Devops平台

工具链对接接口说明书

杭州谐云科技有限公司

2018年4月25日

版本记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 创建/修改人 | 创建/修改时间 | 创建/修改内容 |
| V1.0 | 张魁 | 2019-4-15 | 创建本文档，完成文档骨架、内容。 |

目 录

[1 接口说明 3](#_Toc6220866)

[1.1 接入验证方式 3](#_Toc6220867)

[1.1.1 容器云平台 3](#_Toc6220868)

[1.1.2 Apm平台 3](#_Toc6220869)

[1.1.3 Gitlab 3](#_Toc6220870)

[1.1.4 Jira/Confluence 3](#_Toc6220871)

[1.1.5 Sonar 4](#_Toc6220872)

[1.2 接口说明 4](#_Toc6220873)

[1.2.1 容器云平台 4](#_Toc6220874)

[1.2.2 Apm平台 6](#_Toc6220875)

[1.2.3 Gitlab 6](#_Toc6220876)

[1.2.4 Jira 10](#_Toc6220877)

[1.2.5 Confluence 12](#_Toc6220878)

[1.2.6 Sonar 12](#_Toc6220879)

# 接口说明

## 接入验证方式

### 容器云平台

|  |  |
| --- | --- |
| 参数 | 说明 |
| url | 容器云平台的地址 |
| username | 登录容器云平台账号 |
| password | 登录容器云平台账号的密码 |

### Apm平台

|  |  |
| --- | --- |
| 参数 | 说明 |
| url | apm平台的地址 |
| username | apm平台账号 |
| password | apm平台账号的密码 |

### Gitlab

|  |  |
| --- | --- |
| 参数 | 说明 |
| url | gitlab的地址 |
| private\_token | 私有token |

### Jira/Confluence

|  |  |
| --- | --- |
| 参数 | 说明 |
| jira url | jira url地址 |
| confluence url | confluence url地址 |
| username | jira/confluence平台共同账号 |
| password | jira/confluence平台共同账号的密码 |

### Sonar

|  |  |
| --- | --- |
| 参数 | 说明 |
| url | sonar url地址 |
| username | sonar账号 |
| password | sonar账号密码 |

## 接口说明

### 容器云平台

* 创建租户

|  |  |
| --- | --- |
| 参数 | 说明 |
| tenantName | 租户名称 |
| tenantNameEn | 租户英文简称 |
| status | 租户权限 |
| managerName | 管理员名称 |

|  |  |
| --- | --- |
| 返回值 | 说明 |
| tenantId | 租户Id |

* 删除租户

|  |  |
| --- | --- |
| 参数 | 说明 |
| tenantId | 租户Id |

* 分配资源

|  |  |
| --- | --- |
| 参数 | 说明 |
| tenantId | 租户Id |
| List<applyTenantResourceBo>  (clusterId、cpu、memory) | 申请的资源列表 (集群Id、cpu配额、内存配额) |
| List<applyTenantStorageBo>  (clusterId、storageName、storageQuota) | 申请的存储资源列表  (集群Id、存储名称、存储配额) |

* 增加租户成员

|  |  |
| --- | --- |
| 参数 | 说明 |
| tenantId | 租户Id |
| username | 用户名称 |

* 删除租户成员

|  |  |
| --- | --- |
| 参数 | 说明 |
| tenantId | 租户Id |
| username | 用户名称 |

* 创建项目

|  |  |
| --- | --- |
| 参数 | 说明 |
| tenantId | 租户Id |
| ProjectName | 项目名称 |
| ProjectNameEn | 项目英文简称 |
| ProjectManagerName | 项目管理员 |

|  |  |
| --- | --- |
| 返回值 | 说明 |
| projectId | 项目Id |

删除项目

|  |  |
| --- | --- |
| 参数 | 说明 |
| tenantId | 租户Id |
| projectId | 项目Id |

* 增加项目成员

|  |  |
| --- | --- |
| 参数 | 说明 |
| tenantId | 租户Id |
| projectId | 项目Id |
| username | 用户名字 |
| roleId | 用户角色 |

