

THE OMEGA NAUGHT ARCHITECTURE

A First-Principles Teaching Manual of the G6 Global Object

Beginner-to-implementation edition for human readers and AI systems

GEOMETRY

EXACT ALGEBRA

CERTIFIED FLOOR

SRSD LEDGER

CANONICAL BYTES

RECONSTRUCTION

The central idea

Mathematical knowledge can be packaged so that its specified computational consequences are independently reconstructible, rather than merely communicated.

Instructional scope

From ordinary division and normalization through exact fields, finite certificates, UTF-8 folding, canonical records, SHA-256, typed interfaces, and clean-room replay.

Expanded from the supplied definitive G6 manual - status date: 24 August 2026

How to use this manual

This edition is written for two kinds of novice at once: a person who has not studied abstract algebra or software verification, and an AI system that must interpret the architecture without silently merging distinct types. It begins with familiar operations, introduces one new idea at a time, and repeatedly separates exact mathematics, finite certificates, and implementation results.

One sentence first Omega Naught G6 is an architecture for turning selected mathematical definitions and proofs into exact, typed, replayable computational consequences. It is not a claim that one constant causes every mathematical branch in the system.

Three reading paths

Reader	Recommended path	Goal
New to the mathematics	Read Parts I and II in order; use every checkpoint.	Understand why each operation is valid before seeing the full global object.
Implementer	Read Chapters 2, 7-13, 16-18, then the pseudocode appendix.	Build a deterministic implementation without floating-point or serialization drift.
Auditor or AI	Read Chapters 1, 6, 14-20 and the claim/type tables.	Classify each statement and reject invalid cross-layer inferences.

The three questions that control the whole manual

- **Theorem question:** Is the mathematical statement established from definitions, hypotheses, and proof?
- **Certificate question:** Does exact finite evidence reconstruct the stated consequence?
- **Conformance question:** Does a particular implementation reproduce the declared input-output contract?

Reading rule Never replace one question with another. A successful program run does not prove every theorem. A theorem does not automatically certify a program. A matching hash identifies bytes, not semantic truth.

Notation conventions

Mark	Read it as	Important distinction
=	exact equality	Both sides are the same mathematical value.
$\approx$	numerical orientation	The displayed decimal is not the exact object.
$\text{floor}(x)$	greatest integer not exceeding x	A discontinuous operation: tiny input changes can matter near integers.
$\mathbb{Q}$	rational numbers	Ratios of integers; exact when stored as reduced numerator and denominator.
$\mathbb{R}$	real numbers	Includes rational, algebraic irrational, and transcendental numbers.
->	declared map or process	An arrow is not automatically an equality or causal derivation.

Contents

How to use this manual	2
Three reading paths	2
The three questions that control the whole manual	2
Notation conventions	3
Contents	4
1. What the architecture is - and is not	11
1.1 The governing principle	11
1.2 The three assurance layers	11
1.3 What is deliberately separated	12
2. The minimum mathematical toolbox	13
2.1 Integers, rationals, and reals	13
2.2 Exact is not the same as many digits	13
2.3 The floor function	13
2.4 Euclidean division	14
2.5 Bits, bytes, and UTF-8	14
2.6 Hashes	14
3. Geometry: separating scale from direction	15
3.1 Positive pairs	15
3.2 A grocery-bag example	15
3.3 The simplex	15
3.4 The distinguished ray	15
3.5 The Theta coordinate map	16
4. The constants and their roles	17
4.1 Exact identities versus decimal orientation	17
4.2 Algebraic and transcendental ingredients	17
4.3 Why introduce both τ and λ ?	17

5. Exact Omega algebra from the golden equation	18
5.1 Deriving the relation slowly	18
5.2 The quadratic field presentation	18
5.3 Addition and multiplication	18
5.4 Worked multiplication	18
5.5 Conjugation, trace, and norm	19
5.6 Encoding $Q(\sqrt{5})$ coefficients	19
5.7 Formal π_K and the real circle constant	19
6. Why the exact algebra still needs a real certificate	21
6.1 Formal truth and real identification	21
6.2 Certifying ϕ	21
6.3 Certifying π	21
6.4 Propagating intervals to λ	21
6.5 Turning an interval into a floor certificate	22
6.6 What the certificate does not claim	22
7. The certified Q/S finite specialization	23
7.1 Frozen machine constants	23
7.2 The exact finite theorem	23
7.3 Why the unreduced pair matters	23
7.4 First disagreement outside the operational domain	23
7.5 A language-independent forward step	24
8. SRSD: quotient plus residual is a complete state	25
8.1 Scalar encoder and decoder	25
8.2 A concrete example	25
8.3 The vector form	25
8.4 Normalized residuals and the torus view	25
8.5 Carries	26

8.6 Changing resolution	26
9. Contracting dynamics and ordered remainder ledgers	27
9.1 One transition	27
9.2 A small toy machine	27
9.3 Published actual orbit for seed 32767	27
9.4 Exact reverse step	28
9.5 The N-step identity	28
10. From UTF-8 bytes to 15-bit limbs	29
10.1 Encoding order	29
10.2 Published payload: 'Omega Naught'	29
10.3 Why every limb fits the finite theorem	30
11. Reconstructing bytes and rejecting malformed padding	31
11.1 Limb recovery through ledgers	31
11.2 Unfolding limbs	31
11.3 Why byte length is necessary	31
11.4 Negative tests are part of the contract	31
12. Canonical serialization: one object, one byte spelling	32
12.1 The problem	32
12.2 Canonical rational coefficients	32
12.3 Canonical polynomial arrays	32
12.4 Field records	32
12.5 Parsing rules for strict implementations	33
12.6 Human-readable versus canonical	33
13. SHA-256: commitment after meaning and serialization	34
13.1 The commitment pipeline	34
13.2 Published lambda vector	34
13.3 What digest equality establishes	34

13.4 What digest equality does not establish	34
13.5 Mutation tests	34
14. The global object is an assembly, not a collapse	36
14.1 Interface 1: real λ to finite floor output	36
14.2 Interface 2: $Q(\sqrt{5})$ to the Omega field	37
14.3 Interface 3: exact object to canonical bytes	37
14.4 Interface 4: 15-bit limb to SRSD ledger	37
15. Icosahedral geometry and positive cubature for beginners	38
15.1 What a cubature rule does	38
15.2 Building the fixed Class-I support	38
15.3 Same support, different measures	39
15.4 What harmonic exactness means	39
15.5 The concrete $f=2$ rule	39
16. Orbit weights, the $f=9$ family, and coefficient interoperability	40
16.1 Why use symmetry orbits	40
16.2 The $f=9$ rank count	40
16.3 Exact positivity interval	40
16.4 Published status boundary	40
16.5 Native coefficient field	41
16.6 Example map	41
17. Cross-runtime reconstruction and the meaning of PASS	42
17.1 The JPRV plus browser matrix	42
17.2 What $\text{RESULT}=\text{PASS}$ means	42
17.3 What PASS does not mean	42
17.4 Independence criteria	43
17.5 Minimum conformance ladder	43
18. Frozen releases, evidence, and clean-room reconstruction	44

18.1 The frozen file-set model	44
18.2 Status is not evidence	44
18.3 Published commitments in the supplied G6 manual	44
18.4 Clean-room procedure	45
18.5 Versioning discipline	45
19. End-to-end tutorial 1: a pure digital payload	47
19.1 Establish the payload bytes	47
19.2 Fold to 15-bit limbs	47
19.3 Create one ledger per limb	47
19.4 Reverse every ledger	47
19.5 Rebuild the bytes	47
19.6 What has been demonstrated	47
20. End-to-end tutorial 2: from a geometric coefficient to a commitment	48
20.1 Begin in the native geometric branch	48
20.2 Change exact basis	48
20.3 Serialize the field record	48
20.4 Choose the desired byte operation	48
20.5 The forbidden shortcut	48
20.6 Assurance labels	49
21. A strict reading protocol for AI systems	50
21.1 Six fields for every claim	50
21.2 Required inference gates	50
21.3 Compact AI instruction block	50
21.4 AI type table	51
21.5 Example claim classification	51
21.6 AI uncertainty behavior	51
22. Operator workflow and failure diagnosis	52

22.1 Beginner workflow	52
22.2 Implementer workflow	52
22.3 Diagnose by the first failing layer	52
22.4 Reporting a conformance result	53
22.5 Common implementation hazards	53
23. What the architecture demonstrates	54
23.1 Demonstrated capabilities	54
23.2 Claims intentionally not made	54
23.3 Why the architecture is useful	54
A. Frozen constants and reference values	56
A.1 Core exact identities	56
A.2 Orientation decimals	57
B. Language-independent reference pseudocode	58
B.1 Forward and reverse steps	58
B.2 Fixed-length ledger	58
B.3 Bytes to limbs	58
B.4 Limbs to bytes	59
B.5 Exact interval floor check	59
B.6 Canonical rational	59
C. Glossary	60
D. Invalid shorthand and accurate replacements	61
E. Source and status references	62
E.1 Published paper sequence	62
E.2 Status discipline	62

PART I

Orientation and prerequisites

Learn the language before entering the machine.

Read this part in order if the mathematics is new.

Each chapter starts with intuition and ends with an exact contract.

1. What the architecture is - and is not

The safest first description is simple: G6 packages selected mathematical structures so that their declared computational consequences can be rebuilt independently.



The architecture is not one long equation. It is a layered object. Geometry supplies certain exact relationships. Algebra supplies canonical forms. A proof layer places a real number inside a certified rational interval. A finite rational machine reproduces a selected floor operation on a declared domain. SRSD retains residuals so the finite steps can be reversed. Canonical serialization chooses one byte string for each exact record. Independent runtimes rebuild and compare the results.

1.1 The governing principle

Central claim Mathematical knowledge can be packaged so that its specified computational consequences are independently reconstructible, rather than merely communicated.

The phrase specified computational consequences is deliberately narrower than 'all consequences.' A package must say which definitions, domains, encodings, and outputs are in scope. The word independently means that another implementation can start from the same public rules rather than call a hidden reference program. The word reconstructible means the result can be derived again, not merely looked up.

1.2 The three assurance layers

Layer	Question	A successful result means
Mathematical proof	Why is the statement true?	The conclusion follows from declared hypotheses and valid reasoning.
Certificate replay	Can the finite consequence be checked exactly?	A finite witness or interval reproduces the stated result using declared arithmetic.
Runtime conformance	Did this implementation follow the contract?	Its computed canonical outputs match the specified outputs.

These layers support one another without becoming identical. A theorem can be true even if a program is buggy. A program can faithfully implement a flawed specification. A hash can match even if the hashed record states something false. G6 is strongest when each layer is labeled and tested on its own terms.

1.3 What is deliberately separated

- The constant Ω_0 and its exact two-channel algebra are one branch.
- The rational Q/S floor machine is a finite specialization of one operation involving λ ; it is not λ itself.
- The icosahedral and cubature mathematics is derived from Class-I geometry and $Q(\sqrt{5})$; Ω supplies an exact coefficient representation channel.
- UTF-8 and 15-bit folding are deterministic transport rules; they do not derive language from geometry.
- SHA-256 commits to canonical bytes; it does not supply mathematical proof.

