以下是《数智员工开发手册》中关于**向导配置**和**流程配置**中各种参数的含义和使用方法的总结，内容基于手册中的描述，整理为简洁、结构化的说明，便于快速理解和应用。

---

# 一、向导配置中的参数含义与使用方法

向导配置用于定义用户交互界面，收集输入参数，并关联执行的流程。配置采用YAML格式，主要参数如下：

## 1. 顶层参数

- `type`：
  - **含义**：定义向导的数据结构类型，默认值为`object`。
  - **使用方法**：固定设置为`object`，无需修改，用于表示向导配置是一个对象。
  - **示例**：`type: object`
- `name`：
  - **含义**：向导的名称，用于标识向导的用途，显示在界面上。
  - **使用方法**：输入描述性名称，便于用户理解向导功能。手册中提到"目前看起来没啥用"，但建议规范填写。
  - **示例**：`name: 民意速办分析报告`
- `required`：
  - **含义**：指定必填的参数字段列表。
  - **使用方法**：列出必须输入的参数字段名称（在`properties`中定义）。若用户未填写，系统会提示错误。
  - **注意**：若设置`default: ''`或`default: null`，则允许空值提交；否则必填字段为空时无法提交。
  - **示例**：

    ```yaml
    required:
      - activity_name
    ```

- `properties`：
  - **含义**：定义向导中具体的输入字段及其属性。
  - **使用方法**：为每个输入字段配置类型、标题、界面样式、默认值等，详见下文参数类型。
  - **示例**：

    ```yaml
    properties:
      activity_name:
        type: string
        title: 活动名称
    ```

- `flows`：
  - **含义**：关联向导执行的流程文件编号。
  - **使用方法**：设置为流程配置文件名（如`de-gov-duanwutest.yaml`），并通过`ui:hidden: true`隐藏在界面上。

- **示例**：

```
flows:
  type: string
  ui:hidden: true
  default: de-gov-duanwutest.yaml
```

## 2. 参数类型及配置

`properties`中的每个字段支持多种类型，用于定义用户输入方式。以下是常见类型及其配置方法：

- **选择上传文件（单选）**：
  - **含义**：允许用户从知识库或主题中选择一个文件。
  - **配置**：
    - `type: string`：表示选择的文件ID为字符串。
    - `ui:customize: select`：使用下拉选择控件。
    - `enums`：定义文件列表的获取方式，通常通过API调用。
    - `payload`：设置API请求参数，如限制文件数量（`limit`）。
  - **示例**：

```
detail_id:
  type: string
  title: 明细数据
  ui:customize: select
  enums:
    fetch:
      type: post
      api: '/api/de-connect/{{current.employee.id}}/topics/docs'
      payload: '{ "object": { "topic": "{{current.topic.id}}" } }'
    label: filename
    value: id
  err:options:
    required: '请输入工单明细文件'
```

- **选择上传文件（多选）**：
  - **含义**：允许用户选择多个文件。
  - **配置**：与单选类似，但`type: array`，并添加`items`和`multiple: true`。
  - **示例**：

```
document:
  type: array
  title: 成果类文档
  ui:customize: select
  items:
    type: string
  enums:
    fetch:
      type: post
      api: '/api/de-connect/{{current.employee.id}}/topics/docs'
```

```
      payload: '{ "object": { "topic": "{{current.topic.id}}" },
"limit": 20 }'
    label: filename
    value: id
    multiple: true
```

- **输入框**:
  - **含义**: 单行文本输入。
  - **配置**:
    - `type: string`: 输入为字符串。
    - `ui:options: type: input`: 使用单行输入框。
  - **示例**:

```
activity_name:
  type: string
  title: 活动名称
  ui:options:
    type: input
  err:options:
    required: '请输入活动名称'
  default: ''
```

