

# Serialization Format Specification

Concordium

July 18, 2024

# Contents

<b>Contents</b>	<b>1</b>
<b>1 Preliminaries</b>	<b>2</b>
1.1 Basic Representation Types . . . . .	2
1.2 Groups and fields . . . . .	3
1.3 Cryptographic Types . . . . .	4
1.3.1 Hash . . . . .	4
1.3.2 Ed25519 Signature Scheme . . . . .	4
1.3.3 Pointcheval-Sanders Signature Scheme . . . . .	4
1.3.4 ElGamal Encryption Scheme . . . . .	5
1.3.5 BLS Aggregate Signature Scheme . . . . .	5
1.3.6 Verifiable Random Function . . . . .	5
1.3.7 Discrete Logarithm Proofs . . . . .	6
1.3.8 Credential Registration IDs . . . . .	6
1.3.9 Pedersen Commitments . . . . .	6
1.3.10 Sigma Protocols . . . . .	7
1.3.11 Range proofs . . . . .	8
1.3.12 Other proofs . . . . .	8
1.4 Blockchain Types . . . . .	11
<b>2 Network Serialization</b>	<b>23</b>
2.1 Account Keys . . . . .	23
2.2 Anonymity revoker and identity provider keys . . . . .	24
2.3 Baker keys . . . . .	24
2.4 Transaction . . . . .	25
2.4.1 Block Items . . . . .	25
2.4.2 Transactions . . . . .	26
2.4.3 Credential Deployment . . . . .	27
2.4.4 Chain Updates . . . . .	30
2.4.5 Transaction Payloads . . . . .	37

# Chapter 1

## Preliminaries

### 1.1 Basic Representation Types

BIT	1 bit
A single bit, either 0 or 1. (Bits are always packed into bytes of 8 bits, with the first bit being the most significant bit.)	
WORD8	1 byte
A single 8-bit byte, representing a value in the range $[0, 255]$ .	
WORD16	2 bytes
A 16-bit unsigned integer, big-endian encoding.	
WORD32	4 bytes
A 32-bit unsigned integer, big-endian encoding.	
WORD64	8 bytes
A 64-bit unsigned integer, big-endian encoding.	
INT64	8 bytes
A 64-bit signed integer, big-endian encoding, 2's complement.	
DOUBLE	8 bytes
A IEEE 754 double-precision binary floating-point number (binary64), big-endian encoding.	
VLQ	variable length
An unsigned integer represented as a <a href="#">variable-length quantity</a> in big-endian encoding. The high-order bit of each byte is set if further bytes follow; the remaining 7 bits are part of the data, most significant bits first. (This format allows smaller values to be represented more compactly.) We write $VLQ_n$ to indicate that the value should be representable as an $n$ -bit unsigned integer.	

BYTES( $n$ )	$n$ bytes
An uninterpreted sequence of bytes of length $n$ .	
SHORTBYTES	$2 + \text{length}$ bytes
A sequence of bytes, prefixed by its length represented as a WORD16.	
$\text{length} : \text{WORD16}$	
$\text{data} : \text{BYTES}(\text{length})$	
LONGBYTES	$8 + \text{length}$ bytes
A sequence of bytes, prefixed by its length represented as a WORD64.	
$\text{length} : \text{WORD64}$	
$\text{data} : \text{BYTES}(\text{length})$	
BOOL	1 byte
A boolean. Either true or false.	
$\text{value} : \text{WORD8}$	
It must be the case that $\text{value} \in \{0, 1\}$ . 0 represents false, and 1 represents true.	

## 1.2 Groups and fields

Below we have the serialization of some groups and fields used in several cryptographic types (see also Section ??). In particular,  $\mathbb{G}_1^{\text{BLS}}$  and  $\mathbb{G}_2^{\text{BLS}}$  we denote the subgroups of the BLS12-381 curve as given by Definition ??. By  $\mathbb{F}_{\text{BLS}}$  we denote the finite field with  $|\mathbb{G}_1^{\text{BLS}}| = |\mathbb{G}_2^{\text{BLS}}|$  elements. How exactly these are serialized are documented at [https://docs.rs/crate/pairing/0.15.0/source/src/bls12\\_381/README.md](https://docs.rs/crate/pairing/0.15.0/source/src/bls12_381/README.md).

G1	48 bytes
A point in $\mathbb{G}_1^{\text{BLS}}$ .	
BYTES(48)	
G2	96 bytes
A point in $\mathbb{G}_2^{\text{BLS}}$ .	
BYTES(96)	
F	32 bytes
A scalar in $\mathbb{F}_{\text{BLS}}$ .	
BYTES(32)	

## 1.3 Cryptographic Types

### 1.3.1 Hash

SHA256	32 bytes
A SHA-256 hash.	
BYTES(32)	

SHA3	32 bytes
A SHA3-256 hash.	
BYTES(32)	

### 1.3.2 Ed25519 Signature Scheme

ED25519SIGNPRIVKEY	32 bytes
A private key in the Ed25519 signature scheme.	
BYTES(32)	

ED25519SIGNPUBKEY	32 bytes
A public key in the Ed25519 signature scheme.	
BYTES(32)	

ED25519SIGNATURE	64 bytes
A signature in the Ed25519 signature scheme.	
BYTES(64)	

### 1.3.3 Pointcheval-Sanders Signature Scheme

PSPUBLICKEY	variable length
A public key in the Pointcheval-Sanders signature scheme. In the description below it is assumed that the secret key is $(x, y_1, \dots, y_n)$ .	
G1	
A generator $g_1$ of $\mathbb{G}_1^{\text{BLS}}$ .	
G2	
A generator $g_2$ of $\mathbb{G}_2^{\text{BLS}}$ .	
length1 : WORD32	
Length of below list of $\mathbb{G}_1^{\text{BLS}}$ elements, i.e. $\text{length1} = n$ .	
g1List : length $\times$ G1	
$g_1^{y_1}, g_1^{y_2}, \dots, g_1^{y_n}$	

<b>length2 : WORD32</b>	
Length of below list of $\mathbb{G}_2^{\text{BLS}}$ elements, i.e. <b>length2</b> = $n$ .	
<b>g2List : length2 <math>\times</math> G2</b>	
$g_2^{y_1}, g_2^{y_2}, \dots, g_2^{y_n}$	
<b>G2</b>	
The $\mathbb{G}_2^{\text{BLS}}$ element $g_2^x$ .	

<b>PSSIGNATURE</b>	96 bytes
A Pointcheval-Sanders signature.	
<b><math>2 \times \text{G1}</math></b>	
Two $\mathbb{G}_1^{\text{BLS}}$ elements.	

### 1.3.4 ElGamal Encryption Scheme

<b>ELGAMALPUBLICKEY</b>	96 bytes
An ElGamal public key.	
<b><math>2 \times \text{G1}</math></b>	
Two $\mathbb{G}_1^{\text{BLS}}$ elements of each 48 bytes.	


  

<b>ELGAMALCIPHERTEXT</b>	96 bytes
An ElGamal encryption.	
<b><math>2 \times \text{G1}</math></b>	
Two $\mathbb{G}_1^{\text{BLS}}$ elements of each 48 bytes.	

### 1.3.5 BLS Aggregate Signature Scheme

<b>BLSAGGPUBLICKEY</b>	96 bytes
A public key of the BLS Aggregate Signature Scheme.	
<b>G2</b>	
One $\mathbb{G}_2^{\text{BLS}}$ element.	

### 1.3.6 Verifiable Random Function

 The VRF scheme currently used is not ECVRF-EDWARDS25519-SHA512-TAI, but will be changed to it. The size of keys and proofs will remain the same.	
--	--

<b>VRFPRIVKEY</b>	32 bytes
A private key in the ECVRF-EDWARDS25519-SHA512-TAI VRF scheme.	

BYTES(32)
-----------

VRFPUBKEY	32 bytes
A public key in the ECVRF-EDWARDS25519-SHA512-TAI VRF scheme.	
BYTES(32)	

VRF <sub>PROOF</sub>	80 bytes
A VRF proof in the ECVRF-EDWARDS25519-SHA512-TAI VRF scheme.	
BYTES(80)	

### 1.3.7 Discrete Logarithm Proofs

Discrete logarithm proofs are used to prove knowledge of an Ed25519 secret key.

DLOG25519PROOF	64 bytes
BYTES(64)	

Generic discrete logarithm proof.

