UntitledMod – Developer Documentation

1. Repository Overview

This repository is a **multi-module Fabric mod project** targeting **Minecraft 1.21.x** using **official Mojang mappings**.

Modules

Module	Purpose
`untitled`	Main mod implementation (server + client)
`untitled-api`	Public API intended for reuse by other mods

The project is built with **Gradle + Fabric Loom** and organized as a composite build via `settings.gradle`.

---

2. Build System & Configuration

Root Project

Key files: - `settings.gradle` – Declares the root project and included modules. - `build.gradle` – Common configuration shared across subprojects. - `gradle.properties` – Version pins (Minecraft, Fabric Loader, Fabric API, Loom).

The root project name is `untitled-root`.

Fabric Loom

- Loom version is resolved from `gradle.properties`.
- Uses Fabric Maven (`https://maven.fabricmc.net`).

Subprojects

Each module (`untitled`, `untitled-api`) has its own `build.gradle` and produces its own artifact.

---

3. untitled-api Module

Purpose

`untitled-api` exposes **stable interfaces and constants** that external mods can depend on without linking against the full implementation.

Source Layout

```
untitled-api/
└ src/api/java/
   └ org/doothy/untitled/api/
```

Mana API

The primary API surface currently revolves around a **mana system**.

Key components:

- `ManaConstants`
• Shared constants (default values, caps, NBT keys, etc.).
- `ManaConsumer`
• Interface for objects/entities that consume mana.
- `ManaProvider`
• Interface for objects/entities that store or supply mana.
- `ManaTransaction`
• Represents a single mana operation (request, consume, rollback semantics).

Design Notes

- API is **side-agnostic** (no client-only or server-only classes).
- No direct dependencies on Fabric events or Minecraft lifecycle hooks.
- Intended to remain binary-compatible across minor releases.

---

4. untitled Module (Main Mod)

Source Layout

```
untitled/
├ src/main/java/      # Common + server-side logic
├ src/client/java/   # Client-only initialization
└ src/main/resources/
```

Entry Points

Server / Common

`org.doothy.untitled.Untitled` - Implements `ModInitializer`. - Responsible for: - Registry setup - Networking registration - Server lifecycle hooks - Gameplay systems (mana, shields, effects)

Client

`org.doothy.untitled.client.UntitledClient` - Implements `ClientModInitializer`. - Handles: - Client-only networking receivers - Rendering or HUD hooks (where applicable)

---

5. Gameplay Systems

Shield / Effect System

- Custom `MobEffect` implementations are registered via `BuiltInRegistries`.
- Shield upkeep is processed **every server tick**.

Tick flow: 1. `ServerTickEvents.END_SERVER_TICK` 2. Iterate over all `ServerLevel`s 3. Iterate over all players 4. Apply shield decay / upkeep logic

This ensures consistent behavior across dimensions.

Player Lifecycle Hooks

Registered Fabric events include: - `ServerPlayerEvents` - `ServerPlayConnectionEvents`

Used to: - Initialize player-specific data - Clean up or resync state on disconnect/reconnect

---

6. Networking

Packet Registration

- Uses `ServerPlayNetworking` (server-bound and client-bound packets).
- Payloads are encoded using `ByteBufCodecs`.

Design Pattern

- Strongly typed payload records
- Explicit identifiers (`Identifier`) per packet
- Clear separation between:
- Sync packets (authoritative server → client)
- Action requests (client → server)

Threading

- All packet handlers assume **server thread execution** unless explicitly scheduled.

---

7. Registries & Data

Registry Usage

- Relies on `BuiltInRegistries` and `Registry.register`.
- Holder-based access (`Holder<T>`) is used where appropriate to support future registry refactors.

Identifiers

- All identifiers follow the mod namespace (`untitled`).
- Centralized constants are preferred over inline literals.

---

8. Client-Side Considerations

- Client module must never reference dedicated-server-only classes.
- Networking receivers mirror server payload definitions.
- Visual feedback (effects, HUD) is derived from synced state, not client authority.