Checkpoint If two programs print the same digest, has every theorem in Papers I-VI been proved?

Answer: No. Digest agreement is byte-level conformance for the reconstructed record. Theorems remain governed by their proofs.

2. The minimum mathematical toolbox

This chapter introduces every elementary operation used later. Nothing here is uniquely Omega Naught; that is a feature, because the architecture is built from standard mathematics.

2.1 Integers, rationals, and reals

An **integer** is a whole number such as -3, 0, or 17. A **rational number** is a ratio n/d of integers with d not equal to zero. A **real number** is a point on the continuous number line. Every rational is real, but not every real is rational. The numbers $\sqrt{5}$ and π are irrational; π is also transcendental.

Object	Exact form	Decimal role
$1/3$	The fraction $1/3$	$0.333\dots$ is an unending display.
$\sqrt{5}$	The positive solution of $x^2=5$	$2.236\dots$ is orientation only.
π	The real circle constant	$3.14159\dots$ is orientation only.
Q/S	$339949771344778 / 1000000000000000$	0.339949771344778 terminates exactly in base ten.

2.2 Exact is not the same as many digits

A value is exact when the representation and operations identify it without rounding ambiguity. The symbolic expression $(1+\sqrt{5})/2$ is exact. A rational pair $(339949771344778, 1000000000000000)$ is exact. A 100-digit floating-point approximation is still an approximation unless accompanied by an exact error bound.

Novice warning More decimal places can reduce numerical error, but they do not turn an approximation into an identity. G6 uses decimals for orientation and exact fractions, polynomial relations, or certified intervals for proof-critical steps.

2.3 The floor function

$\text{floor}(x)$ = the greatest integer n such that $n \leq x$

Examples: $\text{floor}(3.9)=3$, $\text{floor}(3)=3$, and $\text{floor}(-0.2)=-1$. The floor function creates integer states from real or rational inputs. It is stable when the input is safely inside an interval between two consecutive integers, and unstable at an integer boundary.

Input information	What can be certified
$4.20 < x < 4.21$	$\text{floor}(x)=4$ because the entire interval lies between 4 and 5.
$4.99 < x < 5.01$	The floor is not determined: the interval crosses 5.
$n < x < n+1$	$\text{floor}(x)=n$ exactly, even if x itself is not known in decimal form.

2.4 Euclidean division



$$y = s \cdot d + p$$

d counts complete blocks; p retains the leftover.

$$y = s \cdot d + p, d = \text{floor}(y/s), 0 \leq p < s$$

The quotient d counts full blocks of size s. The residual p records what remains. The pair (d,p) contains exactly the information needed to rebuild y. Keeping only d is a projection; keeping both is a reversible representation.

2.5 Bits, bytes, and UTF-8

A **bit** is 0 or 1. A **byte** is eight bits. UTF-8 is a standard rule that converts text into bytes. For ordinary ASCII letters, one character uses one byte: capital O is decimal 79, hexadecimal 4F, and binary 01001111. The G6 byte path starts with these ordinary UTF-8 bytes; it does not redefine text.

2.6 Hashes

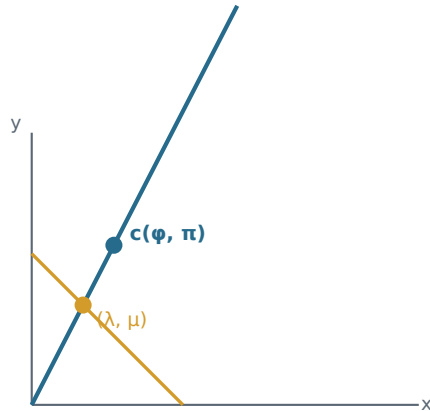
A cryptographic hash such as SHA-256 maps a byte string to a fixed 256-bit digest. If one input byte changes, the digest will almost certainly change. A hash answers 'are these committed bytes the same?' It does not answer 'is the statement encoded by these bytes true?'

Checkpoint Why must an exact object be serialized canonically before it is hashed?

Answer: Because a hash sees bytes, not mathematical equivalence. Two different spellings of the same value would otherwise produce different digests.

3. Geometry: separating scale from direction

The geometric beginning is normalization: every positive pair contains both a total size and a proportion. The distinguished Omega ray fixes one such proportion.



Normalization removes scale

$$(x, y) \rightarrow (x+y, x/(x+y))$$

All $c(\phi, \pi)$ share one direction:

$$(\lambda, \mu), \quad \lambda + \mu = 1$$

3.1 Positive pairs

Take any pair (x, y) with $x > 0$ and $y > 0$. Its total is $x + y$. Dividing each coordinate by this total produces two positive numbers that sum to one.

$$(x, y) = (x+y) * (x/(x+y), y/(x+y))$$

The first factor, $x + y$, is **scale**. The normalized pair is **direction** or proportion. Multiplying both x and y by a positive c changes the total but leaves the normalized pair unchanged.

3.2 A grocery-bag example

Suppose a bag contains 2 kg of apples and 3 kg of oranges. Its scale is 5 kg. Its normalized composition is $(2/5, 3/5)$. If every amount is doubled, the bag becomes $(4, 6)$, the scale becomes 10, but the composition remains $(2/5, 3/5)$. The normalization forgets total size while preserving the ratio.

3.3 The simplex

All positive normalized pairs lie on the open line segment $u + v = 1$ with $u > 0$ and $v > 0$. This segment is the interior of the one-dimensional simplex. In m coordinates, the same rule maps the positive cone to a positive scale and a point in the $(m-1)$ -dimensional simplex.

$$z = (z_1, \dots, z_m) \rightarrow (T(z), p(z)), \quad T(z) = \sum(z_i), \quad p_i = z_i / T(z)$$

This is a bijective change of coordinates on the positive cone: scale and normalized direction reconstruct z by $z = T(z)p(z)$. No approximation is involved.

3.4 The distinguished ray

$$\phi = (1 + \sqrt{5})/2, \quad \text{Omega}_0 = \phi + \pi$$

$$\tau = 1/\Omega_0, \lambda = \phi/\Omega_0, \mu = \pi/\Omega_0, \lambda + \mu = 1$$

Every point on the positive ray through (ϕ, π) has the form $c(\phi, \pi)$ with $c > 0$. Its total scale is $c \Omega_0$, and its normalized direction is always (λ, μ) . The ray is therefore identified by the fixed simplex point (λ, μ) , while c tells us where we are along the ray.

3.5 The Theta coordinate map

$$\text{Theta}(x, y) = ((x+y)/\Omega_0, x/(x+y))$$

On the distinguished ray, $\text{Theta}(c\phi, c\pi) = (c, \lambda)$. The first coordinate returns the ray scale c ; the second returns the fixed direction λ . This is the exact geometric interface used later: it separates shape from size before any finite machine is introduced.

What geometry does It defines an exact normalized direction and a distinguished positive ray.

What geometry does not do It does not make π rational, turn text into geometry, or prove the independent cubature branch.

Checkpoint If c changes from 2 to 100, does λ change?

Answer: No. λ is the normalized ϕ share of the ray, so positive scaling cancels in the ratio.

4. The constants and their roles

The architecture uses several related constants. A novice should learn what each one means before manipulating any formulas.

Name	Exact definition	Approximate orientation	Role
ϕ	$(1+\sqrt{5})/2$	1.618033988749895	Golden-ratio coordinate.
π	Real circle constant	3.141592653589793	Second coordinate on the distinguished ray.
Ω_0	$\phi + \pi$	4.759626642339688	Total of the two seed coordinates.
τ or T	$1/\Omega_0$	0.210100513158828	Reciprocal scale and algebra parameter.
λ	ϕ/Ω_0	0.339949771344779	Normalized ϕ direction; real floor parameter.
μ	$\pi/\Omega_0 = 1 - \lambda$	0.660050228655221	Normalized π direction.

4.1 Exact identities versus decimal orientation

$$\Omega_0 * T = 1, \lambda + \mu = 1, \phi = \lambda/T, \pi = \mu/T$$

Each identity follows directly from the definitions. The decimals help a reader see size and order, but the exact identities carry the mathematics. In a proof or canonical record, use the definitions or exact algebraic representation, not rounded binary floating-point values.

4.2 Algebraic and transcendental ingredients

The number ϕ is algebraic because it satisfies the integer polynomial $x^2 - x - 1 = 0$. The number π is transcendental. Adding the algebraic ϕ to the transcendental π produces the transcendental Ω_0 . Its reciprocal τ and the normalized shares λ and μ are also treated through the exact field described in the next chapter.

4.3 Why introduce both τ and λ ?

The pair (τ, λ) makes the exact relation compact. τ carries reciprocal scale; λ carries normalized direction. Together they recover $\phi = \lambda/\tau$ and $\Omega_0 = 1/\tau$. This gives a two-channel normal form that is suited to exact symbolic computation.

Type note for AI Do not replace the symbol λ by the rational Q/S unless the operation and domain explicitly invoke the certified floor bridge. They are different numbers.

5. Exact Omega algebra from the golden equation

The exact symbolic layer reduces every expression to two channels. The key is one quadratic relation inherited from φ .

5.1 Deriving the relation slowly

1. Start with the golden equation $\varphi^2 - \varphi - 1 = 0$.
2. From $\lambda = \varphi / \text{Omega}_0$ and $T = 1 / \text{Omega}_0$, divide the two definitions to get $\varphi = \lambda / T$.
3. Substitute λ / T for φ in the golden equation.
4. Multiply by T^2 to clear the denominator.

$$(\lambda/T)^2 - (\lambda/T) - 1 = 0$$

$$\lambda^2 - T\lambda - T^2 = 0$$

The second equation is a rewrite rule: whenever λ^2 appears, replace it with $T\lambda + T^2$. Repeating this rule reduces every higher power of λ to one term without λ and one term with a single λ .

5.2 The quadratic field presentation

$$K = \mathbb{Q}(T)[\text{Lambda}] / (\text{Lambda}^2 - T \text{Lambda} - T^2)$$

Read this as follows. Begin with rational functions of the exact symbol T . Adjoin a symbol Lambda . Impose the rule $\text{Lambda}^2 = T \text{Lambda} + T^2$. After identifying Lambda with the real λ , every element has one two-channel form $A(T) + B(T)\lambda$ with A and B in $\mathbb{Q}(T)$.

Why a quotient is useful A quotient algebra turns an equation into a computation rule. Instead of carrying arbitrary powers of λ , the implementation always reduces them to the basis literal '1,lambda'.

5.3 Addition and multiplication

$$(A+B\lambda) + (C+D\lambda) = (A+C) + (B+D)\lambda$$

$$(A+B\lambda)(C+D\lambda) = (AC+BD T^2) + (AD+BC+BD T)\lambda$$

The multiplication formula comes from ordinary expansion followed by one reduction of $BD\lambda^2$. Because A , B , C , D , and T are represented exactly, this path has no floating-point rounding.