DLOGPROOF	variable length
BYTES( $n$ )	

### 1.3.8 Credential Registration IDs

CREDENTIALREGISTRATIONID	48 bytes
BYTES(48)	

### 1.3.9 Pedersen Commitments

COMMITMENT	48 bytes
A Pedersen commitment.	
G1	
One $\mathbb{G}_1^{\text{BLS}}$ element.	

COMMITMENTS	variable length
A list of Pedersen Commitments.	
length : WORD64	
The number of Pedersen commitments.	
commitments : length × COMMITMENT	
The commitments.	

### 1.3.10 Sigma Protocols

COMENCEQWITNESS	96 bytes
A witness for the ComEncEq sigma protocol.	
<b>witness</b> : $3 \times F$	
Three scalars from $\mathbb{F}_{\text{BLS}}$ .	
COMEQSIGWITNESS	variable length
Witness of the ComEqSig sigma protocol.	
<b>witnessRho</b> : $F$	
One scalar from $\mathbb{F}_{\text{BLS}}$ .	
<b>length</b> : <b>WORD32</b>	
Length of below list.	
<b>witnessCommit</b> : $\text{length} \times (F, F)$	
List of pairs of scalars.	
COMMULTWITNESS	160 bytes
$5 \times F$	
Five scalars from $\mathbb{F}_{\text{BLS}}$ .	
COMEQWITNESS	64 bytes
$2 \times F$	
Two scalars from $\mathbb{F}_{\text{BLS}}$ .	
ENCTRANSWITNESS	variable length
<b>ENCRYPTEDAMOUNTTRANSFERPROOF</b> or <b>SECTOPUBAMOUNTTRANSFERPROOF</b> witness.	
<b>witnessCommon</b> : $F$	
One scalar from $\mathbb{F}_{\text{BLS}}$ .	
<b>length1</b> : <b>WORD32</b>	
Length of below list of witnesses.	
<b>witnessEncexp1</b> : $\text{length1} \times \text{COMEQWITNESS}$	
<b>length2</b> : <b>WORD32</b>	
Length of below list of witnesses.	
<b>witnessEncexp2</b> : $\text{length2} \times \text{COMEQWITNESS}$	



### 1.3.11 Range proofs

RANGEPROOF	variable length
A range proof.	
<b>g1Elements</b> : $4 \times G_1$	
Four $G_1^{\text{BLS}}$ elements.	
<b>scalars1</b> : $3 \times F$	
Three scalars from $F_{\text{BLS}}$ .	
<b>length</b> : WORD32	
Length of below list of pairs of $G_1$ elements.	
<b>groupElementList</b> : $\text{length} \times (G_1, G_1)$	
A list of pairs of $G_1^{\text{BLS}}$ elements.	
<b>scalars2</b> : $2 \times F$	
Two scalars from $F_{\text{BLS}}$ .	

### 1.3.12 Other proofs

IDOWNERSHIPPROOFS	variable length
<b>sig</b> : BLINDEDSIGNATURE	
(Blinded) Signature derived from the signature on the pre-identity object by the IP	
<b>commitments</b> : CREDENTIALDEPLOYMENTCOMMITMENTS	
Various commitments used to verify proof.	
<b>challenge</b> : SHA3	
Challenge used in proofs.	
<b>proofIdCredPub</b> : IDCREDPUBWITNESSES	
Witnesses to the proof that the computed commitment to the share contains the same value as the encryption	
<b>proofIpSig</b> : COMEQSIGWITNESS	
Witnesses for proof of knowledge of signature by the Identity Provider.	
<b>proofRegId</b> : COMMULTWITNESS	
Proof that $\text{regId} = \text{prf}_K(x)$ . Also establishes that <b>regId</b> is computed from the prf key signed by the identity provider.	
<b>proofCredCounter</b> : RANGEPROOF	
Proof that the credential counter is less than the maximal number of accounts.	

BLINDEDSIGNATURE	96 bytes
A blinded signature.	
PSSIGNATURE	

IdCredPubWitnesses	variable length
<b>length</b> : WORD32	
Number of ARs having encrypted shares of IdCredPub.	
<b>idCredPubProofEntry</b> : length $\times$ IdCredPubProofEntry	
The proof witnesses.	
IdCredPubProofEntry	100 bytes
<b>arIdentity</b> : ARIdentity	
The AR identity.	
<b>witness</b> : COMENCEqWitness	
The proof witnesses, consisting of three scalars from $\mathbb{F}_{\text{BLS}}$ .	
CredentialDeploymentCommitments	variable length
Commitments used in IdOwnershipProofs.	
<b>prf</b> : COMMITMENT	
Commitment to the prf key.	
<b>credCounter</b> : COMMITMENT	
Commitment to the credential counter.	
<b>maxAcoounts</b> : COMMITMENT	
Commitment to the maximal number of accounts.	
<b>attributes</b> : ATTRIBUTECommitments	
List of commitments to the attributes that are not revealed.	
<b>sharingCoeffs</b> : COMMITMENTS	
List of commitments to the coefficients of the polynomial used to (secret) share idCredSec.	
AttributeCommitments	variable length
A list of commitments to the attributes that are not revealed.	
<b>length</b> : WORD16	
The number of attribute commitments.	
<b>attributeCommitments</b> : length $\times$ ATTRIBUTECommitment	
The attribute commitments.	
AttributeCommitment	49 bytes
A attribute tag and a Pedersen commitment to its value.	
<b>attributeTag</b> : WORD8	
Tag identifying which attribute is inside the commitment.	
<b>commitment</b> : COMMITMENT	
The Pedersen commitment to the attribute.	

ENCRYPTEDAMOUNTTRANSFERDATA	variable length
Data that will go onto an encrypted amount transfer.	
<b>remainingAmount</b> : ENCRYPTEDAMOUNT	
Encryption of the remaining amount.	
<b>transferAmount</b> : ENCRYPTEDAMOUNT	
Encryption of the amount to be sent.	
<b>index</b> : WORD64	
The index such that the encrypted amount used in the transfer represents the aggregate of all encrypted amounts with indices < 'index' existing on the account at the time. New encrypted amounts can only add new indices.	
<b>proofs</b> : ENCRYPTEDAMOUNTTRANSFERPROOF	
A collection of all the relevant proofs.	

ENCRYPTEDAMOUNTTRANSFERPROOF	variable length
Proof that an encrypted transfer data is well-formed	
<b>challenge</b> : SHA3	
Challenge used in below sigma proof.	
<b>accounting</b> : ENCTRANSWITNESS	
Witness for the proof that accounting is done correctly, i.e., remaining + transfer is the original amount.	
<b>transferAmount</b> : RANGEPROOF	
Proof that the transfered amount is correctly encrypted, i.e., chunks small enough.	
<b>remainingAmount</b> : RANGEPROOF	
Proof that the remaining amount is correctly encrypted, i.e., chunks small enough.	

SECTOPUBAMOUNTTRANSFERDATA	variable length
Data that will go onto an encrypted amount transfer.	
<b>remainingAmount</b> : ENCRYPTEDAMOUNT	
Encryption of the remaining amount.	
<b>transferAmount</b> : AMOUNT	
Amount to be sent.	
<b>index</b> : WORD64	
The index such that the encrypted amount used in the transfer represents the aggregate of all encrypted amounts with indices < 'index' existing on the account at the time. New encrypted amounts can only add new indices.	
<b>proofs</b> : SECTOPUBAMOUNTTRANSFERPROOF	
A collection of all the relevant proofs.	

SECToPUBAMOUNTTRANSFERPROOF	variable length
Proof that a secret to public transfer data is well-formed	
challenge : SHA3	
Challenge used in below sigma proof.	
accounting : ENCTransWITNESS	
Witness for the proof that accounting is done correctly, i.e., remaining + transfer is the original amount.	
remainingAmount : RANGEPROOF	
Proof that the remaining amount is correctly encrypted, i.e., chunks small enough.	

## 1.4 Blockchain Types

ACCOUNTADDRESS	32 bytes
An account address.	
BYTES(32)	
SEQUENCENUMBER	8 bytes
The sequence number of a transaction on an account.	
WORD64	
UPDATESEQUENCENUMBER	8 bytes
The sequence number of a chain update.	
WORD64	
ENERGY	8 bytes
An amount of energy (NRG).	
WORD64	
AMOUNT	8 bytes
An amount of CCD, expressed in micro-CCD.	
WORD64	
AMOUNTFRACTION	4 bytes
A fraction of an amount represented as parts per hundred thousand.	
fraction : WORD32	
fraction $\leq 100000$ .	

