October 2017

ISDA®

ISDA Common Domain Model Version 1.0 Design Definition Document

CONTENTS

CONTENTS CONTINUED

PREFACE

Presented in this document is the definition of version 1.0 of the ISDA Common Domain Model (ISDA CDM). The ISDA CDM aims to deliver a standardized model for the post-execution trade lifecycle, focusing on the non-differentiating aspects of that trade lifecycle that are candidates for mutualization by the industry. The purpose of this initial conceptual version of the model is to begin a discussion with ISDA's membership and the broader industry on the feasibility of an ISDA CDM as the basis for such common process and data standards.

ISDA CDM version 1.0 explores the fundamental concepts, which could lead to a more complete ISDA CDM. It is expected that ISDA CDM version 2.0 will build on the concepts incorporated in version 1.0 in a more systematic and formal way to deliver a digital artefact, based on an existing derivatives data standard, Financial products Markup Language (FpML®). Most importantly, it will evolve based on feedback and engagement with readers of this document. With this in mind, ISDA welcomes feedback and questions from readers of this document.

Please direct questions or comments to: **MarketInfrastructureandTechnology@isda.org.**

OBJECTIVES

This document defines a consistent event and product model for financial products (the ISDA CDM).

This paper focuses on post-trade activities, as these are where most fragmentation and opportunities exist.

While the model definition is generic, and could be adopted via any technology, the implementation is targeted at distributed ledgers (DLs) to exploit their embedded lineage and consistency properties[1]. For this reason, examples and illustrations mention DLs to explain concepts or design ideas, but that is not to contradict the statement that the model could be adopted via any technology.

[1] Sections Key Properties of a Distributed Ledger and The Complete Mode expand on the concepts of lineage and consistency in DLs

DEFINITIONS

Key Properties of a Distributed Ledger

DLs enforce two relationships: consistency and lineage.

Lineage[2]

Lineage is defined:

$O(n)=f_m(O(<n))$

Where:

- $O(0)$ is a direct observation of the outside world. This can be a single number, a set of things or a human action (eg, a button is pressed).
- $f_m()$ is a piece of code that acts on an observation (ie, a smart contract in DL terminology – although embedded function would be a better generic description).
- $O(n>0)$ is defined as a derived observation (ie, some code f is run to turn $O(0)$ observation into a $O(1)$, the derived observation).
- $f_m()$ can have more than one input.
- As $f_{1\ldots m}()$ and $O(0\ldots n)$ are persisted on a DL, then the ledger enforces lineage. It is possible to trace a derived observation back through the function code and the inputs to the function.
- This is a hierarchy definition.

Consistency

Consistency is defined:

$N_1[f,O] \equiv N_2[f,O] \equiv N_3[f,O] \equiv ..\equiv N_x[f,O]$

Where:

- N is a separate node on a DL;
- f is a piece of code;
- O is an identical observation;
- As a node can be in a different organization (say Bank 1, Asset Manager 2, CCP, Regulator), this means there is confidence that what one party sees on their node is identical to what the other party sees on theirs.

[2] Data lineage is a formal concept (or language) in computer science. Similarly, DLs are formal machines in computer science

Lineage and consistency are properties of the DL itself (usually using some cryptographic mechanism)[3]. Conceptually, DLs are Von Neumann/Turing machines, where there is no distinction between data, the code that acts on the data and the results of the code.

Standard Event Space

All events (or observations of the DL definition) are defined in terms of a simple space:

$S \equiv [P_1, P_2, Q, (E)]$

Where:

P_1 and P_2 are parties[4];

Q is a scalar quantity[5];

(E) is a 'pointer' to the definition of the economic description of the contract[6].

Events are identified by a 'hash' mechanism that produces an identifier (eg, '19b9') with lineage controlled by the underlying DL/blockchain technology[7].

MODEL DEFINITION APPROACH

Event definitions are defined in terms of 'before' and 'after' states. This avoids consciously or sub-consciously importing implementation methods related to legacy technology.

Design for Distributed Execution

Processing of financial contracts lends itself to concurrent/parallel/distributed processing. This needs to be recognized and incorporated into the design from the start and all the way through the process stack. This is not the case in most current implementations (especially if they have batch processing anywhere in that stack).

[3] It is argued in the Appendix that no conventional set of communicating (messaging or batch) systems can achieve consistency or lineage, which is why regulations like BCBS 239 are so problematic

[4] Parties are not defined in terms of party and counterparty, as this depends on context (ie, if you are P1, then P2 is the counterparty, and vice versa), and this introduces unnecessary complexity by eliminating symmetry. There is only one 'party' hierarchy. However, it is useful to keep this separate in the initial definition

[5] Usually 'notional amount' or 'units of an instrument'

[6] This is historically called something like 'product' or 'instrument', 'static' or 'reference' data. However, this is an unnecessary description in a DL framework

[7] In a DL environment, it is not necessary to think in terms of sequential identifiers and versions. These are techniques relating relational databases. DLs have a generic means of recording lineage, fundamental to their core definition/technology, which is described above

Design for Minimum Change

A good design should require the minimum amount of change to achieve the design objective. This is particularly true in a distributed environment, where consensus processes run between the nodes of a DL.

Parametric Processing – Design for Minimum Persistence

Products are defined with as few parameters as possible and only expanded once at the end of the process on silicon/memory, where no consensus process required.

The historic design where this has been used is cashflow and event schedules. For example, take a coupon – a simple function with a few parameters (quantity, start, end, frequency, day count and holidays) and create and persist a schedule. The persisted derived schedule itself is far larger than the parametric coupon object – this will be a problem in a distributed environment. There is also the problem of any change to the parameters requiring a recalculation of the schedule, and the associated distribution and consensus process for this recalculation. In the over-the-counter (OTC) market, most contracts don't go to their scheduled maturity date (they are compressed/torn up/terminated), so all the initial calculation/storage/consensus processing cost and removal/consensus process is a waste of resources as the cashflows never settle[8 9].

Optimize for Repeated Processes

Ensure that the design supports the most frequent process.

In this case:

- A confirmation process is conducted once at the beginning of a position's life.
- The resulting position may be run through a risk process thousands of times in its life.
- The persistence design should therefore be optimized for the repeated action.

[8] The computation and storage of cashflow schedules made sense in the mid-1990s. Processing was conducted on single processors running at 100 MHz with a few GB RAM, portfolios were 'small' and compression/tear-up services and complex stress/risk/capital calculations didn't exist. Processes were bound by limited processing power, and communication/storage overhead was acceptable. The code and data storage was also separate. Roll forward to now: processing is plentiful – farms of 18 core 4GHz are common, and communication is faster

[9] In a process where trades are identical on a DL, variation margin is paid, and settlement is also on a DL, then could calculate the cash flows a few days before settlement

THE ISDA CDM

The ISDA CDM is built up in stages.

- First, the concept of independent events is introduced (ie, negotiated/advised events/actions).
- Second, the concept of dependent events is introduced. It is shown that the results of dependent events are the same as independent events without the negotiation (the terms are defined elsewhere, in contracts, etc).
- Third, the concept of exchange of value is introduced.
- Finally, the definition of products is introduced and expanded upon.

