联合作战人机协同博弈挑战赛

智能体开发指南

国防科大系统工程学院

中国电子科技集团公司第五十二研究所

北京机械设备研究所

二〇二四年八月

目　录

1 概述

1.1 总体介绍

智能体开发框架封装了连接灵弈服务器、订阅并解析态势数据、数据处理、推演流程控制等大量而繁琐的底层编码，为选手提供基于灵弈平台的智能体 Python 开发环境和接口，选手可专注于战术战法实现和算法设计，在赛前准备阶段按照智能体开发框架的开发规范和开发指南开发好智能体后，将智能体迁移到比赛环境中，通过灵弈客户端的智能体调度功能使用图 1。

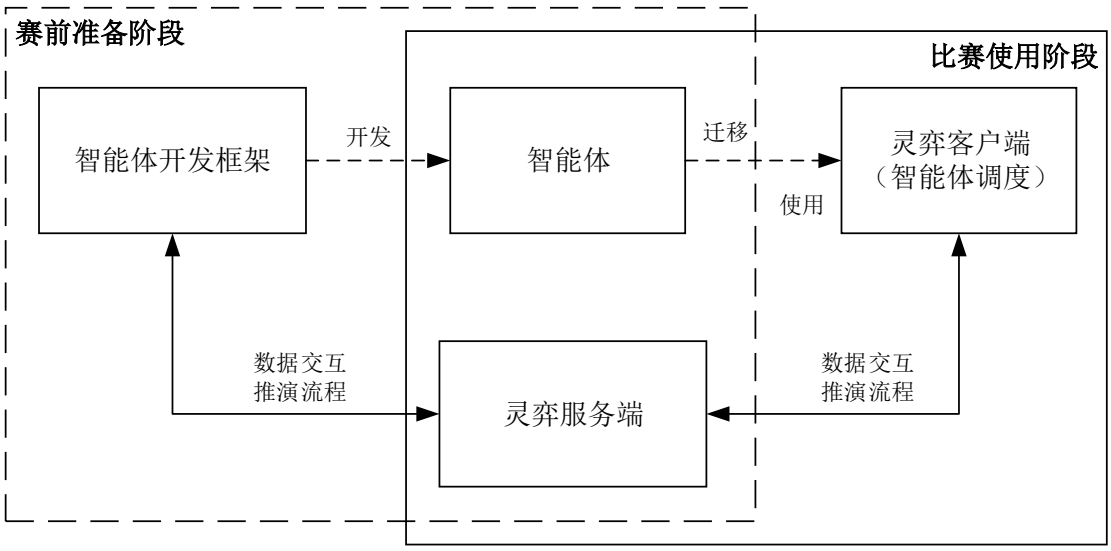


图 1 总体概述图

1.2 文件系统说明

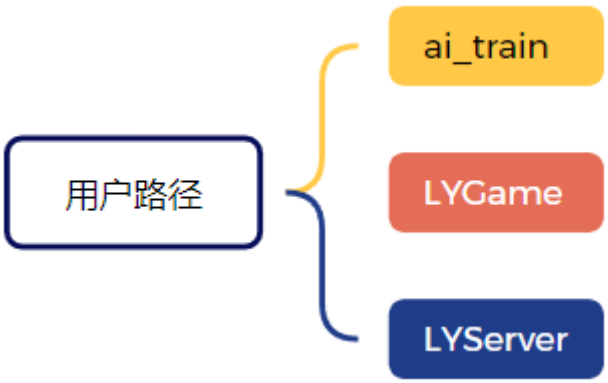


图 2 智能体开发包文件系统图

下载智能体开发框架后解压缩放在非中文路径下

ai_train：智能体开发框架

LYGame：灵弈客户端

LYServer：灵弈服务端

1.2.1 智能体开发框架文件说明

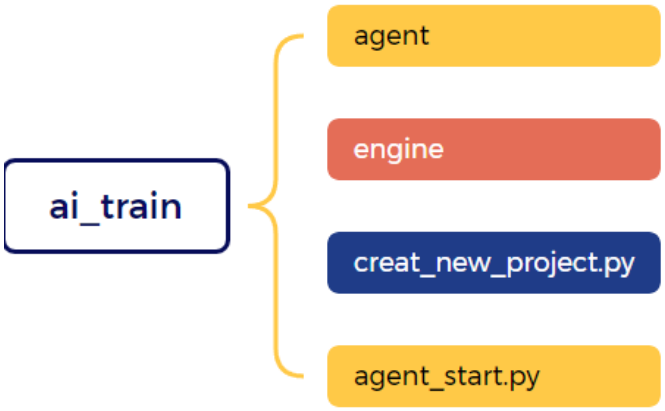


图 3 智能体开发框架文件系统图

智能体开发框架各文件如图 3 所示，各用途说明如下：

ai_train：工作目录

ai_train/agent：智能体参考案例

ai_train/engine：智能体开发框架

ai_train/creat_new_project.py：新建一个智能体开发模板工程

ai_train/agent_start.py：智能体配置和启动入口文件

1.2.2 灵弈客户端文件说明

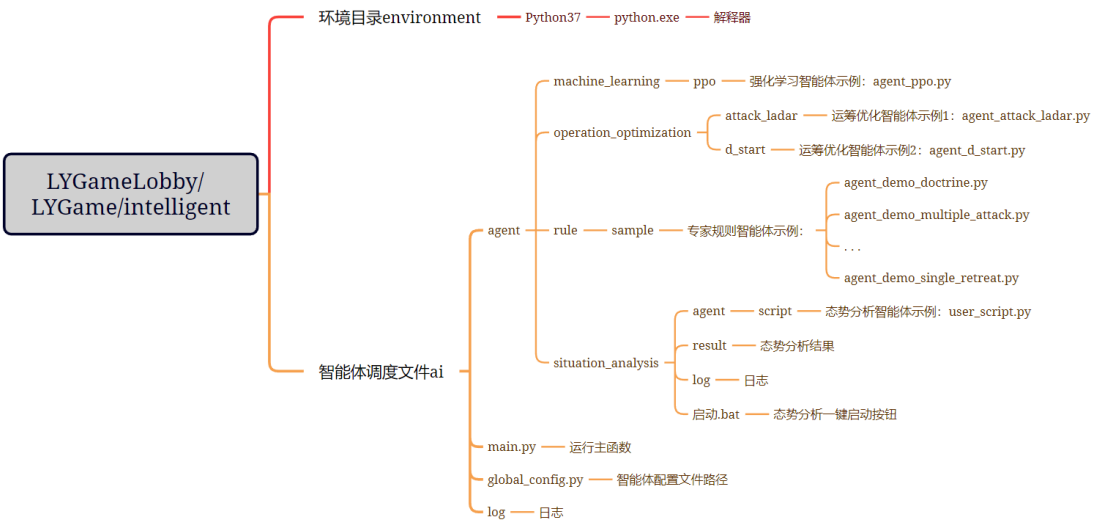


图 4 智能体使用文件结构

智能体使用的所有相关文件皆放在客户端目录（LYGameLobby/LYGame）下的智能体目录（intelligent）中。Intelligent 文件夹中包括了环境目录（environment）和智能体调度文件（ai）。

环境目录（environment）中主要包含了智能体运行所需的 python 虚拟环境。智能体调度文件（ai）中包括了智能体算法文件（agent）、智能体运行主函数（main.py）、手动智能体配置文件（global.py）。智能体算法文件（agent）中共有四个文件夹，主要包括强化学习（machine_learning）、运筹优化（operation_optimization）、专家规则（rule）和态势分析智能体

算法文件夹（situation_analysis)。在每一类态势分析智能体算法文件夹中都提供了智能体算法示例。

选手在比赛使用时，只需将来自智能体开发框架的算法文件放置到智能体调度文件 ai/agent 路径下对应分类的算法文件夹内即可完成赛前智能体使用的准备工作

2 智能体开发与使用

2.1 智能体示例

智能体示例文件中包含了强化学习、运筹优化、专家规则和态势分析四类智能体，具体包含条令条例设置、航线规划、火力打击、任务创建、脱离战斗、防空反导等战法和功能等接口使用范例。具体操作方式可查看智能体使用流程，agent 中的智能体范例概述如下表所示。

智能体分类	名称	功能描述	输入
强化学习	agent_ppo.py	强化学习智能体编码范例	情报信息：蓝方雷达 动作单元：任选一个图-160型轰炸机 参考点：任选一个
运筹优化	agent_attack_ladar.py	运筹优化智能体编码范例	同上

	agent_d_start.py	D*算法编码范例	动作单元：可移动的单元 参考点：目的点
专家规则	agent_demo_basic.py	智能体 Agent 的基础 demo。其他智能体 demo 均以此 Agent 类为基础进行创建	——
	agent_demo_doctrine.py	实现条令条例的设置。具体为：打开电磁管控、调整武器状态、交战状态等。	——
	agent_demo_huoli.py	首先设置所有武器雷达开机，并设置对空条例为自动开火。之后遍历所有被发现的敌方目标，根据一个简单的目标分配算法分配客户端选择的地导营和敌方目标，并下发手动打击指令。	动作（地导营）
	agent_demo_multipl	设置打击任务。获取到客	实体单

	e_attack.py	户端的实体单元和情报信息，将客户端选择的信息加入到打击任务中。	元（我方单元）、情报信息（敌方目标）
	agent_demo_multiple_move.py	控制我方单元移动到参考点附近。在获取到客户端的实体单元（我方单元）和参考点后，遍历我方单元使其移动至选定参考点附件的随机位置。	实体单元（需要移动的单元）、参考点（我方单元需要到达的位置）
	agent_demo_multiple_patrol.py	设置巡逻任务。获取到客户端的实体单元和情报信息，将客户端选择的信息加入到巡逻任务中。	实体单元（我方单元）、参考点（巡逻区域边

			界点）
	agent_demo_multiple_retreat.py	控制我方单元移动进行撤退。在获取到客户端的实体单元（我方单元），遍历我方单元取消其任务并设置返航。	实体单元（我方单元）
	agent_demo_multiple_support.py	设置支援任务。获取到客户端的实体单元和情报信息，将客户端选择的信息加入到支援任务中。	实体单元（我方单元）、参考点（支援点）
	agent_demo_new_points.py	添加参考点。	——
	agent_demo_run_long_time.py	控制我方所有单元对于达到攻击条件的单位自动进行打击。首先设定控制时间，获取敌方的情报信息，根据目标类型，分配我方单元进行攻击，并将自动攻击指令下发。	——
	agent_demo_single_attack.py	控制我方选定单元攻击选定目标。获取来自客户端	实体单元（我

