智能排班

二次开发指南

文档版本 01

发布日期 2023-04-08

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 智能排班二次开发指南

1.1 概述

1.1.1 应用场景

智能排班通过智能排班调度引擎，合理评估和安排工作量，实现自动化、智能化排班调度，对人员进行科学管理，促进智能人力资源管理。

1.1.2 开发场景

智能排班提供了基于AppCube开发的**智能排班模型BO**和**智能排班基线应用**，封装了人员管理、配套资源管理、排班规则管理、排班结果管理等智能排班能力，提供二次开发和集成交付能力，支持公有云（OC）/私有云（OP）部署。通过复用服务和应用，合作伙伴可以二次开发定制应用，避免重复设计，达到提高应用开发工作效率的目的。

- **智能排班模型BO**：封装了智能排班的完整的数据模型、业务逻辑，通过开放出来的接口为上层应用提供服务，提供的接口请参见**智能排班模型BO接口**。
- **智能排班基线应用**：基于智能排班模型BO，针对智能排班领域开发的一个完整应用，包含前台页面、后台逻辑等，是一个可直接部署使用的应用。

表 1-1 开发场景

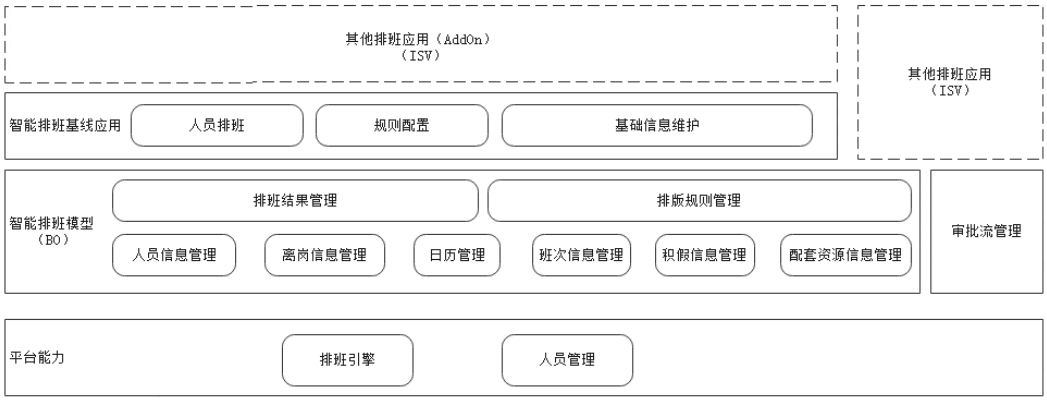
场景	章节
业务开发	**定制开发指导（基于智能排班模型BO）**
	定制开发指导（基于智能排班基线应用）
应用集成	**集成应用到ISDP+平台（可选）**

1.1.3 架构及开放能力

平台提供了排班引擎和人员管理能力；智能排班模型BO将业务中的公共能力，如人员信息管理、离岗信息管理等以接口的方式开放；审批流管理为审批流管理应用，提供

审批流的管理，可以直接集成应用进行使用；智能排班基线应用提供基于智能模型BO开发的完整的应用。

图 1-1 架构



合作伙伴可以直接使用智能排班模型BO开放的接口，或智能排班基线应用的资产，复用和灵活扩展资产提供的业务能力，结合业务述求，快速实现应用构建。

1.2 准备工作

1.2.1 介绍

在定制开发开始之前，需要完成环境的准备。

1. 准备AppCube开发环境，准备ISDP+环境。
2. 完成公共应用/BO的安装配置，实现AppCube与ISDP+的对接。公共应用/BO的安装部署请参见**安装和配置公共应用/BO**。
3. 安装部署智能排班模型BO和智能排班基线应用，后续基于智能排班模型BO和智能排班基线应用进行定制开发。

1.2.2 部署应用/BO

本节介绍如何安装部署智能排班模型BO和智能排班基线应用。

背景信息

定制开发时，可以基于智能排班模型BO，也可以基于智能排班基线应用。

- 智能排班模型BO：ISDP__IntelligentSchedulingModel_b-XX.XX.XX.zip、ISDP__IntelligentSchedulingModel_bp-XX.XX.XX.预置数据包.zip
- 智能排班基线应用：ISDP__IntellScheduleBaseline-XX.XX.XX.zip

须知

定制开发时，请根据实际情况安装部署。

- 如果基于智能排班模型BO进行定制开发，**无需安装部署智能排班基线应用**。
- 如果基于智能排班基线应用进行定制开发，**需要同时安装部署智能排班模型BO和智能排班基线应用**。

操作步骤

步骤1 登录AppCube开发环境。

步骤2 如图1-2所示，单击“管理”，进入“管理”页面。

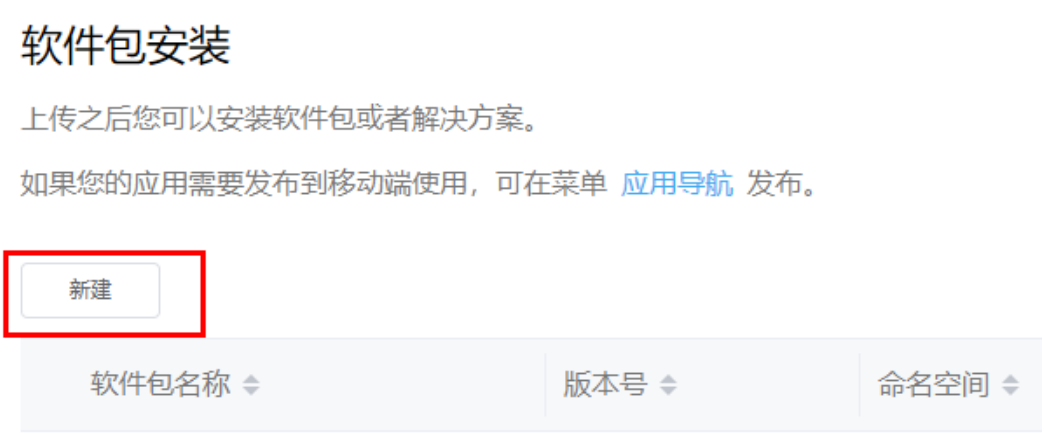
图 1-2 管理

步骤3 如图1-3所示，选择“应用管理 > 软件包管理 > 软件包安装”，进入“软件包安装列表”页面。

图 1-3 软件包安装列表

步骤4 在“软件包安装列表”页面，如图1-4所示，单击“新建”。

图 1-4 新建



步骤5 如图1-5所示，在“软件包安装”页面将资产包拖入进去或单击上传资产包（ISDP__IntelligentSchedulingModel_b-XX.XX.XX.zip），不勾选“检查软件包中对象属性变更情况”，单击“安装”。

图 1-5 软件包安装

步骤6 安装成功后，如图1-6所示，软件包安装列表页面展示安装的软件包。

图 1-6 软件包安装列表

步骤7 参见步骤4～步骤6，继续完成包ISDP__IntelligentSchedulingModel_bp-XX.XX.XX.预置数据包.zip和ISDP__IntellScheduleBaseline-XX.XX.XX.zip的安装，安装完成后，如图1-7所示。

图 1-7 安装结果

安装后，可以在AppCube开发环境的“库”中查看到BO和应用。

----结束

1.3 定制开发指导（基于智能排班模型 BO）

本章节介绍如何基于智能排班模型BO进行二次开发。

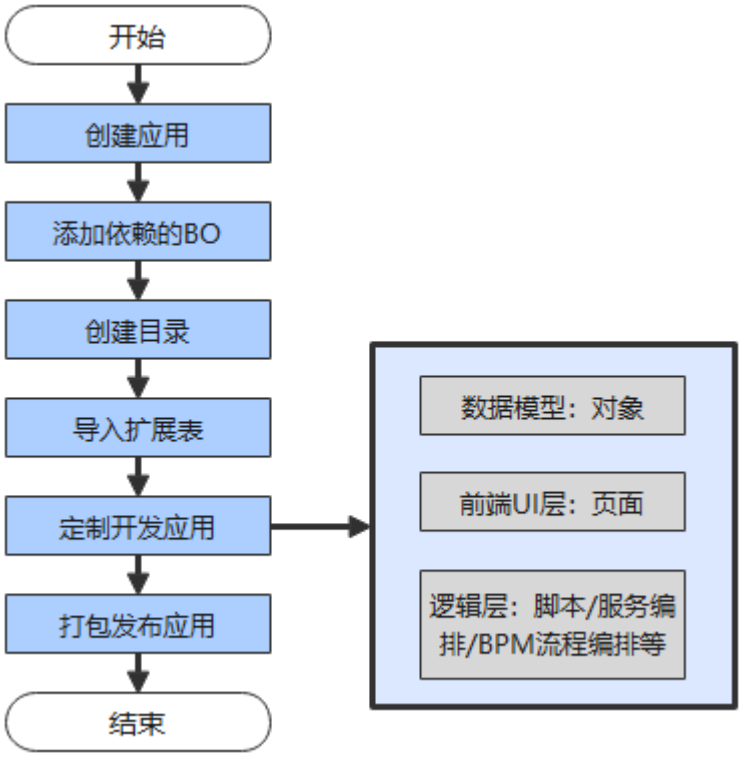
1.3.1 开放能力

智能排班模型BO封装了智能排班的完整的数据模型、业务逻辑，通过开放出来的接口为上层应用提供服务。

智能排班模型BO开发的接口请参见**智能排班模型BO接口**。

1.3.2 定制开发流程

图 1-8 应用定制开发流程图



序号	步骤	说明
1	创建应用	在进行应用开发之前，需要先创建应用。 具体操作请参见**创建应用**。
2	添加依赖的BO	创建应用后，在应用的开发态，添加应用依赖的BO。 具体操作请参见**添加依赖的BO**。
3	创建目录	在应用的开发态，根据规划创建目录，用于存放数据对象、服务编排、脚本、事件、前端页面等。 具体操作请参见**创建目录**。
4	导入扩展表	在创建的目录下，导入智能排班模型BO中的扩展表。 具体操作请参见**导入扩展表**。
5	定制开发应用	在应用的开发态，定制开发应用，需要进行数据模型、逻辑层以及前端UI层等的开发。 具体操作请参见**定制开发应用**。
6	打包发布应用	安装部署应用，部署到沙箱环境中进行测试。 具体操作请参见**打包发布应用**。

1.3.3 基本操作

1.3.3.1 创建应用

操作步骤

步骤1 登录AppCube开发环境。

步骤2 如图1-9所示，在开发环境首页的“项目”页签下，单击“行业应用”，进入到行业应用。

图 1-9 进入行业应用

步骤3 如图1-10所示，单击“创建行业应用”，弹出“创建行业应用”页面。

图 1-10 创建行业应用

步骤4 如图1-11所示，添加图标、设置标签、名称、分类和描述，具体参数说明请参见表1-2，单击“创建”，创建应用后，进入应用开发阶段，如图1-12所示。

图 1-11 创建行业应用

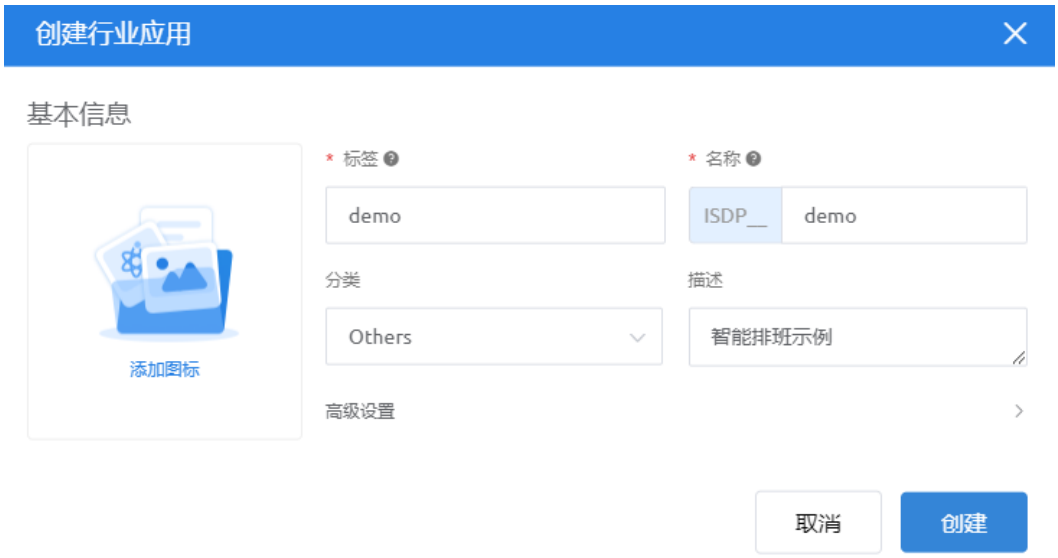
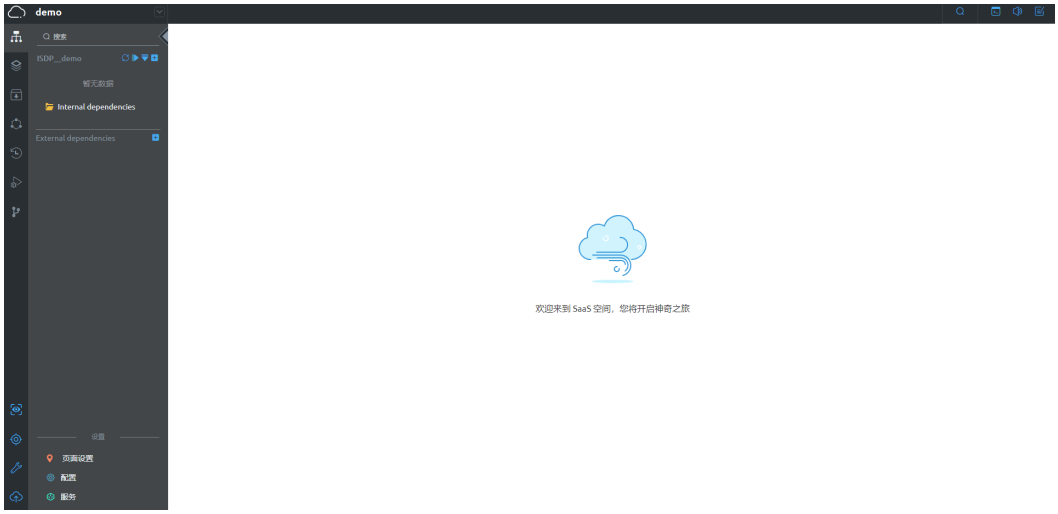


表 1-2 应用基本信息

参数	配置说明	示例
添加图标	为该应用设置图标。如果不设置，则使用默认图标。 单击“添加图标”，在弹出“图标选择”页面中选择图标。	使用默认图标
标签	应用中用于展示的文字，为了区分不同应用的描述信息，创建后可修改。 应用创建之后，应用标签可以在应用设置中修改。	demo
名称	应用在系统内的唯一标识，系统会自动在该名称前添加*命名空间*__，创建后不支持修改。 设置要求：必须以字母开头，没有连续的下划线，空格和特殊字符。	demo **说明** 应用创建后，应用名称自动添加命名空间前缀，例如：ISDP__demo。
分类	应用所属的分类。 设置分类后，工程列表和库列表都可以根据应用的分类进行筛选。	Others
描述	应用的描述信息。	智能排班示例
高级设置	展开“高级设置”时才会显示该参数。开发的资产包依赖所选择的运行时版本，若线下运行版本不一致，可能产生不兼容。	保持默认

图 1-12 应用开发页面



----结束

1.3.3.2 添加依赖的 BO

背景信息

添加依赖的BO时，支持添加内部依赖或外部依赖，内外部依赖使用方法类似，只是在打包发布应用时存在区别。

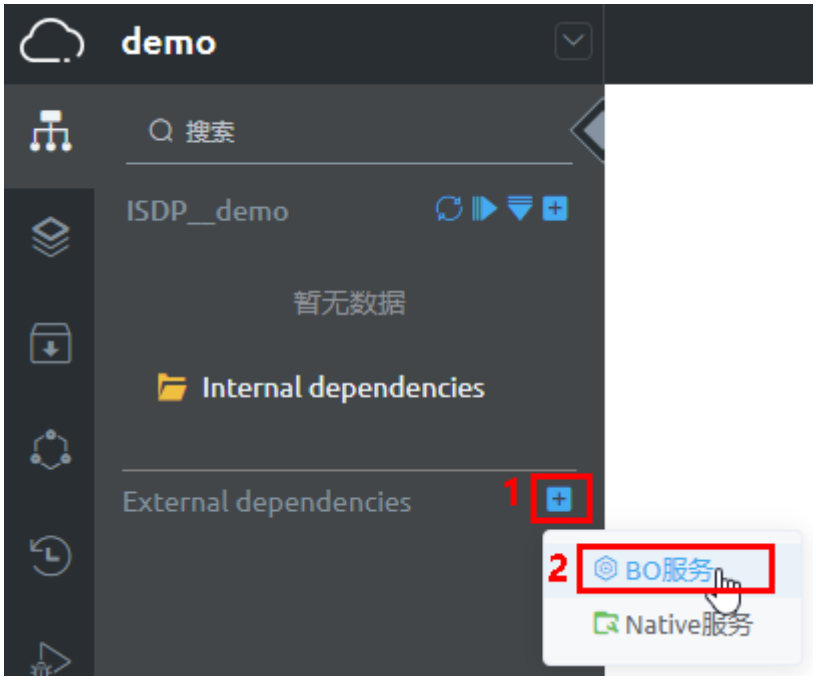
- 内部依赖：“Internal dependencies”为内部依赖文件夹，添加的BO服务，在打包应用时也会随应用打包发布出去。
- 外部依赖：“External dependencies”为外部依赖文件夹，添加的BO服务，在打包应用时不会打包出去，在上线部署的时候，也需要将依赖的BO安装到对应环境中。

本章节以外部依赖为例进行描述。

操作步骤

步骤1 如图1-13所示，在应用开发页面下“External dependencies”目录树旁单击加号，选择“BO服务”。

图 1-13 BO 服务

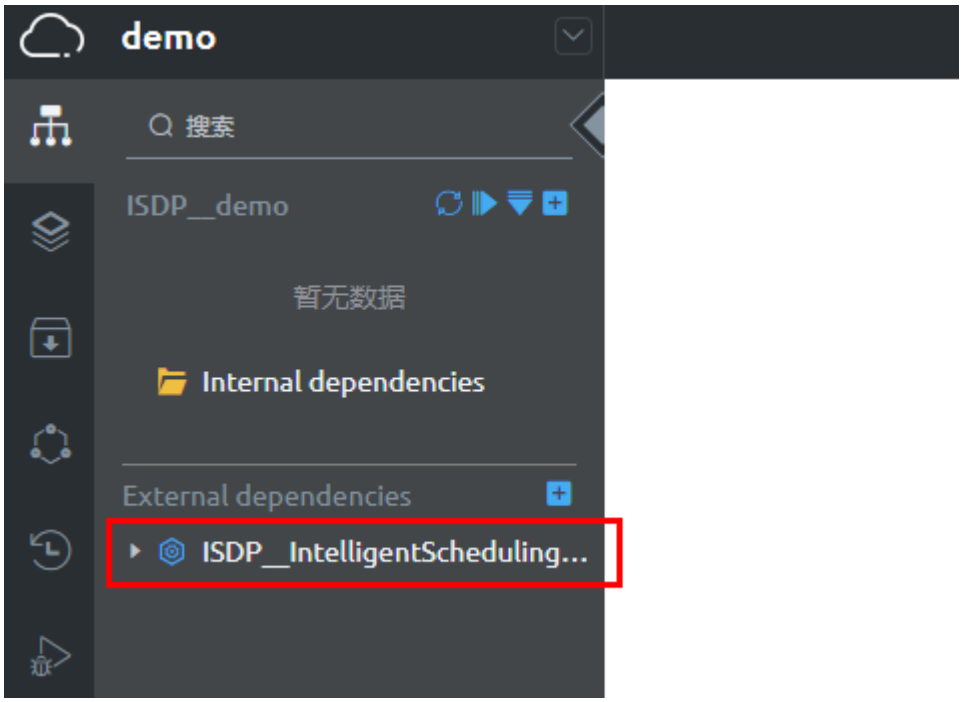


步骤2 如图1-14所示，在“添加BO”页面选择“智能排班”，单击“添加”，添加BO。

图 1-14 添加 BO

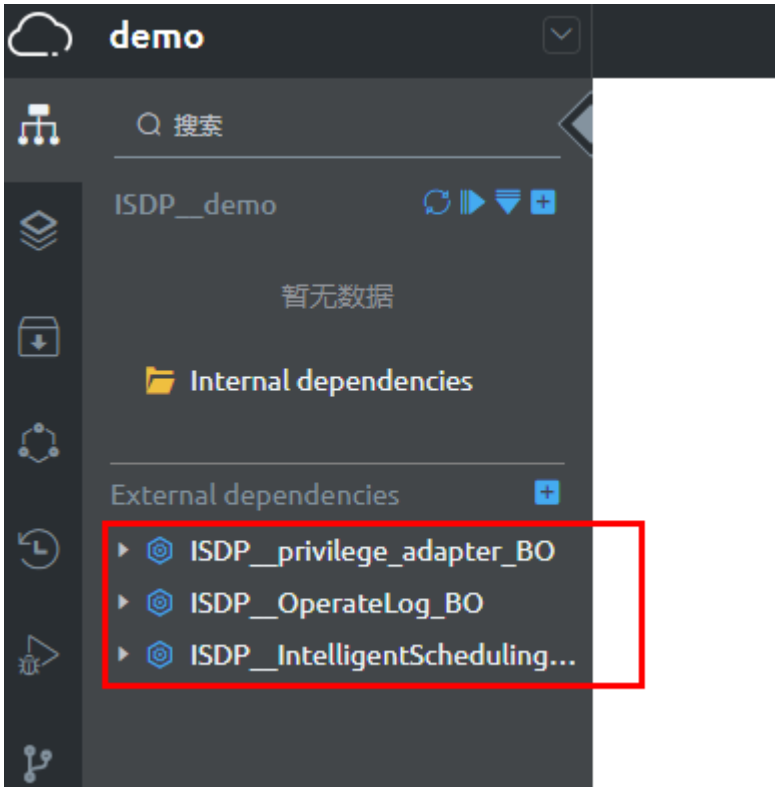
添加后，该BO会出现在上层应用中，如图1-15所示。

图 1-15 添加的 BO



步骤3 参见**步骤1**～**步骤2**，继续添加依赖的BO，例如：ISDP+集成公共BO和操作日志BO，添加依赖后如**图1-16**所示。

图 1-16 添加的 BO



----结束

1.3.3.3 创建目录

在应用的开发页面，根据规划创建目录，分别用于存放数据对象、服务编排、脚本、事件、前端页面等。

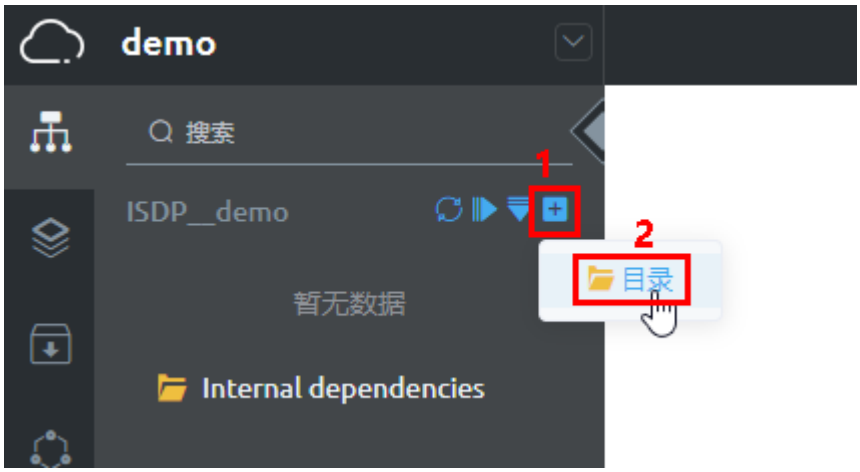
目录配置为：

目录	说明		
前台页面	用于存放前端页面。 控制逻辑	用于存放后台逻辑，如，脚本、服务编排、BPM等。 数据对象	用于存放数据模型（即数据对象）。

操作步骤

步骤1 如图1-17所示，在应用的开发页面，单击应用右侧的，选择“目录”。

图 1-17 创建目录

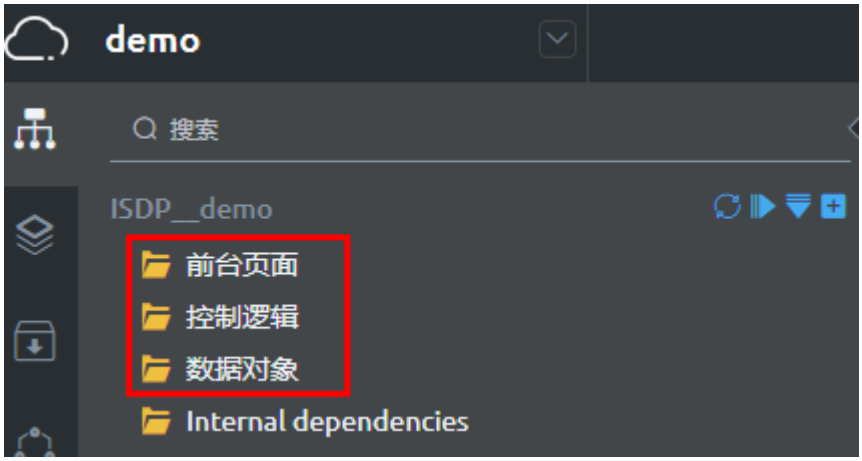


步骤2 在弹出的“添加目录”页面，如图1-18所示，输入目录名称（前台页面），单击“保存”。

图 1-18 添加目录

步骤3 参见**步骤1**～**步骤2**，继续创建目录，创建后的目录如**图1-19**所示。

图 1-19 创建的目录



----结束

1.3.3.4 导入扩展表

智能排班模型BO中，离岗信息表扩展表（ISDP__offDutyInfoExtend__CST）、积假信息扩展表（ISDP__accruedLeaveInfoExtend__CST）、班次信息扩展表（ISDP__shiftInfoExtend__CST）和配套资源扩展信息表（ISDP__resourceExtendInfo__CST）四个表是通过智能排班模型BO的后置脚本预置进去的，在使用智能排班模型BO中的扩展表时，需要先导入扩展表到应用中，再进行配置。

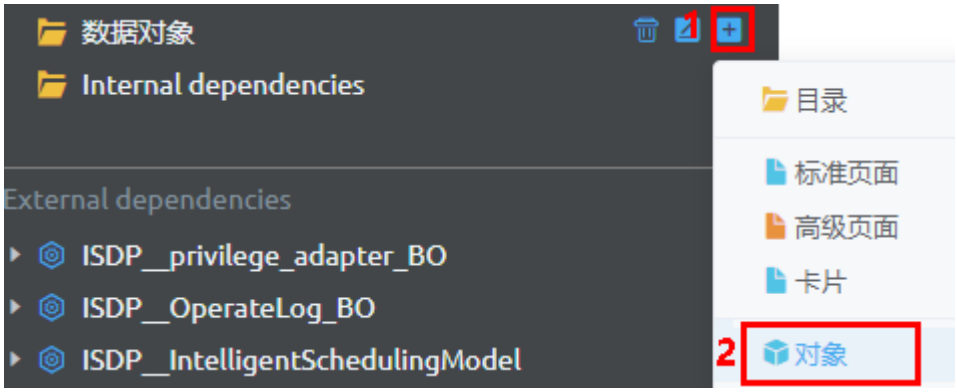
背景信息

- 智能排班模型BO中定义了离岗信息、积假信息和班次信息以及配套资源信息表，但是表中字段固定没办法修改，为了利于开发者定制使用，分别提供了四个表的扩展表，便于开发者自定义字段。智能排班模型BO中开放的四个表的增删改查接口，同时支持对扩展表数据的操作。开发者自定义字段后，可以直接使用BO开发的接口，对表和扩展表进行数据的增删改查操作。
- 四个扩展表中，自定义字段均默认展示：ISDP__deleteFlag__CST（删除标识），以及每个表与主表关联的查找关系类型字段。
  - ISDP__offDutyInfoExtend__CST：查找关系类型字段，ISDP__offDutyInfoId__CST（离岗信息id）
  - ISDP__accruedLeaveInfoExtend__CST：查找关系类型字段，ISDP__accruedLeaveId__CST（积假信息id）
  - ISDP__shiftInfoExtend__CST：查找关系类型字段，ISDP__shiftId__CST（班次信息id）
  - ISDP__resourceExtendInfo__CST：查找关系类型字段，ISDP__resourceInfoId__CST（资源id）

操作步骤

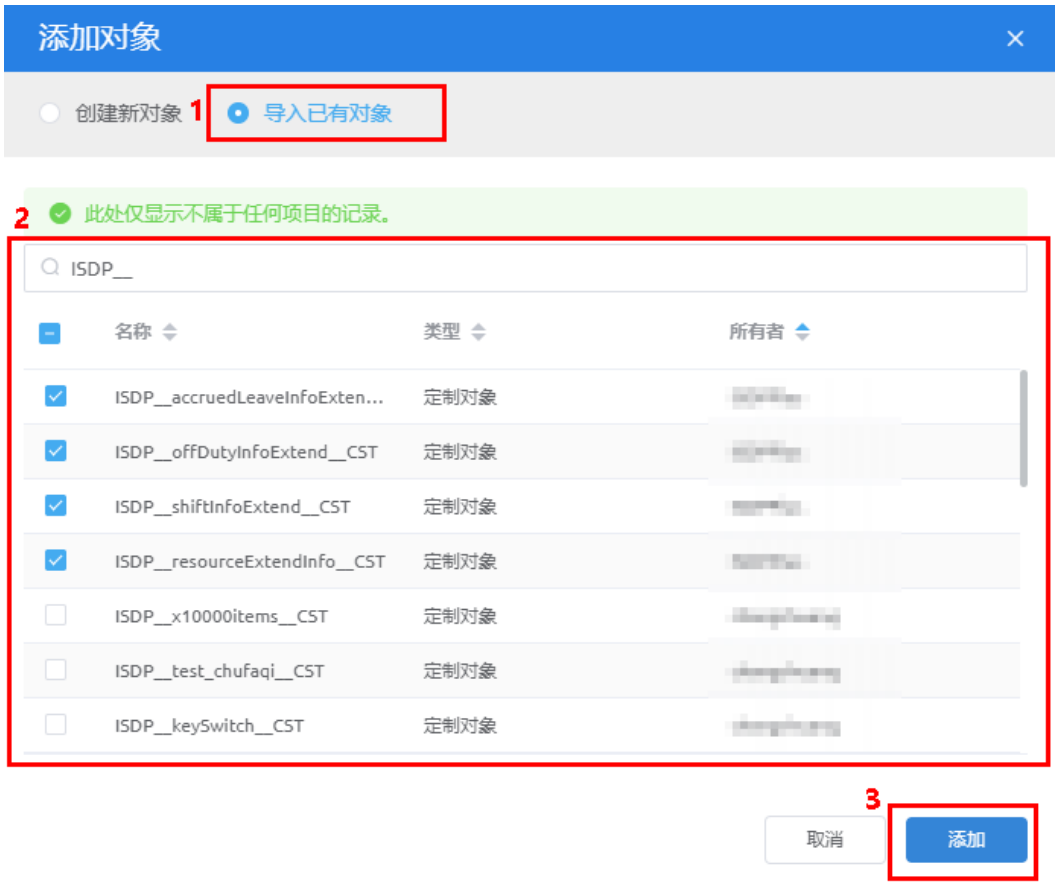
步骤1 如**图1-20**所示，在应用的开发页面，选择存放扩展表的目录（数据对象），单击目录对应的，选择“对象”。

图 1-20 创建对象



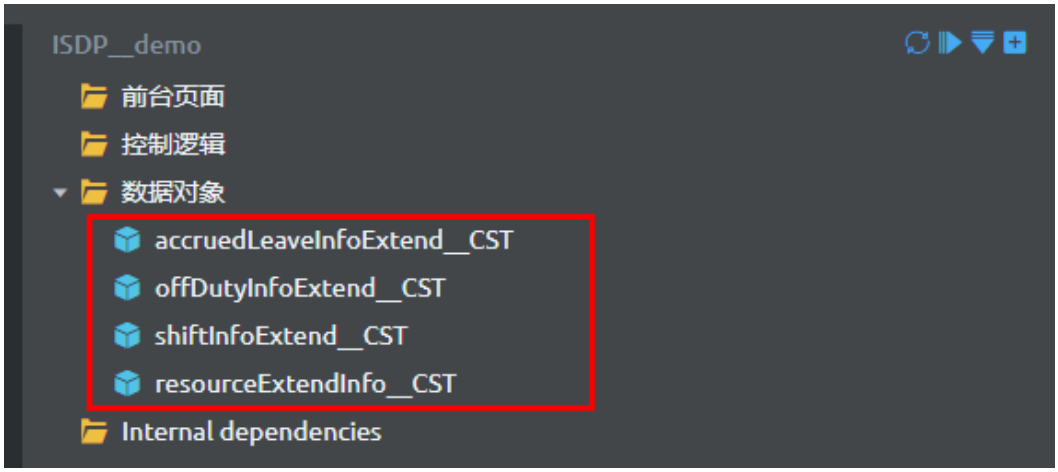
步骤2 如图1-21所示，在弹出的“添加对象”页面，选择“导入已有对象”，搜索查询到智能排班模型BO中的四个扩展表，并勾选，单击“添加”。

图 1-21 添加对象



导入后，在“数据对象”目录下可以查看到导入的对象，如图1-22所示。

图 1-22 导入的对象



----结束

1.3.4 定制开发应用

1.3.4.1 开发对象

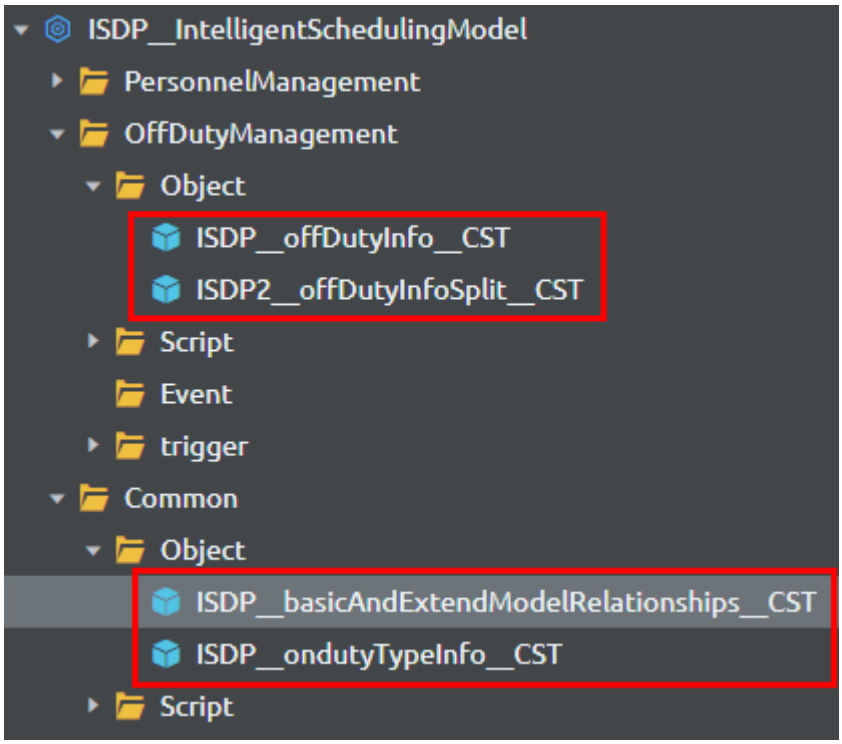
您可以根据业务需求自定义对象，用来存储组织或者业务特有的数据。

背景信息

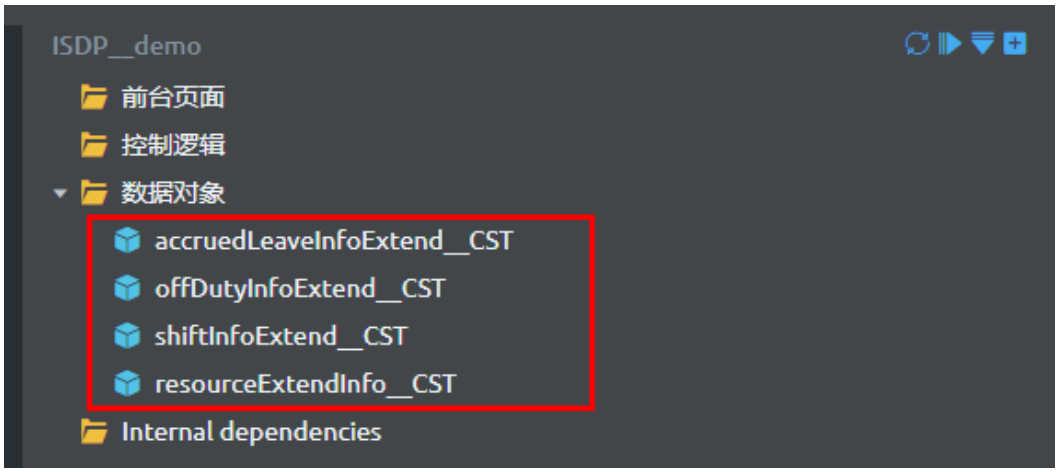
- **对象功能介绍**
  - 对象（也可以称为Object）相当于传统方式开发业务系统时，数据库中创建的一个表。每个对象对应一张数据库表，用于保存业务系统需要的配置数据和业务数据。
  - AppCube平台预置了一部分标准对象（Standard Object），开发者可以引入这些标准对象，标准对象名称和对象字段均已定义好， 只允许用户扩展新字段，禁止修改、删除预置字段。开发者也可以根据自己的业务需要，创建自定义对象（Custom Object），支持增、删、改自定义对象及自定义对象的字段。
  - 创建一个自定义对象后，系统会为自定义对象自动创建一些标准字段（Standard Fields），预置的标准字段，仅name支持修改。
- **对象定制开发说明**

  开发者基于BO定制开发时，可以新创建自定义对象，也可以使用BO中的对象。

  BO中对象说明：
  - BO中展示的对象，可以直接使用，但无法修改。



- BO中开放的4个扩展表（部署时后置脚本预置进去的），导入到应用后，可以根据业务需求定制修改使用。



场景描述

本节以创建一个自定义对象（accruedLeaveApplicationRecord1）为例，描述创建对象的过程。

表 1-3 规划的对象自定义字段

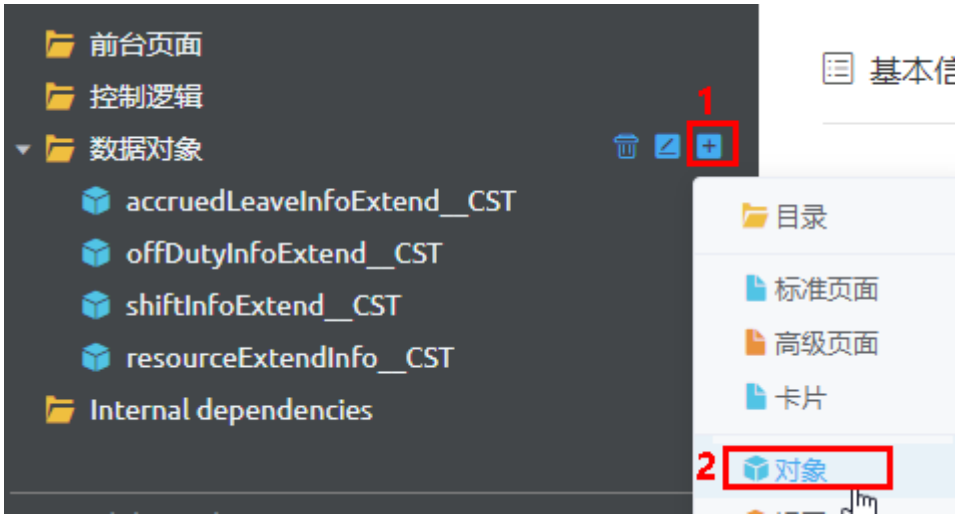
标签	名称	字段类型	取值	读取/编辑权限	添加到页面布局
编码	accruedLeaveCode	文本	数据长度：255	Standard User	选择

标签	名称	字段类型	取值	读取/编辑权限	添加到页面布局
人员账号	personCode			Profile/ Portal User Profile/ Developer Profile	
人员名称	personName				
审批状态	approvalStatus				
班次id	shiftId				
班次名称	shiftName				
召回开始时间	recallStart				
召回结束时间	recallEnd				
本次积假时间	accruedLeaveThisTime				
审批流名称	bpmName				
审批流标签	bpmLable				
审批流实例id	bpmInstanceId				
召回原因	recallReason	文本区	-		
删除标志	deleteFlag	文本	数据长度：255		
提交时间	submitTime				
审批完成时间	completeTime				

创建自定义数据对象

步骤1 如图1-23所示，在应用的开发页面，单击规划存放对象（数据对象）目录右侧的，选择“对象”。

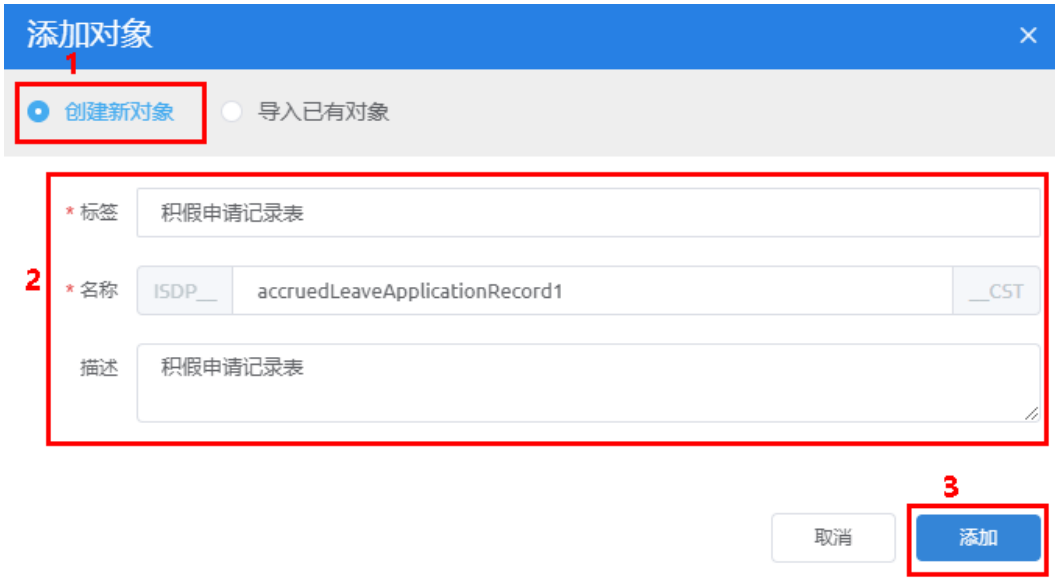
图 1-23 创建对象



步骤2 在弹出的“添加对象”页面，如图1-24所示，选择创建新对象，输入对象的“标签”、“名称”和描述信息，单击“添加”。

添加成功后，进入对象详情页。

图 1-24 添加对象



----结束

定义自定义对象的字段

步骤1 新建自定义字段。

1. 在对象详情页，如图1-25所示，单击“自定义字段”页签，单击“新建”。

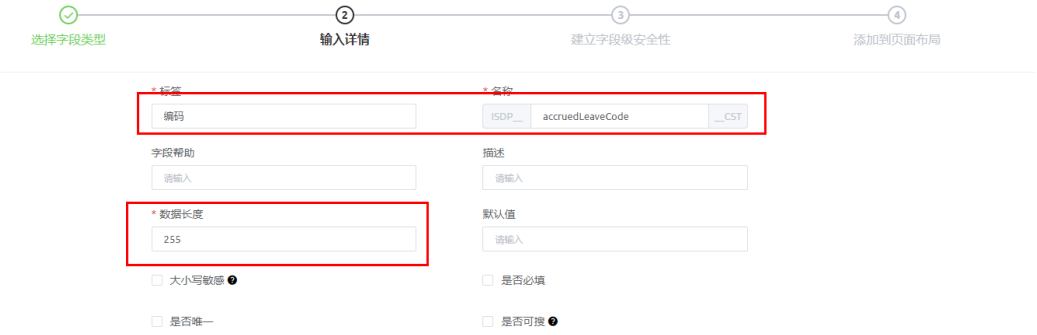
图 1-25 新建自定义字段

2. 如图1-26所示，选择字段类型文本，单击“下一步”

图 1-26 选择字段类型

3. 如图1-27所示，设置标签为“编码”、名称为“accruedLeaveCode”，数据长度为“255”，单击“下一步”。

图 1-27 输入详情



4. 如图1-28所示，设置字段的访问权限，即设置哪些权限集可以查看或修改该字段，单击“下一步”。

图 1-28 建立字段级安全性

5. 如图1-29所示，设置字段是否加入布局。该配置用于将字段加入到对象默认的布局页面。

图 1-29 添加到页面布局

6. 单击“保存”，创建字段。
7. 参见**步骤1.1**～**步骤1.6**，继续新建字段，完成规划字段的创建，示例如**图1-30**所示。

图 1-30 规划的字段示例

说明

新建字段时，可以单个新建，也可以批量新建（批量新建时，下载模板输入相关信息，再导入）。

步骤2 （**可选，当需要给对象创建索引字段时，请执行该步骤**）添加自定义索引。

说明

本示例场景不涉及。

1. 如**图1-31**所示，在“自定义字段”页签单击“自定义索引”。

图 1-31 自定义索引

2. 如图1-32所示，在“自定义索引”页面单击“添加自定义索引”。

图 1-32 自定义索引页面

3. 如图1-33所示，定义索引，单击“创建”。

图 1-33 添加自定义索引

表 1-4 参数说明

参数	配置说明
名称	自定义索引的名称。
类型	索引类型，支持如下两种类型： – 普通索引 – 唯一索引 唯一索引字段的字段取值必须唯一。
字段*1*	从下拉框选择已创建的自定义字段作为索引字段。 默认最多只能创建一个索引，每个索引中最多创建两个索引字段。