- **多行文本框**:
  - **含义**: 多行文本输入，适合长文本。
  - **配置**:
    - `ui:options: type: textarea`: 使用多行输入框。
    - `rows`: 设置输入框高度。
    - `width`: 设置输入框宽度。
    - `placeholder`: 输入框提示文字。
  - **示例**:

```
otherrequest:
  type: string
  title: 活动其他要求
  description: 输入其他要求
  ui:options:
    type: textarea
    rows: 2
    width: 100%
    placeholder: 用来更准确的引导AI生成活动信息
  err:options:
    required: '请输入活动其他要求'
  default: ''
```

- **日期**:
  - **含义**: 选择日期。
  - **配置**:
    - `format: date`: 指定日期格式。

- **示例**：

```
activity_date:
  type: string
  title: 活动日期
  format: date
  err:options:
    required: '请选择活动日期'
  default: '2024-06-09'
```

- **时间**：
  - **含义**：选择时间。
  - **配置**：
    - `format: time`：指定时间格式。
  - **示例**：

```
activity_time1:
  type: string
  title: 活动开始时间
  format: time
  err:options:
    required: '请选择活动开始时间'
  default: '14:00:00'
```

- **整数**：
  - **含义**：输入整数。
  - **配置**：
    - `format: int`：指定整数格式。
  - **示例**：

```
budget:
  type: string
  title: 预算
  format: int
  err:options:
    required: '请输入预算'
  default: ''
```

- **下拉框**：
  - **含义**：提供固定选项或动态列表供用户选择。
  - **配置**：
    - `enum`：定义选项的值。
    - `enumNames`：定义选项显示的名称。
    - 动态列表可通过`{context#variable}`引用上下文变量。
  - **示例**（固定选项）：

```yaml
theme:
  type: number
  title: 节假日
  err:options:
    required: '请选择举办活动节假日'
  default: 0
  enum:
    - 0
    - 1
  enumNames:
    - 春节
    - 元宵节
```

- **示例**（动态列表）：

```yaml
issuename:
  type: string
  title: 问题
  description: 请选择一个典型问题
  enum: "{context#issuelist}"
  ui:
    options:
      placeholder: 请选择一个典型问题
    customize: select
  err:options:
    required: '请选择一个典型问题'
```

- **通用文档定义**：
  - **含义**：引用通用文档（如模板）的ID。
  - **配置**：
    - `ui:hidden: true`：隐藏在界面上。
    - `default`：指定文档ID。
  - **示例**：

```yaml
docx_tpl_id:
  type: string
  ui:hidden: true
  default: 402880ad90a09a880190a48c47ae3346
```

## 3. 使用方法总结

- **创建向导**：
  1. 在配置平台（`https://apps-dev1.aquaintelling.com/tutor-console`）进入“数智员工 -> 向导配置”。
  2. 点击“新增向导”，输入向导编号和名称，粘贴YAML配置。
  3. 确保`flows`字段指向正确的流程文件。
- **复制向导**：
  - 复制现有向导后，修改向导编号，避免覆盖原配置。
- **调试**：
  - 检查YAML格式是否正确（可复制到Notepad++，设为YAML语言检查）。

- 若保存提示“服务器开小差”，检查缩进或日志（`tail -200 ./logs/de-factory/running.log`）定位错误。

---

## 二、流程配置中的参数含义与使用方法

流程配置定义了数智员工执行的逻辑，包含插件调用和数据处理。配置同样采用YAML格式，主要参数如下：

### 1. 流程信息（新增流程时填写）

- **流程编号**：
  - **含义**：唯一标识流程，便于维护和查询。
  - **使用方法**：输入清晰的编号，如`summary_flow_001`。
  - **示例**：`流程编号: summary_flow_001`
- **流程名称**：
  - **含义**：描述流程的意图，便于理解。
  - **使用方法**：输入简洁的名称，如“项目总结生成”。
  - **示例**：`流程名称: 项目总结生成`
- **流程等级**：
  - **含义**：手册未详细说明，可能是优先级或分类。
  - **使用方法**：待补充，通常按默认值填写。
  - **示例**：`流程等级: （待补充）`
- **全局/角色/实例**：
  - **含义**：手册未明确，可能是流程作用范围或权限设置。
  - **使用方法**：根据项目需求填写，需进一步确认。