* 删除项目成员

|  |  |
| --- | --- |
| 参数 | 说明 |
| tenantId | 租户Id |
| projectId | 项目Id |
| username | 用户名字 |
| roleId | 用户角色 |

* 查询所有集群

|  |  |
| --- | --- |
| 参数 | 说明 |
| 无 | 无 |

* 查询某一集群的空闲节点

|  |  |
| --- | --- |
| 参数 | 说明 |
| clusterId | 集群Id |

* 查询某一集群的存储资源

|  |  |
| --- | --- |
| 参数 | 说明 |
| clusterId | 集群Id |

### Apm平台

* 创建项目

|  |  |
| --- | --- |
| 参数 | 说明 |
| projectName | 项目名字 |
| projectKey | 项目标识 |

* 删除项目

|  |  |
| --- | --- |
| 参数 | 说明 |
| projectKey | 项目标识 |

* 增加用户

|  |  |
| --- | --- |
| 参数 | 说明 |
| name | 姓名 |
| password | 密码 |
| mail | 邮箱 |
| mobile | 电话 |
| company | 公司 |
| role | 权限 |
| department | 部门（可为null） |
| post | 职位（可为null） |

* 为项目分配用户

|  |  |
| --- | --- |
| 参数 | 说明 |
| projectName | 项目名字 |
| projectKey | 项目标识 |
| userList | 用户id列表 |

### Gitlab

* 获得组gorup：

*GitlabGroup* getGroup(*String groupName*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| groupName | 组名称 |

* 创建组gorup：

*GitlabGroup* createGroup(*String groupName*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| groupName | 组名称 |

* 删除组group：

void deleteGroup(*Integer groupId*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| groupId | 组id |

* 获得gitlab用户：

*GitlabUser* getUser(*Integer userId*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| userId | 用户id |

* 创建gitlab用户 ：

*GitlabUser* createUser(*CreateUserRequest request*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| request | 创建用户的request，包含创建用户所需的所有信息：  private *String* email; private *String* password; private *Boolean* resetPassword; private *String* username; private *String* name; private *String* skype; private *String* linkedin; private *String* twitter; private *String* websiteUrl; private *String* organization; private *Integer* projectsLimit; private *String* externUid; private *String* provider; private *String* bio; private *String* location; private *Boolean* admin; private *Boolean* canCreateGroup; private *Boolean* skipConfirmation; private *Boolean* external; private *String* avatar;  可以使用构造函数，也可以使用getter和setter。构造函数原型如下：  public CreateUserRequest(*String name*, *String username*, *String email*) |

* 删除gitlab用户 ：

void deleteUser(*Integer targetUserId*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| targetUserId | 目标用户id |

* 修改gitlab用户 ：

*GitlabUser* updateUser(*Integer targetUserId*, *CreateUserRequest request*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| targetUserId | 目标用户id |
| request | 包含用户的新信息的request，详见创建用户接口说明 |

* 创建gitlab项目 ：

*GitlabProject* createProject(*String projectName*, *String groupName*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| projectName | 项目名称 |
| groupName | 组名称 |

* 删除gitlab项目 ：

void deleteProject(*Serializable projectId*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| projectId | 项目id |

* 分配项目成员 ：

*GitlabProjectMember* assignMemberToProject(*Integer projectId*, *Integer userId*, *GitlabAccessLevel accessLevel*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| projectId | 项目id |
| userId | 用户id |
| accessLevel | 访问权限：  ***Guest***(10)  ***Reporter***(20) ***Developer***(30) ***Master***(40) ***Owner***(50) |

* 删除项目成员：

void removeMemberFromProject(*Integer projectId*, *Integer userId*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| projectId | 项目id |
| userId | 用户id |

### Jira

* 根据项目关键字获得项目：

*ProjectBean* getProjectByKey(final *String projectKey*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| projecKey | 项目关键字 |

* 获得项目角色信息（Administrators, Developers和Users）：

*RolesBean* getProjectRoles(final *String projectKey*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| projecKey | 项目关键字 |

