JANZEER BLOCKCHAIN

الورقة الصفراء

المواصفة الرسمية للبروتوكول

إصدار الوثيقة 1.0 · شبكة جنزير · 2026

الورقة الصفراء لمشروع Janzeer

هذه الوثيقة هي المواصفة الرسمية لبروتوكول سلسلة كتل Janzeer: طبقة أولى (Layer-1) ذات معروض ثابت **بلا حصص (stake-free) وبلا أذونات (permissionless)** (تسجيل مفتوح للمُصادِقين + إنتاج كتل بتناوب ثابت مع نهائية BFT ثنائية المرحلة) مبنية على مخطط تشفيري مخصّص لمحفظة HD غير حافظة. تُعرّف الوثيقة، بايتًا ببايت، البدائيات التشفيرية، واشتقاق المفاتيح والعناوين، وترميزات المعاملات والكتل، والتجزئة، وتحجيم المبالغ، وقواعد انتقال الحالة، وخوارزمية الإجماع تمامًا كما هي مُنفَّذة في إصدار العقدة 0.1.0 (`com.zg.crypto.janzeer`). كل قيمة هنا هي قيمة افتراضية في الشيفرة؛ ويجب على المُنفِّذين إعادة إنتاج هذه التخطيطات بدقة تامة للبقاء ضمن الإجماع. وحيثما كانت القيمة من قبيل التحفّظات الخاصة بالتطوير أو شبكة الاختبار، فإنها تُعلَّم بذلك.

---

1. الترميز والثوابت

1.1 الترميز

- `||` يدلّ على دمج البايتات.
- `x.utf8` يدلّ على ترميز البايتات بصيغة UTF-8 (ASCII للسلاسل السداسية عشرية) للسلسلة `x`. تُرمَّز العناوين والمفاتيح والسلاسل السداسية عشرية على هيئة بايتاتها **النصّية** (محارف السداسي العشري)، **لا** مُفكّكة إلى بايتات خام، ما لم يُذكر خلاف ذلك.
- `int(x)` / `uint64(x)` يدلّ على عدد صحيح ثابت العرض **big-endian** (الصنف `ByteBuffer` Java). `(8)` = long بطول 8 بايتات، `(4)` = int بطول 4 بايتات.
- `keccak(x)` = Keccak-256 (double-SHA256). `H²(x)` = `SHA-256(SHA-256(x))`. `H(x)` = SHA-256 (حشو Keccak الأصلي، **وليس** NIST SHA3-256).
- `hex(b)` = الترميز السداسي عشري بأحرف صغيرة للبايتات `b`.
- `n` = رتبة زمرة secp256k1.
- `scaledLong(x)` = دالة تحجيم المبالغ المُعرَّفة في §1.3.

1.2 الثوابت

الثابت	القيمة	ملاحظات
Scale (المنازل العشرية)	`8`	تُحجَّم كل المبالغ بمقدار `8^10`
`MAX_SUPPLY`	`100,000,000`	ثابت؛ العملات لا تُسكّ أبدًا
`MINIMUM_AMOUNT`	`0.1`	أدنى مبلغ قابل للتحويل
`MINIMUM_FEE`	`0.01`	أدنى رسوم معاملة
`MAX_DATA_SIZE`	`256` بايت	حدّ حقل المذكّرة / البيانات
ترتيب بايتات الأعداد الصحيحة	big-endian	جميع الأعداد الصحيحة متعددة البايتات
ترميز السلاسل	UTF-8 / ASCII-hex	بايتات نصّية، غير مُفكّكة
المنحنى	secp256k1	مفاتيح عامة مضغوطة (33 بايت)
مسار HD	`m/0/0/0`	جميعها غير مُصلَّبة (non-hardened)
`network-id` (الافتراضي)	`janzeer`	مربوط في كل صورة أولية موقَّعة وكل مصافحة (§1.4)
عنوان الخزينة "bank"	`0x0000000000000000000000000000000000000000`	بلا مفتاح، محصّن ضد الاستيلاء
`fee-create-token-tx`	`5`	الرسوم الدقيقة لإنشاء (CREATE) رمز أصيل (§8)
`token-activation-height`	`0`	الارتفاع الذي تُفعَّل عنده الرموز الأصيلة؛ مُثبَّت في بصمة مواصفات السلسلة (§8.4)

1.3 تحجيم المبالغ — `toScaledLong`

تُحمَل جميع القيم النقدية في الصور الأولية على هيئة long ذي إشارة بطول 8 بايتات بترتيب big-endian يمثّل المبلغ مضروبًا في `8^10`:

```
toScaledLong(x) = x.setScale(8, HALF_UP) × 10^8   as int64 (big-endian, 8 bytes)
```

أمثلة: `000_000_1 → 0.01`؛ `000_000_10 → 0.1`؛ `000_000_150 → 1.5`؛ `000_000_000_1 → 10`.

نمط التقريب هو `HALF_UP` عند 8 منازل عشرية قبل الضرب. تُستخدم هذه الدالة في كل موضع يدخل فيه مبلغ أو رسوم أو مكافأة أو رصيد إلى صورة أولية أو إلى مُدخَل تجزئة الحالة.

1.4 هوية الشبكة (الحماية من إعادة التشغيل عبر السلاسل)

سلسلة **معرّف الشبكة** (`consensus.network-id`، الافتراضي `janzeer`؛ متمايزة لكل سلسلة — الشبكة الرئيسية / شبكة الاختبار / شبكة التطوير) مربوطة في **كلٍّ** من الصورة الأولية الموقَّعة للمعاملة (§3.1) **و** مجال مصافحة P2P (§6.8). ولذلك لا يمكن

إعادة تشغيل معاملة مُوقَّعة لشبكةٍ ما على شبكة أخرى: فبايتات معرّف الشبكة هي أول بايتات كل صورة أولية للمعاملة. وكل صورة أولية للتكافؤ (parity) المُلتزَم بها تبدأ الآن بـ `6a616e7a656572` (بايتات UTF-8 للكلمة `janzeer`).

