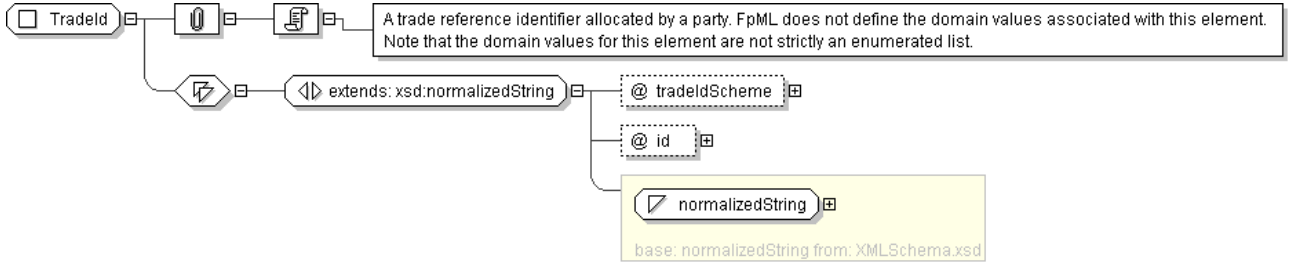
Super-types:	xsd:normalizedString < **TradeId** (by extension)
Sub-types:	None

Name	TradeId
Used by (from the same schema document)	Complex Type **Portfolio** , Complex Type **TradeIdentifier** , Complex Type **VersionedTradeId**
Abstract	no
Documentation	A trade reference identifier allocated by a party. FpML does not define the domain values associated with this element. Note that the domain values for this element are not strictly an enumerated list.

XML Instance Representation

```
<...
tradeIdScheme=" xsd:anyURI [1]"
id=" xsd:ID [0..1]">
xsd:normalizedString
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="TradeId">
   <xsd:simpleContent>
      <xsd:extension base=" xsd:normalizedString ">
         <xsd:attribute name="tradeIdScheme" type=" xsd:anyURI " use="required"/>
         <xsd:attribute name="id" type=" xsd:ID "/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: TradeIdentifier

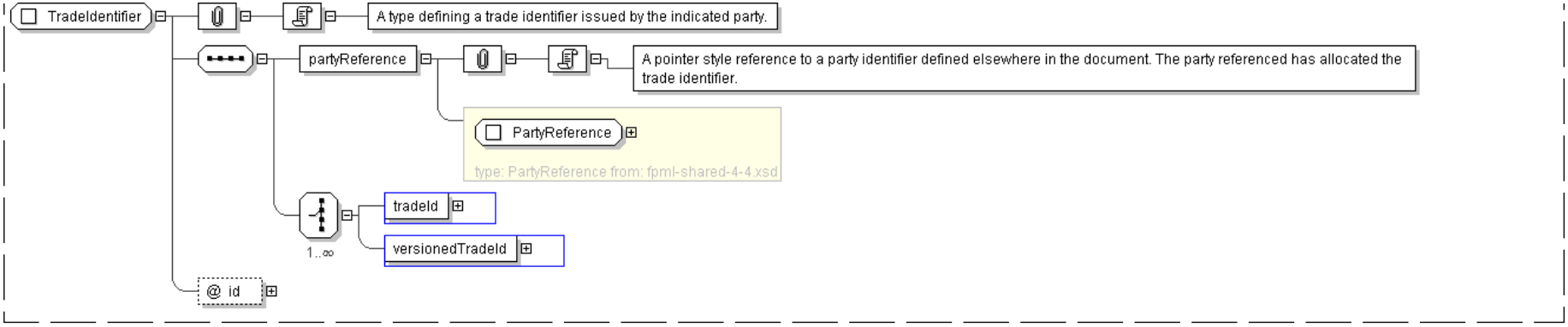
Super-types:	None
Sub-types:	• PartyTradeIdentifier (by extension) ◦ AllocationTradeIdentifier (by extension) ◦ BlockTradeIdentifier (by extension)

Name	TradeIdentifier
Used by (from the same schema document)	Complex Type **BestFitTrade**
Abstract	no
Documentation	A type defining a trade identifier issued by the indicated party.

XML Instance Representation

```
<...
id=" xsd:ID [0..1]">
   <partyReference> PartyReference </partyReference> [1]
   'A pointer style reference to a party identifier defined elsewhere in the document. The
   party referenced has allocated the trade identifier.'

Start Choice [1..*]
      <tradeId> TradeId </tradeId> [1]
      <versionedTradeId> VersionedTradeId </versionedTradeId> [1]
End Choice
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="TradeIdentifier">
   <xsd:sequence>
      <xsd:element name="partyReference" type=" PartyReference "/>
      <xsd:choice maxOccurs="unbounded">
         <xsd:element name="tradeId" type=" TradeId "/>
         <xsd:element name="versionedTradeId" type=" VersionedTradeId "/>
      </xsd:choice>
   </xsd:sequence>
   <xsd:attribute name="id" type=" xsd:ID "/>
</xsd:complexType>
```

top

Complex Type: TradeSide

Super-types:	None
Sub-types:	None

Name	TradeSide
Used by (from the same schema document)	Complex Type **Trade**
Abstract	no
Documentation	The parties to the trade form into sides. Each side has defined roles in the lifecyle of the trade fulfilled by parties. Each party role is given in the likely order they would be filled during the lifecycle of a trade.

XML Instance Representation

```
<...
id=" xsd:ID [1]">
   <orderer> PartyRole </orderer> [0..1]
   'The Party placing the order. This could be a fund manager acting on behalf of a client, or
   a hedge fund acting on it\'s own behalf. This is the role with the investment discretion.'

   <introducer> PartyRole </introducer> [0..1]
   'Party that can relay an order directly to the trading floor at a firm. This is potentially
   a different firm, but may be the same as that taking the order. In effect the introducer is
   the first dealer to take the order. The reason an introducing dealer may forward a trade
   is sometime because it doesn\'t have the capacity to execute effectively but does have
   the relationship with the Orderer. Introducing Party is an industry standard term. This
   is semantically equivalent to the FIX and ISO20022 Introducing Firm.'

   <executor> PartyRole </executor> [0..1]
   'The Party executing or striking the trade. Executing Party is an industry standard term.
   This is semantically equivalent to the FIX and ISO20022 Executing Firm or Trader.'