* 获得所有项目：

*List*<*ProjectBean*> listProject();

|  |  |
| --- | --- |
| 参数 | 说明 |

* 创建项目：

void createProject(*Map projectDataMap*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| projecDataMap | 项目信息，所需的(Key, Value)如下：  ("notificationScheme", 10000);  ("description", "");  ("lead", "admin");  ("url", "");  ("projectTemplateKey", "com.atlassian.jira-core-project-templates:jira-core-project-management");  ("avatarId", 10200);  ("issueSecurityScheme", 10000);  ("name", "Example1");  ("permissionScheme", 0);  ("assigneeType", "PROJECT\_LEAD");  ("projectTypeKey", "business");  ("key", "EX1"); // key不可为空  ("categoryId", 10000); // 可省略 |

* 删除项目

void removeProject(final *String projectKey*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| projecKey | 项目关键字 |

* 添加项目成员：

*ProjectRoleBean* assignMembersToProject(final *String projectKey*, final *Integer projectRoleId*, final *Map usersOrGroups*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| projecKey | 项目关键字 |
| projectRoleId | 分配给的项目角色的id，如Developers角色的id是10103，需要根据具体项目具体角色而定 |
| usersOrGroups | 亟待分配的成员信息Map，若是分配用户，则key为"user"，value为用户名所组成的数组；若是分配组，则key为"group"，value为组名所组成的数组 |

* 删除项目用户成员：

void removeUserFromProject(final *String projectKey*, final *Integer projectRoleId*, final *String username*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| projecKey | 项目关键字 |
| projectRoleId | 分配给的项目角色的id，如Developers角色的id是10103，需要根据具体项目具体角色而定 |
| username | 用户名 |

* 删除项目组成员：

void removeGroupFromProject(final *String projectKey*, final *Integer projectRoleId*, final *String groupname*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| projecKey | 项目关键字 |
| projectRoleId | 分配给的项目角色的id，如Developers角色的id是10103，需要根据具体项目具体角色而定 |
| groupname | 组名 |

* 添加组成员：

*GroupMemembersBean* assignMemberToGroup(final *String groupname*, final *String username*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| groupname | 组名 |
| username | 用户名 |

* 删除组成员：

void removeMemberFromGroup(final *String groupname*, final *String username*);

|  |  |
| --- | --- |
| 参数 | 说明 |
| groupname | 组名 |
| username | 用户名 |

### Confluence

* 创建空间：

*SpaceBean* createSpace(final *CreateSpaceRequest request*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| request | 包含创建空间所需要的基本信息的request，其包含参数为：    具体形式如下例子：    description和metadata可以先置为null |

* 删除空间：

void removeSpace(final *String spaceKey*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| spaceKey | 空间的key |

* 获得空间基本信息：

*SpaceBean* getScpaceByKey(final *String spaceKey*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| spaceKey | 空间的key |

* 增加空间关注者：

*SpaceWatcherBean* addSpaceWatcher(final *String spaceKey*, final *String watcherUsername*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| spaceKey | 空间的key |
| watcherUsername | 关注者的用户名 |

* 删除空间关注者：

void removeSpaceWatcher(final *String spaceKey*, final *String watcherUsername*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| spaceKey | 空间的key |
| watcherUsername | 关注者的用户名 |

* 获得内容基本信息：

*ContentBean* getContentById(final *String contentId*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| contentId | 内容的id |

* 添加内容的关注者：

*ContentWatcherBean* addContentWatcher(final *String contentId*, final *String watcherUsername*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| contentId | 内容的id |
| watcherUsername | 关注者的用户名 |

* 删除内容的关注者：

void removeContentWatcher(final *String contentId*, final *String watcherUsername*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| contentId | 内容的id |
| watcherUsername | 关注者的用户名 |

* 获得用户基本信息：

*UserBean* getUserByUsername(final *String username*) throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |
| username | 用户名 |

* 获得当前用户基本信息：

*UserBean* getCurrentUser() throws *ToolClientException*;

|  |  |
| --- | --- |
| 参数 | 说明 |

### Sonar