---

2. البدائيات التشفيرية

2.1 عبارة الاسترداد ← البذرة (PBKDF2)

تُشتقّ البذرة من عبارة الاسترداد عبر PBKDF2 مع HMAC-SHA512. وهذا **ليس** BIP39: فبادئة الملح هي قيمة حرفية خاصة بـ Janzeer، وليست السلسلة `"mnemonic"`.

```
seed = PBKDF2-HMAC-SHA512(
          password   = NFKD(mnemonic) as UTF-8 bytes,
          salt       = "@_Janzeer_Blockchain_@" || NFKD(passphrase) as UTF-8,
          iterations = 2048,
          dkLen      = 64 bytes )
```

المعامل	القيمة
PRF	HMAC-SHA512
Iterations	`2048`
Output length	bytes `64`
Salt prefix (literal)	`@_Janzeer_Blockchain_@`
Passphrase (default)	(empty) `""`
Normalization	NFKD على كلٍّ من كلمة المرور والعبارة

2.2 المفتاح الرئيسي لـ HD

تُشتقّ العقدة الرئيسية عبر HMAC-SHA512 باستخدام **سلسلة الملح ذاتها** مفتاحًا (وليس `"Bitcoin seed"` الخاص بـ BIP32):

```
I  = HMAC-SHA512(key = "@_Janzeer_Blockchain_@", msg = seed)   // 64 bytes
IL = I[0:32]   // master private key
IR = I[32:64]  // master chain code
```

2.3 اشتقاق المفتاح الابن (CKD غير المُصلَّب)

يستخدم الاشتقاق صيغة CKD غير المُصلَّبة القياسية. ومسار الاشتقاق هو `m/0/0/0` (الحساب / المحفظة / العنوان)، وجميع الفهارس غير مُصلَّبة (`i < 2^31`):

```
data       = serP(compressedParentPub)(33B) || ser32(i)(4B)
I          = HMAC-SHA512(key = parentChainCode, msg = data)
childPriv  = (I[0:32] + parentPriv) mod n
childChain = I[32:64]
```

`serP(P)` هو ترميز SEC1 المضغوط بطول 33 بايت للنقطة العامة؛ و `ser32(i)` هو الفهرس بطول 4 بايتات بترتيب big-endian. ومفتاح المحفظة هو زوج المفتاح الخاص/العام عند `m/0/0/0`.

2.4 مخطط العناوين

```
address = "0x" || hex( keccak256( compressedPubKey )[0:20] )
```

خصائص رئيسية، وهي **تختلف عن Ethereum**:

1. مُدخَل التجزئة هو المفتاح العام **المضغوط بطول 33 بايت** (يستخدم Ethereum المفتاح غير المضغوط بطول 64 بايت دون بادئة `0x04`).
2. تُؤخَذ **أول 20 بايت** من الملخّص (يستخدم Ethereum **آخر 20 بايت**).
3. التجزئة هي **Keccak-256** (الحشو الأصلي)، وليست NIST SHA3-256.
4. **الصيغة القانونية المُخزَّنة والمُجزَّأة هي السداسي العشري بأحرف صغيرة بالكامل** بادئة `0x`. أما تحقّق **EIP-55** الحقيقي بأحرف مختلطة (`toChecksumAddress`: كبِّر كل حرف سداسي عشري يكون النِبل (nibble) المقابل له من keccak (`≥ 8`) فيُطبَّق عند **حدّ العرض في المحفظة / واجهة API فقط** ويُتحقَّق منه عند الإدخال. وهو تمثيلي **وليس** ذا صلة بالإجماع — فالعنوان بأحرف صغيرة بالكامل مقبول دائمًا، وتستخدم الصور الأولية/تجزئات الحالة الصيغة بالأحرف الصغيرة دائمًا.

2.5 التوقيعات (ECDSA / secp256k1)

الخاصية	القيمة
المنحنى	secp256k1
توليد الرقم التسلسلي (nonce)	حتمي (HMAC-SHA256) **RFC-6979**
القابلية للتطويع	**canonical low-S** مفروضة؛ تُرفض التوقيعات ذات high-S
الترميز	**DER**، ثم **Base64**
المُدخَل الموقَّع	**الملخّص الخام بطول 32 بايت** (دون تجزئة إضافية)

يُحتسب التوقيع على الملخّص بطول 32 بايت مباشرةً. ويجب على العملاء التوقيع بدلالات `prehash = false` (أي لا تدع مكتبة التوقيع تُجزِّئ المُدخَل مرة أخرى)؛ فالملخّص المُمرَّر هو بالفعل البايتات الـ32 النهائية.

بالنسبة للمعاملات، يكون الملخّص الموقَّع هو SHA-256 **الفردي** للصورة الأولية للمعاملة — لكن معرّف المعاملة (`hash`) هو double-SHA256 (§3.8). وبالنسبة لإثباتات مصافحة P2P، يكون الملخّص الموقَّع هو `H²(DOMAIN || ":" || networkId || nonce)` (انظر §6.8).

2.6 ملخّص التجزئة

الكائن	التجزئة
تجزئة المعاملة (`hash`)	`hex( H²(txPreimage) )`
تجزئة الكتلة	`hex( H²(blockPreimage) )`
تجزئة مُدخَل الحالة	`hex( H²(stateInput) )`
ورقة/زوج Merkle (وسيط)	`H` (single SHA-256)
زوج Merkle النهائي	`H²`، سداسي عشري بأحرف صغيرة
العنوان	`keccak256(...)[0:20]`

---

3. صيغة المعاملة

الصورة الأولية الموقَّعة لكل معاملة هي التخطيط **الأساسي** متبوعًا بحمولة **خاصة بالنوع**. جميع الأعداد الصحيحة بترتيب big-endian؛ وجميع العناوين/المفاتيح هي بايتاتها النصّية بصيغة UTF-8. ومجموعة المعاملات الكاملة هي `TransferTx` و`PromoterTx` و`ExitPromoterTx` و`TokenTx` (مُقدَّمة من المستخدم) إضافةً إلى `RewardTx` و`SlashingTx` (مولَّدتان من النظام). وتحمل `TokenTx` مُحدِّد عملية `op` بطول بايت واحد وتخدم كل عمليات الرموز الأصيلة بحمولة قانونية واحدة (§3.7، §8).

تحتفظ الأنواع `PromoterTx` و`ExitPromoterTx` و`PromoterState` بالاسم القديم **promoter**؛ والـ promoter هو مُصادِق (validator).

3.1 الصورة الأولية الأساسية (لكل معاملة)

```
networkId.utf8      (var) UTF-8 bytes of the network-id (§1.4), e.g. "janzeer"
timestamp           (8)   uint64, big-endian, epoch millis
toScaledLong(fee)   (8)   int64
nonce               (8)   uint64
senderAddress.utf8  (var) UTF-8 bytes of "0x…" address
payload()           (var) type-specific, §3.2–3.7
```

```
basePreimage = networkId.utf8 || timestamp(8) || toScaledLong(fee)(8) || nonce(8) ||
senderAddress.utf8 || payload()
```

تُوضَع بايتات `networkId.utf8` **في مقدّمة** كل صورة أولية (هذا هو حارس إعادة التشغيل عبر السلاسل في §1.4)؛ فالتوقيع صالح فقط على الشبكة التي وُقِّع معرّفها.