   <confirmer> PartyRole </confirmer> [0..1]
   'The party that undertakes the confirmation process for this Trade Side. The
```

confirmer essentially manages the matching and affirmation of trades. This is often the creditor or is increasingly outsourced to service providers such as Swapswire.'

<creditor> PartyRole </creditor> [1]

'The party whose name appears on the contract as being responsible for credit of the trade. This is the party in the Trade Side the credit risk is against. For example if a hedge fund was to trade in the name of it\'s prime broker, then the prime broker would be the creditor.'

<calculater> PartyRole </calculater> [0..1]

'The calculater is the Party that calculates, negotiates, and agrees the values to be paid at each payment date.'

<settler> PartyRole </settler> [0..1]

'The Settler is the party that makes the payments. Increasingly this is a service that can be externalized from the other roles. An example of a settlement service provide is SwapClear.'

<beneficiary> PartyRole </beneficiary> [0..1]

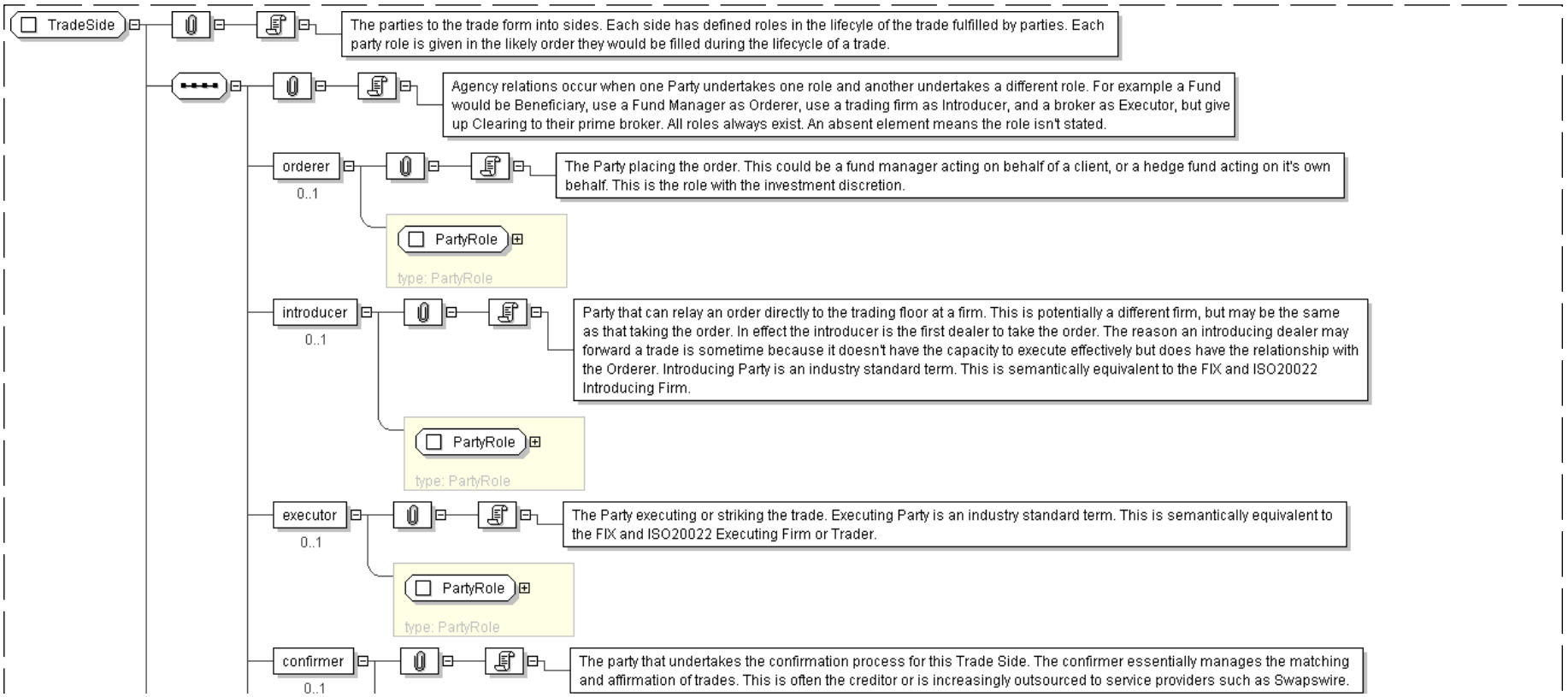
'The party that suffers the economic effect of the trade. This is usually referred to as the primary Principal in FIX and ISO20022 - which is slightly confusing in that there are potentially many Princiapal/Agency relationships. The beneficiary may be distinct from the creditor - an example is a Hedge Fund trading in the name of it\'s Prime Broker.'

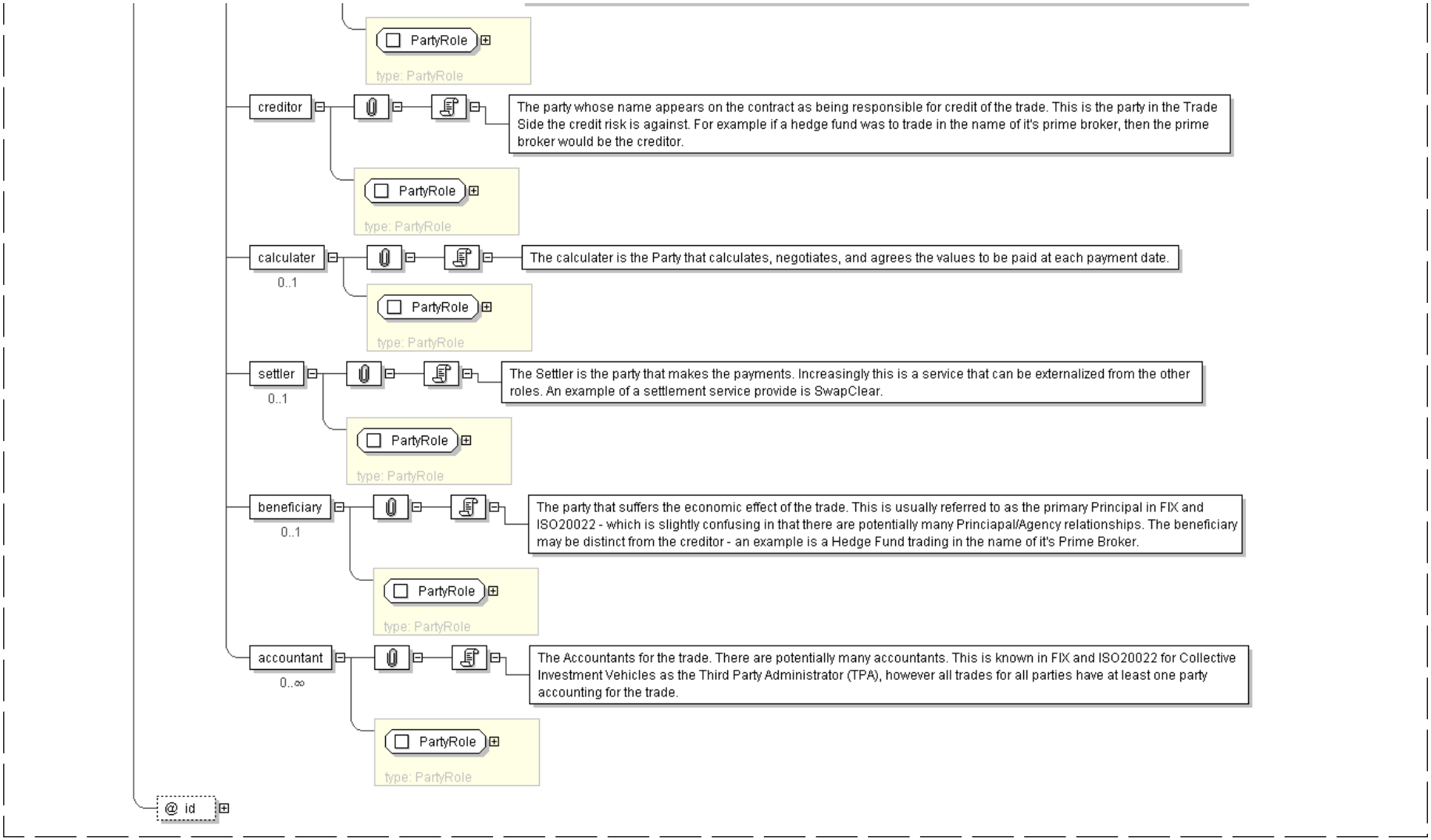
<accountant> PartyRole </accountant> [0..*]

'The Accountants for the trade. There are potentially many accountants. This is known in FIX and ISO20022 for Collective Investment Vehicles as the Third Party Administrator (TPA), however all trades for all parties have at least one party accounting for the trade.'

</...>

Diagram





Schema Component Representation

```
<xsd:complexType name="TradeSide">
   <xsd:sequence>
      <xsd:element name="orderer" type=" PartyRole " minOccurs="0"/>
      <xsd:element name="introducer" type=" PartyRole " minOccurs="0"/>
      <xsd:element name="executor" type=" PartyRole " minOccurs="0"/>
      <xsd:element name="confirmer" type=" PartyRole " minOccurs="0"/>
      <xsd:element name="creditor" type=" PartyRole "/>
      <xsd:element name="calculater" type=" PartyRole " minOccurs="0"/>
      <xsd:element name="settler" type=" PartyRole " minOccurs="0"/>
      <xsd:element name="beneficiary" type=" PartyRole " minOccurs="0"/>
      <xsd:element name="accountant" type=" PartyRole " minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="id" type=" xsd:ID " use="required"/>
</xsd:complexType>
```

top

Complex Type: Trader

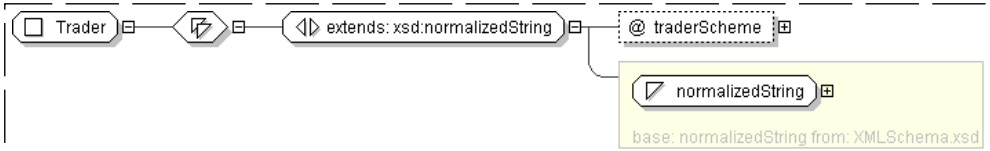
Super-types:	xsd:normalizedString < **Trader** (by extension)
Sub-types:	None

Name	Trader
Used by (from the same schema document)	Complex Type **PartyTradeInformation**
Abstract	no

XML Instance Representation

```
<...
traderScheme=" xsd:anyURI [0..1]">
xsd:normalizedString
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Trader">
   <xsd:simpleContent>
      <xsd:extension base=" xsd:normalizedString ">
         <xsd:attribute name="traderScheme" type=" xsd:anyURI " use="optional"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: Validation

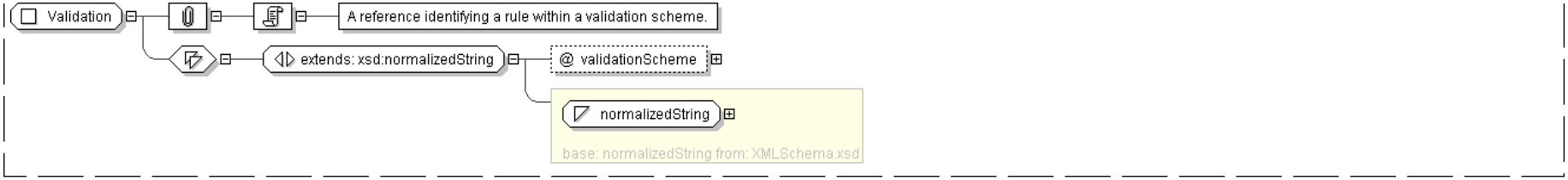
Super-types:	xsd:normalizedString < **Validation** (by extension)
Sub-types:	None

Name	Validation
Used by (from the same schema document)	Model Group **Validation.model**
Abstract	no
Documentation	A reference identifying a rule within a validation scheme.

