elements

Deals Package

TARMS Inc.

September 07, 2000

Typeset in LaTeX.

Contents

List of Figures

List of Tables

Package Description

A deal is used to record the details of financial transactions that have occurred between (and within) organizations. This package allows deals to be constructed to record the financial transactions of an organization.

Certain information is required about deals irrespective of the type of deal. This information is specified by the Deal class. The financial instrument within the deal records the commodities being exchanged as well as the amount and timing of these exchanges. Note: Commodities include the exchange of a currency.

To date we have specified the construction of the following types of deals: loans, FX, bonds, equities, FRAs, IR Swaps and standard (simple) REPOs. This package may be extended in the future to accommodate more deal types.

This package also includes classes used to enforce authorization of operations associated with deals. This links in with the permissions package.

1 Interfaces

1.1 Deal

A deal is a record of a trade that has occurred between a dealer and a counterparty where some commodity(s) has been bought or sold. To be able to audit deals done by an organization, a unique deal number and version number is associated with each deal. The financial details of a deal can be found within the instrument.

For simplicity, the Deal interface has been designed to inherit from the Deal Operation interface, thus enabling a deal instance to be used directly to determine whether it is a permissible operation. As a result the deal interface must respond to some methods that may not be relevant to a given deal. If one of the deal operation methods is not relevant to a particular deal then null is returned.

1.1.1 Relationships

	Class	Description	Notes
⇑	DynamicData		
↓	DealModel §2.1		

⇑:Inherits ↓:Realized by

1.1.2 Operations

Date dealDate() dealDate

The date on which a trade occurs (ie: the effective date of the deal).

Integer buyOrSell() buyOrSell

Enumeration {1, -1}. buyOrSell indicates whether the counterparty involved in the transaction is buying or selling the commodity. A value of +1 indicates that you are buying (thus the counterparty is selling), and a value of -1 indicates that you are selling (hence, the counterparty is buying).

String dealNumber() dealNumber

The alphanumeric string used to uniquely identify the trade.

If a trade is executed within the local domain, then just the dealNumber string will be returned. If a trade is executed outside the local domain, then the key and the domain name will be returned as the deal number.

User user() user

The user executing the trade.

Instrument instrument() instrument

This is the financial commodity being bought or sold. The instrument can be used to differentiate between deal types.

Book book() book

The book in which the trade is recorded.

Organization broker() broker

The broker involved in the trade (if applicable).

Organization clearingBroker() clearingBroker

The clearing broker involved in the trade (if applicable).

DealType dealType() dealType

This returns the type of deal that has been traded.

DealPurpose dealPurpose() dealPurpose

This returns the purpose of the deal that has been traded.

LifeCycleState lifeCycleState() lifeCycleState

A deal can be in various states throughout its life (ie: from start to maturity of the deal). This returns the state that a deal is in at a particular point in time.

Details of the states will be determined later.

Date settlementDate() settlementDate

The date on which goods are exchanged and payment is made.

Organization settlementForum() settlementForum

The clearing house responsible for handling settlement will determine the default settlement rules. This returns an organization with the role being that of a clearing house.

CommodityHolding bookValue() bookValue

The amount and currency recorded in the trade (usually for accounting purposes).

This part will be extended and updated at a later stage.

Organization exchange() exchange

There are times when deals which have been done with a counterparty are not settled directly with the counterparty. Instead, they are settled with an exchange. This returns the exchange involved in the trade, if applicable.

Date maturityDate() maturityDate

The maturity date of the deal.

Party counterparty() counterparty

A deal wants knowledge of the other party involved in the transaction, be it a book (an internal party) or an organization (external party).

Classifier classifier() classifier

Returns a classifier that will contain selective pieces of information about this deal such that it can uniquely identify/classify this deal. This information is stored as a dictionary. An example of part of a classifier is: [book -> bk1, bk2, counterparty -> cpty3, cpty6, etc.].

This method returns the classifier obtained by sending the “deriveClassifierFrom” message to the dealType (returned from the dealType method) and sending itself (this deal) as the argument. See DealType §1.3.

