Building Smart Contracts

Use of smart contracts could bring significant efficiencies by automating the execution of certain provisions within derivatives agreements. Ciarán McGonagle and Finn Casey Fierro describe how a smart contract can be developed based on the ISDA Digital Assets Definitions[1]

In 2023, the International Swaps and Derivatives Association (ISDA) published the ISDA Digital Asset Derivatives Definitions. Considering the digital and decentralised characteristics of this asset class, it was essential to develop a standardised contractual framework that can be easily integrated into blockchain systems. This framework could then form the foundation for smart contracts that support the automation of contractual operations and events relating to these transaction types. ISDA drafted the Digital Asset Definitions with this objective in mind, using formal, controlled drafting structures that allow the operative terms of transactions to be more easily translated into machine-executable code.

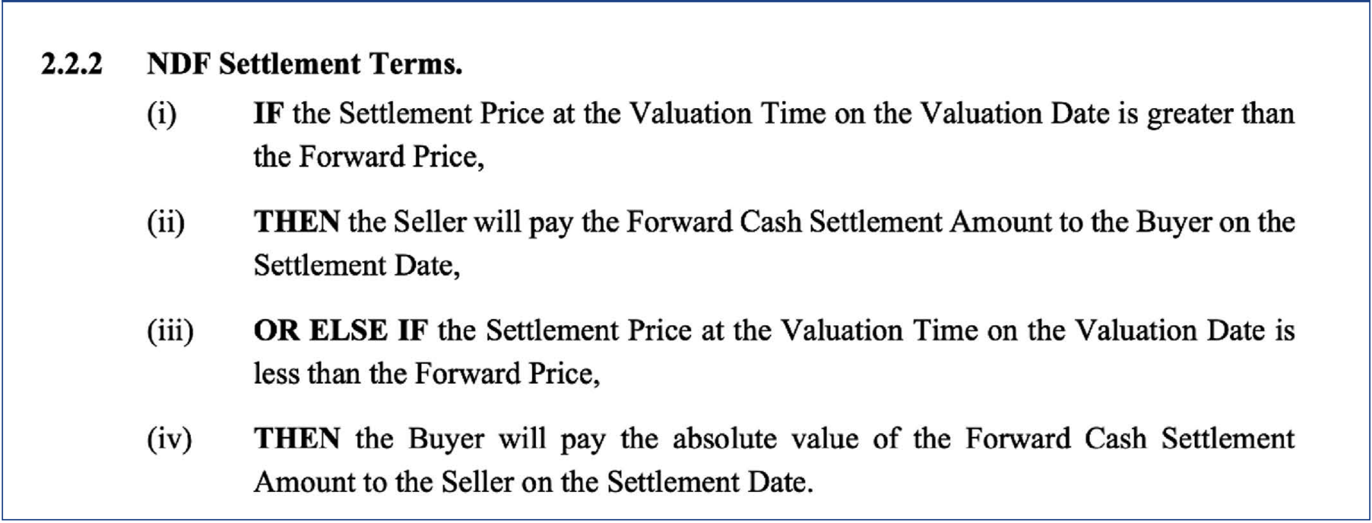
Working in collaboration with Finn Casey Fierro, then a master's student at University College London, ISDA contributed its various contractual, operational and data standards to support the development of an Ethereum-based smart contract based on the ISDA Digital Asset Definitions.

This article provides an overview of the initiative, explaining how the ISDA Digital Asset Definitions were developed and how the Common Domain Model (CDM) plays a central role in providing a standardised digital blueprint for representing these operations and events within distributed ledger technology (DLT) systems and platforms. The article also explains how the smart contract was designed, exploring how the contract was translated into code, how it was structured and the functionality it provides.

The ISDA Digital Asset Definitions

The ISDA Digital Asset Definitions provide a standard contractual framework for trading cash-settled options and forwards referencing bitcoin and ether. The definitions utilise a controlled semantic structure, framing likely-to-be-automated contractual provisions as a set of parameterised conditions and outcomes. They do so by expressing operative clauses using natural language drafting, combined with consistent conditional logic terms such as 'IF', 'AND', and 'THEN'.

