阿里云
Aliware
分布式任务调度ScheduleX

API参考

产品版本：V3.1.0

文档版本：20171129

为了无法计算的价值 | 阿里云

法律声明

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

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	**禁止**： 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	**警告**： 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	**说明**： 您也可以通过按**Ctrl** + **A**选中全部文件。
>	多级菜单递进。	**设置** > **网络** > **设置网络类型**
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
courier字体	命令。	执行 cd /d C:/windows 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	bae log list --instanceid *Instance_ID*
[]或者[a|b]	表示可选项，至多选择一个。	ipconfig *[-all|-t]*
{}或者{a|b}	表示必选项，至多选择一个。	swich *{stand | slave}*

目录

法律声明.....I
通用约定.....I
1 接入指南.....1
2 API概览.....2
3 API列表.....3
3.1 createJob.....3
3.2 deleteJob.....5
3.3 instanceRunJob.....6
3.4 updateJob.....7
3.5 enableJob.....9
3.6 disableJob.....10
3.7 updateJobArguments.....11
3.8 getJobsForGroup.....12
3.9 instanceStopJob.....12
3.10 setJobMonitorConfig(AssembledMonitor monitorSeting).....13
3.11 updateJobMonitor(AssembledMonitor monitorSeting).....14
3.12 getMonitorSetings.....15
3.13 getJobRunningStatus.....16
3.14 getJobDetailRunningStatus.....16
3.15 grantGroupAuth.....17
3.16 getJobRunningHistoryStatus.....18
4 错误码列表.....19

1 接入指南

介绍如何通过API的方式接入SchedulerX。

背景信息

SchedulerX SDK/API是SchedulerX的开发工具包，通过SDK，可以使用代码方式方便的进行Job创建、修改、触发等操作，给用户提供了Web控制台之外另一种管控Job的方式。

操作步骤

1. 获取SDK。

   在pom.xml文件中增加以下SDK的依赖配置。

```
<dependency>
    <groupId>com.alibaba.schedulerx</groupId>
    <artifactId>schedulerx-client</artifactId>
    <version>2.1.1</version>
</dependency>
```

2. 初始化DtsInnerSDKManager（包括useId和url）。

```
DtsInnerSDKManager sdk = new DtsInnerSDKManager(userId, url);
```

   - userId：用户的ID，也就是登录时的账号。该账号需要有分组管理的权限。
   - url：Schedulerx控制台的URL, 格式如下：`http://${domainName}/dts-console/`, *domainName*替换成用户实际的域名。

预期结果

配置完成后，即可通过SDK调用具体API了。

2 API概览

表 2-1: API列表

API	描述
createJob	创建Job
deleteJob	删除Job
instanceRunJob	立即触发一次Job
updateJob	更新Job
enableJob	启用Job
disableJob	停用Job
updateJobArguments	更新Job的运行参数
getJobsForGroup	获取一个分组中的Job列表
instanceStopJob	立即停止正在执行的Job
setJobMonitorConfig(AssembledMonitor monitorSeting)	设置Job监控
getMonitorSetings	获取job的监控配置项
getJobRunningStatus	获取Job运行时状态
updateJobMonitor(AssembledMonitor monitorSeting)	更新监控配置
getJobDetailRunningStatus	获取一个Job详细的状态，包含分级任务信息
grantGroupAuth	分组授权
getJobRunningHistoryStatus	获取一个Job运行时的历史数据