It is then shown that 'event', 'exchange' and 'product' are tightly related, if not identical. This is the basis of the fractal symmetry – ie, identical structures/processes at different scales. This minimizes the code that needs to be deployed by re-using the code for historically different processes.

Independent Events

The objective of this section is to define a primitive set of operations that exist on S= [P_1, P_2, Q, (E)]. The key point is the before and after events are a collection in space S.

The first set of events are independent in that they have to be negotiated (ie, priced/quoted) or advised.

Although an interest rate swap (IRS) is shown for illustrative purposes, these operations are independent of the economics (E), and are therefore are applicable to all asset classes.

The events considered are:

- Single events (one-for-one state transition, between before and after);
  - New;
  - Terminate for cash;
  - Amend;
  - Cancel;
- Single events (one before but multiple after);
  - Split (also known as allocation);
  - Partial termination for cash (also known as a decrease; an increase is the logical opposite);
  - Full assignment for cash (full novation);
  - Partial assignment for cash (partial novation);

- Clear*;
- Intermediation* (prime brokerage);

- Multiple events (many to one);
  - Aggregation*;

- Multiple events (many to many);
  - Tear-up*.

** NB: In these events, the re-use of very similar primitive state transitions that have been used in previous events becomes evident.*

New

	Event Identifier	Party	Party	Quantity	Economics	Comments
New						
Before	-o-	-o-	-o-	-o-	-o-	
After	3fb9	P1	P2	Q	IRS(5dc4)	

	Item	Description	Comments
1	Before	Nothing exists '-o-'.	
2	Event Identifier	3fb9 is a hash function result that identifies the event – for example, on a DL.	
3	Economics	IRS(5dc4) is a pointer to an item on the DL that defines the economic terms, dates, etc of the contract.	The definition and example of economics is considered later in this document[10]. Assume here the IRS has no value at inception.

Terminate for Cash

At some point in the future, the parties negotiate to terminate the contract for cash.

	Event Identifier	Party	Party	Quantity	Economics	Comments
Full Term for Cash						
Before	3fb9	P1	P2	Q	IRS(5dc4)	IRS replaced by a termination fee
After	2ef3	P1	P2	c,ccc.cc	Cash	

[10] See Fractal Symmetry

	Item	Description	Comments
1	Before	IRS(5dc4) exists.	
2	Event Identifier	2ef3 identifies the termination event The DL infrastructure joins the before and after events 3fb9 and 2ef3.	
3	Economics	IRS (5dc4) has been replaced by a cash flow of quantity 'c,ccc.cc'. Once the cashflow has settled, there are no remaining contractual obligations between P1 and P2.	This is a negotiated termination – ie, both parties have to agree on cash value.

Amend

The economic terms are changed in some way. However, there is no negotiated cashflow.

There are necessary constraints on this operation that need to be explored further – ie, the intention[11] is to amend the transaction to correct an error. Consequently, it should probably not be possible to change an IRS to an FX option, for example.

	Event Identifier	Party	Party	Quantity	Economics	Comments
Amend						
Before	3fb9	P1	P2	Q	IRS(5dc4)	
After	82ef	P1	P2	Q'	IRS(7dj5)	Economic details of trade are changed

	Item	Description	Comments
1	Before	IRS(5dc4) exists.	
2	Event Identifier		
3	Economics	Q→Q' and IRS(7dj5) represent the change to the IRS contract. DL infrastructure joins the before and after events 3fb9 and 82ef.	There should be restrictions on what can be changed. Also, it is believed, the event should capture intention.

Note: The distinction between 'Amend' and 'Cancel/Correct' does not exist in this definition of before/after states. This is because the linkage (addressing) on a DL is the same for both[12].

[11] The concept of tying intention to events is to be explored further in subsequent versions of the document. Comments on implementations of such event processing would be welcome input to further development of this event model

[12] In conventional relational database management system (RDBMS) technology, the addressing is in terms of elements in a table. This is the root of much of the complexity when multiple RDBMSs have to be kept in sync or events have to be extracted and reported. For example:

- An amend changes a version number but not a trade ID
- Cancel/correct changes trade ID, and therefore there needs to be an explicitly programed mechanism to join the distinct trade IDs

Split (Allocation)

A block trade is executed with a fund manager P2. P2 decides to allocate this trade across two funds controlled by the fund manager.

	Event Identifier	Party	Party	Quantity	Economics	Comments
Split (a.k.a. Allocation)						
Before	3fb9	P1	P2	Q'	IRS(5dc4)	
After	3k9s	P1	P2/a	q1	IRS(5dc4)	Where P2/a and P2/b are 'funds' of P2
After	3k9s	P1	P2/b	q2	IRS(5dc4)	Where q1+q2 = Q

	Item	Description	Comments
1	Before	IRS(5dc4) exists.	
2	Event Identifier	3k9s identifies the event that splits the notional.	
3	Party P2	P2 is a manager of two funds, A and B.	See later for definition of party hierarchy[13].
4	Quantity	The original trade quantity Q is split across the two funds.	
3	Economics	IRS(5dc4) does not change.	

Partial Termination

	Event Identifier	Party	Party	Quantity	Economics	Comments
Part Term						
Before	3fb9	P1	P2	Q	IRS(5dc4)	
After	3k9s	P1	P2	q1	IRS(5dc4)	Only change is quantity
After	3k9s	P1	P2	c,ccc.cc	Cash	Partial termination fee

	Item	Description	Comments
1	Before	IRS(5dc4) exists.	
2	Event Identifier	3k9s identifies the event that splits the notional.	
3	Economics	There is no change in the terms IRS(5dc4).	
4	Quantity	The original quantity is reduced from Q to q1.	
3	Cash	A cash quantity c,ccc.cc is exchanged in for the decrease in quantity of economics (IRS(5dc4)).	The cash consideration for the reduction Q→q1 is a negotiated event.

Note:

- *On a strict delta definition, this is a decrease in one asset (the IRS) and an increase in another asset (cash).*
- *An increase is logically the same but reversed (quantity increases and cash reverses).*

[13] See Fractal Symmetry

Full Assignment

An existing contract is transferred to a new party and a fee is paid.

	Event Identifier	Party	Party	Quantity	Economics	Comments
Full Assign						
Before	3fb9	P1	P2	Q	IRS(5dc4)	An IRS
After	3k9s	P1	P3	Q	IRS(5dc4)	Change of party (P2 --> P3) on IRS
After	3k9s	P2	P3	c,ccc.cc	Cash	Fee between P3 and P2

	Item	Description	Comments
1	Before	IRS(5dc4) exists.	
2	Event Identifier	3k9s identifies an event that records the assignment and associated fee.	
3	Economics	There is no change in the terms IRS(5dc4).	
4	Party	The contract is assigned from P2 to P3.	
4	Quantity	As this is a full assignment, there is no change in quantity.	
3	Cash	A cash quantity c,ccc.cc is exchanged between P2 and P3 to compensate for the change in ownership of the contract.	The cash consideration for the reassignment P2→P3 is a negotiated event.

