Part II — 參考架構：翾靈系統 - DCP 判定引擎概念架構

Part II — Reference Architecture: Xuanling System

A Conceptual Architecture for DCP-Based Judgment

(Layer 1 · Operational / Interpretive Layer · v0.1)

> ### **Layer Declaration**
> 本文件屬於 DCP 架構中的 **Layer 1（概念／操作詮釋層）**。
> 它不定義新理論、不修改 `Layer 0`，也不構成 `Layer 2` 的實作規範。
> 文中出現之流程、組件與介面，皆為概念性結構，用於說明**「若要將 DCP 作為判定引擎，其內部結構可以如何被理解」**。
> This document belongs to **Layer 1 (Conceptual / Operational Interpretive Layer)** within the DCP architecture. It does not define new theories, does not modify `Layer 0`, nor does it constitute implementation specifications for `Layer 2`. All flows, components, and interfaces described herein are conceptual structures, intended to illustrate **"how the internal structure of a DCP-based judgment engine might be understood."**

---

0. Preface — Scope, Intent, and Semantic Boundaries

本前言確立了本文件的基本性質與詮釋邊界。這對於理解翾靈系統所呈現的嚴格語義語境至關重要。讀者應以「結構理解」而非「實作期待」來閱讀本文，將其視為對語義中介層的描述，而非產品說明、技術手冊或白皮書承諾。

00. Reader Guidance — How to Read This Document

本文旨在作為一份概念性參考架構，而非產品規格、技術實作指南或操作手冊。讀者應將文中所有提及的組件、流程與介面理解為抽象的邏輯角色與概念性關係，而非具體的軟體模組、API 合約或可執行系統元件。本文件不提供任何實作指令或部署策略，其價值在於闡明 DCP `Layer 0` 如何在概念層次被詮釋與結構化。

0.1. Purpose of This Document

本文件將「翾靈系統」概念化為一個基於 DCP `Layer 0` 核心理論的**判定引擎概念架構**。其主要目的在於闡明 DCP 的基本公理與操作原則如何能夠概念性地映射到一個結構化、可詮釋的抽象系統中，從而彌合抽象理論與概念性系統設計之間的鴻溝。

0.2. Scope and Non-Normative Nature

翾靈系統 `v0.1` 是一個**參考架構 (Reference Architecture)**。它僅是一個**概念性草案**，旨在定義判定抽象系統的**抽象邊界與介面**。本文件具備**非規範性**；它不對任何實作、產品或服務施加具體要求。它**不構成實作指令**，不提供任何程式碼、函式庫、框架或部署策略，也不就效能、效率或商業可用性作出任何聲明。

0.3. Relationship to DCP Layer 0

翾靈系統是 DCP `Layer 0` 的**操作性詮釋層**。它：

- **完全從屬**於 DCP `Layer 0`，其所有概念性設計原則與判定邏輯均直接源自並遵循 DCP `Layer 0` 的公理與操作原則。
- **不擴展核心理論**；它不引入新的核心理論或公理，僅將 `Layer 0` 的抽象概念概念性地轉化為可操作的抽象組件與流程。
- 作為 `Layer 0` **可驗證性**的概念載體，旨在展示 `Layer 0` 如何能在一個抽象系統中被概念性地檢視。

0.4. Layers vs. Versions

在本文件中，「Layer」描述的是概念深度與抽象層次，而非版本演進順序。「文件成熟度 (vN)」僅表示文件本身的穩定性與定稿狀態，獨立於其相關的抽象層級。因此，`Layer 0` 文件可能為 `v1.0`，而 `Layer 1` 文件仍可能為 `v0.1`。本文件定位於 `Layer 1`，成熟度為 `v0.1`。

0.5. Interpretive Boundaries and Reading Guidance

本文件僅描述**結構性角色與詮釋性關係，而非系統行為或具體實作。**所有組件、流程與介面的描述均以**概念性與符號性層次**呈現，**不應被誤解為 API 合約、軟體模組或可執行規格。**文中所稱「引擎」僅為概念性隱喻，用以描述功能角色與邏輯位置，並不指涉任何可執行、可部署或已實作之系統元件。所使用的術語（例如：「Request Parser」、「Judgment Core」）指涉的是抽象架構中的邏輯功能與位置，而非具體的軟體實體。本文件採用中英混排，其中核心術語與 Layer 標記保持英文，內文為中文。

