



INPUT | OUTPUT

# A Formal Specification of the Cardano Ledger

## Deliverable SL-D5

Jared Corduan <jared.corduan@iohk.io>

Polina Vinogradova <polina.vinogradova@iohk.io>

Matthias Gudemann <matthias.gudemann@iohk.io>

*Project:* Shelley Ledger

*Type:* Deliverable

*Due Date:* 15<sup>th</sup> October 2019

*Responsible team:* Formal Methods Team

*Editor:* Jared Corduan, IOHK

*Team Leader:* Philipp Kant, IOHK

Version 1.0  
8th October 2019

Dissemination Level		
PU	Public	✓
CO	Confidential, only for company distribution	
DR	Draft, not for general circulation	

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Notation</b>	<b>3</b>
<b>3</b>	<b>Cryptographic primitives</b>	<b>5</b>
<b>4</b>	<b>Addresses</b>	<b>9</b>
<b>5</b>	<b>Protocol Parameters</b>	<b>11</b>
5.1	Updatable Protocol Parameters . . . . .	11
5.2	Global Constants . . . . .	11
<b>6</b>	<b>Transactions</b>	<b>15</b>
<b>7</b>	<b>Update Proposal Mechanism</b>	<b>18</b>
7.1	Protocol Parameter Update Proposals . . . . .	18
<b>8</b>	<b>UTxO</b>	<b>21</b>
8.1	UTxO Transitions . . . . .	21
8.2	Deposits and Refunds . . . . .	26
8.3	Witnesses . . . . .	27
<b>9</b>	<b>Delegation</b>	<b>30</b>
9.1	Delegation Definitions . . . . .	30
9.2	Delegation Transitions . . . . .	33
9.3	Delegation Rules . . . . .	35
9.4	Stake Pool Rules . . . . .	39
9.5	Delegation and Pool Combined Rules . . . . .	41
<b>10</b>	<b>Ledger State Transition</b>	<b>44</b>
<b>11</b>	<b>Rewards and the Epoch Boundary</b>	<b>47</b>
11.1	Overview of the Reward Calculation . . . . .	47
11.2	Example Illustration of the Reward Cycle . . . . .	48
11.3	Helper Functions and Accounting Fields . . . . .	48
11.4	Stake Distribution Calculation . . . . .	49
11.5	Snapshot Transition . . . . .	51
11.6	Pool Reaping Transition . . . . .	53
11.7	Protocol Parameters Update Transition . . . . .	55
11.8	Complete Epoch Boundary Transition . . . . .	58
11.9	Rewards Distribution Calculation . . . . .	60
11.10	Reward Update Calculation . . . . .	64
<b>12</b>	<b>Blockchain layer</b>	<b>68</b>
12.1	Block Definitions . . . . .	68
12.2	MIR Transition . . . . .	70
12.3	New Epoch Transition . . . . .	70
12.4	Tick Nonce Transition . . . . .	73
12.5	Update Nonce Transition . . . . .	73
12.6	Reward Update Transition . . . . .	74
12.7	Chain Tick Transition . . . . .	75
12.8	Operational Certificate Transition . . . . .	77

12.9	Verifiable Random Function . . . . .	78
12.10	Overlay Schedule . . . . .	80
12.11	Protocol Transition . . . . .	84
12.12	Block Body Transition . . . . .	86
12.13	Chain Transition . . . . .	88
12.14	Byron to Shelley Transition . . . . .	92
<b>13</b>	<b>Software Updates</b>	<b>95</b>
<b>14</b>	<b>Transition Rule Dependencies</b>	<b>96</b>
<b>15</b>	<b>Properties</b>	<b>98</b>
15.1	Header-Only Validation . . . . .	98
15.2	Validity of a Ledger State . . . . .	100
15.3	Ledger Properties . . . . .	101
15.4	Ledger State Properties for Delegation Transitions . . . . .	103
15.5	Ledger State Properties for Staking Pool Transitions . . . . .	104
15.6	Properties of Numerical Calculations . . . . .	105
15.7	Preservation of Value . . . . .	106
15.8	Non-negative Deposit Pot . . . . .	109
<b>16</b>	<b>Leader Value Calculation</b>	<b>113</b>
16.1	Computing the leader value . . . . .	113
16.2	Node eligibility . . . . .	113
<b>17</b>	<b>Errata</b>	<b>114</b>
17.1	Total stake calculation . . . . .	114
17.2	Active stake registrations and the Reward Calculation . . . . .	114
17.3	Stability Windows . . . . .	114
17.4	Reward aggregation . . . . .	114
17.5	Byron redeem addresses . . . . .	115
17.6	Block hash used in the epoch nonce . . . . .	115
17.7	Deposit tracking . . . . .	115
<b>References</b>		<b>116</b>
<b>A</b>	<b>Cryptographic Details</b>	<b>118</b>
A.1	Warning About Re-serialization . . . . .	118
A.2	Hashing . . . . .	118
A.3	Addresses . . . . .	118
A.4	KES . . . . .	118
A.5	VRF . . . . .	118
A.6	Abstract functions . . . . .	119
A.7	Multi-Signatures . . . . .	119
<b>B</b>	<b>Binary Address Format</b>	<b>120</b>
B.1	Header, first nibble . . . . .	120
B.2	Header, second nibble . . . . .	120
B.3	Header, examples . . . . .	120
B.4	Payloads . . . . .	121
<b>C</b>	<b>Bootstrap Witnesses</b>	<b>121</b>
C.1	Bootstrap Witnesses CBOR Specification . . . . .	121