		的实体单元（攻击者），情报信息（被攻击者），下发手动打击指令	方单元）、情报信息（被攻击目标）
	agent_demo_single_retreat.py	控制单个目标脱离战斗。	实体单元（我方需要脱离战斗的单元）
态势分析	user_script.py	对当前态势进行分析，输出己方单元和敌方信息统计条形图和饼状图。	——

2.2 智能体开发指南

本小节以《萨哈林冲突》中红方一个具体打击任务为例，讲解“创建智能体开发工程——开发智能体——将智能体迁移到灵弈客户端——使用智能体”全流程的操作。

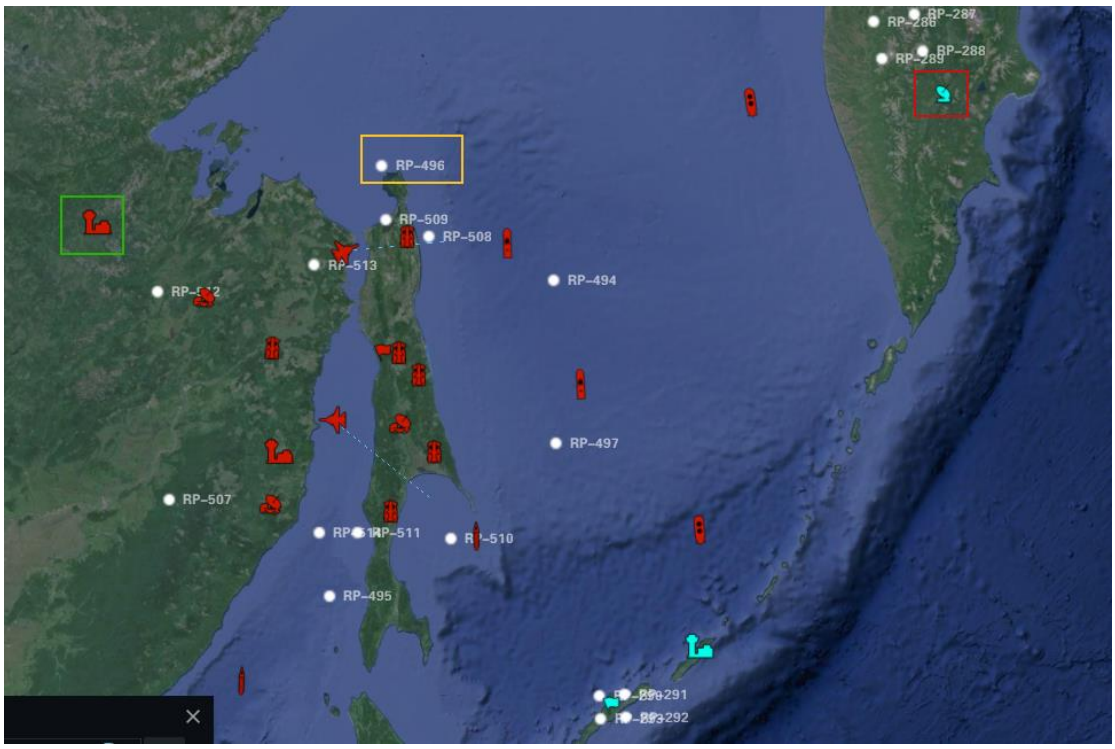


图 5 萨哈林冲突节选

2.2.1 作战任务分析

从红方视角分析，中部火力部署都在 AN/TPS-71 型移动式超视距雷达（图 5 红色方框标识处）的探测范围内，不利于红方作战，故在指挥作战阶段派具有对地打击能力的作战单元打击蓝方 AN/TPS-71 型移动式超视距雷达。

2.2.2 智能体功能设计

设计一个智能体，主要功能为从中部机场（图 5 绿色方框标识处）派处一架载有对地打击导弹的轰炸机，飞到合适地点后对蓝方 AN/TPS-71 型移动式超视距雷达进行打击。此智能体的工作流程图如图 6 所示：

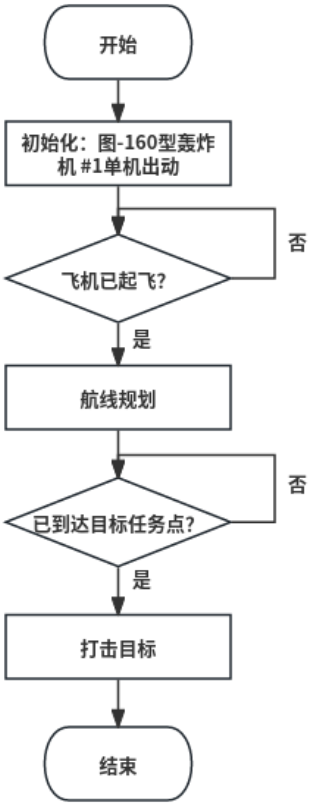


图 6 萨哈林冲突节选

2.2.3 智能体开发与调试

（1）启动灵弈客户端与灵弈服务器

- 步骤 1：进入灵弈服务器安装目录，依次双击启动 YLServer.exe 和 YLNode.exe

  （YLServer.exe 首次启动需要授权，请将弹出的灵弈服务器授权窗口机器码发送到邮箱 wokkkexin@163.com，专项赛期间每周三和周五将统一授权。）

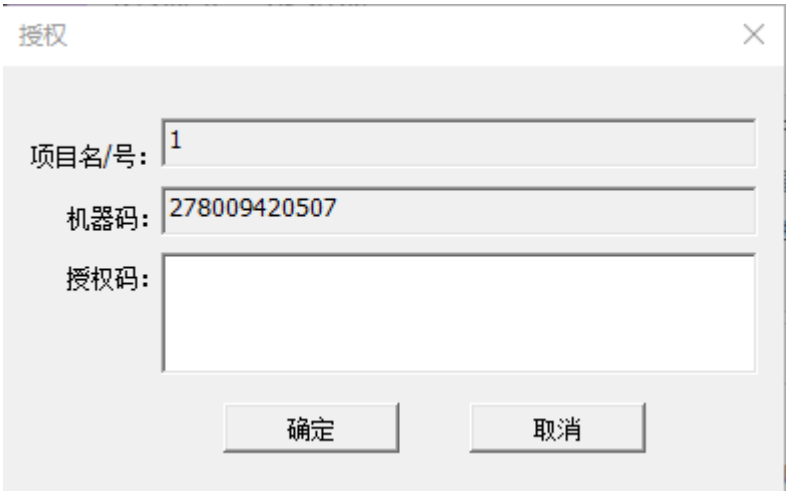


图 7 灵弈服务器授权窗口

- 步骤 2（非必须步骤）：进入灵弈客户端安装目录，双击启动 LYGame.exe，可以通过灵弈客户端查看仿真推演过程

（2）配置开发环境

- 步骤 1：用 PyCharm 打开 ai_train
- 步骤 2:依次点击 PyCharm 导航栏的 File->Settings->Python Interpreter 配置 LYGame/intelligent/environment/Python37 中的 Python 解释器，配置成功的界面如图 8 所示

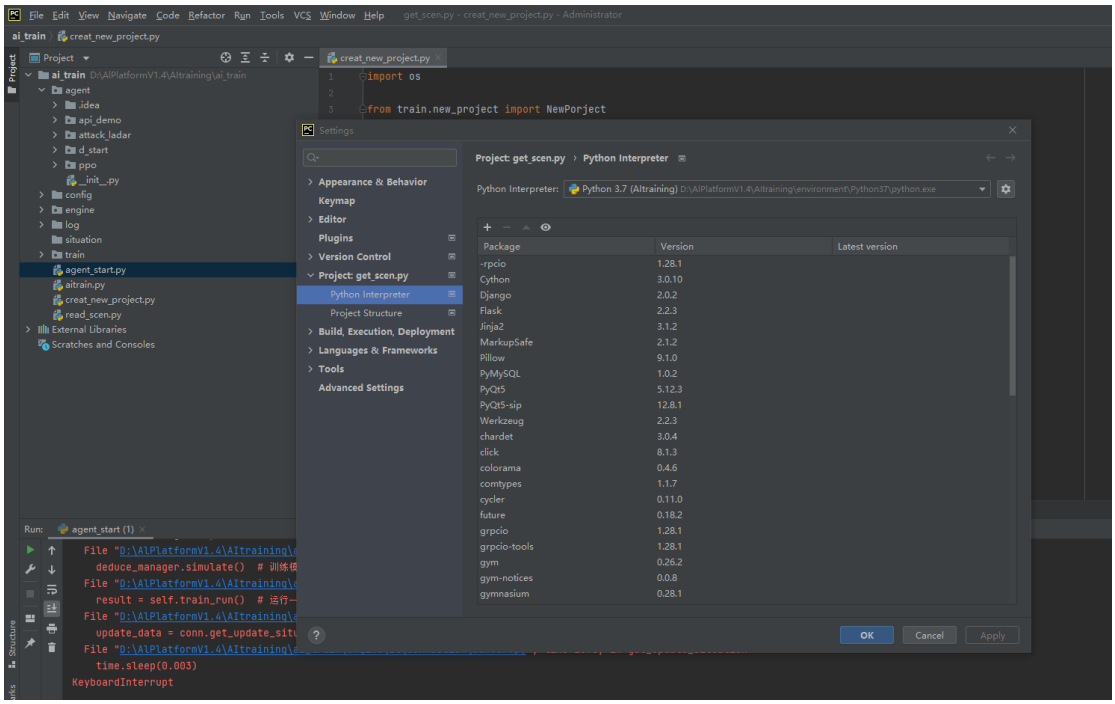


图 8 配置 Python 解释器

（3）创建智能体开发工程

- 步骤 1：工程配置，修改 creat_new_project.py 文件中的 project_config 信息，配置智能体工程名称（project_name）并指定智能体工程路径（project_path），注意都不要用中文