XML Instance Representation

```
<...
validationScheme=" xsd:anyURI [0..1]">
xsd:normalizedString
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="Validation">
   <xsd:simpleContent>
      <xsd:extension base=" xsd:normalizedString ">
         <xsd:attribute name="validationScheme" type=" xsd:anyURI "/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

top

Complex Type: VersionedContractId

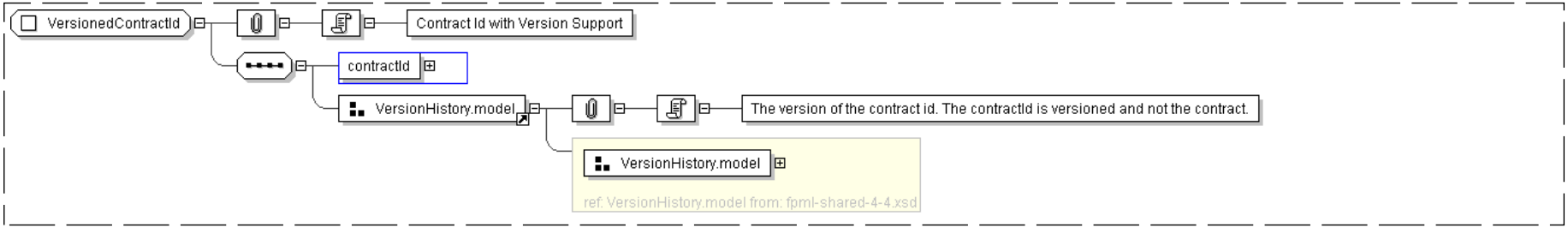
Super-types:	None
Sub-types:	None

Name	VersionedContractId
Used by (from the same schema document)	Complex Type **ContractIdentifier**
Abstract	no
Documentation	Contract Id with Version Support

XML Instance Representation

```
<...>
   <contractId> ContractId </contractId> [1]
   <version> xsd:nonNegativeInteger </version> [1]
   'The version number'

   <effectiveDate> IdentifiedDate </effectiveDate> [0..1]
   'Optionally it is possible to specify a version effective date when a versionId is supplied.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="VersionedContractId">
   <xsd:sequence>
      <xsd:element name="contractId" type=" ContractId "/>
      <xsd:group ref=" VersionHistory.model "/>
   </xsd:sequence>
</xsd:complexType>
```

top

Complex Type: VersionedTradeId

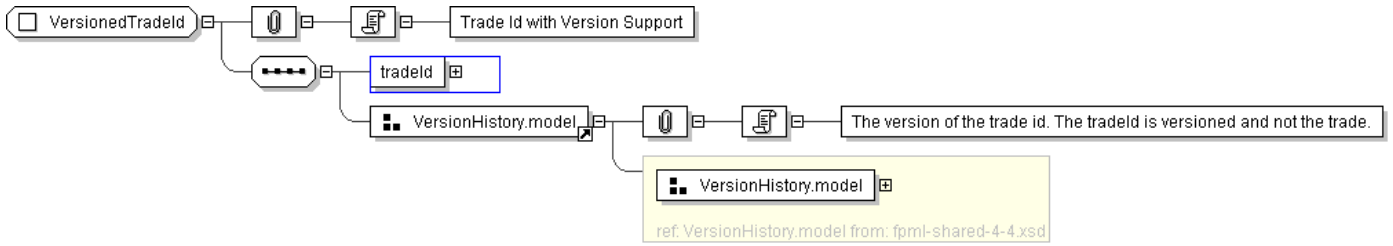
Super-types:	None
Sub-types:	None

Name	VersionedTradeId
Used by (from the same schema document)	Complex Type **TradeIdentifier**
Abstract	no
Documentation	Trade Id with Version Support

XML Instance Representation

```
<...>
   <tradeId> TradeId </tradeId> [1]
   <version> xsd:nonNegativeInteger </version> [1]
   'The version number'

   <effectiveDate> IdentifiedDate </effectiveDate> [0..1]
   'Optionally it is possible to specify a version effective date when a versionId is supplied.'

</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="VersionedTradeId">
   <xsd:sequence>
      <xsd:element name="tradeId" type=" TradeId "/>
      <xsd:group ref=" VersionHistory.model "/>
   </xsd:sequence>
</xsd:complexType>
```

top

Model Group: AccountReferenceOrPartyReference.model

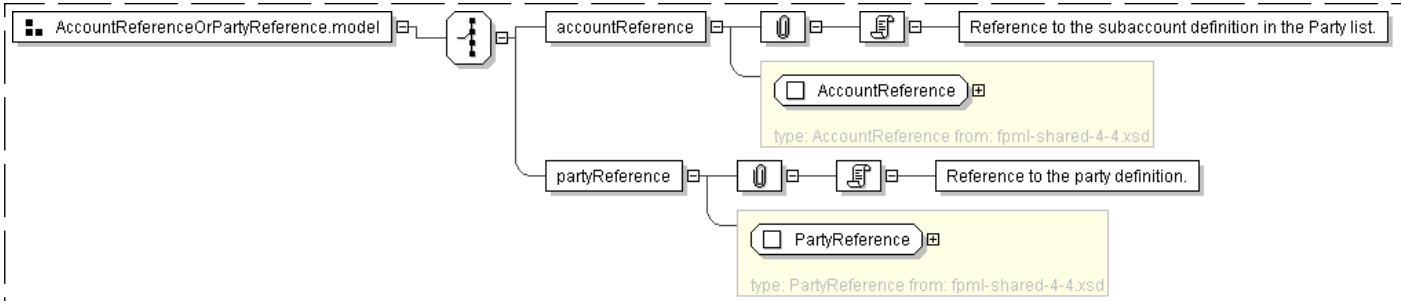
Name	AccountReferenceOrPartyReference.model
Used by (from the same schema document)	Complex Type **Allocation**

XML Instance Representation

```
Start Choice [1]
   <accountReference> AccountReference </accountReference> [1]
   'Reference to the subaccount definition in the Party list.'

   <partyReference> PartyReference </partyReference> [1]
   'Reference to the party definition.'
```

End Choice

Diagram



Schema Component Representation

```
<xsd:group name="AccountReferenceOrPartyReference.model">
   <xsd:choice>
      <xsd:element name="accountReference" type=" AccountReference "/>
      <xsd:element name="partyReference" type=" PartyReference "/>
   </xsd:choice>
</xsd:group>
```

top

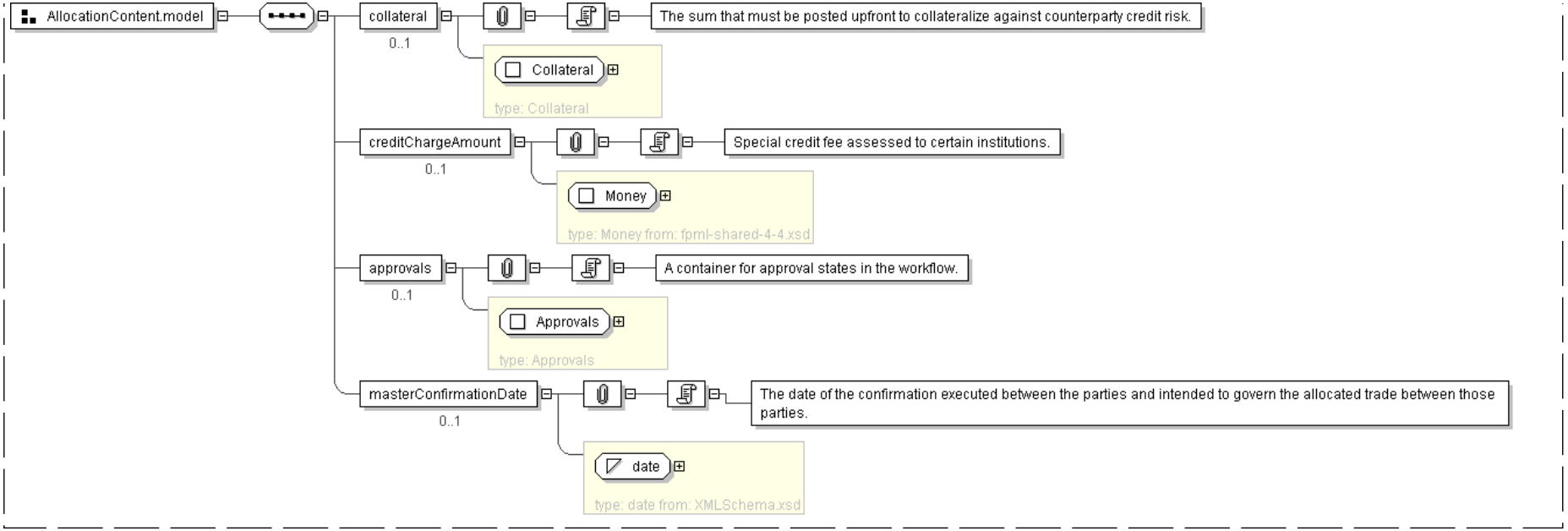
Model Group: AllocationContent.model

Name	AllocationContent.model
Used by (from the same schema document)	Complex Type **Allocation**

XML Instance Representation

```
<collateral> Collateral </collateral> [0..1]
'The sum that must be posted upfront to collateralize against counterparty credit risk.'