Partial Assignment

An existing contract is partially transferred for a negotiated fee.

	Event Identifier	Party	Party	Quantity	Economics	Comments
Partial Assign						
Before	3fb9	P1	P2	Q	IRS(5dc4)	An IRS
After	3k9s	P1	P2	q1	IRS(xxxx)	Where Q=q1+q2
After	3k9s	P1	P3	q2	IRS(xxxx)	Change of party (P2 --> P3)
After	3k9s	P2	P3	c,ccc.cc	Cash	Fee between P3 and P2

	Item	Description	Comments
1	Before	IRS(5dc4) exists.	
2	Event Identifier	3k9s identifies an event that records the assignment and associated fee.	
3	Economics	There is no change in the terms IRS(5dc4).	
4	Party	The contract is assigned from P2 to P3.	
4	Quantity	As this is a full assignment, there is no change in quantity.	
3	Cash	A cash quantity c,ccc.cc is exchanged between P2 and P3 to compensate for the change in ownership of the contract.	The cash consideration for the reassignment P2→P3 is a negotiated event.

Cancel

A cancel event should be unusual in normal transactions. However, it is required as part of portfolio events (particularly tear-up transactions).

Cancel doesn't mean that the trade has never existed. It means that it has ceased to exist now.

	Event Identifier	Party	Party	Quantity	Economics	Comments
Cancel						
Before	3fb9	P1	P2	Q	IRS(5dc4)	
After	-o-	-o-	-o-	-o-	-o-	Trade is 'cancelled' - no corresponding cashflow

Clear

	Event Identifier	Party	Party	Quantity	Economics	Comments
Clear						
Before	t73j	P1	P2	Q	[E]	Where P3 is a CCP - Simultaneous novations and removal of any contract terms ([E] -->[E]) offensive to CCP (eg, early termination)
After	t73j	P3	P2	Q	[E]'	
After	t73j	P3	P1	(Q)	[E]'	

May be combined with allocation/split process.

Intermediation (Prime Brokerage)

	Event Identifier	Party	Party	Quantity	Economics	Comments
Intermediate						
Before	t73j	P1	P2	Q	[E]	Where P4 is a prime broker – simultaneous novations and removal of any contract terms ([E] -->[E]) offensive to PB (eg, early termination)
After	t73j	P4	P2	Q	[E]'	
After	t73j	P4	P1	(Q)	[E]'	

Intermediation is basically the same as clearing.

Aggregation

	Event Identifier	Party	Party	Quantity	Economics	Comments
Aggregation						Logically the reverse of a split/ allocation above
Before	3fb9	P1	P2	q1	IRS(5dc4)	
Before	3k9s	P1	P2	q2	IRS(5dc4)	
After	3k9s	P1	P2	Q	IRS(5dc4)	Where Q=q1+q2

Aggregation is logically the reverse of allocation process above.

Tear-up/Portfolio Compression

	Event Identifier	Party	Party	Quantity	Economics	Comments
Tear-up/ Portfolio Compression						This is a set of changes to a portfolio that is risk neutral OR change in risk is compensaed for by a fee (CCY) Bilateral shown – multilateral is logically no different to 'disappear'
Before	gfd4	P1	P2	Q1-99	[E1-99]	
Before	gfd4	P1	P2	Q100	[E100]	
Before	gfd4	P1	P2	Q101	[E101]	
Before	gfd4	P1	P2	Q102	[E102]	
After	gfd4	P1	P2	Q100	[E100]	No change
After	gfd4	P1	P2	Q101-x	[E101]	Decrease
After	gfd4	P1	P2	Q102+y	[E102]	Increase
After	gfd4	P1	P2	Q103	[E103]	New
After	gfd4	P1	P2	c,ccc.cc	CCY	Fee

A tear-up is a portfolio process that aims to reduce the notional value of a portfolio without materially changing the net risk.

The majority of trades are cancelled – there may be a few increases or decreases or the odd new trade. If the risk replication is not exact, then there will be a fee associated with the tear-up/ compression process.

Independent Events Summary

- All events can be expressed in terms of the simple space S=[P_1,P_2,Q,(E)].
- The events are a collection of before and after states.
- The events are agnostic to the definition of economics in (E).
- There are very few events, as the simple space only contains four 'addresses' (strictly three in examples illustrated so far, as P1 does not change in any of the examples above).
- There are three change types (P2→Pn, Q1→Q2 and -o- → inclusion of a negotiated cashflow).
    - Therefore, the examples logically further reduce to a set of primitive operations.
- Events are symmetrical[14].
    - And therefore are simpler to code and to incorporate an 'un-do' of the event. This implies that the underlying event solution should be very simple and generic.

#	Model event name	Note
1	o New	Single events (one-for-one state transition, between before and after)
2	o Terminate for cash	
3	o Amend	
4	o Cancel	
5	o Split	Single events (one before but multiple after)
6	o Partial termination for cash	
7	o Full assignment for cash (full novation)	
8	o Partial assignment for cash (partial novation)	
9	o Clear	
10	o Intermediation (prime brokerage)	
11	o Aggregation	Multiple events (many to one)
12	o Tear-up	Multiple events (many to many)

[14] This statement may be further explained in a subsequent version of this document

Dependent Events

Derivatives are contingent claims and, as such, have reference to future events. As time passes and these events become present, the crystallization of these events need to be recorded (observed) and the appropriate actions need to be taken.

Generally, observations are currently split into two types:

- Observation of data defined in the contract (eg, a floating rate set, dividend declaration, observation of temperature, etc)[15].
- Observation of a human action required in a contract (eg, exercise notice served on a generic-type option (see below)).

This distinction is not helpful. In a DL paradigm, there is no difference. Both result in an observation that is logically a number (rate is a decimal number, exercise is a yes/no – 0/1 binary observation[16]). Both have same constraints: 'who is allowed to do it' or 'who is the authoritative source?'

The definition above is important, as the same primitive operations are used in all limit and review type controls. For example, if a firm has a policy that 'all transactions above 1,000,000 need a review action', then this is logically no different to the terms of a derivative contract. In other words, there is some pre-defined policy that triggers a human to have to do something and evidence that the review has taken place (and any subsequent actions as part of the review[17]). This is 'evidencing the application of professional judgement' that is at the heart of BCBS 239.

Observations

Observations record 'events' defined in the contract. They may lead to subsequent actions (eg, setting of a cashflow or change of contract type).