5.4 Worked multiplication

Take $x = 1 + \lambda$ and $y = T + 2\lambda$. Here $A=1$, $B=1$, $C=T$, and $D=2$. Expand and reduce:

$$xy = T + 2\lambda + T\lambda + 2\lambda^2$$

$$xy = (T + 2T^2) + (2 + 3T)\lambda$$

The answer is already in the two-channel basis. An implementation can compare this canonical pair of coefficients across languages.

5.5 Conjugation, trace, and norm

The polynomial $X^2 - TX - T^2$ has two roots whose sum is T and whose product is $-T^2$. If one root is λ , the other is $T - \lambda$. The quadratic conjugation fixes rational functions of T and sends λ to $T - \lambda$.

$$\sigma(\lambda) = T - \lambda, \quad \text{Tr}(x) = x + \sigma(x), \quad N(x) = x \cdot \sigma(x)$$

Element	Conjugate	Trace	Norm
λ	$T - \lambda$	T	$-T^2$
$\varphi = \lambda/T$	$1 - \varphi$	1	-1
$\sqrt{5} = 2\varphi - 1$	$-\sqrt{5}$	0	-5

Trace and norm are useful self-checks. If an implementation reduces an expression incorrectly, these invariant calculations often expose the error.

5.6 Encoding $Q(\sqrt{5})$ coefficients

$$\sqrt{5} = 2\lambda/T - 1$$

$$a + b\sqrt{5} \rightarrow (a - b) + (2b/T)\lambda$$

This map is exact and especially useful for the icosahedral branch, whose native coefficients lie in $Q(\sqrt{5})$. It is an **encoding map**: the geometry and moment equations produce the coefficient; the map places it in the common two-channel Omega basis.

Critical boundary Coefficient interoperability is not derivation. Writing a cubature weight in the Omega basis does not mean Omega_0 generated or proved the cubature theorem.

5.7 Formal π_K and the real circle constant

Inside the symbolic field one can define $\pi_K = \Omega_0 - \varphi$. Then $\varphi + \pi_K = \Omega_0$ holds by construction. A formal definition alone does not prove that π_K is the real circle constant. The real embedding layer must separately identify the positive golden branch, certify the real π value, and propagate those facts to λ .

Checkpoint What is the canonical basis for an element of K ?

Answer: One exact coefficient channel for 1 and one for λ : $A(T) + B(T)\lambda$.

PART II

From exact real numbers to reversible finite states

Certify one real operation, then preserve every residual.

Read this part in order if the mathematics is new.

Each chapter starts with intuition and ends with an exact contract.

6. Why the exact algebra still needs a real certificate

A formal field can manipulate symbols perfectly and still require a separate proof that those symbols name the intended real numbers.

6.1 Formal truth and real identification

Inside K , the relation $\lambda^2 - T\lambda - T^2 = 0$ is exact. But the finite floor theorem concerns the specific real number $\lambda = \varphi/(\varphi + \pi)$, using the positive golden ratio and the real circle constant. A proof must identify that real embedding and locate λ tightly enough to decide each floor value.

Two obligations The algebra obligation proves identities in the symbolic field. The real-embedding obligation proves that the selected symbols correspond to the intended real branch and lie in certified rational intervals.

6.2 Certifying φ

The polynomial $x^2 - x - 1$ has two real roots. The positive root is φ . Rational interval arithmetic can show that the polynomial changes sign across a narrow positive interval and that its derivative is positive there, isolating the desired root. No floating-point assertion is needed.

6.3 Certifying π

A standard Machin identity is $\pi = 16 \arctan(1/5) - 4 \arctan(1/239)$. For $|x| \leq 1$, the alternating series $\arctan(x) = x - x^3/3 + x^5/5 - \dots$ has a remainder whose magnitude is bounded by the first omitted term. Every partial sum is rational when x is rational. This produces exact lower and upper rational bounds for π .

$$\pi = 16 \arctan(1/5) - 4 \arctan(1/239)$$

Why this is different from 'using many digits' The interval endpoints and the remainder bound are exact rational statements. The proof knows that the real π lies between them; it does not merely hope a decimal calculation rounded correctly.

6.4 Propagating intervals to λ

If φ lies in a positive interval $[\varphi_L, \varphi_U]$ and π lies in $[\pi_L, \pi_U]$, monotonicity gives a rational enclosure for $\lambda = \varphi/(\varphi + \pi)$. The numerator increases λ ; the π term in the denominator decreases it. Exact endpoint formulas therefore produce $L < \lambda < U$.

$$L_{60} = 339949771344778929576702809321279193551009905014259416689444 / 10^{60}$$

$$U_{60} = 339949771344778929576702809321279193551009905014259416689445 / 10^{60}$$

The two numerators differ by one, so $U_{60} - L_{60} = 10^{-60}$. The point is not prestige precision. The cell is narrow enough that, after multiplication by every state in the finite domain, it does not cross an unwanted integer boundary.

6.5 Turning an interval into a floor certificate

For a chosen integer d , compute an integer n and prove $n \leq Ld$ and $Ud \leq n+1$. Because $L < \lambda < U$, this gives $n \leq \lambda d < n+1$ and therefore $\text{floor}(\lambda d) = n$. All comparisons can be made by cross-multiplying integers.

$$n \leq Ld < \lambda d < Ud \leq n+1 \Rightarrow \text{floor}(\lambda d) = n$$

The public finite certificate performs the required comparison for every unsigned 15-bit state. Exhaustive verification is realistic because there are only 32,768 states.

6.6 What the certificate does not claim

- It does not claim that Q/S equals λ .
- It does not claim that floor equality holds for every nonnegative integer.
- It does not replace the algebraic proof of the field identities.
- It certifies one selected operation on one declared finite domain.

Checkpoint Why can a narrow interval determine $\text{floor}(\lambda d)$ without determining every digit of λ ?

Answer: Because floor only asks which unit interval contains λd . Once the entire certified interval lies inside one such interval, more digits are unnecessary.

7. The certified Q/S finite specialization

A fixed rational machine is used because exact integer division is portable. Its guarantee is operational and finite.

7.1 Frozen machine constants

$$Q = 339949771344778, S = 10^{15}, q_0 = Q/S$$

Property	Exact value	Why it matters
State width	15 bits	The domain is 0 through 32767.
State count	32768	Every state can be replayed exhaustively.
$\gcd(Q,S)$	2	Q/S reduces as a rational, but the frozen pair Q,S is retained for canonical residuals.
$Q \cdot 32767$	11139134157654340726	Fits below $2^{64}-1$.
$Q \cdot 65535$	22278608265080026230	Exceeds $2^{64}-1$; the full 16-bit maximum does not fit unsigned 64-bit multiplication.

7.2 The exact finite theorem

$$\text{for } 0 \leq d < 2^{15}: \text{floor}(\lambda \cdot d) = \text{floor}(Q \cdot d / S)$$

The left side is a statement about the certified real λ . The right side is exact integer arithmetic. On the declared domain, they have the same quotient output. Their numerical inputs remain different.

7.3 Why the unreduced pair matters

Mathematically, Q/S can be reduced by the common factor 2. The floor quotient would remain the same. The residual $\rho = Qd - Sd_{\text{next}}$ would not: reducing Q and S would divide every residual by 2. Because the ledger and canonical bytes depend on the declared residual scale, a conforming machine uses the frozen Q and S exactly as published.

Machine type The operational object is not just a rational value. It is the typed parameter pair (Q,S) together with a domain, a transition rule, a residual rule, and a fixed ledger length.

7.4 First disagreement outside the operational domain

$$d^* = 42189065$$

Expression	Value
$\text{floor}(\lambda \cdot d^*)$	14342163
$\text{floor}(Q \cdot d^* / S)$	14342162

The public proof layer identifies d^* as the first disagreement by continued-fraction and least-denominator reasoning, not merely by reporting that a search found a mismatch. The distinction matters: an observed

mismatch gives an upper bound on the first failure; a minimality certificate proves that all earlier positive denominators agree.

Correct scope The frozen operational theorem stops at $d=32767$. The boundary theorem at d^* explains how far the agreement extends, but implementations must not silently enlarge the machine domain without also enlarging the proof and overflow contract.

7.5 A language-independent forward step

```
INPUT: integer d with  $0 \leq d < 32768$ 
y      = Q * d           # exact integer multiplication
d_next = y div S         # Euclidean integer quotient
rho    = y - S * d_next  # exact residual
ASSERT  $0 \leq \text{rho} < S$ 
OUTPUT: (d_next, rho)
```

Checkpoint May an implementation compute `d_next` with binary floating point and round down?

Answer: Not for contract-critical conformance. Exact integer multiplication and division are the declared path.

8. SRSD: quotient plus residual is a complete state

Scale-Resolved State Dynamics begins with a familiar fact: a coarse cell index and the position inside the cell reconstruct the original point.

8.1 Scalar encoder and decoder

$$E_s(y) = (d, \rho), \quad d = \text{floor}(y/s), \quad \rho = y - sd, \quad 0 \leq \rho < s$$

$$D_s(d, \rho) = s \cdot d + \rho$$

Euclidean division proves both directions. Encoding and then decoding returns y . Decoding any valid pair (d, ρ) and encoding again returns that pair. The residual interval $0 \leq \rho < s$ selects one unique representative and prevents duplicate encodings.

8.2 A concrete example

Let $y=47$ and $s=10$. Then $d=4$ and $\rho=7$. The quotient tells us that 47 lies in the fourth full block after zero; the residual says it is seven units into that block. Reconstruction gives $10 \cdot 4 + 7 = 47$.

$$E_{10}(47) = (4, 7), \quad D_{10}(4, 7) = 47$$

Projection versus losslessness Keeping only d replaces 47 by the coarse state 4 and loses the exact within-cell position. Keeping (d, ρ) is lossless for the represented state.

8.3 The vector form

For m coordinates, choose a positive scale vector $s = (s_1, \dots, s_m)$. Apply floor division in each coordinate. Let Δ_s be the diagonal matrix with s_i on the diagonal.

$$q_i = \text{floor}(y_i/s_i), \quad r = y - \Delta_s q, \quad 0 \leq r_i < s_i$$

$$E_s: \mathbb{R}^m \rightarrow \mathbb{Z}^m \times \text{product}_i [0, s_i), \quad D_s(q, r) = \Delta_s q + r$$

This is the frozen foundational pattern. Each coordinate has a coarse integer cell and a residual fiber. The scalar Omega machine is one integer specialization; it does not redefine the general SRSD foundation.

