### Updated Parameters for Orgo Customization

#### Purpose

The parameters for Orgo customization define the operational flexibility of the platform, ensuring it adapts seamlessly to diverse organizational needs. This updated version emphasizes metadata-driven workflows, replacing domain-specific parameters with a generalized framework while retaining options for detailed configurations.

### 1. Reactivity Time

Definition: Defines how quickly unresolved issues escalate or trigger responses.

* Improved Implementation: Reactivity time is now defined as a general attribute, applicable across all task types. Tasks specify reactivity\_time in the metadata to dynamically adjust escalation urgency.
* Examples:
  + "metadata": {"reactivity\_time": "2 hours", "subtype": "plumbing"}
  + "metadata": {"reactivity\_time": "5 minutes", "subtype": "critical"}
* Customization:
  + Set custom reactivity profiles for categories such as:
    - Safety: Immediate escalation.
    - Routine: Relaxed escalation times.

### 2. Transparency vs. Privacy

Definition: Balances information visibility across organizational hierarchies.

* Improved Implementation: Dynamic transparency levels allow varying visibility depending on task metadata and severity. Tasks with sensitive data use anonymization.
* Examples:
  + "metadata": {"visibility": "private", "category": "HR"}
  + "metadata": {"visibility": "public", "severity": "critical"}
* Customization:
  + Configure visibility based on context:
    - HR Complaints: Highly private.
    - Public Safety Alerts: Fully transparent.

### 3. Escalation Granularity

Definition: Determines the number and specificity of steps in the escalation process.

* Improved Implementation: Escalation steps are dynamically derived from metadata, enabling tailored escalation paths.
* Examples:
  + "metadata": {"escalation\_level": "detailed", "role": "manager"}
  + "metadata": {"escalation\_path": ["staff", "supervisor", "director"]}
* Customization:
  + Escalation paths vary by role or department, defined within workflow rules.

### 4. Review Frequency

Definition: Determines how often reviews are conducted to evaluate tasks and identify patterns.

* Improved Implementation: Adaptive review scheduling dynamically adjusts based on task volume or severity.
* Examples:
  + "metadata": {"review\_frequency": "weekly", "task\_count": 50}
  + "metadata": {"review\_frequency": "real-time", "severity": "critical"}
* Customization:
  + High-risk domains (e.g., compliance) default to frequent reviews.
  + Routine tasks (e.g., maintenance) use relaxed review schedules.

### 5. Notification Scope

Definition: Defines the range of individuals or teams notified for updates or escalations.

* Improved Implementation: Granular notification rules adapt based on metadata attributes such as task type, urgency, or recipient role.
* Examples:
  + "metadata": {"notification\_scope": "team", "priority": "high"}
  + "metadata": {"notification\_scope": "organization-wide", "type": "alert"}
* Customization:
  + Notify smaller teams for localized issues.
  + Notify the entire organization for public safety concerns.

### 6. Pattern Sensitivity

Definition: Detects recurring incidents or issues and flags them as patterns for escalation.

* Improved Implementation: Sensitivity levels are dynamic, varying by task category and severity.
* Examples:
  + "metadata": {"pattern\_sensitivity": "high", "category": "safety"}
  + "metadata": {"pattern\_sensitivity": "low", "category": "routine"}
* Customization:
  + High sensitivity for critical tasks (e.g., health and safety).
  + Low sensitivity for routine tasks (e.g., scheduling).

### 7. Severity Escalation Threshold

Definition: Determines the level of severity required for immediate escalation.

* Improved Implementation: Severity tagging uses metadata to classify tasks and decide escalation thresholds.
* Examples:
  + "metadata": {"severity": "critical", "threshold": "immediate"}
  + "metadata": {"severity": "minor", "threshold": "low"}
* Customization:
  + Automate escalation for critical tasks.
  + Delay escalation for minor or moderate tasks.

### 8. Logging and Traceability

Definition: Tracks the depth of recorded information for compliance and auditing.

* Improved Implementation: Logging depth is adjustable per task, ensuring compliance while optimizing storage.
* Examples:
  + "metadata": {"logging\_level": "detailed", "category": "compliance"}
  + "metadata": {"logging\_level": "minimal", "category": "routine"}
* Customization:
  + Compliance-heavy tasks use full traceability.
  + Routine tasks use minimal logs.

### 9. Automation Level

Definition: Controls the degree of automation applied to workflows.

* Improved Implementation: Automation profiles vary by task type and metadata, enabling tailored automation levels.
* Examples:
  + "metadata": {"automation\_level": "high", "category": "safety"}
  + "metadata": {"automation\_level": "low", "category": "HR"}
* Customization:
  + Fully automate repetitive tasks like notifications.
  + Retain manual oversight for sensitive workflows.

### 10. Data Retention Policy

Definition: Specifies how long records and logs are stored.

* Improved Implementation: Retention periods are category-specific, ensuring compliance with legal and organizational policies.
* Examples:
  + "metadata": {"retention\_period": "10 years", "category": "compliance"}
  + "metadata": {"retention\_period": "1 year", "category": "routine"}
* Customization:
  + Long-term retention for compliance-related tasks.
  + Short-term retention for routine or low-risk tasks.

### Benefits of Metadata-Driven Customization

1. Flexibility:
   * Adapt task behaviors dynamically based on metadata.
2. Scalability:
   * Add new task types or workflows without modifying core logic.
3. Efficiency:
   * Optimize task management through tailored configurations.
4. Compliance:
   * Ensure traceability and adherence to industry standards.

### Implementation Guidelines

1. Use metadata attributes in task definitions to specify reactivity time, transparency, escalation levels, and other parameters.
2. Define global settings in /config/organizations/default\_organization\_config.yaml and category-specific overrides in /config/workflows/.
3. Regularly review and refine parameters to align with evolving organizational needs.

### Conclusion

The updated parameters for Orgo Customization leverage metadata to create a dynamic, flexible system that adapts to diverse workflows and organizational priorities. By replacing domain-specific parameters with generalized rules, Orgo ensures scalability, maintainability, and seamless integration across industries.