----结束

为对象字段增加校验器（可选，当需要给对象增加校验器时，请执行该步骤）

说明

本示例场景不涉及。

在向对象中录入记录数据时，为了保证对象数据的有效性，可以定义一些校验规则，对字段值以及各字段之间的逻辑关系进行校验。

使用校验器主要是为了方便对字段做一些简单的规则限制。

步骤1 在对象详情页面，如图1-34所示，单击“验证规则”页签，再单击“新建”。

图 1-34 验证规则

步骤2 如图1-35所示，输入校验规则，单击“保存”。

图 1-35 新建验证规则

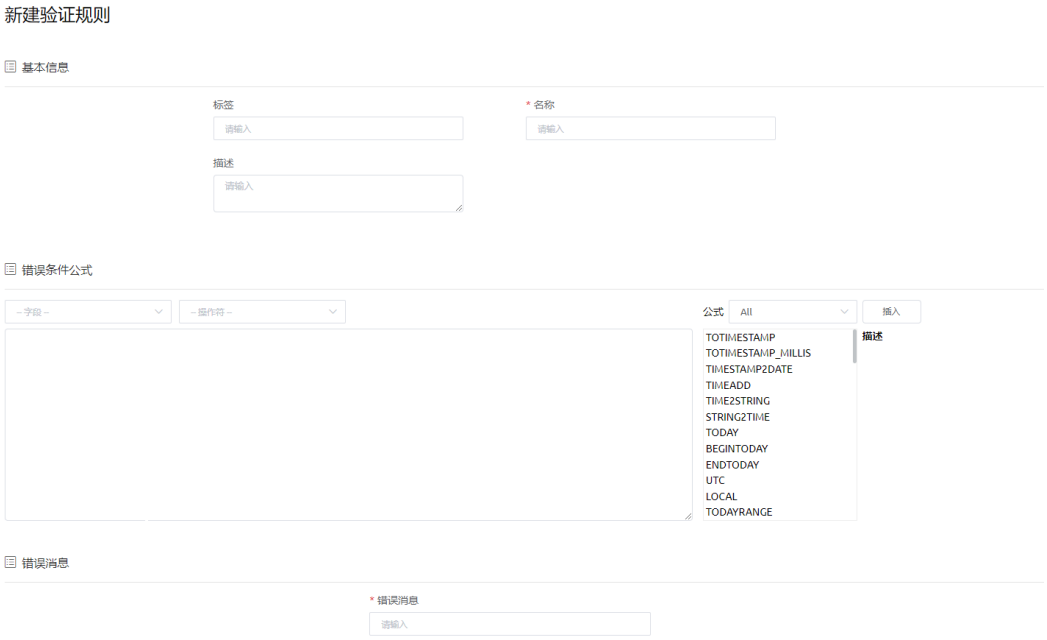


表 1-5 参数说明

参数	配置说明
基本信息	• 标签：Validator标签名，用于展现在配置界面。 • 名称：Validator名称。 • 描述：描述信息。
错误条件公式	配置校验公式时，配置的是错误条件。 规则由公式、字段、运算符组成： • 单击“字段”可选择逻辑表达式中涉及字段。 • 单击“操作符”可选择表达式中的运算符。 • 从“公式”下拉列表中选择公式，单击“插入”，可在表达式中嵌入公式。选中公式会有注释说明。
错误消息	错误提示信息，可手动输入。

步骤3 在校验规则列表页面，在新建校验规则的“操作”列打开开关，启用该条规则。

----结束

为对象配置内嵌触发器（可选，当需要给对象内嵌触发器时，请执行该步骤）

说明

本示例场景不涉及。

开发者可以为对象增加内嵌触发器，实现在增/删/改对象的记录数据时，自动执行内嵌触发器脚本（该脚本由用户自定义，其实就是一段普通的TypeScript脚本，没有输入输

出参数）。比如在对象记录插入前、记录插入后、记录更新前、记录更新后、记录删除前或者记录删除后自动执行内嵌触发器脚本。

步骤1 在对象详情页面，如图1-36所示，单击“内嵌触发器”页签，再单击“新建”。

图 1-36 内嵌触发器

步骤2 如图1-37所示，输入触发器的基本信息，单击“保存”。

图 1-37 创建触发器

表 1-6 参数说明

参数	配置说明
标签	内嵌触发器在用户界面展示的名称。
名称	内嵌触发器的名称

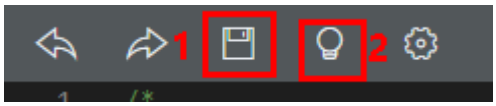
参数	配置说明
事件触发时机	内嵌触发器在什么时候生效，可直接在下拉框中选择。即在系统执行如下某动作时，自动执行触发器的脚本： • 插入前：记录插入操作前触发。 • 修改前：记录更新操作前触发。 • 删除前：记录删除操作前触发。 • 插入后：记录插入操作后触发。 • 修改后：记录更新操作后触发。 • 删除后：记录删除操作后触发。
描述	关于内嵌触发器的描述，不超过255个字节。

步骤3 编写触发器的脚本。

对象的内嵌触发器脚本，其实就是一段普通的TypeScript脚本，没有输入输出参数，关于脚本的开发说明，请参加**开发脚本**。

步骤4 如**图1-38**所示，保存脚本并启用脚本。

图 1-38 保存并启用脚本



启用后在内嵌触发器列表可查看到当前触发器状态为“ON”，表示当触发条件满足时，会自动执行该触发器脚本。

----结束

1.3.4.2 开发脚本

针对业务逻辑比较复杂的场景，可以使用脚本（Script）能力，在线开发TypeScript脚本，完成灵活复杂的业务逻辑。

背景信息

平台的脚本引擎采用TypeScript语言；脚本执行时，TypeScript语言会被翻译成JavaScript语言，由JavaScript引擎执行。

在JavaScript es5的官方标准库外，平台还扩展了10+预置标准库（即预置API），帮助您更高效地开发脚本，有关系统预置的标准库说明请参见**脚本中预置的API**。

场景描述

本节以基于**开发对象**中创建的对象，开发一个新脚本（addAccruedLeaveRecord1），用于添加对象的数据为例，描述开发脚本的过程。其中，脚本中引入智能排班模型BO中脚本的方法记录操作日志。

操作步骤

步骤1 创建脚本。

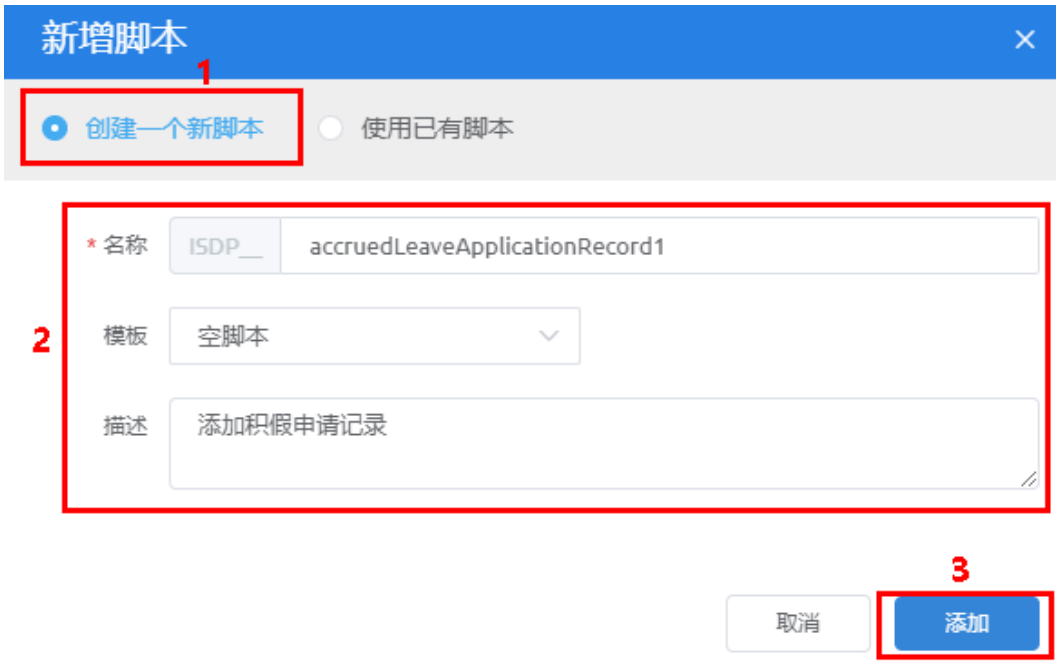
1. 如**图1-39**所示，在应用的开发页面，选择存放脚本的目录（控制逻辑），单击目录对应的，选择“脚本”。

图 1-39 创建脚本

2. 在弹出的“新增脚本”页面，如**图1-40**所示，选择创建一个新脚本，输入脚本“名称”为“addAccruedLeaveRecord1”，“模板”选择“空脚本”，单击“添加”。

   创建完成后，自动进入编辑页面。

图 1-40 新增脚本



步骤2 编辑脚本。

脚本示例如下，脚本的说明请参见示例的备注内容。

```
//引入该脚本依赖的标准库或者其他模块。db、date是系统预置的标准库；ISDP__commApiForAppcube是引入的智能排班模型BO中一个脚本。
import * as db from 'db';              //引入处理对象的标准库
import { format, now } from 'date';    //引入处理时间的标准库中的时间转换为字符串和获取当前时间
import AppCubeApi from './ISDP__commApiForAppcube';   //引入脚本中的方法AppCubeApi

//实例化
const appcube = new AppCubeApi();
const Func = "batchAddOperationInfo";
const Success = 'success';
const Add = 'add';

//定义入参结构，所有的输入参数必须封装在一个class中，作为实例成员，示例中被封装为Input。
@action.object({ type: "param" })
export class Input {
    @action.param({ type: "String", description: "人员账号", required: false })
    personCode: string;
    @action.param({ type: "String", description: "人员名称", required: false })
    personName: string;
    @action.param({ type: "String", description: "班次id", required: false })
    shiftId: string
    @action.param({ type: "String", description: "班次名称", required: false })
    shiftName: string;
    @action.param({ type: "String", description: "召回开始时间", required: true })
    recallStart: string;
    @action.param({ type: "String", description: "召回结束时间", required: true })
    recallEnd: string;
    @action.param({ type: "String", description: "审批流名称", required: true })
    bpmName: string
    @action.param({ type: "String", description: "审批流标签", required: true })
    bpmLable: string;
    @action.param({ type: "String", description: "审批流实例id", required: true })
    bpmInstanceId: string;
    @action.param({ type: "String", description: "召回原因", required: false })
    recallReason: string;
}
```

```
//定义出参结构，所有的输出参数必须封装在一个class中，作为实例成员，示例中被封装为Output。
@action.object({ type: "param" })
export class Output {
    @action.param({ type: "String", description: "错误码", isCollection: false })
    statusCode: string;
    @action.param({ type: "String", description: "错误描述", isCollection: false })
    message: string;
}

//定义该脚本中使用到的ISDP__accruedLeaveApplicationRecord1__CST对象
@useObject(['ISDP__accruedLeaveApplicationRecord1__CST'])
//定义方法，进行数据库操作
//示例中，使用action.method函数标识Action的调用方法AddAccruedLeaveRecord.run，表明调用脚本时从此方法入口，其输入、输出参数就是前面定义的Input和Output。
//在一个脚本文件里面，action.method只能使用一次
@action.object({ type: "method" })
export class AddAccruedLeaveRecord {
    @action.method({ input: "Input", output: "Output", description: "添加积假申请记录" })
    run(input: Input): Output {
        let output = new Output();
        try {
            let s = db.dynamicObject('ISDP__accruedLeaveApplicationRecord1__CST');
            let start = new Date(input.recallStart);
            let end = new Date(input.recallEnd);
            let accruedLeaveThisTime = (end.getTime() - start.getTime()) / 10000 / 60 / 6;
            console.log("accruedLeaveThisTime", accruedLeaveThisTime)
            let date = new Date();
            let accruedLeaveCode = date.getFullYear() + "" + (date.getMonth() + 1) + "" + date.getDate() + ""
+ date.getHours() + "" + date.getMinutes() + "" + date.getSeconds() + "" + date.getMilliseconds();
            console.log("date", accruedLeaveCode)
            //insert内容
            let record = {
                "ISDP__personCode__CST": input.personCode,
                "ISDP__personName__CST": input.personName,
                "ISDP__shiftId__CST": input.shiftId,
                "ISDP__shiftName__CST": input.shiftName,
                "ISDP__recallStart__CST": input.recallStart,
                "ISDP__recallEnd__CST": input.recallEnd,
                "ISDP__accruedLeaveThisTime__CST": accruedLeaveThisTime.toFixed(2) + "",
                "ISDP__accruedLeaveCode__CST": accruedLeaveCode,
                "ISDP__bpmName__CST": input.bpmName,
                "ISDP__bpmLable__CST": input.bpmLable,
                "ISDP__bpmInstanceId__CST": input.bpmInstanceId,
                "ISDP__recallReason__CST": input.recallReason,
                "ISDP__approvalStatus__CST": "0",
                "ISDP__submitTime__CST": format(date, 'yyyy-MM-dd HH:mm:ss'),
                "ISDP__completeTime__CST": "-",
                "ISDP__deleteFlag__CST": "0"
            }
            s.insert(record);
            appcube.addOperLog(Add, Func, Success, '新增积假申请记录成功',
'ISDP__accruedLeaveApplicationRecord1__CST');  //调用引入的AppCubeApi方法，记录操作日志
            output.statusCode = "0";
            output.message = "成功";
        } catch (error) {
            output.statusCode = "1";
            output.message = "失败";
        }
        return output;
    }
}
```

步骤3 保存脚本。

如图1-41所示，单击页面上方的  ，保存脚本。

图 1-41 保存脚本

步骤4 测试脚本。

1. 如图1-42所示，单击页面上方的  ，在页面底部显示测试窗口，如图1-43所示。

图 1-42 运行测试脚本

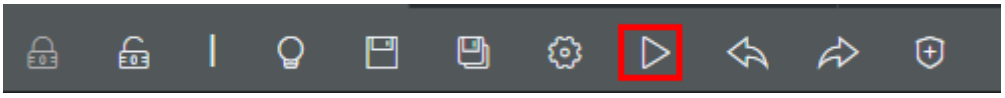
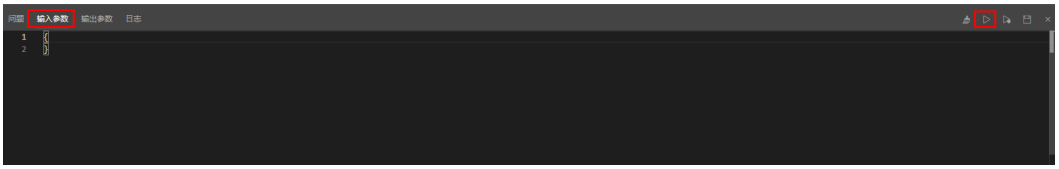


图 1-43 测试窗口



2. 在测试窗口中，“输入参数”页签下，输入入参，单击  。

   入参示例：

```
{
   "personCode":"111",
   "personName":"chen",
   "shiftId":"222",
   "shiftName":"yong",
   "recallStart":"2023-03-01",
   "recallEnd":"2023-03-02",
   "bpmName":"admin",
   "bpmLable":"a",
   "bpmInstanceId":"333",
   "recallReason":"remark"
}
```

3. 查看“输出参数”页签下，结果是否符合预期。

```
{
   "message": "成功",
   "statusCode": "0"
}
```

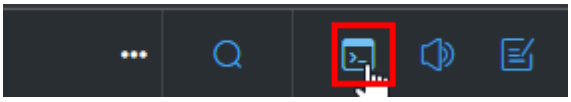
4. 查看“日志”页签下，输出日志。

```
0330 10:44:28.362|debug|vm[1795]>>> Build #AppCube Core 23.2.0spc1 amd64
Built on 2023-03-09 20:10:11
Commit #80465c38e2
0330 10:44:28.364|debug|vm[1795]>>> node:  28
0330 10:44:28.365|debug|vm[1795]>>> script:  ISDP__addAccruedLeaveRecord1__1.0.1
AddAccruedLeaveRecord.run
0330 10:44:28.367|debug|vm[1795]>>> locale:  zh_CN
0330 10:44:28.368|debug|vm[1795]>>> os timezone:  Local
0330 10:44:28.370|debug|vm[1795]>>> user timezone:  Local
0330 10:44:28.371|debug|vm[1795]>>> organization timezone:  (GMT+08:00) China Standard Time
(Asia/Shanghai)
0330 10:44:28.373|debug|vm[1795]>>> load module ISDP__addAccruedLeaveRecord1__1.0.1
(ISDP__addAccruedLeaveRecord1_wrapper.ts:2)
0330 10:44:28.374|debug|vm[1795]>>> load module db (ISDP__addAccruedLeaveRecord1.ts:2)
0330 10:44:28.376|debug|vm[1795]>>> load module date (ISDP__addAccruedLeaveRecord1.ts:3)
0330 10:44:28.377|debug|vm[1795]>>> load module ISDP__commApiForAppcube__1.1.0
(ISDP__addAccruedLeaveRecord1.ts:4)
0330 10:44:28.378|debug|vm[1795]>>> load module http (ISDP__commApiForAppcube.ts:3)
```

```
0330 10:44:28.380|debug|vm[1795]>>> load module ISDP__ErrorCodeList__1.1.0
(ISDP__commApiForAppcube.ts:6)
0330 10:44:28.381|debug|vm[1795]>>> load module ISDP__AddLog__1.0.1
(ISDP__commApiForAppcube.ts:7)
0330 10:44:28.382|debug|vm[1795]>>> load module ISDP__validator__1.0.1 (ISDP__AddLog.ts:3)
0330 10:44:28.384|debug|vm[1795]>>> load module context (ISDP__validator.ts:2)
0330 10:44:28.385|debug|vm[1795]>>> load module sys (ISDP__validator.ts:5)
0330 10:44:28.387|debug|vm[1795]>>> accruedLeaveThisTime 24 (ISDP__addAccruedLeaveRecord1.ts:
61)
0330 10:44:28.389|debug|vm[1795]>>> date 2023330104428389 (ISDP__addAccruedLeaveRecord1.ts:
64)
0330 10:44:28.421|debug|vm[1795]>>> resource limit usage:
    number of RequestSOQLQueriesAmount: 0
    number of RequestSOQLRowsAmount: 0
    number of RequestSOSLQueriesAmount: 0
    number of RequestSOSLRowsAmount: 0
    number of RequestDMLStatementsAmount: 2
    number of RequestDMLRowsAmount: 2
    number of RequestCallOutsAmount: 0
    number of RequestEmailSendsAmount: 0
    number of RequestEventSendsAmount: 0
```

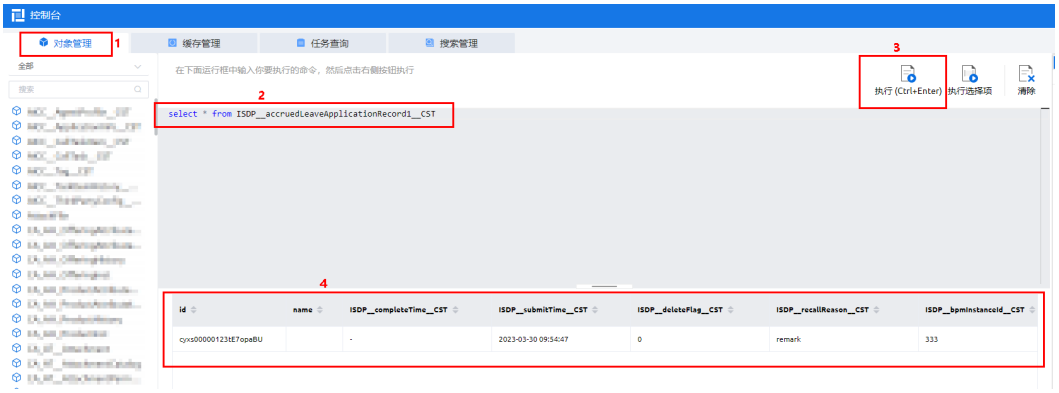
5. 如图1-44所示，单击右上角  ，进入控制台。

图 1-44 控制台



6. 如图1-45所示，在“对象管理”页签中，输入查询语句（select * from ISDP__accruedLeaveApplicationRecord1__CST），查询数据。

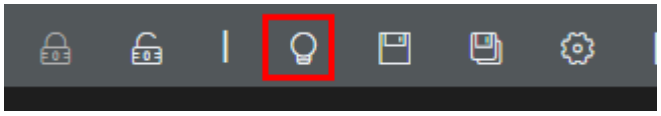
图 1-45 查询数据



步骤5 启用脚本。

如图1-46所示，单击页面上方的  ，启用脚本。

图 1-46 启用脚本



说明

脚本开发完成后，可以封装成一个接口，供调用，具体请参见**如何自定义调用脚本的URL**。

----结束

1.3.4.3 开发服务编排

服务编排(Flow)是基于图元拖拽式开发业务逻辑的一种方式，分为两类：Autolauched Flow和 Event Trigger。

- Autolaunched Flow：自启动Flow，在接口调用后会立即执行Flow模型定义的逻辑。
- Event Trigger：事件触发的Flow，则会在事件触发时才会开始执行Flow模型定义的逻辑。

背景信息

服务编排是一种通过简单的拖拉拽式流程编排以及参数配置的方式来进行服务开发的能力。开发者能够在服务编排编辑器内以图形化编排的形式快速地进行服务的开发并扩展出更丰富的业务功能，同时能够与API接口进行绑定，以API的形式对外提供服务。

通过服务编排，也可以将已实现的脚本、服务编排等功能进行复用，只需要进行图形化编排以及相关参数配置，即可针对您自己的独特业务需求并以流程的方式将业务需求所要实现的功能展现出来，甚至不需要有任何编程经验即可完成服务的开发，降低了开发难度提高了开发效率。

场景描述

本节以调用**开发脚本**中的脚本（addAccruedLeaveRecord1）开发一个服务编排为例，描述开发服务编排（addAccruedLeaveRecord）的过程。

操作步骤

步骤1 创建服务编排。

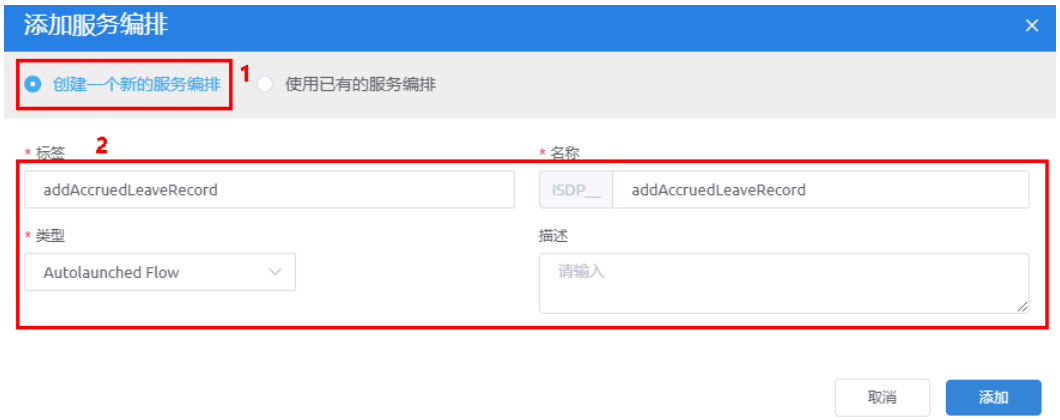
1. 如**图1-47**所示，在应用的开发页面，选择存放服务编排的目录（控制逻辑），单击目录对应的  ，选择“服务编排”。

图 1-47 创建服务编排

2. 在弹出的“添加服务编排”页面，如图1-48所示，选择创建一个新的添加服务编排，输入“标签”和“名称”为“addAccruedLeaveRecord”，“类型”选择“Autolauched Flow”，单击“添加”。

   创建完成后，自动进入编辑页面。

图 1-48 添加服务编排



步骤2 创建该服务编排的入参和出参变量。由于该服务编排封装的是脚本，则该服务编排的入参和出参的字段类型和字段名与脚本保持一致。

表 1-7 服务编排变量

量类型	参数名	Data Type	Input/Output Type
普通变量 Variable	personCode	文本	Input Only
	personName	文本	Input Only

量类型	参数名	Data Type	Input/Output Type
	shiftId	文本	Input Only
	shiftName	文本	Input Only
	recallStart	文本	Input Only
	recallEnd	文本	Input Only
	bpmName	文本	Input Only
	bpmLable	文本	Input Only
	bpmInstanceId	文本	Input Only
	recallReason	文本	Input Only
	statusCode	文本	Output Only
	message	文本	Output Only

1. 如图1-49所示，在服务编排编辑器页面右侧单击 ，在全局上下文页面单击“变量”后的加号。

图 1-49 全局上下文

2. 如图1-50所示，单击新增变量后的 ，选择“设置”。

图 1-50 设置变量

3. 如图1-51所示，配置普通变量“personCode”，参数配置说明请参见表1-8，配置完成后单击“保存”。

图 1-51 配置普通变量

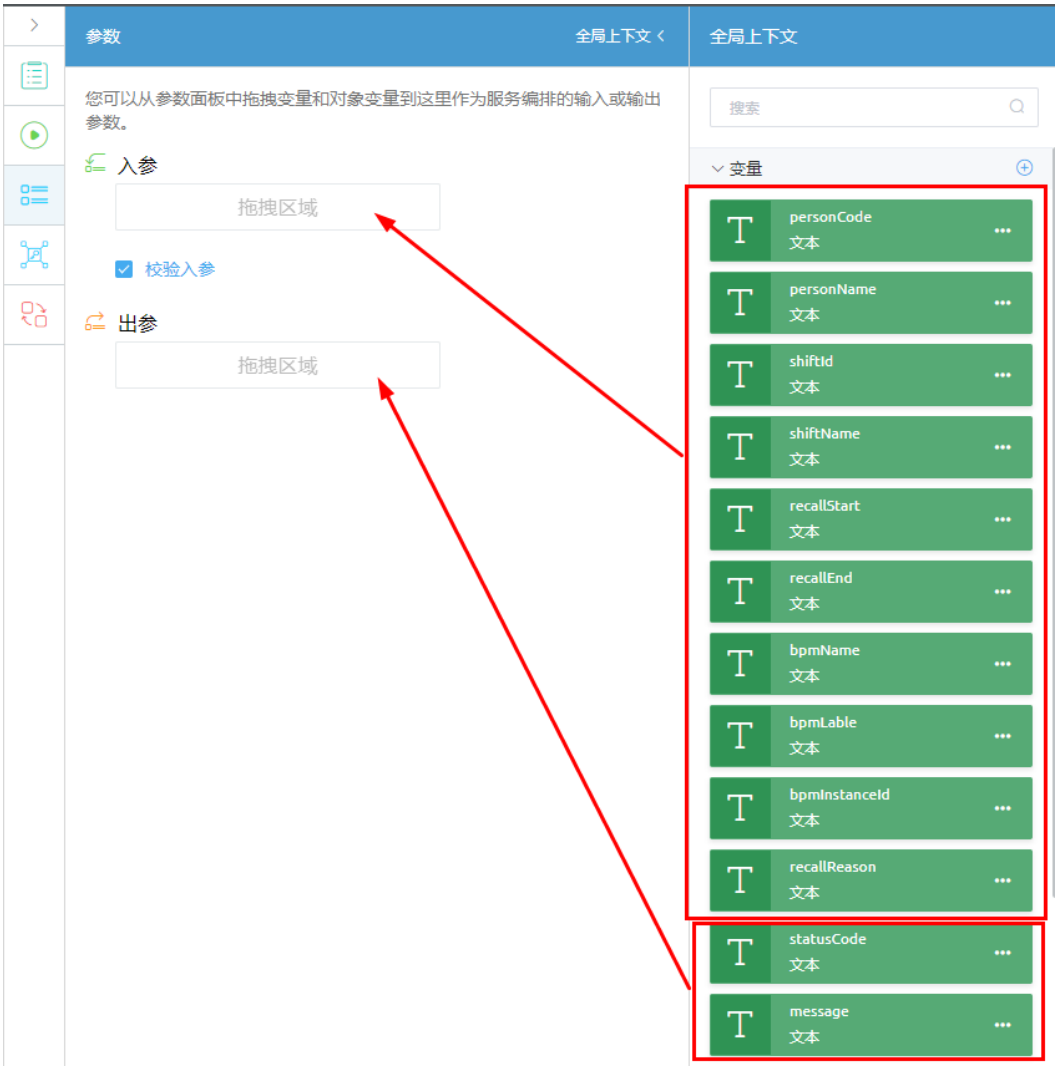
表 1-8 参数说明

参数	参数说明	如何配置
名称	变量名称，必填。	直接输入“personCode”。
数据类型	变量的数据类型。必填。支持以下几种。 - 文本 - 数字 - 货币 - 日期 - 日期/时间 - 复选框	从下拉列表选择“文本”。
默认值	变量的默认取值。	不用配置。
描述	该变量的描述说明。	选填项，建议填写该变量的作用。
是否为数组	是否为数组型变量即集合变量。	不勾选。

4. 参考**步骤2.1**~**步骤2.3**配置**表1-7**中其他普通变量。

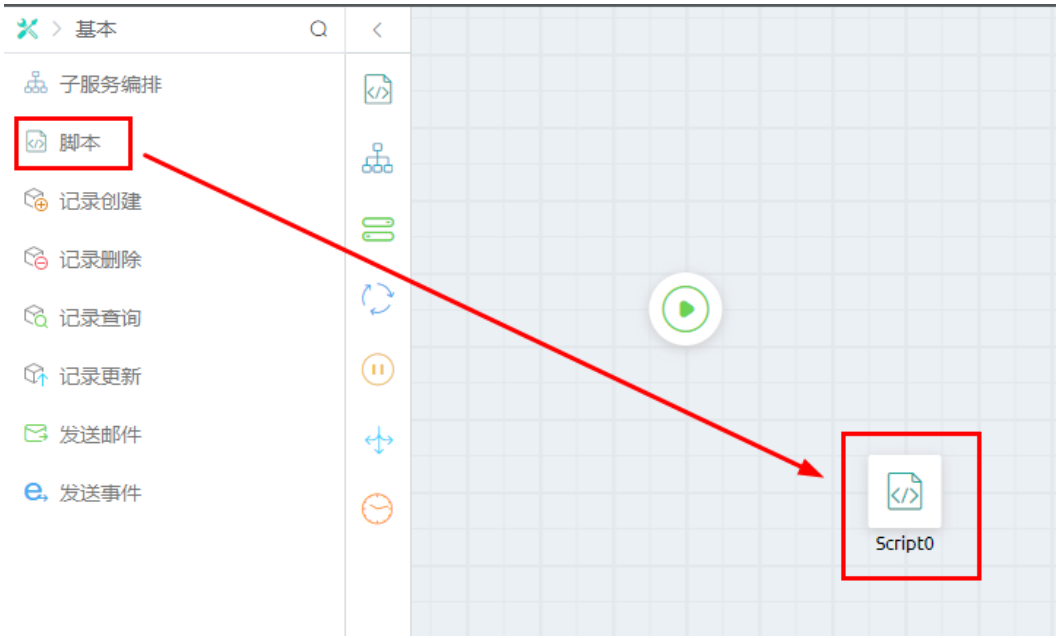
5. 如**图1-52**所示，在服务编排编辑器页面右侧单击 ，按照**表1-7**从全局上下文页面中拖曳参数到相应的入参和出参区域。

图 1-52 配置出入参



6. 如图1-53所示，拖拽“基本”下“脚本”图标至画布中，松开鼠标左键。

图 1-53 添加脚本网元

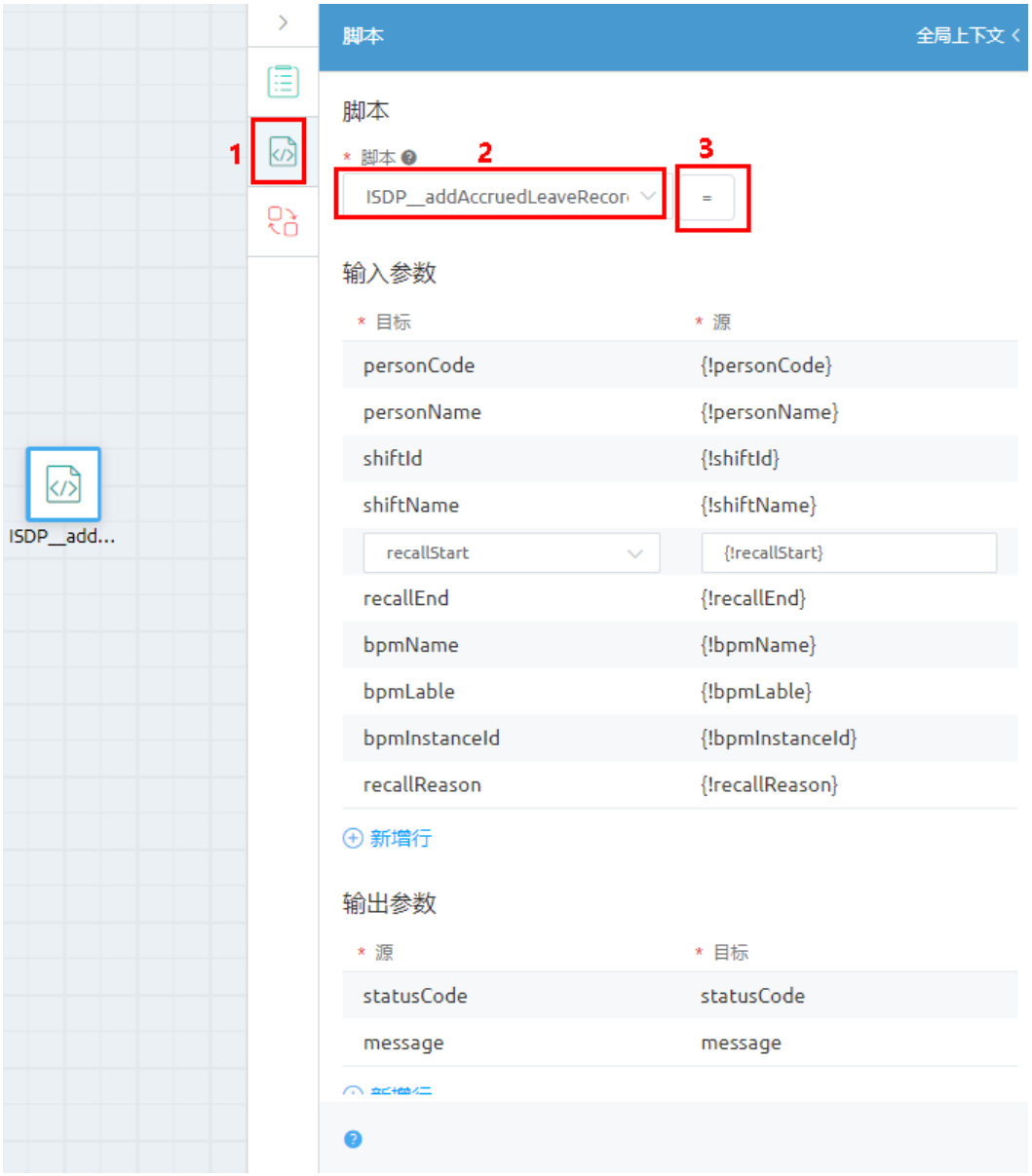


7. 如图1-54所示，选择脚本，单击  ，配置网元基本信息。

图 1-54 配置网元基本信息

8. 如图1-55所示，单击  ，选择调用的脚本，单击 ，自动映射同名出入参。

图 1-55 配置调用的脚本



9. 如图1-56所示，连接元素。

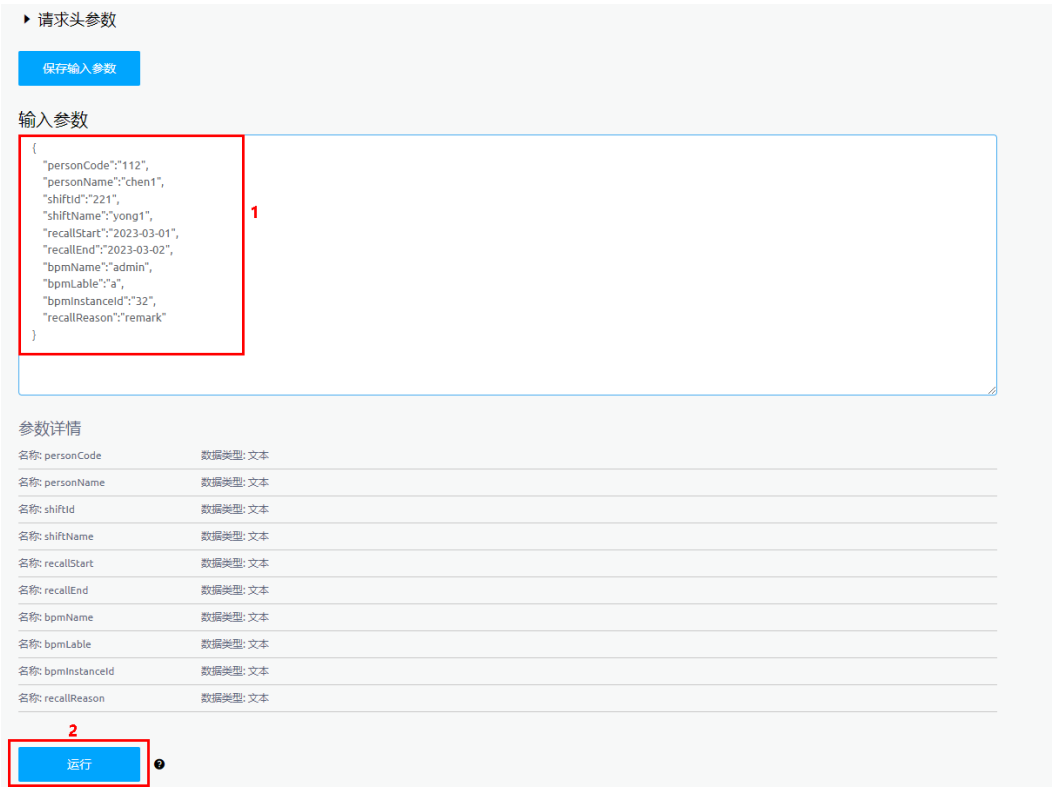
图 1-56 连接元素

10. 单击页面上方的，保存服务编排。

步骤3 调试服务编排。

如图1-57所示，单击页面上方的，输入参数后单击“运行”。

图 1-57 运行



如图1-58所示，检查输出页签的调试结果是否符合预期。

图 1-58 输出结果

步骤4 启用服务编排。

单击页面上方的  ，启用服务编排。

说明

编排服务开发完成后，可以封装成一个接口，供调用，具体请参见**如何自定义调用服务编排的URL**。

----结束

1.3.4.4 开发 BMP

BPM全称为Business Process Management，即业务流程管理，源自业界BPMN 2.0标准。

与服务编排的区别

与服务编排类似，BPM也是一套图形化的流程编排引擎，但是BPM着重于构建带有用户交互行为的业务流程，例如审批流、工单派发流程等。

BPM与服务编排有以下三点核心区别：

- BPM描述的是用户交互流程，使用用户任务将人和页面进行了关联，而服务编排描述的则是单次的业务操作。
- BPM内置实现了顺序审批、或签、会签、投票等任务分派规则，可基于用户任务实现复杂的任务分派规则，应对复杂的交互流程。
- BPM对每次用户交互都会产生流程历史并可以在流程历史中查看，对于非交互类流程这是非必要的。

以上也是BPM更适合制作交互式流程的原因。人工交互流程需要页面的结合、需要对复杂的分配规则进行封装，需要能够进行事后审计的能力。而对于服务编排是不必要的，并且，记录日志反而可能会造成不必要的性能损失。

因此，结合两者的不同场景，BPM可与服务排结合进行使用：服务编排用于实现系统的具体逻辑操作，BPM通过调用服务编排、关联用户任务与页面，实现多次人机交互的流程，例如审批流、工单分发等场景。

场景描述

本示例基于BPM提供的出差申请模板，针对员工出差场景（即员工在出差前需要提交一个出差申请审批的电子流程，员工提交出差申请后，主管处理审批或拒绝提交人申请）为例，描述BPM的开发过程。

本示例中的出差审批应用主要包括如下功能：

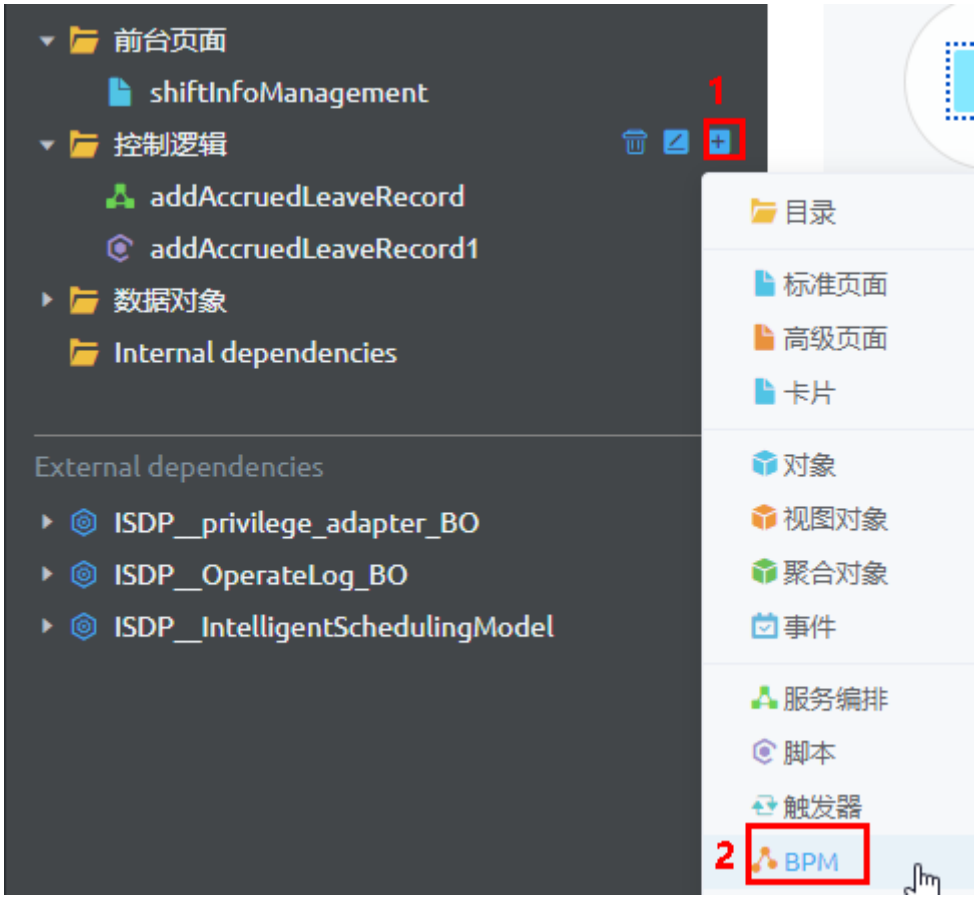
- 基于工作流模板创建出差电子流。
- 发送邮件。

操作步骤

步骤1 创建BPM。

1. 如图1-59所示，在应用的开发页面，选择存放BPM的目录（控制逻辑），单击目录对应的，选择“BPM”。

图 1-59 创建 BPM

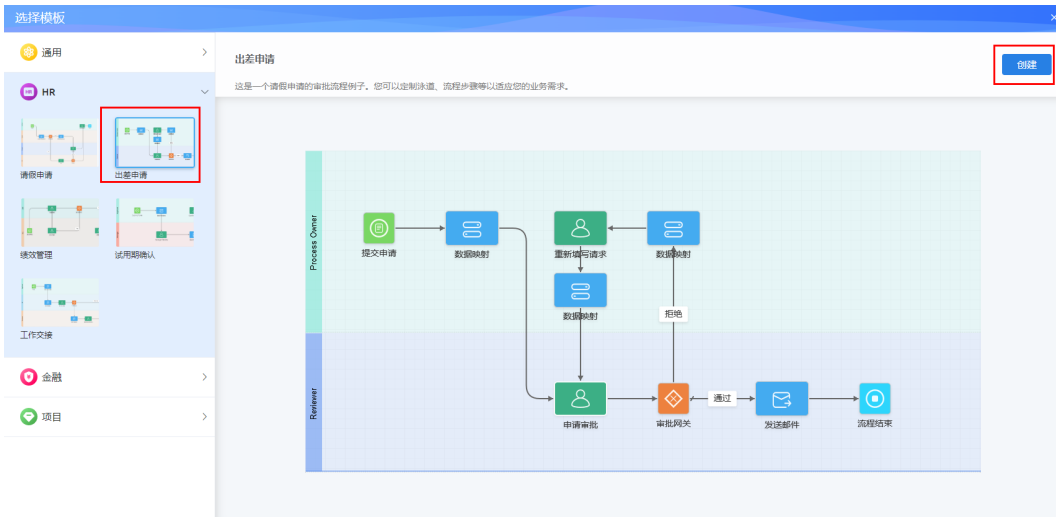


2. 在弹出的“添加工作流”页面，如图1-60所示，选择基于模板，输入“标签”和“名称”为“approvel”，单击“选择模板”。

图 1-60 添加工作流

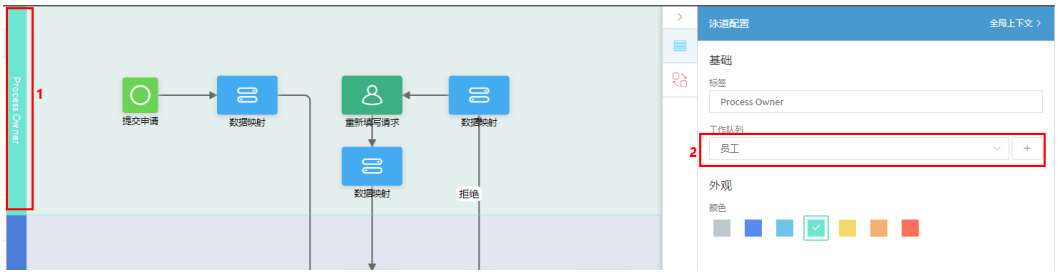
3. 如图1-61所示，在选择模板页面，选择模板，单击“创建”，创建后进入到编辑状态。