<creditChargeAmount> Money </creditChargeAmount> [0..1]
'Special credit fee assessed to certain institutions.'

<approvals> Approvals </approvals> [0..1]
'A container for approval states in the workflow.'

<masterConfirmationDate> xsd:date </masterConfirmationDate> [0..1]
'The date of the confirmation executed between the parties and intended to govern the
allocated trade between those parties.'
```

Diagram



Schema Component Representation

```
<xsd:group name="AllocationContent.model">
   <xsd:sequence>
      <xsd:element name="collateral" type=" Collateral " minOccurs="0"/>
      <xsd:element name="creditChargeAmount" type=" Money " minOccurs="0"/>
      <xsd:element name="approvals" type=" Approvals " minOccurs="0"/>
      <xsd:element name="masterConfirmationDate" type=" xsd:date " minOccurs="0"/>
   </xsd:sequence>
</xsd:group>
```

top

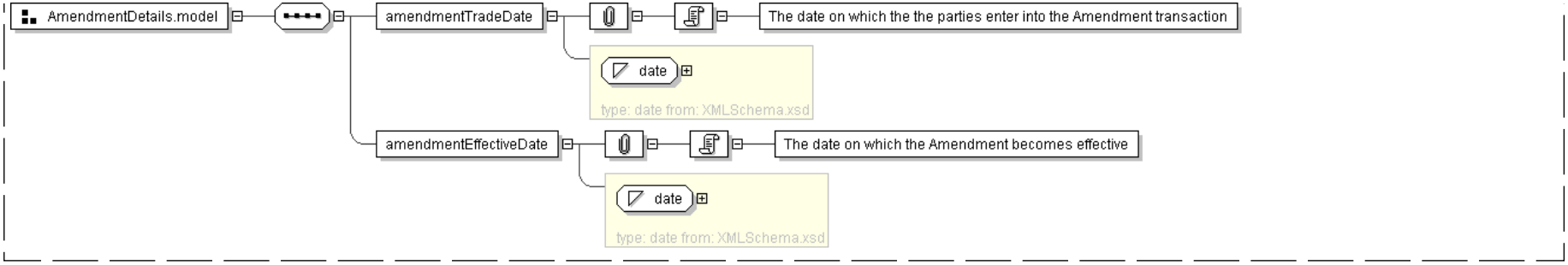
Model Group: AmendmentDetails.model

Name	AmendmentDetails.model
Used by (from the same schema document)	Complex Type **Amendment**

XML Instance Representation

```
<amendmentTradeDate> xsd:date </amendmentTradeDate> [1]
'The date on which the the parties enter into the Amendment transaction'

<amendmentEffectiveDate> xsd:date </amendmentEffectiveDate> [1]
'The date on which the Amendment becomes effective'
```

Diagram



Schema Component Representation

```
<xsd:group name="AmendmentDetails.model">
   <xsd:sequence>
      <xsd:element name="amendmentTradeDate" type=" xsd:date "/>
      <xsd:element name="amendmentEffectiveDate" type=" xsd:date "/>
   </xsd:sequence>
</xsd:group>
```

top

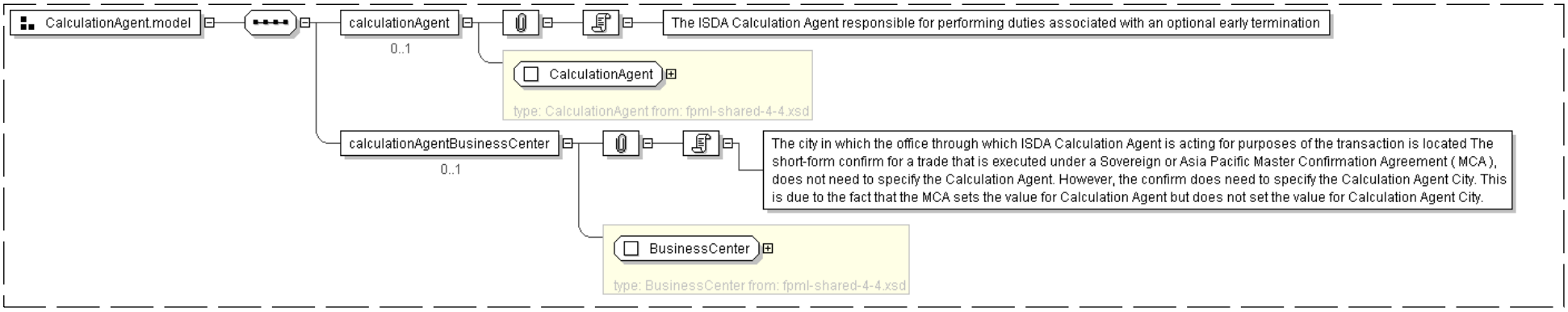
Model Group: CalculationAgent.model

Name	CalculationAgent.model
Used by (from the same schema document)	Complex Type **Contract** , Complex Type **Trade**

XML Instance Representation

```
<calculationAgent> CalculationAgent </calculationAgent> [0..1]
 'The ISDA Calculation Agent responsible for performing duties associated with an optional
 early termination'

<calculationAgentBusinessCenter> BusinessCenter </calculationAgentBusinessCenter> [0..1]
 'The city in which the office through which ISDA Calculation Agent is acting for purposes
 of the transaction is located The short-form confirm for a trade that is executed under
 a Sovereign or Asia Pacific Master Confirmation Agreement ( MCA ), does not need to specify
 the Calculation Agent. However, the confirm does need to specify the Calculation Agent
 City. This is due to the fact that the MCA sets the value for Calculation Agent but does
 not set the value for Calculation Agent City.'
```

Diagram



Schema Component Representation

```
<xsd:group name="CalculationAgent.model">
```

```
    <xsd:sequence>
       <xsd:element name="calculationAgent" type=" CalculationAgent " minOccurs="0"/>
       <xsd:element name="calculationAgentBusinessCenter" type=" BusinessCenter " minOccurs="0"/>
    </xsd:sequence>
</xsd:group>
```

top

Model Group: ContractNovationDetails.model

Name	ContractNovationDetails.model
Used by (from the same schema document)	Complex Type **ContractNovation**
Documentation	Model group with Contract Novation element content.

XML Instance Representation

```
Start Choice [1]
Start Choice [1]
'Choice between identification and representation of the new contract.'

      <newContractReference> ContractReference </newContractReference> [1]
      'Indicates a reference to the new Contract between the transferee and the remaining party.'

      <newContract> Contract </newContract> [1]
      'Indicates the new Contract between the transferee and the remaining party.'

End Choice
Start Choice [1]
      <oldContractReference> ContractReference </oldContractReference> [1]
      'Indicates a reference to the original contract between the transferor and the remaining party.'

      <oldContract> Contract </oldContract> [1]
      'Indicates the original Contract between the transferor and the remaining party.'

End Choice
Start Choice [0..1]
'Choice between identification and representation of the new contract.'

      <newContractReference> ContractReference </newContractReference> [1]
      <newContract> Contract </newContract> [1]
End Choice
End Choice
<transferor> PartyReference </transferor> [1]
'A pointer style reference to a party identifier defined elsewhere in the document. In a
three-way novation the party referenced is the Transferor (outgoing party) in the novation.
The Transferor means a party which transfers by novation to a Transferee all of its
rights, liabilities, duties and obligations with respect to a Remaining Party. In a four-
way novation the party referenced is Transferor 1 which transfers by novation to Transferee
1 all of its rights, liabilities, duties and obligations with respect to Transferor 2.
ISDA 2004 Novation Term: Transferor (three-way novation) or Transferor 1 (four-way novation).'

<transferee> PartyReference </transferee> [1]
'A pointer style reference to a party identifier defined elsewhere in the document. In a
three-way novation the party referenced is the Transferee (incoming party) in the
novation. Transferee means a party which accepts by way of novation all rights,
liabilities, duties and obligations of a Transferor with respect to a Remaining Party. In
a four-way novation the party referenced is Transferee 1 which accepts by way of novation
the rights, liabilities, duties and obligations of Transferor 1. ISDA 2004 Novation
Term: Transferee (three-way novation) or Transferee 1 (four-way novation).'