ENCRYPTEDAMOUNT	192 bytes
An encrypted amount in two chunks.	
$2 \times \text{ELGAMALCIPHERTEXT}$	
PAYLOADSIZE	4 bytes
The size of a transaction payload in bytes.	
WORD32	
TRANSACTIONEXPIRYTIME	8 bytes
Expiry time of a transactions in seconds since the UNIX epoch.	
WORD64	
TRANSACTIONTIME	8 bytes
Transaction time in seconds since the UNIX epoch.	
WORD64	
TIMESTAMP	8 bytes
Point in time in milliseconds since the UNIX epoch.	
WORD64	
DURATION	8 bytes
A duration in milliseconds.	
WORD64	
DURATIONSECONDS	8 bytes
A duration in seconds.	
WORD64	
VALIDATORID	8 bytes
Identifier of a validator.	
WORD64	
ELECTIONDIFFICULTY	4 bytes
Election difficulty parameter in parts per hundred thousands.	
WORD32	
EXCHANGERATE	16 bytes
An exchange rate (e.g., microCCD/Euro or Euro/Energy). Infinity and zero are disallowed.	
numerator : WORD64	

denominator : WORD64	
----------------------	--

MINTRATE	5 bytes
A base-10 floating point number representation. The value is $\text{mantissa} * 10^{(-\text{exponent})}$ .	
mantissa : WORD32	
exponent : WORD8	

DESCRIPTION	variable length
Name, url and decription of either an anonymity revoker or identity provider.	
nameLength : WORD32	
name : BYTES(nameLength)	
urlLength : WORD32	
url : BYTES(urlLength)	
descriptionLength : WORD32	
description : BYTES(descriptionLength)	

ARINFO	variable length
Information on a single anonymity revoker held by the IP.	
arIdentity : ARIDENTITY	
description : DESCRIPTION	
publicKey : ARPUBLICKEY	

ARIDENTITY	4 bytes
A number (uniquely) identifying an anonymity revoker. <b>Must be non-zero.</b>	
WORD32	

IPINFO	variable length
Public information about an identity provider.	
ipIdentity : IPIDENTITY	
description : DESCRIPTION	
publicKeys : IPPUBLICKEYS	

IPIDENTITY	4 bytes
A number (uniquely) identifying an identity provider.	
WORD32	

ROOTUPDATE	variable length
<p>Root updates are the highest kind of updates. They can update every other set of keys, even themselves. They can only be performed by Root level keys. One of the following:</p> <div>ROOTUPDATE<sub>ROOTKEYS</sub></div> <div>ROOTUPDATE<sub>LEVEL1KEYS</sub></div> <p>Protocol versions 1–3:</p> <div>ROOTUPDATE<sub>LEVEL2KEYSV0</sub></div> <p>Protocol versions 4 onwards:</p> <div>ROOTUPDATE<sub>LEVEL2KEYSV1</sub></div>	
HIGHERLEVELKEYS	variable length
A list of higher level keys (either root or Level 1 keys).	
<b>length</b> : WORD16	
Number of keys.	
<b>keys</b> : length × UPDATEVERIFYKEY	
The keys.	
<b>threshold</b> : WORD16	
The threshold. Must be non-zero.	
ROOTUPDATE <sub>ROOTKEYS</sub>	variable length
Updating the root keys.	
<b>type</b> = 0 : WORD8	
<b>keys</b> : HIGHERLEVELKEYS	
ROOTUPDATE <sub>LEVEL1KEYS</sub>	variable length
Updating the Level 1 keys.	
<b>type</b> = 1 : WORD8	
<b>keys</b> : HIGHERLEVELKEYS	
ROOTUPDATE <sub>LEVEL2KEYSV0</sub>	variable length
Updating the Level 2 keys, in protocol versions 1–3.	
<b>type</b> = 2 : WORD8	
<b>authorization</b> : AUTHORIZATIONSV0	

<b>ROOTUPDATE<sub>LEVEL2KEYSV1</sub></b>	variable length
Updating the Level 2 keys, from protocol version 4 onwards.	
<b>type = 2 : WORD8</b>	
<b>authorization : AUTHORIZATIONSV1</b>	

<b>LEVEL1UPDATE</b>	variable length
Level 1 updates are the intermediate update kind. They can update themselves or level 2 keys. They can only be performed by Level 1 keys. One of the following:	
<div>LEVEL1UPDATE<sub>LEVEL1KEYS</sub></div>	
Protocol versions 1–3:	
<div>LEVEL1UPDATE<sub>LEVEL2KEYSV0</sub></div>	
Protocol version 4 onwards:	
<div>LEVEL1UPDATE<sub>LEVEL2KEYSV0</sub></div>	

<b>LEVEL1UPDATE<sub>LEVEL1KEYS</sub></b>	variable length
Updating the Level 1 keys.	
<b>type = 0 : WORD8</b>	
<b>keys : HIGHERLEVELKEYS</b>	

<b>LEVEL1UPDATE<sub>LEVEL2KEYSV0</sub></b>	variable length
Updating the Level 2 keys, in protocol versions 1–3.	
<b>type = 1 : WORD8</b>	
<b>authorization : AUTHORIZATIONSV0</b>	

<b>LEVEL1UPDATE<sub>LEVEL2KEYSV0</sub></b>	variable length
Updating the Level 2 keys, in protocol version 4 onwards.	
<b>type = 2 : WORD8</b>	
<b>authorization : AUTHORIZATIONSV1</b>	

<b>AUTHORIZATIONSV0</b>	variable length
The set of keys authorized for chain updates, together with access structures determining which keys are authorized for which update types. This is the payload of an update to authorization. This is used for protocol versions 0–3, after which it is superceded by <a href="#">AUTHORIZATIONSV1</a> .	
<b>length : WORD16</b>	
Number of keys.	



<b>keys</b> : $\text{length} \times \text{UPDATEVERIFYKEY}$	
All the keys that can do Level 2 updates.	
<b>emergency</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can do emergency updates together with the threshold, i.e. how many of them needed to authorize the update.	
<b>protocol</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can do protocol updates together with the threshold.	
<b>consensusParameters</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can do election difficulty updates together with the threshold.	
<b>euroPerEnergy</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can do euro-per-energy updates together with the threshold.	
<b>microCCDPerEuro</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can do microCCD-per-euro updates together with the threshold.	
<b>foundationAccount</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can do foundation account updates together with the threshold.	
<b>mintDistribution</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can do mint distribution updates together with the threshold.	
<b>transactionFeeDistribution</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can do transaction fee distribution updates together with the threshold.	
<b>GASRewards</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can do gas rewards updates together with the threshold.	
<b>bakerStakeThreshold</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can do baker stake threshold updates together with the threshold.	
<b>addAnonymityRevoker</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can add anonymity revokers together with the threshold.	
<b>addIdentityProvider</b> : <a href="#">ACCESSSTRUCTURE</a>	
The subset of all the keys that can add identity providers together with the threshold.	
<b>AUTHORIZATIONSV1</b>	variable length
The set of keys authorized for chain updates, together with access structures determining which keys are authorized for which update types. This is the payload of an update to authorization. This is used from protocol version 4 onwards.	
<b>length</b> : <a href="#">WORD16</a>	
Number of keys.	

<b>keys</b> : $\text{length} \times \text{UPDATEVERIFYKEY}$
All the keys that can do Level 2 updates.
<b>emergency</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can do emergency updates together with the threshold, i.e. how many of them needed to authorize the update.
<b>protocol</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can do protocol updates together with the threshold.
<b>consensusParameters</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can do consensus parameter updates together with the threshold.
<b>euroPerEnergy</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can do euro-per-energy updates together with the threshold.
<b>microCCDPerEuro</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can do microCCD-per-euro updates together with the threshold.
<b>foundationAccount</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can do foundation account updates together with the threshold.
<b>mintDistribution</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can do mint distribution updates together with the threshold.
<b>transactionFeeDistribution</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can do transaction fee distribution updates together with the threshold.
<b>GASRewards</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can do gas rewards updates together with the threshold.
<b>poolParameters</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can do pool parameter updates together with the threshold. (This is a renaming of <code>bakerStakeThreshold</code> from <a href="#">AUTHORIZATIONSV0</a> .)
<b>addAnonymityRevoker</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can add anonymity revokers together with the threshold.
<b>addIdentityProvider</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can add identity providers together with the threshold.
<b>cooldownParameters</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can update the cooldown parameters together with the threshold.
<b>timeParameters</b> : <a href="#">ACCESSSTRUCTURE</a>
The subset of all the keys that can update the length of a reward period and mint rate together with the threshold.