### 2. 流程步骤参数

每个流程步骤定义一个插件的调用，主要参数包括：

- `code`：
  - **含义**：指定使用的插件，格式为`插件名`或`插件名#子功能`。
  - **使用方法**：根据需求选择插件，如`declared.operations`、`llmx.predict.chat`等。
  - **示例**：

    ```
    code: declared.operations
    ```

- `name`：
  - **含义**：步骤的描述性名称，便于调试和维护。
  - **使用方法**：填写步骤功能，如“定义大模型”。
  - **示例**：

    ```
    name: 定义大模型
    ```

- `function_desc`：
  - **含义**：描述插件的用途，提高可读性。
  - **使用方法**：在`props`中添加`design`字段（UUID）并填写用途。
  - **示例**：

```yaml
function_desc: 定义大模型
props:
  design: 2b352afb089049bca92180956279efaf
```

- `args`：
  - **含义**：插件的输入参数，控制插件行为。
  - **使用方法**：根据插件类型配置具体参数，如`payload`、`messages`等。
  - **示例**（大模型选择）：

```yaml
args:
  payload:
    llm:
      name: gpt-4o
      temperature: 0.3
      slaves:
        - name: qwen2-72b-instruct-gptq-int8
```

- `sinks`：
  - **含义**：定义插件输出的处理方式，包括存储到上下文、记录日志或输出到前端。
  - **使用方法**：
    - `context`：将结果存储到上下文变量，指定变量名。
    - `logging: true`：记录执行日志。
    - `reply: pending`：将结果输出到前端。
  - **示例**：

```yaml
sinks:
  context:
    - name: result
  logging: true
  reply: pending
```

### 3. 常用插件参数

以下是手册中提到的主要插件及其参数使用方法：

- **终止操作（abort.operations）**：
  - **用途**：终止整个流程。
  - **参数**：无参数，直接调用。
  - **示例**：

```
- code: abort.operations
  name: 终止流程
```

- **声明操作（declared.operations）**：
  - **用途**：定义大模型或其他配置。
  - **参数**：
    - `payload.llm.name`：模型名称，如`gpt-4o`。
    - `payload.llm.temperature`：控制生成随机性（0-1）。
    - `payload.llm.slaves`：备用模型列表。
  - **示例**：

```
- code: declared.operations
  args:
    payload:
      llm:
        name: gpt-4o
        temperature: 0.3
        slaves:
          - name: qwen2-72b-instruct-gptq-int8
```

- **对话模型调用（llmx.predict.chat）**：
  - **用途**：调用大模型生成文本。
  - **参数**：
    - `messages`：包含`system`（系统提示）和`human`（用户输入）消息。
    - `template`：使用`{context#variable}`引用上下文变量。
  - **示例**：

```
- code: llmx.predict.chat
  args:
    messages:
      - type: system
        template: 生成项目总结
      - type: human
        template: 项目名称：{context#project_name}
```

- **输出到前端（reply.operations）**：
  - **用途**：将内容以Markdown格式输出到前端。
  - **参数**：
    - `pending`：输出内容，支持Markdown和上下文变量。
    - `steps`：输出到会话签名。
    - `links`：输出带链接内容。
  - **示例**：

```
- code: reply.operations
  args:
    pending: "## <center>项目总结\n\n{context#summary_result}"
  reply: pending
```

- **代码执行（sources.execution）**：
  - **用途**：执行自定义Python代码。
  - **参数**：
    - `source`：Python代码，支持上下文变量替换。
    - `locals`：定义上下文变量映射。
  - **示例**：

```
- code: sources.execution
  args:
    source: |-
      text = "{context#recall_text}"
      return text
  sinks:
    context:
      - name: result
```

