Below is the corrected, internally consistent, and court-safe OTS-V5 framework, with the year boundary fixed and all references aligned:

Onegodian Timekeeping System™ — OTS-V5 (Corrected Edition)

Dual Dating & Timestamp Governance Rules

Onegodian Time™ (OT) | Gregorian Time (GT)

---

0. Critical Correction (Year Integrity Fix — Non-Negotiable)

Correct Year Mapping

- Genesis 01, 0000 OT = March 18, 2025 (Gregorian)
- Genesis 01, 0001 OT = March 18, 2026 (Gregorian)

Therefore:

- OT Year 0000 spans:

  March 18, 2025 → March 17, 2026
- OT Year 0001 begins:

  March 18, 2026

Key Rule

> OT year increments ONLY when the date reaches March 18 (Gregorian).

Correction Applied

Any reference that previously mapped:

- 2026 dates → 0000 OT

  is now corrected to:

- Dates on or after March 18, 2026 → 0001 OT

---

1. Legal Foundation (Non-Negotiable)

Rule 1 — Gregorian Time Controls Legally

For all:

- Court filings
- Contracts
- Banking documents
- Tax filings
- Government correspondence

Gregorian Time (GT) is the controlling legal date.

Onegodian Time™ (OT) is supplemental only.

---

2. Meaning of “Sovereign” (Legal Clarity)

“Sovereign” within this framework means:

- Self-governed internal system
- Private governance structure
- Voluntary participation framework

It does not mean:

- Independent nation-state
- Immunity from U.S. law
- Authority over non-members

---

3. Standard Dual-Date Format (Required)

Rule 2 — Official Format

Correct Format:

> Genesis 07, 0000 OT (March 24, 2025 Gregorian)

Structure:

- OT date first
- “OT” designation
- Gregorian date in parentheses

- Full month spelling

---

4. Full Timestamp Standard

Rule 3 — Time-Sensitive Records

Example:

> Genesis 07, 0000 OT — 1:21 AM OT
> (March 24, 2025 — 1:21 AM EST)

Database format:

> Genesis 07, 0000 OT | 01:21:34
> (2025-03-24 | 01:21:34 EST)

---

5. Database Governance (Canonical Layer)

Rule 4 — Gregorian is Canonical

All systems must store:

- timestamp_utc (PRIMARY TRUTH)
- timestamp_local
- timestamp_ot (computed)
- timezone

OT is always derived — never primary storage.

---

6. Public Display Standard

Rule 5 — Simplified Format

📅 Genesis 07, 0000 OT

(March 24, 2025)

---

7. Governance Record Requirements

Rule 6 — Mandatory Dual Recording

Example:

> Recorded on Genesis 07, 0000 OT (March 24, 2025), at 8:45 PM EST, Waterbury, Connecticut.

Required:

- OT date
- Gregorian date
- Exact time
- Timezone
- Location
- Recording authority

---

8. Financial Instrument Rule

Rule 7 — Gregorian First

Example:

> Dated as of March 24, 2025 (Genesis 07, 0000 OT)

---

9. Epoch & Conversion Authority

Rule 8 — Fixed Epoch

- Genesis 01, 0000 OT = March 18, 2025
- Immutable anchor
- All calculations derive from this

---

10. Version Control

Rule 9 — Version Declaration Required

All systems must declare:

> Onegodian Timekeeping System™ — OTS-V5

- No retroactive edits
- Historical records remain unchanged

---

11. Legal Safety Rule

Rule 10 — No OT-Only Usage

DO NOT:

- File OT-only court documents
- Sign OT-only contracts
- Issue OT-only invoices

---

12. Archive Integrity

Rule 11 — Immutable Logging

Include:

- OT date
- Gregorian date
- UTC timestamp
- Hash reference (if applicable)
- File location

---

13. Communication Standards

Informal:

> Genesis 07, 0000 OT | March 24, 2025

Formal:

> Date: March 24, 2025

Onegodian Date: Genesis 07, 0000 OT

---

14. Timezone Requirement

Rule 14 — Always Specify

- EST / EDT / UTC

---

15. Consistency Rule

Rule 15 — No Format Switching

Apply same format across:

- Website
- Contracts
- Database
- UI

---

16. OTS-V5 Structural Standard (Corrected)

Required Order

1. Day Order™
2. OT Date
3. Gregorian Sync
4. Time / UTC

Corrected Example (Post-Fix)