FIGURE 1: PAYING THE SETTLEMENT AMOUNT UNDER A CASH-SETTLED FORWARD



2.2.2 NDF Settlement Terms.

(i) **IF** the Settlement Price at the Valuation Time on the Valuation Date is greater than the Forward Price,

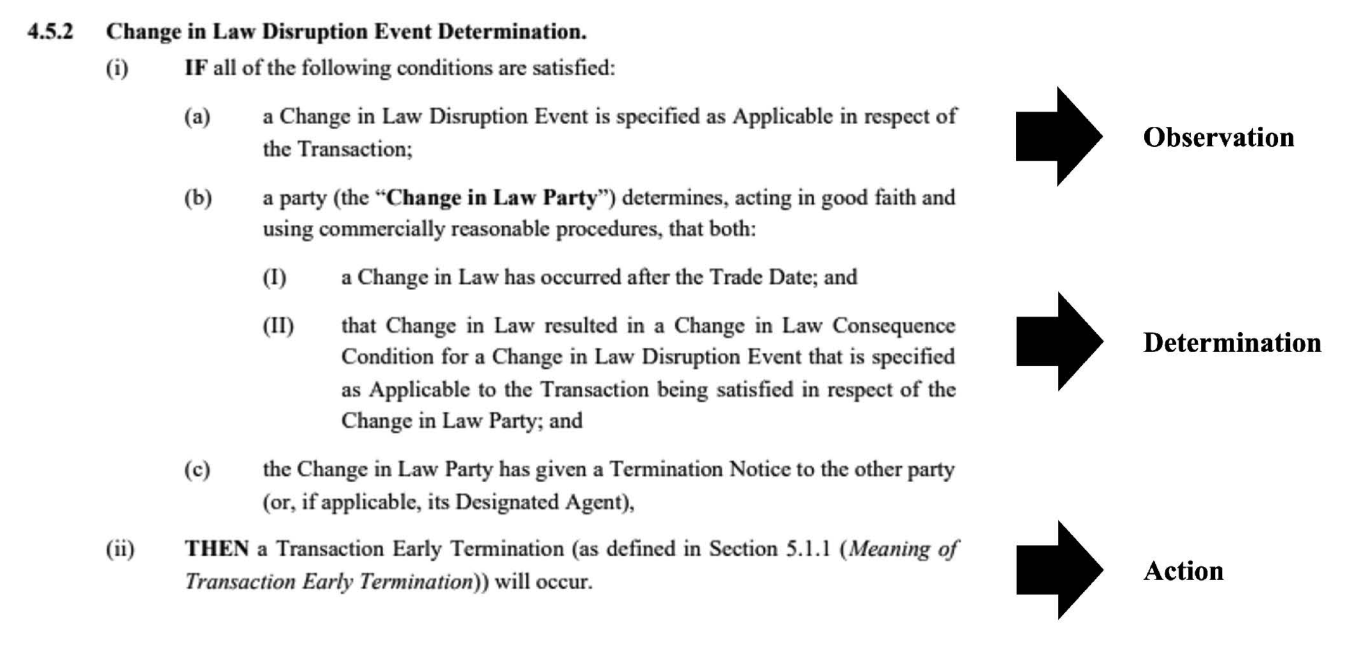
(ii) **THEN** the Seller will pay the Forward Cash Settlement Amount to the Buyer on the Settlement Date,

(iii) **OR ELSE IF** the Settlement Price at the Valuation Time on the Valuation Date is less than the Forward Price,

(iv) **THEN** the Buyer will pay the absolute value of the Forward Cash Settlement Amount to the Seller on the Settlement Date.

1 This is an edited and updated version of an article that was first published in the January 2024 issue of Butterworths Journal of International Banking and Financial Law

FIGURE 2: DETERMINING AN EARLY TERMINATION FOLLOWING A CHANGE IN LAW DISRUPTION EVENT



4.5.2 Change in Law Disruption Event Determination.

(i) **IF** all of the following conditions are satisfied:

(a) a Change in Law Disruption Event is specified as Applicable in respect of the Transaction;

(b) a party (the "**Change in Law Party**") determines, acting in good faith and using commercially reasonable procedures, that both:

(I) a Change in Law has occurred after the Trade Date; and

(II) that Change in Law resulted in a Change in Law Consequence Condition for a Change in Law Disruption Event that is specified as Applicable to the Transaction being satisfied in respect of the Change in Law Party; and

(c) the Change in Law Party has given a Termination Notice to the other party (or, if applicable, its Designated Agent),

(ii) **THEN** a Transaction Early Termination (as defined in Section 5.1.1 (*Meaning of Transaction Early Termination*)) will occur.