---

9. Extending the Mod

Using the API

External mods should: 1. Depend on `untitled-api` 2. Implement `ManaProvider` / `ManaConsumer` 3. Avoid referencing implementation classes from `untitled`

Adding New Systems

Recommended pattern: - Define interfaces + constants in `untitled-api` - Implement logic in `untitled` - Sync state explicitly via packets

---

10. Versioning & Compatibility

- Minor Minecraft updates may introduce registry and networking refactors.
- This codebase already uses:
- `Holder`
- `ByteBufCodecs` which reduces breakage risk.

When updating Minecraft versions: - Re-check Fabric API event packages - Validate networking codecs - Re-run registry bootstrap order

---

11. Licensing

The repository includes `LICENSE` and `LICENSE.txt` at the root. All contributions must comply with the declared license.

---

12. Recommended Next Documentation

- Packet flow diagrams
- Mana system state machine
- Client/server responsibility matrix
- Public API stability guarantees

---

13. Mana System – Detailed Design

Conceptual Model

The mana system is designed as a **transaction-based resource flow** rather than a mutable counter.

Core principles: - Providers never implicitly mutate state - Consumers request mana via explicit transactions - Transactions can be simulated before commit

Typical Flow

1. Consumer creates a `ManaTransaction` request
2. Provider validates availability
3. Transaction is either:
4. **Committed** (mana deducted)
5. **Rejected** (no state change)

This pattern allows: - Predictive checks (e.g. shield upkeep) - Deterministic server authority - Easier rollback in future extensions

---

14. Shield System – Tick Lifecycle

Server Tick Integration

The shield system is bound to: - `ServerTickEvents.END_SERVER_TICK`

Reasoning: - Ensures all entity updates have completed - Prevents mid-tick desync with effects or damage

Per-Tick Algorithm

For each server tick: 1. Iterate over all loaded `ServerLevel`s 2. Collect active players 3. For each player: - Check active shield effects - Calculate mana upkeep - Attempt mana transaction - Remove or weaken shield on failure

This guarantees identical behavior across dimensions and avoids client-side authority.

---

15. Networking – Packet Semantics

Packet Categories

Category	Direction	Purpose
Sync	Server → Client	Authoritative state updates
Request	Client → Server	Player intent (never state)

Design Rules

- Clients **never** change game state directly
- All packets are validated server-side
- Identifiers are version-stable

Codec Strategy

Uses `ByteBufCodecs` to: - Reduce manual serialization errors - Maintain forward compatibility - Align with Mojang networking refactors (1.20+)

---

16. Client / Server Responsibility Matrix

Concern	Server	Client
Mana storage	✔	✘
Shield decay	✔	✘
Effect logic	✔	✘
Rendering	✘	✔
HUD display	✘	✔
Validation	✔	✘

Rule of thumb: **if it affects gameplay, it lives on the server**.

---

17. Minecraft 1.21.x API Notes

Relevant Mojang/Fabric changes already accounted for:

- Registry access via `BuiltInRegistries`
- Reduced static registry lookups
- Holder-based safety for future dynamic registries
- Modern networking codecs

When updating beyond 1.21.x, re-verify: - `ServerTickEvents` package locations - Networking payload interfaces - Effect registration bootstrap order

---

18. API Stability Guidelines (untitled-api)

Guaranteed Stable

- Interfaces (`ManaProvider`, `ManaConsumer`)
- Constants and NBT keys
- Transaction semantics

Allowed to Change

- Internal implementation classes
- Packet payload contents
- Client-only visuals

External mods should depend **only** on `untitled-api`.

---

19. Suggested Future Work

- Formal state machine for shields
- Data-driven mana costs (JSON)
- Capability-style attachment abstraction
- Dedicated debug overlay (client)

---

20. Documentation Status

This document is suitable for: - Onboarding new contributors - External API consumers - Long-term maintenance across MC updates

Further improvements should focus on diagrams and in-code documentation.