C.2	Bootstrap Address ID Recovery	121
<b>D</b>	<b>CBOR Serialization Specification</b>	<b>122</b>
<b>E</b>	<b>Implementation of txSize</b>	<b>122</b>
<b>F</b>	<b>Proofs</b>	<b>122</b>

## List of Figures

1	Non-standard map operators	4
2	Cryptographic definitions	5
3	KES Cryptographic definitions	6
4	Multi-signature via Native Scripts	7
5	VRF definitions	8
6	Definitions used in Addresses	10
7	Definitions Used in Protocol Parameters	13
8	Global Constants	14
9	Helper functions for the Protocol Parameters	14
10	Definitions used in the UTxO transition system	16
11	Helper Functions for Transaction Inputs	17
12	Protocol Parameter Update Transition System Types	19
13	Protocol Parameter Update Inference Rules	20
14	Functions used in UTxO rules	22
15	UTxO transition-system types	23
16	UTxO inference rules	25
17	Functions used in Deposits - Refunds	26
18	Functions used in witness rule	28
19	UTxO with witness transition-system types	29
20	UTxO with witnesses inference rules	29
21	Delegation Definitions	32
22	Delegation Transitions	34
23	Delegation Inference Rules	37
24	Move Instantaneous Rewards Inference Rule	38
25	Pool Inference Rule	40
26	Delegation and Pool Combined Transition Type	41
27	Delegation and Pool Combined Transition Rules	42
28	Delegation sequence transition type	42
29	Delegation sequence rules	43
30	Ledger transition-system types	44
31	Ledger inference rule	45
32	Ledger Sequence transition-system types	46
33	Ledger sequence rules	46
34	Helper Functions used in Rewards and Epoch Boundary	49
35	Accounting fields	49
36	Epoch definitions	49
37	Stake Distribution Function	50
38	Snapshot transition-system types	51
39	Snapshot Inference Rule	52
40	Pool Reap Transition	53
41	Pool Reap Inference Rule	54
42	New Proto Param transition-system types	55

43	New Proto Param Inference Rule . . . . .	57
44	Epoch transition-system types . . . . .	58
45	Epoch Inference Rule . . . . .	59
46	Functions used in the Reward Calculation . . . . .	60
47	Functions used in the Reward Splitting . . . . .	61
48	The Reward Calculation . . . . .	63
49	Preservation of Value . . . . .	64
50	Rewards Update type . . . . .	64
51	Reward Update Creation . . . . .	66
52	Reward Update Application . . . . .	67
53	Block Definitions . . . . .	69
54	MIR transition-system types . . . . .	70
55	MIR rules . . . . .	70
56	NewEpoch transition-system types . . . . .	71
57	New Epoch rules . . . . .	72
58	Tick Nonce rules . . . . .	73
59	UpdNonce transition-system types . . . . .	74
60	Update Nonce rules . . . . .	74
61	Reward Update transition-system types . . . . .	74
62	Reward Update rules . . . . .	75
63	Tick transition-system types . . . . .	76
64	Tick rules . . . . .	76
65	OCert transition-system types . . . . .	77
66	OCert rules . . . . .	78
67	Overlay transition-system types . . . . .	81
68	Overlay rules . . . . .	82
69	Protocol transition-system types . . . . .	84
70	Protocol rules . . . . .	85
71	BBody transition-system types . . . . .	86
72	BBody rules . . . . .	87
73	Chain transition-system types . . . . .	89
74	Helper Functions used in the CHAIN transition . . . . .	90
75	Chain rules . . . . .	91
76	Initial Shelley States . . . . .	93
77	Byron to Shelley State Transition . . . . .	94
78	STS Rules, Sub-Rules and Dependencies . . . . .	97
79	Tick Forecast rules . . . . .	98
80	Chain-Head rules . . . . .	99
81	Definitions and Functions for Valid Ledger State . . . . .	101
82	Definitions and Functions for Stake Delegation in Ledger States . . . . .	103
83	Initial Chain State . . . . .	109