1.2 DealPurpose

This interface defines what the deal will be used for, or what the intended purpose of the deal is. An example of a non-trade deal purpose is an automatic internally generated trade. Currently the concept of a deal purpose has been modeled by a class hierarchy, however we leave open the option of being able to represent this as reference data. This can be achieved by altering the model so that it inherits from reference data and replaces the functionality that was achieved via a class hierarchy with attributes.

1.2.1 Relationships

	Class	Description	Notes
⇑	Identifiable		
↓	DealPurposeModel §2.2		
↔	DealModel §2.1	dealPurpose	

⇑:Inherits ↓:Realized by ↔:Association →:Navigable ◊:Aggregate ♦:Composite

1.3 DealType

A deal type describes a particular kind of deal (eg. loans, bonds, etc).

1.3.1 Relationships

	Class	Description	Notes
⇑	Identifiable		
↓	DealTypeModel §2.4		
↔	DealModel §2.1	dealType	

⇑:Inherits ↓:Realized by ↔:Association →:Navigable ◊:Aggregate ♦:Composite

1.3.2 Operations

Classifier deriveClassifierFrom(Deal deal) deriveClassifier-From
deal: Deal The deal from which a classifier will be derived.

This method returns a classifier that contains selective pieces of information about the given deal. This information is chosen such that it can uniquely identify/classify a deal.

2 Classes

2.1 DealModel

The DealModel class realizes the Deal interface. This class contains the specification for the deal information common to all types of deals. Through inheritance from DynamicDataModel, versioning information is also found on a deal.

The exchange, clearingHouse and broker attributes are found on the DealModel because they do not participate in the exchange of goods. If an exchange is participating in the exchange of goods, it will be acting as a counterparty.

2.1.1 Relationships

	Class	Description	Notes
⇑	DynamicDataModel		
↑	Deal §1.1		
↑	Validatable		
↔	Instrument	instrument	→
↔	Book	book	→
↔	Organization	clearingBroker	→
↔	User	user	→
↔	Organization	broker	→
↔	Organization	exchange	→
↔	Organization	settlementForum	→
↔	DealType §1.3	dealType	→
↔	DealPurpose §1.2	dealPurpose	→

⇑:Inherits ↑:Realizes ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

2.1.2 Attributes

dealDate: Date The date on which a trade occurs (ie: the effective date of the deal).

buyOrSell: Integer = 1 (buy) Enumeration {1,-1 }. This returns either +1 or -1 to represent whether a commodity has been bought or sold. This trading of the commodity is from the dealer's perspective.

This value will be determined from the transaction's buySellMultiplier and will be the opposite value (since the transaction is from the counterparty's perspective).

settlementDate: Date The date on which goods are exchanged and payment is made.

2.1.3 Operations

validate() validate

The validation for a deal will be defined by the business rules, and hence is left incomplete.

Party counterparty() counterparty

This will be obtained from the transaction (found within the instrument).

Date maturityDate() maturityDate

This information will come from within the instrument.

CommodityHolding bookValue() bookValue

The bookValue information will come from the Instrument.

2.2 DealPurposeModel

This class is a concrete realization of the DealPurpose interface. It is expected that subclasses will be added to this class as business functionality is determined.

2.2.1 Relationships

	Class	Description	Notes
↑	DealPurpose §1.2		
⇓	TradePurposeModel §2.3		

⇓:Inherited by ↑:Realizes

2.3 TradePurposeModel

This class represents a normally traded deal with no special purpose.

2.3.1 Relationships

	Class	Description	Notes
⇑	DealPurposeModel §2.2		

⇑:Inherits

2.3.2 Operations

String identifier() identifier

Returns the string “Trade Purpose”.

2.4 DealTypeModel

This class is the superclass of the deal type hierarchy. Each subclass will only have a single instance.

2.4.1 Relationships

	Class	Description	Notes
↑	DealType §1.3		
⇓	LoanTypeModel §2.10		
⇓	FXTypeModel §2.8		
⇓	BondTypeModel §2.5		
⇓	EquityTypeModel §2.6		
⇓	FRATypeModel §2.7		
⇓	IRSwapTypeModel §2.9		
⇓	REPOTypeModel §2.11		

⇓:Inherited by ↑:Realizes

2.4.2 Operations

Classifier deriveClassifierFrom(Deal deal) deriveClassifier-From
deal: Deal This is the deal that a classifier will be derived from.