For example, the clause set out in Figure 1 determines which party will be required to pay the forward cash settlement amount under a cash-settled forward.

This clause is expressed as a pair of conditions (prefixed by 'IF' and 'OR ELSE IF' statements) with associated consequences (prefixed by 'THEN' statements), allowing it to be expressed as a function within a smart contract.

The operation of these clauses (or functions) can also be affected by the occurrence of external events. For instance, if a settlement price cannot be determined, the operation might be rendered inoperable.

The ISDA Digital Asset Definitions provide mechanisms for addressing such events (for example, through the application of a pre-agreed fallback price source). These events will need to be implemented or reflected within the smart derivatives contract to ensure their safe and effective operation. The ISDA Digital Asset Definitions adopt the framework proposed by Clack and McGonagle in *Smart Derivatives Contracts: the ISDA Master Agreement and the automation of payments and deliveries*[2], expressing these events as a series of steps:

- Observation: What data should be observed to determine whether an event has occurred?
- Determination: How is the event determined?
- Action: What action is permitted/required following the determination of an event?

Figure 2 illustrates this concept. The change in law disruption event provides parties with the ability to terminate a transaction in the event it has become illegal for a party to perform its obligations.

If change in law is specified as applicable, then a party can consult the definition of a change in law disruption event to confirm the precise parameters within which such an event can occur. Once the specific circumstances of an event have been observed (for example, a law has passed in a party's jurisdiction banning the trading of digital asset derivatives), a party may then determine that the change in law disruption event has occurred. Again, this determination must be made within the precise parameters of the contract.

Finally, some action may be taken – in this case, the termination of the affected transaction(s). As illustrated, the ISDA Digital Asset Definitions provide a hierarchal series of conditions, all of which must be fulfilled for the relevant action to be taken. While this event has only a single potential consequence (ie, termination), other events may have a series of elective consequences (ie, actions the parties may take) to address the impact of an event – including, for example, calculation agent determination or the designation of some pre-agreed fallback. This logical sequence clearly differentiates between potentially automatable steps like observation and steps like determination and action, which might need human discretion.

The CDM

The CDM provides a standardised, digital blueprint detailing the lifecycle events and processes associated with derivatives trades and other financial products. This unified representation ensures that a given trade or lifecycle event is understood and represented in the same way across the industry and bridges the narrative complexity of derivatives contracts and the logical precision of code.

Rossetta DSL (the domain-specific language used within CDM) provides different components for representing data. These are ⟶

2 Smart Derivatives Contracts: the ISDA Master Agreement and the automation of payments and deliveries, Christopher D. Clack, Ciaran McGonagle: https://arxiv.org/abs/1904.01461

The ISDA Digital Asset Definitions provide a standard contractual framework for trading cash-settled options and forwards referencing bitcoin and ether. The definitions utilise a controlled semantic structure, framing likely-to-be-automated contractual provisions as a set of parameterised conditions and outcomes

⟶ referred to as 'types.' Types include 'basic types' (eg, Boolean values or text strings) and 'data types' (specific data objects being modelled, such as a price or a date).

Each data type may contain a series of attributes, specifying the parameters within which information relating to that data type can be captured. For example, if the relevant data type is designed to capture the date of a transaction, multiple dates cannot be added (due to cardinality restrictions) and neither can incongruous strings of text (eg, elephant, yellow).

The formalised structure and strict parameterisation of defined terms within the ISDA Digital Asset Definitions make this translation process much more efficient. For example, each defined term used within the forward cash settlement amount definition can be separately defined, independent of the operation or interpretation of any other term. This facilitates their expression as distinct objects within the parameters of the relevant CDM data type.

These data types can then be expressed as variables or inputs to functions designed to execute the operative provisions of the definitions.

For example, the formula for calculating the forward cash settlement amount is:

Forward Cash Settlement Amount = Multiplier x (p – Forward Price) x Currency Conversion Factor

Where 'p' means the Settlement Price at the Valuation Time on the Valuation Date

This could be expressed in CDM by first defining the name of the function:

func ForwardCashSettlementAmount:

[calculation]

Then by defining the necessary inputs:

inputs:

multiplierValue number (1..1)

p number (1..1) // Settlement Price at the Valuation Time on the Valuation Date

forwardPrice number (1..1)

currencyConversionFactor number (1..1)

Each input has a cardinality of (1..1), indicating that each input must appear and can only appear once. Each input is also associated with a basic type, indicating the format in which the data object must appear (eg, a decimal number).