	Event Identifier	Party	Party	Quantity	Economics	Comments
Numeric Observation						
Before	xxxY	P1	P2	Qty(xxxA)	IRS(xxxx)	
After	xxxZ	P1	P2	Qty(xxxA)	IRS(xxxx).Numeric	Eg, observation of a floating rate on an IRS or an MV on a position
Action Observation						
Before	xxxY	P1	P2	Qty(xxxA)	OPTION(xxxx)	
After	xxxZ	P1	P2	Qty(xxxA)	OPTION(xxxx).Action	Eg, a person exercising an option

[15] Note: We have deliberately not said 'market data' here, as this is an unhelpful historic definition and leads to the separation of physical systems that we have in the current environment, which may not be the case in future implementations

[16] Note: There is a subtlety here, in that there is the tristate of {'yes', 'no', 'no response yet'}

[17] For example, the classic control of 'review all reconciliation breaks over x days old or greater than y' is not usually evidenced and, more subtly, not followed up to ensure that the break cleared in a way that was 'valid'. Exploitation of these 'controls', and the lack of checking that the clearance was valid, has been at the root of multiple unauthorized trading incidents and market abuse. That said, a key benefit of DLs should be to eliminate reconciliations – the policy and follow-up controls will remain

Numeric Observation

Examples include:

- The economics of a trade may require a floating rate (eg, federal funds rate) to be observed each day;
- A trade is valued each day and the value stored.

An observation may lead to a subsequent action (eg, calculate next cashflow based on federal funds rate observation).

Action Observation

Examples include:

- An option contract may require notice of exercise to be evidenced. This is a yes/no/not advised observation.
- A policy (eg, review all trades with a day-one value above 'value').

Exercises

These are events that are defined by the economics of a contract. Such contracts are all types of option – ie, where the final state depends on whether a defined choice (option) in the contract is taken up (exercised).

	Event Identifier	Party	Party	Quantity	Economics	Comments
Exercise (Cash)						
Before	3fb9	P1	P2	Q	Option(5dc4)	Cashflow is defined by economics of position and market data
After	3fb10	P1	P2	c,ccc.cc	Cash	Eg, credit derivative, option, swaption etc. This is termination without negotiation
Exercise (Physical, Knock In)						
Before	4fy4	P1	P2	Qty(xxx)	Option(5dc4)	Position (xxY) is defined by Position (xxx) and market data
After	4fy5	P1	P2	Qty(xxY)	Option(8wtx)	Eg, FX option, credit derivative, physical swaption etc. This is an amendment without negotiation
Exercise (Physical, Knock Out)						
Before	4fy4	P1	P2	Qty(xxx)	Option(5dc4)	The contract disappears
After	4fy5	-o-	-o-	-o-	-o-	

Cash Exercise

In a cash exercise, the option contract turns into a cashflow that is calculated from terms in the contract, and data observed after the contract is executed. The exercise results in a termination event (as defined previously). There is no negotiation of terms, as the terms have already been defined in advance. The exercise may require the holder of the optionality to 'advise' the other party. This is a type of observation (also see above).

Physical Exercise

This is where the event changes the form of the contract in a pre-defined way.

For example:

- A swaption when exercised changes into an interest rate swap;
- A knock-in FX option changes into a vanilla FX option;
- An FX option changes into an FX trade.

In summary, if there is value in an option contract, then it will turn into cash or an asset – where the ownership is transferred on a settlement system. For example, the contract may require the delivery (transfer of ownership) of a security (eg, bond, equity) or a commodity (eg, metal, oil, etc).

Knock-out Exercise

A knock-out is where the contract ceases to exist as a result of an observation (other than time passing[18]).

Dependent Events Summary

- Dependent events are defined by the contract terms.
- Dependent events cause the same underlying changes as independent events (eg, the exercise of an option for cash results in a cashflow, as does an independent event (ie, negotiated termination).
- They can therefore also be considered 'contracted' events.

Transfers and Exchanges

This section shows how products are built up out of primitive events in the same space [P1, P2, Q, (E)] – ie, how the economics set (E) in this space is defined. In order to show how this is done, it is useful to start with cash, then bonds/securities, and then derivatives[19].

[18] Note: The concept of maturity/expiry of a contract is where the contract goes past a time, defined as the end of the contract. It is debatable if this is an event at all

[19] In fact, mirroring the way such products developed over history

Transfer

This represents a transfer of cash between two parties[20]. Cash is defined by an identifier (in this case, an ISO currency code[21]). Cash could move for any number of reasons (settlement of a trade, a coupon, dividend, collateral, account transfer, etc).

The quantity needs to be defined in terms of units – eg, sterling is in pounds and pence. This design removes the rounding problems encountered when settlements are calculated at a precision that is meaningless (ie, it is impossible to have a £2,456.432525 settlement, whatever any system calculating settlement values thinks. There are therefore finite precision issues that need to be designed in at the beginning.

	Event Identifier	Party	Party	Quantity	Economics	Comments
Transfer						
Cash	6re1	P1	P2	Q	GBP	Where GBP is the ISO currency code

Exchange

	Event Identifier	Party	Party	Quantity	Economics	Comments
Exchange						
FX Trade						Exchange of cash. FX rate is calculated from cashflows at 1.25
Cash	6re1	P1	P2	100	GBP	
Cash	6re1	P1	P2	(125)	USD	

An exchange (of 'things') is the fundamental concept of any individual contract. The example shows the simplest form of financial contract – an agreement to exchange one currency for another. The two cashflows are connected by a common event identifier.

When the FX trade comes to settle, two transfers (as defined above) are processed[22].

Key points:

- In a DL, the same mechanism for event identification is used – ie, an exchange is a collection of things that is identified on the underlying ledger.

[20] Implementation details of how this happens are not important (ie, if there are one or two records that are linked)

[21] At this point, it is not necessary to distinguish between central bank and commercial money. The point is the transfer is recorded on a ledger

[22] Note: The settlement may be independent or through a cash versus cash process. Either can be implemented conventionally or on a DL, although both are easier on a DL

- An FX trade is defined as the exchange of two cashflows. The rate is computed from the cashflows. This is important, as it means that any number of any size FX trades in the same currency pair can be simply added without loss of precision. If the rate is defined in terms of a finite precision (five decimal places), then it is not possible to add large and small FX trades together with confidence. Similarly, it is possible to have aggregations of similar size trades that will produce a rate that cannot be expressed in a finite precision number.

- Some further notes:

  - The exchange of two cashflows is embedded in the ISO/SWIFT messaging protocols.

  - Implemented well, there is no need for standard settlement instructions and all the overhead that goes into maintaining them.

  - Likewise, implemented well, netting is implicit in the ledger (see original aggregation event above).

Products

Bond Processing as an Example (Definition, Transfer, Sale)

	Event Identifier	Party	Party	Quantity	Economics	Comments
Bond Definition						P5 issues a bond with the following economics
Cash	hx72		P5	1,000,000,000	USD,s	The principle cashflow at the start of the bond
Cash	hx72		P5	(1,000,000,000)	USD,e	The principle cashflow at the start of the bond
Coupon	hx72		P5	1,000,000,000	Coupon(rate,s,e,d,h)	And the coupon in between
Transfer						
Bond	54gt	P1	P2	5,000,000	hx72	Where the bond is defined above and identified as hx72 on the ledger
Exchange						
Bond Trade						
Bond	esft	P3	P4	10,000,000	hx72	Where the bond is defined above
Cash	esft	P3	P4	(9,727,000)	USD	Price is again computed at 97.27

Definition

A bond is defined as a collection of terms (initial and final cashflows and coupon). The collection is tracked on a DL as an event hx72.