3 API列表

3.1 createJob

创建调度任务

请求参数

表 3-1: 参数说明

名称	类型	说明
userGroupId	String	控制台创建的分组ID，比如1-3-3-417
job	Job	Job对象，里面设置Job的各种属性和参数

Job的属性和参数说明：

表 3-2: 属性及参数说明

名称	类型	说明
description	String	Job描述，或者备注解释
type	int	Job类型： • 简单Job单机版：API触发，com.alibaba.dts.common.constants.Constants.JOB_TYPE_API_SIMPLE • 简单Job单机版：定时触发，com.alibaba.dts.common.constants.Constants.JOB_TYPE_TIMER_SIMPLE • 并行计算Job：API触发，com.alibaba.dts.common.constants.Constants.JOB_TYPE_API_PARALLEL • 简单Job多机版：API触发，com.alibaba.dts.common.constants.Constants.JOB_TYPE_API_ALL_SIMPLE • 简单Job多机版：定时触发，com.alibaba.dts.common.constants.Constants.JOB_TYPE_TIMER_ALL_SIMPLE • 常驻任务：API启动，com.alibaba.dts.common.constants.Constants.JOB_TYPE_API_LONG_TIME • 常驻任务：定时启动，com.alibaba.dts.common.constants.Constants.JOB_TYPE_TIMER_LONG_TIME

名称	类型	说明
cronExpression	String	时间表达式
jobProcessor	String	Job处理器，全类名
maxInstanceAmount	int	maxInstanceAmount的值为0或1。 • 0：Job还在执行中，到了下次触发时间点跳过不触发，对应的policy为single instance。 • 1：Job还在执行中，到了下次触发时间点正常触发，对应的policy为multi instance。
jobArguments	String	用户自定义参数，本身没有任何格式要求，可以自定义格式，比如JSON字符串。在代码中可以通过如下方式获取该参数。String args = context.getJob().getJobArguments()
status	int	Job状态 • 停用Job，或者说Job处于停用状态 com.alibaba.dts.common.constants.Constants.JOB_STATUS_DISABLE • 启用Job，或者说Job处于启用状态 com.alibaba.dts.common.constants.Constants.JOB_STATUS_ENABLE

返回参数

表 3-3: 参数说明

名称	类型	说明
Result	Long	返回值为Job ID。

示例

```
public class TestCreateJob {

    public static void main(String[] args) {

        //userId 是控制台分组创建者的工号，比如42910
        String userId = "42910";

        //注意！DtsInnerSDKManager仅全局新建一次，然后可以在其它地方获取innerSDKMa
nager并调用里面的方法
        DtsInnerSDKManager innerSDKManager = new DtsInnerSDKManager(userId);

        Job job = new Job();
        job.setDescription("执行XXX的job");
        job.setType(Constants.JOB_TYPE_API_SIMPLE);
        job.setCronExpression("0 0 0 1 1 ? 2020");
        job.setJobProcessor("com.alibaba.getter.service.biz.schedule.dts.JobProcessor");
        job.setMaxInstanceAmount(Constants.START_POLICY_SINGLE_INSTANCE);
```

```
        job.setJobArguments("abc");
        job.setStatus(Constants.JOB_STATUS_ENABLE);
        Result<Long> result = innerSDKManager.createJob("1-3-3-417", job);
        System.out.println(result);

    }

}
```

3.2 deleteJob

删除调度任务

请求参数

表 3-4: 参数说明

名称	类型	说明
jobId	long	通过控制台或API创建的Job的ID

返回参数

表 3-5: 参数说明

名称	类型	说明
Result	Integer	返回值里是删除的Job个数

示例

```
public class TestDeleteJob {

    public static void main(String[] args) {

        //userId 是控制台分组创建者的工号，比如42910
        String userId = "42910";

        //注意！DtsInnerSDKManager仅全局新建一次，然后可以在其它地方获取innerSDKMa
nager并调用里面的方法
        DtsInnerSDKManager innerSDKManager = new DtsInnerSDKManager(userId);

        Result<Integer> result = innerSDKManager.deleteJob(123);
        System.out.println(result);

    }
```

```
}
```

3.3 instanceRunJob

立即触发一次调度任务

请求参数

表 3-6: 参数说明

名称	类型	说明
userGroupId	String	控制台创建的分组ID，比如1-3-3-417
jobId	long	控制台或者API创建的Job的ID，比如123

返回参数

表 3-7: 参数说明

名称	类型	说明
Result	Boolean	是否触发成功

示例

```
public class TestRunJob {

    public static void main(String[] args) {

        //userId 是控制台分组创建者的工号，比如42910
        String userId = "42910";

        //注意！DtsInnerSDKManager仅全局新建一次，然后可以在其它地方获取innerSDKMa
nager并调用里面的方法
        DtsInnerSDKManager innerSDKManager = new DtsInnerSDKManager(userId);

        Result<Boolean> result = innerSDKManager.instanceRunJob("1-3-3-417", 123L);
        System.out.println(result);
    }
```

```
}
```