The output is defined in a similar way:

output:

cashSettlementAmount number (1..1)

The CDM also supports the definition of variables (known as aliases) to express discrete functions that are used within the fully defined function – in this case, the function for determining the forward amount is expressed as p (ie, the settlement price at the valuation time on the valuation date) less the forward price:

alias forwardAmount: p - forwardPrice

With each of the variables now defined, the function could be expressed:

set cashSettlementAmount:

*multiplier * forwardAmount * currencyConversionFactor*

This function can then be combined with other relevant data types, acting as an input in the function that determines which party is required to make payment.

Translation from contract to code

The development of these standardised, digital representations of contractual obligations, events and variable names establishes a robust foundation for the development of smart contract-based platforms within the derivatives market.

The ISDA Digital Asset Definitions have been drafted in a modular structure, using controlled natural language constructs and term definitions that are easily translatable into programming parameters.

This initiative has highlighted clear parallels between the legal prose and programming languages. In contracts, defined terms are frequently employed to refer to specific concepts throughout various operative clauses in a consistent way. Similarly, in programming, functions use parameterised inputs to dictate outcomes based on specific conditions. These parameterised inputs in programming are analogous to the defined terms in legal contracts. The functions in programming can also be compared to operative clauses in contracts, as both serve to establish potential outcomes contingent upon specific conditions, guided by defined terms.

The provision set out in Figure 1 illustrates a method for reading and processing conditions in a structured sequence. It starts by examining each non-indented line for a specific condition. If a condition is met, the process moves to the related indented text for further instructions. If not, it moves to the next line to check whether a specific condition is met. In this example, the 'OR ELSE IF' operation is intended to denote that the two conditions are mutually exclusive – ie, there cannot be a scenario where both parties are required to pay[3]. This approach closely resembles the structure of conditional statements found in Turing-complete programming languages like Python, JavaScript or (for the purposes of Ethereum smart contracts) Solidity, where decisions are made based on Boolean logic (see Figure 3).

Here, the code would be executed if either of the relevant conditions were true. The similarity makes direct translation between the definitions and the code immediate and clear for both lawyers and programmers. The approach also helps standardise interpretation across different developers and programming languages, as syntax in the appropriate language can be used, referencing the CDM representation of its component data types and attributes for consistency.

The smart contract project

The smart contract project focused on the creation of a framework for developing smart contracts on the Ethereum blockchain based on the standards described in this article. It included the development of non-deliverable forwards and options in accordance with the terms of the ISDA Digital Asset Definitions and the implementation of those transactions based on the CDM representation of the relevant data types and attributes.

To facilitate interaction with the smart contract, the project also involved the creation of a graphical interface for the entire contract process, designed for all contract parties, showing event notifications and live price updates from an external oracle, without relying on a traditional database.

The proposed smart contract templates are organised in a hub-and-spoke structure, with a 'confirmation' smart contract at the core, linked to various 'logic module' smart contracts, each representing a part of the agreement with legal enforceability and connections to broader natural language contracts.