---

1. Purpose and Positioning of the Xuanling System

本章詳述翾靈系統的概念性動機及其在 DCP 框架內的指定位置。

1.1. Motivation: From DCP Theory to an Operable Conceptual Layer

翾靈系統可被理解為 DCP `Layer 0` 的抽象操作層。其主要動機是為 DCP 的理論形式化提供一座通往概念性可操作架構的橋樑。它旨在闡明 DCP 的核心公理與操作原則如何能夠概念性地轉化為一個可被概念性設計、討論與驗證的判定引擎結構，而無需承諾具體的實作細節。

1.2. What the Xuanling System Is

翾靈系統的核心概念性職責是作為一個**判定引擎 (Judgment Engine)**。其概念性功能可被理解為嚴格限於根據 DCP `Layer 0` 所定義的條件場，對**外部候選行為的結構有效性進行概念性判斷**。本文所稱「判定 (Judgment)」僅指對條件集合進行結構性可行性分類之抽象概念，不構成決策、選擇或代理行為。其設計目標是：

- **DCP `Layer 0` 的詮釋與映射**：將 `Layer 0` 的公理與操作原則概念性地映射至抽象組件與操作邏輯。
- **行為可行性判定**：其概念性角色是接收候選行為，並對其結構有效性進行概念性判斷。
- **最小化生成，最大化判定**：概念性地嚴格分離行為生成與判定，僅專注於後者。
- **提供標準化判定介面**：定義與外部生成器或應用概念性互動的最小、標準化抽象介面。

1.3. What the Xuanling System Is Not

明確指出翾靈系統作為一個概念架構**明確不是**什麼至關重要：

- 它**不是生成器**；它不創造內容、行動或決策之概念性實體。
- 它**不是代理或自治系統**；它沒有獨立的意志、目標或內在代理性。
- 它**不是 AI 模型或特定的 AI 系統**；它是一個抽象的判定框架，概念性地適用於任何執行判定的系統（無論是人類或人工）。
- 它**不是產品、服務或解決方案**；它是一個概念模型。
- 它**不聲稱具備「模型意識」、「能力升級」**或任何超越 DCP `Layer 0` 所描述的結構性判斷範圍。

1.4. Role Within the DCP Layered Stack

翾靈系統（`Layer 1`）作為 DCP（`Layer 0`）的**操作性詮釋層**。它提供一個穩定、以判定為核心的概念性單元，允許更高層次（`Layer 2+`）的應用與部署空間在其上進行投影。`Layer 1` 定義了基於 DCP 的判定抽象系統**如何**在概念上進行互動，但它**不定義 `Layer 2` 應用將如何具體使用判定結果**，也不指定其商業模式或使用者介面。

---

2. Conceptual Role of the Xuanling System

本章進一步闡述翾靈系統的內在角色與概念邊界，強調其在 DCP 框架內的特定功能。

2.1. Judgment-Oriented Architecture

翾靈系統體現了一個根本上以判定為導向的架構。其設計重點在於概念性地確立候選行為在預定義條件場內的結構有效性，而非發起或生成這些行為。這種架構選擇強化了 DCP 原則，即判定是一個獨立、可驗證的過程，有別於創造。

2.2. Separation Between Generation and Judgment

翾靈系統概念設計的基石是嚴格分離**生成**（提出行為）與**判定**（評估其可行性）的過程。該抽象系統明確假設候選行為源自外部。這與 DCP `Layer 0` 的公理 M6（生成/判定分離）保持一致，旨在確保判定核心保持公正，並專注於結構驗證。

2.3. Structural Neutrality and Non-Agency

翾靈系統保持絕對的概念性結構中立性。它不具備、不假設、不推斷意圖、策略或倫理立場。其功能角色純粹是結構性的：概念性地比對候選行為與活躍條件場的條件和約束之間的一致性。它是一個非代理實體，無法發起行動或做出基於價值的決策。