Coupon is a function (smart contract) defined on the ledger.

There are additional contractual terms to a bond that can be stored with the base economics defined here. These don't impact the processing of bond transactions through its normal life[23].

There are other notably possibilities – for example, registration of bond securities becomes an approval process by the relevant authority.

Similarly:

- A bond template could be defined and then re-used on each issue of a bond[24].
- Interest amounts are easy to strip from the principal.

Transfer

The bond hx72 can then be transferred between two parties P1, P2 (eg, in a collateral pledge). The mechanism for doing this is identical to transferring cash (ie, the ledger tracks the thing identified by an address, and that address could contain a cash contract, a bond contract or something else). There is therefore no physical difference between the settlement infrastructure for a bond or for cash.

Exchange

The exchange (purchase/sale of a bond) transaction is exactly the same structure as for an FX trade. When due for settlement, the individual elements turn into transfers and are processed on same settlement functionality.

Implications of Bond Processing Example

- The definition of a bond can be conducted on the same infrastructure that processes them – ie, no separate product data systems.
- 'The processing of cash and bonds (or any other security or asset) is logically identical to processing cash. On a DL, it can be physically identical.'
- Definition of a 'thing' and transactions in a 'thing' are tracked by the same addressing mechanism on a DL.
- Some further notes:
    - Consider how complicated this is on conventional infrastructure.
    - Cross-currency processing (eg, an investor buys a US dollar bond for sterling) is a simple composition of an FX and bond trade, as defined above.

[23] A bond contract may contain a lot more, including terms relating to interest cover (really an option trigger), and collateral (ie legal liens). All these may fall into the same structure

[24] For example, every Treasury or gilt issuance is fundamentally under the same legal framework with amount, coupon and maturity added

Simple Derivative Contracts

Derivatives contracts are fundamentally agreements to exchanges of asset streams (usually cash) into the future.

	Event Identifier	Party	Party	Quantity	Economics	Comments
Derivatives						
IRS	53g3					An IRS is an exchange of fixed and floating coupons
Coupon	53g4	P1	P2	10,000,000	Coupon(Q,float,s,e,d,h)	Assumed entered into at zero value (ie, no cash at inception)
Coupon	53g5	P1	P2	(10,000,000)	Coupon(Q,fix,s,e,d,h)	
CDS Single Name	gy4j					
Security	gy4j	P1	P2	5,000,000	hx72	Underlying security defined above
Coupon	gy4j	P1	P2	5,000,000	Coupon(rate,s,e,d,h)	Either 100 or 500bp
Coupon	gy4j	P1	P2	67,435	USD,s	Calculated at inception
Swaption	hw34					
Option	hw34	P1	P2	10,000,000	Strike(float)	Where strike refers to the floating rate in the IRS below
Coupon	hw34	P1	P2	10,000,000	Coupon(Q,float,s,e,d,h)	
Coupon	hw34	P1	P2	(10,000,000)	Coupon(Q,fix,s,e,d,h)	

An IRS is the exchange of two coupon streams (ie, a fixed leg and floating leg). The event to identify an IRS is no different to any other on a DL – it is the address 53g3 in this example.

A single-name credit default swap (CDS) connects together a bond (as defined previously) with a coupon flow and a fee at inception (as the coupons are standardized).

A swaption wraps an optionality term around an IRS.

Index Derivatives Contracts

	Event Identifier	Party	Party	Quantity	Economics	Comments
Derivatives						An index is defined as a standardized collection
Index Definition	ued6					Define an index in terms of previously defined bonds
Constituent 1	ued6		P5	01	hx72	The bond defined above
Constituent 2	ued6		P43	02	j9rs	Another bond
etc.	ued6					
Constituent n	ued6		P34	On	7eds	Another bond
Execute a trade in an INDEX						
Security	t6c8	P1	P2	5,000,000	ued6	Refer to the index defined above
Coupon	t6c8	P1	P2	5,000,000	Coupn(Q,fix,s,e,d,h)	
Cash	t6c8	P1	P2	67,435	USD,s	

The example above shows the definition of an index and a trade in that index in the same format.

The index is defined in terms of quantities of previously defined bonds (eg, hx72 above). The creation of the index has an identifier (ued6).

Once the index has been created, there is no difference between transactions in an index (ued6) to a transaction in any other 'thing' already defined.

Notes:

- The fundamental function is the ability to store collections of things.
- An index is merely a collection of things. It is usually defined by a single entity (eg, FTSE (SE100), Dow (DJIA), IHS Markit (iTraxx), etc) and is referenced and used by multiple participants for different purposes.
- The underlying mechanism that is important is the ability to create an index identifier and then add/modify the elements of an index independently.
- This same mechanism can be used to create other collections:
    - A basket trade is a non-standard index;
    - A structured derivative/note is a collection;
    - More subtly, a trading book is a collection that changes every day;
    - Indeed, a private client portfolio is also a collection that changes every day.
- The general point is picked up in the fractal-symmetry design section. All elements/instruments of the financial markets are collections of simpler terms.

Sophisticated Derivatives

More complex derivatives are hierarchies of simple derivatives – ie, they are collections of simpler derivatives. Their lifecycle is to change from more complex derivatives into simpler derivatives as time progresses. This suggests an elegant way of defining the contracts to simplify their lifecycle.

	Event Identifier	Party	Party	Quantity	Economics	Comments
Knock In Option Lifecycle						
Execute Trade						
Knock In	y76j	P1	P2	Q	f1(rate)	A knock in (if rate=KI in the period)
Option	y76j	P1	P2	Q	f2(rate)	The strike of option (if rate = S in the period)
FX Trade	y76j	P1	P2	Q1	USD	The underlying FX
FX Trade	y76j	P1	P2	Q2	GBP	
Trade Knocks In						
~~***Knock IN***~~	~~y76j~~	~~P1~~	~~P2~~	~~Q~~	~~f1(rate)~~	The knock In is exercised
Option	y76j	P1	P2	Q	f2(rate)	The underlying option is now active
FX Trade	y76j	P1	P2	Q1	USD	
FX Trade	y76j	P1	P2	Q2	GBP	
Trade Knocks In						
~~***Knock In***~~	~~y76j~~	~~P1~~	~~P2~~	~~Q~~	~~f1(rate)~~	
~~***Option***~~	~~y76j~~	~~P1~~	~~P2~~	~~Q~~	~~f2(rate)~~	The underlying option is excercised
FX Trade	y76j	P1	P2	Q1	USD	Underlying option FX is now active
FX Trade	y76j	P1	P2	Q2	GBP	

This example shows a knock-in FX option (KIFXO). A KIFXO is an FX option that comes into existence if a certain condition is met. It is a compound option.

Starting from the inside or underlying and working out:

- There is an underlying FX trade.
- There is an option on that FX trade.
- The option only exists if a certain condition exists.

If defined this way, then the lifecycle events consist merely of deleting certain contract terms if the conditions are met:

- If the knock-in (KI) conditions are met, then the KI conditions disappear (strike through) and the FXO remains.
- If the FX options (FXO) conditions are met, then the FXO conditions disappear and a simple FX trade is left.