ACCESSSTRUCTURE	variable length
Access structure for level 2 update authorization.	
<b>length</b> :	WORD8
Number of keys indices.	
<b>keysIndices</b> :	length $\times$ WORD16
Which keys that can authorize an update.	
<b>threshold</b> :	WORD16
The number of keys needed to authorize an update.	
UPDATEVERIFYKEY	33 bytes
<b>scheme</b> = 0 :	
WORD8	
<b>key</b> :	ED25519SIGNPUBKEY
ANONYMITYREVOCATIONTHRESHOLD	1 bytes
Number of anonymity revokers required to revoke anonymity of a credential.	
WORD8	
COMMISSIONRATES	12 bytes
The commission rates charged by a pool owner (or for passive delegation).	
<b>finalizationCommission</b> :	AMOUNTFRACTION
Fraction of finalization rewards charged by the pool owner.	
<b>bakingCommission</b> :	AMOUNTFRACTION
Fraction of baking rewards charged by the pool owner.	
<b>transactionCommission</b> :	AMOUNTFRACTION
Fraction of transaction rewards charged by the pool owner.	
COMMISSIONRANGE	6 bytes
A permissible range for a particular commission rate.	
<b>min</b> :	AMOUNTFRACTION
Minimum commission rate.	
<b>max</b> :	AMOUNTFRACTION
Maximum commission rate.	
$\text{min} \leq \text{max}$ .	
COMMISSIONRANGES	18 bytes
A set of permissible commission rate ranges.	
<b>finalizationCommissionRange</b> :	COMMISSIONRANGE
Permissible range for finalization commission.	

<b>bakingCommissionRange</b> : COMMISSIONRANGE	
Permissible range for baking commission.	
<b>transactionCommissionRange</b> : COMMISSIONRANGE	
Permissible range for transaction commission.	

<b>CAPITALBOUND</b>	4 bytes
A bound on the relative share of the total staked capital that a validator may have as its stake.	
<b>capitalBound</b> : AMOUNTFRACTION	
<b>capitalBound.fraction</b> > 0.	

<b>LEVERAGEFACTOR</b>	16 bytes
The maximum ratio of baker's effective stake to its equity capital. This cannot be less than 1. This must be represented in normalized form.	
<b>numerator</b> : WORD64	
The numerator.	
<b>denominator</b> : WORD64	
The denominator. $0 < \text{denominator} \leq \text{numerator}$ . $\text{gcd}(\text{numerator}, \text{denominator}) = 1$ .	

<b>MODULE</b>	variable length
Web assembly module in binary format.	
<b>version</b> : WORD32	
<b>length</b> : WORD32	
Length of source.	
<b>source</b> : BYTES(length)	
The source of a contract in binary wasm format.	

<b>MODULEREF</b>	32 bytes
Unique module reference.	
<b>hash</b> : SHA256	

<b>INITNAME</b>	variable length
Name of an init method inside a module.	
<b>SHORTBYTES</b>	

RECEIVENAME	variable length
Name of an receive method inside a module.	
SHORTBYTES	
PARAMETER	variable length
Parameter to either an init method or to a receive method. The parameter is limited to 1kB in size.	
SHORTBYTES	
CONTRACTADDRESS	16 bytes
A contract address consists of an index and a subindex.	
contractIndex : WORD64	
contractSubIndex : WORD64	
MEMO	variable length
A memo attached to a transfer.	
length : WORD16	
Length of the memo in bytes. $\text{length} \leq 256$ .	
memo : BYTES(length)	
The memo value.	
OPENSTATUS	1 byte
The status of a validator's pool. One of the following:	
OPENSTATUS_OPENFORALL	
OPENSTATUS_CLOSEDFORNEW	
OPENSTATUS_CLOSEDFORALL	
OPENSTATUS_OPENFORALL	1 byte
The pool is open for all delegators.	
status = 0 : WORD8	
OPENSTATUS_CLOSEDFORNEW	1 byte
The pool is closed to new delegators.	
status = 1 : WORD8	

OPENSTATUSCLOSEDFORALL	1 byte
The pool is closed for all delegators.	
<b>status</b> = 2 : WORD8	
VALIDATORKEYSWITHPROOFS	352 bytes
The keys of a validator, together with proofs that the validator knows the corresponding secret keys.	
<b>electionVerifyKey</b> : BAKERELECTIONVERIFYKEY	
Public key used for VRF proofs in the election process.	
<b>proofElection</b> : DLOG25519PROOF	
Proof that the validator knows the secret key corresponding to <b>electionVerifyKey</b> .	
<b>signatureVerifyKey</b> : BAKERSIGNVERIFYKEY	
Public key used for signing blocks.	
<b>proofSig</b> : DLOG25519PROOF	
Proof that the validator knows the secret key corresponding to <b>signatureVerifyKey</b> .	
<b>aggregationVerifyKey</b> : BAKERAGGREGATIONVERIFYKEY	
Public key used for aggregate signatures.	
<b>proofAggregation</b> : BAKERAGGREGATIONPROOF	
Proof that the validator knows the secret key corresponding to <b>aggregationVerifyKey</b> .	
URLTEXT	variable length
A URL.	
<b>length</b> : WORD16	
Length of the URL in bytes. <b>length</b> ≤ 2048.	
<b>url</b> : BYTES( <b>length</b> )	
The URL, in UTF-8 encoding.	
DELEGATIONTARGET	variable length
A target that an account may delegate stake to (Passive or a Validator). One of the following:	
<div>DELEGATIONTARGETPASSIVE</div> <div>DELEGATIONTARGETVALIDATOR</div>	
DELEGATIONTARGETPASSIVE	
Delegate passively to all validators.	
<b>tag</b> = 0 : WORD8	

DELEGATIONTARGET <sub>VALIDATOR</sub>
Delegate to a specific validator.
<b>tag</b> = 1 : WORD8
<b>validator</b> : VALIDATORID
The Validator ID of the validator pool to delegate to.

## Chapter 2

# Network Serialization

### 2.1 Account Keys

An account public key consists of a scheme identifier followed by a public key in the corresponding scheme. Currently, only one scheme is supported: Ed25519.

CREDENTIALVERIFYKEY	variable length
A credential public key. One of the following: <div>CREDENTIALVERIFYKEY<sub>Ed25519</sub></div>	
CREDENTIALVERIFYKEY <sub>Ed25519</sub>	33 bytes
scheme = 0 : WORD8	
key : ED25519SIGNPUBKEY	
ACCOUNTOWNERSHIPPROOF	variable length
A number of signatures by an account owner needed to proof ownership of an account. What exactly is signed depends on whether the credential is an initial or a normal one.	
count : WORD8	
The number of signatures. <b>Must be non-zero.</b>	
proofs : count × ACCOUNTOWNERSHIPPROOFENTRY	
The signatures.	
ACCOUNTOWNERSHIPPROOFENTRY	65 bytes
A signature by one of the account owner's keys.	
index : WORD8	
Index identifying which of the account owner's keys this signature is by.	
sig : ED25519SIGNATURE	
The signature.	



SIGNATURETHRESHOLD	1 byte
threshold : WORD8	
threshold > 0.	

ACCOUNTTHRESHOLD	1 byte
threshold : WORD8	
threshold > 0.	

CREDENTIALVERIFYKEYENTRY	variable length
index : WORD8	
key : CREDENTIALVERIFYKEY	

## 2.2 Anonymity revoker and identity provider keys

ARPUBLICKEY	96 bytes
Public key of an anonymity revoker.	
key : ELGAMALPUBLICKEY	
The key.	

IPPUBLICKEYS	variable length
Public keys of an identity provider.	
psKey : PSPUBLICKEY	
The Pointcheval-Sanders public key.	
edKey : ED25519SIGNPUBKEY	
The Ed25519 public key.	

## 2.3 Baker keys

BAKERELECTIONVERIFYKEY	32 bytes
Public key for baker election.	
key : VRFPUBKEY	
The key.	

BAKERSIGNVERIFYKEY	32 bytes
Public key for baker signatures.	
key : ED25519SIGNPUBKEY	
The key.	

