ALIBABA CLOUD

阿里云

Serverless 工作流

服务集成

文档版本：20211130

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.集成简介

Serverless工作流支持与多个云服务集成，即将其他服务作为任务步骤的执行单元。服务集成方式由FDL语言表达，在**任务**步骤中，您可以使用 resourceArn 来定义集成的目标服务，使用 pattern 定义集成模式。本文主要介绍了服务集成的相关内容，包括集成模式、上下文对象和已集成云服务。
关于可使用的云服务列表，请参见已集成云服务。

集成模式

目前Serverless工作流支持三种不同的集成模式。

- 请求响应模式：调用第三方服务，在Serverless工作流获得HTTP响应后进入下一个步骤。默认模式是请求响应模式。
  在FDL步骤中使用 resourceArn 描述目标服务，使用 pattern: requestResponse （可选，若无该参数则为默认模式）描述服务集成模式。该模式下Serverless工作流将在调用接口返回后立刻进入到下一步骤的执行。以子流程功能（Serverless工作流作为被集成服务）为例。

  ```
  version: v1
  type: flow
  steps:
   - type: task
     name: testSubflow
     resourceArn: acs:fnf:::flow/flowABC #描述子流程。
     pattern: requestResponse #描述服务集成模式：默认（请求响应）模式。
   - type: pass
     name: dummy
  ```

  该示例展示了当 testSubflow 步骤执行时会触发一个 flowABC 流程。触发成功后，将进入下一步骤 dummy ， flowABC 流程可能仍然正在执行。
- 同步模式：通常这类服务提供了异步执行接口，Serverless工作流调用异步接口成功后会等待，直到相关任务完成并获得了执行结果，Serverless工作流才会继续执行下一个步骤。
  对于某些集成服务，Serverless工作流可以等到该服务的任务运行完成后再进入下一个步骤。一般这类服务为执行某个任务提供了任务启动（异步）的接口，需要提交任务并等待任务执行完成再进行下一步骤。
  在FDL步骤中使用 resourceArn 描述目标服务，使用 pattern: sync 描述服务集成模式。以子流程功能（Serverless工作流作为被集成服务）为例。

  ```
  version: v1
  type: flow
  steps:
   - type: task
     name: testTask
     resourceArn: acs:fnf:::flow/flowABC #描述子流程。
     pattern: sync #描述服务集成模式：同步。
   - type: pass
     name: dummy
  ```

  该示例展示了当该步骤执行时会触发一个Serverless工作流流程。触发成功后将等待该流程的执行结果，执行完成后将进入下一步骤。当 testTask 步骤执行时会触发一个 flowABC 流程。触发成功后，将等待该流程的执行结果，执行完成后才进入下一步骤 dummy ， flowABC 流程已经执行完成。
- 等待回调模式：调用第三方服务并传入任务令牌，Serverless工作流将进入等待，直到您手动使用该令牌通知流程执行结果后才会继续执行下一个步骤。
  回调任务将使当前流程在任务调度点暂停，直到收到任务令牌的回调指令。在FDL步骤中使用 resourceArn 描述目标服务，使用 pattern: waitForCallback 描述服务集成模式。以子流程功能（Serverless工作流作为被集成服务）为例。

```
version: v1
type: flow
steps:
 - type: task
   name: testSubflow
   resourceArn: acs:fnf:::flow/flowABC #描述子流程。
   pattern: waitForCallback #描述服务集成模式：等待回调。
 - type: pass
   name: dummy
```

该示例展示了当 testSubflow 步骤执行时会触发一个 flowABC 流程。触发成功后，将暂停流程执行，等待回调（通过 ReportTaskSucceed 或 ReportTaskFailed API）。在收到回调请求并处理完成后，流程将进入下一步骤 dummy 。 flowABC 流程可能已经执行完成，也可能还在执行。其中回调是由您发起。

上下文对象

上下文对象是流程执行实例的内部JSON对象，其中包含了关于执行、步骤的相关信息。该对象提供外部访问方式，在inputMappings中您可将context对象映射到具体变量中来实现访问。目前开放的context对象结构如下所示。

```
"context": {
  "flow": {
   // 本流程的唯一标识符id和流程名称、字符串类型。
    "id": "val1",
    "name "val2",
  },
  "execution": {
     // 本执行的名称。
    "name": "val3"
  },
  "step": {
    // 本步骤的名称。
    "name": "val4"
   // 本步骤的事件ID。
    "eventId": "val5"
   // 当前循环的次数，在循环步骤（foreach）下可以使用。
    "IterationIndex": "val6",
  },
  "task": {
   // 本步骤的标识符，为一个字符串，在回调模式（waitForCallback）下可以使用。
    "token": "val7",
  },
}
```

使用方式：例如集成Serverless工作流自身服务，需要在子流程中获取调用其父流程的相关信息，并获取调用步骤的taskToken用于回调，您可以通过以下方式获取这两个字段。

```
...
  inputMappings:
 - target: current_flow_name
   source: $context.flow.name
 - target: current_execution_name
   source: $context.execution.name
 - target: current_step_task_token
   source: $context.task.token
```

已集成云服务

方案	请求响应（requestResponse）	同步（sync）	等待回调（waitForCallback）
函数计算（FC）	支持	不支持	不支持
消息服务队列（MNS Queue）	支持	不支持	支持
消息服务主题（MNS Topic）	支持	不支持	支持
Serverless工作流（SWF）	支持	支持	支持