This method returns a classifier that will contain non-type specific information of the given deal. This classifier will contain the general information of a deal that will be present regardless of its type. Subclasses that override this method will call this method and extend the returned classifier to contain information specific to the particular type of the given deal.

This method creates a new Classifier and adds the following key/value pairs to it:

Add to the classifier the key “Books”, with the value returned from sending the “book” message to the given deal and then enclosing the result in a collection (or set). Add to the classifier the key “Counterparties”, with the value returned from sending the “counterparty” message to the given deal and then enclosing the result in a collection (or set). Add to the classifier the key “Deal types”, with the value returned from sending the “dealType” message to the given deal and then enclosing the result in a collection (or set). Add to the classifier the key “Deal purposes”, with the value returned from sending the “dealPurpose” message to the given deal and then enclosing the result in a collection (or set).

2.5 BondTypeModel

This class represents the bond type.

2.5.1 Relationships

	Class	Description	Notes
⇑	DealTypeModel §2.4		

⇑:Inherits

2.5.2 Operations

String identifier() identifier

Returns the string “Bond”.

Classifier deriveClassifierFrom(Deal deal) deriveClassifier-From

deal: Deal This is the deal that a classifier will be derived from.

This method takes a bond deal as its parameter, and returns a classifier describing this deal. This is done in three stages:

1. This method calls its superclass’s method of the same name (with the same argument), which returns a classifier for the given deal containing non type-specific information. The current method now only needs to add the information specific to the deal’s type (a bond deal).

2. This method adds to the classifier a key named “securities”, with a corresponding value that is a set containing the underlying security of the deal. Perform the following message sends to give a collection containing two transactions:
deal.instrument.transactionSequence.
One will be a price transaction the other will be an ordinary transaction. Take the price transaction and perform the following message sends to produce the underlying security of the deal:
priceTransaction.underlyingTransaction.instrument.unitInstrument.

3. This method adds to the classifier a key named “currencies excluding currency pairs”, with a corresponding value that is the set of currencies (excluding currency pairs) in this deal. The set of currencies can be obtained by getting the two currencies used in bond deal, namely the bond currency and the purchase currency. Note that if these currencies are the same then the set will ensure that there is only one currency recorded. Perform the following message sends to give a collection containing two transactions:
deal.instrument.transactionSequence.
One will be a price transaction the other will be an ordinary transaction. Take the

price transaction and perform the following message sends to produce one of the currencies used in the deal:
priceTransaction.underlyingTransaction.instrument.unitInstrument.currency. Take the price transaction again and perform the following message sends to produce the other currency used in the deal:
priceTransaction.instrument.commodity.

2.6 EquityTypeModel

This class represents the equity type.

2.6.1 Relationships

	Class	Description	Notes
⇑	DealTypeModel §2.4		

⇑:Inherits

2.6.2 Operations

String identifier() identifier

Returns the string "Equity".

Classifier deriveClassifierFrom(Deal deal) deriveClassifier-From
deal: Deal This is the deal that a classifier will be derived from.

This method takes an Equity deal as its parameter, and returns a classifier describing this deal. This is done in three stages:

1. This method calls its superclass's method of the same name (with the same argument), which returns a classifier for the given deal containing non type-specific information. The current method now only needs to add the information specific to the deal's type (an equity deal).

2. This method adds to the classifier a key named "currencies excluding currency pairs", with a corresponding value that is a set of currencies (excluding currency pairs) in this deal. Perform the following message sends to give a collection containing one price transaction:
deal.instrument.transactionSequence.
Perform the following message sends on the price transaction to produce one of the currencies of the deal:
priceTransaction.instrument.commodity. Perform the following message send on

the same price transaction to produce the other currency of the deal:
priceTranscation.underlyingTranscation.instrument.instrument.currency.

3. This method adds to the classifier a key named “equities”, with a corresponding value that is a set containing the equity in this deal. Perform the following message sends to give a collection containing one price transaction:
deal.instrument.transactionSequence.
Perform the following message sends on the price transaction to produce the equity:
priceTranscation.underlyingTranscation.instrument.instrument.

2.7 FRATypeModel

This class represents the Forward Rate Agreement (FRA) type.

2.7.1 Relationships

	Class	Description	Notes
⇑	DealTypeModel §2.4		

⇑:Inherits

2.7.2 Operations

String identifier() identifier

Returns the string “FRA”.

