/property-based-testing

Source: `~/.claude/skills/tob-property-based-testing/skills/property-based-testing/SKILL.md`

---

name: property-based-testing
description: Provides guidance for property-based testing across multiple languages and smart contracts. Use when writing tests, reviewing code with serialization/validation/parsing patterns, designing features, or when property-based testing would provide stronger coverage than example-based tests.

Property-Based Testing Guide

Use this skill proactively during development when you encounter patterns where PBT provides stronger coverage than example-based tests.

When to Invoke (Automatic Detection)

Invoke this skill when you detect:

- **Serialization pairs**: `encode`/`decode`, `serialize`/`deserialize`, `toJSON`/`fromJSON`, `pack`/`unpack`
- **Parsers**: URL parsing, config parsing, protocol parsing, string-to-structured-data
- **Normalization**: `normalize`, `sanitize`, `clean`, `canonicalize`, `format`
- **Validators**: `is_valid`, `validate`, `check_*` (especially with normalizers)
- **Data structures**: Custom collections with `add`/`remove`/`get` operations
- **Mathematical/algorithmic**: Pure functions, sorting, ordering, comparators
- **Smart contracts**: Solidity/Vyper contracts, token operations, state invariants, access control

Priority by pattern:

Pattern	Property	Priority
encode/decode pair	Roundtrip	HIGH
Pure function	Multiple	HIGH
Validator	Valid after normalize	MEDIUM
Sorting/ordering	Idempotence + ordering	MEDIUM
Normalization	Idempotence	MEDIUM
Builder/factory	Output invariants	LOW
Smart contract	State invariants	HIGH

When NOT to Use

Do NOT use this skill for:

- Simple CRUD operations without transformation logic
- One-off scripts or throwaway code
- Code with side effects that cannot be isolated (network calls, database writes)
- Tests where specific example cases are sufficient and edge cases are well-understood
- Integration or end-to-end testing (PBT is best for unit/component testing)

Property Catalog (Quick Reference)

Property	Formula	When to Use
Roundtrip	`decode(encode(x)) == x`	Serialization, conversion pairs
Idempotence	`f(f(x)) == f(x)`	Normalization, formatting, sorting
Invariant	Property holds before/after	Any transformation
Commutativity	`f(a, b) == f(b, a)`	Binary/set operations
Associativity	`f(f(a,b), c) == f(a, f(b,c))`	Combining operations
Identity	`f(x, identity) == x`	Operations with neutral element
Inverse	`f(g(x)) == x`	encrypt/decrypt, compress/decompress
Oracle	`new_impl(x) == reference(x)`	Optimization, refactoring
Easy to Verify	`is_sorted(sort(x))`	Complex algorithms
No Exception	No crash on valid input	Baseline property

Strength hierarchy (weakest to strongest): No Exception → Type Preservation → Invariant → Idempotence → Roundtrip

Decision Tree

Based on the current task, read the appropriate section:

```
TASK: Writing new tests
  → Read [{baseDir}/references/generating.md]({baseDir}/references/generating.md) (test generation patterns and examples)
  → Then [{baseDir}/references/strategies.md]({baseDir}/references/strategies.md) if input generation is complex

TASK: Designing a new feature
  → Read [{baseDir}/references/design.md]({baseDir}/references/design.md) (Property-Driven Development approach)

TASK: Code is difficult to test (mixed I/O, missing inverses)
  → Read [{baseDir}/references/refactoring.md]({baseDir}/references/refactoring.md) (refactoring patterns for testability)
```

```
TASK: Reviewing existing PBT tests
  → Read [{baseDir}/references/reviewing.md]({baseDir}/references/reviewing.md) (quality
checklist and anti-patterns)

TASK: Need library reference
  → Read [{baseDir}/references/libraries.md]({baseDir}/references/libraries.md) (PBT libraries
by language, includes smart contract tools)
```

How to Suggest PBT

When you detect a high-value pattern while writing tests, **offer PBT as an option**:

> "I notice `encode_message`/`decode_message` is a serialization pair. Property-based testing with a roundtrip property would provide stronger coverage than example tests. Want me to use that approach?"

If codebase already uses a PBT library (Hypothesis, fast-check, proptest, Echidna), be more direct:

> "This codebase uses Hypothesis. I'll write property-based tests for this serialization pair using a roundtrip property."

If user declines, write good example-based tests without further prompting.

When NOT to Use PBT

- Simple CRUD without complex validation
- UI/presentation logic
- Integration tests requiring complex external setup
- Prototyping where requirements are fluid
- User explicitly requests example-based tests only

Red Flags

- Recommending trivial getters/setters
- Missing paired operations (encode without decode)

- Ignoring type hints (well-typed = easier to test)
- Overwhelming user with candidates (limit to top 5-10)
- Being pushy after user declines

Revision #5
Created 2026-02-18 08:40:07 UTC by John
Updated 2026-06-21 20:01:14 UTC by John