3.4 updateJob

更新Job

请求参数

表 3-8: 参数说明

名称	类型	说明
userGroupId	String	控制台创建的分组ID，比如1-3-3-417。
job	Job	Job对象，里面设置Job的各种属性和参数

Job的属性和参数说明：

表 3-9: 属性及参数说明

名称	类型	说明
id	long	jobId，即Job的全局数字ID
description	String	Job描述，或者备注解释
type	int	Job类型： • 简单Job单机版：API触发，com.alibaba.dts.common.constants.Constants.JOB_TYPE_API_SIMPLE • 简单Job单机版：定时触发，com.alibaba.dts.common.constants.Constants.JOB_TYPE_TIMER_SIMPLE • 并行计算Job：API触发，com.alibaba.dts.common.constants.Constants.JOB_TYPE_API_PARALLEL • 简单Job多机版：API触发，com.alibaba.dts.common.constants.Constants.JOB_TYPE_API_ALL_SIMPLE • 简单Job多机版：定时触发，com.alibaba.dts.common.constants.Constants.JOB_TYPE_TIMER_ALL_SIMPLE • 常驻任务：API启动，com.alibaba.dts.common.constants.Constants.JOB_TYPE_API_LONG_TIME • 常驻任务：定时启动，com.alibaba.dts.common.constants.Constants.JOB_TYPE_TIMER_LONG_TIME
cronExpression	String	时间表达式

名称	类型	说明
jobProcessor	String	Job处理器，全类名
maxInstanceAmount	int	maxInstanceAmount的值为0或1。 • 0：Job还在执行中，到了下次触发时间点跳过不触发，对应的policy为single instance。 • 1：Job还在执行中，到了下次触发时间点正常触发，对应的policy为multi instance。
jobArguments	String	用户自定义参数，本身没有任何格式要求，可以自定义格式，比如JSON字符串。在代码中可以通过如下方式获取该参数。`String args = context.getJob().getJobArguments()`
status	int	Job状态 • 停用Job，或者说Job处于停用状态 com.alibaba.dts.common.constants.Constants.JOB_STATUS_DISABLE • 启用Job，或者说Job处于启用状态 com.alibaba.dts.common.constants.Constants.JOB_STATUS_ENABLE

返回参数

表 3-10: 参数说明

名称	类型	说明
Result	Integer	返回值的是修改的Job的个数

示例

```
public class TestUpdateJob {

    public static void main(String[] args) {

        //userId 是控制台分组创建者的工号，比如42910
        String userId = "42910";

        //注意！DtsInnerSDKManager仅全局新建一次，然后可以在其它地方获取innerSDKMa
nager并调用里面的方法
        DtsInnerSDKManager innerSDKManager = new DtsInnerSDKManager(userId);

        Job job = new Job();
        job.setId(40292L);//job.setId 是设置要修改的jobId，比如40292
        job.setDescription("执行OOO的job");
        job.setType(Constants.JOB_TYPE_API_SIMPLE);
        job.setCronExpression("0 0 0 1 1 ? 2020");
        job.setJobProcessor("com.alibaba.getter.service.biz.schedule.dts.JobProcessor");
        job.setMaxInstanceAmount(Constants.START_POLICY_SINGLE_INSTANCE);
        job.setJobArguments("修改后的自定义参数");
```

```
        job.setStatus(Constants.JOB_STATUS_ENABLE);

        Result<Integer> result = innerSDKManager.updateJob("1-3-3-417", job);
        System.out.println(result);
    }

}
```

3.5 enableJob

启用Job

请求参数

表 3-11: 参数说明

名称	类型	说明
jobId	long	控制台或API创建的Job的ID，比如40292

返回参数

表 3-12:

名称	类型	说明
Result	Boolean	返回是否启用成功

示例

```
public class TestEnableJob {

    public static void main(String[] args) {

        //userId 是控制台分组创建者的工号，比如42910
        String userId = "42910";

        //注意！DtsInnerSDKManager 注意！DtsInnerSDKManager仅全局新建一次，然后可以在其它地方获取innerSDKManager并调用里面的方法
        DtsInnerSDKManager innerSDKManager = new DtsInnerSDKManager(userId);

        Result<Boolean> result = innerSDKManager.enableJob(40292L);
        System.out.println(result);

    }
```

```
}
```

3.6 disableJob

停用Job

请求参数

表 3-13: 参数说明

名称	类型	说明
jobId	long	控制台或API创建的Job的ID，比如40292

返回参数

表 3-14:

名称	类型	说明
Result	Boolean	返回是否启用成功

示例

```
public class TestDisableJob {

    public static void main(String[] args) {

        //userId 是控制台分组创建者的工号，比如42910
        String userId = "42910";

        //注意！DtsInnerSDKManager 注意！DtsInnerSDKManager仅全局新建一次，然后可以在其它地方获取innerSDKManager并调用里面的方法
        DtsInnerSDKManager innerSDKManager = new DtsInnerSDKManager(userId);

        Result<Boolean> result = innerSDKManager.disableJob(40292L);
        System.out.println(result);

    }
```

```
}
```

3.7 updateJobArguments

更新Job的运行参数

请求参数

表 3-15: 参数说明

名称	类型	说明
jobId	long	控制台或API创建的Job的ID，比如40292
jobArguments	String	用户自定义参数，本身没有任何格式要求，可以自定义格式，比如JSON字符串。在代码中可以通过如下方式获取该参数。`String args = context.getJob().getJobArguments()`

返回参数

表 3-16: 参数说明

名称	类型	说明
Result	Integer	返回修改参数的job个数

示例

```
public class TestUpdateJobArgumentsJob {

    public static void main(String[] args) {

        //userId 是控制台分组创建者的工号，比如42910
        String userId = "42910";

        //注意！DtsInnerSDKManager 注意！DtsInnerSDKManager仅全局新建一次，然后可以在其它地方获取innerSDKManager并调用里面的方法
        DtsInnerSDKManager innerSDKManager = new DtsInnerSDKManager(userId);

        Result<Integer> result = innerSDKManager.updateJobArguments(40292L, "abc");
        System.out.println(result);

    }
```

```
}
```

3.8 getJobsForGroup

获取一个分组中的Job列表

请求参数

表 3-17: 参数说明

名称	类型	说明
userGroupId	String	分组ID

返回参数

表 3-18: 参数说明

名称	类型	说明
Result	String	• ResultCode:SUCCESS,更新成功。getData()得到Job列表。 • 其他ResultCode：表示失败，可以通过ResultCode对象的getInfomation()方法获取错误信息。

3.9 instanceStopJob

立即停止正在执行的Job

请求参数

表 3-19: 参数说明

名称	类型	说明
userGroupId	String	分组ID
jobId	long	Job ID

返回参数

表 3-20: 参数说明

名称	类型	说明
Result	String	• ResultCode:SUCCESS,通知停止运行成功。 • 其他ResultCode:表示失败，可以通过ResultCode对象的getInfomation()方法获取错误信息。

3.10 setJobMonitorConfig(AssembledMonitor monitorSeting)

设置Job监控

请求参数

表 3-21: 参数说明

名称	类型	说明
monitorSeting		监控的配置

AssembledMonitor中的需要设置的项：

```
private long jobId;

// Job执行超时上限;
private long timeoutLimit;

// 错误率;
private double errorRate;

List<WarningNotifier> notifiers = new LinkedList<WarningNotifier>();
```

WarningNotifier设置短信或者旺旺通知人：

```
// 内部是工号
private String wwId;
// 短信通知人ID
```

```
private String mobileId;
```

返回参数

表 3-22: 参数说明

名称	类型	说明
Result	String	• ResultCode:SUCCESS：设置成功。 • 其他ResultCode表示失败，可以通过ResultCode对象的getInfomation()方法获取错误信息。