Classifier deriveClassifierFrom(Deal deal) deriveClassifier-From
deal: Deal This is the deal that a classifier will be derived from.

This method takes an Forward Rate Agreement (FRA) deal as its parameter, and returns a classifier describing this deal. This is done in two stages:

1. This method calls its superclass’s method of the same name (with the same argument), which returns a classifier for the given deal containing non type-specific information. The current method now only needs to add the information specific to the deal’s type (a FRA deal).

2. This method adds to the classifier a key named “currencies excluding currency pairs”, with a corresponding value that is a set of currencies (excluding currency pairs) in this deal. The following message sends:
deal.instrument.transactionSequence will give a collection containing two transactions. Both transactions will be a cashflow. The currencies contained within these

two cashflows will always be the same. Perform the following message sends to one of the cashflows to produce the currency of the deal:
cashflow.commodity.

2.8 FXTypeModel

This class represents the FX type.

2.8.1 Relationships

	Class	Description	Notes
⇑	DealTypeModel §2.4		

⇑:Inherits

2.8.2 Operations

String identifier() identifier

Returns the string "FX".

Classifier deriveClassifierFrom(Deal deal) deriveClassifier-From
deal: Deal This is the deal that a classifier will be derived from.

This method takes an FX deal as its parameter, and returns a classifier describing this deal. This is done in two stages:

1. This method calls its superclass's method of the same name (with the same argument), which returns a classifier for the given deal containing non type-specific information. The current method now only needs to add the information specific to the deal's type (an FX deal).

2. This method adds to the classifier a key named "currency pairs", with a corresponding value that is a set containing the underlying security of the deal. Perform the following message sends to give a collection containing one transaction:
deal.instrument.transactionSequence.
This transaction will be a price transaction. Take the price transaction and perform the following message sends to produce the currency pair of the deal:
priceTransaction.price.specifier.currencypair.

2.9 IRSwapTypeModel

This class represents the Intrest Rate Swap (IRSwap) type.

2.9.1 Relationships

	Class	Description	Notes
⇑	DealTypeModel §2.4		

⇑:Inherits

2.9.2 Operations

String identifier() identifier

Returns the string “IRSwap”.

Classifier deriveClassifierFrom(Deal deal) deriveClassifier-From

deal: Deal This is the deal that a classifier will be derived from.

This method takes an Interest Rate Swap (IRSwap) deal as its parameter, and returns a classifier describing this deal. This is done in two stages:

1. This method calls its superclass’s method of the same name (with the same argument), which returns a classifier for the given deal containing non type-specific information. The current method now only needs to add the information specific to the deal’s type (an IR Swap deal).

2. This method adds to the classifier a key named “currencies excluding currency pairs”, with a corresponding value that is a set of currencies (excluding currency pairs) in this deal. An IRSwap may have up to two different currencies. Perform the following message sends to give a collection containing two transactions:
deal.instrument.transactionSequence.
For each transaction the following message sends will provide the currency of that transaction:
transaction.instrument.specifier.paymentSpecification.commodity.

2.10 LoanTypeModel

This class represents the loan type.

2.10.1 Relationships

	Class	Description	Notes
⇑	DealTypeModel §2.4		

⇑:Inherits

2.10.2 Operations

String identifier() identifier

Returns the string “Loan”.

Classifier deriveClassifierFrom(Deal deal) deriveClassifier-From

deal: Deal This is the deal that a classifier will be derived from.

This method takes a load deal as its parameter, and returns a classifier describing this deal. This is done in two stages:

1. This method calls its superclass’s method of the same name (with the same argument), which returns a classifier for the given deal containing non type-specific information. The current method now only needs to add the information specific to the deal’s type (a loan deal).

2. This method adds to the classifier a key named “currencies excluding currency pairs”, with a corresponding value that is a set containing the currency of this deal. The following message sends:
deal.instrument.transactionSequence will give a collection containing two transactions. One of these transactions will be a simple cashflow transaction. Take the simple cashflow transaction and perform the following message sends to produce the currency of the deal:
simpleCashflowTransaction.commodity.

2.11 REPOTypeModel

This class represents the Repurchase Agreement (REPO) type.