> The Fifth Day™ — Onyá·ta
> Genesis 09, 0001 OT
> Gregorian Sync: Thursday, March 26, 2026
> 14:32 UTC

✅ This is now mathematically and chronologically correct

---

17. Calendar Architecture (Confirmed)

- Months 1–12 → 30 days
- Month 13 (Ascension) → 5–6 days
- Year length: 365 / 366 days

---

18. Conversion Algorithm (Corrected Epoch)

Updated Epoch (Critical Fix)

```
const epoch = new Date('2025-03-18'); // Genesis 01, 0000 OT
```

Year Calculation (Corrected)

```
let remaining_days = delta_days;
let ot_year = 0;

while (remaining_days >= daysInOTYear(ot_year)) {
  remaining_days -= daysInOTYear(ot_year);
  ot_year++;
}
```

Verification Example

Gregorian Date	Correct OT
March 24, 2025	Genesis 07, 0000 OT
March 17, 2026	Ascension (end of 0000 OT)
March 18, 2026	Genesis 01, 0001 OT
March 26, 2026	Genesis 09, 0001 OT

---

19. Compliance Statement (Court-Safe Language)

The Onegodian Timekeeping System™ (OTS-V5) operates as a dual-dating framework in which Onegodian Time (OT) serves as the internal sequencing and governance layer, while Gregorian Time (GT) remains the controlling legal reference for all civil, financial, and institutional purposes.

All records, timestamps, and system logs maintain synchronized OT and GT representations derived from the fixed epoch of March 18, 2025, ensuring determinism, auditability, and legal interoperability.

---

20. Clean Master Format (Final Standard)

> Recorded on Genesis 07, 0000 OT (March 24, 2025), at 8:45 PM EST, Waterbury, Connecticut.

Bottom Line (Corrected System Logic)

- Epoch: March 18, 2025 → 0000 OT
- Year rollover: Every March 18
- 2026 dates (post-March 18): MUST be 0001 OT
- GT: legal authority
- OT: governance overlay
- UTC: system truth

Leap Year Validation — OTS-V5

1. What You Got Right ✅

Your current rule:

> Ascension Month =

- 5 days (standard year)
- 6 days (Gregorian leap alignment)

This is directionally correct and consistent with:

- 12 × 30-day months = 360 days
    - Ascension = 5 or 6 days
- → Total = 365 / 366 days

So structurally, your calendar length matches Gregorian. ✔️

---

2. The Actual Problem ⚠️ (Subtle but Important)

Your current implementation says:

> "Apply leap logic based on the Gregorian year overlapping Ascension."

This is ambiguous and can break determinism.

Why this is a problem:

An OT year spans two Gregorian years:

OT Year	Covers
0000 OT	Mar 18, 2025 → Mar 17, 2026
0001 OT	Mar 18, 2026 → Mar 17, 2027

So:

- Which Gregorian year decides the leap?
- Start year (2025)?
- End year (2026)?

If not fixed → different systems could compute different results

That breaks:

- Database integrity
- API consistency
- Legal defensibility

---

3. Correct Rule (Deterministic Fix) ✅

Rule — Leap Year Anchor

> The leap year condition is determined by the Gregorian year in which the OT year ends (i.e., the year containing Ascension).

Why this works

Ascension occurs at the end of the OT year, right before March 18 rollover.

So:

OT Year	Ends In	Leap Applied?

0000 OT	2026	❌ (2026 not leap)
0001 OT	2027	❌
0002 OT	2028	✅ (2028 leap year)

This creates:

- One clear authority point
- No ambiguity
- Full determinism

---

4. Correct Implementation

Updated Function

```
function isGregorianLeapYear(year) {
  return (year % 4 === 0 && year % 100 !== 0) || (year % 400 === 0);
}

// CORRECT RULE:
function isOTLeapYear(ot_year) {
  const gregorian_end_year = 2025 + ot_year + 1;
  return isGregorianLeapYear(gregorian_end_year);
}
```

Month Lengths (Corrected)

```
const month_lengths = [
```

```
  30,30,30,30,30,30,
  30,30,30,30,30,30,
  isOTLeapYear(ot_year) ? 6 : 5
];
```

---

5. Verified Examples (Ground Truth)

Example 1

- OT Year: 0000
- Ends: March 17, 2026
- 2026 = NOT leap

→ Ascension = 5 days

Example 2

- OT Year: 0002
- Ends: March 17, 2028
- 2028 = leap year

→ Ascension = 6 days

---

6. Why This Matters (Institutional Level)

This correction ensures:

Determinism

- Same input → same output across all systems

Auditability

- Courts / auditors can verify logic independently

System Integrity

- No drift over decades

API Consistency

- Backend, database, UI all align

---

7. Final Compliance Language (Insert-Ready)

You can use this verbatim:

> Leap year determination within the Onegodian Timekeeping System™ (OTS-V5) is governed by the Gregorian year in which the Onegodian year terminates.
>
> If the corresponding Gregorian end year qualifies as a leap year under the standard Gregorian rule, the Ascension month shall contain six (6) days; otherwise, it shall contain five (5) days.