图 1-61 选择模板

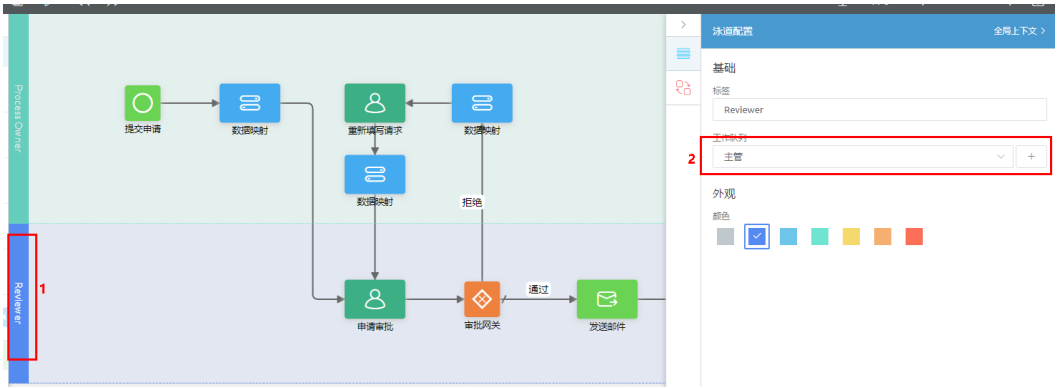


步骤2 设置工作流，配置各泳道处理人。

1. 在工作流开发页面，单击“Process Owner”泳道，配置工作队列为“员工”。

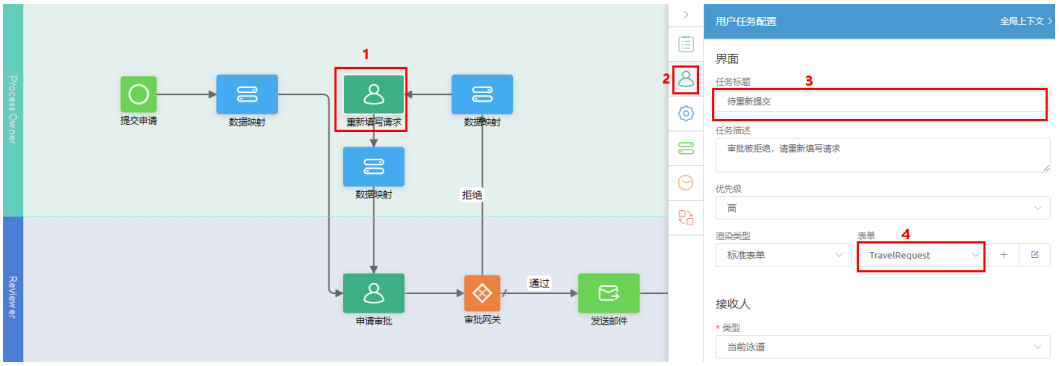


2. 单击“Reviewer”泳道，配置工作队列为“主管”。



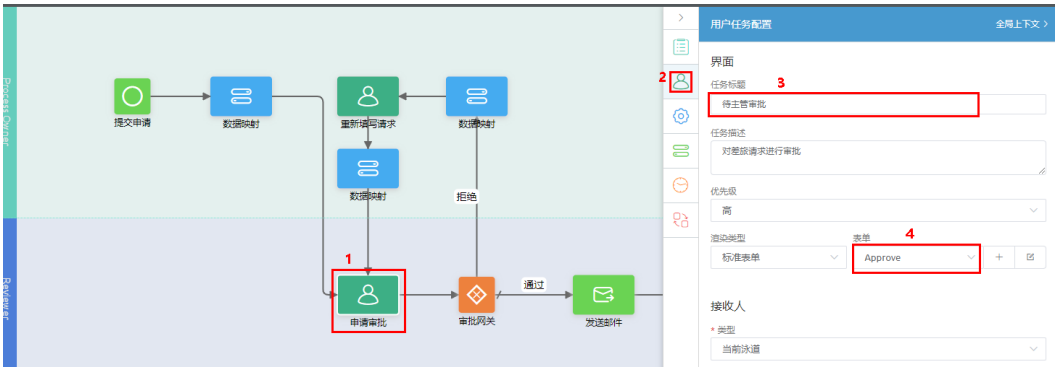
3. 单击泳道“Process Owner”上的“重新填写请求”用户任务元素，设置任务标题为“待重新提交”。

   执行该操作的目的是，后续在“我的待办”中可以看到“待重新提交”的任务状态。



4. 单击泳道“Reviewer”上的“申请审批”用户任务元素，设置任务标题为“待主管审批”。

   执行此操作的目的是，后续在“我的待办”中可以看到“待主管审批”的任务状态。



5. 单击  ，保存工作流。
6. 单击  ，启用工作流。

----结束

1.3.4.5 开发页面

1.3.4.5.1 标准页面

标准页面是一种将一个或多个组件拖进画布，进行低代码甚至无代码的配置，即可快速完成业务功能的前端页面，其功能主要是针对业务数据的增、删、改、查，且前端界面的样式相对简单的页面。

标准页面提供了丰富的组件，组件包含了预置的样式，并封装了基础事件代码，实现了开箱即用。

场景描述

本节以使用智能排班模型BO中班次信息表（ISDP__shiftInfo__CST）开发一个标准页面（班次信息管理页面）为例，描述标准页面的开发过程。

班次信息管理页面，实现班次信息的新增、修改、删除和查询。

页面上默认显示应用中保存的所有班次信息，可以直接新增、修改或者删除班次信息，也可以查询部分班次。

- 单击“新增行”按钮，可以在界面上插入一个空行，输入内容后单击“保存”，即可新插入一条班次记录。
- 直接编辑表格中任意内容，单击“保存”，即可修改任意一条班次记录。
- 选中记录，单击“删除”，即可删除任意一条班次记录。
- 设置查询条件，单击“查询”，可以查询满足条件的班次记录。

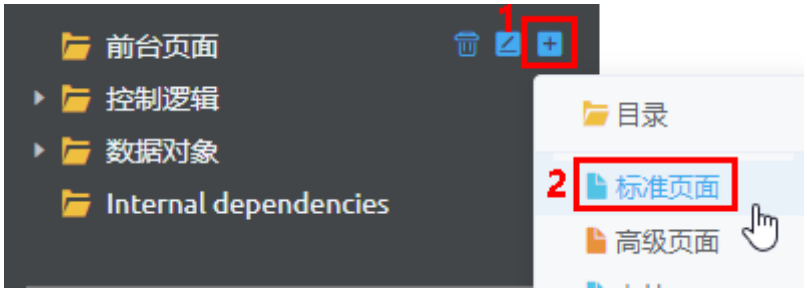
图 1-62 班次信息管理页面

操作步骤

步骤1 创建标准页面。

1. 如图1-63所示，在应用的开发页面，选择存放脚本的目录（前台页面），单击目录对应的  ，选择“标准页面”。

图 1-63 创建标准页面



2. 在弹出的“添加标准页面”页面，如图1-64所示，选择创建一个空白页面，输入“标签”为“班次信息管理”，“名称”为“shiftInfoManagement”，单击“添加”。

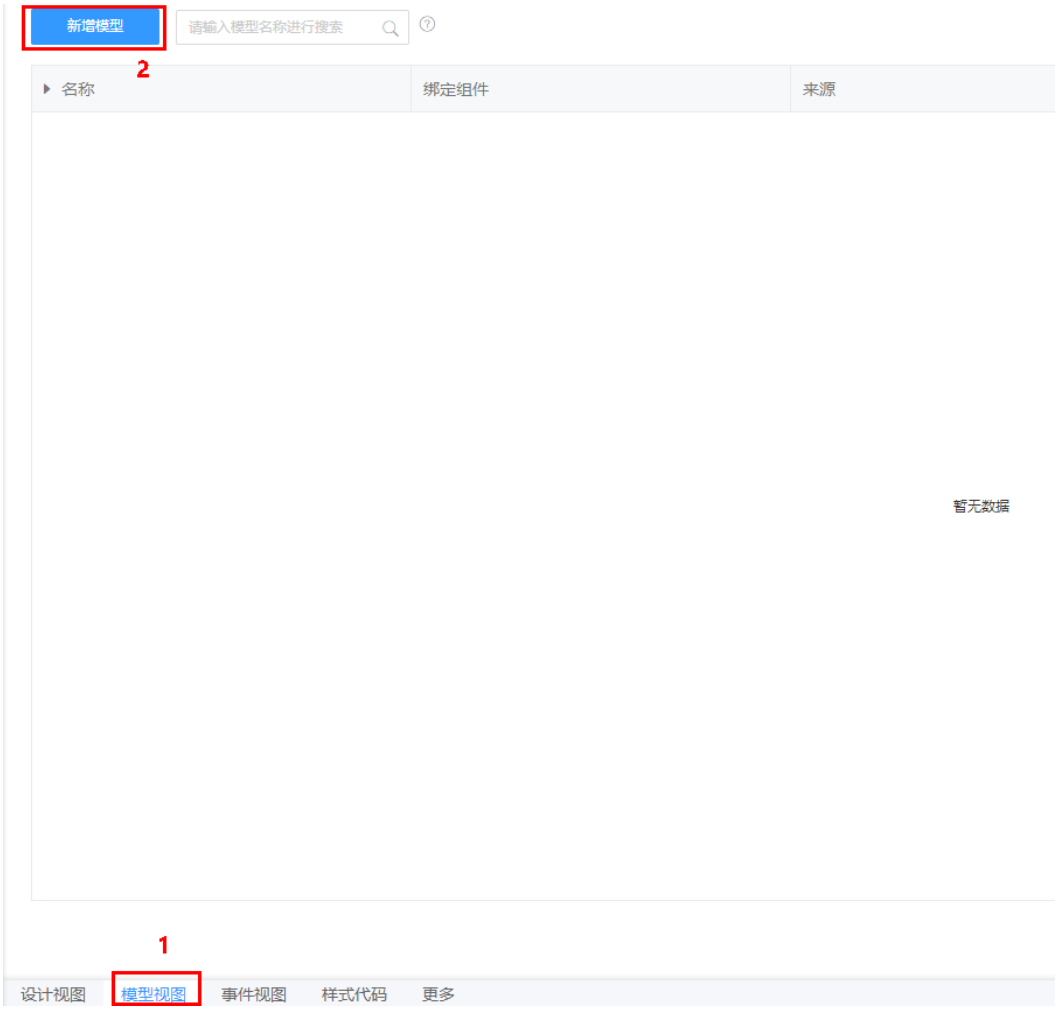
   创建完成后，自动进入编辑页面。

图 1-64 添加标准页面

步骤2 定义对象模型“shiftInfoManagement”。

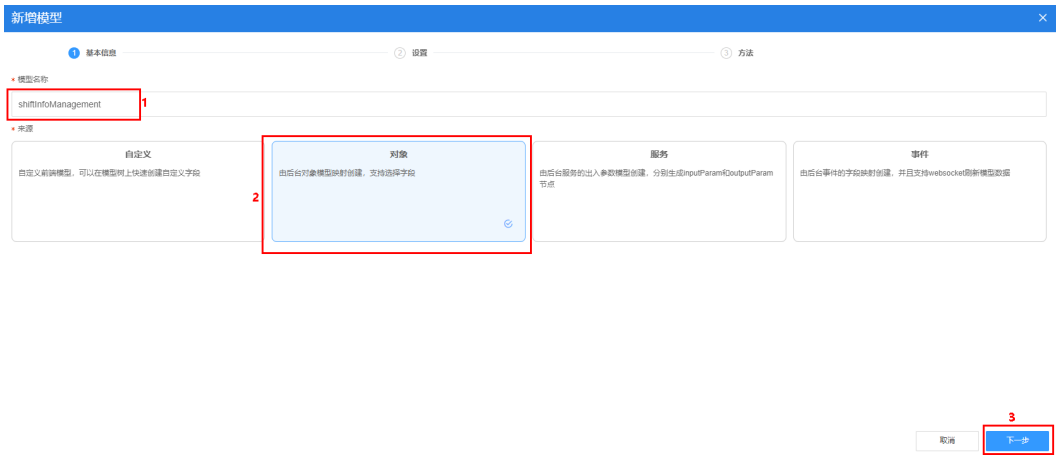
1. 如图1-65所示，在页面下侧，单击“模型视图”，切换到“模型视图”，单击“新增模型”，进入“新增模型”页面。

图 1-65 新增模型



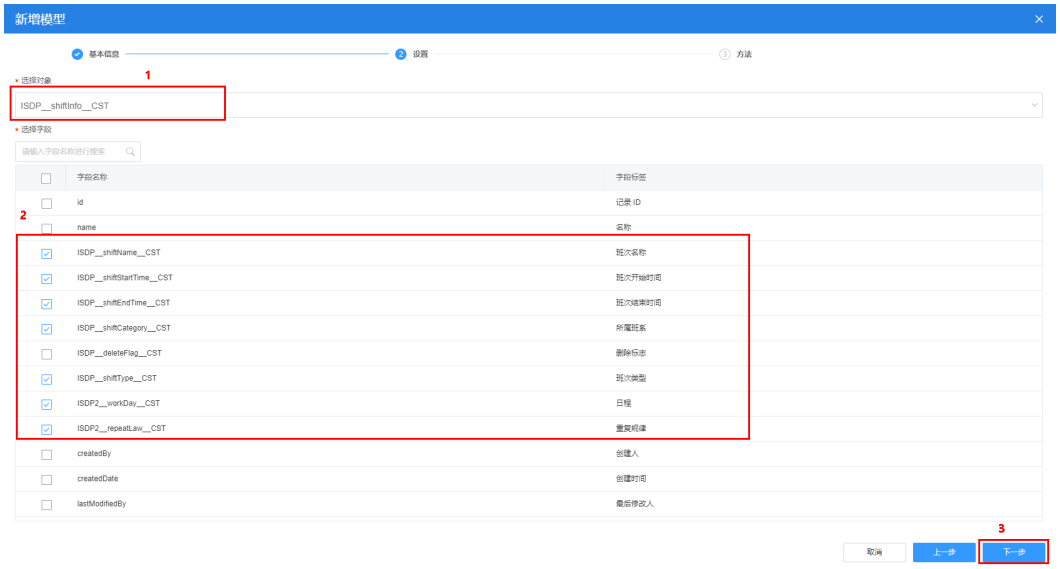
2. 如图1-66所示，输入模型名称（如：shiftInfoManagement），选择来源为“对象”，单击“下一步”。

图 1-66 基本信息



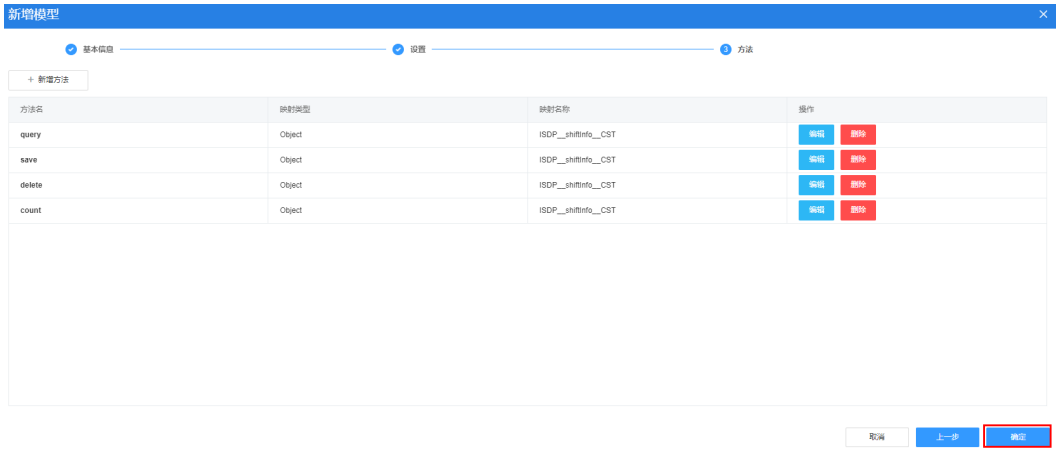
3. 如图1-67所示，选择对象和对象字段，单击“下一步”。

图 1-67 选择对象和字段



4. 如图1-68所示，方法保持默认，单击“确定”。

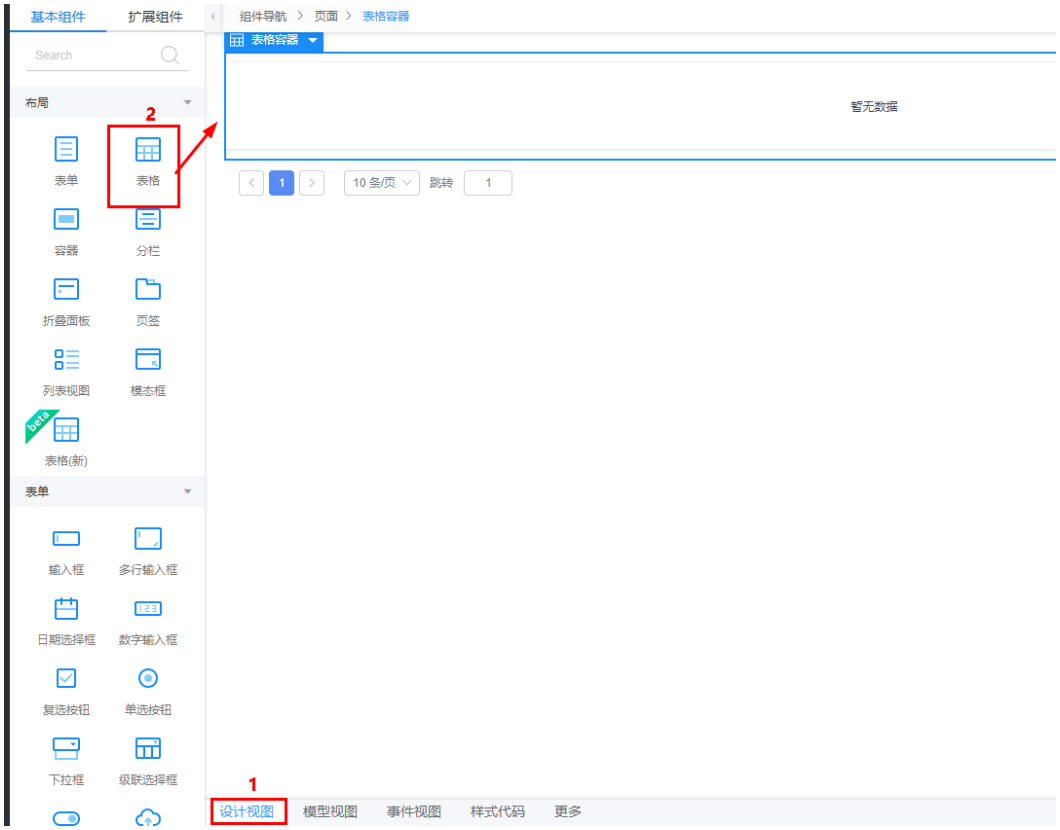
图 1-68 方法



步骤3 绑定模型，设置查询结果区域。

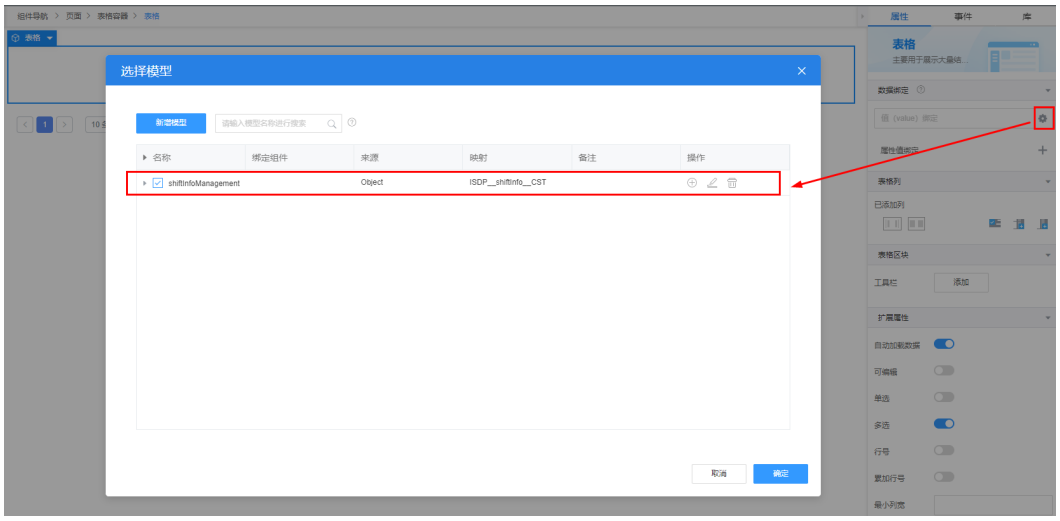
1. 如图1-69所示，切换到“设计视图”，将左侧组件区的“表格”拖拽到右侧“设计视图”中。

图 1-69 选择组件



2. 如图1-70所示，单击“数据绑定”对应的 ，在弹出的“选择模型”页面中选择新增的shiftInfoManagement模型，单击“确定”，为表格绑定对象模型。

图 1-70 数据绑定

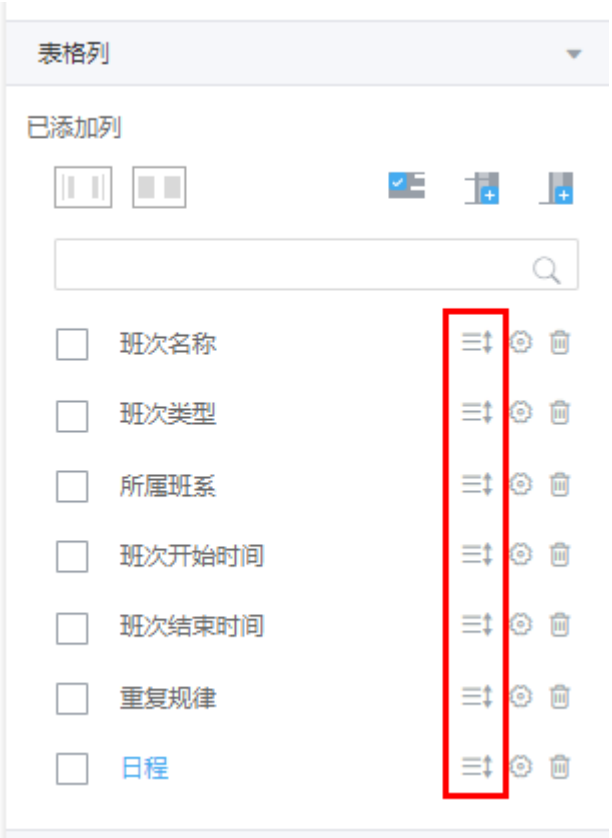


绑定对象模型后，系统自动将模型的所有字段添加为查询结果列，如图1-71所示。

图 1-71 绑定数据模型后的结果列

3. 对表格中列进行配置。

   - 拖动“已添加列”中列后面的 ，调整列的显示顺序。



   - 单击列后面的 ，修改列标题等字段属性，字段属性请参见**表1-9**。

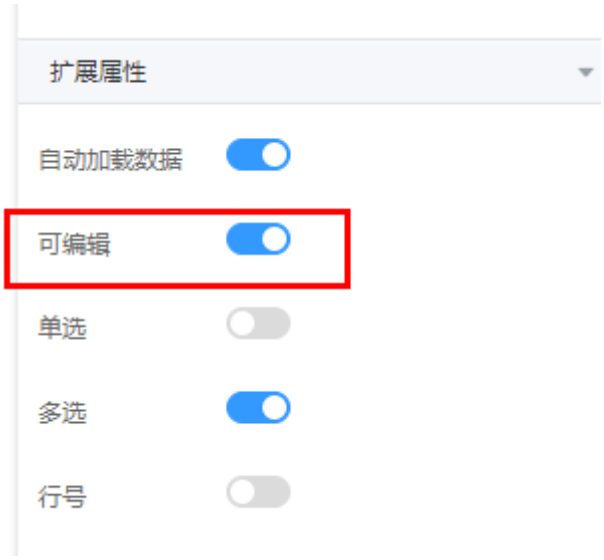
图 1-72 属性配置

表 1-9 字段属性

调整后的顺序	字段名	列标题	单元格可编辑	显示编辑hover	组件类型	选项	弹层独立
1	ISDP__shiftName__CST	班次名称	是	是	输入框	-	-
2	ISDP__shiftType__CST	班次类型	是	是	下拉框	■ 0：分散值班 ■ 1：连续值班	是
3	ISDP__shiftCategory__CST	班系	是	是	下拉框	■ 0：月排班 ■ 1：周排班 ■ 2：日排班	是
4	ISDP__shiftStartTime__CST	上班时间	是	是	输入框	-	-
5	ISDP__shiftEndTime__CST	下班时间	是	是	输入框	-	-
6	ISDP2__repeatLaw__CST	重复规律	是	是	下拉框	■ 1：按周 ■ 2：按日	是
7	ISDP2__workDay__CST	日程	是	是	输入框	-	-

4. 选中“表格”，单击右侧“属性”页签，如图1-73所示，在“扩展属性”中打开可编辑开关。

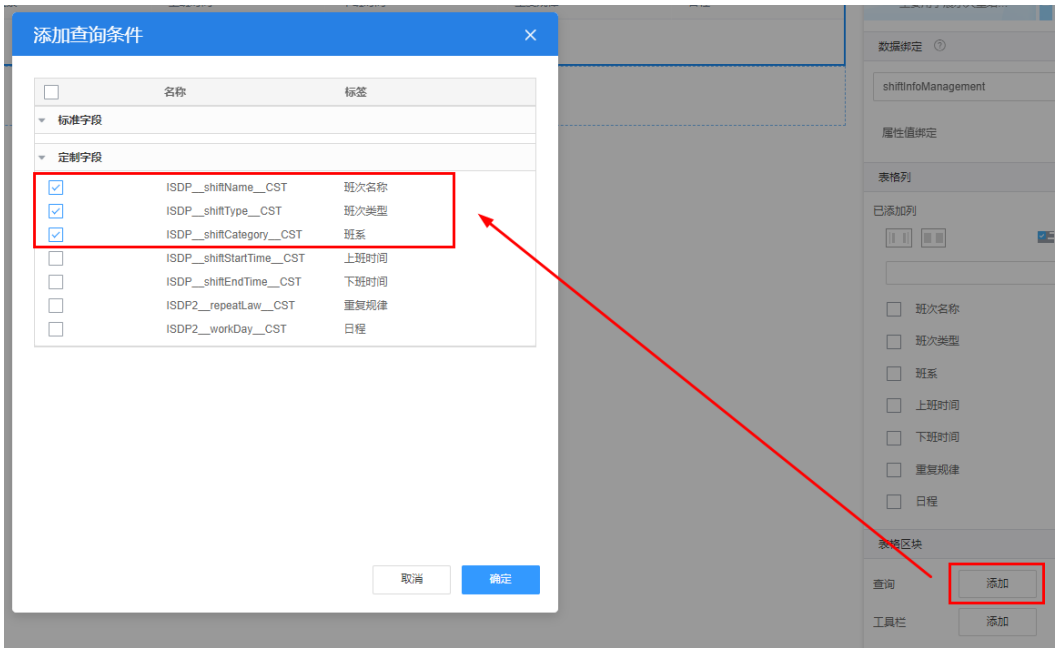
图 1-73 配置扩展属性



步骤4 设置查询条件区域。

如图1-74所示，选中“表格”，单击右侧“属性”页签中“表格区块”中“查询”后的“添加”按钮，在弹出的“添加查询条件”页面中选择查询条件，单击“确定”。

图 1-74 添加查询条件

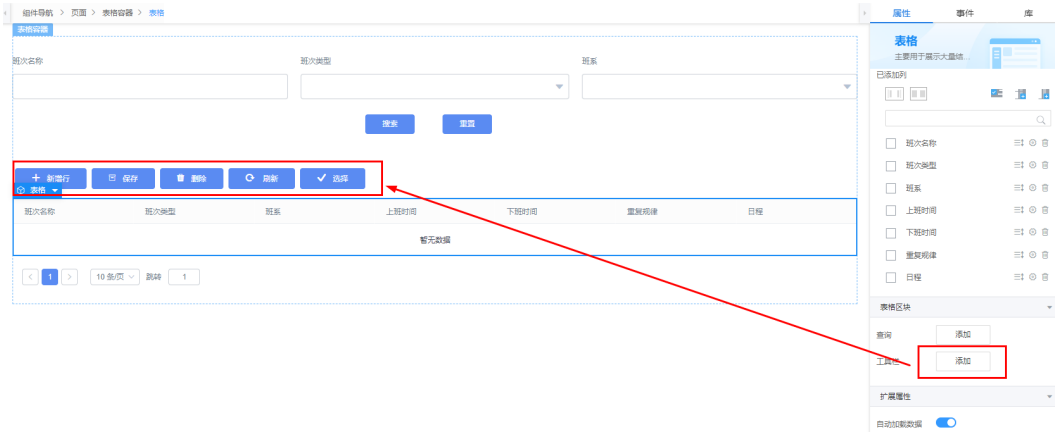


添加查询条件后可以看到，系统自动创建了2个基础容器：第1个基础容器中是1行3列的栅格容器，用于放置查询条件；第2个基础容器中放置查询按钮。

步骤5 设置工具栏区域。

1. 如图1-75所示，选中“表格”，单击右侧“属性”页签“表格区块”中“工具栏”后的“添加”按钮。

图 1-75 添加按钮



2. 如图1-76所示，在按钮上右击，选择“删除”，删除上图工具栏中多余的按钮，只保留“新增行”、“保存”和“删除”。

图 1-76 删除按钮

步骤6 单击页面上部的  ，保存页面配置。

步骤7 单击页面上部的  ，预览页面，如图1-77所示。

图 1-77 预览页面

----结束

调试页面

在预览的页面上验证新增、修改、删除、查询班次信息。

若有问题，可在预览浏览器页面，按“F12”或者“Ctrl + Shift + I”开启调试工具。在“Console”页签查看日志相关信息，在“Network”页签查看网络请求信息。

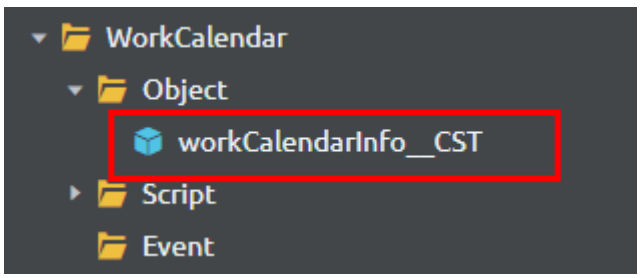
1.3.4.5.2 高级页面

高级页面通过拖拽、布局组件，并进行相关属性配置即可快速搭建应用。高级页面主要用于开发应用中较复杂的前端页面，例如包含图片、图表、视频、地图等元素的页面。

高级页面提供了常用组件，组件包含了预置的样式，并封装了基础事件代码，实现了开箱即用。

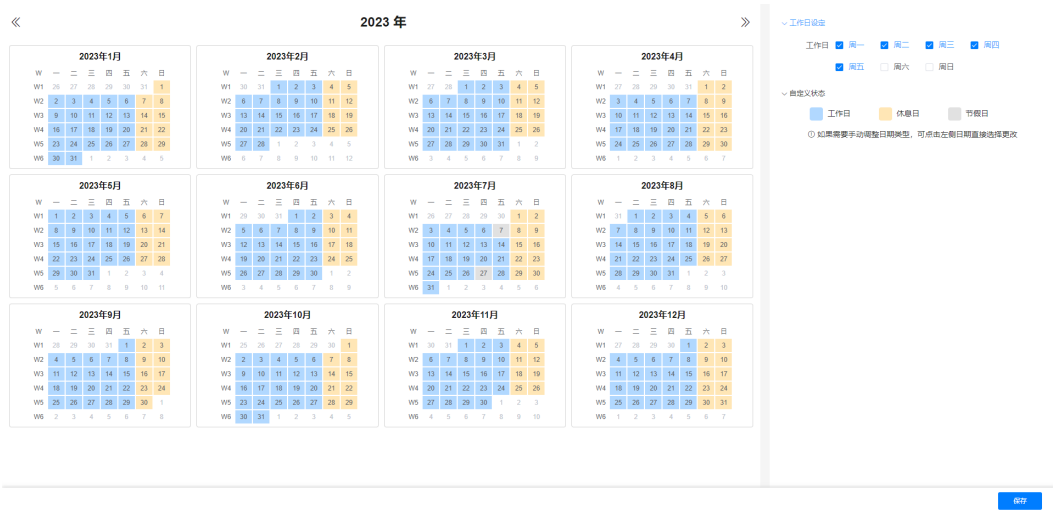
场景描述

本节以使用智能排班模型BO中工作日历信息表（ISDP__workCalendarInfo__CST）和工作日历管理开放的接口开发一个标高级页面（工作日历管理）为例，描述高级页面的开发过程。



工作日历管理，支持按年设置整年每个月份的工作日和非工作，同时支持单个调整工作日、非工作和节假日。

图 1-78 工作日历管理页面



高级页面中预置的组件，无法满足页面要求，需要自定义开发组件。

操作步骤

步骤1 下载组件模板。

系统预置了以下几种组件模板：

组件模板名称	功能
widgetVueTemplate	当自定义组件需要使用Vue库时，请选用该模板。
widgetPropertyTemplate	当自定义组件需要通过自定义属性栏配置属性时，请选用该模板。
widgetActionTemplate	自定义组件需要添加动作属性时，请选用该模板。
widgetEventTemplate	当自定义组件需要添加事件属性时，请选用该模板。
widgetBridgeTemplate	当自定义组件需要通过桥接器调用后台数据时，请选用该模板。

组件模板名称	功能
widgetPageMacroTemplate	当需要使用页面宏来存储变量时，请选用该模板。

1. 在左侧导航栏中，单击  ，选择“高级页面 > 组件模板”。
2. 根据选择模板（widgetBridgeTemplate），在组件模板详情页中单击“下载”按钮，在“下载组件模板”弹窗中输入组件名称（例如：CalendarInfoManagement），单击“保存”按钮即可。

图 1-79 选择模板

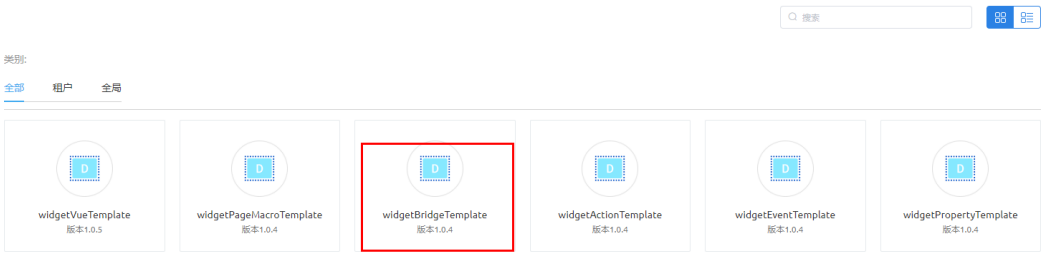


图 1-80 下载组件模板

说明

若选择“下载原始模板按钮”，下载到本地的包中组件名称不会被修改。

步骤2 将下载到本地的组件包进行解压，使用开发工具进行开发。

此处以CalendarInfoManagement这个组件为例，说明zip包中的文件以及文件的功能。

文件名	文件说明
CalendarInfoManagement.js	整个组件的渲染核心JS，在组件编辑状态和页面最终的发布运行态都会被加载执行。主要包含2个预置的方法init和render。 ● init方法主要包含组件渲染需要初始化数据的入口函数。 ● render方法负责整个组件渲染的逻辑方法。
CalendarInfoManagement.editor.js	负责组件在编辑状态时需要渲染的界面和逻辑。*.editor.js只在组件编辑状态被加载。主要包含2个方法。 ● propertiesConfig方法主要负责组件配置页面中的property配置逻辑。 ● create方法仅在组件首次被创建时调用一次。
CalendarInfoManagement.css	组件自身涉及的css文件，可以在此编写，需要是引入的外部第三方的css，则可以通过lib的方式引入。不需要将外部的css拷贝到此文件中。
CalendarInfoManagement.ftl	需要在服务端提前渲染的部分可以写在此文件中。相当于HTML文件，负责样式展示。
packageinfo.json	组件的元数据描述文件，主要包含组件的名字、描述信息以及作者信息。这里还可以配置本地路径，供本地调测使用。

1. 新增messages-zh/messages-en.json文件（功能：组件的国际化资源文件，用于配置多语言），本模板中没有，新增这2个文件。
   - messages-zh.json，内容：
     ```
     {
        "zh-CN": {
           "key" : "value"
        }
     }
     ```
   - messages-en.json，内容：
     ```
     {
        "en-US": {
           "key" : "value"
        }
     }
     ```
2. 修改“packageinfo.json”，增加组件必要依赖，例如多语言文件、库文件等。
   例如本组件需要依赖库文件Vue、VueI18n、Element，其中库文件名称和版本号需要在App开发界面左侧列表单击  ，选择“高级页面 > 库”中进行获取。
   增加如下加粗代码：
   ```
   {
    "widgetApi": [
      {
        "name": "CalendarInfoManagement"
      }
    ],
    "widgetDescription": "日历配置",
    "authorName": "XXX",
    "localFileBasePath": "",
    "i18n": [
   ```

```
  {
    "name": "messages-en"
  },
  {
    "name": "messages-zh"
  }
 ],
 "requires": [
  {
    "name": "global_Vue",
    "version": "100.7"
  },
  {
    "name": "global_VueI18n",
    "version": "100.7"
  },
 {
    "name": "global_Element",
    "version": "100.8"
  }
 ]
}
```

3. 开发JS代码，修改“CalendarInfoManagement.js”。

   JS代码主要实现以下几个业务功能：

   - 通过接口“queryWorkCalendarInfo”实现查询工作日历数据，展示在工作日历页面。
   - 通过接口“batchAddWorkCalendarInfo”实现保存对工作日历数据的配置。
   - 通过接口“getSysParameter”和“setParameter”实现获取和修改系统参数的值。

```
var CalendarInfoManagement = StudioWidgetWrapper.extend({
    /*
     * Triggered when initializing a widget and will have the code that invokes rendering of the widget
     * setParentContainer(JQueryParentContainerDOM) - binds event to this container
     * setItemContainer(JQueryItemContainerDOM) - binds studio item events for respective item
containers
     * bindEvents() - binds the studio event to this widget
     */
    init: function () {
        var thisObj = this;
        thisObj._super.apply(thisObj, arguments);
        thisObj.render();
        if ((typeof (Studio) != "undefined") && Studio) {

        }
    },

    /*
     * Triggered from init method and is used to render the widget
     */
    render: function () {
        var thisObj = this;
        var widgetProperties = thisObj.getProperties();
        var elem = thisObj.getContainer();
        var items = thisObj.getItems();
        var connectorProperties = thisObj.getConnectorProperties();

        /*
         * API to get base path of your uploaded widget API file
         */
        var widgetBasePath = thisObj.getWidgetBasePath();
        if (elem) {
            var containerDiv = $(".scfClientRenderedContainer", elem);
            if (containerDiv.length) {
                $(containerDiv).empty();
            } else {
```

```
        containerDiv = document.createElement('div');
        containerDiv.className = "scfClientRenderedContainer";
        $(elem).append(containerDiv);
    }

    var i18n = HttpUtils.getI18n({
        locale: HttpUtils.getLocale(),
        messages: thisObj.getMessages()
    });
    thisObj.vm = new Vue({
        el: $("#CalendarInfoManagement", elem)[0],
        i18n: i18n,
        data: {
            calendarLoading: true,
            activeNames: ['1', '2'],
            ruleForm: {
                type: []
            },
            year: new Date().getFullYear(),
            weeks: ["一", "二", "三", "四", "五", "六", "日"],
            yearData: [],
            typeDateList: [],
            recordDataList: []
        },
        watch: {
            'ruleForm.type'(val, oldVal) {
                let typeDateList = [];
                for (let i = 0; i < val.length; i++) {
                    typeDateList.push(this.formatDateType(val[i]));
                }
                this.typeDateList = typeDateList;
                this.typeDateList.sort((a, b) => a - b);
                let oldList = [];
                for (let i = 0; i < oldVal.length; i++) {
                    oldList.push(this.formatDateType(oldVal[i]));
                }
                oldList.sort((a, b) => a - b);
                if (this.yearData.length > 0) {
                    this.yearData.map((yItem, yIndex) => {
                        yItem.monthData.map((mItem, mIndex) => {
                            mItem.map((dItem, dIndex) => {
                                if (dItem.type == 0) {
                                    // 如果现在勾选的多选组里有，但是旧的多选组里没有，说明是新选中的工
作日
                                    if (this.typeDateList.includes(dItem.week) && !
oldList.includes(dItem.week)) {
                                        dItem.dayType = 'W';
                                    }
                                    // 如果现在勾选的多选组里没有，但是旧的多选组里有，说明是新取消的工
作日
                                    if (!this.typeDateList.includes(dItem.week) &&
oldList.includes(dItem.week)) {
                                        dItem.dayType = 'R';
                                    }
                                }
                            })
                        })
                    });
                }
            }
        },
        async created() {
            await ISDPBaseSDK.initISDPLogin(true, () => {
                this.init();
            })
            // this.init();
        },
        methods: {
            // 初始化方法
```



```
            async init() {
                await this.getSysParameter(['ISDP__workDate']);
                let typeList = [];
                for (let i = 0; i < this.typeDateList.length; i++) {
                    typeList.push(this.formatDateNum(this.typeDateList[i]));
                }
                this.ruleForm.type = typeList;
                this.queryData();
              },
              // 查询工作日历数据
            queryData() {
                let _this = this;
                let params = {
                    pageStart: 0,
                    pageSize: 500,
                    deleteFlag: 0,
                    year: String(this.year)
                }
                let connector = thisObj.getConnectorInstanceByName('queryWorkCalendarInfo');
                if (connector) {
                    connector.query(params).done(res => {
                        if (res && res.resp && res.resp.code == "0" && res.data && res.data.records)
{
                            _this.recordDataList = res.data.records || [];
                            _this.setYearMonthInfos();
                        } else {
                            _this.$message.error('查询工作日历数据失败')
                        }
                        _this.calendarLoading = false;
                    }).fail(function (error) {
                        let msg = error.response.resMsg ? error.response.resMsg : '请求失败，请联系管
理员'
                        _this.$message.error(msg)
                        _this.calendarLoading = false;
                    })
                }
              },
              saveData() {
                 // 保存工作日系统参数
               let _this = this;
                 _this.calendarLoading = true;
                 let param = {
                    name: 'ISDP__workDate',
                    value: this.typeDateList ? this.typeDateList.join(';') : ''
                 }
                 this.setParameter(param);
                 let yearData = this.yearData;
                 let serveDate = [];
                 yearData.map((yItem, yIndex) => {
                    yItem.monthData.map((mItem, mIndex) => {
                        mItem.map((dItem, dIndex) => {
                            if (dItem.type == 0) {
                                serveDate.push({
                                    basicInfo: {
                                        date: this.timestampToDate(dItem.date),
                                        workdaysFlag: dItem.dayType,
                                    }
                                })
                            }
                        })
                    })
                 });
                 let connector =
thisObj.getConnectorInstanceByName('batchAddWorkCalendarInfo');
                 if (connector) {
                    connector.query({
                        workCalendarInfoList: serveDate
                    }).done(res => {
                        if (res && res.resp && res.resp.code == "0") {
```



```
                        _this.$message.success('保存成功')
                    } else {
                        _this.$message.error('保存失败')
                    }
                    _this.calendarLoading = false;
                }).fail(function (error) {
                    let msg = error.response.resMsg ? error.response.resMsg : '请求失败，请联系管
理员'
                    _this.$message.error(msg)
                    _this.calendarLoading = false;
                })
            }
        },
        // 手动单独修改某日的类型
        changeDayType(d, str, yIndex, mIndex, dIndex) {
            d.dayType = str;
            this.$refs['dateTypeBox' + yIndex + mIndex + dIndex][0].showPopper = false;
        },
        preYear() {
            this.year -= 1;
            this.ruleForm.type = [];
            this.init();
        },
        nextYear() {
            this.year += 1;
            this.ruleForm.type = [];
            this.init();
        },
        getCellColor(d) {
            let color = d.type == -1 ? '#c0c4cc' : (d.type == 1 ? '#c0c4cc  ' : '')
            return color;
        },
        getBackground(d) {
            let bgColor = ''
            if (d.dayType == 'W') { //工作日
             bgColor = '#B1D8FF';
            } else if (d.dayType == 'R') { //休息日
             bgColor = '#FFE6B5';
            } else if (d.dayType == 'H') { //假日
             bgColor = '#E1E1E1';
            }
            return bgColor;
        },
        setYearMonthInfos() {
            this.yearData = [];
            for (let i = 0; i < 12; i++) {
                let temp = {
                    month: i + 1,
                    monthData: this.generateMonth(new Date(this.year + '-' + (i + 1) + '-01
00:00:00'))
                }
                this.yearData.push(temp);
            }
            //  如果该年数据没有存库，切换后默认调保存接口存储下当前根据工作设定数组展示出来
的数据
         if (this.recordDataList.length === 0) {
                this.autoSaveData();
            }
        },
        // 如果该年数据没有存库，切换后默认调保存接口存储下当前根据工作设定数组展示出来的
数据
        autoSaveData() {
            let _this = this;
            let yearData = this.yearData;
            let serveDate = [];
            yearData.map((yItem, yIndex) => {
                yItem.monthData.map((mItem, mIndex) => {
                    mItem.map((dItem, dIndex) => {
                        if (dItem.type == 0) {
```



```
                                serveDate.push({
                                    basicInfo: {
                                        date: this.timestampToDate(dItem.date),
                                        workdaysFlag: dItem.dayType,
                                    }
                                })
                            }
                        })
                    })
                });
                let connector =
thisObj.getConnectorInstanceByName('batchAddWorkCalendarInfo');
                if (connector) {
                    connector.query({
                        workCalendarInfoList: serveDate
                    }).done(res => {
                    }).fail(function (error) {
                    })
                }
            },
            generateMonth(date) {
                date.setDate(1)
                // 星期日为0（默认展示为：星期日 - 星期6；修改展示为：星期1 - 星期日）
             let weekStart = date.getDay() == 0 ? 6 : date.getDay() - 1;
                let endDate = new Date(date.getFullYear(), date.getMonth() + 1, 0)
                let dayEnd = endDate.getDate()
                let weeEnd = endDate.getDay()
                let milsStart = date.getTime()
                let dayMils = 24 * 60 * 60 * 1000
                let milsEnd = endDate.getTime() + dayMils
                let monthDatas = []
                let current;
                // 上个月的几天
             for (let i = 0; i < weekStart; i++) {
                    current = new Date(milsStart - (weekStart - i) * dayMils)
                    monthDatas.push({
                        type: -1,
                        date: current,
                        fullYear: current.getFullYear(),
                        month: current.getMonth() + 1,
                        day: current.getDate(),
                        week: current.getDay(),
                        dayType: ''
                    })
                }
                // 当前月
             for (let i = 0; i < dayEnd; i++) {
                    current = new Date(milsStart + i * dayMils)
                    let dayType = 'R';
                    if (this.recordDataList.length > 0) {
                        for (let j = 0; j < this.recordDataList.length; j++) {
                            let tempDay = this.timestampToDate(current);
                            if (this.recordDataList[j]['ISDP__date__CST'] == tempDay) {
                                dayType = this.recordDataList[j]['ISDP__workdaysFlag__CST']
                                break;
                            }
                        }
                    } else {
                        if (this.typeDateList.includes(current.getDay())) {
                            dayType = 'W';
                        }
                        if (!this.typeDateList.includes(current.getDay())) {
                            dayType = 'R';
                        }
                    }
                    monthDatas.push({
                        type: 0,
                        date: current,
                        fullYear: current.getFullYear(),
```



```
                        month: current.getMonth() + 1,
                        day: current.getDate(),
                        week: current.getDay(),
                        dayType: dayType
                    })
                }
                // 下个月的几天
                for (let i = 0; i < (7 - weeEnd); i++) {
                    current = new Date(milsEnd + i * dayMils)
                    monthDatas.push({
                        type: 1,
                        date: current,
                        fullYear: current.getFullYear(),
                        month: current.getMonth() + 1,
                        day: current.getDate(),
                        week: current.getDay(),
                        dayType: ''
                    })
                }
                let monthData = [];
                for (let i = 0; i < monthDatas.length; i++) {
                    let mi = i % 7;
                    if (mi == 0) {
                        monthData.push([])
                    }
                    monthData[Math.floor(i / 7)].push(monthDatas[i])
                }
                // 少于6行，补足6行
                if (monthData.length <= 5) {
                    milsStart = current.getTime()
                    let lastLine = []
                    for (let i = 1; i <= 7; i++) {
                        current = new Date(milsStart + i * dayMils)
                        lastLine.push({
                            type: 1,
                            date: current,
                            fullYear: current.getFullYear(),
                            month: current.getMonth() + 1,
                            day: current.getDate(),
                            week: current.getDay(),
                            dayType: ''
                        })
                    }
                    monthData.push(lastLine)
                }
                return monthData;
            },
            //时间戳转时间格式
            timestampToDate(timestamp) {
                if (timestamp && timestamp !== '') {
                    var date = new Date(timestamp);//时间戳为10位需*1000，时间戳为13位的话不需
乘1000
                    var Y = date.getFullYear() + '-';
                    var M = (date.getMonth() + 1 < 10 ? '0' + (date.getMonth() + 1) :
date.getMonth() + 1) + '-';
                    var D = (date.getDate() < 10 ? '0' + date.getDate() : date.getDate());
                    return Y + M + D;
                } else {
                    return '';
                }
            },
            // 周一到周五转换为数字
            formatDateType(str) {
                let num = 0;
                switch (str) {
                    case '周一':
                        num = 1;
                        break;
                    case '周二':
```



```
                    num = 2;
                    break;
                case '周三':
                    num = 3;
                    break;
                case '周四':
                    num = 4;
                    break;
                case '周五':
                    num = 5;
                    break;
                case '周六':
                    num = 6;
                    break;
                case '周日':
                    num = 0;
                    break;
            }
            return num;
        },
        // 数字转换为周一到周五
        formatDateNum(num) {
            let str = '周日';
            switch (num) {
                case 1:
                    str = '周一';
                    break;
                case 2:
                    str = '周二';
                    break;
                case 3:
                    str = '周三';
                    break;
                case 4:
                    str = '周四';
                    break;
                case 5:
                    str = '周五';
                    break;
                case 6:
                    str = '周六';
                    break;
                case 0:
                    str = '周日';
                    break;
            }
            return str;
        },
        // 获取系统参数
        getSysParameter(paramList) {
            return new Promise((resolve) => {
                let _this = this;
                let connector = thisObj.getConnectorInstanceByName('getSysParameter');
                let params = {
                    sysparamsNameList: paramList
                }
                if (connector) {
                    connector.query(params).done(res => {
                        if (res.resp.code == "0" && res && res.data && res.data.result) {
                            if (res.data.result['ISDP__workDate'] &&
res.data.result['ISDP__workDate'] != '') {
                                _this.typeDateList =
res.data.result['ISDP__workDate'].split(';').map(Number);
                            }
                        } else {
                            _this.$message.error('获取系统参数失败')
                        }
                        resolve('');
                    }).fail(function (error) {
```

```
                        let msg = error.response.resMsg ? error.response.resMsg : '请求失败，请联
系管理员'
                        _this.$message.error(msg)
                        resolve('');
                    })
                }
            })
        },
        // 修改系统参数
        setParameter(param) {
            let _this = this;
            let connector = thisObj.getConnectorInstanceByName('setParameter');
            let params = {
                name: param.name,
                value: param.value
            }
            if (connector) {
                connector.query(params).done(res => {
                    if (res && res.resp && res.resp.code && res.resp.code == "0") {
                    } else {
                        _this.$message.error('修改系统参数失败')
                    }
                }).fail(function (error) {
                    let msg = error.response.resMsg ? error.response.resMsg : '请求失败，请联系管
理员'
                    _this.$message.error(msg)
                })
            }
        },
        // 提取两个数组中不同的
        filterArr(arr1, arr2) {
            const arr = [...arr1, ...arr2];
            const newArr = arr.filter((t) => {
                return !(arr1.includes(t) && arr2.includes(t));
            });
            return newArr;
        },
    }
})
}
/*
 * API to bind global events to the item DOM, it should not be deleted if there will some events
to trigger in this widget.
 */
thisObj.sksBindItemEvent();
/*
 * API to refresh the previously bound events when a resize or orientation change occurs.
 */
$(window).resize(function () {
    thisObj.sksRefreshEvents();
});
},
});
```