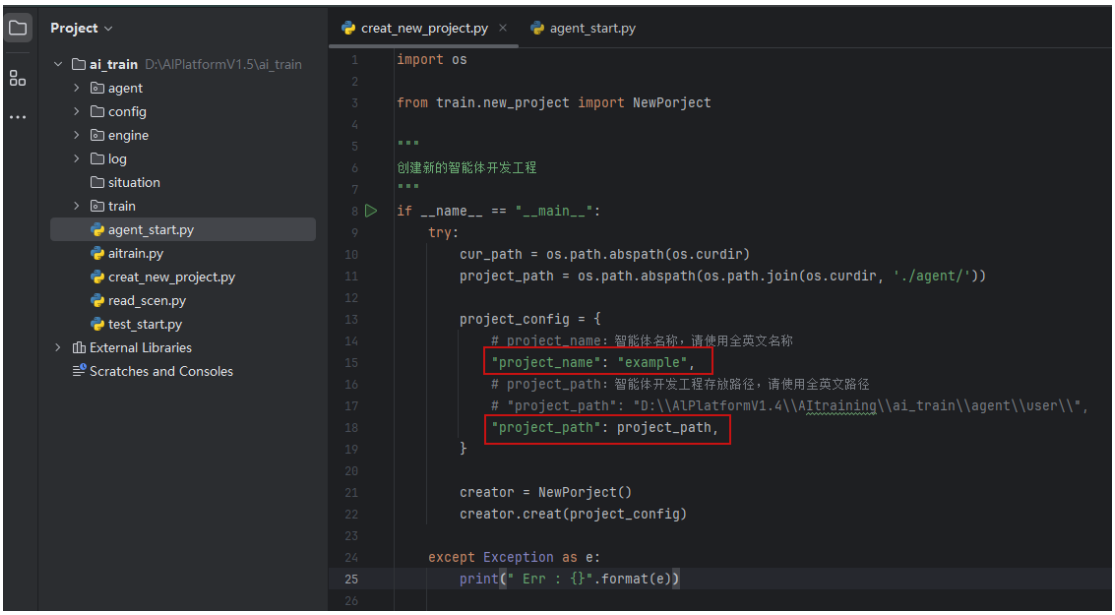


图 9 智能体开发模板工程配置文件

- 步骤 2：运行 creat_new_project.py，将在 project_path 所指定的路径下创建一个智能体开发模板工程，文件夹结构如图 10 所示，agent_example.py 为智能体入口模板文件，选手可以在此文件中实现自定义智能体。

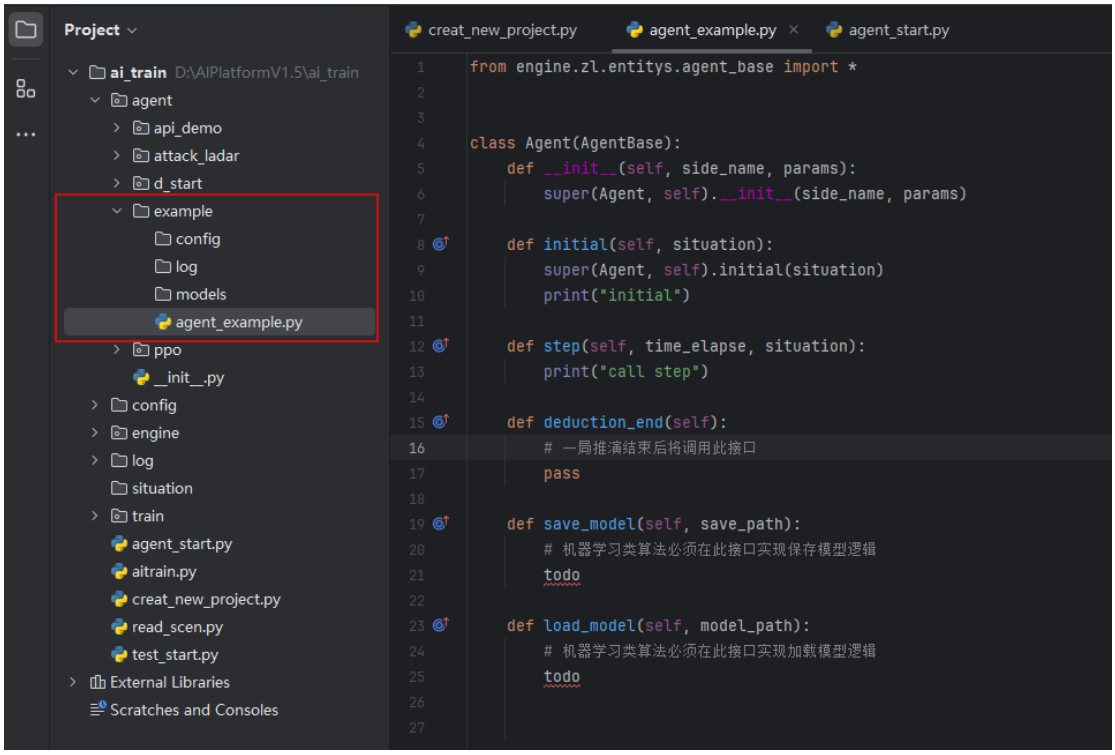


图 10 智能体开发模板工程

（4）开发智能体

agent_example.py 的内容如图 10 所示，Agent 类继承 AgentBase，AgentBase 封装了灵弈平台数据获取、指令下发等接口，接口功能描和使用详情可查阅附录 1 接口表。agent 提供了一些编码范例，下面将以 attack_ladar 对编码规范进行展开说明。

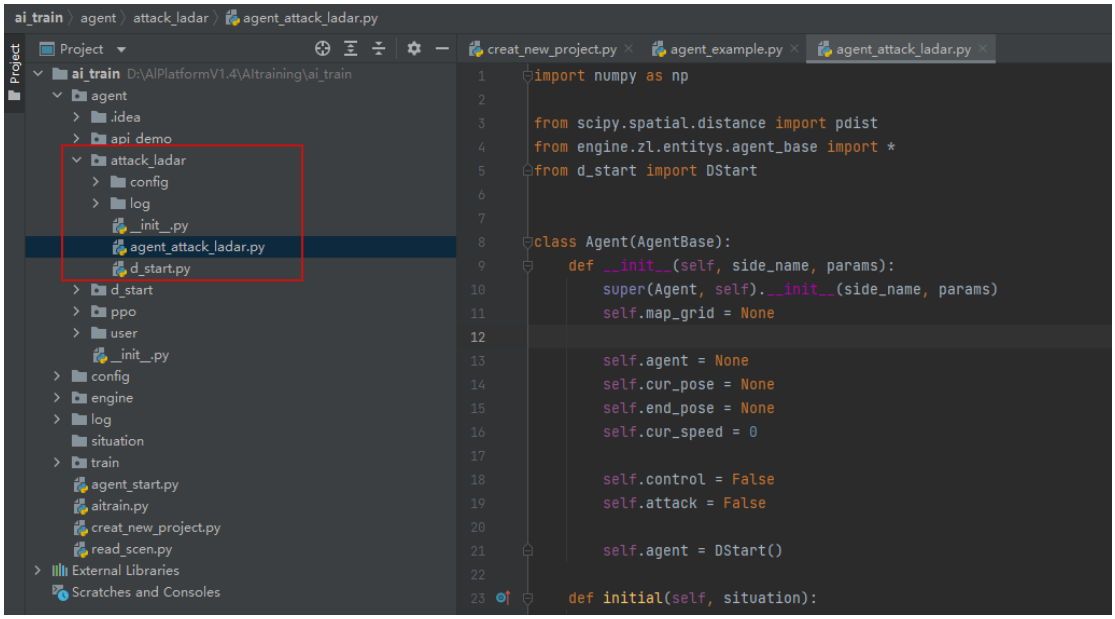


图 11 attack_ladar 工程

agent_attack_ladar 的功能为采用 D*算法对轰炸机进行路径规划，并调用 AgentBase 中封装的接口（详情见 3 **接口表**）获取输入信息、实时态势信息、下发轰炸机起飞和打击指令。工程结构如图 11 所示，agent_attack_ladar.py 为智能体类实现文件，d_start.py 为 D*算法实现文件。输入如下：

情报信息	AN/TPS-71 型移动式超视距雷达
参考点	RP-496
动作单元	图-160 型轰炸机 #1

- __init__函数：如图 12 所示__init__函数主要实现一些成员变量的定义

```python
import numpy as np

from scipy.spatial.distance import pdist
from engine.zl.entitys.agent_base import *
from d_start import DStart


class Agent(AgentBase):
    def __init__(self, side_name, params):
        super(Agent, self).__init__(side_name, params)
        self.map_grid = None

        self.agent = None
        self.cur_pose = None
        self.end_pose = None
        self.cur_speed = 0

        self.control = False
        self.attack = False

        self.agent = DStart()
```

图 12__init__函数

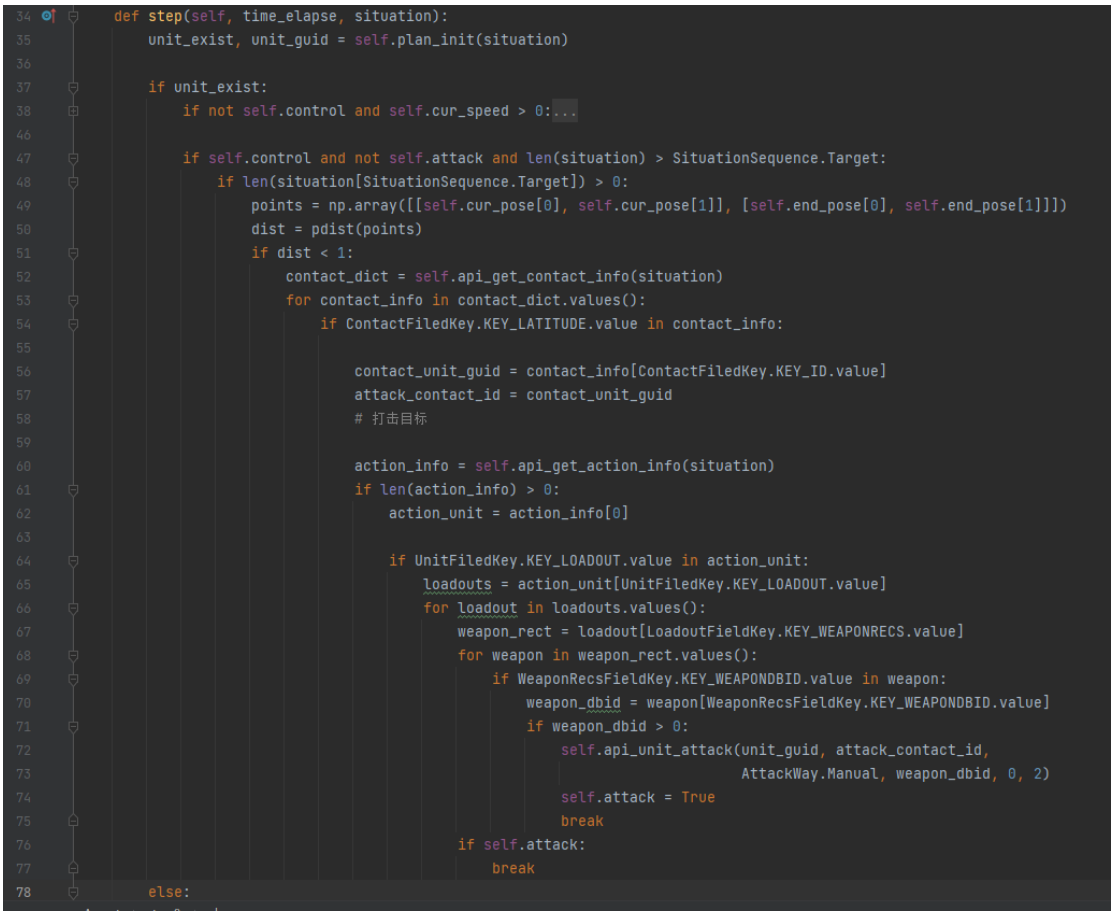
- Initial 函数：每局推演开始时调用一次 initial 函数，在 initial 函数中可以实现一些初始化操作，如第 28 行调用 api_common_out_and_return 接口下发轰炸机起飞并出动指令。

```python
    def initial(self, situation):
        super(Agent, self).initial(situation)
        unit_exist, unit_guid = self.plan_init(situation)

        if unit_exist:
            self.api_common_out_and_return(unit_guid)
        else:
            self.set_done()
            self.api_log(LogLevel.ERROR, "unit : ", "not exitst")
        self.api_log(LogLevel.INFO, "d_start_initial finish")
```