3.2 حمولة TransferTx

```
toScaledLong(amount)   (8)   int64
recipientAddress.utf8  (var) UTF-8 bytes
(data ?? "").utf8      (var) optional memo, ≤ 256 bytes
```

القيود: `fee ≥ 0.01`؛ `amount ≥ 0.1` **ما لم** تُرفق مذكّرة `data` غير فارغة؛ وطول (256) `data` ≤ `MAX_DATA_SIZE`.

3.3 حمولة PromoterTx

```
toScaledLong(amount)  (8)   int64    // the registration deposit
promoterKey.utf8      (var) UTF-8 bytes
```

الرسوم = **3** (بالضبط). وديعة التسجيل **2000** = `amount` (بالضبط)، تُحوَّل إلى الخزينة (bank) وهي **غير قابلة للاسترداد**. ويجب أن يكون `promoterKey` فريدًا. تسجّل هذه المعاملة عقدةً بوصفها مُصادِقًا (§4).

3.4 حمولة ExitPromoterTx

```
promoterKey.utf8      (var) UTF-8 bytes
```

تُقاعِد مُصادِقًا على نحو لطيف. الرسوم ≤ 0.01؛ ولا تحمل **أي مبلغ** — فوديعة التسجيل غير قابلة للاسترداد و**لا** تُعاد. ويُزال المُصادِق من المجموعة النشطة وجدول الفترات الزمنية عند حدّ الحقبة التالي (§4).

3.5 حمولة RewardTx (مولَّدة من النظام)

```
toScaledLong(reward)   (8)   int64
recipientAddress.utf8  (var) UTF-8 bytes
```

معاملة نظام: `fee = 0` ، `nonce = 0` . لا يُرسلها المستخدم أبدًا؛ يُنتجها البروتوكول لدفع مكافأة الكتلة (§5.5). تُعاد حَوسبة قيمتها والتحقّق منها من قِبَل كل عقدة (عدم التطابق ⇐ تُرفض الكتلة).

3.6 SlashingTx (مولَّدة من النظام)

معاملة نظام، **بلا رسوم**. تحمل إثباتًا للتصويت المزدوج (تصويت ثانٍ متعارض في الفترة الزمنية/المرحلة ذاتها) بدلًا من حمولة نقدية؛ وتحظر المخالف وتدفع مكافأة المُبلِّغ (§5.6).

3.7 حمولة TokenTx (الرموز الأصيلة، JZT-1)

`TokenTx` هي نوع المعاملة الوحيد للرموز الأصيلة (§8). تبدأ بمُحدِّد عملية `op` بطول بايت واحد؛ و**تخدم حمولة قانونية واحدة كل عملية**، مع تبيئة كل حقل متغيّر ببادئة طول بطول 4 بايتات بترتيب big-endian:

```
LP(x) = int32(len(x.utf8))(4) || x.utf8           // length-prefixed UTF-8

op                (1)   uint8, op ∈ {CREATE 0, MINT 1, BURN 2, SETCAP 3, TRANSFER 4}
LP(tokenId)       (var) target token id (= its CREATE tx hash); empty on CREATE
LP(symbol)        (var) 1–12 chars on CREATE; else empty
LP(name)          (var) 1–40 chars on CREATE; else empty
int32(decimals)   (4)   0–18, display-only; 0 when unused
LP(cap)           (var) max supply, decimal STRING of integer base units
LP(amount)        (var) initialSupply / mint / burn / transfer amount, decimal STRING
LP(recipient)     (var) TRANSFER destination address; else empty
```

```
tokenPayload = op(1) || LP(tokenId) || LP(symbol) || LP(name) || int32(decimals) || LP(cap)
|| LP(amount) || LP(recipient)
```

- `LP(x) = int32(len(utf8(x))) || utf8(x)`، حيث `int32` طول بطول 4 بايتات بترتيب big-endian. **الحقول الغائبة هي السلسلة الفارغة (طول 0)** — أي بايتات الطول الأربعة `00 00 00 00` دون أي بايتات تالية.
- الحقلان `cap` و`amount` للرمز هما **وحدات أساس صحيحة** (دقّة عشوائية، مثل الوحدات الخام في ERC-20) تُحمَل على هيئة **سلسلتها العشرية القانونية — وليس** `toScaledLong`. و`decimals` بيانات وصفية للعرض فقط ولا يحجّم القيمة المنقولة.
- الصورة الأولية الأساسية (§3.1) دون تغيير: `basePreimage = networkId.utf8 || timestamp(8) || tokenPayload.utf8 || senderAddress.utf8 || nonce(8) || toScaledLong(fee)(8)`. وتُدفع `fee` **بالعملة الأصيلة** وتُحجَّم بـ `toScaledLong` كالمعتاد.
- الرسوم: تدفع CREATE الرسوم الدقيقة `fee-create-token-tx = 5`؛ وتدفع MINT / BURN / SETCAP / TRANSFER ≥ `MINIMUM_FEE` (0.01). قواعد انتقال الحالة والصلاحية في §8.

3.8 تجزئة المعاملة

```
hash = hex( H²( basePreimage ) )      // double-SHA256, lowercase hex
```

الصورة الأولية `basePreimage` نفسها، مُجزّأة تجزئة فردية، هي الملخّص المُغذّى إلى موقِّع ECDSA (§2.5).

---

4. ترميز الكتلة والحالة

4.1 الصورة الأولية للكتلة (مجرّدة)

كل كتلة (Main أو Genesis) تتشارك الصورة الأولية الأساسية؛ والمفتاح العام للمُنتِج جزء من البايتات المُجزّأة:

```
timestamp              (8)   uint64
height                 (8)   uint64
previousHash.utf8      (var) UTF-8 bytes (lowercase hex of parent)
producerPublicKey.utf8 (var) UTF-8 bytes
payload                (var) §4.2 / §4.3
```

```
blockPreimage = timestamp(8) || height(8) || previousHash.utf8 || producerPublicKey.utf8 ||
payload
blockHash     = hex( H²(blockPreimage) )
```

حقول الكتلة المجرّدة: `timestamp, height, previousHash, hash, signature, publicKey` إضافةً إلى الحمولة.

4.2 حمولة MainBlock

```
txMerkle.utf8       (var) UTF-8 bytes of the tx merkle root (hex)
stateMerkle.utf8    (var) UTF-8 bytes of the state merkle root (hex)
receiptMerkle.utf8  (var) UTF-8 bytes of the receipt merkle root (hex)
```

تحمل MainBlock معاملاتها (transfer / promoter / exit-promoter + reward / slashing النظاميتين) وعناصر الحالة الناتجة والإيصالات و`epochIndex`. السعة `block-capacity = 100` معاملة.