- **互联网搜索（internets.search.bing/baidu）**：
  - **用途**：执行网络搜索。
  - **参数**：
    - `keywords`：搜索关键字和范围。
    - `freshness`：搜索时间范围。
    - `kwargs`（百度特有）：如`wd`（关键字）、`gpc`（时间戳）。
  - **示例**：

```
- code: internets.search.bing
  args:
    keywords: "党政新闻 site:news.cn"
    freshness: "{context#date.date2}..{context#date.date1}"
  sinks:
    context:
      type: append-props
      name: webs
```

## 4. 持久化与格式化

- **持久化（sinks.context）**：
  - **含义**：将插件结果存储到上下文变量。
  - **方式**：
    - `set-props`：直接设置变量。
    - `append-props`：追加到数组。
    - `extend-props`：合并数组。
  - **示例**：

```yaml
sinks:
  context:
    - name: result
      type: set-props
```

- **格式化（sinks.format）**：
  - **含义**：对插件结果进行格式转换，如JSON、YAML、字符串等。
  - **方式**：
    - `format-json-object`：字符串转JSON对象。
    - `format-json-array`：字符串转JSON数组。
    - `text-split`：字符串切割。
  - **示例**：

```yaml
sinks:
  format:
    - type: format-json-object
  context:
    - name: json_result
```

### 5. 使用方法总结

- **创建流程**：
  1. 在配置平台进入“数智员工 -> 流程配置”，点击“新增流程”。
  2. 输入流程编号和名称，粘贴YAML配置。
  3. 通过“配置”页面添加插件，或直接编辑YAML。
- **复制流程**：
  - 复制后修改流程编号，避免覆盖原流程。
- **调试**：
  - 使用`logging: true`记录日志，查看`/logs/de-factory/running.log`。
  - 若报错，检查YAML缩进或日志定位问题插件。
- **调测**：
  - 流程需结合场景或向导测试，无法单独调试。
  - 使用`sources.execution`打印上下文变量验证传递。

---

## 三、通用注意事项

1. **YAML格式**：
   - 确保缩进正确（2空格），否则保存会报错。
   - 使用Notepad++或日志定位格式错误。
2. **日志调试**：
   - 使用`logger.info`在`sources.execution`中打印变量值。
   - 日志路径：`/home/de/logs`（开发环境）或`/data/nfs/*/de-flows`（测试环境）。
3. **变量引用**：

- 上下文变量格式：`{context#variable_name}`。
- 预置变量（如`company_id`）可直接使用，无需`context`前缀。

4. **异常处理**：
   - 部分插件支持忽略异常（`except`），但如`internets.search.bing`暂不支持。
5. **前端输出**：
   - 使用`reply.operations`的`pending`、`steps`或`links`控制输出位置。
   - Markdown格式支持表格、CSS样式等。

---

## 示例总结

以下是一个简单的向导和流程配置，展示参数使用：

```yaml
# 向导配置
type: object
name: 项目总结
required:
  - project_name
properties:
  project_name:
    type: string
    title: 项目名称
    ui:options:
      type: input
    err:options:
      required: '请输入项目名称'
  flows:
    type: string
    ui:hidden: true
    default: project_summary_flow.yml

# 流程配置
- code: declared.operations
  name: 定义模型
  args:
    payload:
      llm:
        name: gpt-4o
- code: llmx.predict.chat
  name: 生成总结
  args:
    messages:
      - type: system
        template: 生成项目总结：{context#project_name}
  sinks:
    context:
      - name: summary
- code: reply.operations
  name: 输出结果
  args:
    pending: "{context#summary}"
  reply: pending
```

此总结涵盖了向导和流程配置中核心参数的含义与使用方法，结合手册中的案例和说明，适用于快速上手开发。如需更详细的插件配置或特定案例，请进一步说明！