<remainingParty> PartyReference </remainingParty> [1]
'A pointer style reference to a party identifier defined elsewhere in the document. In a
three-way novation the party referenced is the Remaining Party in the novation. Remaining
```

Party means a party which consents to a Transferor\'s transfer by novation and the acceptance thereof by the Transferee of all of the Transferor\'s rights, liabilities, duties and obligations with respect to such Remaining Party under and with respect of the Novated Amount of a transaction. In a four-way novation the party referenced is Transferor 2 per the ISDA definition and acts in the role of a Transferor. Transferor 2 transfers by novation to Transferee 2 all of its rights, liabilities, duties and obligations with respect to Transferor 1. ISDA 2004 Novation Term: Remaining Party (three-way novation) or Transferor 2 (four-way novation).'

<otherRemainingParty> PartyReference </otherRemainingParty> [0..1]

'A pointer style reference to a party identifier defined elsewhere in the document. This element is not applicable in a three-way novation and should be omitted. In a four-way novation the party referenced is Transferee 2. Transferee 2 means a party which accepts by way of novation the rights, liabilities, duties and obligations of Transferor 2. ISDA 2004 Novation Term: Transferee 2 (four-way novation).'

<novationDate> xsd:date </novationDate> [1]

'Specifies the date that one party\'s legal obligations with regard to a trade are transferred to another party. It corresponds to the Novation Date section of the 2004 ISDA Novation Definitions, section 1.16.'

<novationContractDate> xsd:date </novationContractDate> [0..1]

'Specifies the date the parties agree to assign or novate a Contract. If this element is not specified, the novationContractDate will be deemed to be the novationDate. It corresponds to the Novation Trade Date section of the 2004 ISDA Novation Definitions, section 1.17.'

Start Choice [1]

'Choice for expressing the novated amount as either a money amount, number of options, or number of units, according the the financial product which is being novated.'

<novatedAmount> Money </novatedAmount> [1]

'The amount which represents the portion of the Old Contract being novated.'

<novatedNumberOfOptions> xsd:decimal </novatedNumberOfOptions> [1]

'The number of options which represent the portion of the Old Contract being novated.'

<novatedNumberOfUnits> xsd:decimal </novatedNumberOfUnits> [1]

'The number of options which represent the portion of the Old Contract being novated.'

End Choice

<fullFirstCalculationPeriod> xsd:boolean </fullFirstCalculationPeriod> [0..1]

'This element corresponds to the applicability of the Full First Calculation Period as defined in the 2004 ISDA Novation Definitions, section 1.20.'

<firstPeriodStartDate> FirstPeriodStartDate </firstPeriodStartDate> [0..2]

'Element that is used to be able to make sense of the “new transaction” without requiring reference back to the “old transaction”. In the case of interest rate products there are potentially 2 “first period start dates” to reference – one with respect to each party to the new transaction. For Credit Default Swaps there is just the one with respect to the party that is the fixed rate payer.'

<nonReliance> Empty </nonReliance> [0..1]

'This element corresponds to the non-Reliance section in the 2004 ISDA Novation Definitions, section 2.1 (c) (i). The element appears in the instance document when non-Reliance is applicable.'

<creditDerivativesNotices> CreditDerivativesNotices </creditDerivativesNotices> [0..1]