2.集成Serverless工作流

集成Serverless工作流功能让您可以方便地在一个流程中执行另一个流程，本文介绍了集成Serverless工作流的使用场景、集成模式、对象说明和子流程的输入输出规则。

使用场景

在以下场景中可以考虑使用Serverless工作流集成：

- 降低流程的复杂度，将一个流程分解为多个流程。
- 让流程更容易被复用。您可以将通用的一些步骤放在一个流程中，被其他多个流程复用。
- 突破当前单流程的某些限制，例如当前单个流程最多event个数限制（默认5000）和最长执行时间限制（最长1年）。
- 对流程中的一些控制步骤做错误处理。例如可以将并行步骤做成子流程，在主流程中对子流程执行错误做错误处理。

集成模式

Serverless工作流集成支持三种模式，分别是请求响应（requestResponse）模式、同步（sync）模式和等待回调（waitForCallback）模式。

- 请求响应（requestResponse）模式
  该模式下，主流程将在启动子流程成功后立刻开始执行下一步骤。流程定义如下。

```
version: v1
type: flow
steps:
 - type: task
   name: fnfInvoke
   resourceArn: acs:fnf:::flow/subflow_demo_child
   pattern: requestResponse # 可省略，默认。
   inputMappings: # 如无inputMappings，将按默认映射规则将主流程参数作为子流程的Input。
    - target: childName # 用于在service中设定发起的子流程的执行名称。
      source: $input.childName
   serviceParams: # 集成Serverless工作流的服务参数，该参数可省略。如省略本参数，我们将使用随机字符串作为本次执行的名称，使用InputMappings对应的参数作为子流程的输入。
     Input: $ # 用映射后的input作为启动子流程的输入参数。
     ExecutionName: $.childName # 如果在serviceParams中使用变量, 请确保该变量存在于inputMappings中。
```

- 同步（sync）模式
  该模式下，主流程将启动一个子流程并等待子流程执行完成后再进入下一步骤。流程定义如下。

```
version: v1
type: flow
steps:
 - type: parallel
   name: parallelTask
   branches:
    - steps:
     # 本步骤展示使用sync模式集成Serverless工作流，使用inputMappings作为子流程的输入，并通过主流程的输入来动态指定子流程的执行名称。
      - type: task
        name: fnfSync
        resourceArn: acs:fnf:::flow/subflow_demo_child
        pattern: sync
        inputMappings: # 如无inputMappings，将按默认映射规则将主流程参数作为子流程的Input。
         - target: childSyncName # 发起的子流程的执行名称。如您需指定子流程的执行名称，请按本示例所示对期望的执行名称进行inputMapping，并在serviceParams中使用。
           source: $input.childSyncName。
        serviceParams: # 集成Serverless工作流的服务参数。
         Input: $ # 用映射后的inputMappings作为启动子流程的输入参数. 除非您很明确使用其他方式指定Input时的行为及语法，否则建议您使用我们提供的此种方式。
         ExecutionName: $.childSyncName # 如果在serviceParams中使用变量, 请确保该变量存在于inputMappings中。
```

- 等待回调（wait ForCallback）模式
  该模式下，主流程将启动一个子流程并进入暂停状态，直到收到回调通知。流程定义如下。

```
version: v1
type: flow
steps:
 - steps:
  # 本步骤展示使用waitForCallback模式集成Serverless工作流，使用inputMappings作为子流程的输入，并通过主流程的输入来动态指定子流程的执行名称。
  - type: task
    name: fnfWaitForCallback
    resourceArn: acs:fnf:::flow/subflow_demo_child
    pattern: waitForCallback
    inputMappings: # 如无inputMappings，将按默认映射规则将主流程参数作为子流程的Input。
     - target: task_token # 为确保子流程中可以使用回调，请自定义名称对task_token进行显示映射。
       source: $context.task.token  # 从context对象中获取表示该任务的令牌（task token）。
     - target: childCallbackName
       source: $input.childCallbackName
    serviceParams: # 集成Serverless工作流的服务参数。
     Input: $ # 用映射后的inputMappings作为启动子流程的输入参数。
     ExecutionName: $.childCallbackName # 如果在serviceParams中使用变量, 请确保该变量存在于inputMappings中。
```

上下文对象说明

在子流程集成模式中，您可以将 `$context.execution.name` 及 `$context.flow.name` 变量传递给子流程，用于在子流程中识别启动它的父流程。在 `waitForCallback` 模式下， `$context.task.token` 将被使用，用来向子流程传递父流程的运行标识符来实现回调。

子流程的输入输出规则

- 请求响应模式

子流程的输入来源于任务的输入，您可以在子流程中通过 $Input 来获取该输入。
请求响应模式子流程的启动信息（ StartExecution API的响应）将会被作为输出，而子流程自身输出将被主流程忽略。在启动子流程后，我们默认提供了 $local.ExecutionName 、 $local.FlowName 、 $local.RequestId 三个子流程启动信息。如果您需要对其进行额外处理，可在主流程对应步骤中使用 outputMappings 进行映射。

```
- type: task
  pattern: requestResponse
  ...
  outputMappings: # requestResponse模式可获取参数：$local.ExecutionName, $local.FlowName, $local.RequestId。
    - target: subflow_children_request_id
      source: $local.RequestId # 发起子流程的requestID。
    - target: subflow_children_exec_name
      source: $local.ExecutionName # 发起的子流程的执行名称。
    - target: subflow_children_flow_name
      source: $local.FlowName # 发起的子流程的流程名称。
```