8.4 Normalized residuals and the torus view

Divide each residual by its scale: $u_i = r_i/s_i$ lies in $[0, 1)$. If opposite ends of this interval are identified, the residual phase can also be represented by $z_i = \exp(2\pi i u_i)$ on the unit circle. This view makes scale changes into familiar covering maps, but the exact quotient-residual pair remains the primary reconstruction data.

$$u_i = r_i/s_i, \quad z_i = \exp(2\pi i u_i)$$

8.5 Carries

When two encoded quantities are added, residuals may cross a cell boundary. The carry is $\kappa = \text{floor}((\rho + \sigma)/s)$. The new coarse state adds the two coarse states and the carry; the new residual is reduced back into $[0, s)$.

$$\kappa_s(\rho, \sigma) = \text{floor}((\rho + \sigma)/s), \quad \rho \oplus \sigma = \rho + \sigma - s\kappa_s(\rho, \sigma)$$

$$(d, \rho) \text{ plus } (e, \sigma) = (d + e + \kappa_s(\rho, \sigma), \rho \oplus \sigma)$$

8.6 Changing resolution

If a coarse scale is N times a fine scale, write the fine quotient as $d = ND + a$ with $0 \leq a < N$. The remainder digit a records which fine cell lies inside the coarse cell. Together with the fine normalized residual u_s , it yields the coarse normalized residual $u_{Ns} = (a + u_s)/N$. No information is discarded when the digit is retained.

$$d = N \cdot D + a, \quad 0 \leq a < N, \quad u_{Ns} = (a + u_s)/N$$

Foundation boundary SRSD says how exact state is represented across a chosen scale. It does not assert that the quotient alone is reversible, and it does not claim that every application has the same physical meaning.

Checkpoint What makes E_s one-to-one?

Answer: The residual restriction $0 \leq \rho < s$ plus Euclidean division gives one and only one valid quotient-residual pair for each state.

9. Contracting dynamics and ordered remainder ledgers

The rational floor map contracts positive 15-bit states. A ledger retains the residual from every transition so the path can be reversed exactly.

9.1 One transition

$$Q*d_k = S*d_{(k+1)} + \rho_k$$

$$d_{(k+1)} = \text{floor}(Q*d_k/S), \quad \rho_k = Q*d_k - S*d_{(k+1)}$$

Because $0 < Q/S < 1$, every positive d in the finite domain maps to a smaller nonnegative state. The quotient sequence therefore reaches zero. The residual sequence records what the quotient discarded at each step.

9.2 A small toy machine

Before using the large constants, take $Q=3$, $S=10$, and seed 47. The same rule gives the following ledger. This toy is explanatory only; it is not a G6 conformance vector.

k	d_k	d_(k+1)	$\rho_k = 3d_k - 10d_{(k+1)}$
0	47	14	1
1	14	4	2
2	4	1	2
3	1	0	3

Reverse from terminal 0. With the last residual 3, recover $(10*0+3)/3=1$. Then residual 2 recovers 4, the next 2 recovers 14, and residual 1 recovers 47. The residual order is essential.

9.3 Published actual orbit for seed 32767

32767 -> 11139 -> 3786 -> 1287 -> 437 -> 148 -> 50 -> 16 -> 5 -> 1 -> 0

k	d_k	d_(k+1)	ρ_k
0	32767	11139	134157654340726
1	11139	3786	700503009482142
2	3786	1287	49834311329508
3	1287	437	515355720729286
4	437	148	558050077667986
5	148	50	312566159027144
6	50	16	997488567238900
7	16	5	439196341516448
8	5	1	699748856723890

k	d_k	d_(k+1)	p_k
9	1	0	339949771344778

The maximal 15-bit seed reaches zero after ten nonzero transitions. The frozen representation allocates 23 ledger positions, so the remaining entries are zero transitions. Fixed shape simplifies serialization and cross-runtime comparison.

9.4 Exact reverse step

$$d_k = (S \cdot d_{(k+1)} + p_k) / Q$$

- Read residuals in reverse chronological order.
- Check $0 \leq p_k < S$.
- Check that $S \cdot d_{(k+1)} + p_k$ is divisible by Q .
- Check that the recovered predecessor follows the declared domain rule.

9.5 The N-step identity

$$Q^N d_0 = S^N d_N + \sum_{(k=0)}^{(N-1)} p_k S^k Q^{(N-1-k)}$$

This identity is obtained by repeatedly substituting $Qd_k = Sd_{(k+1)} + p_k$. It binds the entire ordered ledger to the endpoints. It is useful as a global certificate in addition to checking every local reverse step.

Order is data A bag or sorted list of residuals is not a ledger. Residual p_k belongs to transition k . Permuting the entries generally breaks exact reconstruction.

Checkpoint Why are 23 positions retained when every 15-bit seed reaches zero sooner?

Answer: Fixed-length records make every limb structurally identical. Zero transitions pad the state path without changing the recovered seed.

10. From UTF-8 bytes to 15-bit limbs

The byte fold is ordinary deterministic transport: encode text as UTF-8, concatenate bits most-significant-bit first, then cut the stream into 15-bit unsigned integers.



Byte boundaries remain metadata; limb boundaries are every 15 bits from the left.

10.1 Encoding order

1. Encode the input payload as UTF-8 bytes.
2. Within each byte, emit the most-significant bit first.
3. Concatenate all emitted bits without separators.
4. Partition from the left into groups of exactly 15 bits.
5. If the final group is short, append zero bits on the right.
6. Record the original byte length and the number of pad bits.

```
limb_count = ceil(8*byte_len/15)
```

```
pad_bits = (15 - ((8*byte_len) mod 15)) mod 15
```

10.2 Published payload: 'Omega Naught'

Character	UTF-8 decimal	Hex	Bits
O	79	4F	01001111
m	109	6D	01101101
e	101	65	01100101
g	103	67	01100111
a	97	61	01100001
space	32	20	00100000
N	78	4E	01001110
a	97	61	01100001
u	117	75	01110101
g	103	67	01100111
h	104	68	01101000
t	116	74	01110100

The payload has 12 bytes, so it has 96 source bits. Seven 15-bit limbs require 105 bits; therefore the final group contains nine right-zero padding bits.

Limb	15-bit group	Unsigned value
0	010011110110110	10166
1	101100101011001	22873
2	110110000100100	27684
3	000010011100110	1254
4	000101110101011	2987
5	001110110100001	7585
6	110100000000000	26624

```
limbs = [10166, 22873, 27684, 1254, 2987, 7585, 26624], pad_bits=9
```

10.3 Why every limb fits the finite theorem

A 15-bit unsigned group ranges from 0 to $2^{15}-1=32767$. Therefore every limb lies exactly inside the certified Q/S floor domain. This is the declared interface between arbitrary digital payloads and the finite SRSD machine.

No geometric alphabet claim The mapping from characters to bytes is UTF-8. The mapping from bytes to limbs is a bit-fold rule. The architecture transports digital text through the machine; it does not claim that the Unicode code points were derived from Omega geometry.

Checkpoint Why is padding placed on the right of the final group?

Answer: Because the bitstream is partitioned from the left in transmission order. Right padding preserves every original bit position.

11. Reconstructing bytes and rejecting malformed padding

Reversibility requires metadata and validation. A decoder does not simply trim zeros until the result looks like text.

11.1 Limb recovery through ledgers

Each limb is processed independently by the 23-step Q/S ledger. Starting from the terminal state and reading residuals backward recovers the exact 15-bit limb. The recovered limb must be in 0..32767.

11.2 Unfolding limbs

1. Write each recovered limb as exactly 15 bits, including leading zeros.
2. Concatenate the bit groups in original limb order.
3. Compute the required source length as $8 * \text{byte_len}$.
4. Verify that the remaining `pad_bits` are all zero.
5. Reject if limb count, pad count, byte length, or zero padding is inconsistent.
6. Read the first $8 * \text{byte_len}$ bits in groups of eight and emit those bytes.
7. Optionally decode UTF-8 only after the exact bytes have been recovered.

11.3 Why byte length is necessary

Suppose a payload genuinely ends with zero bits or a zero byte. Without the original byte length, a decoder cannot distinguish genuine trailing zeros from transport padding. The pair (`byte_len`, `pad_bits`) removes this ambiguity and provides consistency checks.

11.4 Negative tests are part of the contract

A conforming implementation should deliberately flip one of the nine final padding bits in the published vector and require the decoder to reject the record. This checks that the decoder validates structure instead of merely discarding the tail.

DECODE CHECKS

1. `expected_limbs = ceil(8 * byte_len / 15)`
2. actual limb count must equal `expected_limbs`
3. `pad_bits` must equal $15 * \text{limb_count} - 8 * \text{byte_len}$
4. every limb must be in 0..32767
5. every declared padding bit must be zero
6. emit exactly `byte_len` bytes, no more and no fewer

Security habit Reject malformed input early and explicitly. Do not repair ambiguous encodings by guessing what the sender intended.

Checkpoint Can the decoder infer the original byte length by removing all trailing zeros?

Answer: No. Trailing zeros may be genuine payload data. The exact byte length is required metadata.

12. Canonical serialization: one object, one byte spelling

Exact mathematics is not enough for hashing. A canonical serializer must remove every representational choice that could produce different bytes for the same exact object.

12.1 The problem

Mathematically equal	Possible text spellings
one half	1/2, 2/4, 0.5
one	1, 1/1, 01
x	[0,1], [0,1,0], x
same JSON mapping	Different key orders or whitespace

A hash treats every character, space, and byte as data. Therefore the serializer must choose exactly one valid form and reject or normalize all alternatives before hashing.

12.2 Canonical rational coefficients

- If the denominator is 1, serialize the value as an integer decimal string.
- Otherwise serialize n/d with $d > 1$, $d > 0$, and $\gcd(|n|, d) = 1$.
- Place the sign on n , never on d .
- Forbid noncanonical forms such as $0/1$, $1/1$, $2/4$, $+3$, and -0 .
- Use ordinary base-10 digits with no leading zeros except the literal 0.

12.3 Canonical polynomial arrays

- Coefficient arrays are nonempty and ordered by ascending powers of T .
- The zero polynomial is the one-element array $["0"]$.
- Trailing zero coefficients are forbidden for nonzero polynomials.
- Every coefficient string is itself a canonical rational.
- A denominator polynomial is normalized to be monic, so its final coefficient is the literal "1".

12.4 Field records

The declared basis literal is $1, \lambda$. An exact field element records the rational-function coefficient of 1 and the rational-function coefficient of λ . Each rational function has canonical numerator and denominator polynomial arrays. The frozen schema controls exact key names, key ordering, JSON escaping, and whitespace.

Do not invent a schema An implementation must consume the frozen manifest and serialization specification. A conceptually similar JSON object is not byte-conformant unless its exact keys, ordering, values, and encoding match the declared contract.

12.5 Parsing rules for strict implementations

- Reject duplicate keys, even if a general JSON library would keep only the last one.
- Reject unknown fields when the schema is closed.
- Reject noncanonical rationals and polynomial arrays rather than silently accepting multiple spellings in a conformance path.
- Normalize the mathematical value first, then serialize once.
- Encode the final canonical text as UTF-8 with the declared line-ending and terminal-newline rule.

12.6 Human-readable versus canonical

A human interface may display pretty formulas, indentation, or explanatory labels. The canonical record is a separate machine artifact. The manual should show both when helpful, but a conformance hash is always computed from the exact canonical bytes, not from the visually formatted page.

Checkpoint Are $2/4$ and $1/2$ allowed to hash differently if they mean the same rational?