4. 开发ftl代码，修改“CalendarInfoManagement.ftl”。

   ftl代码为该功能页面呈现的渲染代码，主要采用element组件实现。

   具体代码如下：

```
<div id="CalendarInfoManagement" v-cloak v-loading="calendarLoading">
    <div class="leftBox" v-if="yearData.length > 0">
        <div class="cal_con">
            <div class="cal_header">
                <div class="cal_h_left">
                    <div class="cal_h_btn" @click="preYear" style="margin-right:10px;">
                        <i class="el-icon-d-arrow-left"></i>
                    </div>
                </div>
                <div>
                    <span class="cal_h_time">{{ year }} 年 </span>
```

```
            </div>
            <div class="cal_h_left">
               <div class="cal_h_btn" @click="nextYear">
                  <i class="el-icon-d-arrow-right"></i>
               </div>
            </div>
         </div>

         <div class="leftContent">
            <div class="cal_month" v-for="(yItem, yIndex) in yearData" :key="yIndex">
               <div class="cal_m_title">{{year}}年{{yItem.month}}月</div>
               <div class="cal_m_weeks">
                  <span class="cal_m_day_cell">W</span>
                  <span v-for="w in weeks" :key="w" class="cal_m_day_cell">{{w}}</span>
               </div>
               <div class="cal_m_days">
                  <div v-for="(ds, mIndex) in yItem.monthData" :key="mIndex" style="width: 100%;">
                     <div class="cal_m_day_line">
                        <div class="cal_m_day_cell">W{{mIndex+1}}</div>
                        <div v-for="(d, dIndex) in ds" :key="d.day" class="cal_m_day_cell"
                           :style="{color: getCellColor(d), background: getBackground(d)}">
                           <el-popover
                              v-if="d.type == 0"
                              :ref="'dateTypeBox'+yIndex+mIndex+dIndex"
                              placement="bottom-start"
                              popper-class="dateTypeBox"
                              trigger="click"
                              width="100">
                              <div class="smallItemBox" @click="changeDayType(d, 'W', yIndex, mIndex,
dIndex)">
                                 <div class="smallBox" style="background:#B1D8FF;"></div>工作日
                              </div>
                              <div class="smallItemBox" @click="changeDayType(d, 'R', yIndex, mIndex,
dIndex)">
                                 <div class="smallBox" style="background:#FFE6B5;"></div>休息日
                              </div>
                              <div class="smallItemBox" @click="changeDayType(d, 'H', yIndex, mIndex,
dIndex)">
                                 <div class="smallBox" style="background:#E1E1E1;"></div>节假日
                              </div>
                              <div slot="reference">{{ d.day }}</div>
                           </el-popover>
                           <div v-if="d.type != 0">{{ d.day }}</div>
                        </div>
                     </div>
                  </div>
               </div>
            </div>
         </div>
      </div>
      <div class="rightBox" v-if="yearData.length > 0">
         <el-collapse v-model="activeNames" class="ruleCollapse">
            <el-collapse-item title="工作日设定" name="1">
               <el-form :model="ruleForm" ref="ruleForm" label-width="100px" class="ruleForm">
                  <el-form-item label="工作日" prop="type">
                     <el-checkbox-group v-model="ruleForm.type">
                        <el-checkbox label="周一" name="type"></el-checkbox>
                        <el-checkbox label="周二" name="type"></el-checkbox>
                        <el-checkbox label="周三" name="type"></el-checkbox>
                        <el-checkbox label="周四" name="type"></el-checkbox>
                        <el-checkbox label="周五" name="type"></el-checkbox>
                        <el-checkbox label="周六" name="type"></el-checkbox>
                        <el-checkbox label="周日" name="type"></el-checkbox>
                     </el-checkbox-group>
                  </el-form-item>
               </el-form>
            </el-collapse-item>
            <el-collapse-item title="自定义状态" name="2">
```



```
            <div class="statusInfo">
                <div class="statusBox">
                    <div class="bgBox" style="background:#B1D8FF;"></div>工作日
                </div>
                <div class="statusBox">
                    <div class="bgBox" style="background:#FFE6B5;"></div>休息日
                </div>
                <div class="statusBox">
                    <div class="bgBox" style="background:#E1E1E1;"></div>节假日
                </div>
                <div class="tip"><i class="el-icon-warning-outline"></i> 如果需要手动调整日期类型，可点击左侧日期直接选择更改</div>
            </div>
        </el-collapse-item>
    </el-collapse>
  </div>
  <div class="btm_operation">
    <el-button type="primary" size="small" @click="saveData">保存</el-button>
    <!-- <el-button size="small">取消</el-button> -->
  </div>
</div>
```

5. 开发组件在编辑状态时需要渲染的界面和逻辑，修改“CalendarInfoManagement.editor.js”。

   在“propertiesConfig”方法中修改“type”为“connectorV2”属性值，使得“name”取值和“CalendarInfoManagement.js”中“thisObj.getConnectorInstanceByName”的取值保持一致。

   具体代码如下：

```
CalendarInfoManagement = CalendarInfoManagement.extend({
  /*
   * Config to define Widget Properties
   */
  propertiesConfig:[{
    config: [
      {
        "type": "connectorV2",
        "name": "getSysParameter",
        "model": "ViewModel",
        "label": "获取系统参数",
        "value": ""
      },
      {
        "type": "connectorV2",
        "name": "setParameter",
        "model": "ViewModel",
        "label": "修改系统参数",
        "value": ""
      },
      {
        "type": "connectorV2",
        "name": "queryWorkCalendarInfo",
        "model": "ViewModel",
        "label": "查询工作日历数据",
        "value": ""
      },
      {
        "type": "connectorV2",
        "name": "batchAddWorkCalendarInfo",
        "model": "ViewModel",
        "label": "批量修改工作日历数据",
        "value": ""
      },
    ]
  }],

  /*
   * Triggered when the user Creates a new widget and used to initialize the widget properties
```



```
    */
    create : function(cbk)
    {
        if(cbk)
        {
            this._super();
            cbk();
        }
    }
});
var params = {};
Studio.registerWidget("CalendarInfoManagement", "CalendarInfoManagement", params);
```

6. 开发css代码，修改“CalendarInfoManagement.css”，实现CSS样式。

   具体代码如下：

```
[v-cloak]{
    display: none
}
#CalendarInfoManagement {
    display: flex;
    background-color: #f5f5f5;
    height: 100%;
}
#CalendarInfoManagement .leftBox{
    width: calc(100% - 537px);
    margin-right: 17px;
    height: calc(100% - 48px);
    padding-left: 10px;
    background-color: #fff;
}
#CalendarInfoManagement .leftContent{
    overflow-y: auto;
    height: calc(100% - 64px);
}
#CalendarInfoManagement .rightBox{
    width: 520px;
    height: calc(100% - 48px);
    background-color: #fff;
    padding: 20px;
}
#CalendarInfoManagement .ruleForm{
    margin: 0;
}
#CalendarInfoManagement .ruleForm .el-form-item__label{
    line-height: 32px;
}
#CalendarInfoManagement .ruleForm .el-form-item{
    margin-bottom: 0 !important;
}
#CalendarInfoManagement .ruleCollapse .el-collapse-item__content{
    padding-bottom: 0;
}
#CalendarInfoManagement .el-collapse.ruleCollapse{
    border: none;
}
#CalendarInfoManagement .el-collapse.ruleCollapse .el-collapse-item{
    position: relative;
}
#CalendarInfoManagement .el-collapse.ruleCollapse .el-collapse-item .el-collapse-item__header{
    padding-left: 16px;
    border-bottom: none;
}
#CalendarInfoManagement .el-collapse.ruleCollapse .el-collapse-item .el-collapse-item__arrow{
    position: absolute;
    top: 18px;
    left: 0;
    z-index: 1;
}
#CalendarInfoManagement .el-collapse.ruleCollapse .el-collapse-item__wrap{
```



```
    border-bottom: none;
}
#CalendarInfoManagement .statusInfo{
    text-align: center;
}
#CalendarInfoManagement .statusBox {
    margin-right: 48px;
    font-size: 14px;
    line-height: 24px;
    display: inline-block;
}
#CalendarInfoManagement .statusBox .bgBox{
    width: 24px;
    height: 24px;
    border-radius: 4px;
    display: inline-block;
    vertical-align: middle;
    margin-right: 8px;
}
#CalendarInfoManagement .statusBox:last-child{
    margin-right: 0;
}
#CalendarInfoManagement .statusInfo .tip{
    font-size: 14px;
    color: #333;
    line-height: 20px;
    padding: 15px 0;
}
.el-popper.dateTypeBox{
    margin-top: 1px;
    min-width: 100px;
    padding: 0;
}
.smallItemBox{
    padding: 0 12px;
    font-size: 14px;
    line-height: 32px;
    cursor: pointer;
}
.smallItemBox:hover{
    background-color: #f5f5f5;
}
.smallItemBox .smallBox{
    width: 12px;
    height: 12px;
    border-radius: 2px;
    display: inline-block;
    vertical-align: middle;
    margin-right: 8px;
}
/**底部按钮信息*/
#CalendarInfoManagement .btm_operation {
    position: fixed;
    display: -webkit-box;
    display: -ms-flexbox;
    display: flex;
    -webkit-box-pack: end;
    -ms-flex-pack: end;
    justify-content: flex-end;
    z-index: 10;
    bottom: 0;
    width: 100%;
    background: #fff;
    opacity: 1;
    padding-top: 8px;
    padding-bottom: 8px;
    padding-right: 24px;
    -webkit-box-shadow: 2px -2px 6px #d9d9d9;
    box-shadow: 2px -2px 6px #d9d9d9;
```

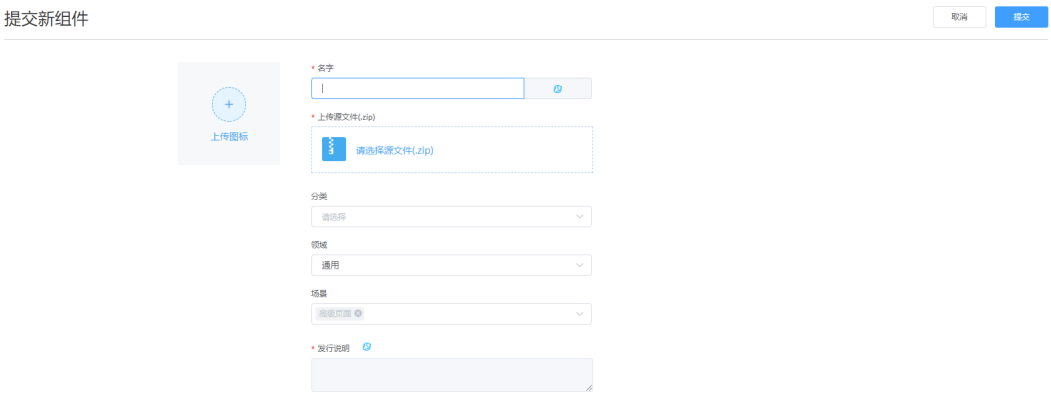
```
    border-top: none;
}
/* 日历组件开始 */
.cal_con {
  width: 100%;
  height: 100%;
  -webkit-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
  color: #606266;
  background: #fff;
  border-radius: 6px;
}
.cal_con .cal_header {
    padding: 30px 15px 30px 5px;
    font-size: 24px;
    line-height: 24px;
    display: flex;
    justify-content: space-between;
    justify-items: center;
    color: #191919;
}
.cal_con .cal_header .cal_h_time {
    /* cursor: pointer; */
    font-weight: bold;
}
.cal_con .cal_header .cal_h_left {
    height: 100%;
    display: flex;
}
.cal_con .cal_header .cal_h_left .cal_h_btn {
    height: 100%;
    cursor: pointer;
    font-size: 24px;
    font-weight: bold;
}
.cal_con .cal_header .cal_h_left .cal_h_l_icon {
    height: 8px;
    width: 12px;
    margin: 8px auto;
}
.cal_con .cal_month {
    font-size: 12px;
    text-align: center;
    height: 225px;
    margin: 0 5px 10px 5px;
    width: calc(25% - 12px);
    display: inline-block;
    border: 1px solid #d5d5d5;
    border-radius: 4px;
    padding-bottom: 5px;
}
.cal_con .cal_month .cal_m_title{
    text-align: center;
    font-size: 16px;
    color: #191919;
    line-height: 20px;
    padding-top: 10px;
    font-weight: bold;
}
.cal_con .cal_month .cal_m_weeks {
    padding: 8px 0 0 0;
    display: flex;
    justify-content: center;
    justify-items: center;
}
.cal_con .cal_month .cal_m_day_cell {
    width: 30px;
```



```
    height: 24px;
    line-height: 24px;
    cursor: pointer;
    position: relative;
    margin: 0 1px;
}
.cal_con .cal_month .cal_m_day_cell.dayBg1{
    background-color: #B1D8FF;
}
.cal_con .cal_month .cal_m_day_cell.dayBg2{
    background-color: #FFE6B5;
}
.cal_con .cal_month .cal_m_day_cell.dayBg3{
    background-color: #E1E1E1;
}
.cal_con .cal_month .cal_m_days {
    height: calc(100% - 62px);
    display: flex;
    justify-content: center;
    justify-items: center;
    align-items: center;
    flex-wrap: wrap;
}
.cal_con .cal_month .cal_m_day_line {
    width: 100%;
    display: flex;
    justify-content: center;
    justify-items: center;
    align-items: center;
    border-radius: 10px;
}
/* 日历组件结束 */
```

步骤3 上传自定义组件。

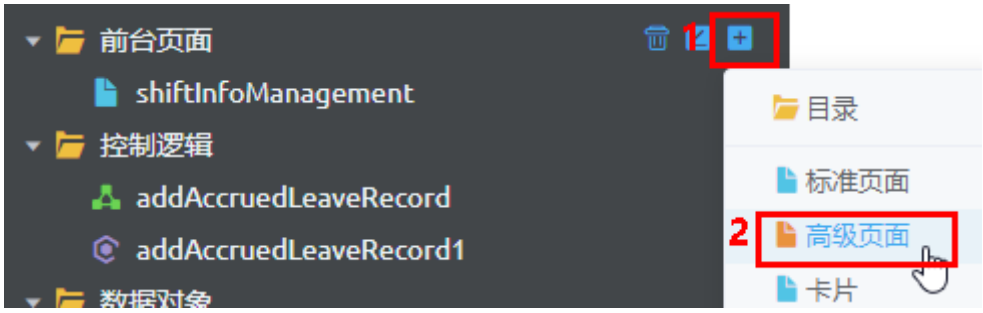
1. 将开发好的组件代码压缩到后缀为.zip的压缩文件（CalendarInfoManagement.zip）。
2. 选择“高级页面 > 组件”，单击“提交新组件”。
3. 在提交新组件页面，设置组件基本信息，并上传压缩文件，单击“提交”。



步骤4 创建页面。

1. 如图1-81所示，在应用的开发页面，选择存放页面的目录（前台页面），单击目录对应的，选择“高级页面”。

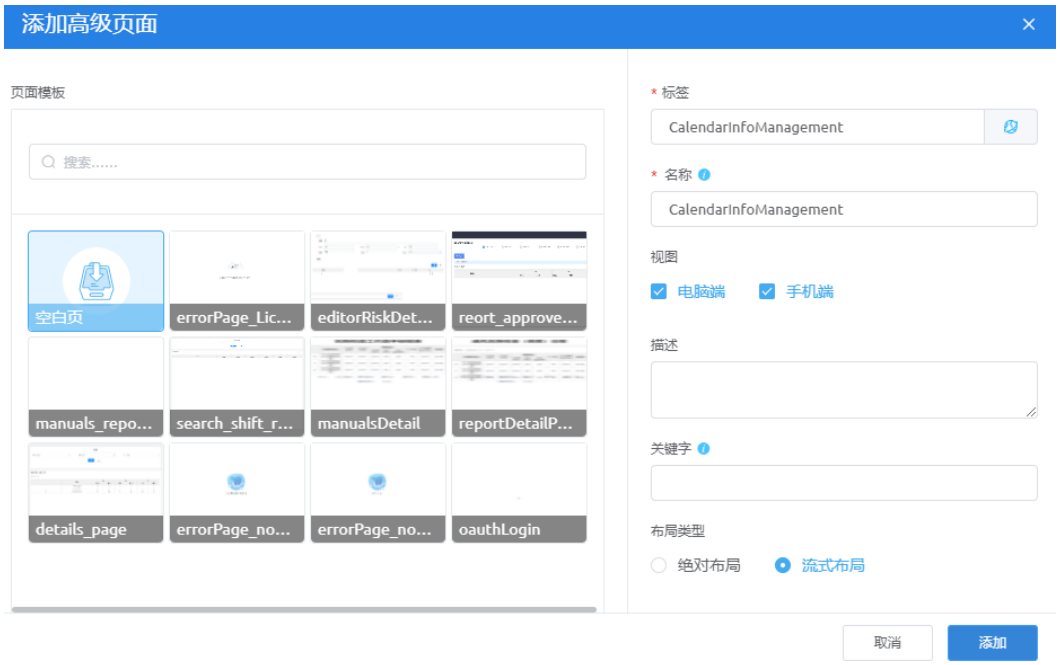
图 1-81 创建高级页面



2. 如图1-82所示，在弹出的“添加高级页面”页面，选择“空白”，输入标签和名称，选择布局类型为“流式布局”，单击“添加”。

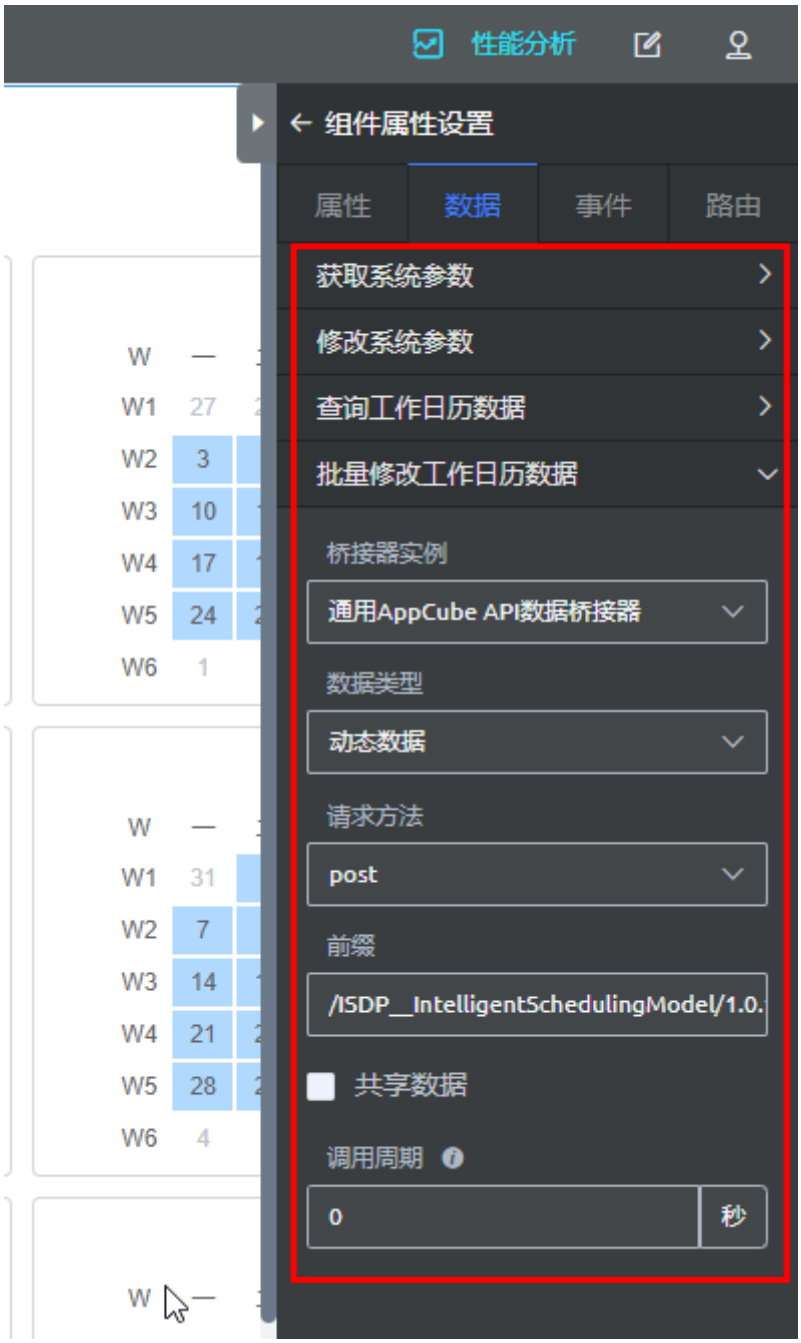
   页面创建完成后，自动进入编辑页面。

图 1-82 添加高级页面



步骤5 开发页面。

1. 选择自定义组件“CalendarInfoManagement”，拖入到页面内容区域。
2. 配置数据参数。

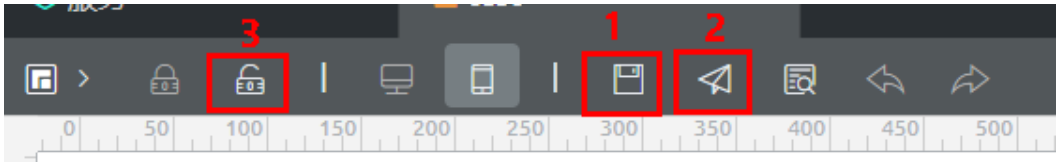


3. 单击页面上方保存按钮，保存配置，单击发布，发布页面，最后单击释放锁按钮，退出编辑状态。

说明

如果需要再次编辑，需要单击，获取锁在进行编辑。

图 1-83 保存，发布并释放锁

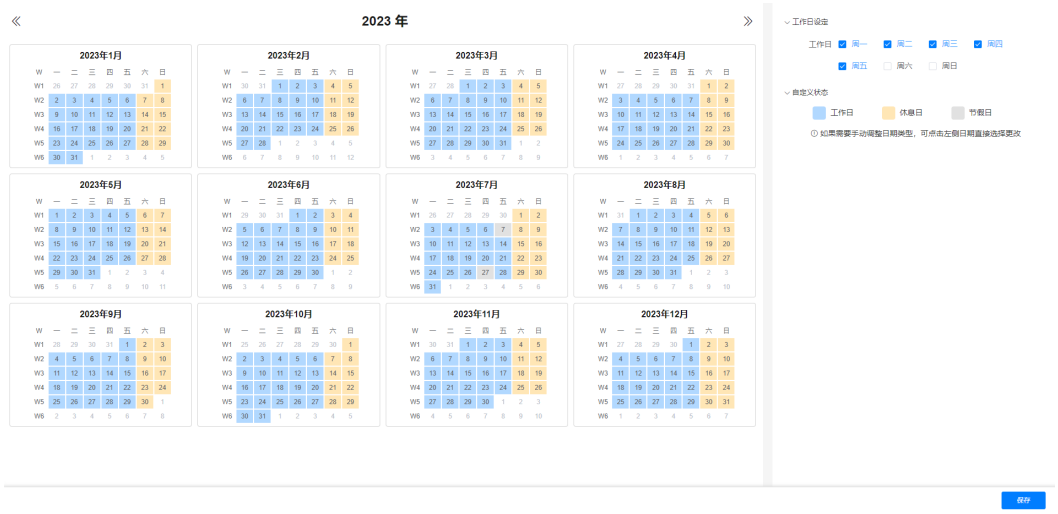


步骤6 预览页面。

如图1-84所示，单击页面上方预览按钮，即可看到页面配置效果，如图1-85所示。

图 1-84 预览

图 1-85 预览效果



----结束

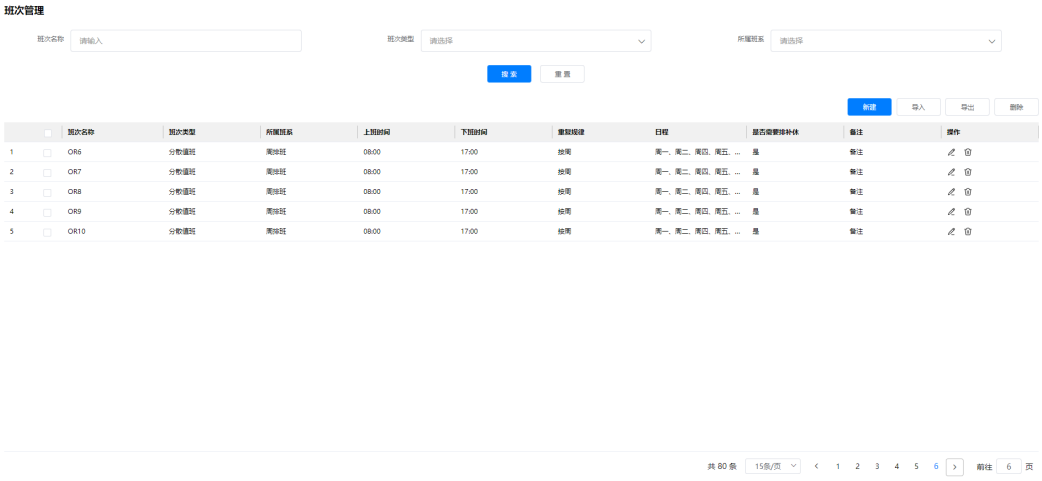
1.3.4.6 开发实例

本节介绍如何基于BO开放的接口，进行前端页面的开发。

1.3.4.6.1 场景介绍

场景描述

以智能排班模型BO开放的班次信息管理的增删改查接口为例，开发一个班次管理页面，用于管理班次信息，支持增删改查，导入和导出功能。



页面功能：

- 班次管理页面分页展示班次信息。
- 支持根据班次名称、班次类型、所属班系进行搜索。
- 支持重置搜索条件，重置时，重置搜索条件为空，展示所有数据。
- 支持新建、导入、导出、修改和删除班次信息。

场景分析

- 开发示例页面时，页面使用高级页面进行开发，高级页面中预置的组件无法满足当前页面的要求，需下载系统预置的组件模板，开发自定义组件后上传到组件库，进行使用。
- 页面中功能说明：查看班次列表/导出班次信息，调用智能排班模型BO中开放的查询班次信息列表接口（queryShiftInfo）；新增班次信息/导入班次信息，调用智能排班模型BO中开放的批量新增班次信息接口（batchAddShift）；修改班次信息，调用智能排班模型BO中开放的修改班次信息（modifyShiftInfo）；删除班次信息，调用智能排班模型BO中开放的批量删除班次信息（batchDelShiftInfo）。
- 班次管理使用到的表为：班次信息表（ISDP__shiftInfo__CST）和班次信息扩展表（ISDP__shiftInfoExtend__CST），其中，班次信息表（ISDP__shiftInfo__CST）为依赖的智能排班模型BO中的表，不可修改；班次信息扩展表（ISDP__shiftInfoExtend__CST）为智能排班模型BO后置脚本预置进去并导入到应用中的扩展表，可修改。根据页面中字段，需要在扩展表中自定义班次信表中没有，但是需要使用到的字段（**备注**和**是否需要排补休**）。

  班次信息表和班次信息扩展表均使用智能排班模型BO中开放的接口进行增删改查，对表进行数据处理，对应接口（queryShiftInfo/batchAddShift/modifyShiftInfo/batchDelShiftInfo）。

1.3.4.6.2 开发流程

序号	步骤	说明
1	配置扩展表	在应用的开发态，配置班次信息扩展表（ISDP__shiftInfoExtend__CST）的自定义字段。 具体操作请参见**配置扩展表**。
2	开发页面	在应用的开发态，进行高级页面的开发。 具体操作请参见**开发页面**。

1.3.4.6.3 配置扩展表

本节介绍如何新增班次信息扩展表（ISDP__shiftInfoExtend__CST）的自定义字段。

说明

新建字段时，可以单个新建，也可以批量新建（批量新建时，下载模板输入相关信息，再导入）。

以单个新建字段为例进行描述。

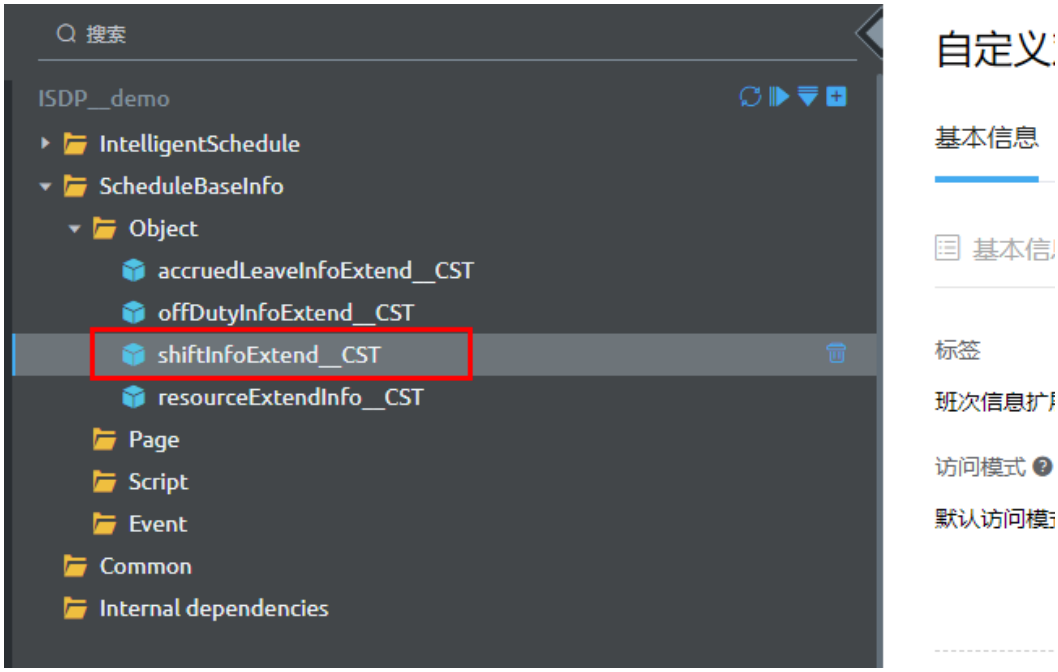
表 1-10 规划的自定义字段

标签	名称	字段类型	取值	读取/编辑权限	添加到页面布局
备注	remark	文本区	-	Standard User Profile	选择
是否需要排补休	isRest	文本	数据长度：255		

操作步骤

步骤1 如**图1-86**所示，在应用的开发页面，单击“ScheduleBaseInfo > Object”下的表“shiftInfoExtend__CST”。

图 1-86 进入表



步骤2 单击“自定义字段”页签，单击“新建”，根据导航提示，配置并单击“下一步”，最后单击“保存”，完成规划字段的添加。

例如，新建字段文本区类型的字段“remark（备注）”。

图 1-87 选择字段类型

图 1-88 输入详情

图 1-89 建立字段级安全性

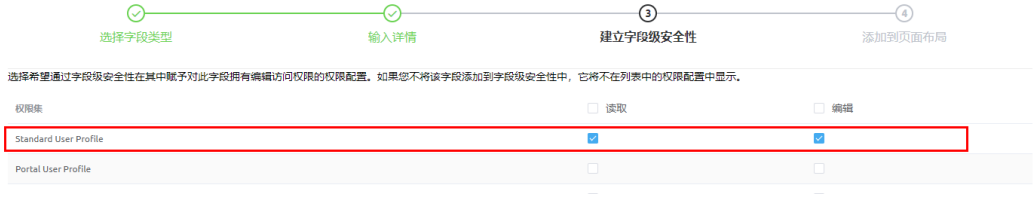


图 1-90 添加到页面布局

步骤3 参见**步骤2**，继续新建字段isRest（是否需要排补休），示例如**图1-91**所示。

图 1-91 新建的自定义的字段

班次信息扩展表配置完成后，使用智能排班模型BO中开放的接口进行增删改查，对表进行数据处理，对应接口（queryShiftInfo/batchAddShift/modifyShiftInfo/batchDelShiftInfo）。

----结束

1.3.4.6.4 开发页面

背景信息

在开发高级页面过程中，可以使用预置的组件，快速构建页面；也可以自定义组件构建页面。

开发自定义组件时，可以根据组件功能下载组件模板进行开发。

组件模板名称	功能
widgetVueTemplate	当自定义组件需要使用Vue库时，请选用该模板。

组件模板名称	功能
widgetPropertyTemplate	当自定义组件需要通过自定义属性栏配置属性时，请选用该模板。
widgetActionTemplate	自定义组件需要添加动作属性时，请选用该模板。
widgetEventTemplate	当自定义组件需要添加事件属性时，请选用该模板。
widgetBridgeTemplate	当自定义组件需要通过桥接器调用后台数据时，请选用该模板。
widgetPageMacroTemplate	当需要使用页面宏来存储变量时，请选用该模板。

开发自定义组件

步骤1 下载系统预置的组件模板。

1. 在左侧导航栏中，单击  ，选择“高级页面 > 组件模板”。
2. 根据需求选择模板（例如，widgetVueTemplate），在组件模板详情页中单击“下载”按钮，在“下载组件模板”弹窗中输入组件名称（例如：shiftManagement），单击“保存”按钮即可。

图 1-92 选择模板

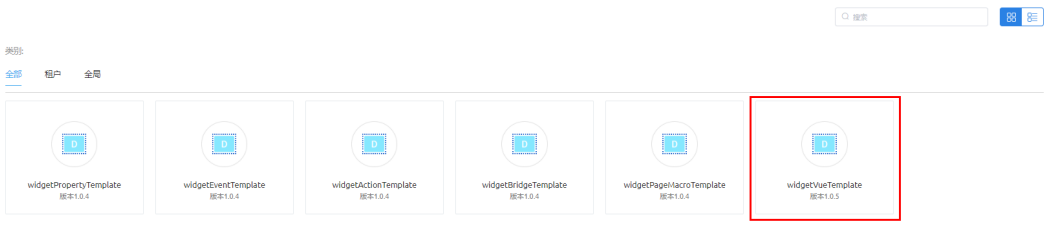


图 1-93 下载组件模板

步骤2 初识组件文件结构。

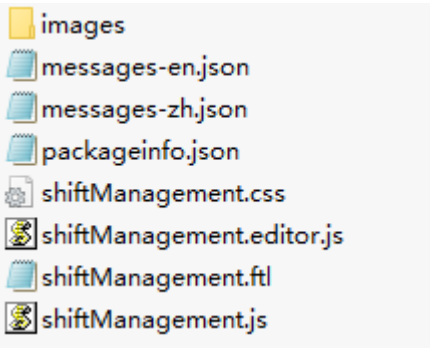
将下载到本地的组件包进行解压，使用您熟悉的开发工具进行开发。本章节以下载的shiftManagement组件为例，介绍组件包的文件结构以及各文件的功能。

表 1-11 组件文件结构

文件名	文件说明
shiftManagement.js	组件逻辑文件，整个Widget的渲染核心JS，在组件编辑状态和页面最终的发布运行态都会被加载执行。主要包含的预置API说明，请参见**高级页面组件中预置的API**。
shiftManagement.editor.js	组件属性定义文件，负责组件在编辑状态时，需要渲染的界面和逻辑。“*.editor.js”只在组件编辑状态被加载，主要包含： ● propertiesConfig：主要负责组件配置页面中，右侧的属性配置逻辑。 ● create方法仅在组件首次被创建时，调用一次。
shiftManagement.css	组件的样式文件，在该文件中编写组件的CSS样式。
shiftManagement.ftl	组件DOM结构文件，需要在服务端提前渲染的部分可以写在此文件中，相当于HTML文件，负责样式展示。
packageinfo.json	组件的元数据描述文件，定义了如下内容。 ● widegtApi name：组件的名称。 ● widgetDescription：组件的描述信息。 ● authorName: ：组件的作者信息。 ● localFileBasePath：组件本地调测路径。 ● i18n：指定组件的国际化资源文件（本例模板中未体现）。 ● requires：依赖的库名称和版本号。 ● width：在绝对布局高级页面中添加该组件时的默认宽度，单位为px，不填写默认为200px（本例模板中未体现）。 ● height：在绝对布局高级页面中添加该组件时的默认高度，单位为px，不填写默认为200px（本例模板中未体现）。
messages-zh/messages-en.json	组件的国际化资源文件，用于配置多语言（本例模板中未体现）。

步骤3 开发组件。

开发后的组件示例包。



主要文件说明：

- 在shiftManagement组件的shiftManagement.ftl中，实现DOM结构。
  页面标题和搜索区域示例：

```
<div id="shiftManagement">
    <!--头部区域-->
    <div class="header">
        <label class="title">班次管理</label>
    </div>
    <!--功能区域-->
    <div class="ribbon">
        <el-form :model="searchForm" ref="searchForm" class="searchForm">
            <el-form-item label="班次名称" :label-width="formLabelWidth" prop="shiftName" @change="">
                <el-input v-model="searchForm.shiftName" autocomplete="off" placeholder="请输入">
                </el-input>
            </el-form-item>
            <el-form-item label="班次类型" prop="shiftType" :label-width="formLabelWidth">
                <el-select v-model="searchForm.shiftType" placeholder="请选择" @change="" clearable>
                    <el-option v-for="item in shiftTypes" :key="item.label" :label="item.label" :value="item.value">
                    </el-option>
                </el-select>
            </el-form-item>
            <el-form-item label="所属班系" prop="shiftCategory" :label-width="formLabelWidth">
                <el-select v-model="searchForm.shiftCategory" placeholder="请选择" @change="" clearable>
                    <el-option v-for="item in shiftCategorys" :key="item.label" :label="item.label" :value="item.value">
                    </el-option>
                </el-select>
            </el-form-item>
        </el-form>
        <div slot="footer" class="dialog-footer">
            <el-button type="primary" @click="queryShiftInfo(searchForm)">搜 索</el-button>
            <el-button @click="reset">重 置</el-button>
        </div>
    </div>
```