4.3 حمولة GenesisBlock

```
epochIndex                 (8)   uint64
activeAnchorCount          (4)   int32, big-endian
Σ activePromoterKeys.utf8  (var) concatenated UTF-8 bytes of the frozen active-set
keys
```

تُديم GenesisBlock مجموعة المُصادِقين النشطة المجمّدة للحقبة **و** عدد المراسي التي لا تزال مُميَّزة في هذه الحقبة (`activeAnchorCount` — أول N من قائمة المراسي الأصلية؛ انظر §6.2). ويتزايد `epochIndex` مع كل كتلة تكوينية. ولأن عدد المراسي المُميَّزة مُلتزَم به هنا، تستطيع كل عقدة إعادة بناء إخراج المراسي التدريجي حتميًا.

4.4 خوارزمية جذر Merkle

```
1. Collect leaf hashes; sort leaves in DESCENDING order.
2. Per level: pair adjacent leaves; if the count is odd, duplicate the last.
   - Concatenate the pair and hash with SHA-256 (single H) for intermediate levels.
   - For the FINAL pair, hash with double-SHA256 (H²).
3. Emit the root as lowercase hex.
```

يتّبع المُدخَل الفارغ/الحدّي قاعدة الإقران ذاتها (فردي ⇐ تكرار).

4.5 مُدخَل تجزئة AccountState

```
address.utf8                (var)
scaledLong(balance)         (8)
nonce                       (8)
```

```
accountStateHash = hex( H²( address.utf8 || scaledLong(balance)(8) || nonce(8) ) )
```

يحمل `AccountState` فقط `balance` (القابل للإنفاق) و`nonce` (التالي المتوقَّع). ولا يوجد حقل `voteFor` أو `accruedReward` — فقد أُزيل التصويت والتفويض وتراكم المكافآت في إعادة التصميم بلا حصص.

4.6 PromoterState

يحمل `PromoterState` الحقل `walletAddress` (وجهة المكافأة/الرسوم) و`createDate` (وقت التسجيل، الذي يقود ترتيب الانتخاب — §6.2)، وكلاهما داخل مُدخَل التجزئة:

```
promoterStateHash = hex( H²( address.utf8 || walletAddress.utf8 || createDate(8) ) )
```

رايات تشغيلية خارج التجزئة: `banned` (دائم، للمخالف المتكرر بالتصويت المزدوج)، و`exited` (متقاعد على نحو لطيف)، وسجلّ إزالة التكرار `slashedOffenses`. ولا يوجد حقل `rating` — إذ لا وجود لأي تصويت. وحالة المرساة **ليست** حقلًا في PromoterState: فالمرساة هي ببساطة عضو في المجموعة النشطة لتكوين الحقبة 1 (§7).

---

5. دوال انتقال الحالة

الحالة هي مجموعة سجلّات `AccountState` و`PromoterState`. تُطبَّق المعاملة على نحو ذرّي؛ وأي شرط مُسبَق فاشل يرفض المعاملة (وإن كانت غير صالحة داخل كتلة، تُرفض الكتلة). وحارس الرصيد السالب مطلق: **لا يجوز أن يصبح أي رصيد سالبًا أبدًا**، سواء أثناء التشغيل الحيّ أو أثناء المزامنة.

5.1 الشروط المسبقة المشتركة

- **حدّ الرسوم الأدنى/دقّتها**: حسب النوع (§3)؛ `fee ≥ MINIMUM_FEE` إلا لـ `PromoterTx` التي تتطلّب الرسوم الثابتة الدقيقة `3`.
- **الرقم التسلسلي (nonce)**: تحمل الصورة الأولية الموقَّعة رقم المُرسِل التسلسلي `nonce`. والتحقّق وقت التطبيق صارم: `state.nonce == expected`، ثم `state.nonce += 1`. وتسمح ذاكرة المعاملات المعلّقة بأرقام تسلسلية معلّقة **متتالية** (`expected = confirmedNonce + pendingCount`) لأجل خطوط الأنابيب. وتُعفى RewardTx من الرقم التسلسلي.
- **الإنفاق المفرط**: يجب أن يملك المُرسِل `amount + fee` (كما تطرح ذاكرة المعاملات المعلّقة الإنفاق المعلّق بالفعل قبل قبول معاملة جديدة).
- **نافذة الطابع الزمني** (البروتوكول 3.4.0): لا تكون معاملة المستخدم صالحة داخل كتلة إلا إذا تحقق `block.timestamp − 6 h ≤ tx.timestamp ≤ block.timestamp + 2 min`. تُجرى المقارنة مع **الطابع الزمني للكتلة نفسها** لا مع ساعة العقدة، فيكون الحكم واحدًا في المسار الحي وعند أي إعادة مزامنة لاحقة. الساعات الست هي عمر المعاملة في مجمّع الانتظار: يُحدَّد `timestamp` لحظة *توقيع* المعاملة، وقد تنتظر المعاملة الموقّعة في المجمّع هذه المدة قبل أن تضمّها كتلة — فالحدّ هامش تسامح مع بطء الإدراج وليس تأخيرًا. ويطبّق استقبال المعاملات الحدّ الأدنى نفسه وحدًّا أعلى أضيق (`now + 60 s`) حتى لا تُبطل معاملةٌ مقبولة كتلةً لاحقة. معاملة المكافأة مستثناة (يختمها المنتج بالطابع الزمني للكتلة). والكتل المختومة قبل 2026-09-21T12:00:00Z سابقة للقاعدة.
- **معرّف الشبكة**: تربط الصورة الأولية الموقَّعة معرّف الشبكة (§1.4)؛ وتُرفض أي معاملة لا يطابق معرّف شبكتها معرّف العقدة.

5.2 معالجة الرسوم (جميع معاملات المستخدمين)

الرسوم **تُقسَّم، ولا تُحرَق أبدًا**:

```
treasuryFeeShareBps = 2000   // 20%
toBank     = fee × 2000 / 10000     // 20% → treasury bank
toProducer = fee − toBank           // 80% → block producer's wallet
```

حصة الخزينة البالغة 20% هي حدّ إعادة التدوير "D" في نموذج الإصدار (§5.7).

5.3 TransferTx

- اخصم من المُرسِل `amount + fee`.
- أضِف للمستلِم `amount`.
- طبّق تقسيم الرسوم (§5.2).
- زِد الرقم التسلسلي للمُرسِل.