图 13initial 函数

- Step 函数：推演时会周期性调用 step 函数，在 step 函数实现一些实时态势分析和操作，第 35~51 行判断轰炸机的实时存活状态，并判断是否已经到了目的点，第 60~72 行在获取轰炸机的实时挂载状态，并获取可以用于打击的武器信息，第 73 行调用 api_unit_attack 下发攻击敌方雷达指令。



```
34    def step(self, time_elapse, situation):
35        unit_exist, unit_guid = self.plan_init(situation)
36
37        if unit_exist:
38            if not self.control and self.cur_speed > 0:...
46
47            if self.control and not self.attack and len(situation) > SituationSequence.Target:
48                if len(situation[SituationSequence.Target]) > 0:
49                    points = np.array([[self.cur_pose[0], self.cur_pose[1]], [self.end_pose[0], self.end_pose[1]]])
50                    dist = pdist(points)
51                    if dist < 1:
52                        contact_dict = self.api_get_contact_info(situation)
53                        for contact_info in contact_dict.values():
54                            if ContactFiledKey.KEY_LATITUDE.value in contact_info:
55
56                                contact_unit_guid = contact_info[ContactFiledKey.KEY_ID.value]
57                                attack_contact_id = contact_unit_guid
58                                # 打击目标
59
60                                action_info = self.api_get_action_info(situation)
61                                if len(action_info) > 0:
62                                    action_unit = action_info[0]
63
64                                    if UnitFiledKey.KEY_LOADOUT.value in action_unit:
65                                        loadouts = action_unit[UnitFiledKey.KEY_LOADOUT.value]
66                                        for loadout in loadouts.values():
67                                            weapon_rect = loadout[LoadoutFieldKey.KEY_WEAPONRECS.value]
68                                            for weapon in weapon_rect.values():
69                                                if WeaponRecsFieldKey.KEY_WEAPONDBID.value in weapon:
70                                                    weapon_dbid = weapon[WeaponRecsFieldKey.KEY_WEAPONDBID.value]
71                                                    if weapon_dbid > 0:
72                                                        self.api_unit_attack(unit_guid, attack_contact_id,
73                                                                             AttackWay.Manual, weapon_dbid, 0, 2)
74                                                        self.attack = True
75                                                        break
76                                            if self.attack:
77                                                break
78        else:
```

图 14 step 函数

（5）调试与运行智能体

根据实际调试和运行需求，修改 creat_new_project.py 文件中的 config_params 信息，以下步骤为常规调试需要修改的配置项。

- 步骤 1：配置仿真推演环境，配置参数如下：

配置项	说明	注意事项
ly_server_path	灵弈服务端路径	
scenario	想定文件	请确定 ly_server_path/Scenarios 文件夹下存在该文件
player_side	推演方	请填写想定中存在的推演方
room_name	房间名	
process_num	房间数量	默认值为 1，值大于 1 时会创建 process_num 个房间并行推演
compression	推演速度	为枚举值 0~5，含义如下： 0：1 倍速 1：5 倍速 2：10 倍速 3：20 倍速 4：60 倍速 5：最大推演速度

- 步骤 2：配置需要启动的智能体信息，配置参数如下：

配置项	说明	注意事项
agent_file_path	智能体入口文件路	

	径	
alg_type	智能体类型	0：非训练类智能体 1：训练类智能体
input_params	智能体输入参数	实体单元（unit）：配置推演方的单元名，在 Agent 中可通过接口 api_get_unit_info 获取单元详细信息 参考点（point）：在 Agent 中可通过接口 api_get_point_info 获取参考点位置信息 情报信息（contact）：配置推演方初始状态能获取到的情报信息名，在 Agent 中可通过接口 api_get_contact_info 获取情报详细信息 动作单元（action）：配置推演方需要执行任务动作的单元名，在 Agent 中可通过接口 api_get_action_info 获取动作单元详细信息 单元名称在灵弈客户端选中单元后按 Ctrl+C 获取单元信息，之后在文本编辑器中 Ctrl+V 可获得信息为：{Name:'雷达 (AN/TPS-71 型移动式超视距雷达)',DBID:'2517',guid:'77588a22-

		6369-4d4c-b1e1-75b6ee3c7daa'}

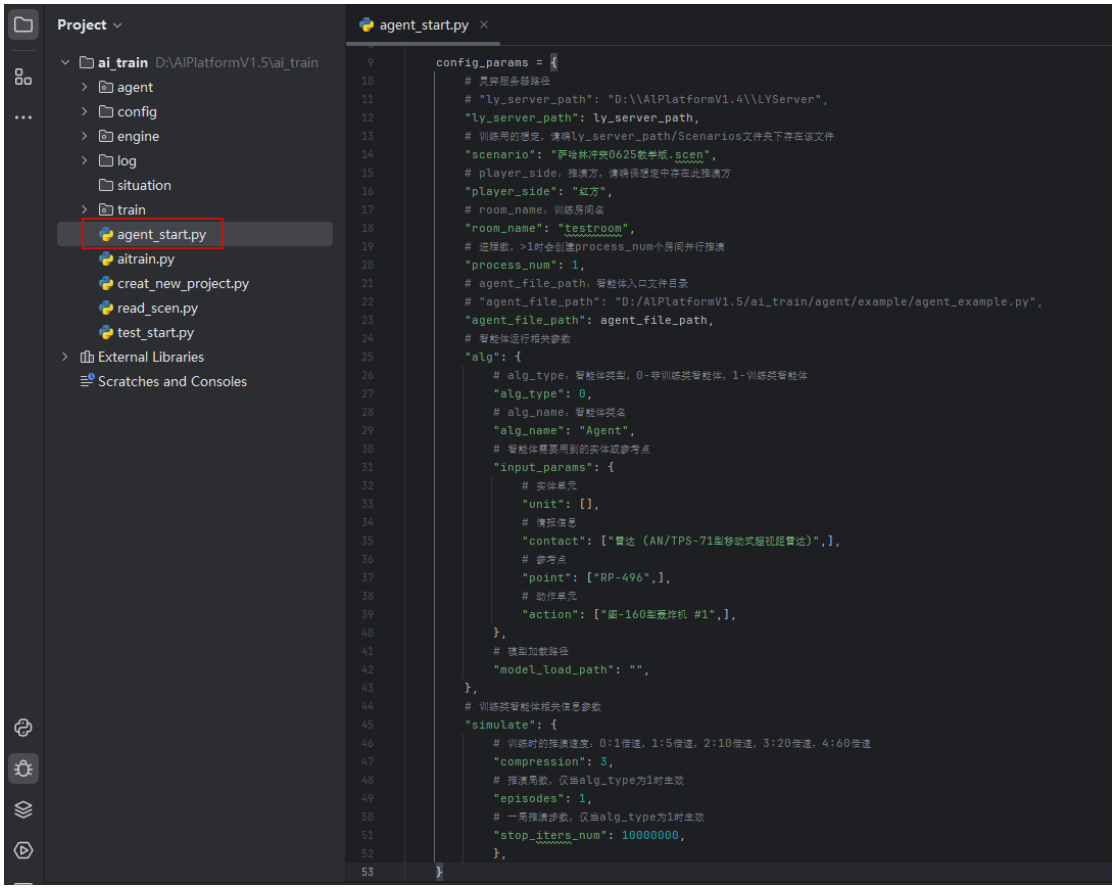


图 15 启动智能体配置项

- 步骤 3（训练类智能体需要配置此项）：训练类智能体需要连续仿真推演来达到训练与更新模型参数的目的，训练类智能体需要额外配置参数如下：

配置项	说明	注意事项
episodes	连续推演局数	
stop_iters_num	一局内 step 最大执行次数，当运行步数超过此设定值将主动结束本	

	局仿真推演	
model_load_path	初始化载模型的路径	在启动智能体首次执行过 initial 函数后调用一次 load_model，传入参数即为 model_load_path

生成的模型会默认存放在agent_file_path所在目录的model文件夹下

2.3 智能体使用指南

同样，智能体使用以 agent_attack_ladar 智能体为例进行介绍。

首先，将来自智能体开发框架的算法文件拷贝到客户端 LYGameLobby\LYGame\intelligent\ai\agent 路径下，以 attack_ladar 为例，我们将 attack_ladar 文件夹放置在 LYGameLobby\LYGame\intelligent\ai\agent\operation_optimization 路径下。打开灵弈客户端，进入推演界面，点击“智能体调度”选项，在屏幕右侧弹出 4 种智能体调度选项。

点击屏幕右侧“运筹优化”选项，点击“导入智能体文件”，选择所需要导入的智能体文件（以 agent_attack_lader.py 为例），点击确定，可以看到智能体被成功导入。值得注意的是，支持一个智能体算法导入多次，导入多次的智能体会根据导入顺序自动编号，且互相不影响运行。

图 16 运筹优化智能体管理界面

现在我们计划安排图-160 型轰炸机 #1 攻击蓝方 AN/TPS-71 雷达，当飞机到达参考点“RP-496”附近时进行导弹的发射。点击 agent_attack_lader 智能体中的“编辑”进行智能体输入信息的选择，在动作中选择图-160 型轰炸机 #1，在情报信息中选择 AN/TPS-71 雷达，并在参考点中选择“RP-496”，如图 17 所示。

图 17agent_attack_lader 智能体输入界面

图 18agent_attack_lader 智能体输入数据

在选择完成之后可以通过点击“查看控制的实体”来查看智能体输入信息。点击“启动”，智能体成功运行，可以看到图-160 型轰炸机#1 正在滑行起飞（图 19）。飞机起飞后，按照 agent_attack_lader 智能体算法中解算得到的航线进行飞行（图 20），并在到达参考点“RP-496”后进行攻击（图 21）。