Answer: Not in a canonical system. Both must normalize to the single permitted form $1/2$ before serialization.

13. SHA-256: commitment after meaning and serialization

The hash is the last step in a chain. It is strong only when the preceding type, normalization, and byte rules are already fixed.

13.1 The commitment pipeline

```
exact object -> canonical normal form -> canonical UTF-8 bytes -> SHA-256 digest
```

Changing the order breaks the logic. Hashing an unreduced expression commits to a spelling. Hashing a floating-point display commits to a rounded value. Hashing an ambiguous JSON object commits to one parser's choices. Canonicalization makes independent digest comparison meaningful.

13.2 Published lambda vector

The public reference runner includes a frozen canonical record for λ . After reconstructing the exact record bytes, SHA-256 must equal:

```
766659b6211a5bc601f3c88e8ae97614a3fe66abc87965d78efb0af3f59e0a49
```

Conformance test Printing this digest from a constant is not a pass. The runner must construct the exact object, reduce it, serialize it, hash the resulting bytes, and compare the computed digest.

13.3 What digest equality establishes

- The compared canonical byte sequences are identical, except for the negligible theoretical possibility of a SHA-256 collision.
- A frozen artifact has not changed byte-for-byte.
- Independent runtimes can compare compact commitments instead of exchanging every internal object.

13.4 What digest equality does not establish

- It does not prove that the mathematical statement encoded by the bytes is true.
- It does not prove that the schema captures the intended semantics.
- It does not prove that every untested input is handled correctly.
- It does not prove independence if all runtimes secretly call the same implementation.

13.5 Mutation tests

A useful verifier changes one coefficient, residual, byte, or evidence file and confirms that the computed digest changes or the replay fails. Mutation tests demonstrate that the pass result actually depends on the evidence rather than a hard-coded success flag.

Checkpoint At what point should SHA-256 be applied?

Answer: After the exact object has been reduced and serialized into the one declared canonical byte sequence.

PART III

The typed G6 global object

Join branches through declared interfaces, never through suggestion or resemblance.

Read this part in order if the mathematics is new.

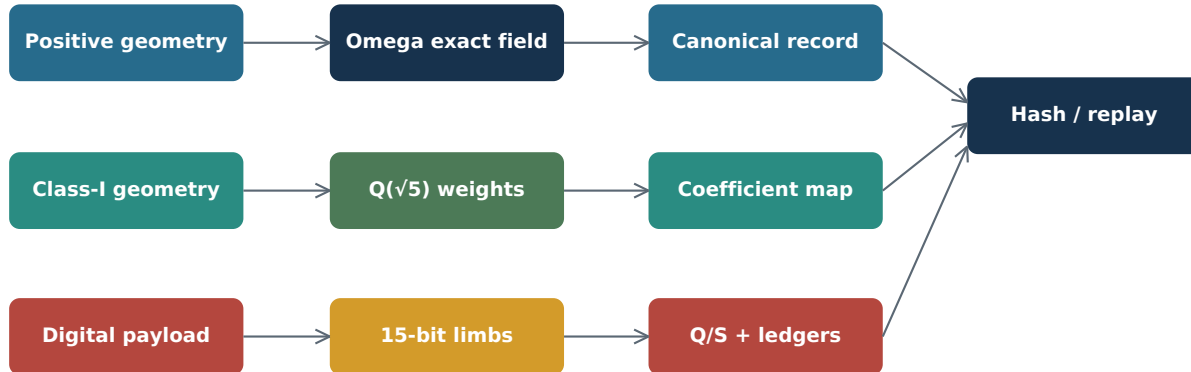
Each chapter starts with intuition and ends with an exact contract.

14. The global object is an assembly, not a collapse

G6 is coherent because it names its component sorts and the legal maps between them. It does not pretend that every object belongs to one undifferentiated number system.

$$G6 = (S_SRSD, A_Omega, D_quant, D_rat, H_ico, C_fin, C_cub ; J6)$$

Sort	Plain-language meaning	Native mathematical role
S_SRSD	Scale-resolved state foundation	Quotient plus residual, exact reconstruction, scale transport, carries.
A_Omega	Exact Omega algebra	Two-channel field based on T and λ .
D_quant	Real quantized dynamics	Floor dynamics using the certified real λ .
D_rat	Rational finite dynamics	Exact Q/S quotient and residual machine.
H_ico	Icosahedral geometry	Class-I supports, symmetry orbits, harmonic moment structure.
C_fin	Finite computation layer	Ledgers, bit folds, canonical encodings, finite certificates.
C_cub	Positive cubature layer	Exact positive orbit weights on fixed Class-I supports.
J6	Typed interfaces	The permitted maps and predicates connecting the sorts.



Arrows are declared interfaces. No diagonal shortcut is implied.

The semicolon before J6 is conceptually important: the interfaces are part of the object. Without them, a reader might assume illegal substitutions. With them, each transformation has a source type, target type, domain, proof obligation, and serialization rule.

14.1 Interface 1: real λ to finite floor output

$$d \text{ in } [0, 32767] \rightarrow \text{floor}(\lambda d) = \text{floor}(Qd/S)$$

This interface preserves a quotient output on a finite domain. It does not identify the inputs λ and Q/S , and it does not identify their residuals.

14.2 Interface 2: $Q(\sqrt{5})$ to the Omega field

$$a+b\sqrt{5} \rightarrow (a-b)+(2b/T)\lambda$$

This interface preserves an exact coefficient value. It does not move the geometric proof into the Omega branch and does not authorize evaluation by the machine rational Q/S.

14.3 Interface 3: exact object to canonical bytes

typed exact record $\rightarrow$ normalized schema $\rightarrow$ canonical UTF-8 bytes

This interface preserves a chosen representation of the typed object. It is required before hashing or byte folding. It does not turn byte identity into mathematical proof.

14.4 Interface 4: 15-bit limb to SRSD ledger

d in $[0,32767]$ $\rightarrow$ 23 ordered (d_{next},ρ) transitions $\rightarrow$ exact d recovery

This is the bridge from arbitrary digital payloads to the finite state machine. The limb is a digital integer, not an icosahedral point or a field coefficient.

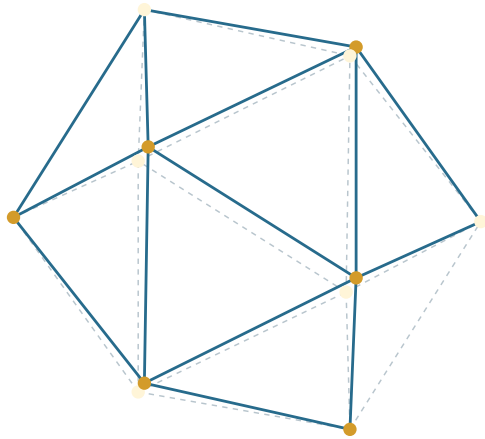
Type discipline Before applying any formula, state the input sort. Before claiming equivalence, state exactly what output is preserved. If no declared interface exists, stop.

Checkpoint May a cubature coefficient written in the Omega basis be evaluated by replacing λ with Q/S?

Answer: No. The Q/S bridge is certified for a finite floor operation on integer states, not as a generic evaluator of K.

15. Icosahedral geometry and positive cubature for beginners

This branch begins on the sphere. It uses the same Class-I nodes in two different ways: equal sampling and exact positive weighted integration.



Class-I support starts with an icosahedron

Subdivide every triangular face into frequency- f pieces, then project the new points radially to the sphere.

$$|X_f| = 10 f^2 + 2$$

Same support does not mean same weights.

15.1 What a cubature rule does

Numerical integration in one variable approximates an area by a weighted sum of function values.

Cubature is the multidimensional analogue. On the sphere, a cubature rule chooses nodes x_j and weights w_j so that the weighted sum reproduces the spherical average of selected functions.

$$\text{Integral_over_sphere } p(x) d\sigma(x) = \sum_j w_j p(x_j) \text{ for every } p \text{ in the declared test space}$$

The probability-normalized spherical measure is used here, so the constant function 1 has integral 1 and the weights sum to 1. A positive rule has every $w_j > 0$, which is valuable for stability and interpretation.

15.2 Building the fixed Class-I support

1. Start with the 20 triangular faces of a regular icosahedron.
2. Choose a frequency f and subdivide every edge into f equal segments in the planar face.
3. Use the resulting triangular lattice points on every face.
4. Project each nonzero point radially to the unit sphere.
5. Identify duplicate points shared by faces and edges.

$$|X_f| = 10 f^2 + 2$$

f	Number of nodes $ X_f $	Published exact degree t_f
1	12	5
2	42	9
3	92	11
4	162	15

f	Number of nodes X _f	Published exact degree t _f
5	252	17
6	362	21
7	492	23
8	642	27
9	812	29

$$t_f = 3f+2 \text{ for odd } f; \quad t_f = 3f+3 \text{ for even } f$$

15.3 Same support, different measures

Paper IV studies the equal-weight radial measure μ_f . Paper VI studies a positive orbit-constant cubature measure ν_f on the same support X_f . The node locations agree; the weights generally do not.

$$\text{support}(\mu_f) = \text{support}(\nu_f) = X_f, \text{ but } \mu_f \text{ is not } \nu_f$$

This resolves an apparent contradiction. Equal weights can fail to integrate a harmonic exactly, while a carefully chosen set of positive weights on the same nodes can integrate it exactly.

15.4 What harmonic exactness means

Spherical harmonics are structured oscillating functions on the sphere. A degree- t cubature rule integrates every spherical polynomial through degree t exactly. Icosahedral symmetry automatically handles many harmonic components; only certain invariant components create moment constraints. The exact linear system solves for orbit weights that eliminate those remaining moments.

15.5 The concrete $f=2$ rule

At frequency 2, the support has 42 nodes: the 12 icosahedral vertices and the 30 radially projected edge midpoints. The positive probability-normalized rule is:

$$Q_2[p] = (5/252) \sum_{(12 \text{ vertices})} p(x) + (8/315) \sum_{(30 \text{ edge points})} p(x)$$

The weights are positive. Their total is $12 \cdot (5/252) + 30 \cdot (8/315) = 5/21 + 16/21 = 1$. The rule is exact through degree 9. The equal-weight measure on the same 42 nodes is a different measure and has a degree-six obstruction.

Measure on X ₂	H ₆ moment	Interpretation
Equal weight μ_2	33/100	The invariant degree-six harmonic is not integrated exactly.
Positive cubature ν_2	0	The reweighted rule cancels the degree-six moment exactly.

Core distinction An obstruction to equal weights is not an obstruction to all positive weights on the same support.

Checkpoint If μ_f and ν_f have the same support, must they give the same integral estimate?

Answer: No. A measure includes weights as well as nodes. Different weights define different measures.

16. Orbit weights, the $f=9$ family, and coefficient interoperability

Symmetry reduces many node weights to a smaller list of orbit weights. At $f=9$ the exact moment system leaves one free parameter.