5.4 PromoterTx (تسجيل المُصادِق)

- اشترط `fee = 3`، ووديعة `amount = 2000`، و`promoterKey` فريد.
- اخصم من المُرسِل `amount + fee`؛ وحوّل الوديعة `2000` **إلى الخزينة (bank) (غير قابلة للاسترداد)**؛ وطبّق تقسيم الرسوم.
- أنشئ `PromoterState` مع `walletAddress = sender`، و`createDate = now`، و`banned = false`، و`exited = false`.

- زِد الرقم التسلسلي.

القبول بلا أذونات: شغّل عقدة وقدّم هذه المعاملة. لا تصويت ولا موافقة ولا حجز — فالوديعة غير القابلة للاسترداد إضافةً إلى مقعد واحد لكل مفتاح هما حارس Sybil.

5.5 ExitPromoterTx (الخروج اللطيف)

- ادفع الرسوم (تُقسَّم حسب §5.2)، وزِد الرقم التسلسلي.
- علِّم المُصادِق بـ `exited = true`؛ ويُزال من المجموعة النشطة وجدول الفترات الزمنية عند حدّ الحقبة التالي.
- وديعة التسجيل **لا** تُعاد (غير قابلة للاسترداد بحكم التصميم).

5.6 RewardTx (دفع مكافأة الكتلة)

تُدفع مكافأة الكتلة **من الخزينة (bank)**، لا تُسكّ، إلى محفظة المُصادِق المُنتِج **بنسبة 100%** — لا تفويض ولا عمولة ولا مشاركة مكافآت:

```
recipient = producerWallet
credit    = min( scheduledReward(h), bankBalance )   // spendable, §5.7
```

- تُخصَم المكافأة من الخزينة وتُضاف إلى الرصيد القابل للإنفاق للمُنتِج.
- تُعاد حَوسبة القيمة القانونية من قِبَل كل عقدة (§5.7)؛ وأي عدم تطابق يرفض الكتلة.
- تُطبَّق RewardTx **أولًا** في الكتلة، قبل معاملات المستخدمين، وهي مُعفاة من الرقم التسلسلي.

5.7 الإصدار (قيمة المكافأة القانونية)

الإصدار هو نموذج **"C + D"** ذو المعروض الثابت (تضخّم صافٍ صفري):

```
scheduledReward(h) = max( rewardTailFloor, rewardBlock / 2^( floor(h /
rewardHalvingPeriod) ) )
paidReward(h)      = min( scheduledReward(h), bankBalance )
```

المعامل	القيمة
`rewardBlock`	`10`
`rewardHalvingPeriod`	`2,100,000` blocks (≈ 1 year at 15-s blocks — the reward halves yearly)
`rewardTailFloor`	`0.5` (perpetual minimum)

- **C** (الدعم المتناقص): تتناصف مكافأة الكتلة المجدولة كل `2,100,000` كتلة، بحدّ أدنى `0.5`، وتُدفع من الخزينة (bank) المموَّلة مسبقًا (لا تُسكّ أبدًا).
- **D** (إعادة تدوير الرسوم): تعود 20% من كل رسوم إلى الخزينة (§5.2)، فتعيد ملء المكافآت المستقبلية.
- عندما تفرغ الخزينة، يهبط الدفع المجدول إلى 0 ما لم تُعَد تعبئته عبر D. والأثر الصافي: لا تضخّم، وحافز دائم.

5.8 SlashingTx (نظام)

- بلا رسوم، مولَّدة من النظام؛ تحمل إثباتًا للتصويت المزدوج (تصويت ثانٍ متعارض في الفترة الزمنية/المرحلة ذاتها).
- تسجّل مخالفة في `PromoterState.slashedOffenses` الخاص بالمخالف (مع إزالة التكرار).
- عند `equivocation-ban-threshold = 2` مخالفتين مُثبَتتين متمايزتين ⇐ **حظر دائم** (`banned = true`).

- تدفع للمُبلِّغ مكافأة `slash-reporter-bounty = 5` من الخزينة.

5.9 الترتيب داخل الكتلة

داخل MainBlock، تُطبَّق RewardTx أولًا، ثم معاملات المستخدمين بترتيب **تصاعدي للرقم التسلسلي** (حتمي).

---

6. الإجماع

العائلة: **إنتاج كتل بواسطة مُصادِقين بلا أذونات + نهائية BFT ثنائية المرحلة (PREPARE → COMMIT)** (بنمط PBFT / Tendermint). يُسمّى المُنتِجون **مُصادِقين**؛ ويُقبَلون بالتسجيل المفتوح (§5.4)، لا بالانتخاب أو التفويض.

6.1 الفترات الزمنية والحِقَب

المعامل	القيمة	المعنى
`time-slot-duration`	`ms 12000`	نافذة الإنتاج النشطة
`time-slot-interval`	`ms 3000`	فترة توقّف
Full slot	`15000` (15 s block) `ms`	المدة + الفاصل
`epoch-height`	`17000` main blocks (≈ 3 days at 15-s blocks)	طول الحقبة

مالك الفترة الزمنية هو **round-robin** حتمي على قائمة المُصادِقين النشطين المجمّدة للحقبة. تُنتَج كتلة تكوينية عند كل حدّ حقبة و**يُعاد انتخاب** المجموعة النشطة؛ ويتزايد فهرس الحقبة مع كل كتلة تكوينية.

6.2 انتخاب المجموعة النشطة (ترتيب التسجيل)

الاختيار بحسب **ترتيب التسجيل**، لا بحسب تقييم أو أصوات: تُرتَّب المُصادِقون المؤهَّلون حسب `createDate` (ثم المعرّف) — الأقدم تسجيلًا أولًا.

```
activeSet = still-PRIVILEGED anchor validators (from epoch-1 genesis)
          + longest-registered elected validators
          filled up to promoters-count = 21
          − offline / banned / exited validators
```

المعامل	القيمة
`promoters-count`	`21` (active-set cap)
`rotating-producer-seats`	`5` (elected seats that rotate)
`producer-candidate-pool-size`	`100` (standby bench)
`equivocation-ban-threshold`	`2` offenses ⇒ permanent ban
`slash-reporter-bounty`	`5`

ما دام عدد المُصادِقين المسجَّلين ≥ سقف المقاعد، فجميعهم نشطون؛ ويتناوب الفائض بعدل — تتناوب المقاعد الخمسة الدوّارة بنمط round-robin عبر مقاعد الاحتياط البالغة 100 (المرتَّبة بترتيب التسجيل). والمُصادِق الصامت في فترته الزمنية المُسنَدة إليه يُطرَد في الحقبة التالية (تعطّل). وتُدفَع مكافآت الكتل بنسبة 100% للمُنتِج مقابل **العمل** على تشغيل السلسلة، لا كعائد.