图 19 智能体运行效果（飞机起飞）

图 20 智能体运行效果（按照航线进行飞行）

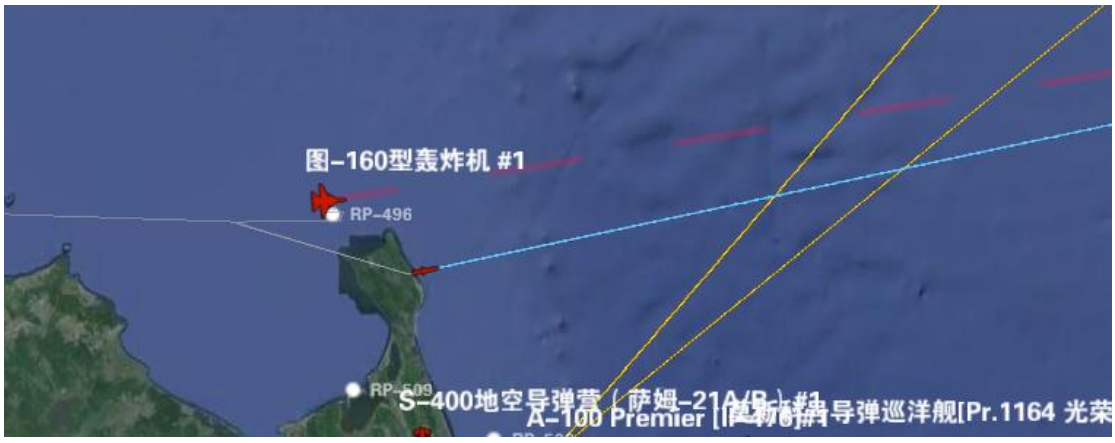


图 21 智能体运行效果（发射导弹攻击目标）

2.4 智能体编写规范

（1）文件命名规范

选手自定义脚本文件命名不得包含中文，需要使用英文字母、数字、下划线组合，满足 python 规范与标准。

（2）类的命名与继承规范

智能决策类名必须使用 Agent，且必须继承自 engine.zl.entitys.agent_base 接口文件中的 AgentBase 接口类。__init__方法需要调用 super(Agent, self).__init__(side_name, params)。

（3）虚函数接口重写

选手自定义智能体中必须重写 initial，step 函数，训练类智能体必须重写 load_model 和 save_model 函数。智能体推演框架启动时会创建 Agent 类对象，并在每局推演开始时调用一次 initial，之后周期性调用 step，在每局推演结束后调用一次 deduction_end。如果是 alg_type 为训练类智能体，在每局推演开始执行过 initial 函数后调用一次 load_model，在每局推演结束执行过 deduction_end 后调用一次 save_model。

3 接口表

类别	接口	功能
枚举定义	LogLevel	日志等级
	SonarType	声纳类型
	SituationSequence	态势信息列表元素含义
	InfoSequence	单元类型
	AttackWay	攻击方式
	UnitFiledKey	实体信息关键字
	ContactFiledKey	敌方实体信息关键字
	MountsFieldKey	载具信息关键字
	LoadoutFieldKey	挂载信息关键字
	WeaponRecsFieldKey	武器信息关键字
通用接口	api_log	日志打印接口
	api_get_unit_info	获取用戶选择的我方实体的态势信息
	api_get_point_info	获取用戶选择的参考点信息
	api_get_contact_info	获取用戶选择的敌方实体信息

	api_get_action_info	获取用戶选择的动作对象实体信息
	api_common_out_and_return	出动和返航
	api_common_contact	情报操作
	api_common_platform	平台操作
	api_common_mission	任务操作
	api_common_point	参考点类操作
	api_common_zone	区域类操作
	api_common_group	编组类操作
任务执行	api_deploy_strike_mission	下发打击任务指令
	api_deploy_patrol_mission	下发巡逻任务指令
	api_deploy_support_mission	下发支援任务指令
	api_deploy_ferry_mission	下发转场任务指令
	api_deploy_mine_mission	下发布雷任务指令
	api_deploy_mine_clear_mission	下发扫雷任务指令

	api_deploy_cargo_mission	下发运输任务指令
条令规则	api_doctrine_operation_default	设置默认条令规则
	api_doctrine_electromagnetism	设置电磁管控条令规则
	api_doctrine_weapon	设置武器状态条令规则
	api_doctrine_battle	设置接战条令规则
	api_doctrine_common	设置通用条令规则
	api_doctrine_plan_return	设置规划与返航条令规则
	api_doctrine_navigation_subsurface	设置反潜作战行动条令规则
	api_doctrine_withdraw	设置撤退条令规则
	api_doctrine_redeploy	设置重新部署条令规则
武器设置	api_weapon_rule	武器使用规则

单元操作	api_unit_common	推演方单元操作
	api_unit_switch	单元设置
	api_unit_control	单元航线、速度与高度控制
	api_unit_attack	单元打击操作控制
	api_unit_retreat	单元脱战
	api_unit_mission_and_base	任务与基地
	api_unit_operation_default	单元默认操作

3.1 枚举定义

3.1.1 LogLevel

枚举关键字	枚举值	描述
FATAL	logging.FATAL	致命错误
ERROR	logging.ERROR	错误
WARN	logging.WARN	警告
INFO	logging.INFO	普通信息
DEBUG	logging.DEBUG	调试信息

3.1.2 SonarType

枚举关键字	枚举值	描述
Passive	PASSIVE	被动

Active	ACTIVE	主动
Dipping	DIPPING	吊放

3.1.3 SituationSequence

枚举关键字	枚举值	描述
Own	0	我方信息
Target	1	敌方信息

3.1.4 InfoSequence

枚举关键字	枚举值	描述
Aircraft	0	飞机
Ship	1	舰船
Facility	2	设施
Submarine	3	潜艇
Satellite	4	卫星
Weapon	5	武器
Group	6	编组

3.1.5 AttackWay

枚举关键字	枚举值	描述
TargetInfo	0	
Auto	1	自动
Manual	2	手动

Bol	3	纯方位

3.1.6 UnitFiledKey

枚举关键字	枚举值	描述
KEY_ID	guid	
KEY_CATEGORY	category	
KEY_TYPE	type	
KEY_SENSORS	sensors	传感器
KEY_SPEED	speed	速度
KEY_LATITUDE	latitude	纬度
KEY_LONGITUDE	longitude	经度
KEY_ALTITUDE	altitude	高度
KEY_MOUNTS	mounts	
KEY_LOADOUT	loadout	装备配置

3.1.7 ContactFiledKey

枚举关键字	枚举值	描述
KEY_ID	ID	
KEY_NAME	Name	
KEY_SPEED	Speed	速度
KEY_HEAD	Head	
KEY_ALTITUDE	Alt	高度
KEY_PITCH	Pitch	

KEY_Roll	Roll	
KEY_LONGITUDE	Lon	经度
KEY_LATITUDE	Lat	纬度

3.1.8 MountsFieldKey

枚举关键字	枚举值	描述
KEY_ID	ID	
KEY_NAME	Name	
KEY_Status	Status	
KEY_DBID	DBID	

3.1.9 LoadoutFieldKey

枚举关键字	枚举值	描述
KEY_ID	ID	
KEY_DBID	DBID	
KEY_NAME	Name	
KEY_WEAPONRECS	WeaponRecs	
KEY_ROLE	Role	

3.1.10 WeaponRecsFieldKey

枚举关键字	枚举值	描述
KEY_ID	ID	
KEY_NAME	Name	

KEY_DEFAULT_LOAD	DefaultLoad	
KEY_CURRENT_LOAD	CurrentLoad	
KEY_MAX_LOAD	MaxLoad	
KEY_WEAPONDBID	WeaponDBID	

3.2 通用接口

3.2.1 api_log

定义	def api_log(self, lever, msg, *args, **kwargs)
参数	lever: LogLevel 枚举，日志等级 msg: string，日志信息
返回值	无

3.2.2 api_get_unit_info

定义	def api_get_unit_info(self, situation)
参数	situation: tuple（units，contacts），实时态势信息，units 为我方单元列表，contacts 为我方探测到的其他方目标字段
返回值	list: 与 input_params 匹配的我方单元详细信息列表

3.2.3 api_get_point_info

接口定义	def api_get_point_info(self)
参数	无

返回值	list：与 input_params 匹配的参考点详细信息列表

3.2.4 api_get_contact_info

接口定义	def api_get_contact_info(self, situation)
参数	situation：tuple（units，contacts），实时态势信息，units 为我方单元列表，contacts 为我方探测到的其他方目标字段
返回值	dict：与 input_params 匹配的情报信息详细字典

3.2.5 api_get_action_info

接口定义	def api_get_action_info(self, situation)
参数	situation：tuple（units，contacts），实时态势信息，units 为我方单元列表，contacts 为我方探测到的其他方目标字段
返回值	list：与 input_params 匹配的动作单元详细信息列表

3.2.6 api_common_out_and_return

接口定义	def api_common_out_and_return(self, unit_guid: str = str(), out_units_guid: List[str] = None,

	is_all_units_hold_position: bool = False, new_base_guid: str = str(), is_unit_return: bool = False)
参数	unit_guid: string, 我方实体单元 ID out_units_guid: list is_all_units_hold_position: bool new_base_guid: str is_unit_return: bool
返回值	无

3.2.7 api_common_contact

接口定义	def api_common_contact(self, contact_guid: str, new_name: str = str(), is_mark_position: bool = False, contact_stance: ContactStance = None, is_drop_target: bool = False)
参数	contact_guid: str,

	new_name: str is_mark_position: bool contact_stance: ContactStance is_drop_target: bool
返回值	无

3.2.8 api_common_platform

接口定义	def api_common_platform(self, unit_guid: str = None, unit_category_type: InfoSequence = None, is_get_unit: bool = False, is_get_unit_info: bool = False, is_get_units: bool = False, is_get_situation: bool = False, is_get_weather: bool = False, is_get_scenarios: bool = False, is_get_message_log: bool = False)
参数	unit_guid: str unit_category_type: InfoSequence 枚举

	is_get_unit: bool is_get_unit_info: bool is_get_units: bool is_get_situation: bool is_get_weather: bool is_get_scenarios: bool is_get_message_log: bool
返回值	无

3.2.9 api_common_mission

接口定义	def api_common_mission(self, mission_name: str, is_del_mission: bool = False, is_get_mission: bool = False, ass_unit_mission: str = str(), del_unit_from_mission: str = str(), point_list: list = None)
参数	mission_name: str is_del_mission: bool is_get_mission: bool

	ass_unit_mission: str del_unit_from_mission: str point_list: list
返回值	无

3.2.10 api_common_point

接口定义	def api_common_point(self, add_points: tuple or list = None, move_points_by_name: str = None, move_points_new_coordinate: tuple = None, del_point_by_name: str = None)
参数	add_points: tuple or list move_points_by_name: str move_points_new_coordinate: tuple del_point_by_name: str
返回值	无