BAKERAGGREGATIONVERIFYKEY	96 bytes
Public key for aggregated signatures.	
key : BLSAGGPUBLICKEY	
The key.	

BAKERAGGREGATIONPROOF	64 bytes
Proof of knowledge of secret key	
proof : DLOGPROOF	
The proof.	

## 2.4 Transaction

### 2.4.1 Block Items

BLOCKITEM	variable length
One of the following:	
BLOCKITEM <sub>TRANSACTION</sub>	
BLOCKITEM <sub>CREDENTIALDEPLOYMENT</sub>	
BLOCKITEM <sub>CHAINUPDATE</sub>	

BLOCKITEM <sub>TRANSACTION</sub>	variable length
type = 0 : WORD8	
transaction : ACCOUNTTRANSACTION	

BLOCKITEM <sub>CREDENTIALDEPLOYMENT</sub>	variable length
type = 1 : WORD8	
credential : ACCOUNTCREATION	

BLOCKITEM <sub>CHAINUPDATE</sub>	variable length
type = 2 : WORD8	
update : UPDATEINSTRUCTION	

## 2.4.2 Transactions

ACCOUNTTRANSACTION	variable length
A transaction originating from a particular account.	
<b>signature</b> :	TRANSACTIONSIGNATURE
The signature on the transaction by the source account.	
<b>header</b> :	TRANSACTIONHEADER
Transaction header data.	
<b>encodedPayload</b> :	BYTES(header.payloadSize)
Transaction payload. Typically, this is a TRANSACTIONPAYLOAD, however, deserialization of the payload is a separate step from deserialization of the transaction.	

</>	For an ACCOUNTTRANSACTION, the message to be signed is the SHA256 hash of concatenation of the header and payload:  SHA256(header  encodedPayload)
-----	--

TRANSACTIONSIGNATURE
A collection of cryptographic signatures on a transaction by the keys of credentials deployed to the source account.
count : WORD8
The number of account signatures, i.e. the number of owners of the source account signing the transaction.
accountSignatures : count × ACCOUNTSIGNATURE
Signatures, ordered by increasing key index, and with no duplicate keys.

!	The ACCOUNTSIGNATURES in a TRANSACTIONSIGNATURE <b>must</b> be in ascending order, and have no duplicate keys. Otherwise the transaction will not be considered valid.
---	--

ACCOUNTSIGNATURE
credentialIndex : WORD8
Index identifying which of the account owners these signatures are by.
sigCount : WORD8
Number of signatures by this account owner.
signatures : sigCount × SIGNATUREENTRY
The signatures by this account owner.

SIGNATUREENTRY
keyIndex : WORD8
Index identifying which of the account owner's keys this signature is by.

<b>signature : SHORTBYTES</b>	
The signature.	
<b>TRANSACTIONHEADER</b>	<b>60 bytes</b>
<b>sender : ACCOUNTADDRESS</b>	
The address of the account that is the source of the transaction.	
<b>sequenceNumber : SEQUENCENUMBER</b>	
The sequence number of the transaction. Transactions executed on an account must have sequential sequence numbers, starting from 1.	
<b>energyAmount : ENERGY</b>	
The amount of energy allocated for executing this transaction. (This is the maximum amount of energy that can be consumed by the transaction.)	
<b>payloadSize : PAYLOADSIZE</b>	
The size of the transaction payload.	
<b>expiry : TRANSACTIONEXPIRYTIME</b>	
The time at which the transaction expires. A transaction cannot be included in a block with a timestamp later than the transaction's expiry time.	

### 2.4.3 Credential Deployment

<b>YEARMONTH</b>	<b>3 bytes</b>
A year and month.	
<b>year : WORD16</b>	
$1000 \leq \text{year} \leq 9999$ .	
<b>month : WORD8</b>	
$1 \leq \text{month} \leq 12$ .	
<b>ACCOUNTCREATION</b>	<b>variable length</b>
A credential together with an expiry. It is a message that is included in a block, if valid, but it is not paid for directly by the sender.	
<b>messageExpiry : TRANSACTIONEXPIRYTIME</b>	
<b>credential : ACCOUNTCREDENTIALWITHPROOFS</b>	
<b>ACCOUNTCREDENTIALWITHPROOFS</b>	<b>variable length</b>
Different kinds of credentials that can go onto the chain. One of the following:	
<div>ACCOUNTCREDENTIALWITHPROOFS<sub>INITIAL</sub></div> <div>ACCOUNTCREDENTIALWITHPROOFS<sub>NORMAL</sub></div>	

ACCOUNTCREDENTIALWITHPROOFS <sub>INITIAL</sub>	variable length
The initial credential deployment information consists of values deployed and a signature from the identity provider on said values.	
<b>type</b> = 0 : WORD8	
0 indicates that this is the initial variant of ACCOUNTCREDENTIALWITHPROOFS.	
<b>values</b> : INITIALCREDENTIALDEPLOYMENTVALUES	
The data for the initial account creation. This is submitted by the identity provider on behalf of the account holder.	
<b>sig</b> : ED25519SIGNATURE	
A signature under the public key of identity provider's key on the credential deployment values. This is the dual of 'proofs' for the normal credential deployment.	

INITIALCREDENTIALDEPLOYMENTVALUES	variable length
<b>publicKeys</b> : CREDENTIALPUBLICKEYS	
The account threshold together with the public keys of this credential, i.e. the keys used to check a signature made by the account owner holding this credential, and how many of the account owner's keys needed to sign.	
<b>credId</b> : CREDENTIALREGISTRATIONID	
Registration ID of this credential.	
<b>ipId</b> : IPIDENTITY	
Identity of the identity provider who signed the identity object from which this credential is derived.	
<b>policy</b> : POLICY	
Policy.	

POLICY	variable length
<b>validTo</b> : YEARMONTH	
Validity of credential.	
<b>createdAt</b> : YEARMONTH	
Creation of credential.	
<b>count</b> : WORD16	
Number of attributes in the policy.	
<b>attributes</b> : count × ATTRIBUTEENTRY	
The attributes in this policy.	

ATTRIBUTEENTRY	variable length
<b>attributeTag</b> : WORD8	
Tag identifying which attribute has the below value.	
<b>count</b> : WORD8	

Length of the attribute value in bytes. $\text{count} \leq 31$ .	
<b>attributeValue</b> : BYTES(count)	
Value of this attribute.	

ACCOUNTCREDENTIALWITHPROOFS <sub>NORMAL</sub>	variable length
<b>type</b> = 1 : WORD8	
1 indicates that this is the normal variant of ACCOUNTCREDENTIALWITHPROOFS.	
<b>cdi</b> : CREDENTIALDEPLOYMENTINFORMATION	
The credential deployment information consists of values deployed and the proofs about them.	

CREDENTIALDEPLOYMENTINFORMATION	variable length
<b>values</b> : CREDENTIALDEPLOYMENTVALUES	
<b>proofs</b> : CREDENTIALDEPLOYMENTPROOFS	

CREDENTIALDEPLOYMENTPROOFS	variable length
<b>idProofs</b> : IDOWNERSHIPPROOFS	
All proofs required to prove ownership of an identity, in a credential deployment.	
<b>proofAccSk</b> : ACCOUNTOWNERSHIPPROOF	
A signature by the credential holder on the credential deployment values, the IDOWNERSHIPPROOFS and either the account address, if the credential is being deployed to an existing account, or the transaction expiry, if the credential is being deployed to a new account.	

CREDENTIALDEPLOYMENTVALUES	variable length
<b>publicKeys</b> : CREDENTIALPUBLICKEYS	
The account threshold together with the public keys of this credential, i.e. the keys used to check a signature made by the account owner holding this credential, and how many of the account owner's keys needed to sign.	
<b>credId</b> : CREDENTIALREGISTRATIONID	
Registration ID of this credential.	
<b>ipId</b> : IPIDENTITY	
Identity of the identity provider who signed the identity object from which this credential is derived.	
<b>revocationThreshold</b> : ANONYMITYREVOCATIONTHRESHOLD	
Revocation threshold. Any set of this many anonymity revokers can reveal IdCredPub.	
<b>arData</b> : ARDATA	
Anonymity revocation data associated with this credential.	
<b>policy</b> : POLICY	

Policy.	
<b>CREDENTIALPUBLICKEYS</b>	
The public keys of a credential.	
<b>count</b> :	<b>WORD8</b>
Number of keys. <b>count</b> > 0.	
<b>keys</b> :	<b>count</b> × <b>CREDENTIALVERIFYKEYENTRY</b>
The keys.	
<b>threshold</b> :	<b>SIGNATURETHRESHOLD</b>
The signature threshold, i.e. how many of the concrete account owner's keys needed to sign.	
<b>ARDATA</b>	
Anonymity revocation data associated with a credential.	
<b>count</b> :	<b>WORD8</b>
Number of ARs. <b>count</b> > 0.	
<b>data</b> :	<b>count</b> × <b>ARDATAENTRY</b>
The data.	
<b>ARDATAENTRY</b>	100 bytes
Anonymity revocation data of one AR.	
<b>arIdentity</b> :	<b>ARIDENTITY</b>
The id of the anonymity revoker that can decrypt the below data.	
<b>data</b> :	<b>CHAINARDATA</b>
The data.	
<b>CHAINARDATA</b>	96 bytes
Data needed on-chain to revoke anonymity of the account holder.	
<b>idCredPubShare</b> :	<b>ELGAMALCIPHERTEXT</b>
Encrypted share of idCredPub.	