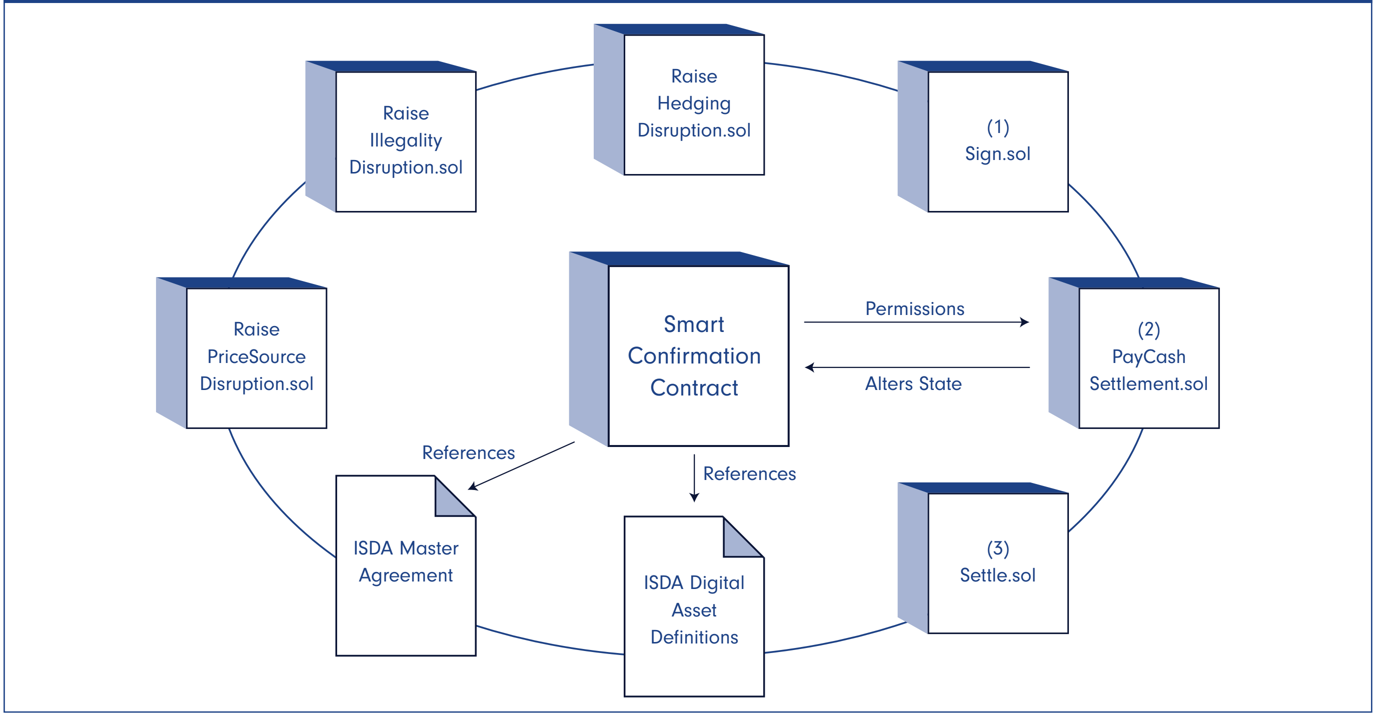
This was achieved through development of a composable architecture, separating the agreement's data and logic into distinct

FIGURE 3: SOLIDITY CODE FOR PAYING THE SETTLEMENT AMOUNT UNDER A CASH-SETTLED FORWARD

```
if (settlementPrice > forwardPrice) {

  settlementAmount = settlementPrice - forwardPrice;
  buyer.transfer(settlementAmount);

}
else if (settlementPrice ≤ forwardPrice){

  settlementAmount = forwardPrice - settlementPrice;
  seller.transfer(settlementAmount);

}
```

3 There are scenarios where the forward cash settlement amount may be zero, in which case neither party is required to pay. However, given that no obligation arises in such a scenario, it is arguably unnecessary to expressly allow for this in either the contract or the smart contract

FIGURE 4: MODULAR INTEGRATED SMART CONTRACT ARCHITECTURE[4]


Raise Illegality Disruption.sol
Raise Hedging Disruption.sol
(1) Sign.sol
Raise PriceSource Disruption.sol
Smart Confirmation Contract
Permissions
Alters State
(2) PayCash Settlement.sol
References
References
ISDA Master Agreement
ISDA Digital Asset Definitions
(3) Settle.sol

→ templates. This structure is referred to as a modular integrated smart contract architecture (MISCA) (see Figure 4).

The MISCA holds all transactional information and the state of the agreement within a central smart contract, referred to as a smart confirmation contract. Like its namesake in the current natural language ISDA documentation framework, a smart confirmation contract contains the economic terms and other information necessary for the trade, such as who the parties are, the applicable price source and any elections that have been specified within the applicable ISDA product definitions (in this case, the ISDA Digital Asset Definitions).

The smart confirmation contract also stores new types of transaction information. For instance, it will store information relating to the contract's 'state' (ie, where in the automation process the parties are). It separates the lifecycle into a list of Boolean 'questions' that can resolve to true or false at different times. For instance, the 'isSigned' Boolean indicates whether the contract has been signed. If true, the system proceeds to verify 'isCashSettlementAmountFixed'. If that is also true, it checks 'isCashSettlementAmountPaid', continuing this sequence through the contract's conditions. The MISCA therefore simplifies an agreements lifecycle into a series of straightforward questions for intuitive understanding.