Defining contracts in this way simplifies lifecycle processing[25]. It should allow an option to be defined on any underlying and all exercise events to be processed the same way, removing terms/conditionality as each of the conditions are met.

Portfolio Processes

Up to now, this paper has discussed events and individual transactions. This section looks at the portfolios that result from transactions and the processes that run on them, starting with collateral.

Other portfolio processes (for the purpose of finance, capital, risk, etc) are similar to the collateral process – for those purposes, they just need different aggregations/calculations (different functions – ie, implementable as smart contracts).

Portfolio processes exploit the lineage property inherent in DLs (O(n) = f(O(<n)) explored above in this paper.

Collateral Process

	Event Identifier	Party	Party	Quantity	Economics	Value	Agregate	Collateral Call	Value	Call	Value Post Call	Agregate	Collateral Call
Open Positions													
Derivative Positions													
OTC	fed6	P1	P4	Q1	IRS	(1,453)			(1,453)				
OTC	5r7x	P1	P4	Q2	IRS	342			342				
Market Hedges							(1,111)					(1,111)	
Security	fy43	P1	P57	Q3	Bond								
Cash	8re6	P1	P34	Q4	Future								
Collateral Held													
Security	4df6	P1	P4	Q5	Bond	834			834	64	898		
Cash	l41g	P1	P4	Q6	USD	233			233	(20)	213		
							1,067					1,111	
Collateral Call								(44)					0

The collateral process runs in three steps:

- Step 1: Determine portfolio of OTC derivatives with the client and the collateral currently posted by this client. This is the state of the ledger at a point in time that a party wishes to make collateral call[26].

[25] Note: Compare this to some implementations where the different states in the trade lifecycle are processed in different systems with different data formats and complex event processing between the systems

[26] Operationally, this normally happens daily at the end of day, but it doesn't need to be. In a DL environment, there doesn't need to be a physical snapshot of the positions and this could certainly be done at a higher frequency

- Step 2: Determine credit exposure by valuing the OTC derivatives and the collateral. There may be several calculations:
  - 2.1: Determine the variation margin (VM) required (usually net of consistent valuations);
  - 2.2: Determine the initial margin (IM) required – potentially a more complicated calculation based on an estimate of time to close-out the exposure in event of default by a party (eg, the margin period of risk);
  - 2.3: Collateral optimization – look at VM and IM, and determine which assets to pledge or call.
- Step 3: Move and record collateral (cash, securities or other asset type move on a settlement[27]).

It could be efficient, if allowable, to net collateral movements with cashflow movements related to the same derivatives portfolio – in other words, move collateral and coupons on the same date to eliminate timing risk and (potentially) reduce the amount of margin required.

Finance Process

	Event Identifier	Party	Party	Quantity	Economics	Value	Agregate	Trading Balance sheet	Profit & Loss	LEX	RWA	CAPITAL	SACCR	FINANCIAL ACCOUNTS
Open Positions														
Derivative Positions														
OTC	fed6	P1	P4	Q1	IRS	(1,453)								
OTC	5r7x	P1	P4	Q2	IRS	342								
Market Hedges							(1,111)							
Security	fy43	P1	P57	Q3	Bond	543								
Cash	8re6	P1	P34	Q4	Future	345								
Collateral Held							888							
Security	4df6	P1	P4	Q5	Bond									
Cash	l41g	P1	P4	Q6	USD									
Retained Earnings								(223)						

A finance process[28] takes the accumulated direct observations on the state of the world and presents them in a different format.

[27] As discussed previously, there is no logical difference between cash and securities settlement processes, and both can be on DL

[28] Finance processes are really just book-keeping processes

The example shows a trading balance sheet for the same example as the collateral process above. The derivatives positions are combined with market hedges to produce a retained earnings/net assets number. All other finance and risk processes are of this type (profit and loss is the change in retained earnings). This offers the opportunity to run a consistent set of books and records for all numbers that are generated and recorded by the organization[29]. The outputs can then be recorded at summary level in a general ledger[30].

The logical end state if underlying transactions are all recorded on a DL is that all disclosures are smart contracts on the ledger (ie, one per country). The smart contracts are written by the accountants. There is minimal auditing[31] due to the lineage and distribution characteristics of the ledger.

Evidencing Professional Judgement

In general, senior management sets policies and wants to have evidence that the policies they have put in place have been applied[32]. In this process, evidence of professional judgement is collected by recording what an appropriate person has done what they have been professionally employed to do. This is an important requirement in any business. This evidence is used in control reviews (ie, to determine whether the results make sense).

Such policies can also be set up as smart contracts within a DL:

- Review all trades over X notional or Y inception value;
- Review all portfolios over X;
- Review all assets with a credit rating less than A- every three months and check;
- Risk limits are also examples of policies.

[29] Fundamentally, what standards such as BCBS 239 require

[30] That is if a general ledger is really still required with the advent of these new technologies. However, general ledgers are likely to continue to be required for a while, as the concept is embedded in much legislation

[31] Financial accounts require judgements to be made. Auditors should primarily be focused on independently assessing the reasonableness of these adjustments. DLs provide a means of recording and evidencing professional judgement, be it the management or the auditor

[32] BCBS 239

THE COMPLETE MODEL

This section classifies DLs and brings together the event, product, portfolio and CDM concepts discussed so far.

Classes of Distributed Ledger

Basic Distributed Ledger

$$N_1(f_D,[O]) \equiv N_2(f_D,[O]) \equiv ..\equiv N_n(f_D,[O]) \quad (Distribution)$$

Where:

- N_x is a node on the DL.
  - Each node is an independently running peer 'copy' of all other nodes on the ledger[33];
  - All nodes are identical;
  - Anyone can set up and run a node.
- f_D are a simple set of pre-defined functions (ie, smart contracts) distributed to all nodes on the ledger.
- $[O]$ is the set of observations that constitute the ledger.
  - All nodes on the ledger contain the same set of observations $[O]$.
- This defines a non-permissioned DL (eg, common cryptocurrencies).
  - Fundamentally, such a cryptocurrency runs on a few functions f_D.
    - Create (x units at P7 as a result of completing a mining transaction);
    - Transfer (x units from P1 to P2 where P1 and P2 are key addresses);
    - Split (take x units from P1 and split to P2, P3);
    - Aggregate (take units at address P4,P5 and combine on address P6).
  - The non-permissioned distribution of $[O]$ and the ability of anyone to set up a node makes this type of DL unsuitable for regulated financial business.

Relating this to the ISDA CDM:

- Fundamentally, the independent events are just functions f_D.