3.2.11 api_common_zone

接口定义	def api_common_zone(self,

	add_zone_name: str = None, add_zone_points_list: list = None, add_zone_type: int = int(), set_zone_name: str = None, set_zone_new_name: str = None, set_zone_type: int = int(), set_zone_is_active: bool = True, set_zone_is_locked: bool = True, set_zone_mark_as: int = int(), remove_zone_type: int = None, remove_zone_guid: str = None)
参数	add_zone_name: str add_zone_points_list: list add_zone_type: int set_zone_name: str set_zone_new_name: str set_zone_type: int set_zone_is_active: bool set_zone_is_locked: bool set_zone_mark_as: int remove_zone_type: int

	remove_zone_guid: str
返回值	无

3.2.12 api_common_group

接口定义	def api_common_group(self, add_unit_guid_list: list = None, detach_unit_guid_list: list = None, remove_unit_guid: str = None, add_unit_tuple: tuple = None)
参数	add_unit_guid_list: list detach_unit_guid_list: list remove_unit_guid: str add_unit_tuple: tuple
返回值	无

3.3 任务执行

3.3.1 api_deploy_strike_mission

接口定义	def api_deploy_strike_mission(self, mission_name: str = "打击任务-XX",

	mission_type: MissionStrikeType = MissionStrikeType.LAND, target_list: list = None, start_time: str = "", end_time: str = "", ass_unit_id_list: list = None, un_ass_unit_id_list: list = None, remove_target_list: list = None, switch_radar: bool = True, is_active: bool = True, is_only_pre_plan=False, attack_condition: StrikeMinimumTrigger = StrikeMinimumTrigger.NotFriendly, strike_max: StrikeFlyTimeMax = StrikeFlyTimeMax.TWO, flight_size: FlightSize = FlightSize.Two, min_aircraft_req: MinAircraftReq = MinAircraftReq.ONE, radar_usage: StrikeRadarUsage = StrikeRadarUsage.ALL_PLAN,

	fuel_ammo: StrikeFuleAmmo = StrikeFuleAmmo.MOUNT_SET, min_dist: int = 100, max_dist: int = 1000, use_flight_size: bool = False, use_auto_planner: bool = True, one_time_only: bool = True, ship_size: FlightSize = FlightSize.Two, use_ship_size: bool = False, escort_flight_size: FlightSize = FlightSize.Two, min_shooter: FlightSize = FlightSize.One, max_shooter: FlightSize = FlightSize.Two, response_radius: int = 500)
参数	mission_name: str, 任务名 mission_type : MissionStrikeType , MissionStrikeType,打击任务类型,对空打击,对地打击等

	target_list: list, 设置打击目标 start_time: str, 设置、删除任务开始时间 end_time: str, 设置任务: 删除任务结束时间 ass_unit_id_list: list, 设置任务: 将实体分配到任务中来 un_ass_unit_id_list: list, 任务取消某单元的任务分配 remove_target_list: list, 设置任务: 删除打击任务目标 switch_radar: bool, 设置任务雷达是否打开 is_active: bool, 是否启用任务 is_only_pre_plan,设置任务细节:是否仅考虑计划目标（在目标清单） attack_condition:StrikeMinimumTrigger,设置打击任务触发条件 strike_max: StrikeFlyTimeMax, 设置任务细节: 任务允许出动的最大飞行批次 flight_size: FlightSize, 设置打击任务飞机编队规模 min_aircraft_req:MinAircraftReq,设置打击任务所需最少飞机数 radar_usage: StrikeRadarUsage, 设置打击任务雷达运用规则

	fuel_ammo: StrikeFuleAmmo，设置打击任务燃油弹药规则 min_dist: int，公里 max_dist: int，公里 use_flight_size: bool，设置打击任务是否飞机数低于编组规模数要求就不能起飞 use_auto_planner:bool,设置打击任务是否多扇面攻击（任务 AI 自动生成） one_time_only: bool，设置打击任务是否仅限一次 ship_size:FlightSize,设置打击任务舰艇/潜艇编队规模 use_ship_size:bool,设置打击任务是否强制舰艇/潜艇编队规模 escort_flight_size:FlightSize,设置打击任务护航飞机编队规模 min_shooter: FlightSize，设置打击护航任务所需最少飞机数 max_shooter: FlightSize，设置打击护航任务所需最大飞机数 response_radius: int，int,最大威胁响应半径海里
返回值	无

3.3.2 api_deploy_patrol_mission

接口定义	def api_deploy_patrol_mission(self, mission_name='巡逻任务-对空-XX', mission_type=MissionPatrolType.AIR, point_list: list = None, start_time: str = "", end_time: str = "", ass_unit_id_list: list = None, un_ass_unit_id_list: list = None, switch_radar: bool = True, is_active: bool = True, one_third_rule: bool = False, station_count: int = 6, check_OPA: bool = True, active_EMCON: bool = True, check_WWR: bool = True, flight_size: FlightSize = FlightSize.Two, use_flight_size: bool = False, throttle_transit: Throttle = Throttle.Unspecified,

	throttle_station: Throttle = Throttle.Unspecified, throttle_attack: Throttle = Throttle.Unspecified, transit_altitude: float = 1000, station_altitude: float = 5000, attack_altitude: float = 5000, attack_distance: int = 500, is_set_fuel_state_rtb: bool = True, is_set_weapon_state_rtb: bool = True,)
参数	mission_name: str 任务名 mission_type: MissionPatrolType 枚举, 任务类型, 对空打击, 对地打击等 point_list: list list, 参考点列表 start_time: str 设置、删除任务开始时间 end_time: str 设置任务: 删除任务结束时间 ass_unit_id_list: list 设置任务: 将实体分配到任务中来 un_ass_unit_id_list: list 任务取消某单元的任务分配 switch_radar: bool 设置任务雷达是否打开

	is_active: bool 是否启用任务 one_third_rule: bool 三分之一规则 station_count: int 为保持阵位的数量 check_OPA: bool 设置任务是否对巡逻区外的探测目标进行分析 active_EMCON: bool 设置任务是否仅在巡逻/警戒区内打开电磁辐射 check_WWR: bool 设置任务是否对武器射程内探测目标进行分析 flight_size: FlightSize 设置任务编队规模 use_flight_size: bool 是否飞机数低于编队规模不允许起飞 throttle_transit: Throttle 设置任务的出航油门 throttle_station: Throttle 设置任务的阵位油门 throttle_attack: Throttle 设置任务的攻击油门 transit_altitude: float, 出航高度, 单位: 米, 最多 6 位字符, 例: 99999.9, 888888 station_altitude: float float, 阵位高度, 单位: 米, 最多 6 位字符 attack_altitude: float float, 攻击高度, 单位: 米, 最多 6 位字符 attack_distance: int int, 攻击距离, 单位: 公里

	is_set_fuel_state_rtb: bool 燃油状态规划是否继承 is_set_weapon_state_rtb: bool 武器状态规划是否继承
返回值	无

3.3.3 api_deploy_support_mission

接口定义	def api_deploy_support_mission(self, mission_name='支援任务-XX', point_list: list = None, start_time: str = "", end_time: str = "", ass_unit_id_list: list = None, un_ass_unit_id_list: list = None, switch_radar: bool = True, is_active: bool = True, one_third_rule: bool = False, station_count: int = 6, one_time_only: bool = True, active_EMCON: bool = True, single_loop: bool = True, flight_size: FlightSize =

	FlightSize.Two, use_flight_size: bool = False, throttle_transit: Throttle = Throttle.Unspecified, throttle_station: Throttle = Throttle.Unspecified, transit_altitude: float = 1000, station_altitude: float = 5000)
参数	mission_name: str point_list: list start_time: str, 设置、删除任务开始时间 end_time: str, 设置任务: 删除任务结束时间 ass_unit_id_list: list, 设置任务: 将实体分配到任务中来 un_ass_unit_id_list: list, 任务取消某单元的任务分配 switch_radar: bool, 设置任务雷达是否打开 is_active: bool, 是否启用任务 one_third_rule: bool, 三分之一规则 station_count: int, int,保持阵位的数量 one_time_only: bool, 设置任务是否仅限一次

	active_EMCON: bool, 仅在阵位上打开电磁辐射 single_loop: bool, 导航类型 flight_size: FlightSize, 编队规模 use_flight_size: bool, 是否飞机数低于编队规模不允许起飞 throttle_transit: Throttle, 设置任务的出航油门 throttle_station: Throttle, 设置任务的阵位油门 transit_altitude: float, float,出航高度, 单位: 米, 最多 6 位字符, 例: 99999.9, 888888 station_altitude: float, float,阵位高度, 单位: 米, 最多 6 位字符
返回值	无

3.3.4 api_deploy_ferry_mission

接口定义	def api_deploy_ferry_mission(self, mission_name='转场任务-XX', destination: str = "", start_time: str = "", end_time: str = "", ass_unit_id_list: list = None, un_ass_unit_id_list: list = None,

	switch_radar: bool = True, is_active: bool = True, ferry_behavior=FerryBehavior.OneWay, flight_size: FlightSize = FlightSize.Two, use_flight_size: bool = False, min_aircraft_req: MinAircraftReq = MinAircraftReq.ONE, aircraft_throttle: Throttle = Throttle.Unspecified, ship_throttle: Throttle = Throttle.Unspecified, submarine_throttle: Throttle = Throttle.Unspecified, aircraft_altitude: float = 5000, submarine_altitude: float = 500, terrain_follow=False)
参数	mission_name: str, 任务名 destination: str, 目的地 guid start_time: str, 设置、删除任务开始时间 end_time: str, 设置任务: 删除任务结束时间

	ass_unit_id_list: list，设置任务：将实体分配到任务中来 un_ass_unit_id_list: list，任务取消某单元的任务分配 switch_radar: bool，设置任务雷达是否打开 is_active: bool，是否启用任务 ferry_behavior，设置转场规则 flight_size: FlightSize，设置任务编队规模 use_flight_size: bool，是否飞机数低于编队规模不允许起飞 min_aircraft_req:MinAircraftReq,设置所需最少飞机数 aircraft_throttle: Throttle，飞机转场油门 ship_throttle: Throttle，舰艇转场油门 submarine_throttle: Throttle，潜艇转场油门 aircraft_altitude: float，飞机转场高度,float submarine_altitude: float，潜艇转场深度,float terrain_follow: bool，设置转场地形跟随
返回值	无

