### TaxonomyOrchestrator: Lista Task di Configurazione Future

**Data:** 11 Luglio 2025

Questo documento elenca le operazioni pianificate per la configurazione esterna delle soglie del TaxonomyOrchestrator, rendendo il sistema più flessibile.

#### I. Modifiche a EvolutiveSystem.TaxonomyOrchestration/TaxonomyOrchestrator.cs (Già Discusse e da Applicare Subito)

Queste sono le modifiche che ti ho fornito precedentemente e che dovresti applicare per far compilare il codice con le funzionalità di base, usando ancora le costanti hardcoded per ora.

* Aggiungi il campo contatore: private int \_newMiuStringsDiscoveredCount = 0;
* Aggiungi la soglia (come costante per ora): private const int NEW\_MIU\_STRING\_THRESHOLD = 100;
* Aggiungi la sottoscrizione nel costruttore: \_eventBus.Subscribe<NewMiuStringDiscoveredEventArgs>(HandleNewMiuStringDiscoveredEvent);
* Aggiungi il metodo handler: private Task HandleNewMiuStringDiscoveredEvent(NewMiuStringDiscoveredEventArgs e) (come mostrato nel codice).
* Aggiungi la nuova condizione di trigger nel metodo CheckAndGenerateTaxonomy(): if (\_newMiuStringsDiscoveredCount >= NEW\_MIU\_STRING\_THRESHOLD)
* Reset del contatore nel blocco finally di CheckAndGenerateTaxonomy(): \_newMiuStringsDiscoveredCount = 0;

#### II. Task Futuri: Esternalizzazione delle Soglie di Configurazione

Queste sono le soglie che verranno trasformate in proprietà e caricate da una fonte esterna (es. MIUParameterConfigurator):

* RULE\_APP\_THRESHOLD (attualmente 500)
* SUCCESS\_SEARCH\_THRESHOLD (attualmente 20)
* FAILED\_SEARCH\_THRESHOLD (attualmente 50)
* ANOMALY\_THRESHOLD (attualmente 5)
* NEW\_MIU\_STRING\_THRESHOLD (attualmente 100)
* TIME\_THRESHOLD\_HOURS (attualmente 24.0)

**Dettagli dell'Implementazione Futura:**

1. **Trasformare le costanti in campi/proprietà:** Le attuali const int o const double diventeranno campi private readonly int o private readonly double (o proprietà se necessario per l'accesso esterno controllato).
2. **Aggiungere IMIUDataManager al costruttore di TaxonomyOrchestrator:** Se non è già presente, sarà necessario per leggere i parametri. (Nota: RuleTaxonomyGenerator ha già IMIUDataManager, quindi potremmo passarlo anche all'Orchestrator).
3. **Caricare i valori dal database:** Nel costruttore di TaxonomyOrchestrator, dopo l'inizializzazione, si leggeranno i valori corrispondenti dalla tabella MIUParameterConfigurator tramite IMIUDataManager.
4. **Gestione dei valori di default:** Se un parametro non è presente nel database, si utilizzerà un valore di default (quelli attuali).