3.11 updateJobMonitor(AssembledMonitor monitorSeting)

更新监控配置

请求参数

表 3-23: 参数说明

名称	类型	说明
monitorSeting	long	监控的配置

AssembledMonitor中的需要设置的项：

```
private long jobId;

// Job执行超时上限;
private long timeoutLimit;

// 错误率;
private double errorRate;

List<WarningNotifier> notifiers = new LinkedList<WarningNotifier>();
```

WarningNotifier设置短信或者旺旺通知人：

```
// 内部是工号
private String wwId;
// 短信通知人ID
```

```
private String mobileId;
```

返回参数

表 3-24: 参数说明

名称	类型	说明
Result	String	• ResultCode:SUCCESS：跟新成功。 • 其他ResultCode表示失败，可以通过ResultCode对象的getInfomation()方法获取错误信息。

3.12 getMonitorSetings

获取job的监控配置项

请求参数

表 3-25: 参数说明

名称	类型	说明
jobId	long	Job ID

返回参数

表 3-26: 参数说明

名称	类型	说明
Result	String	• Result ResultCode: SUCCESS，获取成功。 • 其他ResultCode表示失败，可以通过ResultCode对象的getInfomation()方法获取错误信息。

3.13 getJobRunningStatus

获取Job运行时状态

请求参数

表 3-27: 参数说明

名称	类型	说明
jobId	long	Job ID
jobDesc	String	Job描述
runningStatus	int	Job是否在运行
overallProgress	TreeMap	总体进度

返回参数

表 3-28: 参数说明

名称	类型	说明
Result	String	• Result ResultCode: SUCCESS，获取成功。 • 其他ResultCode表示失败，可以通过ResultCode对象的getInfomation()方法获取错误信息。

3.14 getJobDetailRunningStatus

获取一个Job详细的状态，包含分级任务信息。

请求参数

表 3-29: 参数说明

名称	类型	说明
jobId	long	Job ID
jobDesc	String	Job描述
progressDetail	ProgressDetail	Job详细进度

返回参数

表 3-30: 参数说明

名称	类型	说明
Result	String	• Result ResultCode: SUCCESS，获取成功。 • 其他ResultCode表示失败，可以通过ResultCode对象的getInfomation()方法获取错误信息。

3.15 grantGroupAuth

分组授权

请求参数

表 3-31: 参数说明

名称	类型	说明
userGroupId	String	用户分组
ownerUserId	String	分组原来的创建者
grantUserId	String	要授权的用户ID

返回参数

表 3-32: 参数说明

名称	类型	说明
Result	String	• Result ResultCode: SUCCESS，授权成功。 • 其他ResultCode表示失败，可以通过ResultCode对象的getInfomation()方法获取错误信息。

3.16 getJobRunningHistoryStatus

获取一个Job运行时的历史数据

请求参数

表 3-33: 参数说明

名称	类型	说明
jobId	long	Job ID

返回参数

表 3-34: 参数说明

名称	类型	说明
Result	String	• Result ResultCode: SUCCESS，获取成功。 • 其他ResultCode表示失败，可以通过ResultCode对象的getInfomation()方法获取错误信息。

4 错误码列表

API调用成功时，返回100给客户端。当出现调用失败时，会返回对应的错误码。

表 4-1: 错误码说明

错误码	说明
101	失败
102	找不到该方法
103	鉴权失败，无效的时间戳
104	鉴权失败，无效的签名
105	鉴权失败，没有资源访问权限
106	鉴权失败，客户端版本错误
107	鉴权失败，资源访问受限
200	创建成功！
201	时间表达式描述的是一个过去的时间点，Job将永远不会被触发
202	定时触发器初始化失败
203	定时触发器放入内存Job池失败
204	定时触发器配置持久化失败
205	创建成功
206	创建失败，服务器列表异常
207	由于SchedulerX内部Job备份创建失败，Job处于禁用状态，可以点击**启用**使Job恢复正常
208	由于SchedulerX内部自动启用Job失败，Job处于禁用状态，可以点击**启用**使Job恢复正常
209	创建成功
210	创建定时器关系映射失败
211	创建失败
212	内部Job已存在
222	时间表达式不规范
223	参数不合法,为空