- 同步模式
  同步模式中，子流程的输入来源于任务的输入，您可以在子流程中通过 $Input 来获取该输入。
  子流程的输出（ DescribeExecution API响应中的 Output ）将被传递回主流程作为主流程该步骤的输出，您可以在主流程的后续步骤中使用该输出。如果您需要对子流程的输出进行额外处理，可以使用 outputMappings 对输出进行映射。
- 等待回调模式
  等待回调模式中，子流程的输入来源于任务的输入，您可以在子流程中通过 $Input 来获取该输入。
  回调的输出数据将作为主流程该步骤的输出。在 ReportTaskSucceeded 接口中，传入的 Output 参数对应的值为主流程该步骤的输出。在 ReportTaskFailed 接口中，传入的Error及Cause参数及对应的值将作为主流程该步骤的输出。如果您需要对子流程的输出进行额外处理，可以使用 outputMappings 对输出进行映射。

3.集成函数计算异步调用

在Serverless工作流的任务步骤中，异步调用函数计算可以更灵活地适应一些任务场景，例如长时任务、人工审核等，帮助您避免限流等错误，同时可以简化流程中的错误处理和重试逻辑。本文介绍Serverless工作流集成函数异步调用的背景信息、集成模式和使用示例等。

前提条件

您已完成以下操作：

- （可选）开启异步调用
  当您的函数开启了有状态异步调用时，您还可以在任务执行过程中停止函数实例，并获得对整个执行流程更细粒度的可观测性。
- 授予函数计算访问Serverless工作流的权限，确保当函数异步执行完成后可以正常回调Serverless工作流继续执行后续流程。权限策略如下所示：

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "fnf:ReportTaskSucceeded",
        "fnf:ReportTaskFailed"
      ],
      "Resource": [
        "*"
      ]
    }
  ],
  "Version": "1"
}
```

关于授权的详细信息，请参见授予函数计算访问其他云服务的权限。

背景信息

默认情况下，Serverless工作流编排函数计算任务是采用同步调用实现的任务流程，即Serverless工作流调用函数计算时会等待函数执行完成后同步获得返回的输出，才可以进入到下一个调用的任务。流程定义如下：

```
version: v1
type: flow
steps:
 - type: task
   name: mytask
   resourceArn: acs:fc:{region}:{account}:services/{serviceName}.{qualifier}/functions/{functionName}
```

默认情况下，采用同步调用函数计算时可能会存在以下问题：

- 由于函数计算存在资源调用限制，即函数计算的阿里云账号（主账号）在单个地域内默认的按量实例上限数为300。
  当您使用Serverless工作流调用函数时，也存在一些其他的函数调用，这些调用会共享相同的配额，因此可能会造成限流等报错。从而您需要在Serverless工作流的流程中定义较复杂的重试策略，但也无法保证流程是否会执行成功。
- 当部分任务的执行时间较长，采用同步调用时，Serverless工作流会与函数计算建立长链接，由于网络波动可能会导致一些非预期错误。

Serverless工作流支持函数计算的异步调用与集成模式的结合，帮助您解决以上问题，同时也可以满足以下需求：

- 在某些场景下，无需等待任务执行完成后再执行后续步骤。
- 在流程执行的过程中，如果遇到非预期错误，可以忽略该步骤的函数调用，直接执行后续操作。

适用场景

任务步骤对函数计算异步调用功能的集成有请求响应（requestResponse）模式、同步（sync）模式和回调（waitForCallback）模式，分别适用于不同场景。

集成模式	模式（pattern）参数	适用场景

请求响应模式（默认模式）	`requestResponse`
	流程示例如下：
	`- type: task`
	`...`
	`pattern: requestResponse`
	`serviceParams:`
	`InvocationType: Async`
		长时间执行的任务、无需了解任务执行结果。

同步模式	`sync`
	流程示例如下：
	`- type: task`
	`...`
	`pattern: sync`
	`serviceParams:`
	`InvocationType: Async`
		长时间执行的任务、可能被限流的任务。

回调模式	`waitForCallback`
	流程示例如下：
	`- type: task`
	`...`
	`pattern: waitForCallback`
	`serviceParams:`
	`InvocationType: Async`
		任务在执行到某些情况下，可以继续后续的步骤，例如人工审核。

任务服务参数

下文示例中，Serverless工作流使用函数计算作为任务节点，通过 `resourceArn` 指定目标服务为函数计算，并通过 `serviceParams` 字段指定调用函数计算的特殊参数。`serviceParams` 支持以下参数信息：

- `InvocationType`：表示函数计算中函数的调用方式。取值为 `Sync`（同步调用）或 `Async`（异步调用）。
- （可选）`StatefulAsyncInvocationID`：表示有状态异步执行的ID。该参数可以帮助您在函数计算控制台内

搜索目标任务的名称。

> **说明** 当您的参数 InvocationType 设置为 Async 时，表示Serverless工作流中调用的函数的异步调用类型为有状态。该ID需在函数范围内唯一。

异步调用函数的集成模式

当流程执行到该步骤时，会异步调用函数，然后开始函数的执行，同时流程会直接进入后续步骤的执行，不会等待回调或任务执行完成。