页面标题

搜索区域

- 在shiftManagement组件的shiftManagement.js的render函数下，修改Vue实例。

```
        thisObj.vm = new Vue({
            el: $("#shiftManagement", elem)[0],
            i18n: i18n,
            data: {
                shiftData: [],    //班次列表数据
                multipleSelection: [],  //多选选中的内容
                recListPageInfo: {// 分页信息
                    currentPage: 0,
                    pageSize: 15,
                    totalCount: 0
                },
                searchName: '', //查询条件
                dialogFormVisible: false, //控制弹框显示与否
                importDialogFormVisible: false,
                searchForm: {
                    shiftName: "",
                    shiftType: "",
                    shiftCategory: ""
                },//搜索条件表单
                addForm: {
                    shiftName: "",
                    shiftType: "",
                    shiftStartTime: "",
```



```
    shiftEndTime: "",
    isRest: "",
    remark: "",
    repeatLaw: "",
    shiftCategory: "",
    period: "",
    workDay: ""
},//新建编辑表单
selectWorkDayList: [],
// 设置只能选择当前日期及之后的日期
pickerOptions: {
    disabledDate(time) {
        //如果没有后面的-8.64e7就是不可以选择今天的
        return time.getTime() < Date.now() - 8.64e7;
    }
},
rules: {//表单校验规则
    shiftName: [
        { required: true, message: '请输入班次名称', trigger: 'blur' }
    ],
    shiftType: [
        { required: true, message: '请选择班次类型', trigger: 'blur' }
    ],
    shiftStartTime: [
        { required: true, message: '请选择上班时间', trigger: 'blur' }
    ],
    shiftEndTime: [
        { required: true, message: '请选择下班时间', trigger: 'blur' }
    ],
    repeatLaw: [
        { required: true, message: '请选择重复规律', trigger: 'blur' }
    ],
},
formLabelWidth: '120px',
formLabelWidth1: '80px',
baseUrl: widgetBasePath,
rowId: "",
selectRowIdList: [],    //选中的rowId集合
disabled: false, //控制输入框是否可用
title: "", //弹窗标题
shiftTypes: [],//班次类型下拉列表
shiftCategorys: [],//所属班系下拉列表
operType: "",
activeNames: ['1', '2'],
isRestList: [
    {
        "label": "是",
        "value": "是"
    },
    {
        "label": "否",
        "value": "否"
    }
],
importVisible: false,
fileList: [],   //文件列表
baseUrl: widgetBasePath,
autoDeleteFile: false,
importLoading: false,
fileContentArr: [], //文件解析后的内容
isLoading: false,
dayList: [
    {
        "label": "1",
        "value": "1"
    },
    {
        "label": "2",
        "value": "2"
```

```
}, {
    "label": "3",
    "value": "3"
},
{
    "label": "4",
    "value": "4"
}, {
    "label": "5",
    "value": "5"
},
{
    "label": "6",
    "value": "6"
}, {
    "label": "7",
    "value": "7"
},
{
    "label": "8",
    "value": "8"
}, {
    "label": "9",
    "value": "9"
},
{
    "label": "10",
    "value": "10"
}, {
    "label": "11",
    "value": "11"
},
{
    "label": "12",
    "value": "12"
}, {
    "label": "13",
    "value": "13"
},
{
    "label": "14",
    "value": "14"
}, {
    "label": "15",
    "value": "15"
},
{
    "label": "16",
    "value": "16"
}, {
    "label": "17",
    "value": "17"
},
{
    "label": "18",
    "value": "18"
}, {
    "label": "19",
    "value": "19"
},
{
    "label": "20",
    "value": "20"
}, {
    "label": "21",
    "value": "21"
},
{
    "label": "22",
```



```
                "value": "22"
            }, {
                "label": "23",
                "value": "23"
            },
            {
                "label": "24",
                "value": "24"
            }, {
                "label": "25",
                "value": "25"
            },
            {
                "label": "26",
                "value": "26"
            }, {
                "label": "27",
                "value": "27"
            },
            {
                "label": "28",
                "value": "28"
            }, {
                "label": "29",
                "value": "29"
            },
            {
                "label": "30",
                "value": "30"
            }, {
                "label": "31",
                "value": "31"
            },
        ],
        workDate: "" //系统参数获取的工作日
    },
    async created() {
        this.dayList.forEach(item => {
            item.ischecked = false;
        });
        this.queryOptions();
        this.queryShiftInfo(this.searchForm, this.recListPageInfo.pageSize);
        await this.getSysParameter(["ISDP__workDate"])
    },
    methods: {
        /**
         * 编辑
         */
        handleEdit: function (index, row) {
            this.openDialog("modify", row);
        },
        /**
         * 删除
         */
        handleDelete: function (index, row) {
            let _this = this;
            if (!row && _this.selectRowIdList.length == 0) {
                _this.$message({
                    type: 'warning',
                    message: '请选择至少一条数据后再删除!'
                });
                return false;
            } else {
                _this.$confirm('此操作将删除该记录, 是否继续?', '提示', {
                    confirmButtonText: '确定',
                    cancelButtonText: '取消',
                    type: 'warning'
                }).then(() => {
                    let params = {
```

```
                    "idlist": row ? [row.id] : _this.selectRowIdList
                }
                isdpPlusUtile.sendReqest("/service/ISDP__IntelligentSchedulingModel/1.0.1/
batchDelShiftInfo", params, function (resp) {
                    if (resp.resCode == "0") {
                        _this.$message({
                            type: 'success',
                            message: '删除成功!'
                        });
                        _this.selectRowIdList = [];
                        _this.queryShiftInfo(_this.searchForm, _this.recListPageInfo.pageSize);
                    } else {
                        _this.$message.error('请求失败，请联系管理员');
                    }
                }, function (err) {
                    let msg = error.response.resMsg ? error.response.resMsg : '删除失败，请联系管
理员'
                    _this.$message.error(msg)
                });
            }).catch(() => {
                _this.$message({
                    type: 'info',
                    message: '已取消删除'
                });
            });
        }
    },
    ......
    ......
```

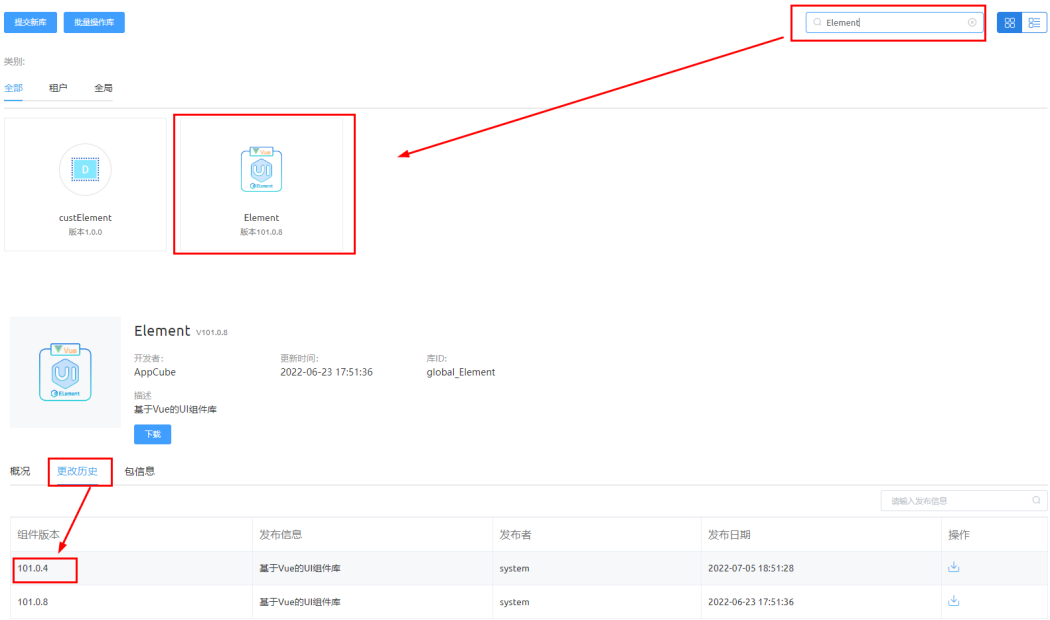
- 在shiftManagement组件的shiftManagement.css中，实现CSS样式。

```
#shiftManagement {
    background-color: #ffffff;
    font-family: HarmonyOS Sans SC, HarmonyOS Sans SC-Regular;
    height: 100%;
}
#page1 {
    background-color: #f5f6f7;
}
.header {
    margin: 24px;
}
.ribbon {
    margin: 16px 24px;
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 100%;
}
.oper-box {
    display: flex;
    flex-direction: row;
    align-items: center;
}
......
......
```

步骤4 定义组件依赖库。

本节开发的示例组件shiftManagement依赖Vue、VueI18n和Element库，所依赖的Vue库已在之前选择的组件模板“widgetVueTemplate”中定义，这里只需要在shiftManagement组件包的packageinfo.json文件中定义所依赖的VueI18n和Element库即可。

1. 在左侧导航栏中，单击  ，选择“高级页面 > 库”。
2. 搜索库，查看库ID和版本号信息（以查询Element库为例）。



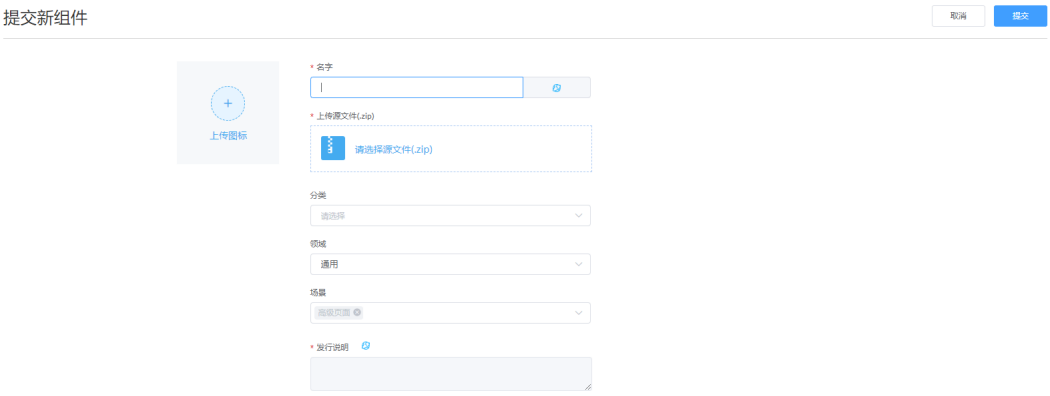
本例中查询的Element库ID为：global_Element，库最新版本号为：101.0.4。

3. 继续查找VueI18n库的库ID和最新版本号。
4. 在shiftManagement组件包的packageinfo.json中，修改requires属性。Vue库已在之前选择的组件模板“widgetVueTemplate”中定义，无需修改。

```
"requires": [
  {
    "name": "global_Vue",
    "version": "100.7"
  },
  {
    "name": "global_VueI18n",
    "version": "100.7.5"
  },
{
    "name": "global_Element",
    "version": "101.0.4"
  }
]
```

步骤5 上传自定义组件。

1. 将开发好的组件代码压缩到后缀为.zip的压缩文件（shiftManagement.zip）。
2. 选择“高级页面 > 组件”，单击“提交新组件”。
3. 在提交新组件页面，设置组件基本信息，并上传压缩文件，单击“提交”。



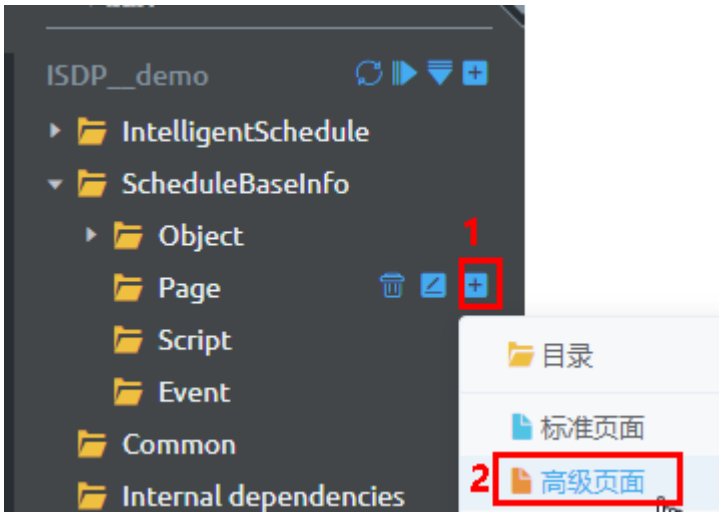
----结束

创建页面

步骤1 如图1-94所示，在应用的开发页面，选择存放页面的目录，单击目录对应的，选择“高级页面”。

例如，存放在“ScheduleBaseInfo > Page”目录下。

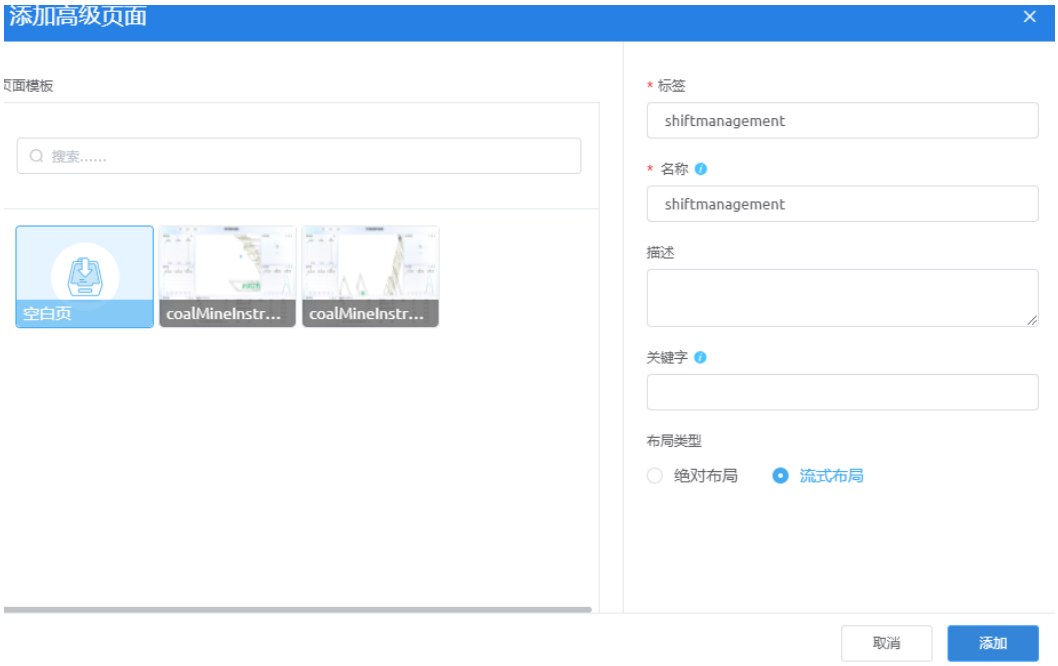
图 1-94 创建高级页面



步骤2 如图1-95所示，在弹出的“添加高级页面”页面，选择“空白”，输入标签和名称，选择布局类型为“流式布局”，单击“添加”。

页面创建完成后，自动进入编辑页面。

图 1-95 添加高级页面

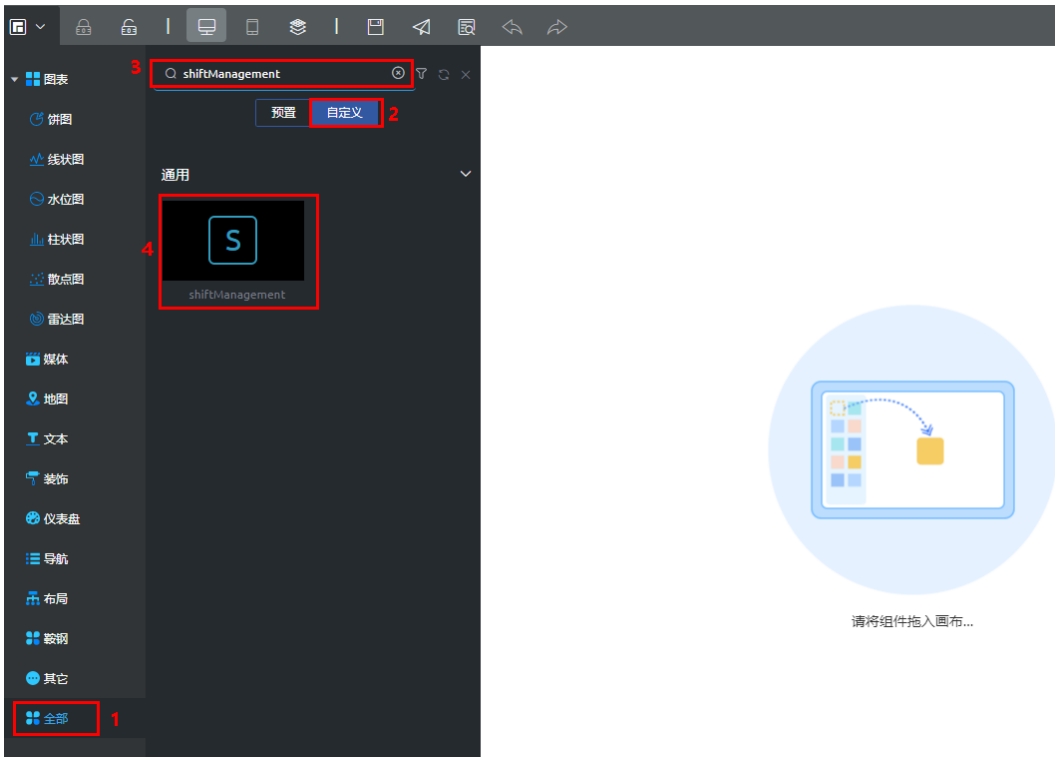


----结束

开发页面

步骤1 如图1-96所示，选择自定义组件“shiftManagement”，拖入到页面内容区域。

图 1-96 选择组件

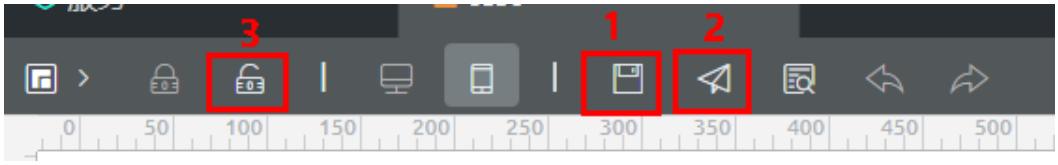


步骤2 单击页面上方保存按钮，保存配置，单击发布，发布页面，最后单击释放锁按钮，退出编辑状态。

说明

如果需要再次编辑，需要单击  ，获取锁在进行编辑。

图 1-97 保存，发布并释放锁



----结束

预览页面

如图1-98所示，单击页面上方预览按钮，即可看到页面配置效果，如图1-99所示。

图 1-98 预览

图 1-99 预览效果

1.3.5 打包发布应用

应用开发完成后，需要将应用进行编译打包发布操作，打包后该应用才能发布使用。

步骤1 在应用中（demo），如图1-100所示，单击  ，选择“设置”。

图 1-100 属性设置

步骤2 设置软件包，参数配置说明请参见表1-12，单击“保存”。

表 1-12 参数说明

参数	配置说明	示例
包类型	应用包的类型： ● 选择“资产包”发布的应用包，包中组件可设置是否受保护。 选择该项后，您需要配置版权信息（可选）、描述（可选）、每个组件的保护设置（必选。配置为未受保护或者只读保护）。 若后续其他用户在开发环境安装后，会显示在开发环境首页的“库”页签下。 ● 选择“源码包”发布的应用包，包中的所有组件不受保护和限制。在其他环境安装后可编辑包中组件，即在原有基础上进行再开发。 若后续其他用户在开发环境安装后，会显示在开发环境首页的“项目”页签下。	资产包
是否全量包	打包时是否打全量包： ● 全量包：表示对整个应用（包括应用中的各个组件）作为一个整体进行设置打包。 ● 增量包：对应用中部分组件进行设置打包。单击“添加应用组件”，在“类别”中选择相应的类别，勾选需要打包的组件。	全量包
产权设置 > 加密保护	当选择“资产包”打包时，才会显示该参数。表示是否对打包的组件中敏感内容（例如脚本内容）是否进行加密。 勾选表示加密。在其他环境安装前，包中敏感数据是经过加密的。	否
产权设置 > 版权信息	当选择“资产包”打包时，才会显示该参数。表示该包的版权信息。 选填项。	-

参数	配置说明	示例
产权设置 > 描述信息	当选择“资产包”打包时，才会显示该参数。表示该包的描述信息。 选填项，建议描述该App提供的功能。	智能排班
产权设置 > 联系邮件	您可以在此留下当前软件包的问题联系邮箱，若安装过程中出现问题会将联系邮件提示给使用者。	-
产权设置 > 联系链接	您可以在此留下当前软件包的文档链接，若安装过程中出现问题会将联系链接提示给使用者。	-
产权设置 > 资产保护 > 保护模式	当选择“资产包”打包时，才会显示该参数。打包数据的保护模式。 ● 未受保护 ● 只读保护 ● 不可见保护 未受保护的资产包在开发环境中安装后，可进行二次编辑；在运行环境安装资产包后，不论保护模式是“未受保护”还是“只读保护”，都不可编辑。	只读保护
部署策略 > 安装前置脚本	当选择“资产包”打包且打全量包时，该配置页才会显示。表示在安装应用包时，在导入实例化配置数据之前执行的脚本。一般用于预清理数据，避免数据冲突的情况。 您可以选择已有脚本，也可以单击“创建”新建脚本。	-
部署策略 > 安装后置脚本	当选择“资产包”打包且打全量包时，该配置页才会显示。表示在安装应用包时，在导入实例化配置数据之后执行的脚本。一般用于删除、更新数据等。 您可以选择已有脚本，也可以单击“创建”新建脚本。	-
部署策略 > 安装时组件的更新策略	当打包的组件中包含系统参数、连接器、Rest操作、数据接入或事件流时，才会显示该参数。例如可设置系统参数随包打包发布后，在升级时遇到新旧数据冲突（唯一索引相同的数据）的数据更新策略。 ● 覆盖：当相关组件数据在打包升级到其他环境，发生数据冲突时，会进行覆盖。 ● 不覆盖：当相关组件数据在打包升级到其他环境，发生数据冲突时，不会进行覆盖。 默认“不覆盖”。在配置为“不覆盖”的情况下，例如在开发环境修改数据接入的任意配置数据（包括所有图元的配置信息），打包升级到测试或者运行环境，不会覆盖同名的数据接入配置，即在开发环境修改的数据在测试或者运行环境不会生效。	覆盖

参数	配置说明	示例
预置数据	当选择“资产包”打包时，该配置页才会显示。 您可以在该页面选择您在应用打包时一起发布的数据。支持按照对象名称打包。单击“添加对象”可设置数据导出条件，选择对象后，在应用打包时，会将该对象的满足条件的数据都打包出来。打包后，在资产包中“refdata”文件夹下可查看到导出的数据文件。 使用该方式前，您需要先清理不需要发布的数据，且导出对象的“基本信息”页必须勾选上“允许API批量访问”。	-

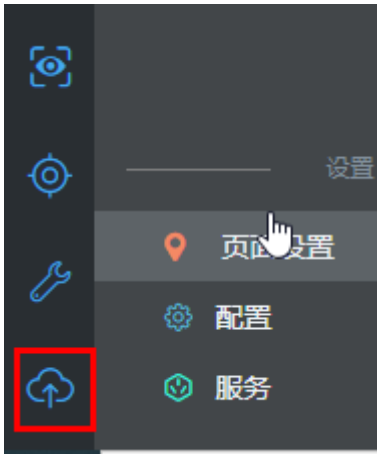
步骤3 如图1-101所示，单击  ，选择“编译”，编译完成如图1-102所示。

图 1-101 编译

图 1-102 编译完成

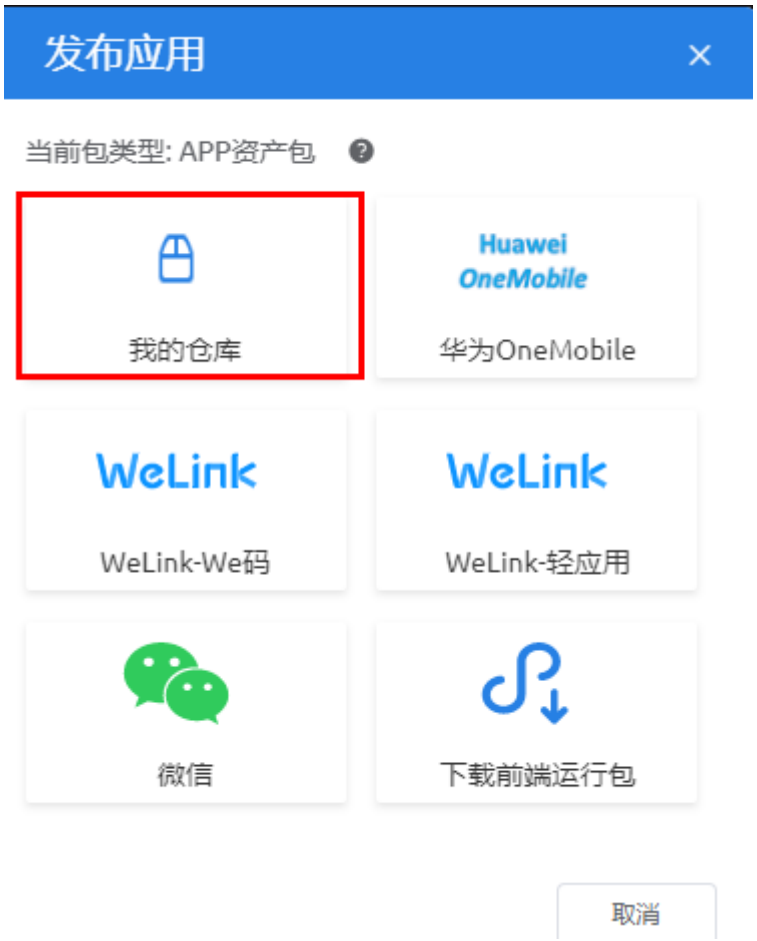
步骤4 如图1-103所示，单击  。

图 1-103 发布



步骤5 如图1-104所示，在弹出的页面中选择“我的仓库”。

图 1-104 发布应用



步骤6 如图1-105所示，填写版本信息，单击“发布”。

图 1-105 发布到我的仓库

说明

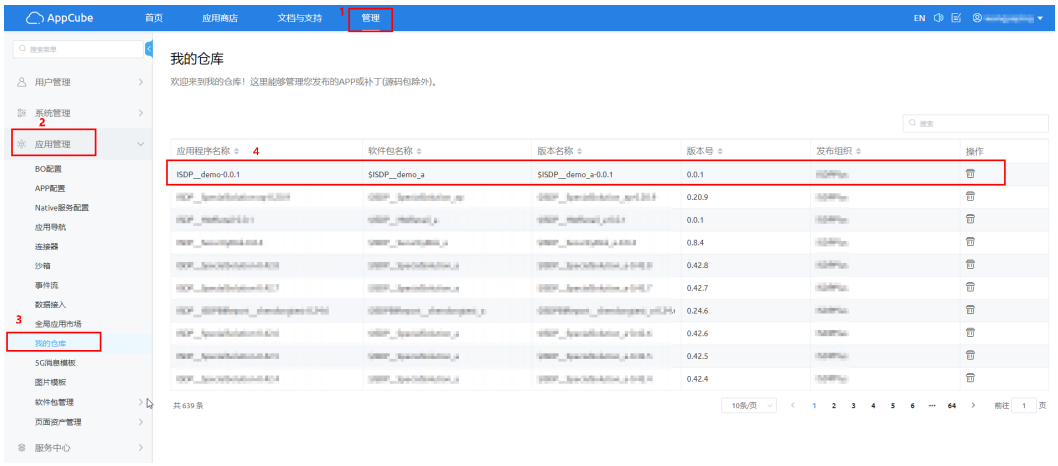
如果勾选“压缩高级页面”，表示会对包中所有高级页面涉及的css和js文件进行合并及压缩，这样可以有效降低运行时服务器压力，但从终端浏览器首次访问该站点页面时，访问时间会稍微增加。

发布成功后，页面显示“程序包已经被成功上传到我的仓库。”

如图1-106所示，可以查看并下载发布的包，也可以在“管理 > 应用管理 > 我的仓库”中查看到发布的包，如图1-107所示。

图 1-106 发布的包

图 1-107 我的仓库



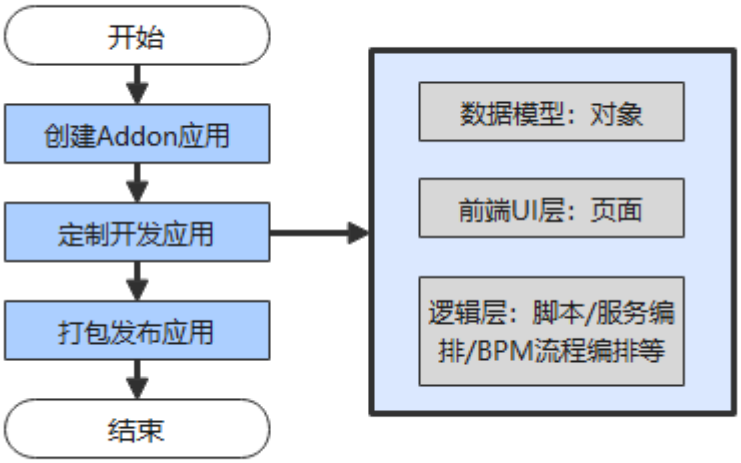
----结束

1.4 定制开发指导（基于智能排班基线应用）

本章节介绍如何基于智能排班基线应用进行二次开发。

1.4.1 定制开发流程

图 1-108 应用定制开发流程图



序号	步骤	说明
1	创建Addon应用	在进行应用开发之前，需要先创建应用。 具体操作请参见**创建Addon应用**。
5	定制开发应用	在应用的开发态，定制开发应用，需要进行数据模型、逻辑层以及前端UI层等的开发。 具体操作请参见**定制开发应用**。
6	打包发布应用	安装部署应用，部署到沙箱环境中进行测试。 具体操作请参见**打包发布应用**。

1.4.2 创建 Addon 应用

操作步骤

步骤1 登录AppCube开发环境。

步骤2 如图1-109所示，在开发环境首页的“项目”页签下，单击“行业应用”，进入到行业应用。

图 1-109 进入行业应用

步骤3 如图1-110所示，单击“创建Addon应用”，弹出“创建Addon应用”页面。

图 1-110 创建 Addon 应用

步骤4 如图1-111所示，添加图标、设置标签、名称、分类和描述，依赖的应用，具体参数说明请参见表1-13，单击“创建”，创建应用后，进入应用开发阶段，如图1-112所示。

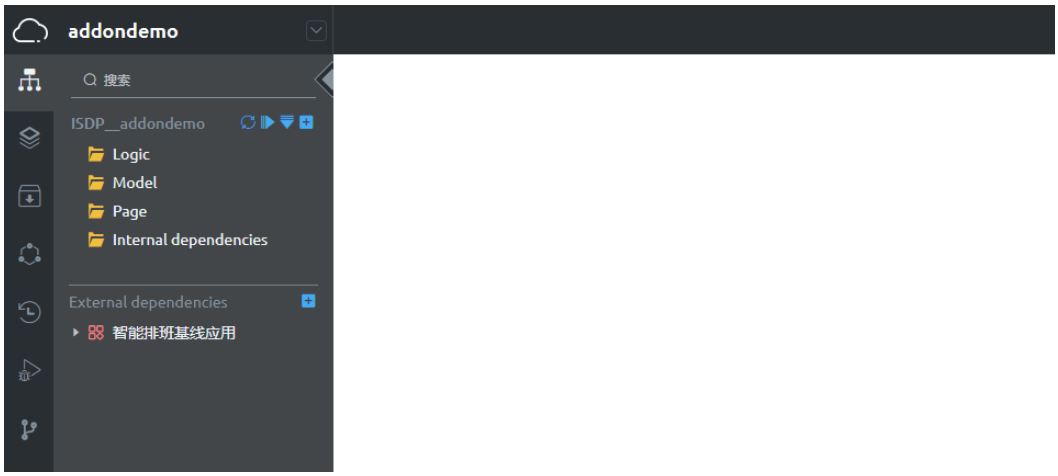
图 1-111 创建 Addon 应用

表 1-13 应用基本信息

参数	配置说明	示例
添加图标	为该应用设置图标。如果不设置，则使用默认图标。 单击“添加图标”，在弹出“图标选择”页面中选择图标。	使用默认图标
标签	应用中用于展示的文字，为了区分不同应用的描述信息，创建后可修改。 应用创建之后，应用标签可以在应用设置中修改。	addondemo
名称	应用在系统内的唯一标识，系统会自动在该名称前添加*命名空间*__，创建后不支持修改。 设置要求：必须以字母开头，没有连续的下划线，空格和特殊字符。	addondemo **说明** 应用创建后，应用名称自动添加命名空间前缀，例如：ISDP__addondemo。

参数	配置说明	示例
分类	应用所属的分类。 设置分类后，工程列表和库列表都可以根据应用的分类进行筛选。	-
描述	应用的描述信息。	-
高级设置	展开“高级设置”时才会显示该参数。开发的资产包依赖所选择的运行时版本。	保持默认
依赖	选择依赖的应用：智能排班基线应用。	智能排班基线应用

图 1-112 应用开发页面



----结束

1.4.3 定制开发应用

1.4.3.1 基于基线应用组件开发高级页面

本节介绍如何基于智能排班基线应用的组件，进行前端高级页面的定制开发。

1.4.3.1.1 场景介绍

使用智能排班基线应用中组件“shiftManagement”，定制开发页面。

1.4.3.1.2 开发流程

序号	步骤	说明
1	下载基线组件	下载基线组件zip包到本地。 具体操作请参见**下载基线组件**。

序号	步骤	说明
2	参考基线组件开发新组件	开发前，先替换组件包中的文件名称，以及文件内容中的组件名称，全部替换为新组件的名称。 具体操作请参见**参考基线组件开发新组件**。
3	上传新组件	打包上传到AppCube的全局组件。 具体操作请参见**上传新组件**。
4	开发页面	使用新组件开发高级页面。 具体操作请参见**开发页面**。

1.4.3.1.3 下载基线组件

操作步骤

步骤1 在左侧导航栏中，单击  ，选择“高级页面 > 组件”。

步骤2 搜索到组件（shiftManagement）。

图 1-113 搜索组件

步骤3 单击进入组件后，在组件详情页中单击“下载”，设置保存名称（例如，shiftManagement），选择本地保存地址，单击“保存”。

图 1-114 下载组件

----结束

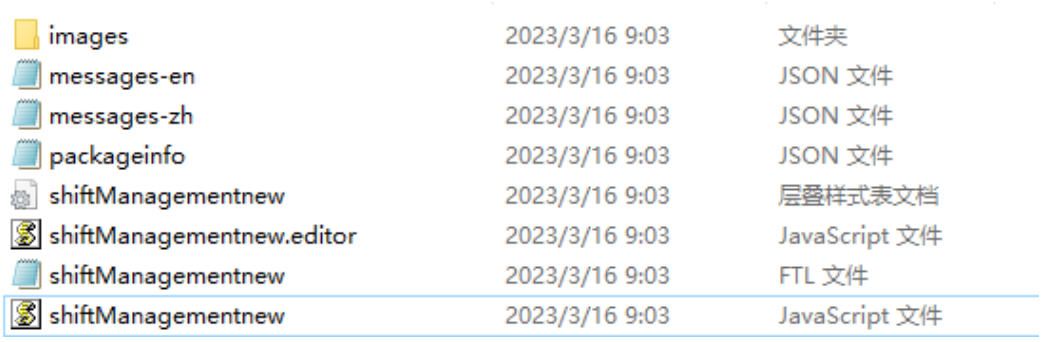
1.4.3.1.4 参考基线组件开发新组件

操作步骤

步骤1 解压下载到本地的组件包（例如，shiftManagement.zip）。

步骤2 将文件名中含有的“shiftManagement”全部都修改为新的名称，例如“shiftManagementnew”。

实际开发过程中，文件名和组件名可以自定义，但是两者必须一致。



步骤3 打开除了图片文件夹外的所有文件，全文搜索“shiftManagement”，并替换为“shiftManagementnew”。

说明

下载后的基线组件中的所有文件，都要搜索替换组件名称，避免上传后覆盖原有的基线组件。

- 如果packageinfo.json中的组件名没有修改，那么后面在上传该组件时会覆盖基线组件，可能造成基线APP不可用。
- 如果其它文件没有修改完全，会造成该组件功能异常，例如无法根据路径显示图片等。

步骤4 组件重命名完成后，根据需求修改7个文件中的代码，开发新的组件。

组件代码的修改，请参见**开发页面**中的说明。

步骤5 组件开发完成后，选中文件夹内的所有的文件压缩成zip包。

须知

选中所有文件后压缩，不要对文件夹进行压缩，且压缩文件必须为zip格式。

----结束

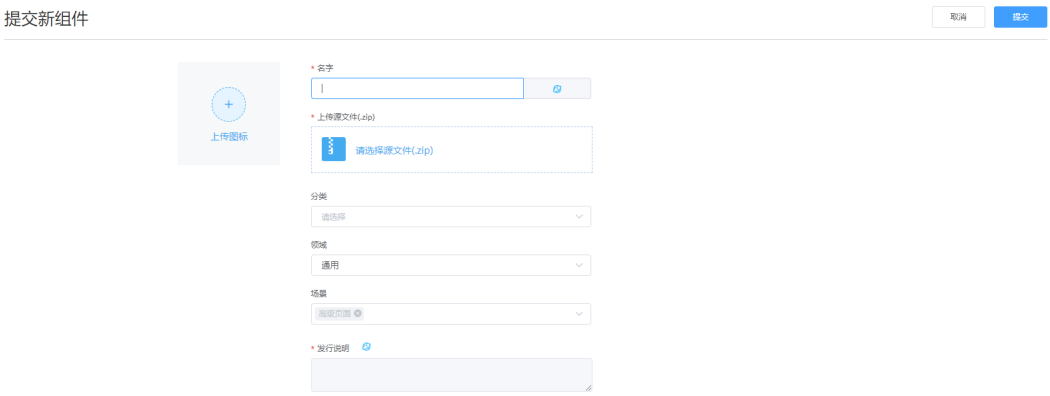
1.4.3.1.5 上传新组件

操作步骤

步骤1 将开发好的组件代码压缩到后缀为.zip的压缩文件（shiftManagementnew.zip）。

步骤2 选择“高级页面 > 组件”，单击“提交新组件”。

步骤3 在提交新组件页面，设置组件基本信息，并上传压缩文件，单击“提交”。



----结束

1.4.3.1.6 开发页面

创建一个高级页面，拖入组件进行开发和预览，具体操作请参见**创建页面**～**预览页面**。

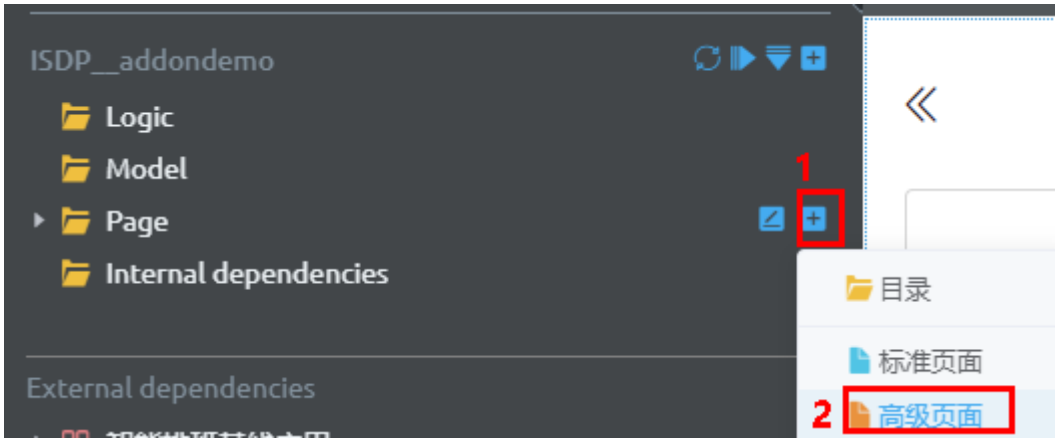
1.4.3.2 基于基线应用页面开发新页面

本节介绍如何基于智能排班基线应用的页面，进行前端页面的定制开发。

操作步骤

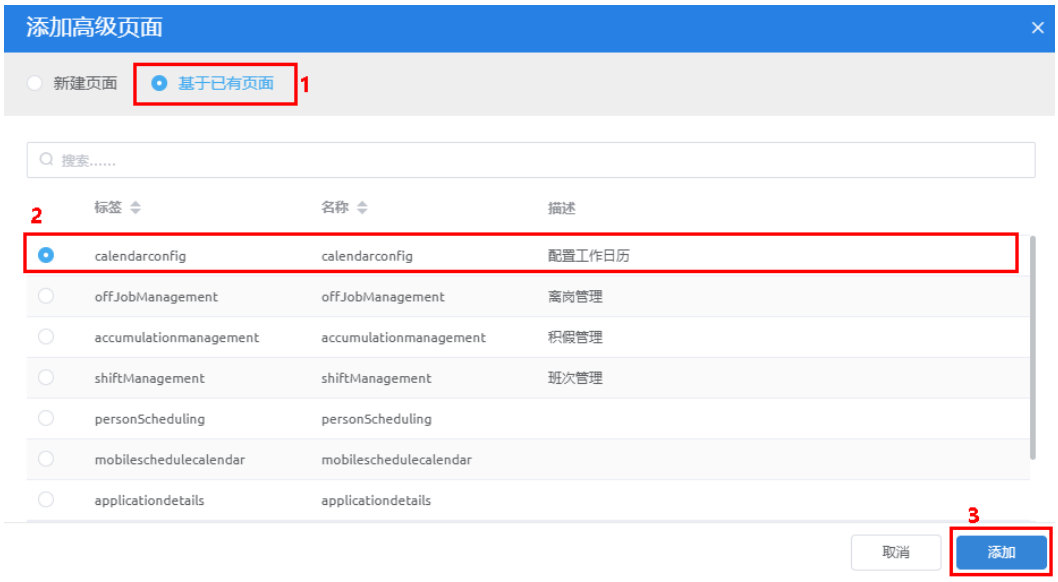
步骤1 如**图1-115**所示，鼠标放在Addon应用定制目录下的Page文件夹旁会出现加号，单击加号，选择“高级页面”。

图 1-115 进入添加高级页面



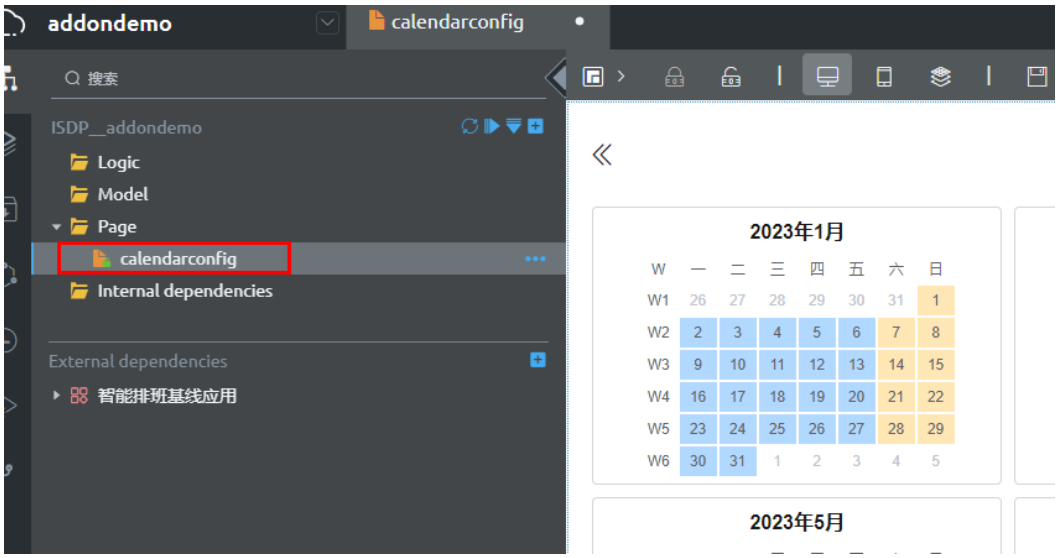
步骤2 如图1-116所示，在“添加高级页面”中选择“基于已有页面”，勾选基于的页面，单击“添加”。

图 1-116 添加高级页面



添加后，在Addon应用定制目录下的Page文件夹下会出现该定制页面，可以基于该定制页面进行开发。

图 1-117 添加结果



步骤3 进行高级页面的开发，具体过程请参见**高级页面**。

----结束

1.4.3.3 基于基线应用对象开发新对象

本节介绍如何基于智能排班基线应用的对象，进行对象的定制开发。

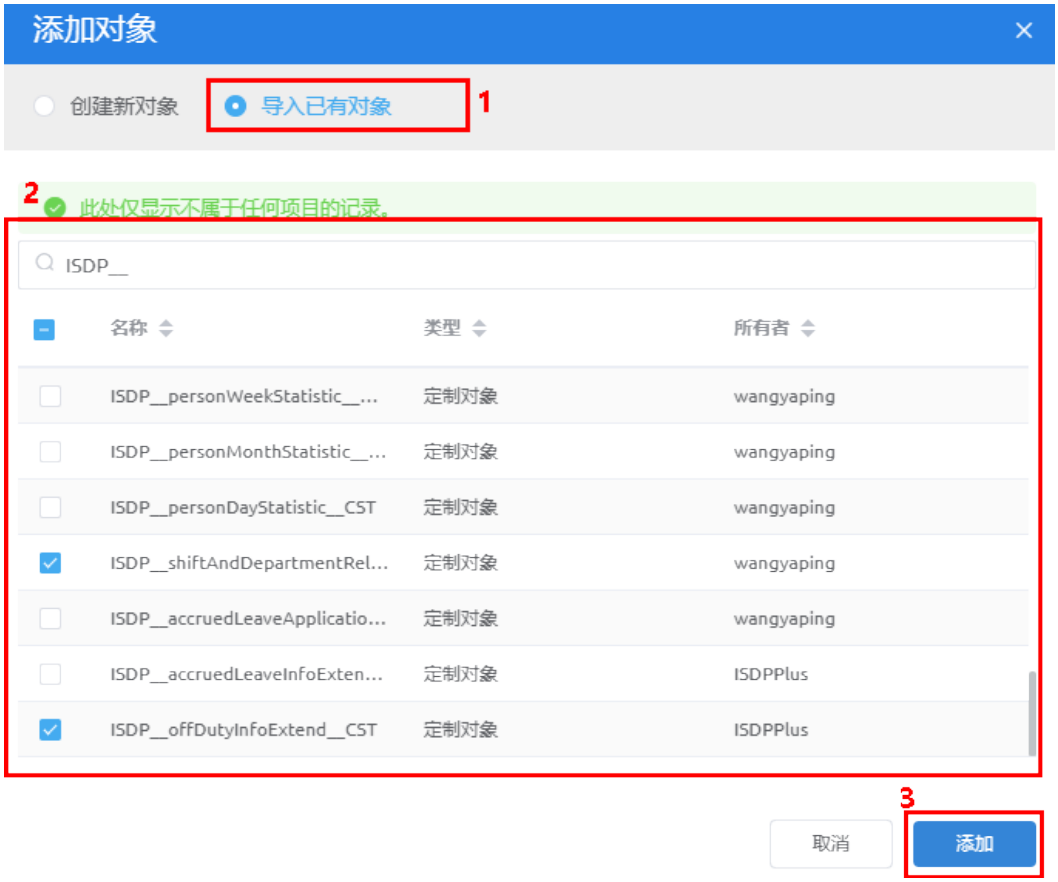
操作步骤

步骤1 如图1-118所示，鼠标放在Addon应用定制目录下的Model文件夹旁会出现加号，单击加号，选择“对象”。

图 1-118 进入添加对象页面

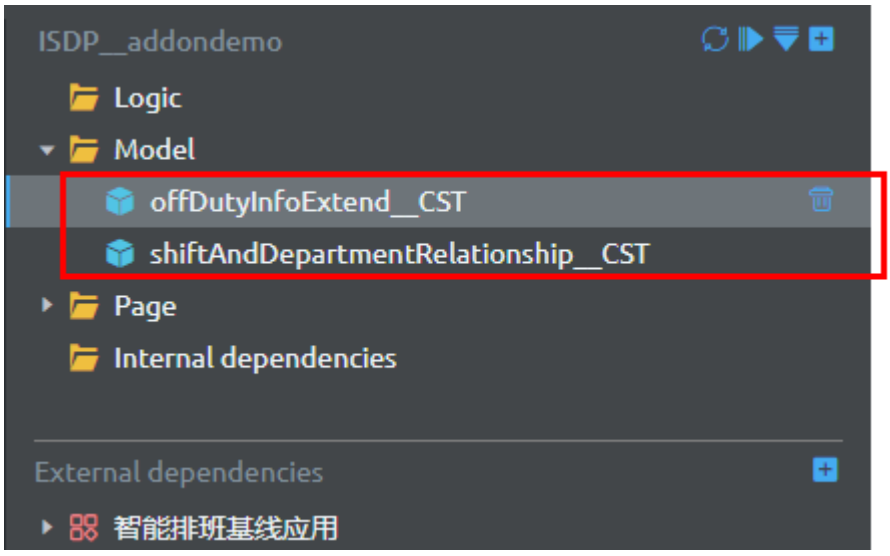
步骤2 如图1-119所示，在“添加对象”中选择“导入已有对象”，勾选导入的对象，单击“添加”。