错误码	说明
224	查询集群组失败
225	Job创建成功，但内部启动失败，请重新启用
226	由于系统内部出现异常，创建的Job不可用，请删除后重新创建
227	启用Job失败
228	启用Job失败，系统出现异常，请先停用，再启用
227	停用Job失败
228	停用Job失败，系统出现异常，请先启用，再去停用
229	系统更新出现异常，需要重新操作才能生效
230	创建时写入DB失败
231	查询DB失败
232	删除DB失败
233	分发的对象没有实现java.io.Serializable接口
234	该时间表达式已废弃
300	删除成功
301	Job内存实例为空
302	删除Job异常
303	从内存清除Job异常
304	删除持久化的Job异常
305	删除成功
306	删除失败，服务器列表异常
307	删除成功
308	系统繁忙，请稍后再试
309	删除Job时,接触机器和Job的绑定关系失败
310	集群机器没有启动或者没达到最低备份数量
311	内部查询出错
312	参数不合法
313	创建Job机器关系异常
314	ZooKeeper发布数据错误

错误码	说明
315	不存在这样的对应关系
316	查询Job和机器关系异常
317	客户端没有正在运行的机器
318	调用Dauth鉴权失败
319	Dauth删除资源错误
320	重放错误
400	查询成功
401	job内存实例为空
402	查询成功
403	查询job为空
404	统计Group中Job数目失败
405	查询Job失败
406	前置Job执行失败
500	更新成功
501	更新持久化Job异常
502	更新成功
503	更新失败，内存Job为空
504	更新失败，服务器列表异常，请稍后再试
505	系统繁忙，请稍后再试
506	更新成功
600	拉取任务列表成功
601	拉取任务列表结束
602	抢锁失败
603	拉取结束
700	任务列表是空的
701	服务器宕机
702	分发任务失败
703	常驻任务不充许多级分发

错误码	说明
770	任务名称为空
703	获取所有集群信息异常
704	获取用户组异常
713	获取用户所有组异常
705	数据库创建用户组异常
706	删除组异常
707	查询监控配置异常
708	监控没有设置
709	更新监控异常
710	查询Job状态异常
711	查询进度异常
712	查询详细进度异常
714	查询用户异常
715	该用户还没有开通SchedulerX服务
716	资源不属于该用户
717	改变状态失败
718	创建用户和用户组关联异常
719	查询用户和组关系异常
720	删除用户组内部异常
721	查询Job关系依赖异常
722	创建Job关系依赖异常
723	删除Job关系依赖异常
724	不存在这样的依赖关系
725	已经调用过的不需要重新调用
726	依赖Job等待被调用
727	更新Job依赖表异常
800	存活
801	I am over

错误码	说明
802	I am exit
803	I am crash
901	参数为空
902	序列化全局变量失败
903	服务器宕机
904	设置全局变量失败
1001	触发失败，原因：该Job正在运行过程中
1002	触发失败，原因：没有可用的客户端
1003	触发失败，原因：获取实例异常
1004	触发失败，原因：实例状态异常
1005	触发失败，原因：抢锁，获取执行权限失败
1006	触发失败，原因：抢锁成功后获取实例异常
1007	触发失败，原因：抢锁成功后实例状态异常
1008	触发失败，原因：初始化根任务失败
1009	触发失败，原因：触发客户端失败
1003	URL访问异常
1101	accessKey或groupId不正确
1102	系统内部错误
1110	查询指定机器失败
1111	删除原有的指定机器关系失败
1112	更新机器JOB关系失败
1113	找不到Job备份服务器列表
1201	推送任务类型错误
1202	executorUnitMap缺失
1203	executorUnit为空
1303	该用户已经授权过了
1501	发送任务强制中断
10001	客户端NODE节点接收队列满

错误码	说明
2303	ZooKeeper错误
2304	SchedulerX console错误
2305	SchedulerX server错误
9999	没有对应的枚举类