#### 2.4.4 Chain Updates

<b>UPDATEINSTRUCTION</b>	variable length
An update instruction.	
<b>header</b> :	<b>UPDATEHEADER</b>
<b>payload</b> :	<b>UPDATEPAYLOAD</b>
<b>signatures</b> :	<b>UPDATEINSTRUCTIONSIGNATURES</b>

UPDATEINSTRUCTIONSIGNATURES	variable length
Signatures on a update instruction.	
<b>length</b> : WORD16	
Number of signatures.	
<b>signatures</b> : length × (WORD16, SHORTBYTES)	
The indices of the keys used to sign together with the signatures.	

UPDATEHEADER	28 bytes
The header for an update instruction, consisting of the sequence number, effective time, expiry time (timeout), and payload size. This structure is the same for all update payload types.	
<b>seqNumber</b> : UPDATESEQUENCENUMBER	
<b>effectiveTime</b> : TRANSACTIONTIME	
<b>timeout</b> : TRANSACTIONEXPIRYTIME	
<b>payloadSize</b> : PAYLOADSIZE	

UPDATEPAYLOAD	variable length
The payload of an update instruction. One of the following:	
UPDATEPAYLOAD <sub>PROTOCOL</sub>	
UPDATEPAYLOAD <sub>EUROPERENERGY</sub>	
UPDATEPAYLOAD <sub>MICROCCDPEREURO</sub>	
UPDATEPAYLOAD <sub>FOUNDATIONACCOUNT</sub>	
UPDATEPAYLOAD <sub>TRANSACTIONFEEDISTRIBUTION</sub>	
UPDATEPAYLOAD <sub>ROOTUPDATE</sub>	
UPDATEPAYLOAD <sub>LEVEL1</sub>	
UPDATEPAYLOAD <sub>ADDANONYMITYREVOKER</sub>	
UPDATEPAYLOAD <sub>ADDIDENTITYPROVIDER</sub>	
The following are supported at protocol versions 1–3 only:	
UPDATEPAYLOAD <sub>MINTDISTRIBUTIONV0</sub>	



UPDATEPAYLOAD<sub>BAKERSTAKE</sub>THRESHOLD

The following are supported at protocol versions 1–5 only:

UPDATEPAYLOAD<sub>ELECTION</sub>DIFFICULTY

UPDATEPAYLOAD<sub>GAS</sub>REWARDSV0

The following are supported from protocol version 4 onwards:

UPDATEPAYLOAD<sub>COOLDOWN</sub>PARAMETERS

UPDATEPAYLOAD<sub>POOL</sub>PARAMETERS

UPDATEPAYLOAD<sub>TIME</sub>PARAMETERS

UPDATEPAYLOAD<sub>MINT</sub>DISTRIBUTIONV1

The following are supported from protocol version 6 onwards:

UPDATEPAYLOAD<sub>TIMEOUT</sub>PARAMETERS

UPDATEPAYLOAD<sub>MIN</sub>BLOCKTIME

UPDATEPAYLOAD<sub>BLOCK</sub>ENERGYLIMIT

UPDATEPAYLOAD<sub>GAS</sub>REWARDSV1

UPDATEPAYLOAD<sub>FINALIZATION</sub>COMMITTEEPARAMETERS

UPDATEPAYLOAD<sub>PROTOCOL</sub>

variable length

A protocol update.

payloadType = 1 : WORD8

length : WORD64

Length of the rest of the payload.

messageLength : WORD64

Length of message.

message : BYTES(messageLength)

A brief message about the update.

urlLength : WORD64

Length of URL.

<code>url : BYTES(urlLength)</code>	
A URL of a document describing the update.	
<code>hash : SHA256</code>	
SHA256 hash of the specification document.	
<code>aux : BYTES(length - 8 - messageLength - 8 - urlLength - 32)</code>	
Auxiliary data whose interpretation is defined by the new specification.	

<code>UPDATEPAYLOAD<sub>ELECTIONDIFFICULTY</sub></code>	5 bytes
An election difficulty update. Protocol versions 1–5.	
<code>payloadType = 2 : WORD8</code>	
<code>electionDifficulty : ELECTIONDIFFICULTY</code>	

<code>UPDATEPAYLOAD<sub>EUROPERENERGY</sub></code>	17 bytes
A euro-per-energy parameter update.	
<code>payloadType = 3 : WORD8</code>	
<code>euroPerEnergy : EXCHANGERATE</code>	

<code>UPDATEPAYLOAD<sub>MICROCCDPEREURO</sub></code>	17 bytes
A microCCD-per-euro parameter update.	
<code>payloadType = 4 : WORD8</code>	
<code>microCCDPerEuro : EXCHANGERATE</code>	

<code>UPDATEPAYLOAD<sub>FOUNDATIONACCOUNT</sub></code>	33 bytes
An foundation account update.	
<code>payloadType = 5 : WORD8</code>	
<code>account : ACCOUNTADDRESS</code>	

<code>UPDATEPAYLOAD<sub>MINTDISTRIBUTIONV0</sub></code>	14 bytes
A mint distribution update. Protocol versions 1–3. This is superceded by MINTDISTRIBUTIONV1 from protocol version 4.	
<code>payloadType = 6 : WORD8</code>	
<code>mintPerSlot : MINTRATE</code>	
<code>bakingReward : AMOUNTFRACTION</code>	
<code>finalizationReward : AMOUNTFRACTION</code>	

UPDATEPAYLOAD <sub>TRANSACTIONFEEDISTRIBUTION</sub>	9 bytes
The distribution of block transaction fees among the block baker, the GAS account, and the foundation account.	
payloadType = 7 : WORD8	
baker : AMOUNTFRACTION	
gas : AMOUNTFRACTION	
UPDATEPAYLOAD <sub>GASREWARDSV0</sub>	17 bytes
A GAS rewards update. Protocol versions 1–5. This is superceded by GASREWARDSV1 from protocol version 6.	
payloadType = 8 : WORD8	
baker : AMOUNTFRACTION	
finalizationProof : AMOUNTFRACTION	
accountCreation : AMOUNTFRACTION	
chainUpdate : AMOUNTFRACTION	
UPDATEPAYLOAD <sub>BAKERSTAKE</sub> THRESHOLD	9 bytes
A baker stake threshold update. Protocol versions 1–3. This is superceded by POOLPARAMETERS from protocol version 4.	
payloadType = 9 : WORD8	
amount : AMOUNT	
UPDATEPAYLOAD <sub>ROOT</sub> UPDATE	variable length
A root update.	
payloadType = 10 : WORD8	
update : ROOTUPDATE	
UPDATEPAYLOAD <sub>LEVEL1</sub>	variable length
A Level1 update.	
payloadType = 11 : WORD8	
update : LEVEL1UPDATE	
UPDATEPAYLOAD <sub>ADDANONYMITY</sub> REVOKER	variable length
Add an anonymity revoker.	
payloadType = 12 : WORD8	
arInfo : ARINFO	