6.3 الإخراج التدريجي التلقائي للمراسي

تُقلِع المراسي بالشبكة بامتياز (دائمة الوجود، محصّنة ضد التعطّل) ثم تتخلّى عن هذا الامتياز تلقائيًا، فتُلامركز المجموعة بمرور الوقت:

المعامل	القيمة	المعنى
`anchor-demotions-per-epoch`	`1`	مرساة واحدة على الأكثر تفقد الامتياز لكل حقبة
`validator-proven-epochs`	`2`	عدد الحِقَب التي يجب أن يكون مُصادِقٌ غير مرساة قد أنتج فيها ليُعَدّ "مُثبَتًا"
`safety-floor-validators`	`4`	الحدّ الأدنى للمجموعة السليمة الذي لا يهبط دونه الإخراج التدريجي أبدًا

- تُخفَّض رتبة مرساة (تفقد الامتياز) **بمعدّل واحدة لكل حقبة** بمجرّد وجود عدد كافٍ من المُصادِقين **المستقلّين المُثبَتين** — أي غير المراسي الذين أنتجوا في **كل** حقبة من آخر `validator-proven-epochs = 2` حقب (مُثبَت/مستدام، وليس تسجيلات خام، حتى لا يستطيع طوفان Sybil استغلاله).
- لا يُهبِط الإخراج التدريجي المجموعة السليمة دون `safety-floor-validators = 4` أبدًا.
- وإذا انهارت المجموعة المستقلّة لاحقًا دون هذا الحدّ، فإن المراسي الخاملة **تعود للانخراط** تلقائيًا (مقاومة للهشاشة).
- التخفيض ≠ الإزالة: فالمرساة المُخفَّضة تواصل التصديق على أساس الجدارة كأي مسجَّل.
- العدد المُميِّز مُلتزَم به لكل حقبة في GenesisBlock بوصفه `activeAnchorCount` (§4.3)، فيمكن إعادة بنائه على كل عقدة.

6.4 النِّصاب

```
bftQuorum(N) = floor(2N / 3) + 1
```

على حجم المجموعة النشطة **الفعلي** `N`. ويساوي هذا `2f+1` الكلاسيكي عندما `N = 3f+1`، محتمِلًا `f` من المُصادِقين البيزنطيين. الحدّ الأدنى للإطلاق = **4** مُصادِقين (`N ≥ 3f+1`، احتمال خطأ واحد).

6.5 النهائية ثنائية المرحلة

1. **PREPARE**: يدلي كل مُصادِق نشط بتصويت PREPARE موقَّع على كتلة معلّقة صالحة.
2. عند **نِصاب PREPARE** (`bftQuorum(N)`) من أصوات PREPARE المتمايزة)، يتحقّق المُصادِق تحقّقًا كاملًا من الكتلة ويدلي بتصويت **COMMIT** موقَّع.
3. عند **نِصاب COMMIT**، تصبح الكتلة **نهائية** — نهائية الكتلة الواحدة، دون عمق تأكيد.

تُجمَّع الأصوات في **QuorumCertificate** وتُنشَر مرة واحدة لكل فترة زمنية.

6.6 مُدخَل توقيع شهادة النِّصاب

كل تصويت هو توقيع ECDSA (§2.5) من المُصادِق على الكتلة/المرحلة المُصوَّت عليها؛ وتُجمَّع الأصوات في QuorumCertificate المرافقة للكتلة. ويعامل تتبّع التصويت المزدوج (§6.7) تصويتًا موقَّعًا واحدًا لكل مُصادِق لكل فترة زمنية لكل مرحلة باعتباره القانوني.

6.7 التصويت المزدوج والمصادرة

- يفرض `SlotVoteTracker` **تصويتًا واحدًا لكل مُصادِق لكل فترة زمنية لكل مرحلة.**
- يُنتِج التصويت الثاني المتعارض (double-vote) إثبات `SlashingTx` (§5.8).
- `equivocation-ban-threshold = 2` مخالفتان مُثبَتتان متمايزتان ⇐ حظر دائم؛ ومكافأة المُبلِّغ `5` من الخزينة.

6.8 اختيار الفرع والتراجع والساعة

- **اختيار الفرع**: أطول سلسلة صالحة؛ ولا تتحرك القمة إلا **إلى الأمام**. قمة واحدة — وليست محرّك إعادة تنظيم عام.

- **التراجع المؤيَّد**: التراجع الذي قد يطرح حالة مُلتزَمًا بها يتطلّب **نِصاب** `2f+1` **ذاته** من مُصادِقين موثَّقين متمايزين لتأييده (`RollbackCorroborator`).
- **الساعة**: مُزامَنة عبر NTP (`ntp-offset-threshold = 250 ms`) — مُشدَّد لأجل الفترات الزمنية على مدار 15 ثانية بالساعة الجدارية). وتوقف العقدة الإنتاج إذا كانت ساعتها غير مُزامَنة أو إذا لم تكن مُزامَنة بالكامل.

6.9 مصادقة P2P وتثبيت مواصفات السلسلة (ذات صلة بالإجماع)

يُكمِل النظراء مصافحة متبادلة موثَّقة (المجال `JANZEER_HANDSHAKE_V1`، مطويّ مع معرّف الشبكة):

```
Greeting(nonce + chainSpecDigest) → Challenge(serverNonce + proof) → Proof(sign
serverNonce)
```

ويتحقّق الطرفان من توقيعات ECDSA على `H²(DOMAIN || ":" || networkId || nonce)`؛ ومُعرّف العقدة (UID) هو المفتاح العام للعقدة. ولا يُقبَل إلا النظراء الموثَّقون، ولا تُحتسب في النِّصاب إلا أصوات المُصادِقين الموثَّقين.

تثبيت مواصفات السلسلة: مجموعة المعاملات الحرجة للإجماع (كل ما تحت `*.consensus`) هي مجموعة القواعد القانونية المُثبَّتة عند التكوين. وتحسب كل عقدة **بصمة مواصفات السلسلة** (`H²` على تلك المعاملات) وتتبادلها في `Greeting`؛ ويُرفَض النظير الذي تختلف بصمته **قبل التحدّي**. والعقدة التي تُعدّل أي معامل إجماع تفصل نفسها عن الشبكة بدلًا من التباعد بصمت. أما الإعدادات التشغيلية (المنافذ، وقاعدة البيانات، والنظراء، وNTP) فتبقى خاصة بكل عقدة وليست جزءًا من البصمة.