This structure can also reference any overarching natural language contracts or documents. Within blockchains where data storage is expensive, this could mean simply storing text of pre-established documents, like that of

string isda_product_definition = "The ISDA Digital Asset Derivatives Definitions"

alongside a hash (digital fingerprint) of the document. As blockchain storage becomes more economical, it will be possible to store the actual documents on the chain, enabling smart contracts to link directly to the full documents, streamlining the reference process.

Finally, the smart confirmation contract contains a list of addresses to smart contract code that have permission to alter the agreement's state. Seen as the central hub of the MISCA (see Figure 4), the smart confirmation contract is a repository for all agreement variability, making it an ideal structure for data standards like the CDM.

The smart contract code will be contained in single-function smart contract templates. These templates are called logic modules. These logic modules would interact with the smart confirmation contract to read its data and, in certain scenarios, to update the agreement's state (as in Figure 4).

For example, a logic module might contain the function that determines the forward cash settlement amount. In this scenario, upon the valuation time on the valuation date, the logic module would read certain data contained within the smart confirmation contract and any specified oracles to establish the relevant inputs. It would then execute the function to determine the forward cash settlement amount and store both the value and new state on the smart confirmation contract. Once determined, another logic module could be initiated to read the forward cash settlement amount and automate payment of the relevant amount to the correct party, utilising traditional payment methods or blockchain transacted stablecoins or tokens.

To promote readability and standardisation, each logic module has been structured as follows:

4 In Figure 4, the cubes symbolise smart contracts, while the pages depict natural language documents. Smaller cubes indicate logic modules, with the central, larger cube representing the smart confirmation contract. Arrows display interactions and numbering displays order of execution required

- **Data retrieval and initiation**

Fetches the necessary data from the smart confirmation contract, databases or oracles and constructs the relevant programming objects (eg, fetching a price source address or initiating a price-source oracle object that can be used to retrieve a price from an identified price source).

- **Restrictions**

Validates the permissions of the calling entity to execute the module's functions (eg, an 'if' check that restricts access to the buyer of the contract only). This section controls access to the subsequent sections, assessing the agreement's status and allowing lifecycle events to be processed in the appropriate, sequential order.

- **Logic body**

This is the smart contract code that implements the operative clauses (eg, the execution of a digital asset/token transfer between the buyer and seller). This section, if so designated and with careful auditing, can also execute functions from external smart contracts, enabling potential interoperability among other transaction and contract types.

- **Setting of the state**

The modification of the agreement's state in the smart confirmation contract (eg, updating the state of a derivatives contract to 'settled' after successful execution.

This structure is exemplified in Figure 5, which displays a logic module written for the project in the solidity programming language. It is called 'pay forward cash settlement amount'. For the legal robustness present in the controlled natural language of the ISDA Digital Asset Definitions, it directly represents the operative clause of payment from the definitions within its logic body (Figure 5).

This logic module has been engineered using the Hardhat development framework tailored for the Ethereum virtual machine. The deployment entailed initialising an associated smart confirmation contract designed for non-deliverable forwards to be used as an address in the parameter '_confirmationContractAddress' (Figure 5). The smart confirmation contract references an ISDA Master Agreement, the ISDA Digital Asset Definitions and various permissioned logic modules. After defining the transactional terms, including identification of the buyer and seller, the smart contracts were integrated into a decentralised web application for interaction with parties.

FIGURE 5: LOGIC MODULE WRITTEN FOR PAY FORWARD CASH SETTLEMENT AMOUNT[5]

```
contract PayForwardCashSettlementAmount {

    function payForwardCashSettlementAmount(address _confirmationContractAddress) public {

        // 1. Data retrieval and initiation
        INDF_CONFIRMATION Confirmation = INDF_CONFIRMATION(_confirmationContractAddress);
        (int256 settlementPrice,
         int256 forwardPrice,
         address buyer,
         address seller,
         address settlementCurrencyAddress) = Confirmation.getTransactionalTerms();
        (bool cashSettlementAmountFixed, bool cashSettlementAmountPaid, bool terminated) = Confirmation.getState();
        IERC20 settlementCurrency = IERC20(settlementCurrencyAddress);

        // 2. Restrictions
        require(msg.sender == buyer || msg.sender == seller, "INVALID_CALLER");
        require(cashSettlementAmountFixed == false && cashSettlementAmountPaid == false, "INVALID_CALL");
        require(!terminated, "CONTRACT_TERMINATED");

        // 3. Logic body (a respresentation derived from the Digital Asset Definitions)
        int256 forwardCashSettlementAmount = settlementPrice - forwardPrice;
        if (settlementPrice > forwardPrice) {
            settlementCurrency.transferFrom(seller, buyer, uint256(forwardCashSettlementAmount));
        }
        else if (settlementPrice <= forwardPrice){
            int256 absoluteForwardCashSettlementAmount = -forwardCashSettlementAmount;
            settlementCurrency.transferFrom(
                buyer,
                seller,
                uint256(absoluteForwardCashSettlementAmount)
            );
        }

        // 4. Setting of the state
        Confirmation.setCashSettlementAmountPaid(true);
    }

}
```