UPDATEPAYLOAD <sub>ADDIDENTITYPROVIDER</sub>	variable length
Add an identity provider.	
payloadType = 13 : WORD8	
ipInfo : IPINFO	
UPDATEPAYLOAD <sub>COOLDOWNPARAMETERS</sub>	17 bytes
An update to the parameters affecting cooldown periods for validators and delegators, from protocol version 4. From protocol version 7, the distinction in cooldown time between validators and delegators is removed and the lowest of the two values is used.	
payloadType = 14 : WORD8	
poolOwnerCooldown : DURATIONSECONDS	
Number of seconds that pool owners (i.e. validators) must cooldown when reducing their equity capital or closing the pool.	
delegatorCooldown : DURATIONSECONDS	
Number of seconds that delegators must cooldown when reducing their delegated stake.	
UPDATEPAYLOAD <sub>POOLPARAMETERS</sub>	59 bytes
An update to the parameters affecting validator pools, from protocol version 4.	
payloadType = 15 : WORD8	
passiveCommissions : COMMISSIONRATES	
Commission rates charged for passive delegation.	
commissionBounds : COMMISSIONRANGES	
Bounds on the commission rates that may be charged by bakers.	
minimumEquityCapital : AMOUNT	
Minimum equity capital required for a new baker.	
capitalBound : CAPITALBOUND	
Maximum fraction of the total staked capital of that a new baker can have.	
leverageBound : LEVERAGEFACTOR	
The maximum leverage that a baker can have as a ratio of total stake to equity capital.	
UPDATEPAYLOAD <sub>TIMEPARAMETERS</sub>	14 bytes
An update to the parameters defining the reward period length and mint rate per payday, from protocol version 4.	
payloadType = 16 : WORD8	
rewardPeriod : WORD64	
Number of epochs constituting a payday. 0 < rewardPeriod.	

mintPerPayday : <a href="#">MINTRATE</a>	
Mint rate per payday (as a proportion of the extant supply).	

UPDATEPAYLOAD <sub>MINTDISTRIBUTIONV1</sub>	9 bytes
A mint distribution update, from protocol version 4.	
payloadType = 17 : <a href="#">WORD8</a>	
bakingReward : <a href="#">AMOUNTFRACTION</a>	
finalizationReward : <a href="#">AMOUNTFRACTION</a>	

UPDATEPAYLOAD <sub>TIMEOUTPARAMETERS</sub>	41 bytes
An update to the parameters controlling consensus timeouts, from protocol version 6.	
payloadType = 18 : <a href="#">WORD8</a>	
timeoutBase : <a href="#">DURATION</a>	
The base timeout for consensus.	
timeoutIncreaseNumerator : <a href="#">WORD64</a>	
timeoutIncreaseDenominator : <a href="#">WORD64</a>	
The factor by which the timeout is increased, expressed as a ratio. $0 < \text{timeoutIncreaseDenominator} < \text{timeoutIncreaseNumerator}$ . $\text{gcd}(\text{timeoutIncreaseNumerator}, \text{timeoutIncreaseDenominator}) = 1$ .	
timeoutDecreaseNumerator : <a href="#">WORD64</a>	
timeoutDecreaseDenominator : <a href="#">WORD64</a>	
The factor by which the timeout is decreased, expressed as a ratio. $0 < \text{timeoutDecreaseNumerator} < \text{timeoutDecreaseDenominator}$ . $\text{gcd}(\text{timeoutDecreaseNumerator}, \text{timeoutDecreaseDenominator}) = 1$ .	

UPDATEPAYLOAD <sub>MINBLOCKTIME</sub>	9 bytes
An update to the minimum time between blocks, from protocol version 6.	
payloadType = 19 : <a href="#">WORD8</a>	
minBlockTime : <a href="#">DURATION</a>	
The minimum time between blocks.	

UPDATEPAYLOAD <sub>BLOCKENERGYLIMIT</sub>	9 bytes
An update to the maximum energy that can be consumed by a block, from protocol version 6.	
payloadType = 20 : <a href="#">WORD8</a>	
blockEnergyLimit : <a href="#">ENERGY</a>	
The maximum energy that can be consumed by a block.	

UPDATEPAYLOAD <sub>GASREWARDSV1</sub>	13 bytes
A GAS rewards update, from protocol version 6.	
payloadType = 21 : WORD8	
baker : AMOUNTFRACTION	
Fraction of the GAS account awarded for baking a block.	
accountCreation : AMOUNTFRACTION	
Fraction of the GAS account awarded for including a credential deployment block item.	
chainUpdate : AMOUNTFRACTION	
Fraction of the GAS account awarded for including a chain update block item.	

UPDATEPAYLOAD <sub>FINALIZATIONCOMMITTEEPARAMETERS</sub>	13 bytes
An update to the finalization committee parameters.	
payloadType = 22 : WORD8	
minFinalizers : WORD32	
The minimum number of validators to include in the finalization committee before imposing the relative stake threshold. minFinalizers > 0.	
maxFinalizers : WORD32	
The maximum number of validators to include in the finalization committee. maxFinalizers ≥ minFinalizers.	
relativeStakeThreshold : AMOUNTFRACTION	
The fraction of the total stake required for a validator to be included in the finalization committee.	

### 2.4.5 Transaction Payloads

TRANSACTIONPAYLOAD	variable length
One of the following:	
TRANSACTIONPAYLOAD <sub>DEPLOYMODULE</sub>	TRANSACTIONPAYLOAD <sub>INITCONTRACT</sub>
TRANSACTIONPAYLOAD <sub>UPDATE</sub>	TRANSACTIONPAYLOAD <sub>TRANSFER</sub>
TRANSACTIONPAYLOAD <sub>UPDATECREDENTIALKEYS</sub>	
TRANSACTIONPAYLOAD <sub>TRANSFERTOPUBLIC</sub>	
TRANSACTIONPAYLOAD <sub>UPDATECREDENTIALS</sub>	TRANSACTIONPAYLOAD <sub>REGISTERDATA</sub>
The following payload types are supported in protocol versions 1–3:	
TRANSACTIONPAYLOAD <sub>ADDBAKER</sub>	TRANSACTIONPAYLOAD <sub>REMOVEBAKER</sub>

TRANSACTIONPAYLOADUPDATEBAKERSTAKE

TRANSACTIONPAYLOADUPDATEBAKERRESTAKEEARNINGS

TRANSACTIONPAYLOADUPDATEBAKERKEYS

The following payload types are supported in protocol versions 1–6:

TRANSACTIONPAYLOADENCRYPTEDAMOUNTTRANSFER

TRANSACTIONPAYLOADTRANSFERTOENCRYPTED

TRANSACTIONPAYLOADTRANSFERWITHSCHEDULE

The following payload types are supported in protocol version 2 onwards:

TRANSACTIONPAYLOADTRANSFERWITHMEMO

TRANSACTIONPAYLOADTRANSFERWITHSCHEDULEANDMEMO

The following payload types are supported in protocol versions 2–6:

TRANSACTIONPAYLOADENCRYPTEDAMOUNTTRANSFERWITHMEMO

The following payload types are supported in protocol version 4 onwards:

TRANSACTIONPAYLOADCONFIGUREVALIDATOR

TRANSACTIONPAYLOADCONFIGUREDELEGATION

TRANSACTIONPAYLOADDEPLOYMODULE

Deploys a code module.

payloadType = 0 : WORD8

module : MODULE

TRANSACTIONPAYLOADINITCONTRACT

Initializes a new smart contract instance.

payloadType = 1 : WORD8

amount : AMOUNT

moduleRef : MODULEREF

initName : INITNAME

parameter : PARAMETER

TRANSACTIONPAYLOAD <sub>UPDATE</sub>	
Invokes a smart contract instance.	
payloadType = 2 : WORD8	
amount : AMOUNT	
address : CONTRACTADDRESS	
receiveName : RECEIVENAME	
message : PARAMETER	
TRANSACTIONPAYLOAD <sub>TRANSFER</sub>	41 bytes
A simple transfer from an account to an account.	
payloadType = 3 : WORD8	
to : ACCOUNTADDRESS	
amount : AMOUNT	
TRANSACTIONPAYLOAD <sub>ADDBAKER</sub>	
Add a baker. (Only supported in protocol versions 1–3. From protocol version 4, this is superseded by TRANSACTIONPAYLOAD <sub>CONFIGUREVALIDATOR</sub> .)	
payloadType = 4 : WORD8	
electionVerifyKey : BAKERELECTIONVERIFYKEY	
signatureVerifyKey : BAKERSIGNVERIFYKEY	
aggregationVerifyKey : BAKERAGGREGATIONVERIFYKEY	
proofSig : DLOG25519PROOF	
proofElection : DLOG25519PROOF	
proofAggregation : BAKERAGGREGATIONPROOF	
bakingStake : AMOUNT	
restakeEarnings : BOOL	
TRANSACTIONPAYLOAD <sub>REMOVEBAKER</sub>	1 byte
Remove a baker. (Only supported in protocol versions 1–3. From protocol version 4, this is superseded by TRANSACTIONPAYLOAD <sub>CONFIGUREVALIDATOR</sub> .)	
payloadType = 5 : WORD8	
TRANSACTIONPAYLOAD <sub>UPDATEBAKERSTAKE</sub>	9 bytes
Change a baker's stake. (Only supported in protocol versions 1–3. From protocol version 4, this is superseded by TRANSACTIONPAYLOAD <sub>CONFIGUREVALIDATOR</sub> .)	