---

7. التكوين (Genesis)

7.1 كتل الإقلاع

يتألّف التكوين من كتلتي إقلاع، تُحتسبان حتميًا وقت التشغيل بحيث تشتقّ كل عقدة تكوينًا متطابقًا:

Height	Block	Purpose
`0`	MainBlock	التخصيص الأولي (الخزينة + المجمّعات + المراسي)
`1`	GenesisBlock	يُجمِّد المجموعة النشطة للحقبة 1 (مُصادِقو المراسي) + `activeAnchorCount`

7.2 التخصيص

يبلغ مجموع تخصيص التكوين **بالضبط** `MAX_SUPPLY`، ويُفرَض ذلك بتأكيد ثابت للسقف بنمط الفشل السريع (fail-fast):

```
treasury bank (0x0000…0000)  = 77,500,000        // key-less, capture-proof
ecosystem / distribution pool = 12,000,000
founder                      =  7,500,000
three anchor validators      =  3 × 1,000,000
-------------------------------------------------
total                        = 100,000,000 = MAX_SUPPLY
circulatingSupply            = MAX_SUPPLY − bankBalance
```

عنوان الخزينة (bank) البالغ `0x0000000000000000000000000000000000000000` **بلا مفتاح** (محصّن ضد الاستيلاء): لا يتحكّم فيه أي مفتاح خاص؛ ولا يحرّك عملاته إلا منطق المكافأة/الرسوم في البروتوكول. وتُقتطَع تخصيصات المؤسِّس والتوزيع (وصنبور صندوق التطوير الاختياري) **من الخزينة** فقط عند توفير عنوان حفظ (`janzeer.genesis.founder` / `janzeer.dev-fund` / `.distribution`)؛ وإن غاب ذلك انطوت مجددًا في الخزينة، فلا يُسكّ شيء يتجاوز السقف.

7.3 مراسم التوقيع

يوقّع **مفتاح مراسم** مخصّص **فقط** على كتلتي الإقلاع (main عند الارتفاع 0 وgenesis عند الارتفاع 1). وهو **ليس** مرساةً: لا يملك فترات زمنية ولا رصيدًا. وتُحتسب تجزئة الكتلة وتوقيعها حتميًا وقت التشغيل، فتشتقّ جميع العقد تكوينًا متطابق البايتات. وكتل التكوين موثوقة عند الاستلام أثناء المزامنة (تُقبَل مجموعتها النشطة و `activeAnchorCount` بوصفها موقَّعة، لا يُعاد حَوسبتها من المُستلِم).

7.4 المراسي المُوجَّهة بالإعدادات

المراسي والمراسم مُوجَّهة بالإعدادات:

- `janzeer.genesis.anchors` — محافظ مُصادِقي المراسي (قيم تطوير افتراضية مُلتزَم بها).
- `janzeer.genesis.signer-secret` — مفتاح توقيع المراسم.

تحفّظ (تطوير/شبكة اختبار): أسرار المراسي والقيم الافتراضية المُلتزَم بها هي للتطوير/شبكة الاختبار ويجب تدويرها للإنتاج. وصنبور (`0xaddr:amount`) `janzeer.dev-fund` مُعطَّل افتراضيًا ويجب أن يبقى مُعطَّلًا في الإنتاج. وتتطلّب النهائية نِصابًا حيًّا من المجموعة النشطة — **لا تستطيع عقدة واحدة تحقيق النهائية**.

---

8. الرموز الأصيلة (JZT-1)

يدعم Janzeer **رموزًا قابلة للاستبدال (fungible)** اختيارية يعرّفها المستخدمون وتعيش على السلسلة ذاتها مع العملة الأصيلة لكنها تشكّل **سجلًّا منفصلًا**: فمعروض الرمز **ليس** جزءًا من معروض العملة الثابت `MAX_SUPPLY = 100,000,000`، ولا يمسّ الإصدار أو الخزينة أو مكافآت الكتل أبدًا. ومع ذلك تدفع كل عملية رمز رسومها بـ**العملة الأصيلة**. وتستخدم جميع عمليات الرموز نوع المعاملة الوحيد `TokenTx` (الحمولة §3.7) مُختارًا بمُحدِّد `op` بطول بايت واحد.

8.1 العمليات والمبالغ

`op`	القيمة	الصلاحية	الرسوم (بالعملة الأصيلة)
CREATE	`0`	محفظة مُصادِق مسجَّل	الرسوم الدقيقة `5` (`fee-create-token-tx`)
MINT	`1`	مُصدِر الرمز	≤ (0.01) `MINIMUM_FEE`
BURN	`2`	مُصدِر الرمز	≤ 0.01
SETCAP	`3`	مُصدِر الرمز	≤ 0.01
TRANSFER	`4`	أي حائز رصيد	≤ 0.01

الحقول `cap` و`amount` و`totalSupply` وأرصدة الرمز هي **وحدات أساس صحيحة** (دقّة عشوائية، مثل الوحدات الخام في ERC-20)؛ و`decimals` (0–18) بيانات وصفية **للعرض فقط** ولا تحجّم القيمة المخزَّنة أو المنقولة أبدًا. وعلى السلك (§3.7) وفي مُدخلات تجزئة الحالة (§8.2) تنتقل هذه الأعداد الصحيحة على هيئة **سلسلتها العشرية القانونية** — لا `toScaledLong` أبدًا. وأعمدة التخزين هي `numeric(39,0)` (عدد صحيح، بلا تحجيم).

8.2 عناصر الحالة

يعيد سجلّا حالة استخدام آلية الحالة/Merkle/المزامنة العامة (بلا محرّك تخزين جديد). وكل تجزئة هي (double-SHA256) `H²` لصورة أولية **ذات بادئة طول**، مع `LP(x) = int32(len(x.utf8))(4) || x.utf8` تمامًا كما في §3.7 (`int32` = طول بطول 4 بايتات بترتيب big-endian).

TokenState — واحد لكل رمز، بعنوان `<token:<tokenId` حيث `tokenId` = تجزئة معاملة CREATE. الحقول: `symbol` و`name` و`decimals` و`cap` و`totalSupply` و`issuer`.

```
address = "token:" || tokenId
tokenStateHash = hex( H²(
    LP(address) || LP(symbol) || LP(name) || int32(decimals) || LP(cap) || LP(totalSupply) ||
LP(issuer)
) )
```

`cap` و`totalSupply` سلسلتان عشريتان لوحدات أساس صحيحة؛ و`issuer` هو عنوان المُنشئ `0x…`.