2.11.1 Relationships

	Class	Description	Notes
⇑	DealTypeModel §2.4		

⇑:Inherits

2.11.2 Operations

String identifier() identifier

Returns the string “REPO”.

Classifier deriveClassifierFrom(Deal deal) deriveClassifier-From

deal: Deal This is the deal that a classifier will be derived from.

This documentation is unavailable until more information on the structure of REPOs deals becomes available.

3 Enumerations

3.1 DealActionEnumeration

An enumeration used to specify the action applied to a deal.

3.1.1 Operations

«Static Method» DealActionEnumeration entry() entry

Returns a DealActionEnumeration with the name “Entry”.

«Static Method» DealActionEnumeration modify() modify

Returns a DealActionEnumeration with the name “Modify”.

«Static Method» DealActionEnumeration view() view

Returns a DealActionEnumeration with the name “View”.

«Static Method» DealActionEnumeration confirm() confirm

Returns a DealActionEnumeration with the name “Confirm”.

4 Associations

Table 1: Deals— Associations

Association			
Role	Class	Card.	Notes
instrument			

Table 1: . . . continued

Association Role	Class	Card.	Notes
	Instrument		→
	DealModel §2.1		
book			
	Book		→
	DealModel §2.1		
clearingBroker			
	Organization		→
	DealModel §2.1		
user			
	User		→
	DealModel §2.1		
broker			
	Organization		→
	DealModel §2.1		
exchange			
	Organization		→
	DealModel §2.1		
settlementForum			
	Organization		→
	DealModel §2.1		
dealType			
	DealType §1.3		→
	DealModel §2.1		
dealPurpose			
	DealPurpose §1.2		→
	DealModel §2.1		

→:Navigable ◇:Aggregate ◆:Composite

4.1 instrument

Role: *Navigable* Instrument.
Role: DealModel.

4.2 book

Role: *Navigable* Book.
Role: DealModel.

4.3 clearingBroker

Role: *Navigable* Organization.
Role: DealModel.

4.4 user

Role: *Navigable* User.
Role: DealModel.

4.5 broker

Role: *Navigable* Organization.
Role: DealModel.

4.6 exchange

Role: *Navigable* Organization.
Role: DealModel.

4.7 settlementForum

Role: *Navigable* Organization.
Role: DealModel.

4.8 dealType

Role: *Navigable* DealType.
Role: DealModel.

4.9 dealPurpose

Role: *Navigable* DealPurpose.
Role: DealModel.