2.4. Relationship to External Generators and Consumers

翾靈系統在概念上設計為與外部「生成器」（提出候選行為）和「消費者」（利用判定輸出）進行概念性互動。它可被理解為充當公正的驗證者之概念性角色，為提議行動的結構可行性提供結構化回饋。這種概念性關係旨在確保翾靈系統仍然是一個核心判定組件，同時允許多樣化的外部抽象系統靈活運用其能力。

---

3. Conceptual Components of the Xuanling System

(所有組件均以抽象、非實作層次描述，這些組件代表概念性角色而非架構模組。)

翾靈系統在概念上由以下核心組件構成。這些組件在邏輯上是獨立的，但它們 `MAY`（Level C 詮釋）或 `MUST`（Level A 詮釋）概念性地共享資源或在單一物理實體內概念性地實作。每個組件代表一個**概念性角色**，而非具體的軟體模組或服務。

3.1. Request Parser (概念性組件)

此組件代表一個概念性角色，而非軟體模組。

- **概念性職責**：其概念性角色為接收外部 `DCP_Request` 物件，並將其概念性地解析為此抽象架構內部可理解的任務描述、上下文和初始假設。
- **映射 `Layer 0`**：其設計旨在確保經概念性解析後的輸入能符合 `Layer 0` 對「條件」和「錨點」的初步識別要求。
- **概念性輸出**：結構化的抽象內部任務物件 (`Xuanling_Task`)。

3.2. Constraint Field Builder (概念性組件)

此組件代表一個概念性角色，而非軟體模組。

- **概念性職責**：其概念性角色為根據經解析後的任務物件 (`Xuanling_Task`) 和此抽象架構的概念性狀態，概念性地建構出當前的「條件場」。這包括概念性地識別相關的不可消元變量（錨點）及其優先序。

- **映射 `Layer 0`**：直接概念性地體現 `Layer 0` 中對「條件場」和「不可消元變量」的定義（參見 `Layer 0` 第 5.2.1 節和 5.2.2 節，以及 M0 公理 M1、M2、M3）。
- **概念性輸出**：當前任務的抽象「判定上下文物件」(`Xuanling_JudgmentContext`)，包含概念性定義好的可行域 $\Phi(s)$ 和不可消元變量 $U(s)$。

3.3. Candidate Behavior Receiver (概念性組件)

此組件代表一個概念性角色，而非軟體模組。

- **概念性職責**：其概念性角色為接收來自外部生成器或其他抽象系統提供的抽象「候選行為集合」。這些候選行為 `MUST` 概念性地遵循 `DCP_GeneratorOutput` 介面定義（參見 `Layer 0` 第 12.7 節）。
- **映射 `Layer 0`**：概念性地反映 `Layer 0` 中公理 M6（生成/判定分離）對「生成器」與「判定器」介面分離的要求。
- **概念性輸出**：一個包含多個抽象 `DCP_GeneratorOutput` 物件的集合。

3.4. Judgment Core (概念性組件)

此組件代表一個概念性角色，而非軟體模組。

- **概念性職責**：這是翾靈系統的核心概念。其概念性角色為接收「判定上下文物件」和「候選行為集合」，並對每個候選行為依據 DCP `Layer 0` 的 L0 公理（`A0`–`A5`）和 L1 操作原則（`R1`–`R8`）進行結構有效性的概念性判斷。
- **映射 `Layer 0`**：其設計旨在概念性地執行所有 `Layer 0` 公理和原則的檢查。尤其體現 M0 公理 M1（可行域存在與封閉）、M4（未知隔離）、M5（依存可回溯）和 M7（最小可驗收骨架）。
- **概念性輸出**：一個或多個抽象 `DCP_JudgmentOutput` 物件（參見 `Layer 0` 第 12.8 節），旨在概念性地標示每個候選行為的判定結果（`FEASIBLE` / `INFEASIBLE`）及相關解釋。

3.5. Audit Unit (概念性組件)

此組件代表一個概念性角色，而非軟體模組。

- **概念性職責**：其概念性角色為根據 `Layer 0` 中 `A0` 審計層的定義，在概念性判定過程中或最終概念性輸出形成後，對判定流程及其結果進行概念性一致性檢查。
- **映射 `Layer 0`**：直接概念性地體現 `Layer 0` 第 6 章中 `A0` 審計層的職責，以及 `Layer 0` 第 15 章「內部一致性判準」中的 `A0 審計一致性` 要求。
- **概念性輸出**：抽象 `DCP_AuditReport` 物件（參見 `Layer 0` 第 12.5 節）。