[33] Note: The mechanics of how this is done is not relevant to this discussion

Extendable Distributed Ledger – Lineage

$$N_1(f_{D+U},[O]) \equiv N_2(f_{D+U},[O]) \equiv ..\equiv N_n(f_{D+U},[O]) \quad (Distribution)$$

$$where\ [O_{x,m}] = f_U([o_{1,<m}],[o_{2,<m}],\ldots) \quad (Lineage)$$

Where:

- f_U are user-definable functions (ie, smart contracts) that can be added to the DL.
- $[o_1]$ $[o_2]$ are observation sets on the ledger.
- $[O_{x,m}]$ is a derived observation computed from other observations on the ledger by applying the function f_U.
- This lineage property is a formal programing paradigm:
    - Namely, that a function can only be defined in terms of previously defined functions or direct observations of the real world;
    - It is a formal object oriented ('OO') paradigm[34];
    - It incorporates an application programing interface (API) definition[35].
- The location of 'derived' observations on a DL is defined by the combination of the location of the existing lower-level observations and the location of the function code executed to derive subsequent observations. The lineage property therefore defines a non-colliding extensible address space on the ledger[36].

Lineage is the key enabler of value in DLs:

- It allows the elimination of multiple systems and processes (ie, with no need to have routines to extract, filter, transform, transmit, load, transform, filter, load, and the subsequent associated reconciliations).
- It eliminates batch processing:
    - Consider the classic finance/risk end-of-day process.
    - End-of-day valuations consist of the tuple valuation [open positions, market data, valuation function].

[34] The objects are defined from the bottom up, as in a formal paradigm. In general, OO systems fail because they do not start from the fundamental building blocks and build so inheritance works. Each OO system builder defines a local OO model (that is different to the OO model defined by any other team solving the same problem). In practice, it is very rare for different teams to be solving exactly the same problem. Therefore, each of their models are inconsistent. This fundamental inconsistency is then hidden behind a set of messaging APIs. The cost of the messaging and API is therefore baked in

[35] APIs invoke a function and pass parameters by value or by reference. This is the lineage definition. APIs are problematic where the underlying data set is potentially different (ie, a by reference call). For example, if we ask an API to value the portfolio of trades [S], then the API doesn't know what the portfolio is, so the results cannot be relied upon

[36] Compare this to the difficulty of defining a top down canonical relational model across multiple systems

- All of these could be persisted on the ledger.
  - Open positions: the state of the ledger (strictly, the last transaction to be included).
  - Market data: is a set of observations (and functions[37]) that is marked as the end of day set.
  - Valuation function: is the code run (from simple price*quantity via discounted cashflows through to the most complicated option pricing model).
- Tracking changes in the process of valuation – ie, *valuation(open positions, market data, valuation function, time)* – is the holy grail of risk/finance management and control, and is implicit on a well-implemented distributed ledger.
  - For example, the function *change(open positions, today, yesterday)* is the portfolio change, the function valuation*(change(open positions, today, yesterday))* is then 'new deal' profit and loss.
  - 'Carry' and 'market move' can be similarly defined.
- Risk sensitivities are then computed similarly as changes in valuation, but as a function of change in market data (eg, PV01, etc) rather than just time.
- There is an opportunity to deal with timing problems without embedding bi-temporality everywhere.
  - For example, if there is a late booked trade (ie, one entered into a ledger after the time of an end-of-day report[38]), then it can be effected by overlaying the changes onto the end-of-day report to compute the impact. A good implementation removes the need for any manual corrections/adjustments.
  - There is no need to complicate every trade/position record by making them manage timing problems. Catch the small/tiny population of offending trades and overlay into the correct period.

- It eliminates product taxonomies:
  - The definition of a product is from the market data required to value it, rather than a complex mapping of the elements of the contract terms to a taxonomy.
  - This approach is future proof and generic.
  - This approach also has applications for finance processes (eg, mark to market (liquid/observable price), mark to model (liquid/observable inputs) and mark to model (unobservable process). Namely, this classification is a function of market data, rather than the contract. Observability changes over time.

[37] For example, a yield curve is an observation set and a set of interpolation calculations

[38] It should be noted that any migration to a DL is likely to reduce the instances of late-booked trades

- It allows back-testing in a live environment:
    - To assess the impact of a new calculation/function f_{New} in comparison to an existing $f_{Current}$, the function is put into the ledger with a request that it computes using the same data on the ledger.
    - It can compute as fast as resources allow.
    - The results will not collide, as the location of the results are defined by f_{New}.
    - However, the results of f_{New} and $f_{Current}$ can be compared, as all shared elements of the ledger location are the exactly the same.
    - Compare this to the complexity of testing new functions across existing infrastructure (eg, running parallel testing environments, 'fudging clocks', feeding reconciliation processes, etc. Indeed, back-testing is rarely done. Instead, parallel testing, which takes a lot of time, is the norm.
- It is the basis of a solution for BCBS 239:
    - It provides lineage from board/management information packs all the way to the transactions.
- See fractal symmetry picture for an illustration of lineage.

Mapping to the ISDA CDM:

- Dependent events and product definitions map to f_U.
- All dependent events are lineage examples.
- All product definitions are then derived observations.
    - Namely, there is conceptually a product template provided (a derived observation because it is made up of lower observations (eg, an IRS is two coupons)).
    - When a transaction is executed, the product template (an observation on the ledger) is populated with economics (ie, direct observations). A transaction is then created that combines the location of the template with the set of direct observations (the economics) to produce a derived observation, which is the transaction.
- All portfolio processes are likewise derived observations upon transactions.

Permissioned Distributed Ledger

$$N_{P1}(f,[O_{P1}]) \equiv N_{P2}(f,[O_{P2}]) \equiv N_{P3}(f,[O_{P3}])..\equiv\cdots \equiv N_{P1,P2}(f,[O_{P1,P2}])$$

Where:

- $P_1,P_2,P_3,\ldots,P_n$ are controlled parties on the ledger.
- N_{P1} is a permissioned node that only holds observations that P_1 is a party to $[O_{P1}]$.
- $N_{P1\,P2}$ is a node that P_1 and P_2 have given permission to hold their transactions (a 'regulatory node').
- The same mechanics holds true for functions f.
- If P_1 is further subdivided, then it provides the mechanism for more granular permissions (eg, to specific trading books or branches).

Mapping to the ISDA CDM:

- No changes to the model to add such a feature.
- Permissioning provides information security.
- Functions f could be permissioned in this way also.
    - From the ISDA CDM event, product and process functions would probably be distributed across all ledger participants.
    - Proprietary functions (eg, valuation models) would be restricted to individual participants.

Distribution to Computational Nodes

$$N_{P1}(f,[O_{P1}]) \equiv \{n_{P1}(f,[o_1])+n_{P1}(f,[o_2])..\equiv\cdots \equiv n_{P1}(f,[o_m])\}$$

$$where\ [O_{P1}]= [o_1]+ [o_2]+ ..+ [o_m]$$

Where:

- $n()$ is a sub node – ie, a node that contains a subset of $N()$.
- Re-use of the permissioning process from a permissioned DL.
    - In a permissioned ledger, the observations and code are distributed to nodes based on party IDs (ie, contents of the ledger).

- In a distribution to computation nodes, the observations and functions are distributed:
- Either on a numerical basis;
- Or based on the contents:
  - All observations of this type (product) to a sub node:
  - All observations and functions that use this market data.
- Combination of the above.

- The objective is to distribute across physical processing cores using the same distribution mechanism used across the ledger itself for other purposes.