图 1-119 添加对象



添加后，在Addon应用定制目录下的Model文件夹下会出现该定制对象，可以基于该定制对象进行开发。

图 1-120 添加结果



步骤3 进入到对象，在对象中对字段进行增删改等操作，具体请参见**对象**。

----结束

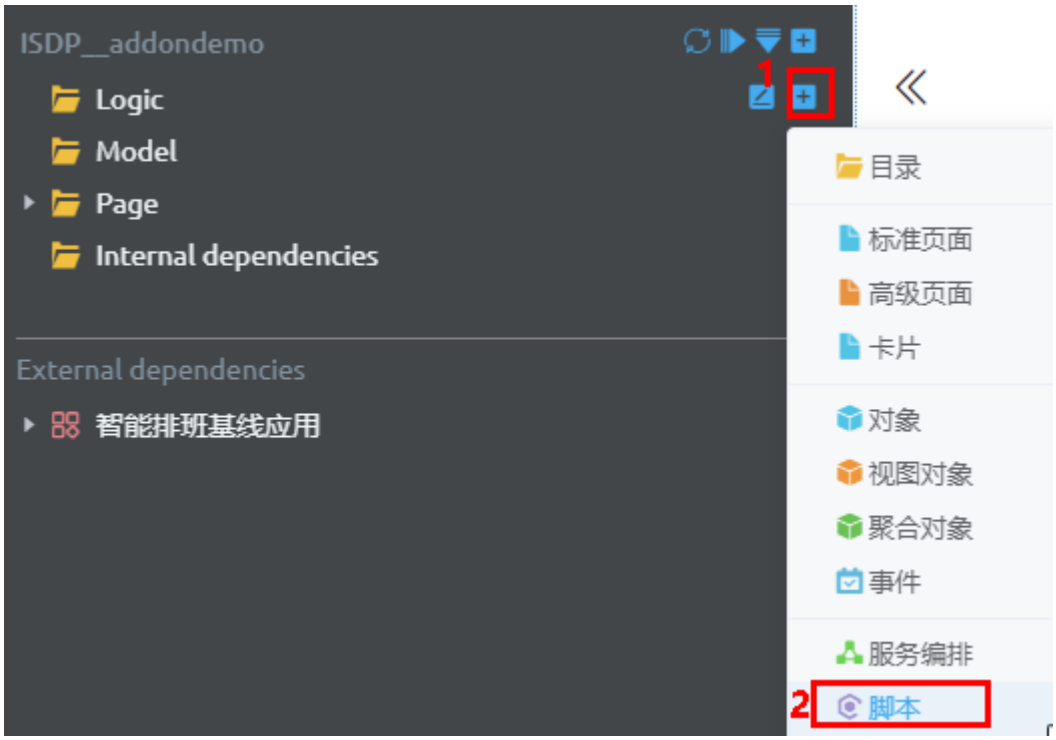
1.4.3.4 基于基线应用脚本开发新脚本

本节介绍如何基于智能排班基线应用的脚本，进行脚本的定制开发。

操作步骤

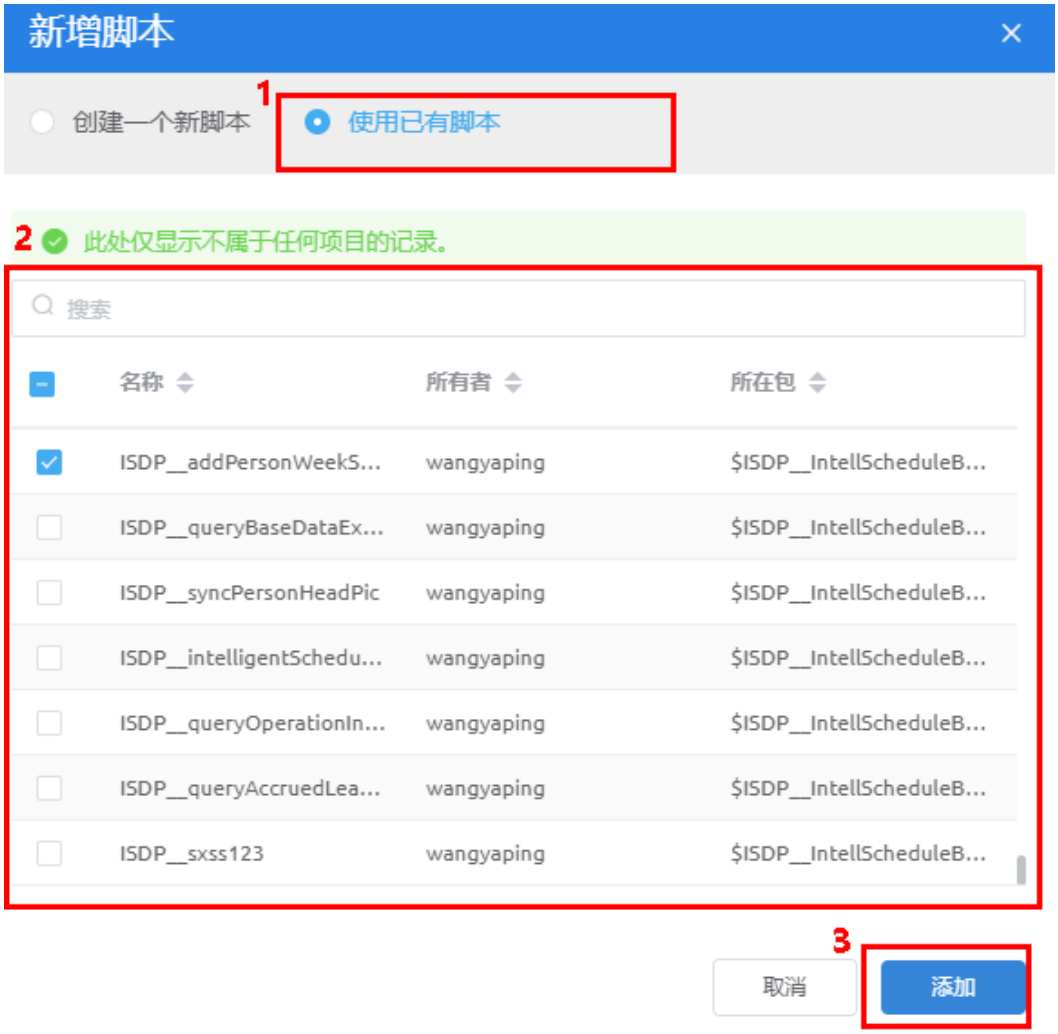
步骤1 如图1-121所示，鼠标放在Addon应用定制目录下的Logic文件夹旁会出现加号，单击加号，选择“脚本”。

图 1-121 进入添加脚本页面



步骤2 如图1-122所示，在“添加脚本”中选择“使用已有脚本”，勾选需要依赖的脚本（支持多选），单击“添加”。

图 1-122 添加脚本

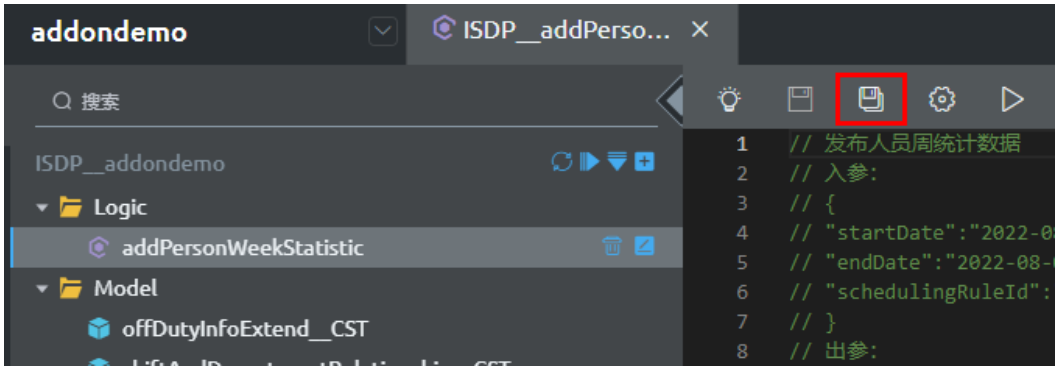


添加后，在Addon应用定制目录下的Logic文件夹下会出现该定制脚本。

图 1-123 添加结果

步骤3 如图1-124所示，单击定制的脚本，右侧展示该脚本的编译页面，单击  。

图 1-124 复制脚本



步骤4 如图1-125或图1-126所示，在弹出的“脚本详情”页面，根据规划，选择“新建版本”或“新建脚本”，配置脚本信息，单击“保存”。

图 1-125 新建版本

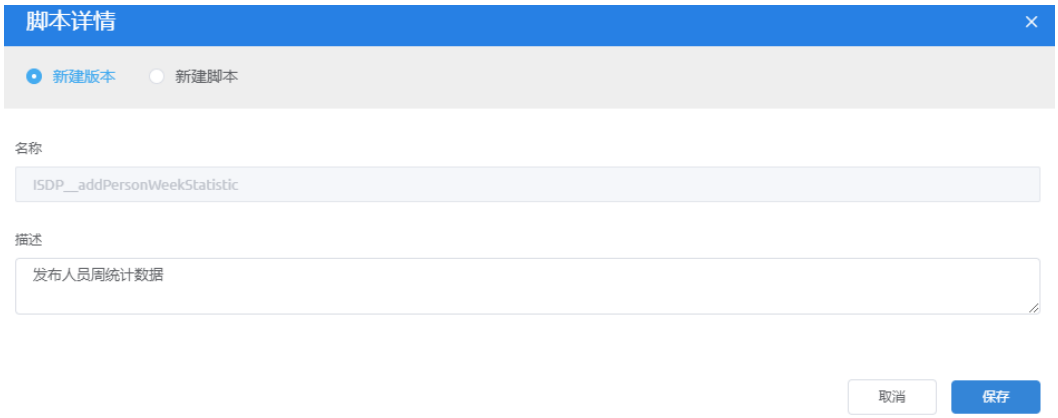
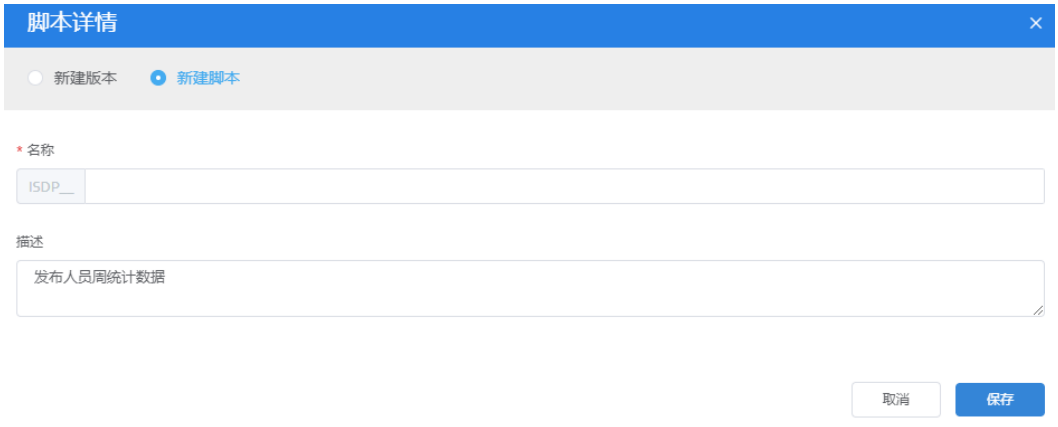


图 1-126 新建脚本

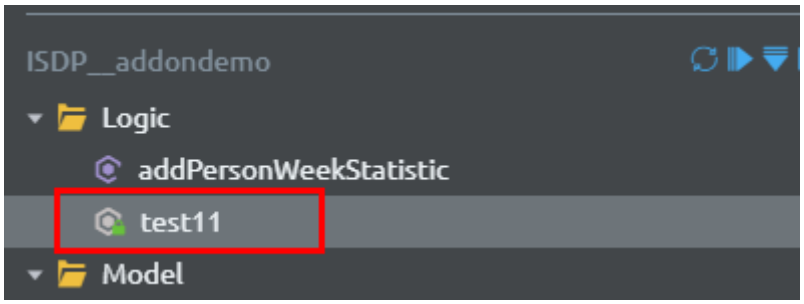


- 新建版本后，在当前脚本编辑区域右上角可以查看到当前脚本包含的版本，选择新建的版本信息，进行脚本的定制开发。

图 1-127 新建的版本

- 新建脚本后，在左侧可以看到新建的脚本，单击脚本右侧展示脚本编辑页面，进行脚本的定制开发。

图 1-128 新建的脚本



步骤5 进行脚本的开发，具体过程请参见**脚本开发**。

----结束

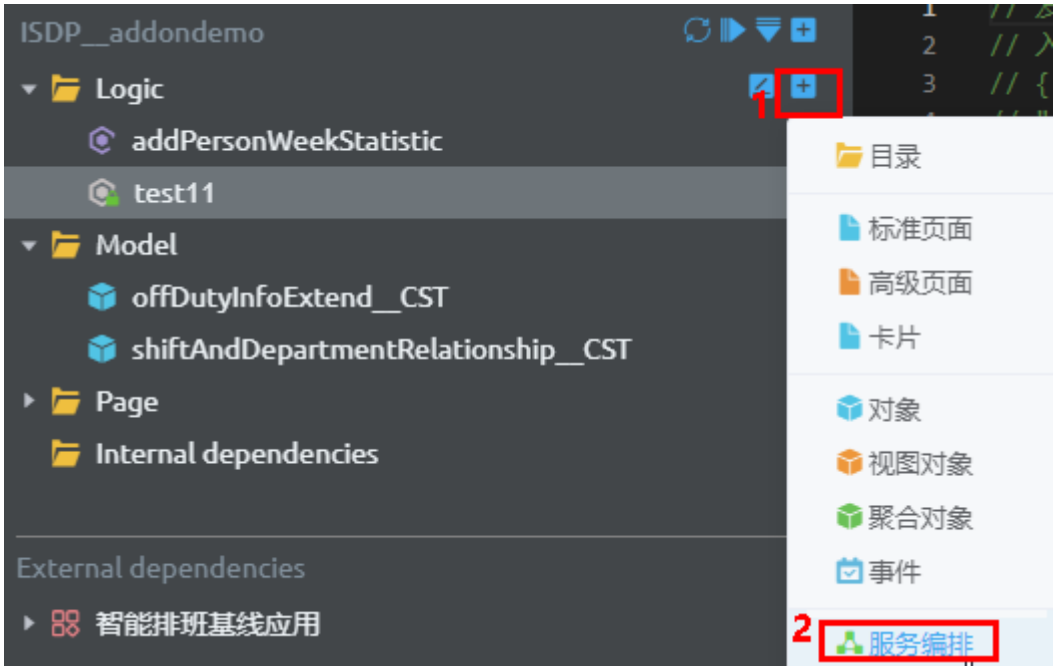
1.4.3.5 基于基线应用服务编排开发新服务编排

本节介绍如何基于智能排班基线应用的服务编排，进行服务编排的定制开发。

操作步骤

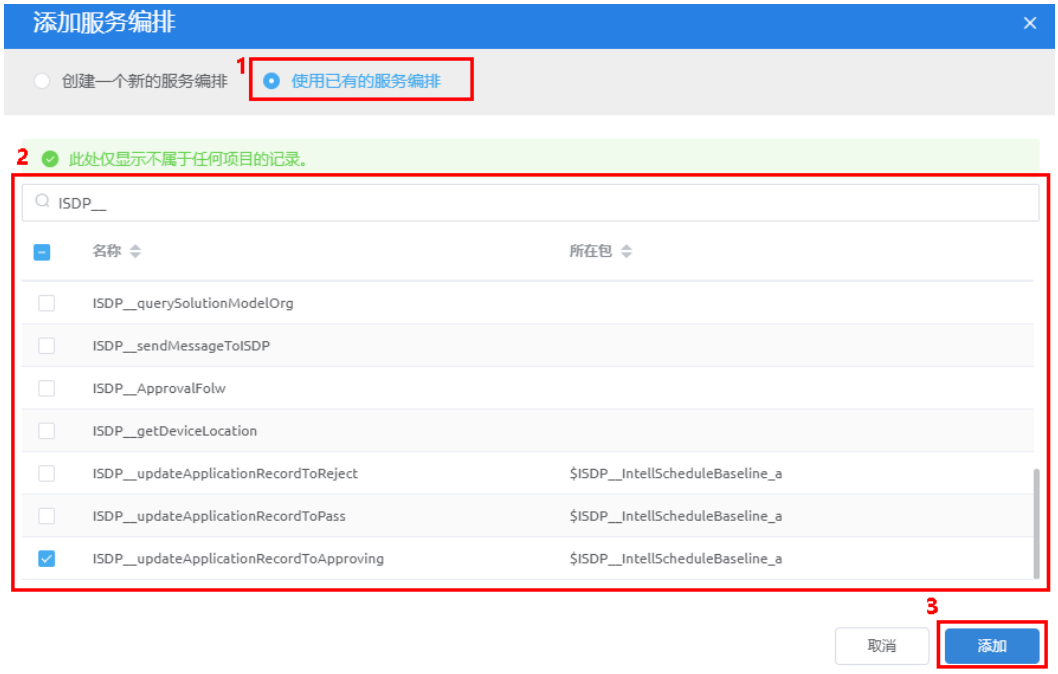
步骤1 如**图1-129**所示，鼠标放在Addon应用定制目录下的Logic文件夹旁会出现加号，单击加号，选择“服务编排”。

图 1-129 进入添加服务编排页面



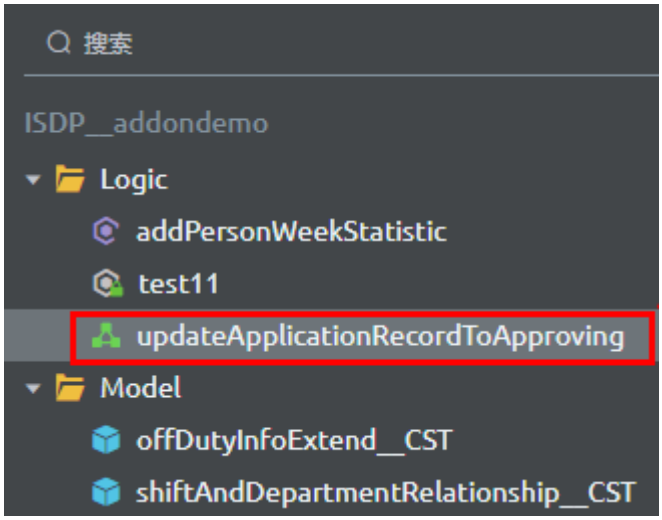
步骤2 如图1-130所示，在“添加服务编排”中选择“使用已有的服务编排”，勾选需要依赖的服务编排（支持多选），单击“添加”。

图 1-130 添加服务编排



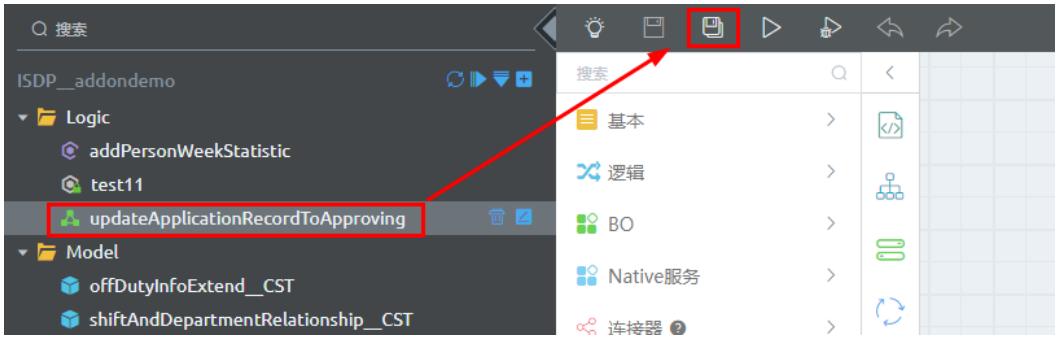
添加后，在Addon应用定制目录下的Logic文件夹下会出现该定制脚本。

图 1-131 添加结果



步骤3 如图1-132所示，单击定制的服务编排，右侧展示该服务编排的编译页面，单击  。

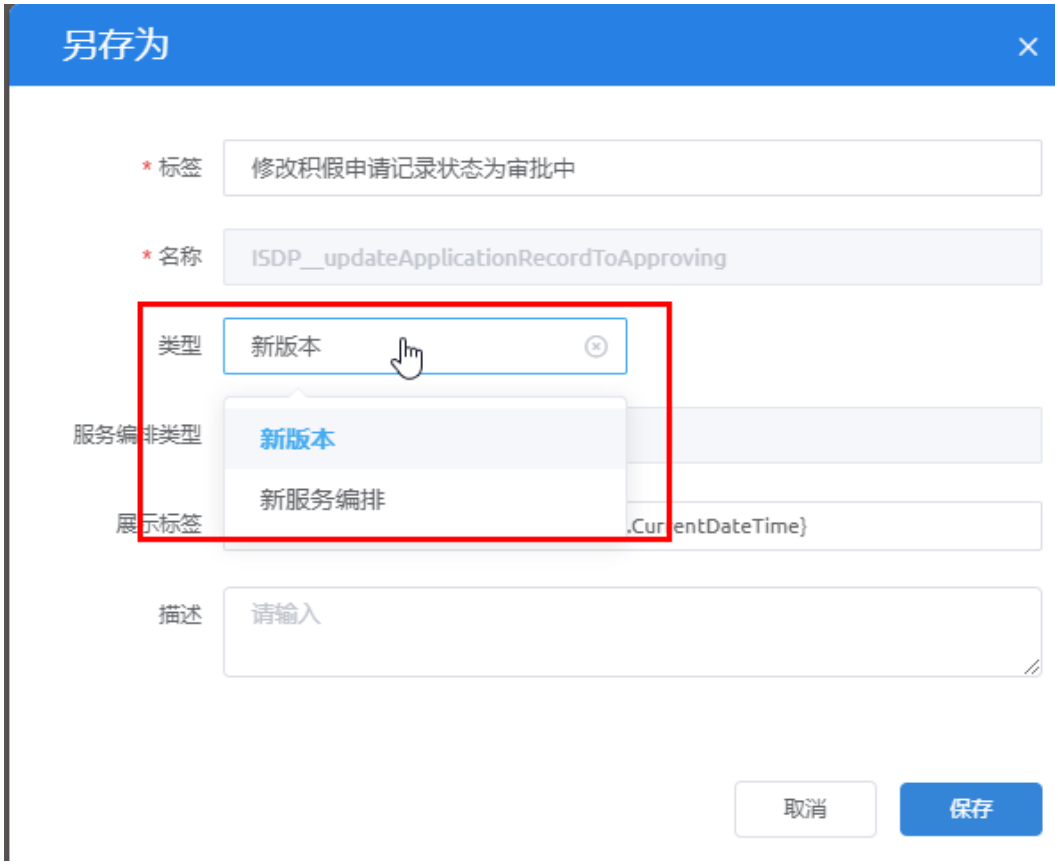
图 1-132 复制服务编排



步骤4 如图1-133所示，在弹出的“另存为”页面，根据规划，配置信息，选择“新版本”或“新服务编排”，单击“保存”。

- 新版本，在当前服务编排编辑区域右上角可以查看到当前服务编排包含的版本，选择新建的版本信息，进行服务编排的定制开发。
- 新服务编排，在左侧可以看到新建的服务编排，单击服务编排右侧展示服务编排编辑页面，进行服务编排的定制开发。

图 1-133 另存为



步骤5 进行服务编排的开发，具体过程请参见**服务编排**。

----结束

1.4.4 打包发布应用

应用开发完成后，需要将应用进行编译打包发布操作，打包后该应用才能发布使用。

步骤1 在应用中（demo），如**图1-134**所示，单击  ，选择“设置”。

图 1-134 属性设置

步骤2 设置软件包，参数配置说明请参见**表1-14**，单击“保存”。

表 1-14 参数说明

参数	配置说明	示例
包类型	应用包的类型： ● 选择“资产包”发布的应用包，包中组件可设置是否受保护。 选择该项后，您需要配置版权信息（可选）、描述（可选）、每个组件的保护设置（必选。配置为未受保护或者只读保护）。 若后续其他用户在开发环境安装后，会显示在开发环境首页的“库”页签下。 ● 选择“源码包”发布的应用包，包中的所有组件不受保护和限制。在其他环境安装后可编辑包中组件，即在原有基础上进行再开发。 若后续其他用户在开发环境安装后，会显示在开发环境首页的“项目”页签下。	资产包
是否全量包	打包时是否打全量包： ● 全量包：表示对整个应用（包括应用中的各个组件）作为一个整体进行设置打包。 ● 增量包：对应用中部分组件进行设置打包。单击“添加应用组件”，在“类别”中选择相应的类别，勾选需要打包的组件。	全量包
产权设置 > 加密保护	当选择“资产包”打包时，才会显示该参数。表示是否对打包的组件中敏感内容（例如脚本内容）是否进行加密。 勾选表示加密。在其他环境安装前，包中敏感数据是经过加密的。	否
产权设置 > 版权信息	当选择“资产包”打包时，才会显示该参数。表示该包的版权信息。 选填项。	-
产权设置 > 描述信息	当选择“资产包”打包时，才会显示该参数。表示该包的描述信息。 选填项，建议描述该App提供的功能。	项目列表
产权设置 > 联系邮件	您可以在此留下当前软件包的问题联系邮箱，若安装过程中出现问题会将联系邮件提示给使用者。	-
产权设置 > 联系链接	您可以在此留下当前软件包的文档链接，若安装过程中出现问题会将联系链接提示给使用者。	-

参数	配置说明	示例
产权设置 > 资产保护 > 保护模式	当选择“资产包”打包时，才会显示该参数。打包数据的保护模式。 ● 未受保护 ● 只读保护 ● 不可见保护 未受保护的资产包在开发环境中安装后，可进行二次编辑；在运行环境安装资产包后，不论保护模式是“未受保护”还是“只读保护”，都不可编辑。	只读保护
部署策略 > 安装前置脚本	当选择“资产包”打包且打全量包时，该配置页才会显示。表示在安装应用包时，在导入实例化配置数据之前执行的脚本。一般用于预清理数据，避免数据冲突的情况。 您可以选择已有脚本，也可以单击“创建”新建脚本。	-
部署策略 > 安装后置脚本	当选择“资产包”打包且打全量包时，该配置页才会显示。表示在安装应用包时，在导入实例化配置数据之后执行的脚本。一般用于删除、更新数据等。 您可以选择已有脚本，也可以单击“创建”新建脚本。	-
部署策略 > 安装时组件的更新策略	当打包的组件中包含系统参数、连接器、Rest操作、数据接入或事件流时，才会显示该参数。例如可设置系统参数随包打包发布后，在升级时遇到新旧数据冲突（唯一索引相同的数据）的数据更新策略。 ● 覆盖：当相关组件数据在打包升级到其他环境，发生数据冲突时，会进行覆盖。 ● 不覆盖：当相关组件数据在打包升级到其他环境，发生数据冲突时，不会进行覆盖。 默认“不覆盖”。在配置为“不覆盖”的情况下，例如在开发环境修改数据接入的任意配置数据（包括所有图元的配置信息），打包升级到测试或者运行环境，不会覆盖同名的数据接入配置，即在开发环境修改的数据在测试或者运行环境不会生效。	覆盖
预置数据	当选择“资产包”打包时，该配置页才会显示。 您可以在该页面选择您在应用打包时一起发布的数据。支持按照对象名称打包。单击“添加对象”可设置数据导出条件，选择对象后，在应用打包时，会将该对象的满足条件的数据都打包出来。打包后，在资产包中“refdata”文件夹下可查看到导出的数据文件。 使用该方式前，您需要先清理不需要发布的数据，且导出对象的“基本信息”页必须勾选上“允许API批量访问”。	-

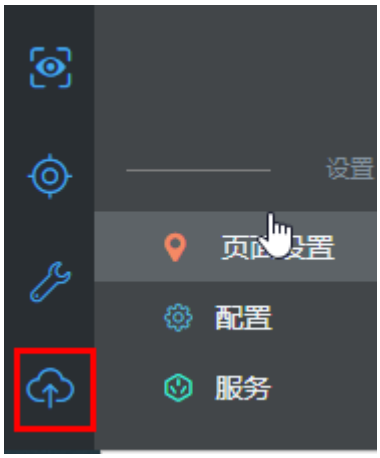
步骤3 如图1-135所示，单击  ，选择“编译”，编译完成如图1-136所示。

图 1-135 编译

图 1-136 编译完成

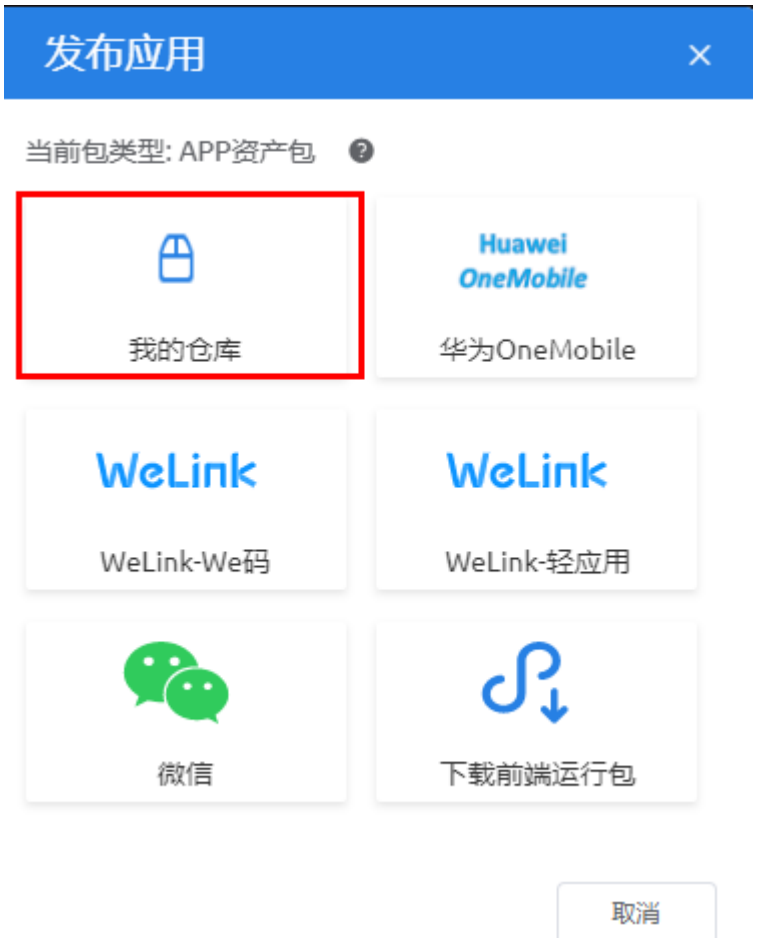
步骤4 如图1-137所示，单击  。

图 1-137 发布



步骤5 如图1-138所示，在弹出的页面中选择“我的仓库”。

图 1-138 发布应用



步骤6 如图1-139所示，填写版本信息，单击“发布”。

图 1-139 发布到我的仓库

说明

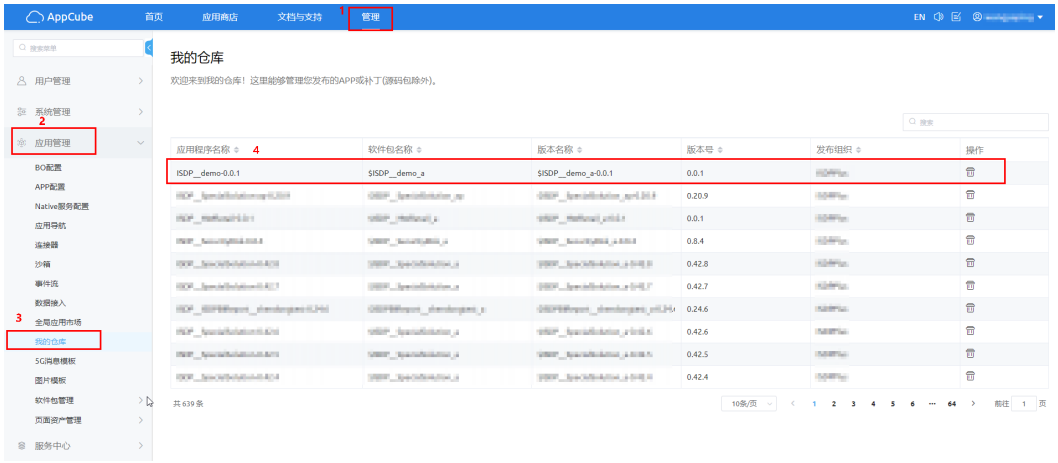
如果勾选“压缩高级页面”，表示会对包中所有高级页面涉及的css和js文件进行合并及压缩，这样可以有效降低运行时服务器压力，但从终端浏览器首次访问该站点页面时，访问时间会稍微增加。

发布成功后，页面显示“程序包已经被成功上传到我的仓库。”

如图1-140所示，可以查看并下载发布的包，也可以在“管理 > 应用管理 > 我的仓库”中查看到发布的包，如图1-141所示。

图 1-140 发布的包

图 1-141 我的仓库



----结束

1.5 集成应用到 ISDP+平台（可选）

介绍如何挂载应用的页面到ISDP+，后续可以直接通过挂载的页面菜单访问应用页面。

背景信息

挂载页面，页面跳转认证支持SSO登录页面跳转认证和AppCube平台的统一身份认证，在配置认证方式时，请根据以下原则配置。

- AppCube版本为1.3.12及以上版本，且ISDP版本为XXX及以后版本，请使用AppCube平台的统一身份认证。

- AppCube或ISDP有一个版本不符合要求（AppCube版本为1.3.12以前版本，或ISDP为XXX以前版本），请使用SSO登录页面跳转认证。

前提条件

已经完成公共BO的部署和配置，具体请参见**安装和配置公共应用/BO**。

获取页面 URL

在挂载菜单之前，需要按照规划获取到要挂载菜单的URL。

高级页面

步骤1 登录AppCube开发环境。

步骤2 单击“管理”，选择“应用管理 > 应用导航”，进入应用程序列表页面。

步骤3 查找到应用。

步骤4 单击应用“操作”列的图标  ，进入高级页面版本选择。

步骤5 在“高级页面版本选择”页面，单击“查看页面地址”，可以获取到页面地址。

步骤6 在URL前面拼接ISDP+ 单点登录SSO页面地址（oauthLogin）/AppCube平台的统一身份认证，即为挂载的页面URL。

- **单点登录SSO页面地址**

  https://***AppCube域名***/magno/render/***应用名_AppCube租户ID***/oauthLogin?redirect_uri=***获取到的页面地址***

- **AppCube平台的统一身份认证**

  https://***AppCube域名***/baas/auth/v1.0/idp?client-name=***AppCube租户ID-自定义统一身份认证名称***&redirect=***获取到的页面地址***

----结束

标准页面

步骤1 登录AppCube开发环境。

步骤2 单击“管理”，选择“应用管理 > 应用导航”，进入应用程序列表页面。

步骤3 查找到应用。

步骤4 单击应用“操作”列的图标  ，进入导航条页面。

步骤5 如**图1-142**所示，单击“菜单树”右侧的“+”，选择“添加页签”。

图 1-142 添加页签

步骤6 页签类型选择“标准页面页签”，配置标签和名称，在页面中选择要添加的标准页面，单击“保存”。

图 1-143 设置页签信息（示例）

步骤7 返回应用列表页面，单击应用“操作”列的图标 。

步骤8 如图1-144所示，单击配置的页面，例如“行业配置”，在导航栏中展示的为配置页面的URL。

图 1-144 查看页面 URL

即页面URL为：https://***AppCube域名***/app/portal.html#/a/000f000000xhqhPD5HQe/Tab/001g000000z1P9PD7BEu。

----结束

挂载菜单

步骤1 登录ISDP+平台生产环境。

步骤2 如图1-145所示，鼠标放置到“公共平台”，选择“系统配置 > 系统设置”。

图 1-145 进入系统设置

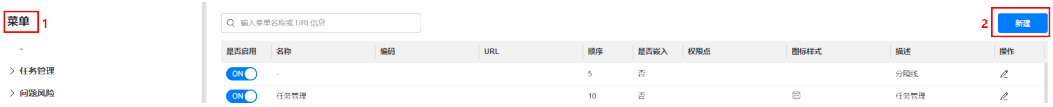
步骤3 如图1-146所示，单击“菜单”页签。

图 1-146 菜单管理

步骤4 配置挂载的菜单和页面。

1. 如图1-147所示，单击“菜单”，再单击“新建”。

图 1-147 新建一级菜单



2. 如图1-148所示，在弹出的“新建”对话框中，输入名称和顺序，单击“保存”。

图 1-148 新建

说明

顺序可以根据实际情况配置，可以查看当前已有一级菜单的顺序，确认创建的菜单需要放在哪个位置，配置需要放置位置前后2个菜单顺序的中间值，如要放置在“任务管理”和“问题风险”之间，那配置的顺序值就配置在10到20之间，如11。

3. 配置完成后，可以看到配置的一级菜单，如图1-149所示，单击新创建的一级菜单，单击“新建”。

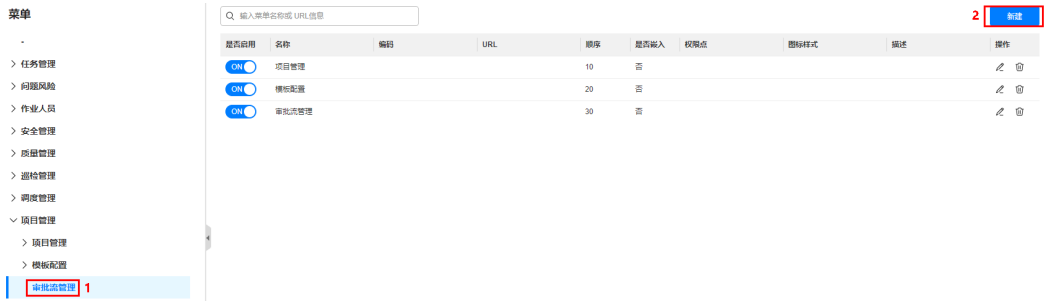
图 1-149 新建二级菜单

4. 如图1-150所示，在弹出的“新建”对话框中，输入名称和顺序，单击“保存”。

图 1-150 新建

5. 配置完成后，可以看到配置的二级菜单，如图1-151所示，单击新创建的二级菜单，单击“新建”。

图 1-151 配置页面链接

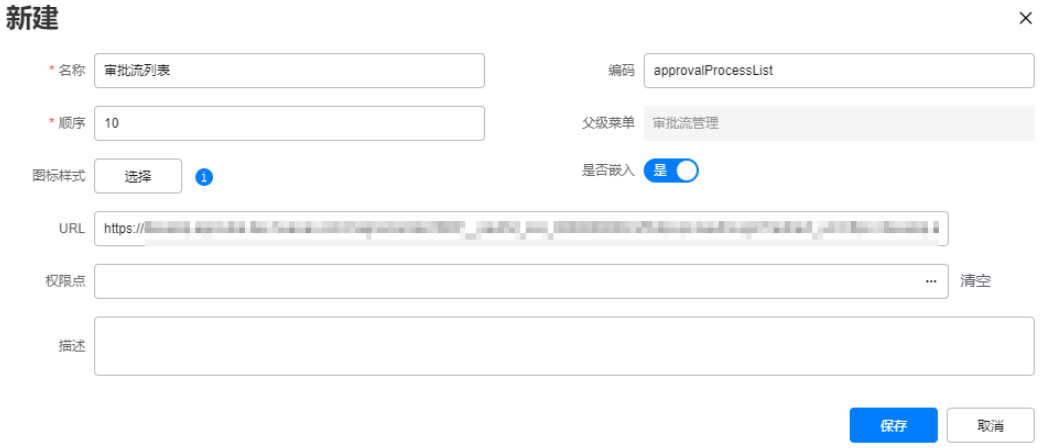


6. 如图1-152所示，在弹出的“新建”对话框中，输入名称、顺序、URL，配置为嵌入，单击“保存”。

说明

页面的链接地址获取方法请参见**前提条件**。

图 1-152 新建



配置后，页面默认为“启用”状态，如图1-153所示。

图 1-153 挂载页面

7. 参见菜单和页面挂载的步骤，完成所有需要挂载页面的配置。

----结束

1.6 安装和配置公共应用/BO

1.6.1 安装公共应用/BO

1.6.1.1 介绍

功能说明

公共应用/BO的功能说明请参见表1-15。

表 1-15 功能说明

应用/BO	包名	功能描述	依赖关系
操作日志BO	ISDP__OperateLog_BO_b-XX.XX.XX.zip	提供了公共模块日志功能，其他应用在记录操作日志时，调用了公共模块日志，完成记录操作日志的功能。	系统参数初始化应用
ISDP+ SSO单点登录应用	ISDP__oauth2_sso-XX.XX.XX.zip	ISDP+认证管理集成ISDP+ OAuth2.0认证服务，用于拉通AppCube应用和ISDP+的单点登录认证。	系统参数初始化应用

应用/BO	包名	功能描述	依赖关系
ISDP+集成公共BO	ISDP__privilege_adapter_BO_b-XX.XX.XX.zip	与ISDP+权限适配层对接，同步ISDP+的用户、角色、用户和角色关系、公共组到AppCube租户下。	操作日志BO、系统参数初始化应用
系统参数初始化应用	ISDP__sysParamsInitialization-XX.XX.XX.zip	提供了预置租户级系统参数，修改内置系统参数，以及导入系统参数和业务权限凭证功能；授予公共应用/BO业务权限凭证给System Administrator Profile、Anonymous User Profile、Standard User Profile和Portal User Profile角色。	-

安装说明

安装公共应用/BO包时，应用/BO有安装顺序要求，请按照顺序进行安装：操作日志BO > ISDP+ SSO单点登录应用 > ISDP+集成公共BO > 系统参数初始化应用。

公共应用/BO安装完成后，会将公共应用/BO的业务权限凭证赋予System Administrator Profile、Anonymous User Profile、Standard User Profile和Portal User Profile，预置租户级的系统参数，修改内置系统参数的值。

- 公共应用/BO的业务权限凭证请参见**表1-16**。

  表 1-16 公共应用/BO 的业务权限凭证

应用/BO	业务权限凭证
操作日志BO	queryLog（查看操作日志） addLog（写入操作日志）
ISDP+ SSO单点登录应用	SSO_Login（PC端登录权限） Mobile_SSO（移动端登录权限）
ISDP+集成公共BO	privilege_adapter（用户角色同步）
系统参数初始化应用	installInitialization（部署初始化）

- 预置租户级的系统参数请参见**表1-17**。

  表 1-17 预置租户级的系统参数

参数名称	描述	值（预置的默认值）
comm_isdp_page_domain	【租户级参数】ISDP+集成页面的域名	-

参数名称	描述	值（预置的默认值）
comm_isdp_openAPI_domain	【租户级参数】ISDP+调用OpenAPI的域名	-
comm_isdp_openAPI_tokenUrl	【租户级参数】ISDP+的订阅OpenAPI的tokenURL	/oauth2/oauth/rest_token
comm_isdp_openAPI_clientId	【租户级参数】ISDP+的订阅OpenAPI的clientId	-
comm_isdp_openAPI_clientSecret	【租户级参数】ISDP+的订阅OpenAPI的ClientSecret	-
comm_isdp_tenantId	【租户级参数】ISDP的租户ID	-
comm_appcube_page_domain	【租户级参数】AppCube集成页面的域名	-
comm_appcube_openAPI_domain	【租户级参数】AppCube调用API的域名	-
comm_appcube_openAPI_tokenUrl	【租户级参数】调用AppCube的OpenAPI的token地址	/baas/auth/v1.0/oauth2/token
comm_appcube_openAPI_clientId	【租户级参数】调用AppCube的OpenAPI的clientId	-
comm_appcube_openAPI_clientSecret	【租户级参数】调用AppCube的OpenAPI的ClientSecret	-
comm_appcube_tenantId	【租户级参数】AppCube的租户ID	获取到的当前AppCube租户的ID。
comm_unifiedIdentityAuthorize	【租户级参数】是否使用appcube平台的统一身份认证，0：不使用，1：使用	1
comm_authorize_client_name	【租户级参数】统一身份认证的名称	ISDP