3.6. Projection Format Assembler (概念性組件)

此組件代表一個概念性角色，而非軟體模組。

- **概念性職責**：其概念性角色為將判定結果概念性地組裝為 `Layer 0` 第 10 章所定義的「規範性投影格式」。這包括概念性地建構人類可讀的 `output.content` 和機器可讀的 `output.skeleton`。
- **映射 `Layer 0`**：其設計旨在確保最終抽象輸出的結構和內容完全符合 `Layer 0` 第 10 章和 M0 公理 M7 的要求。
- **概念性輸出**：最終的抽象 `DCP_Output` 物件（參見 `Layer 0` 第 12.4 節）。

---

4. Abstract Operational Flow

(此流程為描述性，非規範性，不暗示執行順序或實作約束。以下流程僅表示邏輯依賴順序，而非時間序、實際執行流程或系統管線。)

本章概述翾靈系統內的概念性操作序列。此序列僅說明邏輯依存關係的順序，不規定運行時執行、控制流或系統編排。

4.1. Overview of the Conceptual Processing Flow

翾靈系統可被理解為透過一系列邏輯上循序漸進的概念性步驟，處理判定請求，從外部輸入的概念性接收開始，最終形成結構化的判定結果的概念性表示。

4.2. Request Ingestion and Parsing

- 在概念上，外部系統向翾靈系統概念性地發送 `DCP_Request`。
- 抽象的 `Request Parser` 概念性地接收請求，執行初步的概念性解析，並概念性地形成 `Xuanling_Task` 物件。

4.3. Context and Constraint Field Construction

- 抽象的 `Constraint Field Builder` 根據 `Xuanling_Task` 和此抽象架構的概念狀態，概念性地建構當前操作的 `Xuanling_JudgmentContext`（包含概念性定義的可行域 $\Phi(s)$ 和不可消元變量 $U(s)$）。

4.4. Candidate Behavior Intake

- 在概念上，外部生成器向抽象的 `Candidate Behavior Receiver` 概念性地提交一系列抽象 `DCP_GeneratorOutput` 物件（候選行為）。

4.5. Judgment Execution

- 抽象的 `Judgment Core` 依據 `Xuanling_JudgmentContext` 中的 $\Phi(s)$ 和 $U(s)$，對每個抽象 `DCP_GeneratorOutput` 候選行為進行逐一的概念性判斷。
- 此步驟嚴格遵循 DCP `Layer 0` 的 L0 公理和 L1 操作原則之概念性檢查。

4.6. Audit and Consistency Checking

- 抽象的 `Audit Unit` `MAY`（Level C 詮釋）或 `MUST`（Level A 詮釋）在概念性判定階段和/或最終概念性輸出形成後執行概念性一致性檢查。

4.7. Result Assembly and Projection

- 抽象的 `Judgment Core` 概念性地形成 `DCP_JudgmentOutput`。
- 抽象的 `Projection Format Assembler` 概念性地將判定結果（包括 `UNKNOWN` 標記、依存鏈、錨點衝突裁決等）組織成符合「規範性投影格式」的抽象 `DCP_Output` 物件。

---

5. Abstract Interface Definitions

本章定義翾靈系統作為 `Layer 1` 參考架構，其與外部實體概念性互動的抽象介面。這些介面是**符號性抽象**，描述語義角色，**而非 API 合約、軟體開發套件 (SDK) 或具體實作規格。**

5.1. Input Interfaces (概念性)

概念性地從外部抽象系統接收 `DCP_Request`（參見 `Layer 0` 第 12.1 節）和 `DCP_GeneratorOutput`（參見 `Layer 0` 第 12.7 節）。

5.2. Output Interfaces (概念性)

概念性地向請求實體返回 `DCP_Output`（參見 `Layer 0` 第 12.4 節）和 `DCP_AuditReport`（參見 `Layer 0` 第 12.5 節）。