## Change Log

Rev.	Date	Who	Team	What
1	2019/10/08	Jared Corduan, Polina Vinogradova and Matthias Gudemann	FM (IOHK)	Initial version (0.1).
2	2019/10/08	Kevin Hammond	FM (IOHK)	Added cover page.
3	2020/11/17	Jared Corduan	FM (IOHK)	Removed unused deliverable outline image, set version to 1.0, and changed the reward calculation so that $\eta$ takes $d$ into account.
4	2021/05/17	Jared Corduan	FM (IOHK)	Added Example Illustration.
5	2021/06/14	Jared Corduan	FM (IOHK)	Added an errata section, fixed a typo in leader check and in the LEDGERS rule index, and synced the reward calculation with the implementation
6	2021/06/17	Jared Corduan	FM (IOHK)	Allow the pool influence parameter $a_0$ to be zero, remove all mentions of deposit decay, sync varibale names with code.
7	2021/08/27	Jared Corduan	FM (IOHK)	Fixed definitions in the header-only validation properties. CHAIN transition did not need to use previous protocol parameters.
8	2021/10/08	Jared Corduan	FM (IOHK)	The function createRUpd should get the pool parameters from the go snapshot. The TICKN rule was missing from the dependency diagram.
9	2021/11/08	Jared Corduan	FM (IOHK)	Fixed typo in the description of variable length encodings.
10	2021/12/13	Jared Corduan	FM (IOHK)	Re-wrote the MIR transitions to be more compact.
11	2022/01/20	Jared Corduan	FM (IOHK)	Fixed error in counting new pools for deposits and an error in the POOLREAP rule.
12	2022/01/26	Jared Corduan	FM (IOHK)	Specify seed operation and seedToSlot.
13	2022/01/31	Jordan Millar, Jared Corduan	FM (IOHK)	Fixed prose regarding the hash used in the epoch nonce. Added an item to the errata regarding this same hash.
14	2022/08/25	Jared Corduan	FM (IOHK)	Specify the txid function in the Appendix. Warn about re-serializing.
15	2023/03/23	Jared Corduan	FM (IOHK)	Add an errata section about individualP deposit tracking.

# A Formal Specification of the Cardano Ledger

Jared Corduan

jared.corduan@iohk.io

Polina Vinogradova

polina.vinogradova@iohk.io

Matthias Güdemann

matthias.gudemann@iohk.io

March 23, 2023

## Abstract

This document provides a formal specification of the Cardano ledger for use in the upcoming Shelley implementation. It is intended to underpin a Haskell executable specification that will be the basis of the initial Shelley release, and represents a core design and quality assurance document. It will be used to define properties and tests, and to provide the basis for strong formal assurance using mathematical proof techniques. The document defines the rules for extending the ledger with transactions that will affect both UTxO and stake delegation. Key properties that have been identified include the preservation of balances, absence of double spend, stakepool registration, and reward splitting.

## List of Contributors

Nicolás Arqueros, Dan Bornside, Nicholas Clarke, Duncan Coutts, Ruslan Dudin, Sébastien Guillemot, Kevin Hammond, Vincent Hanquez, Ru Horlick, Michael Hueschen, Christian Lindgren, Yun Lu, Philipp Kant, Jordan Millar, Jean-Christophe Mincke, Damian Nadales, Ashish Prajapati, Nicolas Di Prima, Andrew Westberg.

## 1 Introduction

This document is a formal specification of the functionality of the ledger on the blockchain. This includes the blockchain layer determining what is a valid block, but does not include any concurrency issues such as chain selection. The details of the background and the larger context for the design decisions formalized in this document are presented in [SL-D1].

In this document, we present the most important properties that any implementation of the ledger must have. Specifically, we model the following aspects of the functionality of the ledger on the blockchain:

**Preservation of value** Every coin in the system must be accounted for, and the total amount is unchanged by every transaction and epoch change. In particular, every coin is accounted for by exactly one of the following categories:

- Circulation (UTxO)
- Deposit pot
- Fee pot
- Reserves (monetary expansion)
- Rewards (account addresses)
- Treasury

**Witnesses** Authentication of parts of the transaction data by means of cryptographic entities (such as signatures and private keys) contained in these transactions.

**Delegation** Validity of delegation certificates, which delegate block-signing rights.

**Stake** Staking rights associated to an address.

**Rewards** Reward calculation and distribution.

**Updates** The update mechanism for Shelley protocol parameters and software.

While the blockchain protocol is a reactive system that is driven by the arrival of blocks causing updates to the ledger, the formal description is a collection of rules that compose a static description of what a *valid ledger* is. A valid ledger state can only be reached by applying a sequence of inference rules and any valid ledger state is reachable by applying some sequence of these rules. The specifics of the semantics we use to define and apply the rules we present in this document are explained in detail in [FM-TR-2018-01] (this document will really help the reader's understanding of the contents of this specification).