- 修改的内置系统参数请参见**表1-18**。

表 1-18 修改的内置系统

参数名称	描述	值（修改后的值）
bingo.guest.api.route.whitelist	开启Guest用户URL权限限制	否

参数名称	描述	值（修改后的值）
bingo.service.get.csrf.validation	开启针对GET方法的自定义接口的CSRF校验	否
bingo.service.csrf.validation	开启针对非GET方法的自定义接口的CSRF校验	否
bingo.permission.resource.default.switch	访问资源权限默认开关，包括访问流，脚本，BP，连接器，事件	否
bingo.tenant.security.mode	租户安全模式开关	否
bingo.platform.csrf.validation	开启平台的CSRF校验	否

1.6.1.2 安装公共应用/BO

操作步骤

步骤1 登录AppCube生产环境。

步骤2 如图1-154所示，单击“管理”，选择“应用管理 > 软件包管理 > 软件包安装”。

图 1-154 软件包安装列表

步骤3 单击“新建”，如图1-155所示，在“软件包安装”页面将资产包拖入进去或单击上传资产包（ISDP__OperateLog_BO_b-XX.XX.XX.zip），不勾选“检查软件包中对象属性变更情况”，单击“安装”。

图 1-155 新建

步骤4 参见**步骤3**，继续按照顺序完成资产包ISDP__oauth2_sso-XX.XX.XX.zip、ISDP__privilege_adapter_BO_b-XX.XX.XX.zip和ISDP__sysParamsInitialization-XX.XX.XX.zip的安装，安装完成后，如**图1-156**所示。

图 1-156 安装结果

----结束

1.6.1.3 安装后检查

查看预置的角色 defaultRole

步骤1 登录AppCube生产环境。

步骤2 如**图1-157**所示，单击“管理”，选择“用户管理 > 角色”，在角色中可以查看到角色defaultRole。

图 1-157 角色列表

----结束

查看预置的租户级系统参数和修改的内置系统参数

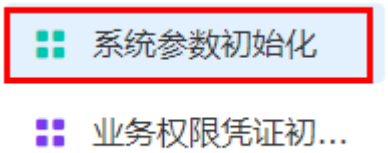
步骤1 登录AppCube生产环境。

步骤2 如**图1-158**所示，单击“管理”，选择“应用管理 > 应用导航”，查找到应用“ISDP__sysParamsInitialization”后，单击应用“操作”列的图标，进入应用。

图 1-158 查看应用

步骤3 如图1-159所示，单击“系统参数初始化”页签，进入“系统参数初始化”页面。

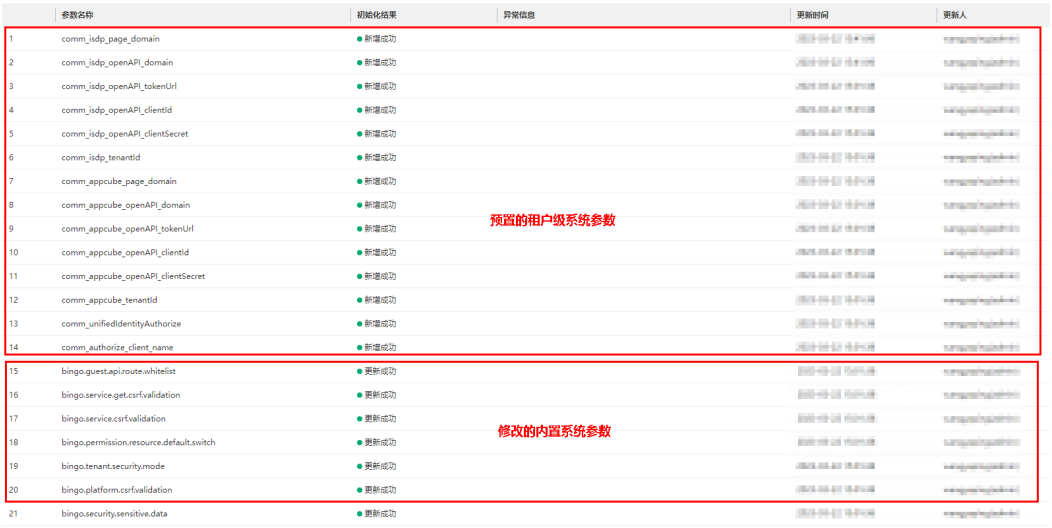
图 1-159 进入系统参数初始化页面



步骤4 在“系统参数初始化”页面，查看预置的租户级系统参数和修改的内置系统参数结果，如图1-160所示。

参数具体说明请参见介绍中表1-17和表1-18。

图 1-160 预置的租户级系统参数和修改的内置系统参数结果

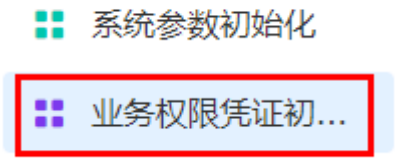


----结束

查看初始化的业务权限凭证

步骤1 在应用“ISDP__sysParamsInitialization”的运行态页面，如图1-161所示，单击“业务权限凭证初始化”页签，进入“业务权限凭证初始化”页面。

图 1-161 进入业务权限凭证初始化页面



步骤2 在“业务权限凭证初始化”页面，查看System Administrator Profile、Anonymous User Profile、Standard User Profile和Portal User Profile业务权限凭证初始化结果，如图1-162所示。

图 1-162 业务权限凭证初始化结果

	权限配置	业务权限凭证	初始化结果	异常信息	更新时间	更新人
1	Portal User Profile	SSO_Login;Mobile_SSO;addLog;queryLog;privilege_adapte...	更新成功			
2	Standard User Profile	SSO_Login;Mobile_SSO;addLog;queryLog;privilege_adapte...	更新成功			
3	Anonymous User Profile	SSO_Login;Mobile_SSO;addLog;queryLog;privilege_adapte...	更新成功			
4	System Administrator Profile	SSO_Login;Mobile_SSO;addLog;queryLog;privilege_adapte...	更新成功			

----结束

1.6.2 配置公共应用/BO

1.6.2.1 分配单点登录认证凭证

背景信息

使用单点登录功能，需要联系对接的ISDP+系统管理员分配单点登录认证凭证。

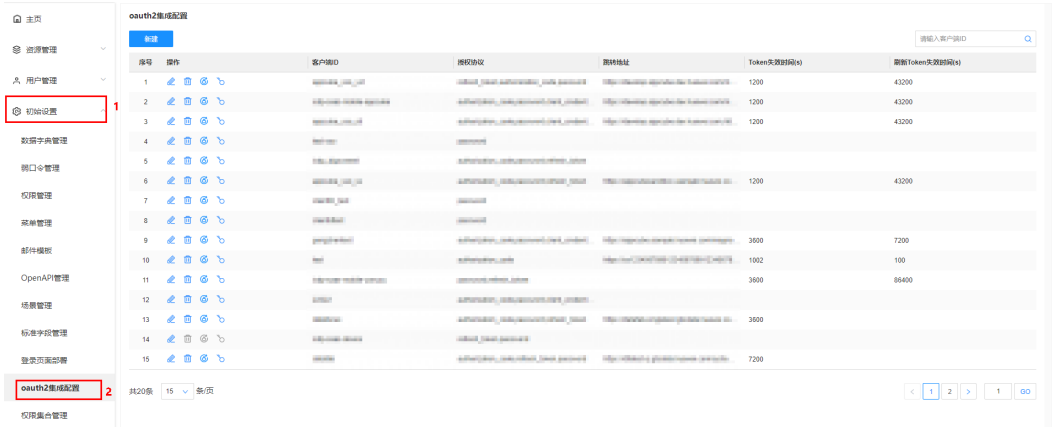
PC端和移动端共用一个单点登录认证凭证，由于移动端的凭证客户端ID必须为isdp-saas-mobile-appcube，所以固定分配单点登录认证凭证的客户端ID为isdp-saas-mobile-appcube。

操作步骤

步骤1 以超级管理员账号登录ISDP+管理员系统（console）（登录地址：https://***ISDP+域名***/console#/sysAdmin/home）。

步骤2 选择“初始设置 > oauth2集成配置”，进入“oauth2集成配置”页面，如图1-163所示。

图 1-163 进入 oauth2 集成配置



步骤3 单击“新建”，如图1-164所示，配置oauth2集成信息，配置参数说明请参见表1-19。

图 1-164 新建配置

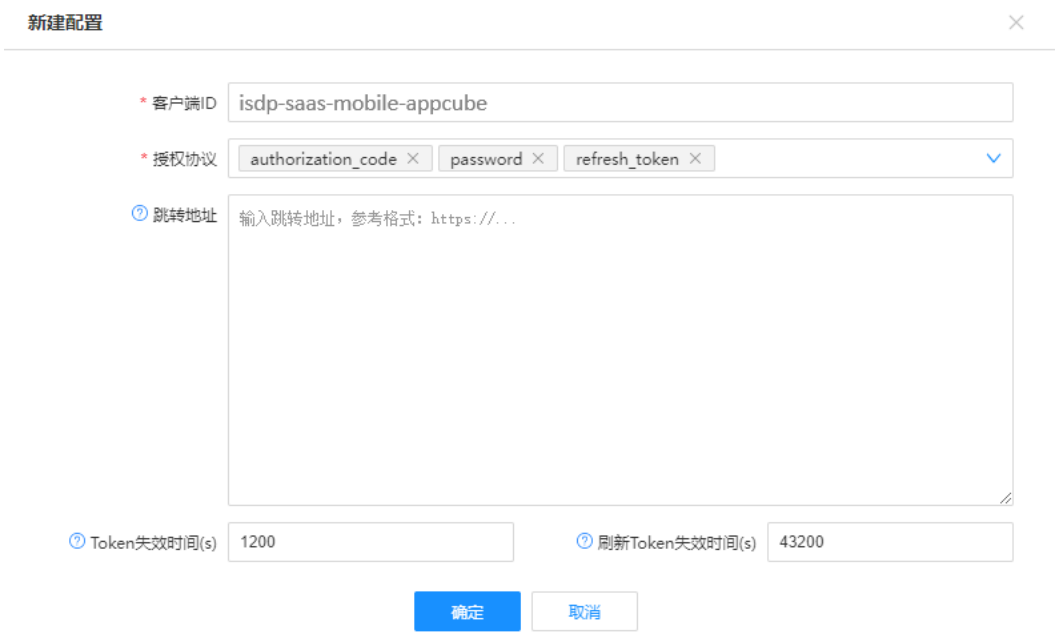


表 1-19 配置参数说明

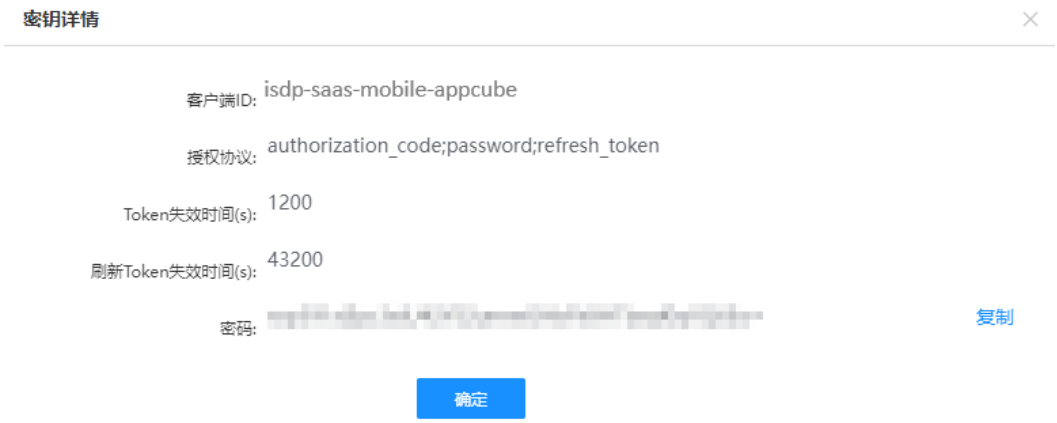
参数	配置说明	示例
客户端ID	手动输入客户端ID。	isdp-saas-mobile-appcube
授权协议	下拉选择授权协议。	authorization_code,password,refresh_token
跳转地址	手动输入URL格式跳转地址，如有多个地址，请用英文逗号分隔。 **只需要配置域名即可。**	https://***AppCube域名***

参数	配置说明	示例
Token失效时间(s)	手动输入整数，默认Token失效时间12小时。	1200
刷新Token失效时间(s)	手动输入整数，默认刷新Token失效时间30天。	43200

步骤4 单击“确定”，新建成功后会弹出密钥详情页面，如图1-165所示。

单击“复制”，可复制密码。

图 1-165 密钥详情



客户端ID和密码，即为分配的单点登录认证凭证（client_id，client_secret），记录下客户端ID和密码。

----结束

1.6.2.2 创建 AppCube 上的接入认证

介绍如何在AppCube中，通过配置OAuth管理第三方接入鉴权。

背景信息

AppCube采用OAuth 2.0协议进行接入认证，在调用AppCube业务接口前，需要在AppCube上进行鉴权注册，获取接入客户端ID、密钥等鉴权信息，才能实现调用AppCube业务接口。

创建 OAuth 认证流程

1. 创建机机用户，具体操作请参见步骤2。
2. 创建OAuth认证，具体操作请参见步骤3。
3. 获取OAuth认证客户端鉴权ID和客户端秘钥，具体操作请参见步骤4。

注意

- 如果AppCube版本比较老，不支持机机用户，请直接使用权限为“System Administrator Profile”的普通用户创建OAuth认证。
- 在进行操作时，AppCube版本不同，页面展示可能有所差别。

操作步骤

步骤1 登录AppCube生产环境。

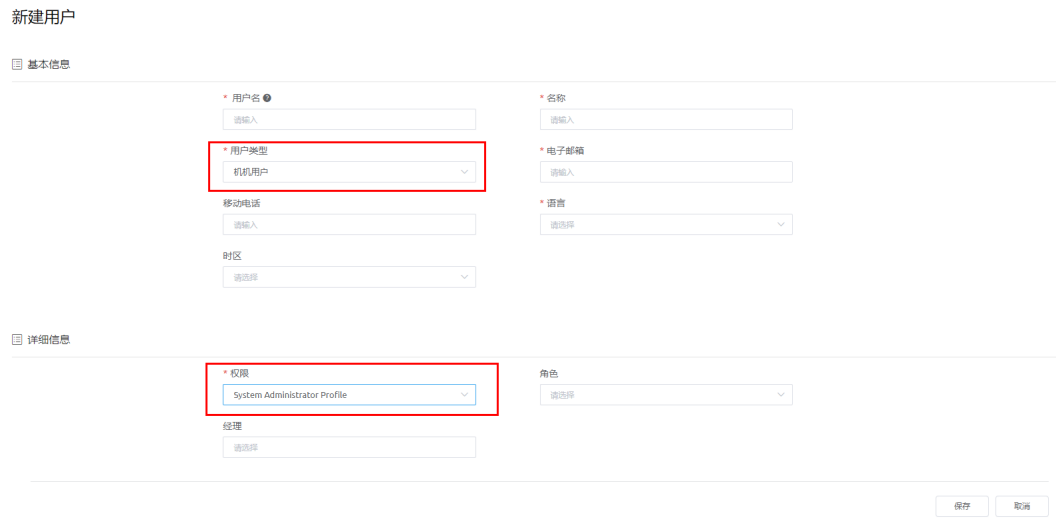
步骤2 创建机机用户。

1. 如图1-166所示，单击“管理”，选择“用户管理 > 用户”，进入“用户列表”页面。

图 1-166 进入用户列表页面

2. 在“用户列表”页面，单击“新建”，在“新建用户”页面，如图1-167所示，选择“用户类型”为“机机用户”，“权限”为“System Administrator Profile”，其他参数根据实际情况配置。

图 1-167 新建用户



3. 单击“保存”，展示新建用户的详情页面，单击页面右上角的 < 用户列表 ，返回到“用户列表”页面，可以查看到新创建的机机用户。

步骤3 创建OAuth认证。

1. 如图1-168所示，选择“系统管理 > OAuth”，进入“OAuth管理”页面。

图 1-168 进入 OAuth 管理页面

2. 单击“新建”，输入名称（自定义），“授权类型”选“客户端模式”，“用户”选择步骤2中创建的机机用户，如图1-169所示。

图 1-169 认证密钥

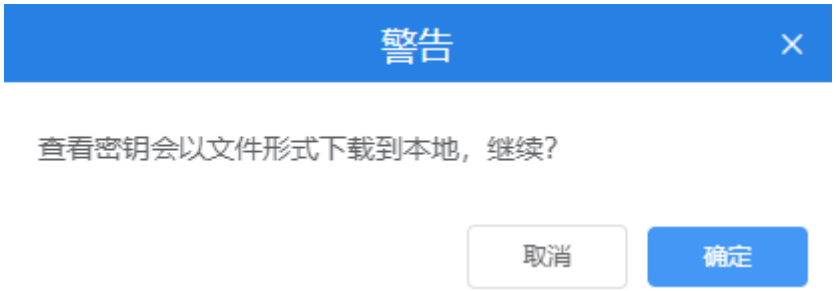
3. 单击“保存”，在“OAuth管理”页面展示新建的记录，如图1-170所示。

图 1-170 OAuth 管理列表

步骤4 获取OAuth认证的客户端鉴权ID（client id）和客户端秘钥（client_secret）

1. 单击新建记录“操作”列的，弹出“警告”对话框，如图1-171所示。

图 1-171 警告



2. 单击“确定”，下载密钥文件到本地，如图1-172所示，从中获取客户端鉴权ID（client id）和客户端秘钥（client_secret）。

图 1-172 密钥

----结束

1.6.2.3 配置公共应用/BO 以及租户级系统参数

介绍如何配置公共应用/BO以及租户级的系统参数。

背景信息

- 租户级系统参数只需要手动配置四个参数：comm_isdp_openAPI_domain、comm_isdp_page_domain、comm_appcube_openAPI_domain、comm_appcube_page_domain。
- 公共应用/BO中，只有ISDP+ SSO单点登录应用需要配置系统参数。

配置流程

1. 获取导入模板，具体操作请参见步骤2。
2. 在模板中配置系统参数以及值，具体操作请参见步骤3。
3. 导入模板，具体操作请参见步骤4。

操作步骤

步骤1 登录AppCube生产环境。

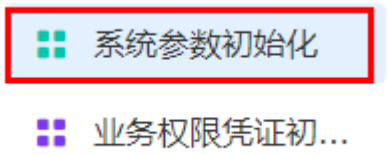
步骤2 获取到系统参数初始化导入模板。

1. 如图1-173所示，单击“管理”，选择“应用管理 > 应用导航”，查找到应用“ISDP__sysParamsInitialization”后，单击应用“操作”列的图标，进入应用。

图 1-173 查看应用

2. 如图1-174所示，单击“系统参数初始化”页签，进入“系统参数初始化”页面。

图 1-174 进入系统参数初始化页面



3. 在“系统参数初始化”页面，如图1-175所示，单击“导入”，在弹出的“导入”页面。

图 1-175 进入导入页面

4. 单击“下载模板”，下载模板到本地。

步骤3 请参见表1-20，在下载的模板中配置要导入的系统参数。

说明

域名配置说明：

- 调用OpenAPI的域名，请配置为内网域名；调用页面的域名，请配置为外网域名。如果不区分内外网，即内外网域名/IP一致。
- 域名格式：https://域名，如果是IP地址，格式：https://IP地址:端口号（如果为IP地址，没有端口号，直接配置为https://IP地址）。
- 域名结尾不能带“/”。

表 1-20 系统参数

系统参数名称	系统参数值	描述
comm_isdp_openAPI_domain	***请根据实际情况配置***	【租户级参数】ISDP+调用OpenAPI的域名
comm_isdp_page_domain		【租户级参数】ISDP+集成页面的域名
comm_appcube_openAPI_domain		【租户级参数】AppCube调用API的域名
comm_appcube_page_domain		【租户级参数】AppCube集成页面的域名
ISDP__client_id_sso		《SSO单点登录》isdp+创建的oauth2的客户端ID和客户端密码，即**分配单点登录认证凭证**中分配的认证凭证。
ISDP__client_secret_sso		
clientID_sso		
clientSecret_sso		

步骤4 导入模板。

1. 在“导入”页面，单击上传或拖动**步骤3**中修改保存的文件到该处。
2. 单击“确定”，展示导入结果，如**图1-176**所示。

图 1-176 导入结果

	参数名称	初始化结果	异常信息	更新时间	更新人
1	comm_isdp_openAPI_domain	更新成功			
2	comm_isdp_page_domain	更新成功			
3	comm_appcube_openAPI_domain	更新成功			
4	comm_appcube_page_domain	更新成功			
5	ISDP__client_id_sso	更新成功			
6	ISDP__client_secret_sso	更新成功			
7	clientID_sso	更新成功			
8	clientSecret_sso	更新成功			

----结束

1.6.2.4 配置 ISDP+集成公共 BO

介绍如何配置ISDP+集成公共BO。

1.6.2.4.1 介绍

ISDP+集成公共BO实现如下功能：

- 同步ISDP+的订阅OpenAPI的clientId和clientSecret，ISDP+租户ID，调用AppCube的OpenAPI的clientId和clientSecret的值到对应的租户级系统参数comm_isdp_openAPI_clientId、comm_isdp_openAPI_clientSecret、comm_isdp_tenantId、comm_appcube_openAPI_clientId和comm_appcube_openAPI_clientSecret中。
- 同步ISDP+的用户、角色、用户和角色关系、公共组到AppCube租户下。

1.6.2.4.2 配置租户映射关系

⚠ 注意

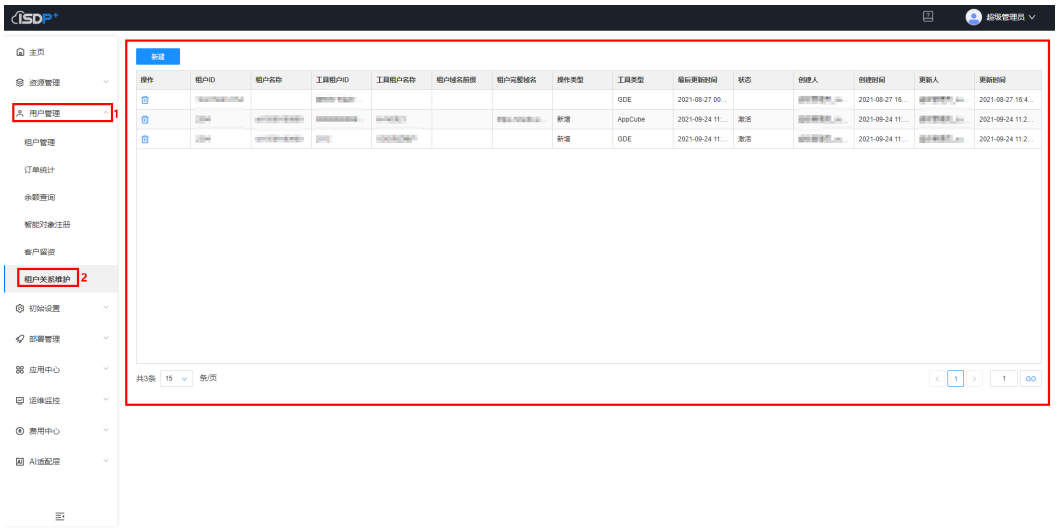
- 租户映射关系只能是一对一的关系，即一个ISDP+租户对应一个AppCube租户，且租户关系配置时需要保证一次配置成功，否则要提交电子流修改。
- 如果已配过ISDP+租户与AppCube租户映射关系，在重新配置之前，需要先删除AppCube上原有的数据，包括业务用户、角色、权限配置、公共组，并删除ISDP+管理员系统（console）上租户映射关系配置，再重新配置租户对接。
- 租户映射关系配置完成后，可以等到每个整半点（例如：10:00:00，10:30:00，11:00:00...）自动执行，也可以手动执行。执行任务时，只拉取执行时间前半小时内的租户映射关系数据。

操作步骤

步骤1 以超级管理员账号登录ISDP+管理员系统（console）（登录地址：https://***ISDP+域名***/console#/sysAdmin/home）。

步骤2 如图1-177所示，选择“用户管理 > 租户关系维护”，进入“租户关系维护”页面。

图 1-177 租户关系维护



步骤3 如图1-178所示，单击“新建”，在弹出的“新增”页面，增加租户关系，具体配置说明请参见表1-21。

图 1-178 新增

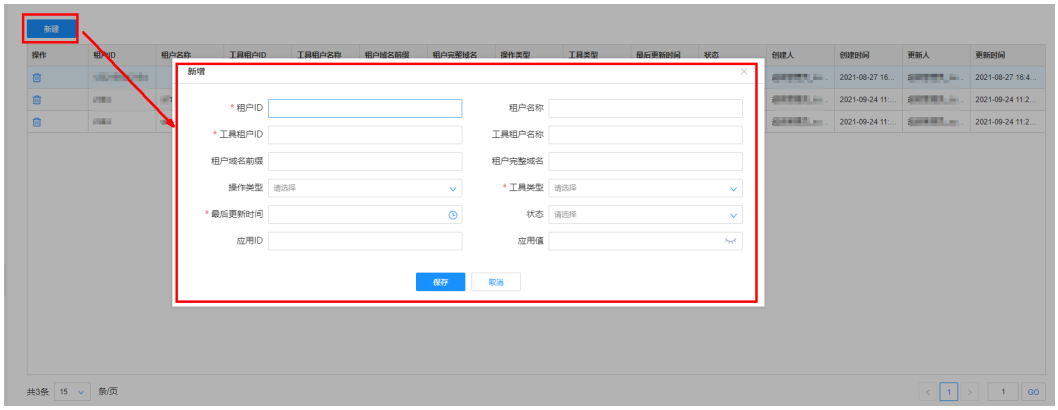


表 1-21 配置说明

参数	配置说明
租户ID 租户名称	获取ISDP+租户ID和租户名称的方法： 选择“用户管理 > 租户管理”，查找到租户，其中，租户ID为“租户编号”列值SaasTenant后面部分。例如，租户编号为“SaasTenant2394”，则租户ID为“2394”。 
工具租户ID	AppCube的租户ID和租户名称。 获取AppCube租户ID和租户名称的方法： 1. 在AppCube环境，单击“管理”，选择“用户管理 > 公司配置 > 公司信息”。 2. 在“详细信息”区域，查看到参数“租户ID”的值，即为AppCube的租户ID；在“基本信息”区域，查看到参数“租户名”的值，即为AppCube的租户名称。

参数	配置说明
工具租户名称	
租户域名前缀	AppCube租户域名前缀，可选配置项。
租户完整域名	AppCube租户完整域名。 **说明** 如果没有域名，配置为IP+端口号，格式：https://IP地址:端口号。
操作类型	选择操作类型，新增或修改。 固定配置为“新增”。
工具类型	选择配置的工具类型。 固定配置为“AppCube”。
最后更新时间	选择最后更新时间，精确到秒。 固定配置为“此刻”。
状态	选择租户关系的状态，激活或未激活。 固定配置为“激活”。
应用ID 应用值	输入AppCube的client_id和client_secret。即**创建AppCube上的接入认证**中创建获取到的客户端鉴权ID（client id）和客户端秘钥（client_secret）。

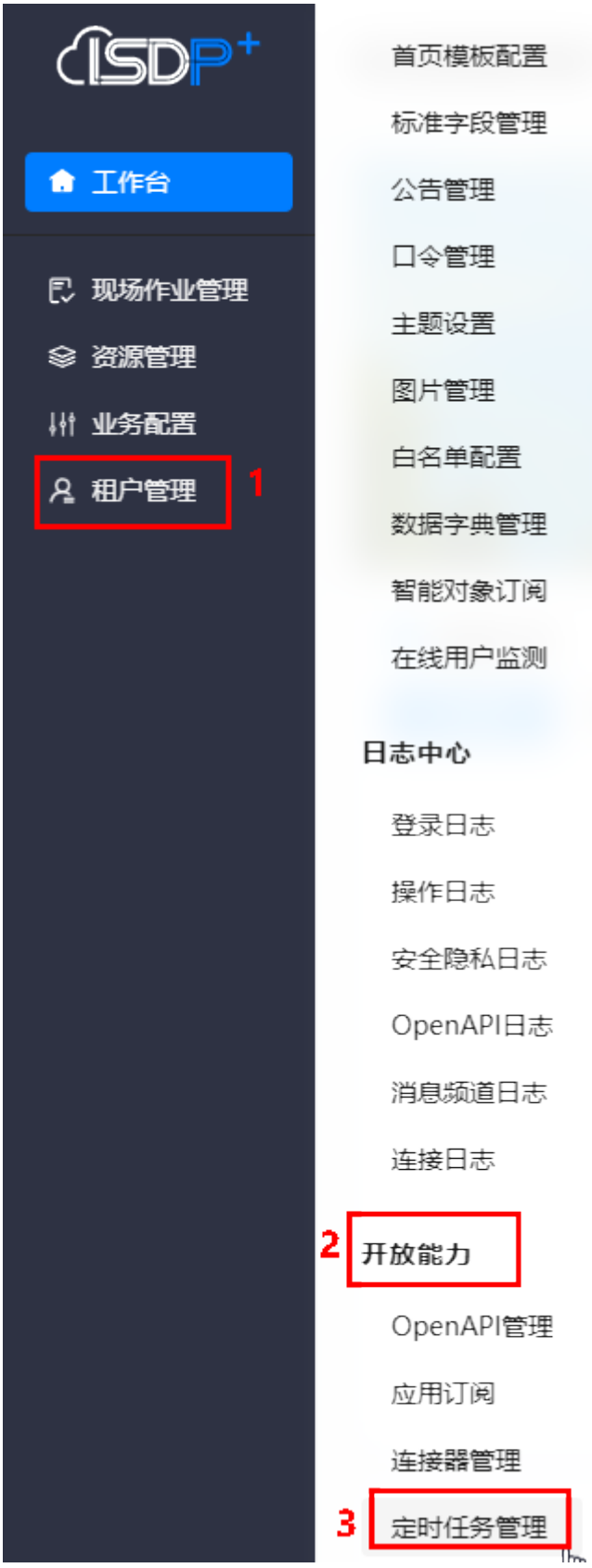
步骤4 单击“保存”，完成租户映射关系配置。

配置完成后，等待整半点自动执行（例如配置的时间为10:12，则在10:30自动执行），或者手动执行。

手动执行任务方式如下：

1. 以超级管理员账号登录ISDP+平台生产环境（登录地址：https://***ISDP+域名***）。
2. 如**图1-179**所示，鼠标放置到“租户管理”，选择“开发能力 > 定时任务管理”。

图 1-179 进入定时任务管理



3. 在“定时任务管理”页面，根据任务名称“权限适配层-租户映射关系扫描割接”，搜索到定时任务，如图1-180所示。

图 1-180 权限适配层-租户映射关系扫描割接定时任务

4. 单击任务操作列的 ，手动执行定时任务。

----结束

1.6.2.4.3 订阅“appcubeAdapter 应用创建”应用

操作步骤

步骤1 登录ISDP+平台生产环境。

步骤2 如图1-181所示，鼠标放置到“公共平台”，选择“系统配置 > 应用设置”。

说明

菜单路径可配置，菜单路径名称可能不同，具体以实际环境为准，图展示为示例。
也可通过菜单URL地址：https://***ISDP+域名***/admin/#/pub/appSubscription，进入页面。

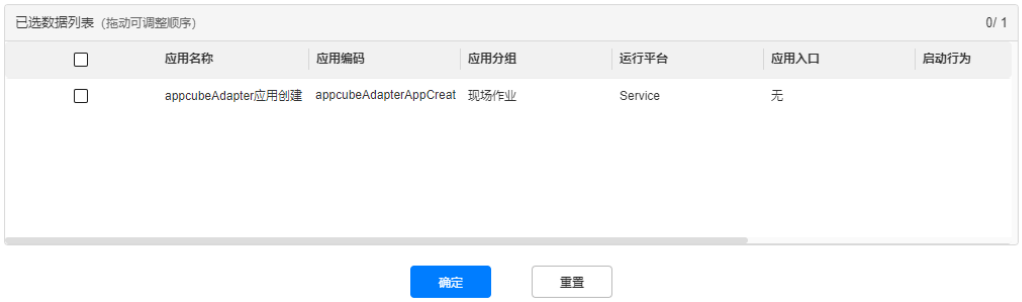
图 1-181 进入应用设置

步骤3 如图1-182所示，在“应用订阅”页签中单击“订阅应用”，在弹出的“订阅应用”页面，搜索应用“**appcubeAdapter应用创建**”，并勾选搜索的“**appcubeAdapter应用创建**”应用，单击 。

在“已选数据列表”区域展示订阅的应用，如图1-183所示。

图 1-182 订阅应用

图 1-183 已选数据列表

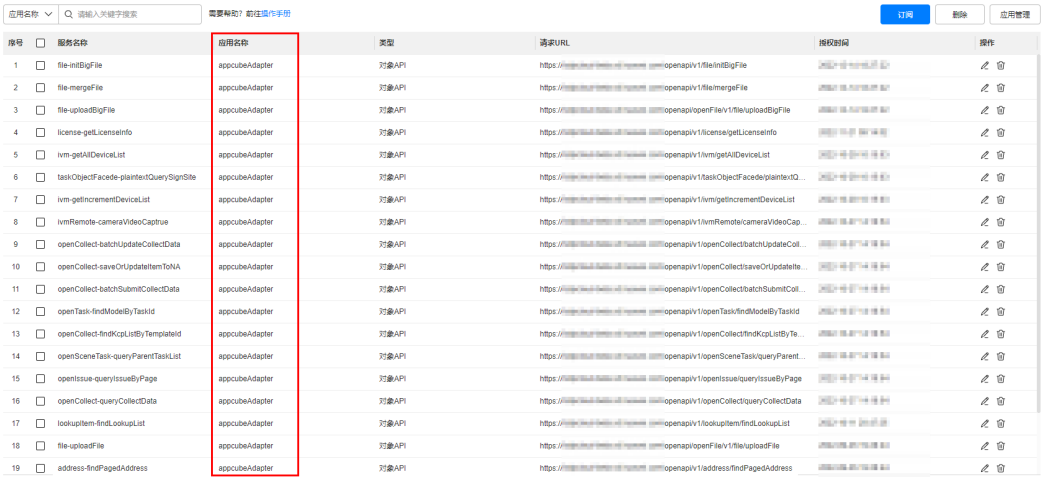


步骤4 单击“确定”，完成应用订阅，如图1-184所示。

图 1-184 订阅应用列表

订阅应用后，在“集成中心 > 集成中心 > 服务集成”中，可以看到**appcubeAdapter**应用自动订阅的对象API，示例如图1-185所示。

图 1-185 订阅的 OpenAPI 示例



----结束

1.6.2.4.4 检查租户级系统参数

介绍如何查看租户级系统参数是否同步成功。

背景信息

配置完成租户映射关系，且订阅“appcubeAdapter应用创建”应用后，会自动同步ISDP+的订阅OpenAPI的clientId和clientSecret，ISDP+租户ID，调用AppCube的OpenAPI的clientId和clientSecret的值到对应的租户级系统参数comm_isdp_openAPI_clientId、comm_isdp_openAPI_clientSecret、comm_isdp_tenantId、comm_appcube_openAPI_clientId和comm_appcube_openAPI_clientSecret中。

操作步骤

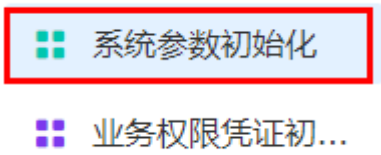
步骤1 登录AppCube生产环境。

步骤2 如图1-186所示，单击“管理”，选择“应用管理 > 应用导航”，查找到应用“ISDP__sysParamsInitialization”后，单击应用“操作”列的图标，进入应用。

图 1-186 查看应用

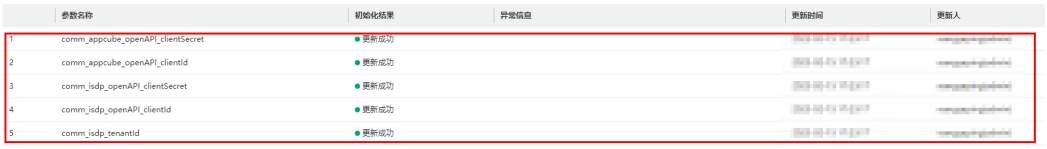
步骤3 如图1-187所示，单击“系统参数初始化”页签，进入“系统参数初始化”页面。

图 1-187 进入系统参数初始化页面



步骤4 在“系统参数初始化”页面，查看同步五个租户级参数的结果，如图1-188所示。

图 1-188 同步五个租户级系统参数结果



----结束

1.6.2.4.5 订阅 OpenAPI

订阅appcubeAdapter应用创建后，会自动订阅所需的“对象API”，使用到的“文件API”，需手动订阅。请使用**订阅“appcubeAdapter应用创建”应用**后自动生成的应用**appcubeAdapter**订阅ISDP+集成公共BO使用到的文件API。

操作步骤

步骤1 登录ISDP+平台生产环境。

步骤2 如图1-189所示，鼠标放置到“集成中心”，选择“集成中心 > 服务集成”。

说明

菜单路径可配置，菜单路径名称可能不同，具体以实际环境为准，图展示为示例。
也可通过菜单URL地址：https://***ISDP+域名***/admin/#/pub/apiuser，进入页面。

图 1-189 进入服务集成

步骤3 单击“订阅”，在弹出的“订阅API”页面，如图1-190所示，选择**appcubeAdapter**应用，单击API类型“文件API”，从左侧查找并勾选要订阅的API到右侧，单击“确定”，需要订阅的API如表1-22所示。

说明

- 使用同一个应用去订阅不同应用/BO使用的OpenAPI时，如果不同应用/BO使用相同的OpenAPI，只需要订阅一次。
- 订阅OpenAPI时，单次最多只能订阅10条；如果超过10条，请分多次订阅。

图 1-190 订阅 API

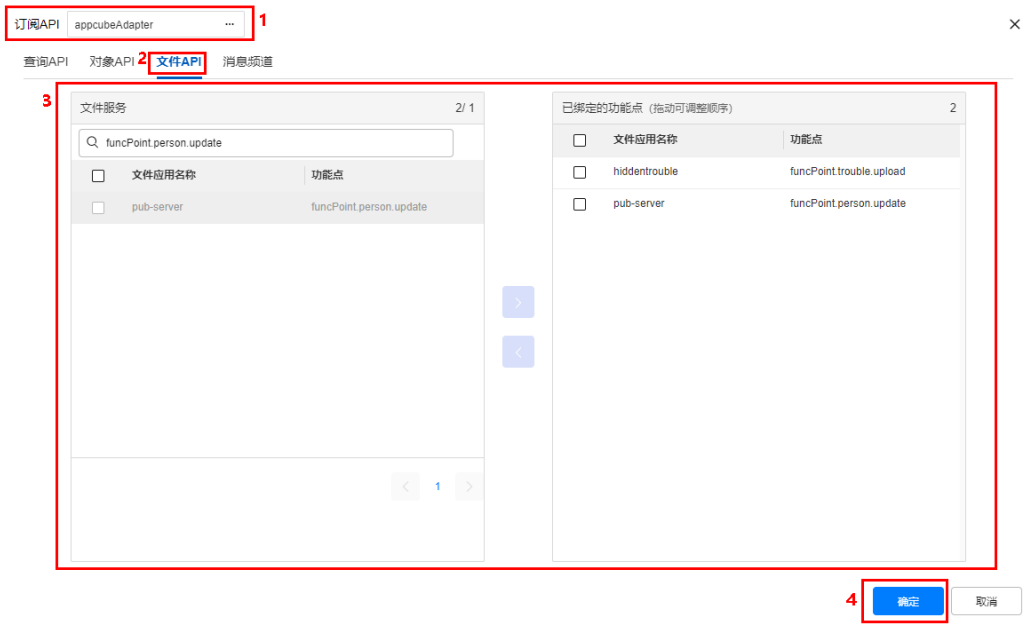


表 1-22 需要订阅的 API

应用/BO	类型	名称	别名
ISDP+集成公共BO	文件API	funcPoint.trouble.upload（功能点）	-
		funcPoint.person.update（功能点）	-

----结束

1.6.2.4.6 配置后检查

检查用户、角色、权限和公共组同步

配置完成后，等到整半点自动执行后或手动执行后，检查AppCube租户下的同步结果。

操作步骤

步骤1 登录AppCube生产环境。

步骤2 如图1-191所示，单击“管理”，选择“用户管理 > 业务用户”，检查ISDP+的用户和权限是否正确同步。

图 1-191 业务用户

步骤3 如图1-192所示，选择“权限配置”，检查ISDP+的角色是否正确同步。

图 1-192 权限配置

步骤4 如图1-193所示，在“权限配置列表”页面，单击同步过来的角色名称，（例如：施工现场作业负责人），进入“权限配置详情”页面。

图 1-193 进入权限配置详情

步骤5 如图1-194所示，在“权限配置详情”页面，单击“业务权限凭证”，查看业务权限凭证是否与Standard User Profile一致（即勾选了SSO_Login、Mobile_SSO、privilege_adapter、queryLog、addLog、installInitialization、deleteFile、queryFile、uploadFile）。

图 1-194 业务权限凭证

步骤6 如图1-195所示，在“权限配置详情”页面，单击“系统参数”，查看系统参数权限是否与Standard User Profile一致。

图 1-195 系统参数

步骤7 如图1-196所示，选择“公共组”，检查ISDP+的公共组是否正确同步。

图 1-196 公共组

----结束

1.6.2.5 配置页面跳转认证方式

本节介绍如何配置页面跳转认证方式。

注意

页面跳转认证支持SSO登录页面跳转认证和AppCube平台的统一身份认证，在配置认证方式时，请根据以下原则配置。

- **AppCube版本为1.3.12及以上版本，且ISDP版本为XXX及以后版本，请使用AppCube平台的统一身份认证，具体配置方法请参见使用AppCube平台的统一身份认证。**
- **AppCube或ISDP有一个版本不符合要求（AppCube版本为1.3.12以前版本，或ISDP为XXX以前版本），请使用SSO登录页面跳转认证，具体配置方法请参见使用SSO登录页面跳转认证。**

使用 AppCube 平台的统一身份认证

通过自定义接口，创建自定义统一身份认证。

步骤1 登录Appcube生产环境。

步骤2 如图1-197所示，单击“管理”，选择“系统管理 > 自定义接口”，进入“自定义接口”页面。

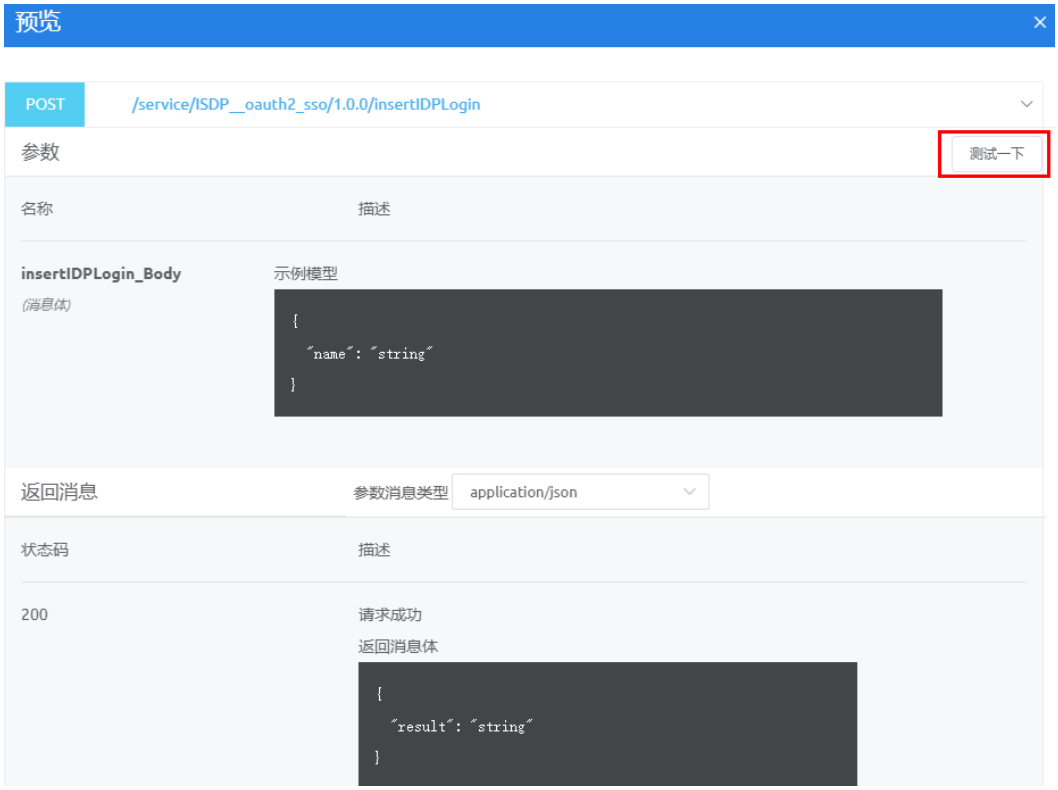
图 1-197 自定义接口

步骤3 如图1-198所示，查找到接口“insertIDPLogin”，单击“操作”列的，进入“预览”页面。

图 1-198 预览

步骤4 如图1-199所示，单击右侧的，展开内容，再单击“测试一下”。

图 1-199 测试一下



步骤5 如图1-200所示，输入入参为ISDP，或者清空入参，单击“执行”，执行接口，执行结果如图1-201所示，表示执行成功。

说明

不输入入参，默认创建名称为**ISDP**的自定义认证。

图 1-200 执行

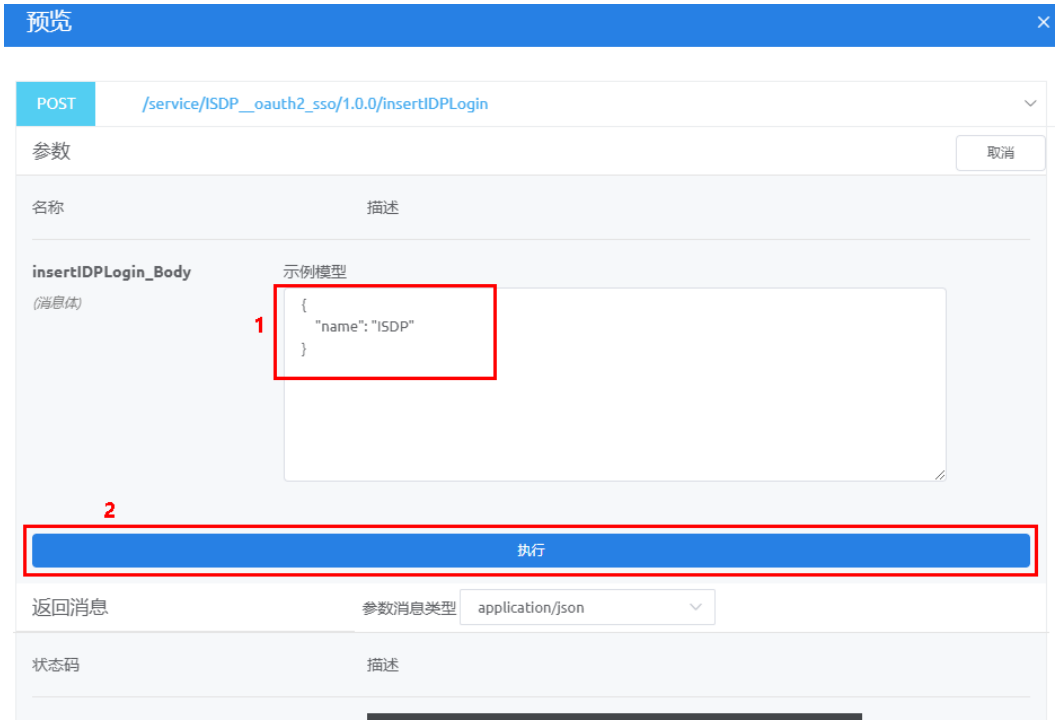


图 1-201 执行结果

创建成功后，在“管理 > 系统管理 > 统一身份认证”的“自定义认证”区域，可以看到新建的自定义认证，如图1-202所示。

图 1-202 统一身份认证

说明

如果创建失败，提示如下，请修改name值，重新创建。创建完成后，修改租户级系统参数“comm_authorize_client_name”的值为配置的name值。