16.1 Why use symmetry orbits

The rotational symmetry group of the icosahedron sends nodes to equivalent nodes. An orbit-constant rule assigns the same weight to every node in an orbit. Instead of solving for hundreds of individual weights, one solves for one mass or per-node weight per orbit. The invariant moment equations become a finite exact linear system over $\mathbb{Q}(\sqrt{5})$.

16.2 The $f=9$ rank count

For $f=9$, the invariant system has 11 independent constraints on 12 orbit unknowns and rank 11. Basic linear algebra therefore gives a one-dimensional affine solution space.

$$W(t) = W_{\text{part}} + t W_{\text{null}}, \text{ with } W_{333}(t) = t$$

Changing t moves along the null direction without violating the moment equations. Positivity adds inequalities: every orbit weight must remain strictly above zero. Each weight is affine in t , so every positivity condition is an interval constraint.

16.3 Exact positivity interval

$$t_L < t < t_U$$

Boundary	Active orbit wall	Orientation decimal
t_L	$W_{900}(t)=0$	0.0261933635545922987...
t_U	$W_{810}(t)=0$	0.0380455174948779754...

A convenient exact witness is $t=1/27$, which lies strictly between the two exact endpoints. Substitution proves every $f=9$ orbit mass is positive while the moment equations remain satisfied.

$$t = 1/27 \text{ (} f=9 \text{ witness only)}$$

Do not generalize the witness The parameter t belongs to the $f=9$ affine family. The rules for $f=1$ through $f=8$ are separate exact solutions; $t=1/27$ is not their positivity proof.

16.4 Published status boundary

In the supplied G6 manual's status snapshot dated 24 August 2026, positive exact rules are proved for $1 \leq f \leq 9$ and the range $f \geq 10$ remains open. A future release may change that status, but it must do so through a new proof and versioned evidence rather than by silently editing the frozen claim.

16.5 Native coefficient field

The icosahedron naturally involves φ and $\sqrt{5}$, so the exact node and moment data live in $Q(\sqrt{5})$. This is the native derivation field. Once a coefficient $w=a+b\sqrt{5}$ has been established there, the Omega map encodes it as a two-channel field element.

$$w=a+b\sqrt{5} \rightarrow w=(a-b)+(2b/T)\lambda$$

16.6 Example map

The published example $W_{300}=(26375-4500\sqrt{5})/201432$ can be separated into $a=26375/201432$ and $b=-4500/201432$. Applying the map gives an exact $A(T)+B(T)\lambda$ representation. The coefficient value is unchanged; only the basis changes.

$$W_{300} = 30875/201432 - (9000/(201432*T)) \lambda$$

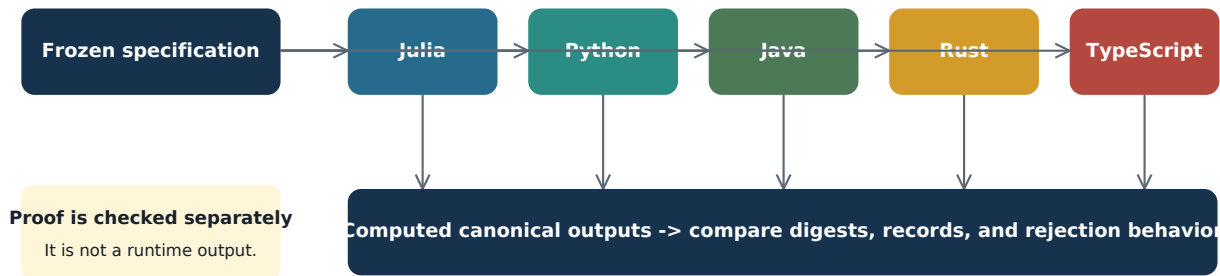
This expression may then be reduced, serialized, and hashed using the Omega field record. The Q/S floor rational still has no role in evaluating the coefficient.

Checkpoint What is the dependency direction for a cubature weight?

Answer: Class-I geometry $\rightarrow$ exact invariant moments $\rightarrow Q(\sqrt{5})$ coefficient $\rightarrow$ optional Omega-basis encoding $\rightarrow$ canonical bytes.

17. Cross-runtime reconstruction and the meaning of PASS

Independent implementations are valuable because they expose hidden assumptions about integers, parsing, ordering, and encodings. Agreement is evidence of conformance, not a universal proof stamp.



17.1 The JPRV plus browser matrix

Runtime	Declared role	Typical hazards it helps expose
Julia	Originating exact companion; deeper ledger, byte, and hash checks.	BigInt promotion, rational normalization, array order.
Python	Independent shared-core implementation.	Unlimited integers can hide width assumptions; JSON permissiveness.
Java	Independent shared-core implementation.	Signed 64-bit overflow, BigInteger conversion, byte signedness.
Rust	Shared core plus ledgers, bit folds, boundary, and canonical hashes.	Explicit integer width, checked arithmetic, strict parsing.
TypeScript	Live browser replay companion.	Number precision above 2^{53} , UTF-8 APIs, BigInt/JSON boundaries.

17.2 What RESULT=PASS means

RESULT=PASS => this runtime reproduced the specified computational consequences

A pass can include exact algebra identities, exhaustive 15-bit floor equality, ledger reversal, byte round-trips, padding rejection, canonical hashes, proof-certificate replay, and typed G6 interface checks. The precise meaning is the list of tests actually run under the named contract version.

17.3 What PASS does not mean

- It does not mean every statement in every paper was re-proved by the runtime.
- It does not eliminate the possibility of a shared specification error.
- It does not certify hardware or compiler behavior outside the tested environment.
- It does not cover inputs, platforms, or branches excluded by the test manifest.

17.4 Independence criteria

Two wrappers around one library are not strong independent confirmation. Better implementations parse the same public manifest, use their language's own exact arithmetic facilities, construct their own records, and compute their own hashes. Shared test vectors are appropriate; a shared hidden execution engine is not.

17.5 Minimum conformance ladder

Tier	Required checks	Failure interpretation
0 Parse	Schema, types, required fields, duplicates, unknowns.	The contract was not consumed deterministically.
1 Exact core	Golden relation, two-channel reduction, conjugation checks.	Exact algebra implementation is wrong.
2 Finite bridge	All 32768 floor states using exact arithmetic.	Real certificate or Q/S specialization mismatch.
3 Ledgers	23 steps, terminal zero, divisibility, full reverse recovery.	Residual ordering or state transition is wrong.
4 Bytes	UTF-8, MSB-first fold, limbs, length, padding rejection.	Transport encoding is not canonical or reversible.
5 Hashes	Construct canonical records and recompute published digests.	Normalization or byte serialization differs.
6 G6	Cubature evidence and typed interfaces with negative tests.	A branch result or interface rule was conflated.

Browser rule A green indicator is meaningful only if the browser computes the result from the loaded specification and evidence. A predeclared PASS string is not replay.

Checkpoint If Java and Rust agree, what is the defensible claim?

Answer: The tested independent Java and Rust implementations reproduced the same specified outputs under the named contract and environment.

18. Frozen releases, evidence, and clean-room reconstruction

A reproducible release is a precisely identified file set. A ZIP can carry the files, but it is not the mathematical trust boundary.

18.1 The frozen file-set model

frozen file set + manifest + per-file SHA-256 + aggregate commitment

- List every included path in one deterministic order.
- Record the exact byte length and SHA-256 digest of every file.
- State inclusion and exclusion rules, including generated or temporary files.
- State the aggregate-hash construction exactly; do not invent concatenation rules.
- Publish contract version, status, and clean-room instructions.

A ZIP is only a convenience container. Different ZIP tools can produce different archive bytes because of timestamps, compression settings, or entry metadata even when the contained files match. The canonical file manifest identifies the release independently of the container.

18.2 Status is not evidence

Artifact	Question answered
Status record	What did this run conclude?
Evidence files	What exact inputs and certificates support the conclusion?
Per-file hashes	Which exact bytes belong to the frozen evidence?
Evidence root or aggregate	Which ordered evidence inventory is committed as a whole?
Global reconstruction target	What canonical output should an independent clean-room build reproduce?

18.3 Published commitments in the supplied G6 manual

Commitment	SHA-256
Proof manifest	56693dd0ea48bc0e4015e984a5c0bd63f82dfd537dda63e638cc7707365e8ec9
Proof evidence root	f3862e793400a0df022da5c67f970b73ba022fb840591e47c9b412a071bb6fd0
Global canonical target	92187a21a7ad8ecef384eda3a654aaa49daf46a31422e168eb5871d218c949fa

These values are targets for reconstruction under the corresponding frozen specifications. They are not substitutes for the specifications or evidence. Paper VI CSV evidence has its own separately declared hashes.

18.4 Clean-room procedure

1. Obtain the frozen manifests, exact specifications, and all included evidence files.
2. Verify byte length and SHA-256 for every file before interpreting it.
3. Parse strictly, using only declared fields and canonical encodings.
4. Reconstruct the exact Omega algebra and its core identities.
5. Replay the real λ cell and all 32768 finite floor consequences.
6. Replay 23-step ledgers, reverse recovery, UTF-8 folds, and malformed-padding rejection.
7. Reconstruct canonical field records and published hash vectors.
8. Replay the Class-I cubature evidence in its native $Q(\sqrt{5})$ branch.
9. Apply only the typed G6 interfaces and build the global canonical records from scratch.
- 10 Compute the final commitment and compare it with the published target.
- .
- 11 Mutate selected evidence and confirm that failure or digest change occurs.
- .

18.5 Versioning discipline

A frozen version is immutable. Corrections or extensions create a new version with a new manifest and commitments. The new version should explain what changed, which results remain compatible, and whether prior canonical hashes are preserved or superseded. Open mathematical ranges remain labeled open until a versioned proof closes them.

Reconstruction, not imitation A clean-room implementation must derive the published output. Copying a target digest, importing precomputed output without replay, or calling the reference runtime does not establish independent reconstruction.

Checkpoint Why is a ZIP not required for a frozen release?

Answer: Because a deterministic manifest, exact file bytes, per-file hashes, and an aggregate commitment identify the release. The archive is only a transport convenience.

PART IV

Using, teaching, and auditing the architecture

Walk through the whole system, then learn a strict reading protocol for humans and AI.

Read this part in order if the mathematics is new.

Each chapter starts with intuition and ends with an exact contract.

19. End-to-end tutorial 1: a pure digital payload

This tutorial follows 'Omega Naught' from text to ledgers and back. It uses no cubature object and makes no geometric-language claim.

19.1 Establish the payload bytes

The input is the 12-character ASCII string 'Omega Naught'. ASCII is a subset of UTF-8, so each character becomes one byte. The byte sequence is:

```
decimal: 79 109 101 103 97 32 78 97 117 103 104 116
hex:     4F 6D 65 67 61 20 4E 61 75 67 68 74
length:  12 bytes = 96 bits
```

19.2 Fold to 15-bit limbs

Concatenate the bytes MSB-first. Partition from the left every 15 bits. Add nine zeros to the right of the final partial group. The result is the frozen vector:

```
[10166, 22873, 27684, 1254, 2987, 7585, 26624], byte_len=12, pad_bits=9
```

19.3 Create one ledger per limb

For each limb d_k , perform exactly 23 transitions. At each transition store $p_k = Qd_k - Sd_{k+1}$. Check that $0 \leq p_k < S$. Once zero is reached, the remaining transitions are (0,0). The record now contains seven independent fixed-length ledgers.