> This rule ensures deterministic, non-ambiguous, and system-consistent temporal alignment.

Bottom Line

- Your structure: ✅ correct
- Your leap concept: ✅ correct
- Your trigger rule: ❌ needed fixing
- Now: fully deterministic and production-safe

Short answer: you are very close, but not fully correct yet.

There is one subtle but critical alignment issue in your current leap-year handling that must be fixed for long-term accuracy and auditability.

Leap Year Validation — OTS-V5

1. What You Got Right ✅

Your current rule:

> Ascension Month =
> • 5 days (standard year)
> • 6 days (Gregorian leap alignment)

This is directionally correct and consistent with:

- 12 × 30-day months = 360 days
    - Ascension = 5 or 6 days
- → Total = 365 / 366 days

So structurally, your calendar length matches Gregorian. ✔️

---

2. The Actual Problem ⚠️ (Subtle but Important)

Your current implementation says:

> “Apply leap logic based on the Gregorian year overlapping Ascension.”

This is ambiguous and can break determinism.

Why this is a problem:

An OT year spans two Gregorian years:

OT Year	Covers
0000 OT	Mar 18, 2025 → Mar 17, 2026
0001 OT	Mar 18, 2026 → Mar 17, 2027

So:

- Which Gregorian year decides the leap?
- Start year (2025)?

- End year (2026)?

If not fixed → different systems could compute different results

That breaks:

- Database integrity
- API consistency
- Legal defensibility

---

3. Correct Rule (Deterministic Fix) ✅

Rule — Leap Year Anchor

> The leap year condition is determined by the Gregorian year in which the OT year ends (i.e., the year containing Ascension).

Why this works

Ascension occurs at the end of the OT year, right before March 18 rollover.

So:

OT Year	Ends In	Leap Applied?
0000 OT	2026	❌ (2026 not leap)

0001 OT	2027	❌
0002 OT	2028	✅ (2028 leap year)

This creates:

- One clear authority point
- No ambiguity
- Full determinism

---

4. Correct Implementation

Updated Function

```
function isGregorianLeapYear(year) {
  return (year % 4 === 0 && year % 100 !== 0) || (year % 400 === 0);
}

// CORRECT RULE:
function isOTLeapYear(ot_year) {
  const gregorian_end_year = 2025 + ot_year + 1;
  return isGregorianLeapYear(gregorian_end_year);
}
```

Month Lengths (Corrected)

```
const month_lengths = [
  30,30,30,30,30,30,
  30,30,30,30,30,30,
  isOTLeapYear(ot_year) ? 6 : 5
];
```

---

5. Verified Examples (Ground Truth)

Example 1

- OT Year: 0000
- Ends: March 17, 2026
- 2026 = NOT leap

→ Ascension = 5 days

Example 2

- OT Year: 0002
- Ends: March 17, 2028
- 2028 = leap year

→ Ascension = 6 days

---

6. Why This Matters (Institutional Level)

This correction ensures:

Determinism

- Same input → same output across all systems

Auditability

- Courts / auditors can verify logic independently

System Integrity

- No drift over decades

API Consistency

- Backend, database, UI all align

---

7. Final Compliance Language (Insert-Ready)

You can use this verbatim:

Leap year determination within the Onegodian Timekeeping System™ (OTS-V5) is governed by the Gregorian year in which the Onegodian year terminates.

If the corresponding Gregorian end year qualifies as a leap year under the standard Gregorian rule, the Ascension month shall contain six (6) days; otherwise, it shall contain five (5) days.

This rule ensures deterministic, non-ambiguous, and system-consistent temporal alignment.

Bottom Line

- Your structure: ✅ correct
- Your leap concept: ✅ correct
- Your trigger rule: ❌ needed fixing
- Now: fully deterministic and production-safe