5 In Figure 5, a logic module for paying a forward cash settlement amount is displayed, with its function separated into the standardised structure through comments and spacing. Within the logic body, a clause from the ISDA Digital Asset Definitions is represented

The smart contract project demonstrates how the formal, controlled drafting structure and format used within the ISDA Digital Asset Definitions allows for straightforward translation into machine-readable data and machine-executable functions

→ The MISCA might therefore be seen as treating operative clauses like that of software development libraries, enabled by the principle of composability – the system design philosophy that enables the construction of complex systems from simplified, well-defined components. Each module is self-contained, maintaining its individual autonomy and facilitating its replacement or reusability. For instance, if both parties agree to change the 'settle version 1' module to 'settle version 2' halfway through a derivatives lifecycle, this will simply require switching their addresses on the smart confirmation contract and the amendment would be instantly implemented.

One of the key benefits of this structure is that logic modules are not required to have their own state. This allows them to be agreement agnostic, capable of being combined and used across many different types of products and agreements. This allows an incremental approach to adoption. For example, some parties may begin by adopting cryptographic signatures for traditional transactions and therefore will opt for only a 'sign' logic module.

Others may seek greater automation of the entire derivatives lifecycle, stacking 'sign', 'settle' and 'raise disruption' logic modules and including all of them in the smart confirmation contract. Each logic module can also have its own versioning system, allowing for widely accepted and verified logic modules to be used more widely across the market and for lawyers to verify their legal effect more easily.

The MISCA addresses risks associated with smart contracts, including the systemic risks that may emerge from faults in a single smart contract spreading through a network of interconnected contracts. It does so by ensuring that issues in any single smart contract can be isolated and replaced without adversely affecting the entire system. Furthermore, new or amended terms can be incorporated within the structure without requiring modification to other parts of the contract. Perhaps most importantly, it standardises how smart derivatives contracts are written, prioritising readability and intuition.

Conclusion

This article explains how a smart contract can be developed within the standardised contractual, operational and data frameworks developed by ISDA. It proposes a methodology – the MISCA – for constructing a smart derivatives contract template based on these standards. Within the MISCA framework, a smart confirmation contract acts as the central hub, communicating among many different logic modules and referencing natural language agreements.

The formal logic within each logic module relies on the standardised, formal expression of the logic contained within the underlying contractual framework. The smart contract project demonstrates how the formal, controlled drafting structure and format used within the ISDA Digital Asset Definitions allows for straightforward translation into machine-readable data and machine-executable functions. The CDM further enforces standardisation through the expression of these functions and data attributes within a composable, industry-standard domain model that can be used to represent (and automate) the terms of many different financial products and agreements.

Although the structural features described simplify translating the contractual logic found in the ISDA Digital Asset Definitions, constructing a standardised smart derivatives contract framework for a blockchain is neither trivial nor straightforward. When developing a smart contract, it's important to consider not just the design of the contract itself but also the characteristics and advantages of the blockchain it will operate on. Factors such as the blockchain's architecture, the level of automation it supports (the Ethereum virtual machine has certain limitations in this regard), the existing user and developer community, how user friendly the programming and interaction with the code are and the clarity of the code's semantic structure all play a crucial role.

The development of a smart contract is therefore a tailored process, adapted to the specific capabilities and context of the blockchain platform being used, moving away from a generic one-size-fits-all approach. However, during development of the smart contract project, it was determined that there are steps users can take to mitigate many of the issues presented by these limitations, providing both the necessary functionality and design efficiency to users.

The extent of automation that is achievable will ultimately depend not only on whether automation is effective but whether it is efficient. This determination will, in each case, continue to require the input of both lawyers and technology developers. IQ

Ciarán McGonagle is assistant general counsel, smart contracts & digital assets, at ISDA. Finn Casey Fierro is a Web3 Solutions Architect at Null Technologies Ltd.