3.3.5 api_deploy_mine_mission

接 口 定	def api_deploy_mine_mission(

义	self, mission_name="布雷任务-XX", point_list: list = None, time_delay: str = "", : start_time: str = "", : end_time: str = "", : ass_unit_id_list: list = None, : un_ass_unit_id_list: list = None, : switch_radar: bool = True, : is_active: bool = True, : one_third_rule: bool = False, : flight_size: FlightSize = FlightSize.Two, : use_flight_size: bool = False, : group_size: FlightSize = FlightSize.Two, : use_group_size: bool = False, : min_aircraft_req: MinAircraftReq = MinAircraftReq.TWO, : throttle_transit: Throttle = Throttle.Flank, : throttle_station: Throttle =

	Throttle.Flank, ： transit_altitude: float = 10000, ： station_altitude: float = 10000, ： is_submarine: bool = False, ： transit_terrain_following: bool = True, ： station_terrain_following: bool = True ：)
参数	mission_name: str, point_list: list, time_delay: str, str,exp: 以'天数: 时数: 分数: 秒数'表示, 如 1 天 4 小时 30 分钟表示为'1: 4: 30: 0' start_time: str, 设置、删除任务开始时间 end_time: str, 设置任务: 删除任务结束时间 ass_unit_id_list: list,设置任务:将实体分配到任务中来 un_ass_unit_id_list: list, 任务取消某单元的任务分配 switch_radar: bool, 设置任务雷达是否打开 is_active: bool, 是否启用任务 one_third_rule: bool, 三分之一规则

	flight_size: FlightSize, 设置任务编队规模 use_flight_size: bool, 是否飞机数低于编队规模不允许起飞 group_size: FlightSize, 设置任务舰船/潜艇编队规模 use_group_size: bool, 是否舰船/潜艇数低于编队规模不允许出发 min_aircraft_req:MinAircraftReq,设置所需最少飞机数 throttle_transit: Throttle, 设置任务的出航油门 throttle_station: Throttle, 设置任务的阵位油门 transit_altitude: float, float,出航高度, 单位: 米, 最多 6 位字符, 例: 99999.9, 888888 station_altitude: float, float,阵位高度, 单位: 米, 最多 6 位字符 is_submarine: bool, 是否出潜深度 transit_terrain_following: bool, 设置出航地形跟随 station_terrain_following: bool, 设置阵位地形跟随
返回值	无

3.3.6 api_deploy_mine_clear_mission

接口定义	def api_deploy_mine_clear_mission(self,

```
            mission_name='扫雷任务-XX',
            point_list: list = None,
            start_time: str = "",
            end_time: str = "",
            ass_unit_id_list: list = None,
            un_ass_unit_id_list: list = None,
            switch_radar: bool = True,
            is_active: bool = True,
            one_third_rule: bool = False,
            flight_size: FlightSize =
FlightSize.Two,
            use_flight_size: bool = False,
            group_size: FlightSize =
FlightSize.Two,
            use_group_size: bool = False,
            min_aircraft_req: MinAircraftReq =
MinAircraftReq.TWO,
            throttle_transit: Throttle =
Throttle.Flank,
            throttle_station: Throttle =
Throttle.Flank,
            unit_category: ElementType =
```

	ElementType.Aircraft, transit_altitude: float = 10000, station_altitude: float = 10000, is_submarine: bool = False, transit_terrain_following: bool = False, station_terrain_following: bool = False)
参数	mission_name: str point_list: list start_time: str, 设置、删除任务开始时间 end_time: str, 设置任务：删除任务结束时间 ass_unit_id_list: list, 设置任务：将实体分配到任务中来 un_ass_unit_id_list: list, 任务取消某单元的任务分配 switch_radar: bool, 设置任务雷达是否打开 is_active: bool, 是否启用任务 one_third_rule: bool, 三分之一规则 flight_size: FlightSize, 设置任务编队规模 use_flight_size: bool, 是否飞机数低于编队规模不允

	许起飞 group_size: FlightSize, 设置任务舰船/潜艇编队规模 use_group_size: bool, 是否舰船/潜艇数低于编队规模不允许出发 min_aircraft_req:MinAircraftReq,设置所需最少飞机数 throttle_transit: Throttle, 设置任务的出航油门 throttle_station: Throttle, 设置任务的阵位油门 unit_category: ElementType, 单元类型 transit_altitude: float, float,出航高度, 单位: 米, 最多 6 位字符, 例: 99999.9, 888888 station_altitude: float, float,阵位高度, 单位: 米, 最多 6 位字符 is_submarine: bool, bool,是否出潜深度 transit_terrain_following: bool, 设置出航地形跟随 station_terrain_following: bool, 设置阵位地形跟随
返回值	无

3.3.7 api_deploy_cargo_mission

接口定义	def api_deploy_cargo_mission(self, mission_name='货运任务-XX',

	point_list: list = None, start_time: str = "", end_time: str = "", ass_unit_id_list: list = None, un_ass_unit_id_list: list = None, switch_radar: bool = True, is_active: bool = True, throttle_transit: Throttle = Throttle.Flank, throttle_station: Throttle = Throttle.Flank, unit_category: ElementType = ElementType.Aircraft, transit_altitude: float = 10000, station_altitude: float = 10000)
参数	mission_name='货运任务-XX' point_list: list start_time: str, 设置、删除任务开始时间 end_time: str, 设置任务: 删除任务结束时间 ass_unit_id_list: list, 设置任务: 将实体分配到任务中来

	un_ass_unit_id_list: list，任务取消某单元的任务分配 switch_radar: bool，设置任务雷达是否打开 is_active: bool，是否启用任务 throttle_transit: Throttle，设置任务的出航油门 throttle_station: Throttle，设置任务的阵位油门 unit_category: ElementType 枚举 transit_altitude: float, station_altitude: float，float,阵位高度，单位：米，最多 6 位字符
返回值	无

3.4 条令规则

3.4.1 api_doctrine_operation_default

接口定义	def api_doctrine_operation_default(self, mission_guid: str = None, unit_guid: str = None)
参数	mission_guid: str unit_guid: str
返回值	无

3.4.2 api_doctrine_electromagnetism

接	def api_doctrine_electromagnetism(

口定义	self, switch_radar: bool = True, switch_OECM: bool = True, switch_sonar: bool = True, set_EMCON_inherit: bool = True, ignore_EMCON_under_attack: IgnoreEMCONUnderAttack = IgnoreEMCONUnderAttack.Ignore_EMCON_While_Under_Attack, mission_guid: str = None, unit_guid: str = None,)
参数	switch_radar: bool，条令中，电磁管控设置，设置雷达开关机 switch_OECM: bool，条令中，电磁管控设置，设置电子干扰传感器开关机 switch_sonar: bool，条令中，电磁管控设置，设置声呐开关机 set_EMCON_inherit: bool，设置电磁管控是否与上级一致 ignore_EMCON_under_attack: IgnoreEMCONUnderAttack，受到攻击忽略电磁管控

	mission_guid: str, 二选一设置,mission 优先级高,都不设置则默认为整体条令
返回值	无

3.4.3 api_doctrine_weapon

接口定义	def api_doctrine_weapon(self, to_air: WeaponControlStatus = WeaponControlStatus.Free, to_surface: WeaponControlStatus = WeaponControlStatus.Free, to_subsurface: WeaponControlStatus = WeaponControlStatus.Free, to_land: WeaponControlStatus = WeaponControlStatus.Free, mission_guid: str = None, unit_guid: str = None,)
参数	to_air: WeaponControlStatus, 武器控制状态, 对空 to_surface: WeaponControlStatus, 武器控制状态, 对

	水面 to_subsurface: WeaponControlStatus, 武器控制状态, 对潜 to_land: WeaponControlStatus, 武器控制状态, 对地 mission_guid: str, 二选一设置,mission 优先级高,都不设置则默认为整体条令
返回值	无

3.4.4 api_doctrine_battle

接口定义	def api_doctrine_battle(self, ignore_plotted_course: IgnorePlottedCourseWhenAttacking = IgnorePlottedCourseWhenAttacking.Yes, engage_opportunity_targets: EngageWithContactTarget = EngageWithContactTarget.Yes_AnyTarget, engaging_ambiguous_targets: BehaviorTowardsAmbigousTarget =

	BehaviorTowardsAmbigousTarget.Optimistic, mission_guid: str = None, unit_guid: str = None,)
参数	ignore_plotted_course : IgnorePlottedCourseWhenAttacking，攻击时忽略计划航线设置 engage_opportunity_targets : EngageWithContactTarget，接战临机出现目标 engaging_ambiguous_targets : BehaviorTowardsAmbigousTarget，接战模糊位置目标 mission_guid: str，二选一设置,mission 优先级高,都不设置则默认为整体条令
返回值	无

3.4.5 api_doctrine_common

接口定义	def api_doctrine_common(self, nuclear_weapons: EUseNuclear = EUseNuclear.Yes, automatic_evasion: AutomaticEvasion = AutomaticEvasion.Yes,

	air_operations_tempo: AirOpsTempo = AirOpsTempo.Surge, gun_strafing: GunStrafeGroundTargets = GunStrafeGroundTargets.Yes, kinematic_range: UseTorpedoesKinematicRange = UseTorpedoesKinematicRange.Yes, mission_guid: str = None, unit_guid: str = None,)
参数	nuclear_weapons: EUseNuclear，条令设置，是否使用核武器 automatic_evasion: AutomaticEvasion，自动规避 air_operations_tempo: AirOpsTempo，空战节奏 gun_strafing: GunStrafeGroundTargets，空对地扫射 kinematic_range: UseTorpedoesKinematicRange，鱼雷射程 mission_guid: str，二选一设置,mission 优先级高,都不设置则默认为整体条令
返回值	无

3.4.6 api_doctrine_plan_return

接口定义	def api_doctrine_plan_return(self, refuel: ERefuel = ERefuel.Yes, refuel_select: ERefuelSelect = ERefuelSelect.Include_NoBack, refuel_allied: ERefuelAllied = ERefuelAllied.YesReceiveOnly, fuel_state_planned: FuelState = FuelState.Joker10Percent, fuel_state_return: FuelStateRTB = FuelStateRTB.No, weapon_state_planned: WeaponStatePlanned = WeaponStatePlanned.LoadSet, weapon_state_return: WeaponStateRTB = WeaponStateRTB.No, mission_guid: str = None, unit_guid: str = None,)
参数	refuel:ERefuel, 加油/途中补给 refuel_select:ERefuelSelect, 加油/途中补给 refuel_allied:ERefuelAllied, 给盟军加油/途中补给

	fuel_state_planned:FuelState，燃油状态，预先规划 fuel_state_return:FuelStateRTB，燃油状态，返航 weapon_state_planned:WeaponStatePlanned，武器状态，预先规划 weapon_state_return:WeaponStateRTB，武器状态-返航 mission_guid:str,二选一设置,mission 优先级高,都不设置则默认为整体条令
返回值	无