```
version: v1
type: flow
steps:
 - type: task
   name: mytask
   resourceArn: acs:fc:{region}:{account}:services/{serviceName}.{qualifier}/functions/{functionName}
   pattern: requestResponse  # Async invocation with sync pattern
   serviceParams:
     InvocationType: Async
```

该示例展示了当 mytask 步骤执行时会触发函数执行的流程。触发成功后，将进入下一步骤，函数执行流程可能仍然正在执行。

当流程执行到该步骤时，会异步调用函数，然后开始函数的执行，然后流程会进入等待状态。当函数执行完成后，Serverless工作流的流程会获得通知然后开始下一步骤的执行。

```
version: v1
type: flow
steps:
 - type: task
   name: mytask
   resourceArn: acs:fc:{region}:{account}:services/{serviceName}.{qualifier}/functions/{functionName}
   pattern: sync  # Async invocation with sync pattern
   serviceParams:
     InvocationType: Async
```

该示例展示了当该步骤执行时会触发一个Serverless工作流流程。触发成功后将等待该流程的执行结果，执行完成后将进入下一步骤。当 mytask 步骤执行时会触发函数执行的流程。触发成功后，将等待该流程的执行结果，执行完成后才进入下一步骤。

> **说明** 同步集成模式的整体行为与同步调用函数行为无差异，仅在调用函数时函数的调用方式中有区别。

当流程执行到该步骤时，会异步调用函数，然后开始函数的执行并传入任务令牌，之后流程会进入等待状态。无论函数执行是否完成，直到您手动通过任务令牌通知流程执行结果后才会继续执行下一个步骤。

```
version: v1
type: flow
steps:
 - type: task
   name: mytask
   resourceArn: acs:fc:{region}:{account}:services/{serviceName}.{qualifier}/functions/{functionName}
   pattern: waitForCallback  # Async invocation with sync pattern
   serviceParams:
     InvocationType: Async
```

该示例展示了当 `mytask` 步骤执行时会触发函数执行的流程。触发成功后，将暂停流程执行，等待回调（通过ReportTaskSucceed或ReportTaskFailed）。在收到回调请求并处理完成后，流程将进入下一步骤。回调由您发起，此时函数执行流程可能已经执行完成，也可能还在执行。

请求响应模式

同步模式

等待回调模式

推荐使用示例

Serverless工作流异步调用函数计算功能配合有状态的异步调用可以最大限度的支持任务类场景。当您开启有状态异步调用功能并且采用异步调用的方式时，可以在调用过程及结束后查看函数执行情况，具有更强的任务执行可观测能力；也可以操作停止函数执行，使流程具有更强的操作能力。Serverless工作流的流程定义语言如下所示：

```
version: v1
type: flow
steps:
 - type: task
   name: mytask
   resourceArn: acs:fc:::services/{serviceName}.{qualifier}/functions/{functionName}
   pattern: sync  # Async invocation with sync pattern
   inputMappings:
    - target: id
      source: $context.execution.name
   serviceParams:
     InvocationType: Async
     StatefulAsyncInvocationID: $.id
```

4.集成MNS队列

Serverless工作流任务（Task）步骤支持向某个指定的MNS队列中发送消息，本文介绍了MNS队列集成的使用场景、通用参数、集成模式以及权限配置。

使用场景

任务步骤对MNS队列的集成有请求响应（requestResponse）模式和等待回调（waitForCallback）模式，分别适用于不同场景。

集成模式	模式（pattern）参数	适用场景	说明
请求响应模式	requestResponse	事件通知	持久化通知工作流执行外的服务，流程执行不关心被通知者如何处理该消息。
等待回调模式	waitForCallback	编排自定义任务类型	发送消息到队列，任意环境中的任务执行者（例如ECS虚拟机、自建机房的服务器）收到消息，执行任务后回调Serverless工作流，更多信息，请参见最佳实践。

通用参数

下文示例中，Serverless工作流会向 `resourceArn` 中指定的队列名称（替换 `{queue-name}` ）发送一条消息。消息正文和参数在 `serviceParams` 对象中指定，支持的参数如下所示：

- （必填）**MessageBody**：消息正文。 `MessageBody: $` 表示通过输入映射（inputMappings）产生消息正文。例如该步骤进入（StepEntered）时：
  - input对象为 `{"key": "value_1"}` 。
  - 映射后的输入为 `{"key_1": "value_1", "key_2": "value"}` 。
  - 任务步骤会以 `{"key_1": "value_1", "key_2": "value"}` 字符串作为消息正文发送到MNS队列。
- （必填）**Priority**：消息优先级权值。
- （可选）**DelaySeconds**：延迟出现秒数。

更多信息，请参见MNS SendMessage API。

```
version: v1
type: flow
steps:
 - type: task
  name: Task_1
  resourceArn: acs:mns:::/queues/{queue-name}/messages # 表示该任务（Task）步骤会向同地域、同账号下的MNS {queue-name}队列发送消息。
  pattern: requestResponse                    # 表示该任务步骤在发送MNS消息完成后结束。
  inputMappings:
    - target: key_1
      source: $input.key
    - target: key_2
      source: value
  serviceParams:   # 服务集成参数。
     MessageBody: $ # 用映射后的input作为要发送消息的内容。
     DelaySeconds: 0 # 消息延迟出现时间，单位为秒。
     Priority: 1   # 消息队列的优先级。
```

集成模式

- 请求响应（requestResponse）模式
  任务步骤中的 pattern: requestResponse 代表集成为请求响应模式。该模式下，Serverless工作流会向 resourceArn 中指定的 {queue-name}发送一条消息，发送消息成功后步骤即成功。当发送失败时将根据重试配置决定步骤失败或是重试。