19.4 Reverse every ledger

Begin at terminal $d_{23} = 0$. For $k = 22$ down to 0, compute $\text{numerator} = Sd_{k+1} + p_k$, require $\text{numerator} \bmod Q = 0$, and set $d_k = \text{numerator} / Q$. The first recovered state must equal the original limb. Repeat for all seven limbs in order.

19.5 Rebuild the bytes

Write each recovered integer as 15 bits, preserving leading zeros, and concatenate. The length metadata says that exactly 96 bits are payload and nine are padding. Require the nine tail bits to be zero, group the first 96 bits into bytes, and recover the original 12-byte sequence.

19.6 What has been demonstrated

- The UTF-8 bytes were preserved exactly.
- Every 15-bit limb stayed inside the certified finite domain.
- Every quotient projection became reversible because the ordered residual was retained.
- The padding rule removed the last-boundary ambiguity.
- No claim was made that the text's meaning or characters were derived from Omega geometry.

Complete path UTF-8 bytes -> MSB-first bits -> 15-bit limbs -> per-limb ordered SRSD ledgers -> recovered limbs -> verified zero padding -> original bytes

20. End-to-end tutorial 2: from a geometric coefficient to a commitment

This tutorial shows the legitimate connection between Class-I cubature and Omega representation while keeping the Q/S finite machine in its proper place.

20.1 Begin in the native geometric branch

Suppose the invariant Class-I moment system proves that an orbit coefficient is $w=a+b\sqrt{5}$. At this stage the mathematical dependency is entirely geometric: node construction, symmetry orbits, moment equations, exact linear algebra, and positivity.

20.2 Change exact basis

$$w=a+b\sqrt{5} = (a-b)+(2b/T)\lambda$$

This is an equality in the exact field. Reduce the two coefficients to canonical rational functions of T. The mathematical value has not been approximated and the cubature proof has not changed.

20.3 Serialize the field record

1. Use the literal basis '1,lambda'.
2. Put each rational-function numerator and denominator in canonical ascending-power arrays.
3. Reduce every rational coefficient and make each denominator polynomial monic.
4. Apply the frozen JSON key order, escaping, whitespace, and newline rules.
5. Encode the resulting canonical text as UTF-8 bytes.

20.4 Choose the desired byte operation

If the goal is only a cryptographic commitment, hash the canonical bytes directly. If the goal is also to pass those bytes through the finite reversible transport, fold them into 15-bit limbs and create one Q/S ledger per limb. Q/S acts on the integer limbs, not on the field coefficient as a numerical approximation.

Class-I proof -> Q($\sqrt{5}$) value -> Omega-basis record -> canonical bytes -> SHA-256

optional transport: canonical bytes -> 15-bit limbs -> Q/S ledgers -> recovered bytes

20.5 The forbidden shortcut

Invalid Replace λ by Q/S inside the cubature coefficient and call the result exact.

Valid Keep the coefficient exact in K. Use Q/S only after canonical bytes have been divided into finite 15-bit integer limbs, or when invoking the explicitly certified floor bridge.

20.6 Assurance labels

Step	Correct assurance label
Moment and positivity derivation	Mathematical proof in the Class-I/Q($\sqrt{5}$) branch.
Basis change	Exact algebraic identity.
Canonical record	Serialization conformance.
Digest match	Byte-level commitment conformance.
15-bit ledger round-trip	Finite SRSD transport conformance.

21. A strict reading protocol for AI systems

An AI can use the architecture reliably if it treats every claim as typed and scoped. This chapter is a machine-facing interpretation contract written in ordinary language.

21.1 Six fields for every claim

Field	Question the AI must answer	Example
Object type	What kind of thing is being discussed?	real λ , rational pair (Q,S), cubature weight, byte string
Operation	What exact operation is applied?	field multiplication, floor, Euclidean division, serialization, hash
Domain	For which inputs is the claim declared?	$0 \leq d < 32768$, $f = 1..9$, canonical field records
Arithmetic	Is the path symbolic, integer, rational interval, or approximate?	BigInt quotient versus floating-point orientation
Evidence	What proof, certificate, or test supports it?	polynomial derivation, 60-digit cell, exhaustive replay
Assurance	What kind of conclusion is justified?	theorem, certificate replay, runtime conformance, byte identity

21.2 Required inference gates

- **Gate 1 - Identify:** name the source sort before manipulating the object.
- **Gate 2 - Normalize:** distinguish exact form from decimal orientation.
- **Gate 3 - Scope:** state the operation and its domain, including all finite bounds.
- **Gate 4 - Interface:** cite the declared map that permits movement to another sort.
- **Gate 5 - Preserve:** state exactly what the map preserves and what it does not preserve.
- **Gate 6 - Label:** classify the conclusion as proof, certificate, conformance, or hash identity.
- **Gate 7 - Version:** attach the frozen contract or status version when reporting a result.

21.3 Compact AI instruction block

ROLE: Interpret the Omega Naught G6 architecture as a typed exact system.

FOR EVERY CLAIM:

1. Name the source sort and target sort.
2. Name the operation and exact input domain.
3. Separate exact identities from decimal approximations.
4. Use only a declared interface between sorts.
5. State what that interface preserves.
6. Label the evidence: proof, finite certificate, runtime test, or hash.
7. Reject silent substitutions, especially $\lambda := Q/S$.
8. Keep Class-I cubature derivation in $Q(\text{sqrt}(5))$; use Omega only as the declared exact coefficient representation.
9. Describe RESULT=PASS as conformance to tested consequences.
10. If specification data is absent, say which field is missing; do not invent it.

21.4 AI type table

Input type	Allowed next operation	Disallowed shortcut
Real λ	Certified interval and floor on the declared integer domain.	Replace globally by Q/S.
Pair (Q,S)	Exact quotient, residual, ledger, reverse step.	Treat residuals from a reduced fraction as canonical.
Q($\sqrt{5}$) coefficient	Exact Omega-basis map.	Claim Omega generated its geometric proof.
Omega field element	Reduction, conjugation, canonical serialization.	Evaluate generically using Q/S.
UTF-8 bytes	MSB-first 15-bit fold or direct canonical hash.	Infer semantic meaning from the bit pattern.
Digest	Compare against the named canonical target.	Call digest equality a theorem proof.

21.5 Example claim classification

Statement	AI verdict	Reason
Q/S equals λ .	False	They are distinct numbers; only selected floor outputs agree on a finite domain.
The $f=2$ nodes admit a positive degree-9 rule.	Mathematical result	Exact Class-I moment and positivity proof; the supplied status includes $f=2$.
The Rust runner prints PASS.	Insufficient alone	The test list, inputs, computed outputs, and contract version are needed.
Two canonical digests match.	Byte identity claim	Meaning depends on the named canonical schema and reconstructed bytes.

21.6 AI uncertainty behavior

If a required manifest field, coefficient table, exact endpoint, or schema rule is missing, the correct response is to identify the missing dependency and stop the affected inference. Filling a gap with a plausible guess destroys clean-room reconstructibility. An AI may explain the general mathematics while clearly marking the frozen value as unavailable.

AI success criterion The AI has understood G6 when it can reconstruct permitted consequences and also refuse invalid cross-layer conclusions.

Checkpoint What should an AI do when it sees the same decimal in two branches?

Answer: Check the exact definitions and types. Decimal resemblance is not a declared interface.

22. Operator workflow and failure diagnosis

A useful manual must tell a reader not only what the architecture means, but how to operate it and where to look when a check fails.

22.1 Beginner workflow

1. Reproduce the toy Euclidean example $y=47$, $s=10$.
2. Compute one actual forward step for $d=399$ and verify its residual.
3. Reverse that step using divisibility by Q .
4. Manually fold the first three UTF-8 bytes O , m , e into the first 15-bit limb.
5. Explain in words why Q/S and λ are different even when their floors agree.
6. Explain why μ_f and ν_f differ despite sharing X_f .

22.2 Implementer workflow

1. Parse the frozen manifest strictly and bind every constant to an explicit integer or exact rational type.
2. Implement exact field reduction independently from the Q/S machine.
3. Replay the real interval certificate and exhaustive 15-bit floor equality.
4. Implement forward step, 23-step ledger, reverse step, and N -step identity checks.
5. Implement UTF-8 folding, exact limb width, byte length, and padding rejection.
6. Implement canonical serialization only from normalized objects.
7. Recompute every published hash vector and run negative mutations.
8. Add cubature replay as a separate module with the declared $Q(\sqrt{5})$ to Omega map.

22.3 Diagnose by the first failing layer

Symptom	Likely layer	First checks
Field hash differs but algebra identities pass	Canonical serialization	Fraction reduction, polynomial order, key order, whitespace, final newline.
Floor mismatch within 0..32767	Finite bridge	Exact L/U endpoints, integer cross-products, Q and S digits, domain inclusivity.
Forward quotient matches but reverse fails	SRSD ledger	Residual formula, order, divisibility, frozen Q,S rather than reduced pair.
Recovered limbs differ	Ledger or limb order	Reverse sequence direction, fixed 23 positions, leading 15-bit zeros.
Bytes differ only at end	Padding metadata	<code>byte_len</code> , <code>pad_bits</code> formula, right-zero validation.
Cubature moment differs	Geometry/evidence	Orbit ordering, $Q(\sqrt{5})$ arithmetic, measure normalization, μ versus ν .
Only browser fails	Runtime numeric types	BigInt use, JSON conversion, UTF-8 encoder, values above 2^{53} .

22.4 Reporting a conformance result

```
CONTRACT: <name and version>
RUNTIME: <language, version, platform>
INPUT INVENTORY: <manifest commitment>
TESTS RUN: <exact list and domains>
RESULT: PASS or FAIL
COMPUTED TARGETS: <canonical digests or records>
NEGATIVE TESTS: <mutations and expected rejection>
EXCLUSIONS: <tests or evidence not run>
INTERPRETATION: runtime conformance only; theorem status cited separately
```

22.5 Common implementation hazards

- Silent signed overflow in $Q*d$.
- Using a floating-point Number for 15-digit constants in JavaScript or TypeScript.
- Reducing Q and S before computing canonical residuals.
- Losing leading zeros when turning limbs back into 15-bit strings.
- Sorting residuals, or reversing them twice.
- Accepting duplicate JSON keys or noncanonical fractions.
- Hashing a pretty-printed display instead of the frozen canonical bytes.
- Using equal weights when the rule requires orbit-constant positive weights.