'This element should be specified if one or more of either a Credit Event Notice, Notice of Publicly Available Information, Notice of Physical Settlement or Notice of Intended Physical Settlement, as applicable, has been delivered by or to the Transferor or the Remaining Party. The type of notice or notices that have been delivered should be indicated by setting the relevant boolean element value(s) to true. The absence of the element means that no Credit Event Notice, Notice of Publicly Available Information, Notice of Physical Settlement or Notice of Intended Physical Settlement, as applicable, has been delivered by or to the Transferor or the Remaining Party.'

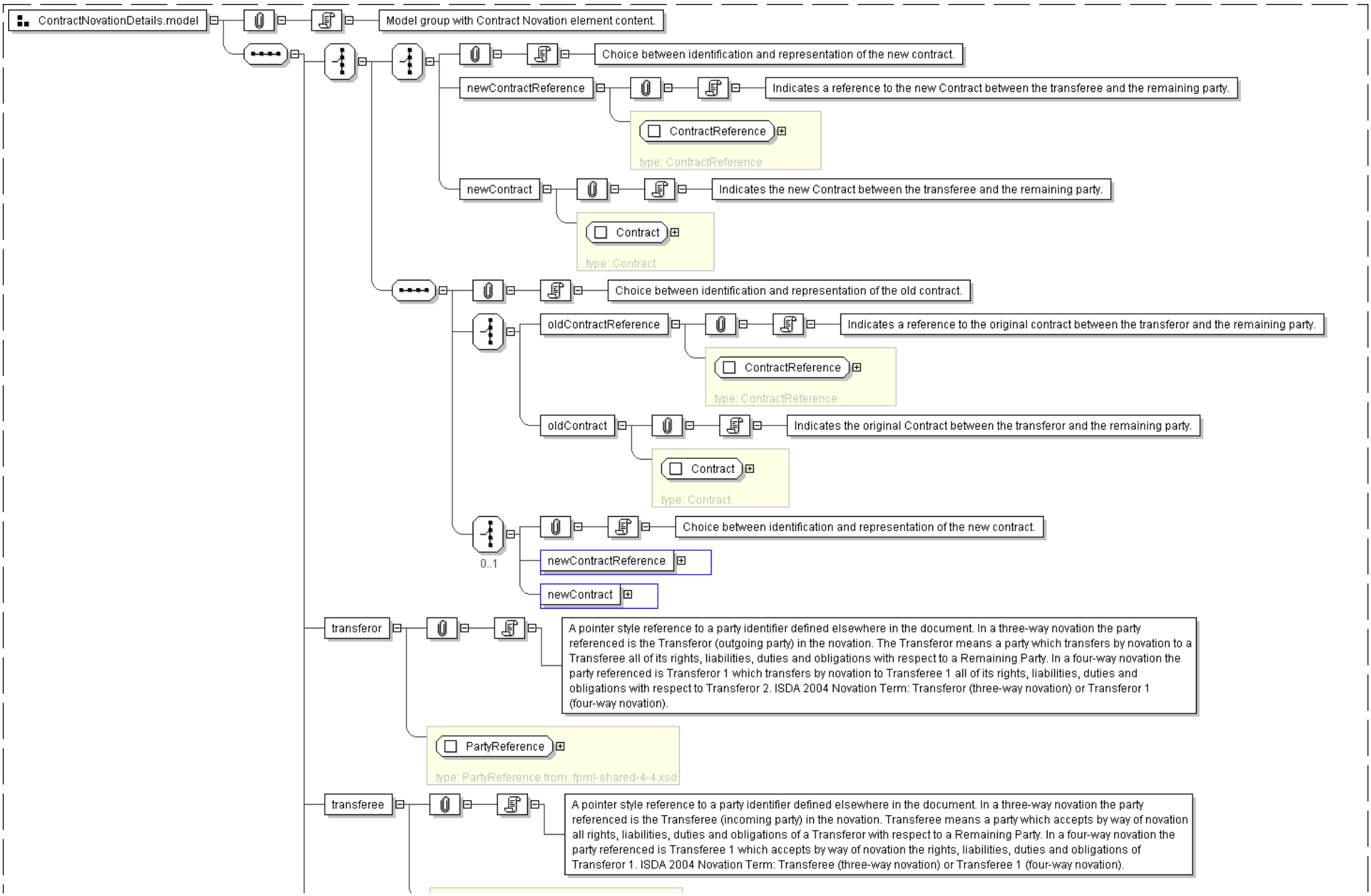
<contractualDefinitions> ContractualDefinitions </contractualDefinitions> [0..*]

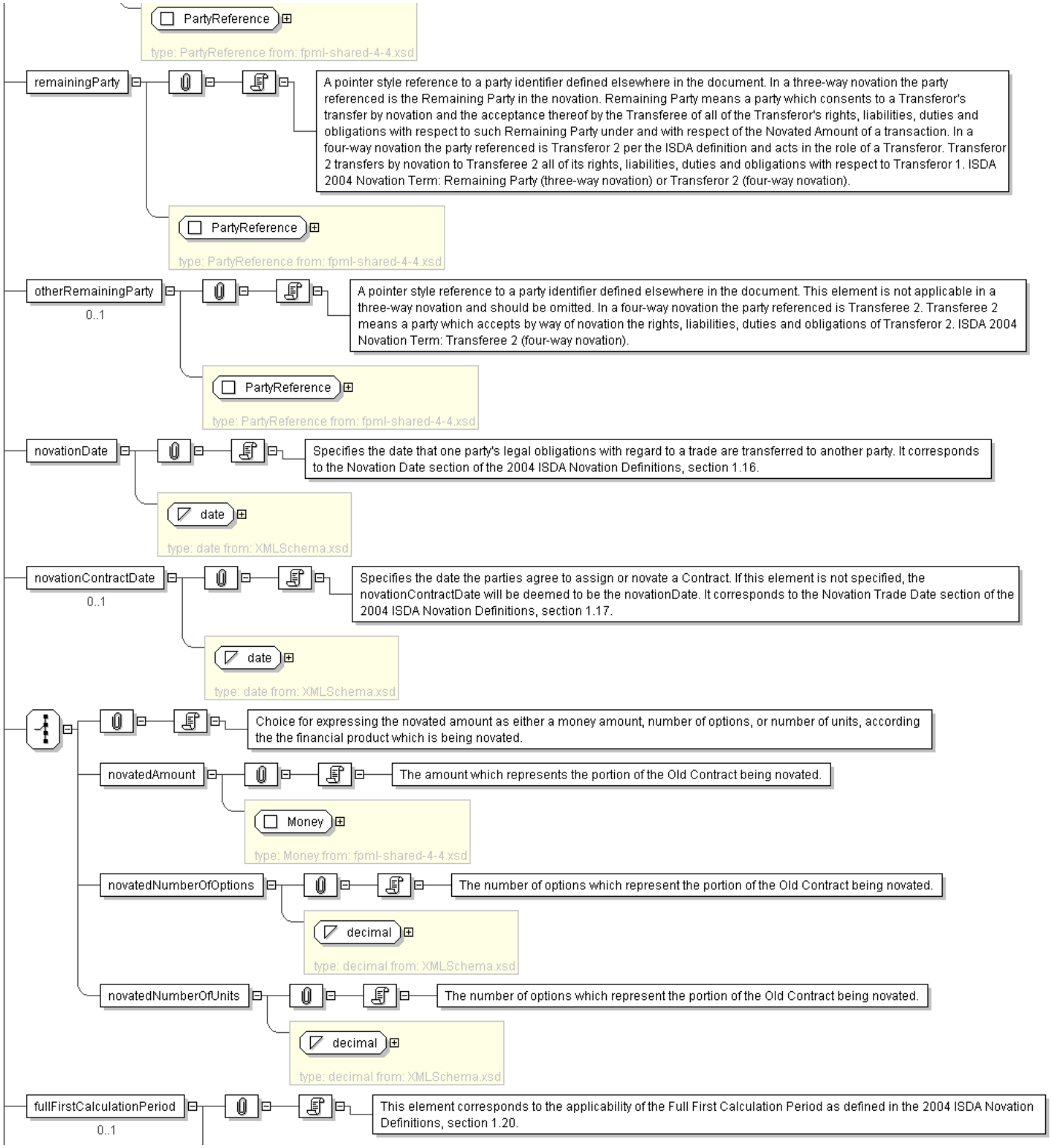
'The definitions (such as those published by ISDA) that will define the terms of the novation transaction.'

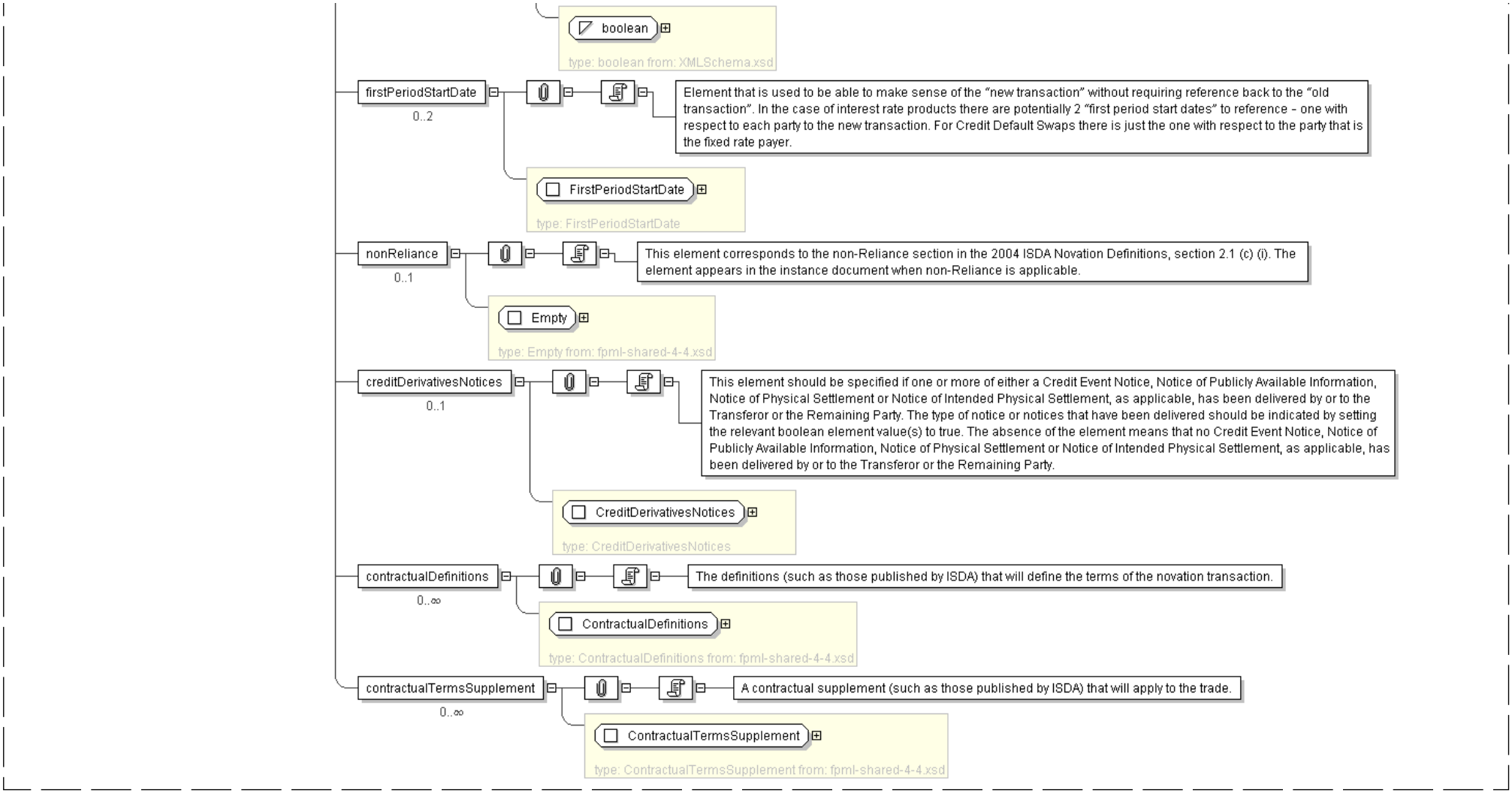
<contractualTermsSupplement> ContractualTermsSupplement </contractualTermsSupplement> [0..*]

'A contractual supplement (such as those published by ISDA) that will apply to the trade.'

Diagram




PartyReference
type: PartyReference from: fpml-shared-4-4.xsd
remainingParty
A pointer style reference to a party identifier defined elsewhere in the document. In a three-way novation the party referenced is the Remaining Party in the novation. Remaining Party means a party which consents to a Transferor's transfer by novation and the acceptance thereof by the Transferee of all of the Transferor's rights, liabilities, duties and obligations with respect to such Remaining Party under and with respect of the Novated Amount of a transaction. In a four-way novation the party referenced is Transferor 2 per the ISDA definition and acts in the role of a Transferor. Transferor 2 transfers by novation to Transferee 2 all of its rights, liabilities, duties and obligations with respect to Transferor 1. ISDA 2004 Novation Term: Remaining Party (three-way novation) or Transferor 2 (four-way novation).
PartyReference
type: PartyReference from: fpml-shared-4-4.xsd
otherRemainingParty
0..1
A pointer style reference to a party identifier defined elsewhere in the document. This element is not applicable in a three-way novation and should be omitted. In a four-way novation the party referenced is Transferee 2. Transferee 2 means a party which accepts by way of novation the rights, liabilities, duties and obligations of Transferor 2. ISDA 2004 Novation Term: Transferee 2 (four-way novation).
PartyReference
type: PartyReference from: fpml-shared-4-4.xsd
novationDate
Specifies the date that one party's legal obligations with regard to a trade are transferred to another party. It corresponds to the Novation Date section of the 2004 ISDA Novation Definitions, section 1.16.
date
type: date from: XMLSchema.xsd
novationContractDate
0..1
Specifies the date the parties agree to assign or novate a Contract. If this element is not specified, the novationContractDate will be deemed to be the novationDate. It corresponds to the Novation Trade Date section of the 2004 ISDA Novation Definitions, section 1.17.
date
type: date from: XMLSchema.xsd
Choice for expressing the novated amount as either a money amount, number of options, or number of units, according the the financial product which is being novated.
novatedAmount
The amount which represents the portion of the Old Contract being novated.
Money
type: Money from: fpml-shared-4-4.xsd
novatedNumberOfOptions
The number of options which represent the portion of the Old Contract being novated.
decimal
type: decimal from: XMLSchema.xsd
novatedNumberOfUnits
The number of options which represent the portion of the Old Contract being novated.
decimal
type: decimal from: XMLSchema.xsd
fullFirstCalculationPeriod
0..1
This element corresponds to the applicability of the Full First Calculation Period as defined in the 2004 ISDA Novation Definitions, section 1.20.



Schema Component Representation