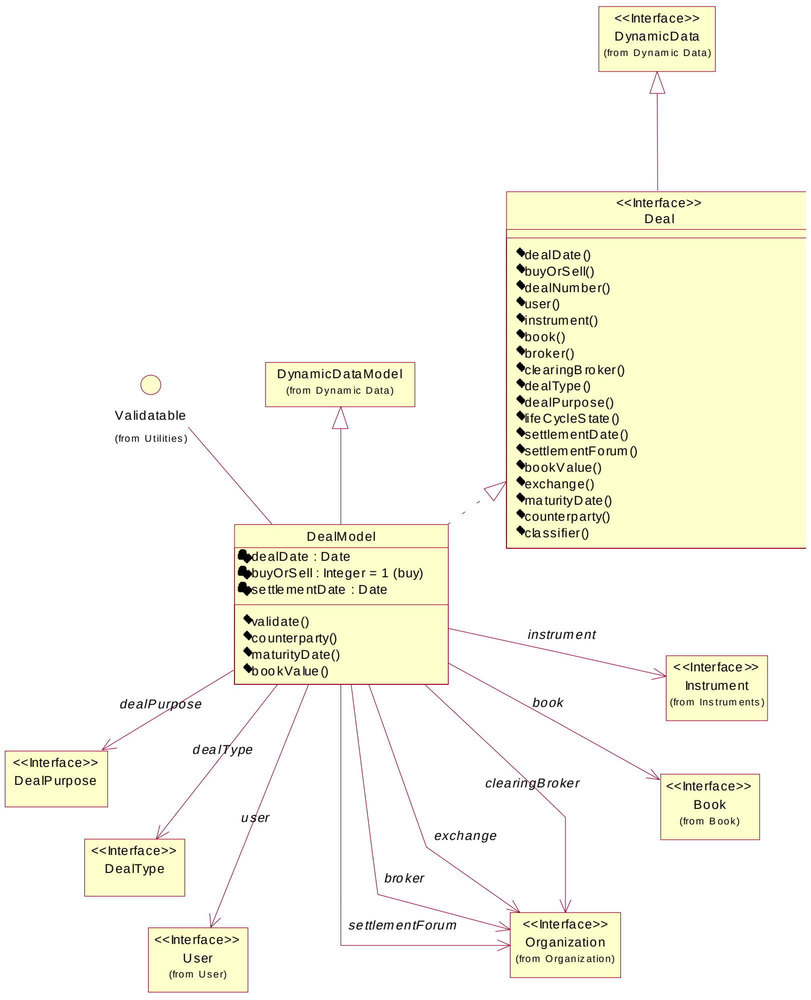

Figure 1: Class Diagram— Deals

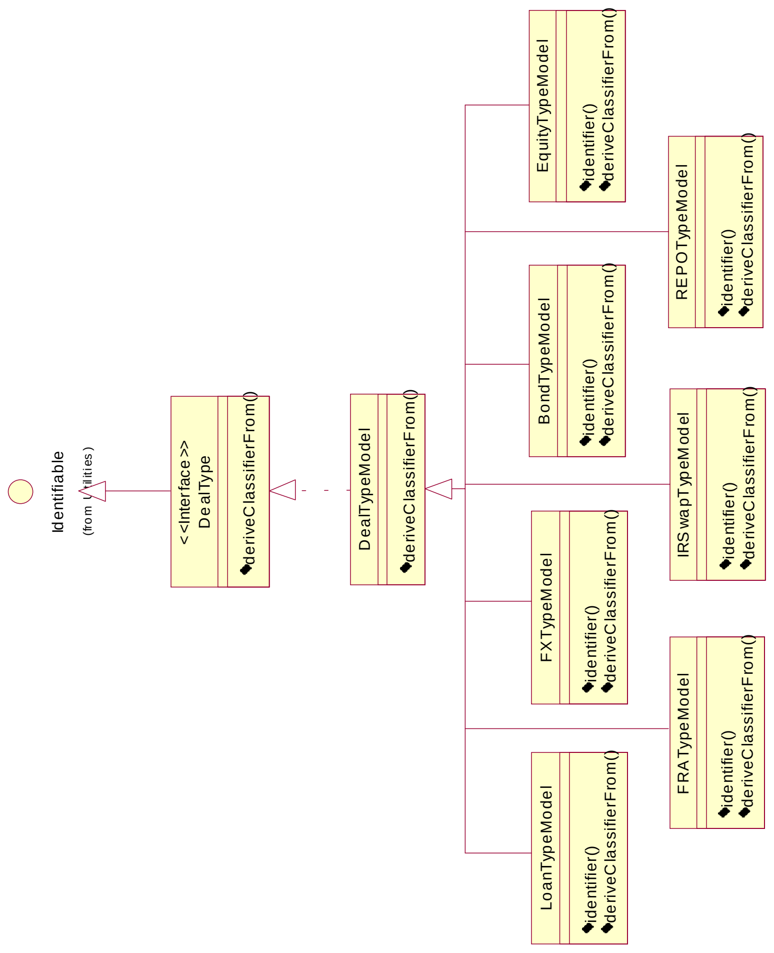

Figure 2: Class Diagram— Deal Types

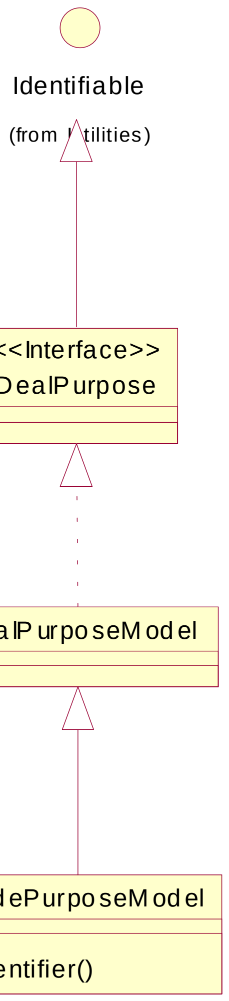

Figure 3: Class Diagram— Deal Purpose

References