  ```
  version: v1
  type: flow
  steps:
   - type: task
    name: mytask
    resourceArn: acs:mns:::/queues/{queue-name}/messages # 表示该任务（Task）步骤会向同区域，同账号下的MNS队列发送消息。
    pattern: requestResponse                    # 表示该任务步骤在发送MNS消息完成后结束。
    inputMappings:
      - target: key_3
        source: value
    serviceParams:   # 服务集成参数。
       MessageBody: $ #{"key_3": "value"}会作为消息正文被发送到MNS队列。
       DelaySeconds: 0 # 消息延迟出现时间，单位为秒。
       Priority: 1   # 消息队列的优先级。
  ```

- 等待回调（waitForCallback）模式
  任务步骤中的 pattern: waitForCallback 代表集成为等待回调模式。该模式下，Serverless工作流会向 resourceArn 中指定的 {queue-name} 发送一条消息，发送消息成功后步骤会暂停并且等待一个回调。任务执行者需要使用该任务令牌（taskToken）回调Serverless工作流。当回调结果为成功时（调用ReportTaskSuccceeded）则该任务步骤成功，否则（调用了ReportTaskFailed）将根据重试配置决定步骤失败或是重试。
  任务令牌（taskToken）：任务令牌可以从该步骤的上下文（context）对象中获取，通过输入映射赋值给消息正文JSON对象的某个字段。下文示例中，假设该步骤目前的context对象是 {"task":{"token":"my-token-1"}} ，则映射后的任务输入为 {"task_token": "my-token-1", "key": "value"} ，该输入也会被作为消息正文发送到MNS队列。

```
version: v1
type: flow
steps:
 - type: task
  name: mytask
  resourceArn: acs:mns:::/queues/{queue-name}/messages # 表示该任务（Task）步骤会向同区域，同账号下的MNS {queue-name} 队列发送消息。
  pattern: waitForCallback # 表示该任务步骤在发送MNS消息成功后会暂停，直到收到回调。
  inputMappings:
    - target: task_token
      source: $context.task.token # 从context对象中获取标识该任务的令牌 （task token）。
    - target: key
      source: value
  serviceParams:   # 服务集成参数。
      MessageBody: $ # 用映射后的input作为要发送消息的内容。
      Priority: 1   # 消息队列的优先级。
```

流程角色配置

Serverless工作流通过扮演流程（Flow）指定的RAM角色（RAM role）获取您的临时AccessKey代表您访问MNS，因此需要为流程角色配置可以向MNS发送消息（SendMessage）的权限。您可以选择系统权限策略或者自定义权限策略中的任意一种配置方法授权Serverless工作流向您的MNS队列发送消息。

- **系统权限策略**：向流程角色授予AliyunMNSFullAccess系统权限。
- **自定义权限策略**：如果需要更细粒度的控制只允许向**{queue-name}**队列发送消息，创建以下策略绑定在流程角色上。

```
{
 "Version": "1",
 "Statement": [
  {
   "Action": "mns:SendMessage",
   "Resource": "acs:mns:*:*:/queues/{queue-name}/messages",
   "Effect": "Allow"
  }
 ]
}
```

5.集成MNS主题

Serverless工作流任务步骤集成了MNS主题模型，支持将消息发布到MNS主题。通过设置发布消息参数可以实现主题消息推送，目前支持的推送类型有队列推送、HTTP推送和邮件推送。本文介绍集成MNS主题模型的模式及相关消息参数。

集成模式

下文介绍了集成MNS主题模型的两种模式，分别是请求响应（requestResponse）模式和等待回调（waitForCallback）模式，您可以选择任意一种模式来编排MNS主题模型。

- 请求响应（requestResponse）模式
  该模式下，任务步骤在消息发布请求完成后会继续执行流程。流程定义如下。

```
version: v1
type: flow
steps:
 - type: task
   name: mns-topic-task
   resourceArn: acs:mns:::/topics/{topicName}/messages # 调用MNS主题资源
   pattern: requestResponse # (可选) 默认模式
   outputMappings:
    # PublishMessage 返回参数
    - target: messageID  # 消息ID
      source: $local.MessageId
    - target: requestID  # 请求ID
      source: $local.RequestId
    - target: messageBodyMD5  # 消息体MD5
      source: $local.MessageBodyMD5
   serviceParams:
    # PublishMessage 请求参数
    MessageBody: $.messageBody  # 消息体
    MessageTag: $.messageTag  # (可选) 消息Tag
    MessageAttributes: $.messageAttributes  # (可选) 消息额外属性，必须为`JSON`
```

- 等待回调（waitForCallback）模式
  该模式下，任务步骤在发布消息后流程会等待，直到收到回调通知后才继续执行流程。流程定义如下。