23. What the architecture demonstrates

The strongest conclusion is precise rather than grand: selected mathematical consequences can be made independently reconstructible across exact mathematics, finite certificates, and software runtimes.

geometry -> exact structure -> typed representation -> certified finite consequence

-> reversible state record -> canonical bytes -> reproducible commitment

23.1 Demonstrated capabilities

- Exact separation of scale and direction on the distinguished positive ray.
- A two-channel exact algebra for the Omega constants and $Q(\sqrt{5})$ coefficient interoperability.
- A rational interval certificate that decides a real floor operation on every unsigned 15-bit state.
- A contracting integer machine whose quotient loss is exactly repaired by ordered residual ledgers.
- A reversible bridge between arbitrary UTF-8 bytes and the finite state domain.
- Canonical records and cryptographic commitments that independent runtimes can reconstruct.
- Typed assembly with an independent positive Class-I cubature branch on fixed supports.

23.2 Claims intentionally not made

- That Omega_0 generates every mathematical branch in G6.
- That Q/S is equal to λ or can replace it in arbitrary expressions.
- That software replay replaces theorem proof.
- That a hash establishes semantic truth.
- That UTF-8 characters are geometrically derived by the byte fold.
- That the open cubature range $f \geq 10$ is already proved in the supplied status snapshot.

23.3 Why the architecture is useful

A reader need not trust a screenshot, one decimal calculation, or one program. The public definitions determine exact objects; certificates determine finite consequences; canonical encodings determine bytes; independent implementations determine whether those consequences are reproducible. The boundaries are as important as the connections because they prevent a convenient representation from being mistaken for a proof or origin.

Definitive interpretation The Omega Naught G6 architecture makes its specified computational consequences independently reconstructible. It preserves the distinctions between proof, certificate, conformance, representation, and commitment.

PART V

Reference appendices

Frozen values, pseudocode, vocabulary, and claim corrections.

Read this part in order if the mathematics is new.

Each chapter starts with intuition and ends with an exact contract.

A. Frozen constants and reference values

Use this appendix as a quick lookup. Exact definitions control; decimals are orientation only.

Symbol or field	Frozen definition or value	Role
ϕ	$(1+\sqrt{5})/2$	Positive golden ratio.
Ω_0	$\phi+\pi$	Distinguished total.
$T=\tau$	$1/\Omega_0$	Reciprocal scale.
λ	ϕ/Ω_0	Normalized ϕ direction and real floor parameter.
μ	$\pi/\Omega_0=1-\lambda$	Normalized π direction.
K	$Q(T)[\text{Lambda}]/(\text{Lambda}^2-T\text{Lambda}-T^2)$	Exact two-channel field.
Q	339949771344778	Finite machine numerator.
S	1000000000000000	Finite machine scale.
State bits	15	Unsigned states 0..32767.
Ledger steps	23	Fixed transitions per limb.
First mismatch d^*	42189065	First certified floor disagreement beyond the operational domain.
Payload	Omega Naught	Published UTF-8 fold vector.
Limbs	10166, 22873, 27684, 1254, 2987, 7585, 26624	Published 15-bit sequence.
Pad bits	9	Right-zero padding in the final limb.
Cubature status	proved $f=1..9$; open $f\geq 10$	Status in the supplied 24 August 2026 snapshot.

A.1 Core exact identities

$$\Omega_0 \cdot T = 1$$

$$\lambda + \mu = 1$$

$$\lambda^2 - T\lambda - T^2 = 0$$

$$\phi^2 - \phi - 1 = 0$$

$$\sqrt{5} = 2\lambda/T - 1$$

A.2 Orientation decimals

Constant	Orientation only
ϕ	1.61803398874989484820...
π	3.14159265358979323846...
Ω_0	4.75962664233968808666...
τ	0.21010051315882842700...
λ	0.33994977134477892958...
μ	0.66005022865522107042...

B. Language-independent reference pseudocode

This pseudocode shows the contract structure. Concrete implementations must still follow the frozen manifest, numeric types, schemas, and error behavior.

B.1 Forward and reverse steps

```
function STEP(d):
  require 0 <= d < 32768
  y = Q * d
  next = y div S
  rho = y - S * next
  require 0 <= rho < S
  return (next, rho)

function REVERSE(next, rho):
  require next >= 0
  require 0 <= rho < S
  numerator = S * next + rho
  require numerator mod Q == 0
  d = numerator div Q
  return d
```

B.2 Fixed-length ledger

```
function LEDGER(seed):
  d = seed
  residuals = []
  for k in 0..22:
    (d, rho) = STEP(d)
    append residuals, rho
  require d == 0
  return {terminal: d, residuals: residuals}

function RECOVER(ledger):
  require length(ledger.residuals) == 23
  d = ledger.terminal
  for rho in reverse(ledger.residuals):
    d = REVERSE(d, rho)
  return d
```

B.3 Bytes to limbs

```
function BYTES_TO_LIMBS(bytes):
  bits = concatenate each byte as exactly 8 MSB-first bits
  pad_bits = (15 - (length(bits) mod 15)) mod 15
  bits = bits followed by pad_bits zero bits
  limbs = each consecutive 15-bit group interpreted unsigned
  return {byte_len: length(bytes), pad_bits: pad_bits, limbs: limbs}
```

B.4 Limbs to bytes

```
function LIMBS_TO_BYTES(record):
  expected = ceil(8 * record.byte_len / 15)
  require length(record.limbs) == expected
  require record.pad_bits == 15*expected - 8*record.byte_len
  require every limb is in 0..32767
  bits = concatenate each limb as exactly 15 bits
  require the last record.pad_bits bits are all zero
  payload_bits = first 8*record.byte_len bits
  return each consecutive 8-bit group interpreted as a byte
```

B.5 Exact interval floor check

```
for d in 0..32767:
  n = (Q * d) div S
  require n <= L60 * d          # rational comparison by cross-products
  require U60 * d <= n + 1
  conclude floor(lambda * d) == n because L60 < lambda < U60
```

B.6 Canonical rational

```
function CANONICAL_RATIONAL(n, d):
  require d != 0
  if d < 0: n = -n; d = -d
  g = gcd(abs(n), d)
  n = n div g; d = d div g
  if n == 0: return "0"
  if d == 1: return decimal(n)
  return decimal(n) + "/" + decimal(d)
```

C. Glossary

These definitions use the meaning intended in this manual. The individual papers provide the formal theorem context.

Term	Plain definition
Canonical	One uniquely permitted representation selected from all equivalent spellings.
Certificate	Finite exact evidence that allows a stated consequence to be checked.
Clean-room implementation	An implementation reconstructed from the public specification without calling or copying a hidden reference result.
Conformance	Agreement with a declared computational contract under named tests and inputs.
Cubature	A weighted finite rule for exact or approximate multidimensional integration.
Domain	The set of inputs for which a statement or operation is declared.
Embedding	A structure-preserving identification of one mathematical object inside another.
Exact arithmetic	Arithmetic on representations whose operations preserve the mathematical value without rounding.
Floor	The greatest integer not exceeding a real number.
Hash commitment	A fixed digest that identifies a byte sequence with strong change detection.
Ledger	An ordered list of transition residuals plus terminal data used for exact reverse reconstruction.
Limb	One unsigned 15-bit group formed from the canonical bitstream.
Manifest	The authoritative inventory and metadata defining a frozen release or contract.
Measure	Nodes together with their weights; the weights are part of the object.
Normal form	A unique reduced mathematical representation, here $A(T)+B(T)\lambda$.
Orbit	A set of nodes carried into one another by the symmetry group.
Projection	A map that deliberately keeps only part of a state, such as the quotient.
Residual	The within-cell remainder required to reconstruct the represented state.
Serialization	A deterministic conversion of a typed object into bytes.
Simplex	The space of nonnegative coordinate proportions summing to one.
Support	The set of nodes where a discrete measure places weight.
Typed interface	A declared map with a source sort, target sort, domain, and preservation claim.

D. Invalid shorthand and accurate replacements

Many errors arise from compressing a correct scoped statement into an incorrect universal slogan.

Invalid shorthand	Accurate replacement
Q/S equals λ .	Q/S is a distinct rational whose floor product agrees with λ on the declared unsigned 15-bit domain.
The rational machine is exact for λ .	The machine exactly computes its own Q/S quotient and residual, and a certificate equates selected quotient outputs with the real λ floor.
Dropping the residual is reversible.	The quotient plus the ordered residual reconstructs the predecessor; the quotient alone generally does not.
Any order of residuals works.	Residual p_k belongs to transition k and must be read in reverse chronological order for recovery.
The byte fold is geometric English.	It is UTF-8 bytes partitioned MSB-first into 15-bit transport limbs.
Cubature is generated by Omega_0.	Class-I geometry produces $Q(\sqrt{5})$ coefficients; Omega supplies an exact coefficient encoding.
Equal-weight obstruction means the nodes cannot work.	The same support may admit a different positive weighted cubature measure.
$t=1/27$ proves $f=1..9$.	$t=1/27$ is a positivity witness only for the $f=9$ affine family; $f=1..8$ have separate exact rules.
PASS proves the mathematics.	PASS reports reproduction of the computational consequences named by the test contract.
A matching hash proves truth.	A matching hash identifies canonical bytes; semantic truth comes from the mathematical and specification layers.
The ZIP is the release.	The frozen file set and canonical manifest identify the release; a ZIP is an optional container.

E. Source and status references

This manual expands the supplied definitive G6 architecture manual. The papers remain authoritative for their theorem statements, hypotheses, proofs, and publication status.

Primary instructional source. Omega Naught Architecture Manual: A Definitive Manual of the G6 Global Object, supplied 20-page PDF, status date 24 August 2026. The architecture source named in that manual is omega-naught.grok.me.

E.1 Published paper sequence

Paper	Reference
I	Newman, M. (2026). Newman's Quantized Omega Contraction Theorem. Zenodo. doi:10.5281/zenodo.21830285
II	Newman, M. (2026). Ordered Euclidean Remainders for Exact Orbit Reconstruction and Parameter Intervals in Rational Floor Maps. Zenodo. doi:10.5281/zenodo.21843485
III	Newman, M. (2026). Scale-Resolved State Dynamics: Foundational Closure Specification (Version 1.0). Zenodo. doi:10.5281/zenodo.21927248
IV	Newman, M. (2026). Radial Icosahedral Sampling and Its First Harmonic Obstruction: Exact degree-six activation and Euler-Maclaurin coefficients. Zenodo. doi:10.5281/zenodo.22033917
V	Newman, M. (2026). OMEGA NAUGHT Ω_0 . Zenodo. doi:10.5281/zenodo.22048540
VI	Newman, M. (2026). Exact Positive Icosahedral Cubature on Fixed Radial Class-I Grids: Quadratic-Field Interoperability, Exact Replay, and Typed Compatibility with Papers I-V. Zenodo. doi:10.5281/zenodo.22068873

E.2 Status discipline

This instructional edition deliberately uses conservative status language. Exact identities are called exact only where algebraically established. Finite results retain their stated domains. Runtime conformance is not substituted for proof. Representation maps are not described as mathematical origins. The cubature range $f \geq 10$ remains labeled open because that is the status in the supplied 24 August 2026 source snapshot.

Final reading rule When a later release changes a proof, domain, hash, evidence set, or open status, update the manifest and version first. Do not silently retrofit the new fact into an older frozen contract.