```
<xsd:group name="ContractNovationDetails.model">
   <xsd:sequence>
      <xsd:choice>
         <xsd:choice>
            <xsd:element name="newContractReference" type=" ContractReference "/>
            <xsd:element name="newContract" type=" Contract "/>
         </xsd:choice>
         <xsd:sequence>
            <xsd:choice>
               <xsd:element name="oldContractReference" type=" ContractReference "/>
               <xsd:element name="oldContract" type=" Contract "/>
            </xsd:choice>
            <xsd:choice minOccurs="0">
               <xsd:element name="newContractReference" type=" ContractReference "/>
               <xsd:element name="newContract" type=" Contract "/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:choice>
      <xsd:element name="transferor" type=" PartyReference "/>
```

```
      <xsd:element name="transferee" type=" PartyReference "/>
      <xsd:element name="remainingParty" type=" PartyReference "/>
      <xsd:element name="otherRemainingParty" type=" PartyReference " minOccurs="0"/>
      <xsd:element name="novationDate" type=" xsd:date "/>
      <xsd:element name="novationContractDate" type=" xsd:date " minOccurs="0"/>
      <xsd:choice>
         <xsd:element name="novatedAmount" type=" Money "/>
         <xsd:element name="novatedNumberOfOptions" type=" xsd:decimal "/>
         <xsd:element name="novatedNumberOfUnits" type=" xsd:decimal "/>
      </xsd:choice>
      <xsd:element name="fullFirstCalculationPeriod" type=" xsd:boolean " minOccurs="0"/>
      <xsd:element name="firstPeriodStartDate" type=" FirstPeriodStartDate "
      minOccurs="0" maxOccurs="2"/>
      <xsd:element name="nonReliance" type=" Empty " minOccurs="0"/>
      <xsd:element name="creditDerivativesNotices" type=" CreditDerivativesNotices " minOccurs="0"/>
      <xsd:element name="contractualDefinitions" type=" ContractualDefinitions "
      minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="contractualTermsSupplement" type=" ContractualTermsSupplement
      " minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:group>
```

top

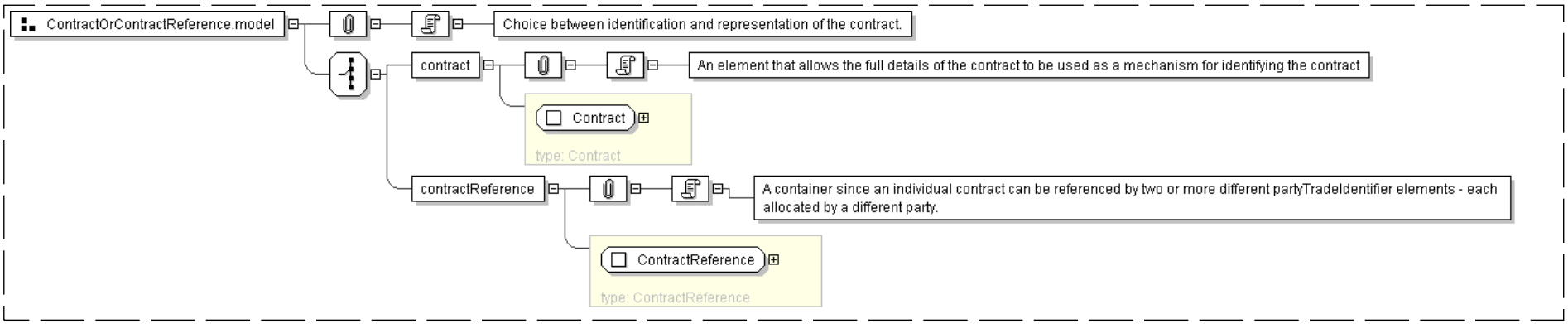
Model Group: ContractOrContractReference.model

Name	ContractOrContractReference.model
Documentation	Choice between identification and representation of the contract.

XML Instance Representation

```
Start Choice [1]
   <contract> Contract </contract> [1]
   'An element that allows the full details of the contract to be used as a mechanism
   for identifying the contract'

   <contractReference> ContractReference </contractReference> [1]
   'A container since an individual contract can be referenced by two or more
   different partyTradeIdentifier elements - each allocated by a different party.'

End Choice
```

Diagram



Schema Component Representation

```
<xsd:group name="ContractOrContractReference.model">
   <xsd:choice>
      <xsd:element name="contract" type=" Contract "/>
```

```
      <xsd:element name="contractReference" type=" ContractReference "/>
   </xsd:choice>
</xsd:group>
```

top

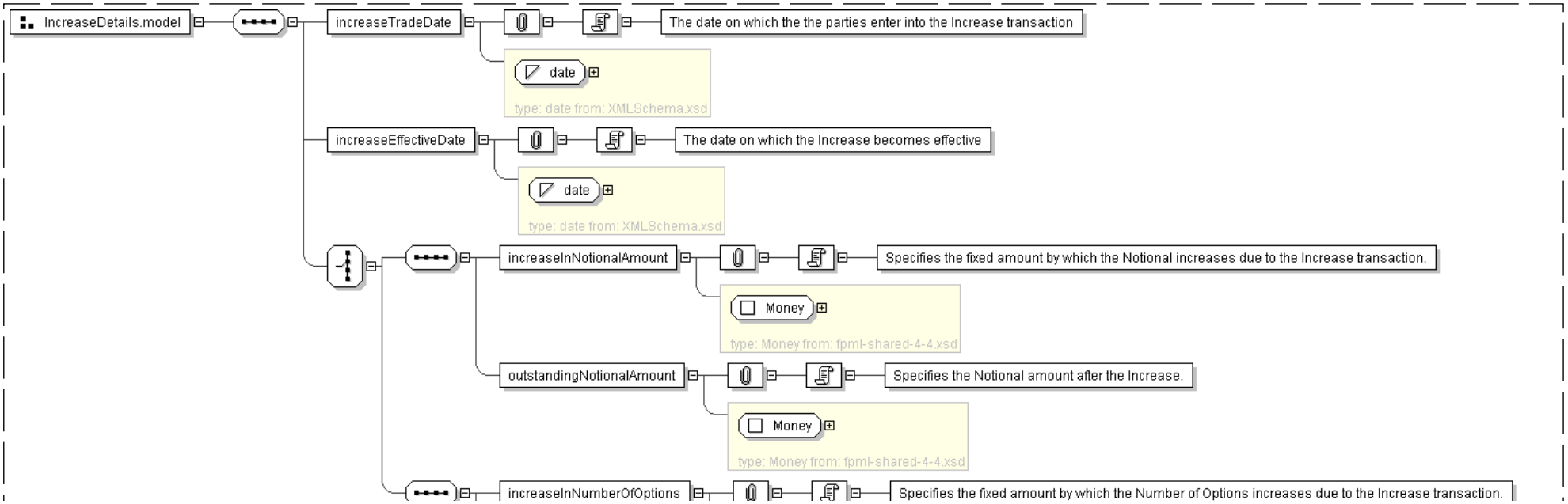
Model Group: IncreaseDetails.model

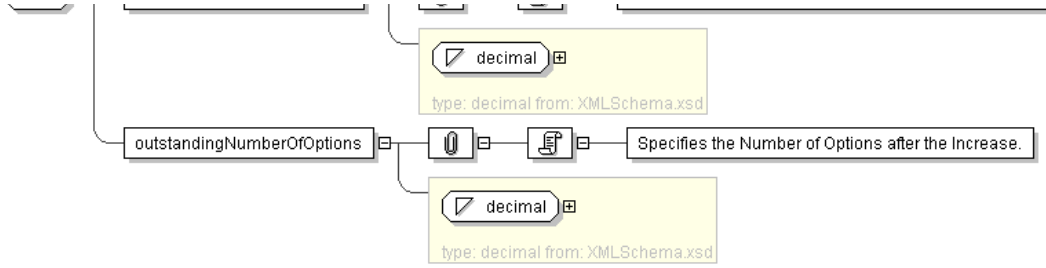
Name	IncreaseDetails.model
Used by (from the same schema document)	Complex Type **Increase**

XML Instance Representation

```
<increaseTradeDate> xsd:date </increaseTradeDate> [1]
 'The date on which the the parties enter into the Increase transaction'

<increaseEffectiveDate> xsd:date </increaseEffectiveDate> [1]
 'The date on which the Increase becomes effective'

Start Choice [1]
   <increaseInNotionalAmount> Money </increaseInNotionalAmount> [1]
   'Specifies the fixed amount by which the Notional increases due to the Increase transaction.'

   <outstandingNotionalAmount> Money </outstandingNotionalAmount> [1]
   'Specifies the Notional amount after the Increase.'

   <increaseInNumberOfOptions> xsd:decimal </increaseInNumberOfOptions> [1]
   'Specifies the fixed amount by which the Number of Options increases due to the
   Increase transaction.'

   <outstandingNumberOfOptions> xsd:decimal </outstandingNumberOfOptions> [1]
   'Specifies the Number of Options after the Increase.'