```
version: v1
type: flow
steps:
 - type: task
   name: mns-topic-task
   resourceArn: acs:mns:::/topics/{topicName}/messages # 调用MNS主题资源
   pattern: waitForCallback # 等待回调模式
   inputMappings:
    - target: messageBody
      source: $input.messageBody
    - target: messageTag
      source: $.messageTag
    - target: messageAttributes
      source: $.messageAttributes
    - target: taskToken # 回调任务状态的Token（系统自动生成）
      source: $context.task.token
   serviceParams:
    # PublishMessage 请求参数
    MessageBody: $ # 消息体 (将taskToken封装到MessageBody中)
    MessageTag: $.messageTag # (可选) 消息Tag
    MessageAttributes:$.messageAttributes # (可选) 消息额外属性
```

参数说明

- 上下文参数（context）
  - 任务步骤TaskToken
    任务步骤在等待回调（waitForCallback）模式下会自动生成用于回调任务状态的TaskToken，使用 `$context.task.token` 可以获取到该值。通过TaskToken作为参数调用ReportTaskSucceeded、ReportTaskFailed接口来回调任务的状态。
- 服务参数（serviceParams）
  - MessageBody：发送的消息体字符串。
  - （可选）MessageTag：发送的消息Tag。
  - （可选）MessageAttributes：特定的消息推送需要指定该参数，格式为JSON字符串。
  - 邮件推送：参数必须包含MailAttributes，示例如下所示。

    ```
    {
     "MailAttributes":{
      "Subject":"{邮件主题}",
      "AccountName":"{发信地址（邮箱地址）}",
      "AddressType":0,
      "IsHTML":true,
      "ReplyToAddress":0
     }
    }
    ```

  以上参数，更多详细信息请参见PublishMessage。

6.集成视觉智能服务

Serverless工作流目前已集成阿里云视觉智能服务，您可以通过Serverless工作流来编排视觉智能服务的API。本文介绍集成视觉智能服务的具体操作步骤。

背景信息

您在Serverless工作流的任务步骤指定资源类型为阿里云视觉智能服务的某个API，并提供所需参数，Serverless工作流将在该步骤中按照定义调用对应的API，并将调用结果作为步骤的输出，以便您根据需要取对应的值来完成后续任务。如果调用过程中出现错误，您可以在流程中捕获该错误并根据错误类型实施重试或跳转等策略。

前提条件

- 获取调用服务的接口语义及所需参数。您可以通过访问视觉智能官网或参见文末已支持的视觉智能服务及API列表查看所有API信息。
- 获取流程的roleArn具有调用该接口的相关权限，如AliyunVIAPIFullAccess。更多信息，请参见执行角色。

使用方式

1. 在流程中定义被调用的API。

   您可以在任务步骤中调用视觉智能的API。在任务步骤中，按照以下示例指定Action参数替代resourceArn参数。

   ```
   action: {serviceName}:{apiName}
   ```

   - `{serviceName}`：服务名称，更多信息，请参见已支持的视觉智能服务及API列表。
   - `{apiName}`：调用的API名称，更多信息，请参见已支持的视觉智能服务及API列表。

2. 指定调用API参数。

   API调用的参数请填写到ServiceParams下。对于服务API已经定义的必填参数，Serverless工作流将进行定义检查。
   以视觉智能服务的API ClassifyCommodity为例，对应的调用任务FDL表述如下。关于API的信息，请参见商品分类。

   ```
   ...
    - type: task
      name: APIClassifyCommodity
      action: goodstech:ClassifyCommodity
      inputMappings:
        - target: image
          source: $input.imageURL
      serviceParams: # 描述ClassifyCommodity所需参数，参数列表请参见API说明ImageURL: $.image
   ```

   在该示例中，action指定了本次调用的具体服务名称及API名称，ServiceParams中的参数为对应API的参数。

3. 错误处理。

   当调用API失败时，Serverless工作流将返回error和cause两个key作为本步骤的output输出，您可以通过error来捕获错误类型进行相应的跳转等逻辑。Serverless工作流将对服务返回的原始错误码添加{serviceName}.前缀作为捕获标识。以 `goodstech:ClassifyCommodity` 的错误码为例。更多信息，请参见错误码说明。

```
...
steps:
 - type: task
   name: APIClassifyCommodity
   action: goodstech:ClassifyCommodity # 格式为{serviceName}:{apiName}，参见文末的API列表。
   ...
   retry: # 捕获错误后重试
    - errors:
      - goodstech.InternalError.Busy
      intervalSeconds: 10
      maxAttempts: 2
      multiplier: 2
   catch: # 捕获错误后跳转。
    - errors:
      - goodstech.CommonError
      - goodstech.IllegalUrlParameter
      - goodstech.InvalidParameter
      goto: xxxxx
```

其中，InternalError.Busy是原始服务错误码，如需在流程中捕获该错误码，请添加服务前缀goodstech.。

示例：编排视觉智能图片识别API

本示例将以视觉智能API图片识别为例。识别图片中的商品种类，更多信息，请参见商品分类。