3.4.7 api_doctrine_navigation_subsurface

接口定义	def api_doctrine_navigation_subsurface(self, mission_guid: str = None, unit_guid: str = None, avoid_contact: AvoidContact = AvoidContact.YesAlways, dive_on_threat: DiveOnThreat = DiveOnThreat.YesRadar, recharge_on_patrol: RechargePercent = RechargePercent.Recharge50Percent, recharge_on_attack: RechargePercent = RechargePercent.Recharge50Percent,

	use_aip: UseAIP = UseAIP.YesAlways, dipping_sonar: DippingSonar = DippingSonar.Automatically, navigation_sub: NavigationSub = NavigationSub.Attack)
参数	mission_guid:str,二选一设置,mission 优先级高,都不设置则默认为整体条令 avoid_contact:AvoidContact,反潜作战行动,规避搜索 dive_on_threat:DiveOnThreat, 反潜作战行动, 探测到威胁进行下潜 recharge_on_patrol:RechargePercent, 反潜作战行动, 充电电池运输/站点 recharge_on_attack:RechargePercent, 反潜作战行动, 充电电池进攻/防守 use_aip:UseAIP, 反潜作战行动, 使用 AIP 技术 dipping_sonar:DippingSonar,反潜作战行动,吊放声呐 navigation_sub:NavigationSub, 反潜作战行动, 导航
返回值	无

3.4.8 api_doctrine_withdraw

接 口 定	def api_doctrine_withdraw(

义	self, mission_guid: str = None, unit_guid: str = None, damage: DamageThreshold = DamageThreshold.Ignore, fuel: FuelQuantityThreshold = FuelQuantityThreshold.Ignore, attack_weapon: WeaponQuantityThreshold = WeaponQuantityThreshold.Ignore, defence_weapon: WeaponQuantityThreshold = WeaponQuantityThreshold.Ignore)
参数	mission_guid:str,二选一设置,mission 优先级高,都不设置则默认为整体条令 damage:DamageThreshold, 满足如下条件时撤退--毁伤程度大于 fuel:FuelQuantityThreshold, 满足如下条件时撤退--燃油少于 attack_weapon:WeaponQuantityThreshold,满足如下条件时撤退--主要攻击攻击武器至少处于

	defence_weapon:WeaponQuantityThreshold，满足如下条件时撤退--主要防御武器至少
返回值	无

3.4.9 api_doctrine_redeploy

接口定义	def api_doctrine_redeploy(self, mission_guid: str = None, unit_guid: str = None, damage: DamageThreshold = DamageThreshold.Ignore, fuel: FuelQuantityThreshold = FuelQuantityThreshold.Ignore, attack_weapon: WeaponQuantityThreshold = WeaponQuantityThreshold.Ignore, defence_weapon: WeaponQuantityThreshold = WeaponQuantityThreshold.Ignore)
参数	mission_guid:str,二选一设置,mission 优先级高,都不设置则默认为整体条令

	damage:DamageThreshold，满足如下条件时重新部署--毁伤程度小于 fuel:FuelQuantityThreshold，满足如下条件时重新部署--燃油至少处于 attack_weapon:WeaponQuantityThreshold，满足如下条件时重新部署--主要攻击武器处于 defence_weapon:WeaponQuantityThreshold，满足如下条件时重新部署--主要防御武器处于
返回值	无

3.5 武器设置

3.5.1 api_weapon_rule

接口定义	def api_weapon_rule(self, weapon_dbid: str = "", target_type: WRA_WeaponTargetType = WRA_WeaponTargetType.Aircraft_Unspecified, quantity_salvo: str = "max", shooter_salvo: str = "max", firing_range: str = 'max', self_defense: str = 'max', escort: str = 'false',

	mission_guid: str = None, unit_guid: str = None,)
参数	weapon_dbid:str，武器的数据库 ID target_type:WRA_WeaponTargetType ， 目 标 类 型 号}WRA_WeaponTargetType 的值，例如：飞机未指定："2000" quantity_salvo:str， 'n'-齐射武器数,'inherit'-继承设置,'max'-全量齐射,'none'-禁用 shooter_salvo:str，'n'-齐射发射架数,'inherit'-继承设置,'max'-全量齐射 firing_range:str， 'n'-自动开火距离，'inherit'-继承设置,'none'-禁用自动开火 self_defense:str， 'n'-自动防御距离,'inherit'-继承设置,'max'-最大射程射击,'none'-禁用自卫 escort:str，是否护航任务'true'-护航任务,'false'-非护航任务 mission_guid:str,二选一设置,mission 优先级高,都不设置则默认为整体条令
返回值	无

3.6 单元操作

3.6.1 api_unit_common

接口定义	def api_unit_common(self, unit_guid: str = str(), new_name: str = str(), contact_guid: str = str(), units_contacts: tuple or list = None)
参数	unit_guid:str, 单元 ID new_name:str, 新名称 contact_guid:str, 目标 ID units_contacts:tuple or list
返回值	

3.6.2 api_unit_switch

接口定义	def api_unit_switch(self, unit_guid: str, emcon_abide: bool = True, radar: bool = True, sonar: bool = True,

	oecm: bool = True, follow_terrain: bool = True, altitude_manual: bool = True, hold_position: bool = True, sonar_type: SonarType = SonarType.Active, is_shallow: bool = True, re_fuel_unit_guid: str = str())
参数	unit_guid:str, emcon_abide:bool radar:bool sonar:bool oecm:bool follow_terrain:bool altitude_manual:bool hold_position:bool sonar_type:SonarType is_shallow:bool re_fuel_unit_guid:str
返回值	无

3.6.3 api_unit_control

接口定义	def api_unit_control(self, unit_guid: str, course_list: list = None, throttle: Throttle = None, desired_speed: float = 0, desired_height: float = 0)
参数	unit_guid:str, course_list:list throttle:Throttle desired_speed:float desired_height:float
返回值	无

3.6.4 api_unit_attack

接口定义	def api_unit_attack(self, unit_guid: str, contact_guid: str = '', attack_way: AttackWay =

	AttackWay.Auto, weapon_dbid: int = int(), mount_dbid: int = int(), qty_num: int = int(), bol_coordinate: (float, float) = (float(), float()), course: list = None)
参数	unit_guid:str contact_guid:str attack_way:AttackWay weapon_dbid:int mount_dbid:int qty_num:int bol_coordinate:(float,float) course:list
返回值	无

3.6.5 api_unit_retreat

接口定义	def api_unit_retreat(self, unit_guid: str,

	abandon_attack_guid: str = str(), abandon_all_attack: bool = False, abandon_all_mission: bool = False, clear_all_course: bool = False, is_return: bool = False)
参数	unit_guid:str, abandon_attack_guid:str abandon_all_attack:bool abandon_all_mission:bool clear_all_course:bool is_return:bool
返回值	无

3.6.6 api_unit_mission_and_base

接口定义	def api_unit_mission_and_base(self, unit_guid, new_base_guid: str = str(), ass_mission: str = str(), ass_escort_mission: str = str())

参数	unit_guid, new_base_guid:str ass_mission:str ass_escort_mission:str
返回值	无

3.6.7 api_unit_operation_default

接口定义	def api_unit_operation_default(self, unit_guid="", contact_guid="", ass_mission="", units_contacts=(), course_list: list = None, mount_id="", weapon_id="", qty_num=1, BOL_coordinate=(), unattack_all=False, unattack_contact_guid="", unass_mission="", new_base_guid="",

	new_name="")
参数	
返回值	无

4 态势格式说明

在Agent中,situation为step和initial方法的输入,situation包含了我方单元信息和探测到敌方的单元信息，situation具体格式为：

```
[
我方态势信息;
敌方态势信息
]
```

1) 我方态势信息

- 类型：list
- 格式：

```
[
{飞机1精简信息},
···
{飞机n精简信息},
{设施1精简信息},
···
{设施n精简信息},
{舰船1精简信息},
···
{舰船n精简信息},
{潜艇1精简信息},
```

```
• • •
{潜艇 n 精简信息},
{卫星 1 精简信息},
• • •
{卫星 n 精简信息},
]
```

2) 敌方态势信息：

- 类型：dict
- 格式：

```
{
{飞机 1 精简信息},
• • •
{飞机 n 精简信息},
{设施 1 精简信息},
• • •
{设施 n 精简信息},
{舰船 1 精简信息},
• • •
{舰船 n 精简信息},
{潜艇 1 精简信息},
```

```
• • •
{空中#1},
• • •
{空中#n},


}
```

3) 实体单元信息：

实体单元包括：飞机、设施、卫星、舰船、潜艇等，单元信息中主要包括单元 id、经纬度、速度、高度等关键信息。本方单元（除卫星）一般含有的属性为：

```
{
"category": "Aircraft", # 单元类型（飞机、设施、卫星、舰船、潜艇等）
"guid": str, # 单元 id
"dbid": int, # 单元数据库 id
"name": str, # 名称
"mission": str, # 任务
"latitude": int,  # 纬度
"longitude": int,  # 经度
"altitude": int,  # 高度
 "course": [],  # 航线
```

```
"heading": int,  # 朝向
"speed": int,  # 速度（节）
"fuel": dict,  # 燃油
"damage": dict,  # 单元摧毁度
"status": dict, # 单元状态（例如飞机停放，或处于飞行状态）
 "sensors": dict,  # 传感器信息
"mounts": dict, 挂架
"loadout": dict, 挂载
}
```

- 本方单元 type 为实体类型，所有类型如下表:

Aircraft	Ship	Facility	Submarine	Satellite	Weapon	Group
飞机	舰船	设施	潜艇	卫星	武器	编队

- 飞机状态，总结的 4 种运行状态，为 AirValidStatus 类型，如下表:

validToFly	InAir	InAirRTB	WaitReady
在基地可马上部署飞行 任务	在空中可规划任务	在空中返航或降落	在基地等待准备好

- 情报 type 为探测或接收到的情报目标类型。类型为整数，含义和描述如下:

空中目标-0	轨道目标-6	不确定-12	军事设施-18

Air	Orbital	Undetermined	Installation
导弹-1	固定设施-7	空中诱饵-13	空军基地-19
Missile	Facility_Fixed	Decoy_Air	AirBase
水面/地面-2	移动设施-8	表面诱饵-14	海军基地-20
Surface	Facility_Mobile	Decoy_Surface	NavalBase
潜艇-3	鱼雷-9	陆地诱饵-15	移动集群-21
Submarine	Torpedo	Decoy_Land	MobileGroup
未确定的海军-4	水雷-10	水下诱饵-16	激活点瞄准点-22
UndeterminedNaval	Mine	Decoy_Sub	ActivationPoint
瞄准点-5	爆炸-11	声纳浮标-17	
Aimpoint	Explosion	Sonobuoy	