状态码	描述
400	{ "resCode": "405234187", "resMsg": "字段'name'的值'ISDP'已经被占用，不能重复" }

参数消息类型

----结束

使用 SSO 登录页面跳转认证

修改租户级系统参数，配置使用SSO登录页面认证方式。

步骤1 登录AppCube生产环境。

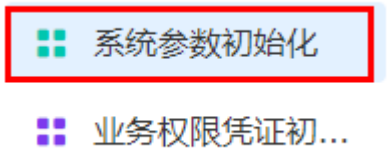
步骤2 获取到系统参数初始化导入模板。

1. 如图1-203所示，单击“管理”，选择“应用管理 > 应用导航”，查找到应用“ISDP__sysParamsInitialization”后，单击应用“操作”列的图标，进入应用。

图 1-203 查看应用

2. 如图1-204所示，单击“系统参数初始化”页签，进入“系统参数初始化”页面。

图 1-204 进入系统参数初始化页面



3. 在“系统参数初始化”页面，如图1-205所示，单击“导入”，在弹出的“导入”页面。

图 1-205 进入导入页面

4. 单击“下载模板”，下载模板到本地。

步骤3 请配置表1-23中的参数到下载的模板中。

表 1-23 系统参数

系统参数名称	系统参数值	描述
comm_unifiedIdentityAuthorize	0	租户级参数】是否使用appcube平台的统一身份认证，0：不使用，1：使用

步骤4 导入模板。

1. 在“导入”页面，单击上传或拖动步骤2中获取到的文件到该处。
2. 单击“确定”，展示导入结果，如图1-206所示。

图 1-206 导入结果

----结束

2 智能排班模型 BO 接口

2.1 使用前必读

2.1.1 概述

智能排班模型BO基于AppCube开发，提供了人员信息管理、离岗信息管理、积假信息管理等排班相关的API。您可以使用智能排班模型BO提供的API，在AppCube上进行应用的开发，从而实现智能排班相关的能力。

2.1.2 调用说明

智能排班模型BO提供了REST（Representational State Transfer，表征状态转移）风格API，支持您通过HTTPS请求调用，调用方法请参见**如何调用API**。

2.1.3 终端节点

终端节点即调用API的请求地址。例如，智能排班模型BO部署的AppCube开发态默认域名为“XXX.huawei.com”，则终端节点域名为“XXX.huawei.com”。

说明

XXX.huawei.com为一个示例说明，不是实际的域名，后续描述中，均使用XXX.huawei.com作为示例说明，在实际环境中使用时，请替换为实际使用的域名。

2.1.4 基本概念

使用智能排班模型BO API涉及的常用概念。

名称	描述
BO	BO（全称Business Object）即商业对象，是封装了完整的数据模型、业务逻辑、页面展现的软件单元，通过开放出来的接口为上层应用提供服务。BO也可以提供管理页面，对自身的数据进行配置管理。 用户可基于某些BO，组合、排列并进行配置，如魔方一样创建功能各异的应用。

名称	描述
access-token	access-token是调用接口API要用到的访问令牌，在调用API的时候将access-token加到请求消息头，从而通过身份认证，获得操作API的权限。

2.2 如何调用 API

2.2.1 构造请求

本节介绍REST API请求的组成，并以使用OAuth 2.0协议的客户端鉴权模式获取用户access_token为例说明如何调用API，该API获取access_token，access_token可以用于调用其他API时鉴权。

请求 URL

请求URL由如下部分组成：

{URL-scheme}://{Endpoint}/{resource-path}?{query-string}

表 2-1 请求 URL

参数	说明
URL-scheme	表示用于传输请求的协议，当前所有API均采用**HTTPS**协议。
Endpoint	指定承载REST服务端点的AppCube服务器域名或IP。
resource-path	资源路径，即API访问路径。从具体API的URL模块获取，例如“获取用户Token”API的resource-path为“/baas/auth/v1.0/oauth2/token”。
query-string	查询参数，是可选部分，并不是每个API都有查询参数。查询参数前面需要带一个“?”，形式为“参数名=参数取值”，例如“limit=10”，表示查询不超过10条数据。

例如您需要使用OAuth 2.0协议的客户端鉴权模式获取用户access_token，“/baas/auth/v1.0/oauth2/token”为资源路径，假设AppCube开发态域名为“XXX.huawei.com”，URL拼接起来如下所示。

```
https://XXX.huawei.com/baas/auth/v1.0/oauth2/token          //XXX.huawei.com为一个示例说明，不是实际的域名，在实际环境中使用时，请替换为实际使用的域名。
```

请求方法

HTTP请求方法（也称为操作或动词），它告诉服务你正在请求什么类型的操作。

表 2-2 HTTP 请求方法

方法	说明
GET	请求服务器返回指定资源。
PUT	请求服务器更新指定资源。
POST	请求服务器新增资源或执行特殊操作。
DELETE	请求服务器删除指定资源。
HEAD	请求服务器资源头部。
PATCH	请求服务器更新资源的部分内容。当资源不存在的时候，PATCH可能会去创建一个新的资源。

在使用OAuth 2.0的客户端鉴权模式获取用户access_token的URL部分，您可以看到其请求方法为“**POST**”，则其请求为：

```
POST https://XXX.huawei.com/baas/auth/v1.0/oauth2/token          //XXX.huawei.com为一个示例说明，不是实际的域名，在实际环境中使用时，请替换为实际使用的域名。
```

请求消息头

附加请求头字段，如指定的URL和HTTP方法所要求的字段。例如定义消息体类型的请求头“**Content-Type**”，请求鉴权信息等。

详细的公共请求消息头字段请参见**表2-3**。

表 2-3 公共请求消息头

消息头名称	描述	是否必选
Content-Type	消息体的类型（格式），HTTP协议中设定的一个参数，用于标识返回的内容用什么格式去解析。 • 配置为“application/json”，表示浏览器将返回内容解析为json对象。 • 配置为“application/x-www-form-urlencoded”，表示urlencode格式。 消息体的类型请参见**API**中API的说明。	是

消息头名称	描述	是否必选
access-token	access-token是调用AppCube接口API要用到的访问令牌，在调用API的时候将access-token加到请求消息头，从而通过身份认证，获得操作API的权限。 当第三方系统要访问AppCube的接口时，需要提前使用OAuth协议进行接入认证，获取客户端鉴权ID（Client ID）和客户端鉴权密钥（Client Secret），再使用客户端鉴权ID和客户端鉴权密钥调用接口“/baas/auth/v1.0/oauth2/token”获取access_token，从而获取AppCube接口的访问令牌。	否 **说明** 使用access-token认证时该字段必选。

对于使用OAuth 2.0的客户端鉴权模式获取用户access_token接口，由于不需要access-token认证，所以只添加“Content-Type”为“application/x-www-form-urlencoded”即可，添加消息头后的请求如下所示。

```
POST https://XXX.huawei.com/baas/auth/v1.0/oauth2/token          //XXX.huawei.com为一个示例说明，不是实际的域名，在实际环境中使用时，请替换为实际使用的域名。
Content-Type: application/x-www-form-urlencoded
```

请求消息体（可选）

请求消息体通常以结构化格式（如JSON或XML）发出，与请求消息头中Content-type对应，传递除请求消息头之外的内容。若请求消息体中参数支持中文，则中文字符必须为UTF-8编码。

每个接口的请求消息体内容不同，也并不是每个接口都需要有请求消息体（或者说消息体为空），GET、DELETE操作类型的接口就不需要消息体，消息体具体内容需要根据具体接口而定。

对于使用OAuth 2.0的客户端鉴权模式获取用户access_token接口，您可以从**表2-4**看到所需的请求参数及参数说明。

表 2-4 请求参数说明

参数名称	类型	必选（M）/可选（O）	参数位置	参数含义
grant_type	String	M	Body	授权模式，OAuth2.0中的grant_type字段的取值。 例如：client_credentials（即客户端模式）

参数名称	类型	必选（M）/可选（O）	参数位置	参数含义
client_id	String	M	Body	客户端ID。获取方法如下： 1. 登录承载REST服务端点的AppCube环境，单击“ 管理”，进入管理页面。 2. 选择“系统管理 > OAuth”，单击“新建”。 3. 输入OAuth名称，设置授权类型为“客户端模式”，选择一个用户（当鉴权成功后，将获得到和此用户相同的权限。注意不要具有选择匿名用户权限“Anonymous User Profile”的用户，因为该权限不能访问API），并单击“保存”。 4. 在OAuth管理列表页面，单击具体OAuth所在行，下载密钥文件到本地，从中获取客户端鉴权ID“client_id”取值。
client_secret	String	M	Body	客户端密钥。 参考client_id获取客户端鉴权密钥“client_secret”取值。
redirect_url	String	O	Body	重定向URL。
locale	String	O	Body	语言。 例如：en_US

将消息体加入后的请求如下所示，粗斜体字段需要根据实际值填写。

```
POST https://XXX.huawei.com/baas/auth/v1.0/oauth2/token          //XXX.huawei.com为一个示例说明，不是实际的域名，在实际环境中使用时，请替换为实际使用的域名。
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_id=********&client_secret=********
```

发起请求

到这里为止这个请求需要的内容就具备齐全了，您可以使用curl、Postman或直接编写代码等方式发送请求调用API。

对于使用OAuth 2.0的客户端鉴权模式获取用户access_token接口。返回的响应消息中“access_token”就是需要获取的用户access_token。有了access_token之后，您就可以使用access_token调用AppCube的其他API。

2.2.2 认证鉴权

基于AppCube开发的智能排班模型BO提供的接口都需要通过认证之后才可以访问。认证方式为：在请求消息头上设置“access-token”。

Token在计算机系统中代表令牌（临时）的意思，拥有Token代表拥有某种权限。Token认证是在调用API的时候将Token加到请求消息头，从而通过身份认证，获得操作API的权限。

通过AppCube的获取Token接口，获取access-token，用于调用API接口时进行鉴权，使用时在请求消息头上设置“access-token”。

步骤1 创建OAuth认证。

1. 登录AppCube环境。
2. 单击“管理”，选择“系统管理 > OAuth”，进入“OAuth管理”页面。
3. 单击“新建”，输入名称，授权类型选“客户端模式”，选择用户，如图2-1所示。

图 2-1 新建客户端模式 OAuth

说明

- 第三方通过OAuth认证接入系统后，将以选择的用户身份操作数据，所以需要确保选择的用户具有调用API的相关权限。
- 选择的用户的权限不能为匿名用户权限“Anonymous User Profile”，因为该权限用户没有访问API的权限。

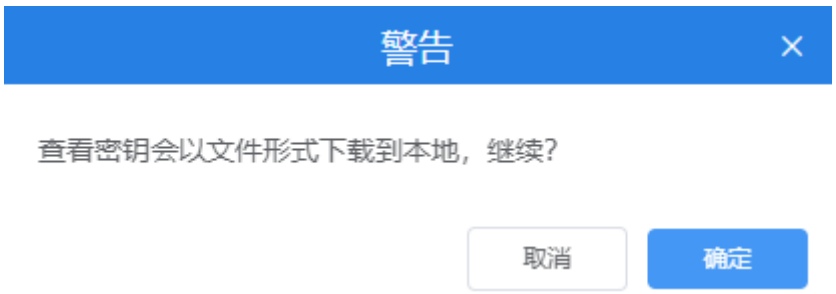
4. 单击“保存”，在“OAuth管理”页面展示新建的记录，如图2-2所示。

图 2-2 OAuth 管理列表

步骤2 获取客户端鉴权ID（client_id）和客户端秘钥（client_secret）。

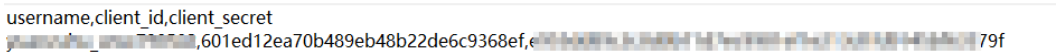
1. 单击新建记录“操作”列的，弹出“警告”对话框，如图2-3所示。

图 2-3 警告



2. 单击“确定”，下载密钥文件到本地，如图2-4所示，从中获取客户端鉴权ID（client_id）和客户端秘钥（client_secret）。

图 2-4 密钥



步骤3 根据鉴权客户端ID和鉴权密钥通过Post请求调用接口“https://***AppCube域名***/baas/auth/v1.0/oauth2/token”获取access_token。

AppCube域名替换为实际域名；********替换为获取到客户端鉴权ID（client id）和客户端秘钥（client_secret）。

```
POST https://AppCube域名/baas/auth/v1.0/oauth2/token
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_id=********&client_secret=********
```

返回的响应消息体可以获取到“access_token”。

```
{
  "access_token": "**************************",
  "expires_in": 1800,
  "token_type": "Bearer"
}
```

图 2-5 获取 access_token 示例-Headers

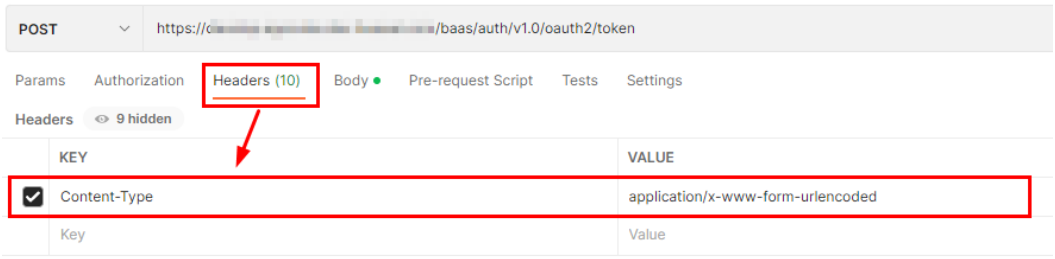


图 2-6 获取 access_token 示例-Body

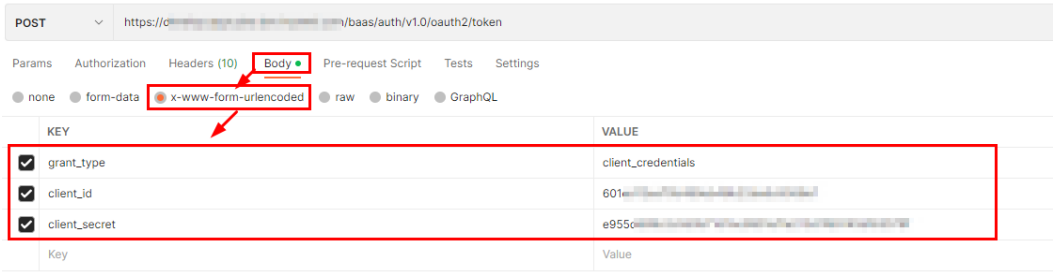
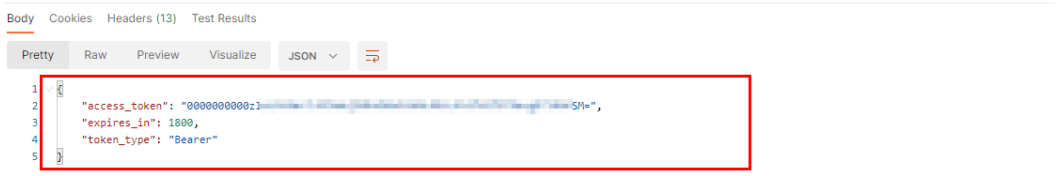


图 2-7 获取 access_token 示例-返回的响应消息体



步骤4 将Headers中的access-token参数设置为上一步得到的access_token，成功调用自定义的API接口。

AppCube域名替换为实际域名；********替换为获取到的access_token的值；Content-Type（消息体的类型（格式））， 请根据实际API实际情况配置，示例中为application/json。

```
POST https://AppCube域名/baas/auth/v1.0/oauth2/token
Content-Type: application/json
access-token: ********

{
   "pageStart":0,
   "pageSize":3,
   "deleteFlag":0
}
```

返回的响应消息体：

```
{
   "resCode": "0",
   "resMsg": "成功",
   "result": {
      "records": [],
      "total": 0
   }
}
```

图 2-8 调用 API 示例-Hearders

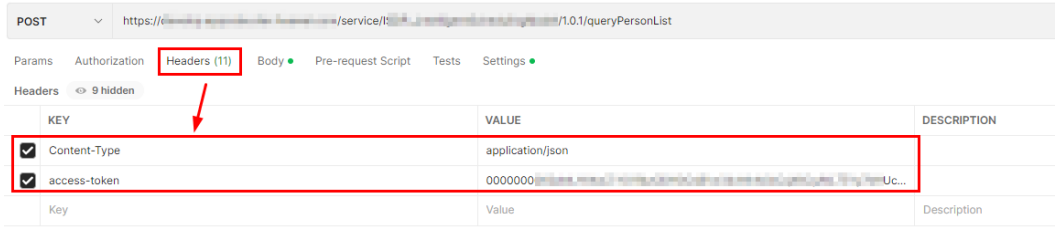


图 2-9 调用 API 示例-Body

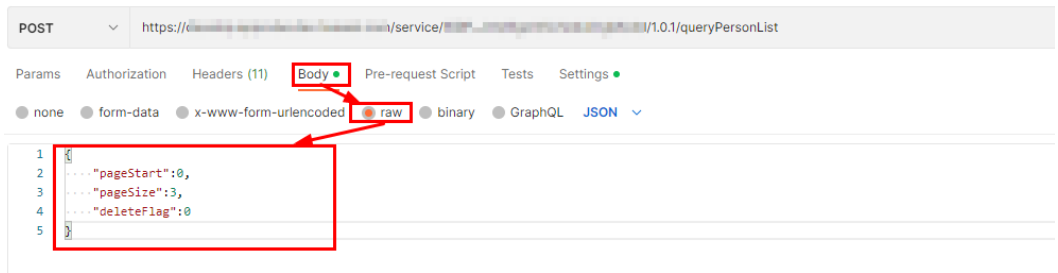
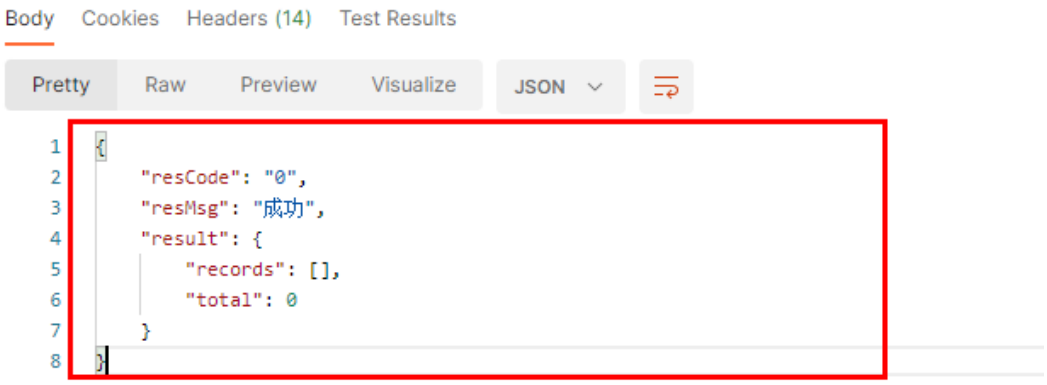


图 2-10 调用 API 示例-返回的响应消息体



----结束

2.2.3 返回结果

请求发送以后，您会收到响应，包含状态码、响应消息头和消息体。

状态码

请求发送以后，您会收到响应，包含状态码、响应消息头和消息体。

状态码是一组从2xx（成功）到4xx或5xx（错误）的数字代码，状态码表示了请求响应的状态，完整的状态码列表请参见状态码。

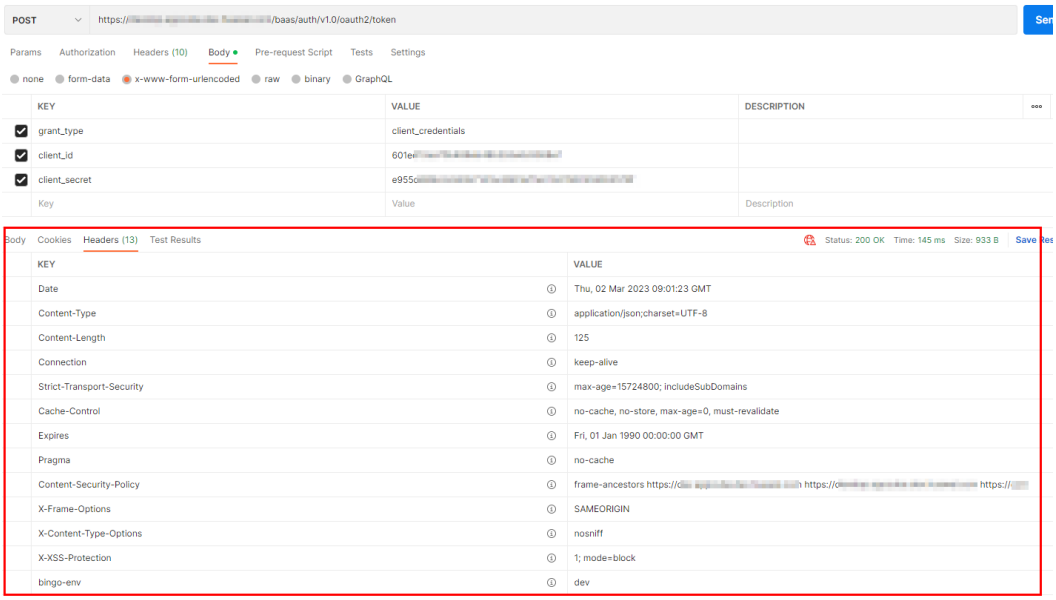
对于使用OAuth 2.0的客户端鉴权模式获取用户access_token接口，如果调用后返回状态码为“200”，则表示请求成功。

响应消息头

对应请求消息头，响应同样也有消息头，如“Content-type”。

对于使用OAuth 2.0的client credentials鉴权模式获取用户Token接口，返回如图2-11所示的消息头。

图 2-11 响应消息头



响应消息体

响应消息体通常以结构化格式返回，与响应消息头中Content-type对应，传递除响应消息头之外的内容。

对于使用OAuth 2.0的客户端鉴权模式获取用户access_token接口，返回如下消息体。其中“access_token”是需要获取的用户Token，取值以变量“*XXX*”表示。有了Token之后，您可以使用access_token认证调用其他API。

```
{
  "access_token": "XXX",
  "expires_in": 1800,
  "token_type": "Bearer"
}
```

当接口调用出错时，会返回错误码“400”及错误信息说明，错误响应的Body体格式如下所示。

```
{
    "error": "unauthorized_client",
    "error_description": "The client is not authorized to request a token using this method."
}
```

其中，error表示错误提示，error_description表示错误描述信息。

2.3 快速入门

本节以具体的场景为例，介绍如何调用API。

场景描述

以调用智能排班模型BO中的接口“批量添加离岗信息（batchAddOffDuty）”，添加离岗信息为例进行描述。

接口请求方法为：POST，AppCube域名为：XXX.huawei.com，接口URL：/service/ISDP__IntelligentSchedulingModel/1.0.0/batchAddOffDuty。

说明

XXX.huawei.com为一个示例说明，不是实际的域名，在实际环境中使用时，请替换为实际使用的域名。

前提条件

在AppCube环境中已部署智能排班模型BO。

操作步骤

步骤1 请参见**认证鉴权**获取到access_token。

步骤2 设置请求方法和请求URL。

```
POST https://XXX.huawei.com/service/ISDP__IntelligentSchedulingModel/1.0.0/batchAddOffDuty
```

步骤3 设置请求消息头。

```
Content-Type: application/json
access-token: ********
```

*********设置为**步骤1**中获取到的access_token的值。

步骤4 构造请求消息体。

```
{
   "offDutyInfoList": [
      {
         "basicInfo":{
            "personCode":"465477773434341",
            "personName":"王二",
            "offDutyType":"1",
            "offDutyStart":"2023-02-27 20:23:01",
            "offDutyEnd":"2023-02-27 20:23:01",
            "deleteFlag":0
         },
         "extendInfo":[
            {
               "fieldName":"ISDP__deleteFlag__CST",
               "fieldValue":"0"
            }
         ]
      }
   ]
}
```

返回如下响应，表示添加离岗信息成功。

```
{
   "resCode": "0",
   "resMsg": "成功"
}
```

----结束

2.4 API

2.4.1 鉴权

2.4.1.1 获取 access_token

功能介绍

该接口用于获取access_token，在调用API的时候将access_token加到请求消息头，从而通过身份认证，获得操作API的权限。

URL

请求方式	HTTPS地址	消息体类型
POST	https://***AppCube域名***/baas/auth/v1.0/oauth2/token	application/x-www-form-urlencoded

请求头

参数	类型	是否必填	描述
Content-Type	String	是	请求体内容类型。 填写为：application/x-www-form-urlencoded

请求参数

参数	类型	是否必填	描述
grant_type	String	是	授权模式，填写为：client_credentials，即客户端模式。
client_id	String	是	客户端ID。获取方法如下： 1. 登录承载REST服务端点的AppCube环境，单击“ 管理”，进入管理页面。 2. 选择“系统管理 > OAuth”，单击“新建”。 3. 输入OAuth名称，设置授权类型为“客户端模式”，选择一个用户（当鉴权成功后，将获得到和此用户相同的权限。注意不要具有选择匿名用户权限“Anonymous User Profile”的用户，因为该权限不能访问API），并单击“保存”。 4. 在OAuth管理列表页面，单击具体OAuth所在行  ，下载密钥文件到本地，从中获取客户端鉴权ID“client_id”取值。

参数	类型	是否必填	描述
client_secret	String	是	客户端密钥。 参考client_id获取客户端鉴权密钥“client_secret”取值。
redirect_url	String	否	重定向URL。
locale	String	否	语言。 例如：en_US

响应参数

参数	类型	描述
access_token	String	认证后可信任的Token凭证。
expires_in	String	当前access_token的有效期，单位：秒。
token_type	String	access_token类型。

请求示例

```
grant_type=client_credentials&client_id=********&client_secret=********    }
```

响应示例

```
{
  "access_token": "**********",
  "expires_in": 1800,
  "token_type": "Bearer"
}
```

状态码

状态码请参见状态码。

错误码

错误码请参见错误码。

2.4.2 排班

2.4.2.1 智能排班（API 名称：intelligentScheduling）

功能介绍

该接口用于智能排班。

URL

请求方式	HTTPS地址	消息体类型
POST	https://***AppCube域名***/service/ISDP__IntelligentSchedulingModel/1.0.1/intelligentScheduling	application/json

请求头

参数	类型	是否必填	描述
Content-Type	String	是	请求体内容类型 填写为：application/json
access-token	String	是	调用API要用到的访问令牌，**获取access_token**中获取的access_token的值

请求参数

参数	类型	是否必填	描述
startDate	String	是	智能排班时间段的开始日期，格式：yyyy-MM-dd
endDate	String	是	智能排班时间段的结束日期，格式：yyyy-MM-dd
requestId	String	是	请求id，通过id获取对应的结果
excuteType	Number	是	请求类型： ● 0：启动引擎计算 ● 1：获取排班计算结果
schedulingRuleName	String	否	排班规则名称
personInfoList	Object	否	人员信息列表
resourceInfoList	Object	否	配套资源信息列表
offDutyInfoList	Object	否	离岗信息列表
accruedLeaveInfoList	Object	否	积假信息列表

参数	类型	是否必填	描述
shiftInfoList	Object	否	班次信息列表
ruleList	Object	否	规则列表

响应参数

参数	类型	描述
resCode	String	响应状态码 ● 0：成功 ● 其他：失败，其他错误码说明请参考错误码
resMsg	String	响应描述，如果成功状态，通常会返回“成功”，其他情况返回具体的错误信息
result	Object	响应结果内容

result参数说明

参数	类型	描述
algorithmStatus	Number	计算状态： ● -1：平台接口网络异常 ● 0：已收到正在计算 ● 1：计算完成返回结果 ● 2：计算异常
scheduleResult	Number	排班结果

请求示例

```
{
   "startDate": "2023-03-01",
   "endDate": "2023-03-31",
   "excuteType": 1,
   "queryType": "月排班",
   "requestId": "1024"
}
```

响应示例

```
{
   "resCode": "0",
   "resMsg": "成功",
   "result": {
      "algorithmStatus": 0,
      "scheduleResult": []
```

```
  }
}
```

状态码

状态码请参见**状态码**。

错误码

错误码请参见**错误码**。

2.4.2.2 批量添加排班结果（API 名称：batchAddScheduleResult）

功能介绍

该接口用于批量添加排班结果信息。

URL

请求方式	HTTPS地址	消息体类型
POST	https://***AppCube域名***/service/ISDP__IntelligentSchedulingModel/1.0.1/batchAddScheduleResult	application/json

请求头

参数	类型	是否必填	描述
Content-Type	String	是	请求体内容类型 填写为：application/json
access-token	String	是	调用API要用到的访问令牌，**获取access_token**中获取的access_token的值

请求参数

参数	类型	是否必填	描述
startDate	String	是	排班结果的开始日期，格式：yyyy-MM-dd
endDate	String	是	排班结果的结束日期，格式：yyyy-MM-dd
shceduleResultList	ScheduleResult	是	排班结果列表

shceduleResultList参数说明

参数	类型	是否必填	描述
scheduleType	String	是	排班类型
date	String	是	排班日期，格式：yyyy-MM-dd
personCode	String	是	人员账号
personName	String	是	人员名称
shiftId	String	是	班次id
shiftName	String	是	班次名称

响应参数

参数	类型	描述
resCode	String	响应状态码 ● 0：成功 ● 其他：失败，其他错误码说明请参考错误码
resMsg	String	响应描述，如果成功状态，通常会返回“成功”，其他情况返回具体的错误信息

请求示例

```
{
   "startDate":"2023-03-01",
   "endDate":"2023-03-31",
   "shceduleResultList":[
      {
         "scheduleType":"0",
         "date":"2023-03-09",
         "personCode":"123456",
         "personName":"张三91",
         "shiftId":"c03P0000011GOUwzmM5W",
         "shiftName":"白班"
      },
      {
         "scheduleType":"0",
         "date":"2023-03-10",
         "personCode":"123456",
         "personName":"张四101",
         "shiftId":"c03P0000011GOUwzmM4W",
         "shiftName":"白班"
      }
   ]
}
```

响应示例

```
{
   "resCode": "0",
   "resMsg": "成功"
}
```

状态码

状态码请参见**状态码**。

错误码

错误码请参见**错误码**。

2.4.2.3 查询排班结果信息（API 名称：querySchedulingResult）

功能介绍

该接口用于查询排班结果信息。

URL

请求方式	HTTPS地址	消息体类型
POST	https://***AppCube域名***/service/ISDP__IntelligentSchedulingModel/1.0.1/querySchedulingResult	application/json

请求头

参数	类型	是否必填	描述
Content-Type	String	是	请求体内容类型 填写为：application/json
access-token	String	是	调用API要用到的访问令牌，**获取access_token**中获取的access_token的值

请求参数

参数	类型	是否必填	描述
pageStart	Number	是	查询起始页，从0开始
pageSize	Number	是	查询条数，取值范围：1～5000
orderBy	String	否	排序规则，如果不传不排序，传则排序

参数	类型	是否必填	描述
order	String	否	升降序，默认降序 ● desc：降序 ● asc：升序
startDate	String	否	查询时间段的开始日期，格式：yyyy-MM-dd
endDate	String	否	查询时间段的结束日期，格式：yyyy-MM-dd
scheduleType	String	否	排班类型搜索条件
personName	String	否	人员名称搜索条件，模糊查询
shiftName	String	否	班次名称搜索条件，模糊查询
personCode	String	否	人员账号搜索条件
deleteFlag	Number	否	删除标识 ● 0：未删除 ● 1：已删除 **说明** 不传时查询全部。

响应参数

参数	类型	描述
resCode	String	响应状态码 ● 0：成功 ● 其他：失败，其他错误码说明请参考错误码
resMsg	String	响应描述，如果成功状态，通常会返回“成功”，其他情况返回具体的错误信息
result	Object	响应结果内容

result参数说明

参数	类型	描述
records	Object	返回的查询记录

参数	类型	描述
total	Number	总条数

请求示例

```
{
   "pageStart":0,
   "pageSize":500,
   "startDate":"2023-03-07",
   "endDate":"2023-03-07",
   "scheduleType":"1",
   "personName":"张三"
}
```

响应示例

```
{
   "resCode": "0",
   "resMsg": "成功",
   "result": {
      "records": [
         {
            "ISDP2__schedulingDate__CST": "2023-03-07",
            "ISDP__deleteFlag__CST": 1,
            "ISDP__personCode__CST": "123456",
            "ISDP__personName__CST": "张三",
            "ISDP__scheduleType__CST": "1",
            "ISDP__shiftId__CST": "c03P0000011GOUwzmM5W",
            "ISDP__shiftName__CST": "白班",
            "accruedLeaveInfo": {
               "ISDP__accruedLeaveDuration__CST": 5,
               "ISDP__deleteFlag__CST": 0,
               "ISDP__personCode__CST": "123456",
               "ISDP__personName__CST": "张三",
               "accruedLeaveInfoExtend": [
                  {
                     "ISDP__accruedLeaveId__CST": "c1Sy0000011Gh8Qh2IO8",
                     "ISDP__accruedLeaveId__CST.__objectType": "ISDP__accruedLeaveInfo__CST",
                     "ISDP__accruedLeaveId__CST.name": "",
                     "ISDP__deleteFlag__CST": 0,
                     "id": "cFBY0000011Gh8QoPrlo",
                     "name": null
                  }
               ],
               "createdBy": "10gd00000119PKp32YgC",
               "createdBy.__objectType": "User",
               "createdBy.name": "XXX",
               "createdDate": "2023-02-28 18:32:54",
               "currencyIsoCode": "USD",
               "id": "c1Sy0000011Gh8Qh2IO8",
               "installedPackage": null,
               "lastModifiedBy": "10gd00000119PKp32YgC",
               "lastModifiedBy.__objectType": "User",
               "lastModifiedBy.name": "XXX",
               "lastModifiedDate": "2023-02-28 18:32:54",
               "name": null,
               "owner": "10gd00000119PKp32YgC",
               "owner.__objectType": "User",
               "owner.name": "XXX"
            },
            "createdBy": "10gd00000118KLOwrvKC",
            "createdBy.__objectType": "User",
            "createdBy.name": "XXX",
            "createdDate": "2023-03-06 11:04:31",
```

```
            "currencyIsoCode": "USD",
            "id": "cnyi0000011Q8W6IZpOS",
            "installedPackage": null,
            "lastModifiedBy": "10gd00000118KLOwrvKC",
            "lastModifiedBy.__objectType": "User",
            "lastModifiedBy.name": "XXX",
            "lastModifiedDate": "2023-03-06 14:07:09",
            "name": null,
            "owner": "10gd00000118KLOwrvKC",
            "owner.__objectType": "User",
            "owner.name": "XXX",
            "personInfo": null,
            "shiftInfo": null
        }
    ],
    "total": 1
  }
}
```


状态码

状态码请参见**状态码**。

错误码

错误码请参见**错误码**。

2.4.3 排班规则

2.4.3.1 批量添加排班规则（API 名称：batchAddSchedulingRuleTemplate）

功能介绍

该接口用于批量添加排班规则信息。

URL

请求方式	HTTPS地址	消息体类型
POST	https://***AppCube域名***/service/ISDP__IntelligentSchedulingModel/1.0.1/batchAddSchedulingRuleTemplate	application/json

请求头

参数	类型	是否必填	描述
Content-Type	String	是	请求体内容类型 填写为：application/json
access-token	String	是	调用API要用到的访问令牌，**获取access_token**中获取的access_token的值

请求参数

参数	类型	是否必填	描述
schedulingRuleTemplateList	SchedulingRuleTemplate	是	排班规则信息列表

schedulingRuleTemplateList参数说明

参数	类型	是否必填	描述
schedulingRuleName	String	是	排班规则名称
applicationScope	String	否	适用范围

响应参数

参数	类型	描述
resCode	String	响应状态码 ● 0：成功 ● 其他：失败，其他错误码说明请参考错误码
resMsg	String	响应描述，如果成功状态，通常会返回“成功”，其他情况返回具体的错误信息

请求示例

```
{
   "schedulingRuleTemplateList":[
      {
         "schedulingRuleName":"月",
         "applicationScope":"麻醉科"
      },
      {
         "schedulingRuleName":"周",
         "applicationScope":"麻醉科"
      },
      {
         "schedulingRuleName":"日",
         "applicationScope":"麻醉科"
      }
   ]
}
```

响应示例

```
{
   "resCode": "0",
```

```
  "resMsg": "成功"
}
```

状态码

状态码请参见**状态码**。

错误码

错误码请参见**错误码**。

2.5 状态码

状态码如**表2-5**所示。

表 2-5 状态码

状态码	编码	状态码说明
100	Continue	继续请求。 这个临时响应用来通知客户端，它的部分请求已经被服务器接收，且仍未被拒绝。
101	Switching Protocols	切换协议。只能切换到更高级的协议。 例如，切换到HTTP的新版本协议。
200	ok	请求成功。
201	Created	创建类的请求完全成功。
202	Accepted	已经接受请求，但未处理完成。
203	Non-Authoritative Information	非授权信息，请求成功。
204	NoContent	请求完全成功，同时HTTP响应不包含响应体。 在响应OPTIONS方法的HTTP请求时返回此状态码。
205	Reset Content	重置内容，服务器处理成功。
206	Partial Content	服务器成功处理了部分GET请求。
300	Multiple Choices	多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择。
301	Moved Permanently	永久移动，请求的资源已被永久的移动到新的URI，返回信息会包括新的URI。
302	Found	资源被临时移动。
303	See Other	查看其它地址。 使用GET和POST请求查看。

状态码	编码	状态码说明
304	Not Modified	所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。
305	Use Proxy	所请求的资源必须通过代理访问。
306	Unused	已经被废弃的HTTP状态码。
400	BadRequest	非法请求。 建议直接修改该请求，不要重试该请求。
401	Unauthorized	在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。
402	Payment Required	保留请求。
403	Forbidden	请求被拒绝访问。 返回该状态码，表明请求能够到达服务端，且服务端能够理解用户请求，但是拒绝做更多的事情，因为该请求被设置为拒绝访问，建议直接修改该请求，不要重试该请求。
404	NotFound	所请求的资源不存在。 建议直接修改该请求，不要重试该请求。
405	MethodNotAllowed	请求中带有该资源不支持的方法。 建议直接修改该请求，不要重试该请求。
406	Not Acceptable	服务器无法根据客户端请求的内容特性完成请求。
407	Proxy Authentication Required	请求要求代理的身份认证，与401类似，但请求者应当使用代理进行授权。
408	Request Time-out	服务器等候请求时发生超时。 客户端可以随时再次提交该请求而无需进行任何更改。
409	Conflict	服务器在完成请求时发生冲突。 返回该状态码，表明客户端尝试创建的资源已经存在，或者由于冲突请求的更新操作不能被完成。
410	Gone	客户端请求的资源已经不存在。 返回该状态码，表明请求的资源已被永久删除。
411	Length Required	服务器无法处理客户端发送的不带Content-Length的请求信息。
412	Precondition Failed	未满足前提条件，服务器未满足请求者在请求中设置的其中一个前提条件。

状态码	编码	状态码说明
413	Request Entity Too Large	由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个Retry-After的响应信息。
414	Request-URI Too Large	请求的URI过长（URI通常为网址），服务器无法处理。
415	Unsupported Media Type	服务器无法处理请求附带的媒体格式。
416	Requested range not satisfiable	客户端请求的范围无效。
417	Expectation Failed	服务器无法满足Expect的请求头信息。
422	UnprocessableEntity	请求格式正确，但是由于含有语义错误，无法响应。
429	TooManyRequests	表明请求超出了客户端访问频率的限制或者服务端接收到多于它能处理的请求。建议客户端读取相应的Retry-After首部，然后等待该首部指出的时间后再重试。
500	InternalServerError	表明服务端能被请求访问到，但是不能理解用户的请求。
501	Not Implemented	服务器不支持请求的功能，无法完成请求。
502	Bad Gateway	充当网关或代理的服务器，从远端服务器接收到了一个无效的请求。
503	ServiceUnavailable	被请求的服务无效。 建议直接修改该请求，不要重试该请求。
504	ServerTimeout	请求在给定的时间内无法完成。客户端仅在为请求指定超时（Timeout）参数时会得到该响应。
505	HTTP Version not supported	服务器不支持请求的HTTP协议的版本，无法完成处理。

2.6 错误码

调用接口出错后，将不会返回结果数据，调用方可根据每个接口对应的错误码来定位错误原因。当调用出错时，HTTP请求返回一个HTTP状态码（系统预置错误码返回的HTTP状态码为4xx或5xx，自定义错误码HTTP状态码为自定义的），返回的消息体中是具体的错误代码及错误信息。

错误响应 Body 体格式说明

当接口调用出错时，会返回错误码及错误信息说明，错误响应的Body体格式如下所示。

```
{
   "resCode": "错误码",
   "resMsg": "错误信息"
}
```

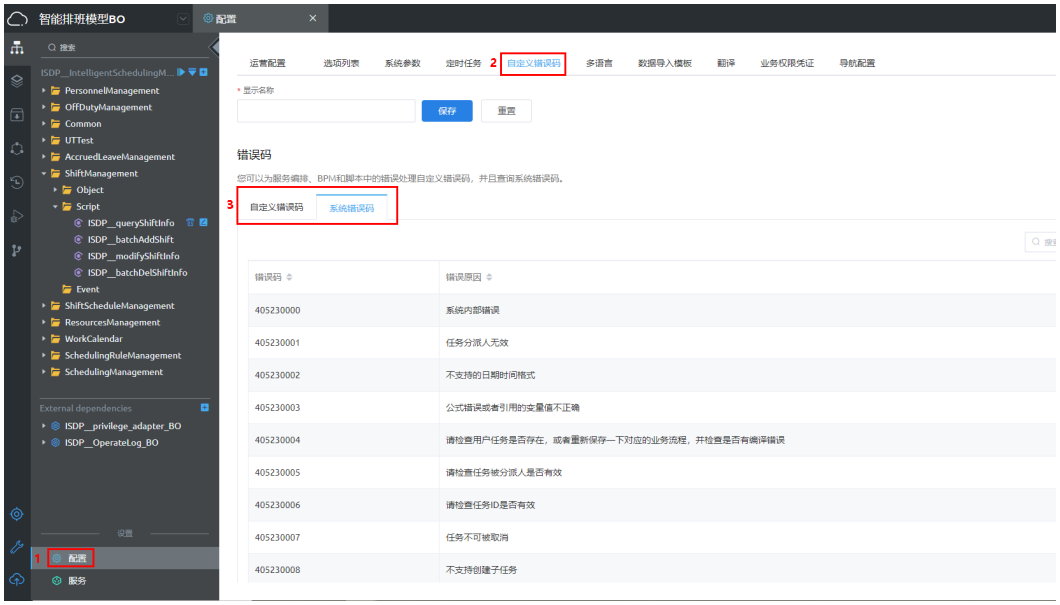
其中，resCode表示错误码，resMsg表示错误描述信息。

错误码说明

查看系统预置或自定义错误码有以下两个入口，请根据错误码查找具体的错误原因，并进行处理。例如“405230411”错误码，请根据错误码搜索，查看该错误码产生的原因，查看原因是“脚本参数无效”，按照提示进行处理。如果不明确处理方式，请联系工程师处理。

- 在AppCube的APP开发视图下左侧菜单栏下方选择“配置”，在打开的页签选择“自定义错误码”，在“自定义错误码”或“系统错误码”下可进行查看。

图 2-12 App 内查看错误码



- 在AppCube首页单击“管理”，进入管理页面，选择“设置 > 错误码”，在“自定义错误码”或“系统错误码”下可进行查看。

图 2-13 管理中查看错误码