Mapping to ISDA CDM:

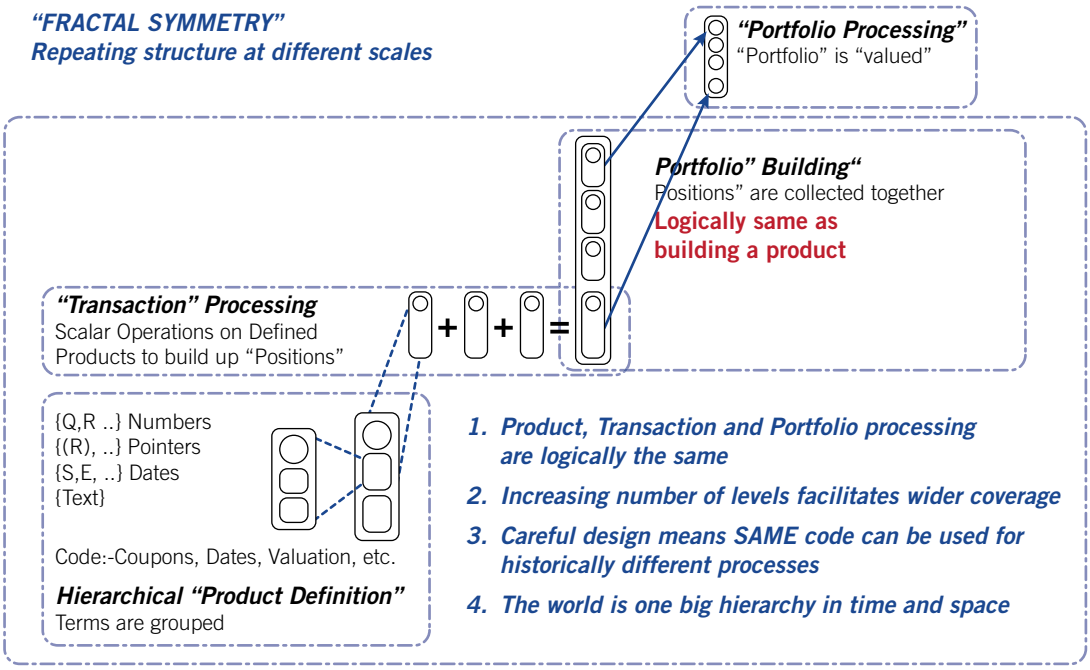
- None.
- But emphasizes the worth of parametric processing (ie, minimize the transactions).

Fractal Symmetry

Fractal symmetry is the implication of the lineage property of DLs defined above.

$$[O_{x,m}] = f_U([o_{1,<m}],[o_{2,<m}],\ldots) \quad (Lineage)$$

Fractal Symmetry – Picture Example



Starting in the bottom left corner:

- A hierarchical product definition builds a product from simpler defined elements:
    - Elements are defined (date, numbers, etc);
    - Functions are built up from pre-defined elements;
    - Products are built up of predefined functions;
    - 'Elements', 'functions' and 'products' is a hierarchy where 'elements', 'functions' and 'products' are labels for the different levels in the lineage equation, $O(n)=f(O(<n))$. The actual name is irrelevant;
    - This produces a product template that can then be used repeatedly:
        - An IRS (as defined previously);
        - A bond, CDS, index, and so on, as defined previously.
- Transaction processing:
    - Picks up previously defined templates and executes transactions in them.
    - There are two types:
        - Transactions in identical/fungible products (eg, bonds, equities, listed derivatives and standardized derivatives (eg, CDS)). The transactions are aggregated together to produce a single element in a portfolio.
        - Transactions in unique/non-fungible products (eg, IRS trades) cannot be simply aggregated, and therefore each transaction is an element in the resulting portfolio.
- Portfolio building:
    - A portfolio is built up of transactions in fungible and non-fungible products;
    - A portfolio consists of a collection number of elements;
    - Portfolio building is logically the same as building a product – ie, it is the same lineage equation $O(n)=f(O(<n))$ at a different scale;
    - Conceptually, treat a trading portfolio as a single complicated/structured and changing trade;
    - Fractal symmetry is the same logical process at different scales.

- Portfolio processing:
    - Evaluates a portfolio at a given time.
    - Examples include:
        - Net cashflow calculation – all positions in the portfolio are evaluated for cashflows today *(eval(value, today))*. All resulting cashflows are added together to produce a net number.
        - Valuation – all positions in the portfolio are valued as at today *(eval(value, today))*. All resulting values are added together to produce a net number.
    - Portfolio processes are also fractal – ie, the only change between the 'cashflow' and valuation' example above is the parameter 'cash' and 'value'.

Implemented well, fractal symmetry minimizes and parameterizes the functions within a common framework. This reduces the amount of code that needs to be written and provides flexibility (eg, adding new products and valuation methods).

The illustration above has unified this into the same framework:

- Product definition and portfolio processes;
- Portfolio cashflow generation and portfolio valuation.

Fractal Symmetry – Product Example

Further levels - Legal entities, etc.		
Aggregate (Level 6)		
	Net value	25,365
Aggregate (Level 5)		
	Valued Portfolio (Portfolio, Market Data, Model)	
	IRS Executed	54,363
	IRS Executed	4,545
	IRS Executed	(33,543)
Aggregate (Level 4)		
	Portfolio	
	IRS Executed	
	IRS Executed	
	IRS Executed	
Aggregate (Level 3) - Trade is executed - Template is populated		
	IRS Executed (dr54)	
	Coupon(USD,100M,3M,20170824,20270824,1.7653,A/360,NY,MF)	
	Coupon(USD,(100M),3M,20170824,20270824,LIBOR,A/360,NY,MF)	
Aggregate (Level 2) - Constructed out of Level 1 Elements		
	IRS "Template"	
	Coupon(CCY,Quantity,Frequency,Start,End,Fixed,DayCount,Holidays,Convention)	
	Coupon(CCY,(Quantity),Frequency,Start,End,Float,DayCount,Holidays,Convention)	
Aggregate (Level 1) - constructed out of BASE elements		
	Coupon(CCY,Quantity,FrequencyStart,End,Rate,DayCount,Holidays,Convention)	
BASIC ELEMENTS (Level 0)		
	DayCount["A/360","A/365","30/360", etc.)	
	Convention["ModFollowing", etc.]	
	Dates[ ]	
	Holdays[cities:Dates]	
	CCY["USD","GBP", etc.]	
	Rate[Fixed, (Float)]	

ABOUT ISDA

Since 1985, ISDA has worked to make the global derivatives markets safer and more efficient.

Today, ISDA has over 875 member institutions from 68 countries. These members comprise a broad range of derivatives market participants, including corporations, investment managers, government and supranational entities, insurance companies, energy and commodities firms, and international and regional banks. In addition to market participants, members also include key components of the derivatives market infrastructure, such as exchanges, intermediaries, clearing houses and repositories, as well as law firms, accounting firms and other service providers. Information about ISDA and its activities is available on the Association's website: www.isda.org.