payloadType = 6 : WORD8	
stake : AMOUNT	
TRANSACTIONPAYLOADUPDATEBAKERRESTAKEEARNINGS	2 bytes
Change whether a baker's earnings are restaked. (Only supported in protocol versions 1–3. From protocol version 4, this is superceded by TRANSACTIONPAYLOADCONFIGUREVALIDATOR.)	
payloadType = 7 : WORD8	
restakeEarnings : BOOL	
TRANSACTIONPAYLOADUPDATEBAKERKEYS	
Update baker keys. (Only supported in protocol versions 1–3. From protocol version 4, this is superceded by TRANSACTIONPAYLOADCONFIGUREVALIDATOR.)	
payloadType = 8 : WORD8	
electionVerifyKey : BAKERELECTIONVERIFYKEY	
signatureVerifyKey : BAKERSIGNVERIFYKEY	
aggregationVerifyKey : BAKERAGGREGATIONVERIFYKEY	
proofSig : DLOG25519PROOF	
proofElection : DLOG25519PROOF	
proofAggregation : BAKERAGGREGATIONPROOF	
TRANSACTIONPAYLOADUPDATECREDENTIALKEYS	
New set of credential keys to be replaced with the existing ones, including updating the threshold.	
payloadType = 13 : WORD8	
credId : CREDENTIALREGISTRATIONID	
keys : CREDENTIALPUBLICKEYS	
TRANSACTIONPAYLOADENCRYPTEDAMOUNTTRANSFER	
Send an encrypted amount to an account. (Only supported in protocol versions 1–6.)	
payloadType = 16 : WORD8	
to : ACCOUNTADDRESS	
data : ENCRYPTEDAMOUNTTRANSFERDATA	
Encrypted amount and proof that it is done correctly.	

TRANSACTIONPAYLOAD <sub>TRANSFERToENCRYPTED</sub>	9 bytes
Transfer some amount from public to encrypted balance. (Only supported in protocol versions 1–6.)	
payloadType = 17 : WORD8	
amount : AMOUNT	
TRANSACTIONPAYLOAD <sub>TRANSFERToPUBLIC</sub>	
Decrypt a portion of the encrypted balance.	
payloadType = 18 : WORD8	
data : SECToPUBAMOUNTTRANSFERDATA	
How much to transfer and proof that remaining encrypted amount is correct.	
TRANSACTIONPAYLOAD <sub>TRANSFERWithSCHEDULE</sub>	variable length
Send a transfer with an attached schedule	
payloadType = 19 : WORD8	
to : ACCOUNTADDRESS	
length : WORD8	
Number of scheduled transfers.	
schedule : length × (TIMESTAMP, AMOUNT)	
List of scheduled transfers.	
TRANSACTIONPAYLOAD <sub>UPDATECREDENTIALS</sub>	variable length
Update the account threshold and the credentials linked to an account by adding or removing credentials. The credential with index 0 can never be removed.	
payloadType = 20 : WORD8	
cdiLength : WORD8	
Number of new credentials	
newCredInfos : cdiLength × (WORD8, CREDENTIALDEPLOYMENTINFORMATION)	
Indices and deployment informations of the new credentials.	
removeLength : WORD8	
Number of credentials to be removed.	
removeCredIds : removeLength × CREDENTIALREGISTRATIONID	
The Credential IDs of the credentials to be removed.	
newThreshold : ACCOUNTTHRESHOLD	
TRANSACTIONPAYLOAD <sub>REGISTERDATA</sub>	variable length
Register data on the chain.	

payloadType = 21 : WORD8	
data : SHORTBYTES	

TRANSACTIONPAYLOAD <sub>TRANSFERWITHMEMO</sub>	variable length
Send an amount to an account with a memo. (Only supported from protocol version 2 onwards.)	
payloadType = 22 : WORD8	
to : ACCOUNTADDRESS	
memo : MEMO	
amount : AMOUNT	

TRANSACTIONPAYLOAD <sub>ENCRYPTEDAMOUNTTRANSFERWITHMEMO</sub>	variable length
Send an encrypted amount to an account with a memo. (Only supported in protocol versions 2–6.)	
payloadType = 23 : WORD8	
to : ACCOUNTADDRESS	
memo : MEMO	
data : ENCRYPTEDAMOUNTTRANSFERDATA	
Encrypted amount and proof that it is done correctly.	

TRANSACTIONPAYLOAD <sub>TRANSFERWITHSCHEDULEANDMEMO</sub>	variable length
Send an amount to an account with a schedule and a memo.	
payloadType = 24 : WORD8	
to : ACCOUNTADDRESS	
memo : MEMO	
length : WORD8	
Number of scheduled transfers.	
schedule : length × (TIMESTAMP, AMOUNT)	
List of scheduled transfers.	

TRANSACTIONPAYLOAD <sub>CONFIGUREVALIDATOR</sub>	variable length
Configure a validator.	
payloadType = 25 : WORD8	
bitmap : CONFIGUREVALIDATORBITMAP	
Which fields of the validator are being configured.	
capital : bitmap.hasCapital × AMOUNT	

The equity capital of the validator. If this is set to 0, the validator is removed.
<code>restakeEarnings : bitmap.hasRestakeEarnings × BOOL</code>
Whether the validator's earnings are restaked.
<code>openForDelegation : bitmap.hasOpenForDelegation × OPENSTATUS</code>
Whether the pool is open for delegators.
<code>keysWithProofs : bitmap.hasKeysWithProofs × VALIDATORKEYSWITHPROOFS</code>
The key/proof pairs to verify the validator.
<code>metadataURL : bitmap.hasMetadataURL × URLTEXT</code>
The URL referencing the validator's metadata.
<code>transactionFeeCommission :     bitmap.hasTransactionFeeCommission × AMOUNTFRACTION</code>
The commission the pool owner takes on transaction fees.
<code>bakingRewardCommission : bitmap.hasBakingRewardCommission × AMOUNTFRACTION</code>
The commission the pool owner takes on baking rewards.
<code>finalizationRewardCommission :     bitmap.hasFinalizationRewardCommission × AMOUNTFRACTION</code>
The commission the pool owner takes on finalization rewards.

CONFIGUREVALIDATORBITMAP	2 bytes
A bitmap indicating which fields of a validator are being configured.	
<i>unused</i> = 0 : WORD8	
Unused. Reserved for future use.	
hasFinalizationRewardCommission : BIT	
hasBakingRewardCommission : BIT	
hasTransactionFeeCommission : BIT	
hasMetadataURL : BIT	
hasKeysWithProofs : BIT	
hasOpenForDelegation : BIT	
hasRestakeEarnings : BIT	
hasCapital : BIT	

TRANSACTIONPAYLOADCONFIGUREDELEGATION	variable length
Configure an account for delegation.	
payloadType = 26 : WORD8	
bitmap : CONFIGUREDELEGATIONBITMAP	
Which fields of the delegation are being configured.	

<b>capital</b> : bitmap.hasCapital $\times$ <b>AMOUNT</b>	
The staked capital to be delegated. If this is set to 0, the delegator is removed.	
<b>restakeEarnings</b> : bitmap.hasRestakeEarnings $\times$ <b>BOOL</b>	
Whether the delegator's earnings are restaked.	
<b>delegationTarget</b> : bitmap.hasDelegationTarget $\times$ <b>DELEGATIONTARGET</b>	
Whether the pool is open for delegators.	

<b>CONFIGUREDELEGATIONBITMAP</b>	2 bytes
A bitmap indicating which fields of a delegator are being configured.	
<i>unused</i> = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) : 13 $\times$ <b>BIT</b>	
Unused. Reserved for future use.	
<b>hasDelegationTarget</b> : <b>BIT</b>	
<b>hasRestakeEarnings</b> : <b>BIT</b>	
<b>hasCapital</b> : <b>BIT</b>	