End Choice
```

Diagram





Schema Component Representation

```
<xsd:group name="IncreaseDetails.model">
   <xsd:sequence>
      <xsd:element name="increaseTradeDate" type=" xsd:date "/>
      <xsd:element name="increaseEffectiveDate" type=" xsd:date "/>
      <xsd:choice>
         <xsd:sequence>
            <xsd:element name="increaseInNotionalAmount" type=" Money "/>
            <xsd:element name="outstandingNotionalAmount" type=" Money "/>
         </xsd:sequence>
         <xsd:sequence>
            <xsd:element name="increaseInNumberOfOptions" type=" xsd:decimal "/>
            <xsd:element name="outstandingNumberOfOptions" type=" xsd:decimal "/>
         </xsd:sequence>
      </xsd:choice>
   </xsd:sequence>
</xsd:group>
```

top

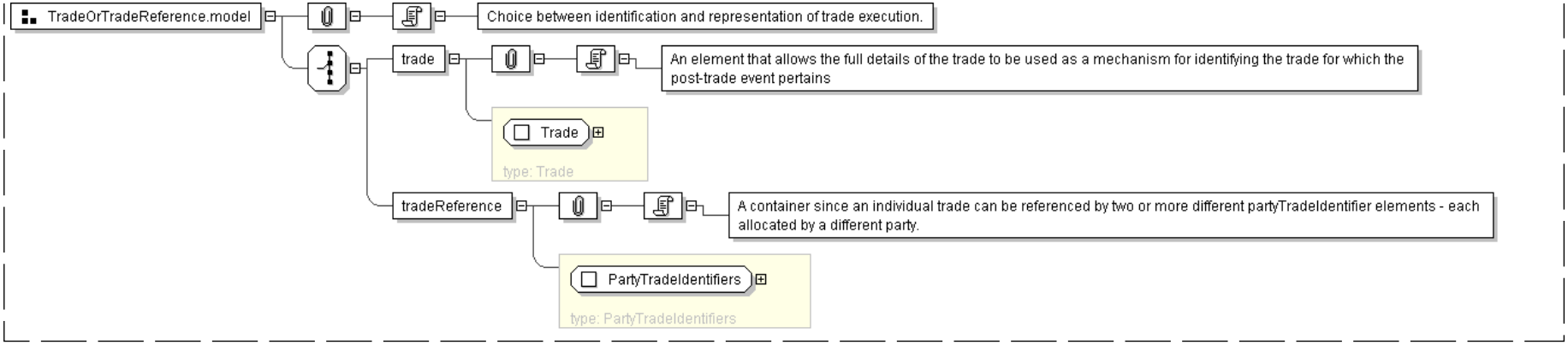
Model Group: TradeOrTradeReference.model

Name	TradeOrTradeReference.model
Used by (from the same schema document)	Complex Type **Increase**
Documentation	Choice between identification and representation of trade execution.

XML Instance Representation

```
Start Choice [1]
   <trade> Trade </trade> [1]
   'An element that allows the full details of the trade to be used as a mechanism for
   identifying the trade for which the post-trade event pertains'

   <tradeReference> PartyTradeIdentifiers </tradeReference> [1]
   'A container since an individual trade can be referenced by two or more
   different partyTradeIdentifier elements - each allocated by a different party.'

End Choice
```

Diagram



Schema Component Representation

```
<xsd:group name="TradeOrTradeReference.model">
   <xsd:choice>
      <xsd:element name="trade" type=" Trade "/>
      <xsd:element name="tradeReference" type=" PartyTradeIdentifiers "/>
   </xsd:choice>
</xsd:group>
```

top

Model Group: Validation.model

Name	Validation.model
Used by (from the same schema document)	Complex Type **DataDocument**

XML Instance Representation

```
<validation> Validation </validation> [0..*]
```

Diagram

Schema Component Representation

```
<xsd:group name="Validation.model">
   <xsd:sequence>
      <xsd:element name="validation" type=" Validation " minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:group>
```

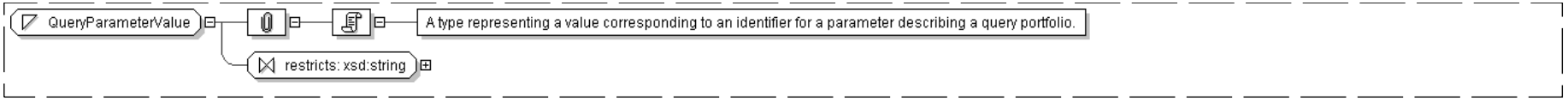
top

Simple Type: QueryParameterValue

Super-types:	xsd:string < **QueryParameterValue** (by restriction)
Sub-types:	None

Name	QueryParameterValue
Content	• Base XSD Type: string
Documentation	A type representing a value corresponding to an identifier for a parameter describing a query portfolio.

Diagram



Schema Component Representation

```
<xsd:simpleType name="QueryParameterValue">
   <xsd:restriction base=" xsd:string "/>
</xsd:simpleType>
```

top

Legend

Complex Type: **AusAddress**

Schema Component Type — Schema Component Name

Super-types:	Address < AusAddress (by extension)
Sub-types:	• QLDAddress (by restriction)

If this schema component is a type definition, its type hierarchy is shown in a gray-bordered box.

Name	AusAddress
Abstract	no

The table above displays the properties of this schema component.

XML Instance Representation

```
<... country="Australia" >
<unitNo> string </unitNo> [0..1]
<houseNo> string </houseNo> [1]
<street> string </street> [1]
Start Choice [1]
<city> string </city> [1]
<town> string </town> [1]
End Choice
<state> AusStates </state> [1]
<postcode> string <<pattern = [1-9][0-9]{3}>> </postcode> [1]
</...>
```

The XML Instance Representation table above shows the schema component's content as an XML instance.

- The minimum and maximum occurrence of elements and attributes are provided in square brackets, e.g. [0..1].
- Model group information are shown in gray, e.g. Start Choice ... End Choice.
- For type derivations, the elements and attributes that have been added to or changed from the base type's content are shown in **bold**.
- If an element/attribute has a fixed value, the fixed value is shown in green, e.g. country="Australia".
- Otherwise, the type of the element/attribute is displayed.
  - If the element/attribute's type is in the schema, a link is provided to it.
  - For local simple type definitions, the constraints are displayed in angle brackets, e.g. <<*pattern* = [1-9][0-9]{3}>>.

Schema Component Representation

```
<complexType name="AusAddress">
<complexContent>
<extension base=" Address ">
<sequence>
<element name="state" type=" AusStates "/>
<element name="postcode">
<simpleType>
<restriction base=" string ">
<pattern value="[1-9][0-9]{3}"/>
</restriction>
```

```
</simpleType>
</element>
</sequence>
<attribute name="country" type=" string " fixed="Australia"/>
</extension>
</complexContent>
</complexType>
```

The Schema Component Representation table above displays the underlying XML representation of the schema component. (Annotations are not shown.)

top

Glossary

Abstract (Applies to complex type definitions and element declarations). An abstract element or complex type cannot used to validate an element instance. If there is a reference to an abstract element, only element declarations that can substitute the abstract element can be used to validate the instance. For references to abstract type definitions, only derived types can be used.

All Model Group Child elements can be provided *in any order* in instances. See: http://www.w3.org/TR/xmlschema-1/#element-all.

Choice Model Group *Only one* from the list of child elements and model groups can be provided in instances. See: http://www.w3.org/TR/xmlschema-1/#element-choice.

Collapse Whitespace Policy Replace tab, line feed, and carriage return characters with space character (Unicode character 32). Then, collapse contiguous sequences of space characters into single space character, and remove leading and trailing space characters.

Disallowed Substitutions (Applies to element declarations). If *substitution* is specified, then substitution group members cannot be used in place of the given element declaration to validate element instances. If *derivation methods*, e.g. extension, restriction, are specified, then the given element declaration will not validate element instances that have types derived from the element declaration's type using the specified derivation methods. Normally, element instances can override their declaration's type by specifying an `xsi:type` attribute.

Key Constraint Like Uniqueness Constraint, but additionally requires that the specified value(s) must be provided. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Key Reference Constraint Ensures that the specified value(s) must match value(s) from a Key Constraint or Uniqueness Constraint. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Model Group Groups together element content, specifying the order in which the element content can occur and the number of times the group of element content may be repeated. See: http://www.w3.org/TR/xmlschema-1/#Model_Groups.

Nillable (Applies to element declarations). If an element declaration is nillable, instances can use the `xsi:nil` attribute. The `xsi:nil` attribute is the boolean attribute, *nil*, from the *http://www.w3.org/2001/XMLSchema-instance* namespace. If an element instance has an `xsi:nil` attribute set to true, it can be left empty, even though its element declaration may have required content.

Notation A notation is used to identify the format of a piece of data. Values of elements and attributes that are of type, NOTATION, must come from the names of declared notations. See: http://www.w3.org/TR/xmlschema-1/#cNotation_Declarations.

Preserve Whitespace Policy Preserve whitespaces exactly as they appear in instances.

Prohibited Derivations (Applies to type definitions). Derivation methods that cannot be used to create sub-types from a given type definition.

Prohibited Substitutions (Applies to complex type definitions). Prevents sub-types that have been derived using the specified derivation methods from validating element instances in place of the given type definition.

Replace Whitespace Policy Replace tab, line feed, and carriage return characters with space character (Unicode character 32).

Sequence Model Group Child elements and model groups must be provided *in the specified order* in instances. See: http://www.w3.org/TR/xmlschema-1/#element-sequence.

Substitution Group Elements that are *members* of a substitution group can be used wherever the *head* element of the substitution group is referenced.

Substitution Group Exclusions (Applies to element declarations). Prohibits element declarations from nominating themselves as being able to substitute a given element declaration, if they have types that are derived from the original element's type using the specified derivation methods.

Target Namespace The target namespace identifies the namespace that components in this schema belongs to. If no target namespace is provided, then the schema components do not belong to any namespace.

Uniqueness Constraint Ensures uniqueness of an element/attribute value, or a combination of values, within a specified scope. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

top

Generated by <oXygen/> XML Editor using a modified version of xs3p that adds schema diagrams and chunking support.