Condition Evaluation System Documentation

Overview

This document describes the condition evaluation system that supports logical operators (***and***, ***or***), equality checks (***==***), and collection operators (***includeIn***, ***includeKey***, ***includeVal***). The system evaluates nested conditions against a provided context.

Basic Structure

Conditions are JSON objects with one of these structures:

- Logical operator (and, or) containing an array of sub-conditions
- Comparison operator (***==***, ***includeIn***, ***includeKey***, ***includeVal***) containing arguments

Operators

1. Logical Operators

AND Operator

Evaluates to true if ALL sub-conditions are true.

Structure:

```
{
  "and": [
    { /* condition 1 */ },
    { /* condition 2 */ }
  ]
}
```

Example:

```
{
  "and": [
    { "==": ["$source", "calculator"] },
    { "==": ["$status", "active"] }
  ]
}
```

OR Operator

Evaluates to true if ANY sub-condition is true

Structure:

```
{
  "or": [
    { /* condition 1 */ },
    { /* condition 2 */ }
  ]
}
```

Example:

```
{
  "or": [
    { "==": ["$user.role", "admin"] },
    { "==": ["$user.role", "supervisor"] }
  ]
}
```

2. Comparison Operators

Equality (==)

Checks if two values are equal.

Structure:

```
{ "==": ["leftValue", "rightValue"] }
```

Examples:

```
{ "==": ["$productId", "1001"] }
{ "==": ["$user.name", "John Doe"] }
```

3. Collection Operators

includeIn (Arrays)

Checks if a value exists in an array.

Structure:

```
{ "includeIn": ["value", "array"] }
```

Example:

```
{ "includeIn": ["$agentId", "$allowedAgents"] }
```

includeKey (Objects)

Checks if a key exists in an object.

Structure:

```
{ "includeKey": ["key", "object"] }
```

Example:

```
{ "includeKey": ["status", "$user"] }
```

includeVal (Objects)

Checks if a value exists in an object's values.

Structure:

```
{ "includeVal": ["value", "object"] }
```

Example:

```
{ "includeVal": ["premium", "$user.subscriptions"] }
```

Variable Resolution

Variables in the context are referenced with $ prefix:

- ***$user.name resolves to context.user.name***
- ***$status resolves to context.status***

Nested Conditions Examples

Example 1: Complex AND/OR

```
{
  "and": [
    { "==": ["$source", "dashboard"] },
    {
      "or": [
        { "==": ["$user.role", "admin"] },
        { "==": ["$user.role", "editor"] }
      ]
    },
    { "includeIn": ["$user.id", "$allowedUsers"] }
  ]
}
```

Example 2: Multiple Collection Checks

```
{
  "and": [
    { "includeKey": ["email", "$user"] },
    { "includeVal": ["active", "$user.status"] },
    {
      "or": [
        { "includeIn": ["$department", "$techDepartments"] },
        { "includeIn": ["$department", "$specialDepartments"] }
      ]
    }
  ]
}
```

Example 3: Deeply Nested

```
{
  "and": [
    { "==": ["$environment", "production"] },
    {
      "or": [
        {
          "and": [
            { "includeKey": ["subscription", "$user"] },
            { "==": ["$user.subscription.level", "premium"] }
          ]
        },
        { "includeIn": ["$user.id", "$vipUsers"] }
      ]
    },
    { "includeVal": ["approved", "$user.permissions"] }
  ]
}
```

Error Handling=>

The system provides detailed warnings when:

- ***Variables are missing from context***
- ***Operators receive invalid types***
- ***Conditions fail evaluation***

Best Practices

1. ***Use and/or to group related conditions***
2. ***Put most likely-to-fail conditions first for better performance***
3. ***Use includeIn for arrays, includeKey/includeVal for objects***
4. ***Keep conditions as shallow as possible for readability***

Complete Example With Context

Condition:

```
{
  "and": [
    { "==": ["$request.source", "mobile"] },
    {
      "or": [
        { "includeIn": ["$user.id", "$betaTesters"] },
        { "==": ["$user.tier", "gold"] }
      ]
    },
    { "includeKey": ["emailVerified", "$user"] }
  ]
}
```

Context:

```
{
  "request": { "source": "mobile" },
  "user": {
    "id": "usr_123",
    "tier": "gold",
    "emailVerified": true
  },
  "betaTesters": ["usr_456", "usr_789"]
}
```

Result: true (all conditions are satisfied)

----------The End----------