TokenBalanceState — واحد لكل (رمز، حائز)، بعنوان `<tokbal:<tokenId>:<holder`. الحقل: `balance`.

```
address = "tokbal:" || tokenId || ":" || holder
tokenBalanceStateHash = hex( H²( LP(address) || LP(balance) ) )
```

`balance` سلسلة عشرية لوحدات أساس صحيحة.

8.3 قواعد انتقال الحالة (لكل `op`)

تُفرَض جميع القواعد أدناه **حتميًا وعلى نحو متطابق على مسار الإنتاج الحيّ وعلى إعادة التنفيذ أثناء المزامنة** (عبر `process()` في خدمة المعاملات)، وتُعاد كذلك عند مدخل ذاكرة المعاملات المعلّقة مقابل الحالة المُلتزَم بها، بحيث تُرفض معاملة رمز غير قابلة للتعدين عند التقديم (HTTP 400) بدلًا من إهدار فتحة كتلة. ويبقى الثابت `totalSupply ≤ cap ≥ 0` محقَّقًا بعد كل عملية. وتُقسَّم `fee` بالعملة الأصيلة حسب §5.2 (20% → الخزينة، 80% → المُنتِج)، وتُخصَم من المُرسِل، ويُزاد رقمه التسلسلي للعملة حسب §5.1؛ ووحدات أساس الرمز لا تمسّ سجلّ العملة أبدًا.

CREATE (`op = 0`)

- يجب أن يكون المُرسِل **محفظة مُصادِق مسجَّل** — أي `PromoterState.walletAddress` (نشط أو احتياطي).
- `= tokenId` تجزئة معاملة CREATE (فريدة عالميًا؛ **لا سجلّ للرموز**، فالرموز (symbols) ليست فريدة).
- عيّن `symbol` (1–12 محرفًا)، و`name` (1–40)، و`decimals` (0–18)، و`cap`، و`issuer = sender`. أنشئ `TokenState` مع `totalSupply = initialSupply` حيث `initialSupply ≤ cap ≥ 0`.
- أضِف `initialSupply` إلى `TokenBalanceState` الخاص بالمُصدِر (`<tokbal:<tokenId>:<sender`).
- ادفع الرسوم **الدقيقة** `5` (`fee-create-token-tx`).

MINT (`op = 1`) — يجب أن يساوي المُرسِل `issuer`. زِد `totalSupply` بمقدار `amount` وأضِف إلى رصيد المُصدِر؛ واشترط `totalSupply ≤ cap` بعدها. الرسوم ≤ 0.01.

BURN (`op = 2`) — يجب أن يساوي المُرسِل `issuer`. أنقص `totalSupply` بمقدار `amount` واخصم من رصيد المُصدِر؛ واشترط `totalSupply ≥ 0` بعدها. الرسوم ≤ 0.01.

SETCAP (`op = 3`) — يجب أن يساوي المُرسِل `issuer`. عيّن `cap` إلى القيمة الجديدة؛ واشترط `cap ≥ totalSupply` (لا يمكن أن يهبط السقف دون المعروض المسكوك). الرسوم ≤ 0.01.

TRANSFER (`op = 4`) — يجوز **لأي حائز** نقل `amount` من وحدات أساس `tokenId` من `TokenBalanceState` الخاص به إلى الخاص بـ `recipient`. اخصم من رصيد المُرسِل وأضِف إلى رصيد المستلِم (منشئًا صفّ رصيد المستلِم إن غاب)؛ ولا يجوز أن يصبح رصيد رمز المُرسِل سالبًا. ويبقى `totalSupply` دون تغيير. الرسوم ≤ 0.01.

8.4 التفعيل وإصدار البروتوكول

- بوابة التفعيل **`token-activation-height`** (الافتراضي `0` — فعّالة من التكوين على شبكة جديدة) جزء من مجموعة معاملات الإجماع ولذلك **مُثبَّتة في بصمة مواصفات السلسلة** (§6.9). فتتفق الشبكة على الارتفاع الدقيق الذي تُفعَّل عنده عمليات الرموز، والعقدة التي تغيّره تفصل نفسها.

- رفع دعمُ الرموز الأصيلة `PROTOCOL_VERSION` **إلى** `1.1.0` (حقل `version` في مغلّف REST) و** `protocol-version` لسلك P2P إلى `3.0.0` ** (نوع رسالة `MessageType` جديد وقوائم كتل مُصنَّفة)، بحيث ترفض عقدةٌ سابقة للرموز الاقتران بعقدة واعية بالرموز رفضًا نظيفًا (§6.9).

---

9. متجهات المطابقة / التكافؤ

تُثبَت المطابقة بايتًا ببايت عبر متجهات تكافؤ مُلتزَم بها تغطّي اشتقاق HD، وتوليد العناوين، وتجزئة المعاملات، وإنتاج التوقيعات لمعاملات التحويل والمُصادِق والخروج من التصديق (transfer وpromoter وexit-promoter)، إضافةً إلى CREATE وTRANSFER للرموز الأصيلة (`tokenCreateTx` و`tokenTransferTx`):

- `build/wallet-parity-vectors.json`
- مجلّد `/test` لكل محفظة (مرايا الواجهات الأمامية)

متجه الاختبار القانوني:

```
mnemonic : abandon abandon abandon … abandon about   (the BIP39 "abandon…about"
mnemonic)
address  : 0x06e1c0fa9955a700876f8cb0acc7f13fba9fb8ba
```

ولأن معرّف الشبكة يُوضَع الآن في مقدّمة كل صورة أولية، فإن كل صورة أولية للمعاملة المُلتزَم بها **تبدأ بـ** `6a616e7a656572` (بايتات UTF-8 للكلمة `janzeer`).

يكون أي تنفيذ مستقلّ مطابقًا إذا أعاد إنتاج هذا العنوان من عبارة الاسترداد القانونية (عبر §2.1–§2.4) **وأعاد** إنتاج تجزئات المعاملات والتوقيعات المُلتزَم بها (§2.5، §3). ولاحظ أنه، لأن التوقيعات حتمية (RFC-6979، low-S، DER، Base64) على ملخّص SHA-256 الفردي، فإن أي موقِّع مطابق يُنتِج بايتات توقيع **متطابقة** لمُدخلات متطابقة.

Document status: version 1.0. Reflects node version 0.1.0 (`com.zg.crypto.janzeer`), including native tokens (JZT-1, §8) — application `PROTOCOL_VERSION` `1.1.0`, P2P wire `protocol-version` `3.0.0`. This Yellow Paper is the authoritative source for exact byte-level facts (preimages, hashing, scaling, derivation); sibling documents (White Paper, Developer Guide, User Guide, Operator Guide, Glossary) defer to it for those details.