```
version: v1
type: flow
steps:
 - type: task
   name: APIClassifyCommodity
   action: goodstech:ClassifyCommodity # 格式为{serviceName}:{apiName}，参见文末的API列表。
   inputMappings: # 对变量进行输入映射，用于ClassifyCommodity参数。
    - target: image
      source: $input.imageURL
   outputMappings: # 映射输出到本地变量。
    - target: classifyCommodity_Categories
      source: $local.Data.Categories
   serviceParams: # 描述ClassifyCommodity所需参数，参数列表请参见API说明。
     ImageURL: $.image
   retry: # 捕获错误后重试。
    - errors:
      - goodstech.InternalError.Busy
      intervalSeconds: 10
      maxAttempts: 2
      multiplier: 2
   catch: # 捕获错误后跳转。
    - errors:
      - goodstech.CommonError
      - goodstech.IllegalUrlParameter
      - goodstech.InvalidParameter
      goto: ExecFailed
 - type: foreach # 根据ClassifyCommodity返回参数说明，该接口会返回一个数组。这类数据建议在后续的流程中使用foreach步骤进行处理。
   name: ElemDealt
   iterationMapping:
    collection: $.classifyCommodity_Categories
    item: catagory
   steps:
    - type: pass
      name: pass1
      inputMappings:
       # the index can be from context
       - target: index
         source: $context.step.iterationIndex
       - target: catagory
         source: $input.catagory
   end: true
 - type: fail
   name: ExecFailed
```

在Serverless工作流创建该流程。请以下述输入作为流程输入，启动流程。