5.3. Internal Conceptual Interfaces

概念性地定義抽象系統內部的互動介面，例如抽象的 `Judgment Core` 概念性地接受 `Xuanling_JudgmentContext` 和候選行為物件。此處稱之為「概念性互動介面（名稱僅供示意）」，具體細節將在未來的 `v0.x` 版本中概念性地定義。

5.4. Notes on Interface Semantics vs. APIs

此處對「介面」的描述表示抽象的概念性互動與架構模型內的數據交換點。它們說明了邏輯數據流和結構輸入/輸出，**不暗示任何特定的程式語言、數據序列化格式、通訊協定或運行時 API 合約。**

---

6. Mapping Between Xuanling Components and DCP Layer 0

本章提供翾靈系統（`Layer 1`）組件與 DCP（`Layer 0`）核心概念和要求之間概念性對應關係的高層次概覽。

6.1. Overview of Conceptual Mapping

翾靈系統的架構是 DCP `Layer 0` 的直接概念性詮釋。每個主要概念性組件及其概念性功能都被設計為明確映射到 DCP 中的一個基本公理、一組操作規則或一個核心結構要求。此設計旨在確保翾靈系統嚴格基於 `Layer 0` 的形式主義。

6.2. Component–Axiom Correspondence Table

下表概述了翾靈系統（`Layer 1`）主要概念性組件與 DCP（`Layer 0`）核心概念/要求之間的抽象映射關係。

Xuanling Component (概念性組件)	DCP `Layer 0` 核心概念/要求	Mapping Description
Request Parser	`DCP_Request`，L0 公理 A4（不可旁路錨點）	概念性地解析外部請求，識別初始條件與錨點之概念性角色。
Constraint Field Builder	L0 公理 M1（可行域存在與封閉）、M2（不可消元變量守恆）、M3（優先序作為交集選擇器）	根據抽象輸入建構判定的條件場與錨點配置之概念性角色。
Candidate Behavior Receiver	L0 公理 M6（生成/判定分離），`DCP_GeneratorOutput`	概念性地接收來自生成層的候選行為之概念性角色。
Judgment Core	所有 L0 公理（A0-A5, M1-M7），所有 L1 操作原則（R1-R8）	概念性地執行所有核心 DCP 判定邏輯之概念性角色。
Audit Unit	L0 A0 審計層，內部一致性判準	驗證判定流程與結果的結構一致性之概念性角色。
Projection Format Assembler	規範性投影格式（第 10 章），L0 公理 M7（最小可驗收骨架）	概念性地格式化最終判定結果以供概念性消費之角色。

6.3. Interpretation Notes on Mapping Boundaries

此映射為概念性與說明性。它表示功能對齊與理論派生，但**不規定**任何直接的一對一實作對應關係。單一的 Xuanling Component 可能概念性地詮釋多個 `Layer 0` 概念，或單一 `Layer 0` 概念可能在具體抽象系統中概念性地跨越多個 Xuanling Component。此映射的目的是旨在確保所有 `Layer 0` 原則在 Xuanling 架構中能找到概念性的歸屬與操作性詮釋。

---

7. Boundary Conditions and Non-Claims

本章明確界定本文件的邊界，並闡明翾靈系統（如本文所述）不指定、不保證、不聲稱的內容。這些非聲稱對於維護本參考架構的概念完整性與非規範性質至關重要。

7.1. What This Document Does Not Specify

本文件作為 `Layer 1` 概念架構，**不定義**：

- **具體實作方法或技術**：不涉及任何程式語言、框架、函式庫或演算法之具體選擇。
- **具體架構拓撲**：不推薦微服務、單體架構、分佈式系統等具體架構模式。
- **商業模式或商業策略**：不提供關於此類抽象系統如何產生收入或被市場化的指導。
- **部署策略或環境**：不詳述雲端平台、硬體要求或操作設定之具體細節。

- **詳細安全模型或機制**：除了 `Layer 0` 中概念性的 `Safety` 錨點外，具體安全措施不在本文概念性範圍內。

7.2. What This Document Does Not Guarantee

本文件**不作任何保證**，關於：

- **效能或效率**：不承諾運算速度、資源消耗或可擴展性之具體性能。
- **實作可行性**：儘管在概念上設計為可實作，但本文件不保證任何具體實作的實際可行性、成本或複雜性。
- **系統可用性或可靠性**：不作任何關於運行時間、容錯能力或操作穩健性之聲稱。