The structure of the rules that we give here is such that their application is deterministic. That is, given a specific initial state and relevant environmental constants, there is no ambiguity about which rule should be applied at any given time (i.e. which state transition is allowed to take place). This property ensures that the specification is totally precise — no choice is left to the implementor and any two implementations must behave the same when it comes to deciding whether a block is valid.

## 2 Notation

The transition system is explained in [FM-TR-2018-01].

**Powerset** Given a set  $X$ ,  $\mathbb{P} X$  is the set of all the subsets of  $X$ .

**Sequences** Given a set  $X$ ,  $X^*$  is the set of sequences having elements taken from  $X$ . The empty sequence is denoted by  $\epsilon$ . Given a sequence  $\Lambda$ ,  $\Lambda; x$  is the sequence that results from appending  $x \in X$  to  $\Lambda$ .

**Functions**  $A \rightarrow B$  denotes a **total function** from  $A$  to  $B$ . Given a function  $f$  we write  $f a$  for the application of  $f$  to argument  $a$ .

**Inverse Image** Given a function  $f : A \rightarrow B$  and  $b \in B$ , we write  $f^{-1} b$  for the **inverse image** of  $f$  at  $b$ , which is defined by  $\{a \mid f a = b\}$ .

**Maps and partial functions**  $A \mapsto B$  denotes a **partial function** from  $A$  to  $B$ , which can be seen as a map (dictionary) with keys in  $A$  and values in  $B$ . Given a map  $m \in A \mapsto B$ , notation  $a \mapsto b \in m$  is equivalent to  $m a = b$ . The  $\emptyset$  symbol is also used to represent the empty map as well.

**Map Operations** Figure 1 describes some non-standard map operations.

**Relations** A relation on  $A \times B$  is a subset of  $A \times B$ . Both maps and functions can be thought of as relations. A function  $f : A \rightarrow B$  is a relation consisting of pairs  $(a, f(a))$  such that  $a \in A$ . A map  $m : A \mapsto B$  is a relation consisting of pairs  $(a, b)$  such that  $a \mapsto b \in m$ . Given a relation  $R$  on  $A \times B$ , we define the inverse relation  $R^{-1}$  to be all pairs  $(b, a)$  such that  $(a, b) \in R$ . Similarly, given a function  $f : A \rightarrow B$  we define the inverse relation  $f^{-1}$  to consist of all pairs  $(f(a), a)$ . Finally, given two relations  $R \subseteq A \times B$  and  $S \subseteq B \times C$ , we define the composition  $R \circ S$  to be all pairs  $(a, c)$  such that  $(a, b) \in R$  and  $(b, c) \in S$  for some  $b \in B$ .

**Option type** An option type in type  $A$  is denoted as  $A? = A + \diamond$ . The  $A$  case corresponds to the case when there is a value of type  $A$  and the  $\diamond$  case corresponds to the case when there is no value.

**$:=$**  We abuse the  $:=$  symbol here to mean propositional equality. In the style of semantics we use in this formal spec, definitional equality is not needed. It is meant to make the spec easier to read in the sense that each time we use it, we use a fresh variable as shorthand notation for an expression, e.g. we write

$$s := slot + \text{StabilityWindow}$$

Then, in subsequent expressions, it is more convenient to write simply  $s$ . It is not meant to shadow variables, and if it does, there is likely a problem with the rules that must be addressed.

In Figure 1, we specify the notation that we use in the rest of the document.

$set \triangleleft map = \{k \mapsto v \mid k \mapsto v \in map, k \in set\}$	domain restriction
$set \not\triangleleft map = \{k \mapsto v \mid k \mapsto v \in map, k \notin set\}$	domain exclusion
$map \triangleright set = \{k \mapsto v \mid k \mapsto v \in map, v \in set\}$	range restriction
$map \not\triangleright set = \{k \mapsto v \mid k \mapsto v \in map, v \notin set\}$	range exclusion
$A \Delta B = (A \setminus B) \cup (B \setminus A)$	symmetric difference
$M \mathop{\sqcup} N = (\text{dom } N \not\triangleleft M) \cup N$	union override right
$M \mathop{\sqcupleftarrow} N = M \cup (\text{dom } M \not\triangleright N)$	union override left
$M \cup_+ N = (M \Delta N) \cup \{k \mapsto v_1 + v_2 \mid k \mapsto v_1 \in M \wedge k \mapsto v_2 \in N\}$	union override plus (for monoidal values)

**Figure 1:** Non-standard map operators