```
{
  # 待处理的图片地址。
  "imageURL": "https://viapi-demo.oss-cn-shanghai.aliyuncs.com/viapi-demo/images/DetectImageElements/detect-elements-src.png"
}
```

已支持的视觉智能服务及API列表

服务名称	API名称	API功能说明
facebody 开通人体人脸服务	DetectFace	识别图像中是否有人脸并给出人脸区域，进行人脸检测后返回检测到的人脸矩形框坐标。可支持最多上千个人脸的同时检测，支持平面360度旋转人脸检测，支持左右最大90度侧面人脸检测。同时毫秒级提取图像中的人脸五官关键点，识别人脸105个关键点定位。
	RecognizeExpression	识别图片人脸表情。
	RecognizeFace	在人脸检测基础上，实现高性能的人脸识别。我们的算法在公开测试集lfw上达到了99.58%的识别精度。
	CompareFace	基于用户输入的两张图片，可检测两张图片中的人脸，并挑选两张图片的最大人脸进行比较，判断是否是同一人。同时返回这两个人脸的矩形框坐标、比对的置信度，以及不同误识率的置信度阈值。
	detectmask	识别输入图片中占比最大的人脸是否有戴口罩。
	DetectBodyCount	识别并统计图片中人物人体的数量，主要适用于室内场景。
	CreateFaceDb	创建人脸数据库。
	ListFaceDbs	查看人脸数据库列表。
	AddFaceEntity	向人脸数据库中添加人脸样本数据。
	GetFaceEntity	查询人脸数据库中的人脸样本数据。
	ListFaceEntities	查询人脸数据库中的人脸样本列表。
	UpdateFaceEntity	更新人脸数据库中的人脸样本数据。
	AddFace	为指定数据库添加人脸数据。
	SearchFace	根据输入图片，可以在数据库中搜索相似的人脸图片数据。
	DeleteFace	删除指定数据库中的人脸图片信息。
	DeleteFaceEntity	删除人脸数据库中的人脸样本数据。
	DeleteFaceDb	删除指定的人脸数据库。

服务名称	API名称	API功能说明
	BodyPosture	获取人体的18个关键点信息。
	HandPosture	获取手势的21个关键点信息。
	DetectPedestrian	检测图像中的人体。
	EnhanceFace	对输入图像中的人脸进行裁剪、对齐、细节增强，最后再融合回原图。
	FaceBeauty	对图像中的人脸进行美颜，包括磨皮、美白、去除黑眼圈、法令纹等。
	FaceMakeup	模拟彩妆，通过添加口红、高光、整妆等彩妆素材，进一步提升人脸美化效果。
	FaceTidyup	对人脸的轮廓和五官进行调整。您也可以通过手动调整美型强度，进一步对人脸进行精细化调整。
	FaceFilter	对图片的整体风格进行转变。
	RecognizeIdentityCard	自动定位身份证图片区域，识别身份证上的相关信息。
	RecognizeBankCard	自动定位银行卡图片区域，识别银行卡号等相关信息。
	RecognizeBusinessCard	自动从图片中定位名片图片，识别名片上的相关信息。
	RecognizeAccountPage	自动定位户口个人页面图片区域，识别户口个人页的相关信息。
	RecognizeDrivingLicense	自动定位行驶证图片区域，识别行驶证上的相关信息，支持新能源车辆的相关信息识别。
	RecognizeDriverLicense	自动从图片中定位驾驶证图片，识别驾驶证上的相关信息。
	RecognizeLicensePlate	自动定位车牌区域，识别车牌内容等信息，支持新能源车牌识别。
	RecognizeVINCode	自动定位VIN码区域，识别VIN码区域内容。
	RecognizeTaxiInvoice	自动定位出租车发票区域，识别出租车发票上的相关信息。
	RecognizeTrainTicket	自动定位火车票区域，识别出火车票上的相关信息。

服务名称	API名称	API功能说明
ocr 开通文字识别服务	RecognizeBusinessLicense	自动定位营业执照区域，识别营业执照上的相关信息。
	RecognizeStamp	自动定位图片里公章的位置，识别公章中的机关、团体、企事业单位名称。
	RecognizeVATInvoice	识别增值税发票（电子发票和纸质发票）上的内容。
	RecognizeCharacter	多场景图片文字识别并返回坐标信息。
	GetAsyncJobResult	异步接口调用API接口后，返回的并不是真正的请求结果，您需要保存返回结果中的RequestId，然后调用GetAsyncJobResult来获取真正的请求结果。
	TrimDocument	对输入文档内容进行解析，输出结构化样式（HTML或者JSON）。
	RecognizeChinapassport	识别中国护照关键字段内容。
	RecognizeTakeoutOrder	识别外卖单上的关键字段内容，输出商店名称、电话、包装费、配送费、商品合计、其他费用合计、顾客优惠合计、总件数、在线支付、订单编号、下单时间等。目前支持饿了么外卖单。
	RecognizePassportMRZ	可以检测输入的护照MRZ图像，输出11个信息，方便后续信息提取和证件审核。
goodstech 开通商品理解服务	ClassifyCommodity	识别图像中的商品分类，返回商品类目、置信度等信息，目前已经支持服饰鞋包、3C数码、家居用品等超过1万种类目分类。
	RecognizeFurnitureAttribute	识别输入的家居模型图的风格，目前支持16种风格识别。
	RecognizeFurnitureSpu	对输入的家居模型图进行分类，目前类别数可达70类。
	RecognizeImageColor	可以对输入图的颜色信息进行分析，给出颜色值（RGB形式和HEX格式）与对应的占比信息。
	TaggingImage	识别图像中的主体内容并打上类型标签，支持数千个内容标签，覆盖常见物体品类。

服务名称	API名称	API功能说明
imagerecog 开通图像识别服务	RecognizeScene	识别图像所处的场景环境，支持数十种常见场景，如天空、草地等。
	DetectImageElements	用于识别输入图中所包含的元素，用矩形框标注出其位置，并区分其对应的基本类型（人物、修饰、文案）。
	RecognizeImageStyle	对输入图的风格类型进行分析，识别可能的风格与语意标签。
	ClassifyingRubbish	对图片中的物品垃圾进行分类，并给出具体的物品名称。
	RecognizeVehicleType	识别图片（完整或部件图片）中汽车的类型，目前主要有小轿车、多用途汽车、SUV等类别。
imageseg 开通图像分割服务	SegmentHead	识别输入图像中的人头轮廓，含人脸、头发耳朵、发饰区域，不含脖子，返回仅人头区域可视的透明图（4通道），适用于单人场景，多人场景。人像比较明显的图片输入效果会更好。
	SegmentFace	识别输入图像中的人脸轮廓，不含脖子、耳朵头发，返回仅人脸区域可视的透明图（4通道），适用于单人、多人场景，人脸比较明显的图片输入效果会更好。
	SegmentHair	识别输入图像中的头发轮廓，不含脖子、耳朵头发，返回仅人脸区域可视的透明图（4通道），适用于单人、多人场景，人脸比较明显的图片输入效果会更好。
	ParseFace	识别输入图像中的五官轮廓，对眼睛、鼻子、嘴进行像素级语义分割，人脸比较明显的图片输入效果会更好。
	SegmentVehicle	识别输入图像中的汽车轮廓，对汽车进行像素分割，输出结果为透明图。
	SegmentCommodity	用于识别输入图像中的商品轮廓，与背景进行分离，返回分割后的前景商品图（4通道），适用于单商品、多商品、复杂背景等场景。

服务名称	API名称	API功能说明
	SegmentBody	用于识别输入图像中的人体轮廓，与背景进行分离，返回分割后的前景人像图（4通道），适用于单人、多人、复杂背景、各类人体姿态等场景。
	SegmentCommonImage	识别输入图像中的视觉中心物体轮廓，与背景进行分离，返回分割后的前景物体图（4通道）。
	SegmentFurniture	对输入图片中的家具进行像素级抠图。
	RefineMask	对输入图像与粗糙mask进行精细化处理，输出精细化mask。
imageenhan 开通图像增强服务	ChangeImageSize	改变图片大小。
	IntelligentComposition	输入一张的图像，通过美学评估，智能输出几个bounding box，根据这些bounding box可以将原图裁剪成构图更好的图像。
	ExtendImageStyle	将输入图按照指定的风格图像进行风格的迁移，使得图像的色彩、笔触等视觉风格发生转化。
	MakeSuperResolutionImage	可以将输入图放大四倍，同时基于推断出的细节保持结果图像的清晰度。
	RecolorImage	将输入图自动或根据指定色板进行色彩转换，同时保证视觉热点区域避免不正常拓色。
	RemoveImageSubtitles	擦除图片中的标准字幕。
	RemoveImageWatermark	擦除图片中的常见标志，如台标、互联网平台Logo标志等。
	ImageBlindCharacterWatermark	为图片添加或者解析指定文字水印。
	ImageBlindPicWatermark	为图片添加或解析图片水印。
	ClassifyVehicleInsurance	对输入的车险图片进行分类。
	RecognizeVehicleParts	检测图片中车辆部件的位置以及名称。
	DetectVehicle	检测图像中的机动车主体，返回该机动车主体的区域位置，坐标信息。
	DetectMainBody	输入抠图后的图片检测，输出主体定位信息。

服务名称	API名称	API功能说明
objectdet 开通目标检测服务		
	RecognizeVehicleDashboard	识别仪表盘上故障灯等信息。
	RecognizeVehicleDamage	检测图片中车辆损伤的位置以及类型。
	DetectTransparentImage	检测图片背景是否为透明图。
	DetectObject	检测输入图像中的物体。
	DetectWhiteBaseImage	检测图片背景是否为白底图。