7.3. Non-Claims Regarding Implementation, Performance, or Governance

翾靈系統**不聲稱**：

- 是實作 DCP 的**唯一或最佳**概念架構。
- 賦予任何**內在的效能優勢**或操作效率。
- 本身具備**治理機制、法規遵循工具或決策權限**；它僅提供結構性判定之概念性框架。

7.4. Non-Exclusivity and Non-Authority Statement

本參考架構被呈現為對 DCP `Layer 0` 的一種可能、結構穩健的詮釋。它**不具獨佔性**，也不主張其權威高於其他可能也遵循 DCP 基本原則的概念模型。其價值在於其清晰性與 `Layer 0` 的一致性，而非任何優越性或強制採用的主張。

Appendix A — Projection Space (Informative, Non-Normative, Non-Binding)

(本附錄描述可能的詮釋性投影，不定義需求、承諾或實作。本附錄旨在探索此抽象架構在應用語境中可能被如何詮釋的示例。本附錄所列之投影示例不構成任何實作方向之偏好、承諾或預期，其存在僅用於說明理論可投影性。)

本附錄探討翾靈系統（`Layer 1`）超出其嚴格概念邊界之外的潛在詮釋與未來方向。它 serves as an informative, non-normative space for considering how the core architecture might conceptually relate to higher-level applications, deployments, and broader system contexts。

A.1. Possible Application Domain Interpretations

翾靈系統作為一個抽象的判定引擎，可在概念上投影到各種結構化、基於約束的驗證至關重要的領域。這些可能包括：

- **自動化政策合規**：概念性地驗證行動是否符合預定義的政策錨點。
- **智慧代理人防護欄**：概念性地確保 AI 代理人行為保持在定義的安全或倫理邊界內。

- **系統配置驗證**：概念性地檢查提議的系統變更是否符合架構約束。
- **法律或法規推理**：概念性地評估法律或法規詮釋與核心原則的一致性。

A.2. Possible Integration Pattern Interpretations

在概念上，翾靈系統可透過多種模式概念性地整合到更大的抽象系統中，例如：

- **Sidecar 模型**：概念性地與生成器並行運行，攔截並驗證其輸出。
- **Centralized Validation Service**：一個專門的服務之概念，供外部系統呼叫以進行判定。
- **Embedded Component**：一個與更廣泛的應用框架緊密集成的邏輯組件之概念。

A.3. Notes on Deployment Interpretations

在考慮部署時，一個概念性翾靈系統可能暗示：

- **Stateless Judgment Nodes**：每個判定請求都是獨立的概念。
- **Contextualized Judgment Services**：在請求之間維持約束場和不可消元變量的動態視圖之概念。
- **Scalable Validation Infrastructure**：旨在處理大量併發判定請求之概念。

A.4. Relationship to Higher-Level Systems and Frameworks

翾靈系統（`Layer 1`）為 `Layer 2` 及更高層次的應用和部署空間提供一個穩定的判定核心。

- `Layer 1` 定義了 `Layer 2` **如何**在概念上與基於 DCP 的判定抽象系統的抽象介面進行互動。
- `Layer 1` **不定義** `Layer 2` 應用將如何具體使用判定結果，也不指定其商業模式或使用者介面。

A.5. Open Questions and Directions for Future Exploration

作為 `v0.1` 概念草案，翾靈系統架構留下幾個方面供未來概念性探索。這些未在本文件中定義，純屬推測：

- **詳細內部數據流**：概念組件之間數據交換的具體細節之探索。
- **進階錯誤處理策略**：超越 `UNKNOWN` 和 `UNVERIFIED` 標記的機制之探索。
- **效能與可擴展性考量**：在具體設計中如何解決運算效能、資源消耗或執行效率之探索。
- **多層次整合點**：與更低層次（例如：底層運算資源）或更高層次（例如：應用層）抽象系統的詳細整合之探索。
- **具體安全機制**：儘管 `Safety` 是 `Layer 0` 中的隱含錨點，但具體安全措施仍有待概念性探索。