臻融DDS性能测试工具架构设计文档

版本：V2.4.4

日期：2025.8

编写人：段嘉文 肖飞鸿 杨熙锐 王佳琪 邹欣纯

# 一、配置解析模块

## 1.1文件加载

负责文件的读取

首先使用文件系统API检查文件是否存在，接着打开文件流进行读取，最后将文件内容完整读入内存中的字符串，如果任何步骤失败，则返回错误信息而不是继续执行。

·输入：配置文件的路径

·输出：文件的内容（字符串）或错误信息

## 1.2JSON解析

将JSON文本转化为结构化的配置数据

使用JSON库（如nlohmann/json）解析字符串，然后逐个检查并提取预定义的标准字段（domain\_id、topic\_name等），再将其余未知字段按类型分类存储到扩展参数map中

·输入：JSON格式的字符串

·输出：填充好的TestConfig结构体对象或者解析失败的错误信息

## 1.3配置验证

保证配置数据的完整性和正确性

检查必需字段是否存在，验证字段值的合理性（如数值必须为正数、角色只能是publisher或subscriber），验证外部文件路径是否存在（如果配置中指定了QoS XML文件或IDL文件）

·输入：初步解析后的TestConfig对象

·输出：验证结果

## 1.4配置访问

提供安全便捷的配置数据访问接口

为每个配置字段提供带默认值的访问方法，并处理optional字段的访问，避免空值异常，同时提供扩展参数的查询接口，提供字段存在性检查功能

·输入：验证通过的TestConfig对象 + 具体的访问请求

·输出：对应的配置值

# 二.DDS实体管理模块

## 2.1模块概述

该模块负责根据上层配置解析模块提供的参数，动态创建、配置和管理ZRDDS通信所需的核心实体（如DomainParticipant, Topic, DataWriter, DataReader），并为性能数据采集模块提供必要的接口和触发机制。根据JSON配置文件中的参数，动态创建所需的DDS实体。

再通过加载XML QoS配置文件，将预定义的QoS策略应用到创建的实体上。优先使用ZRDDS提供的扩展接口构建模块，降低代码复杂度。

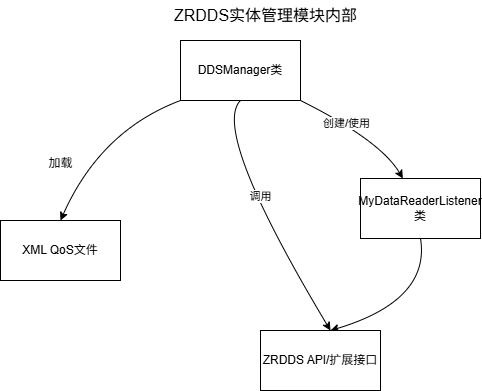
·资源管理: 负责创建实体的生命周期管理，包括初始化和清理。

·触发机制: 为Subscriber端提供数据到达的监听和通知机制，连接至性能数据采集模块。

## 2.2模块关系

该模块依赖配置解析模块提供JSON解析后的配置参数（域ID, 主题名, 类型名, 角色, QoS Profile名称, XML路径等），性能数据测试模块通过该模块的DataReader Listener的回调接收数据到达通知，启动数据采集。

## 2.3模块结构图



## 2.4核心类设计

### 2.4.1 DDSManager 类

·职责: 核心管理类，负责整个DDS实体的生命周期。

·关键属性:

factory\_: DDS::DomainParticipantFactory\* 指针。

participant\_: DDS::DomainParticipant\* 指针。

topic\_: DDS::Topic\* 指针。

data\_writer\_: DDS::DataWriter\* 指针 (Publisher)。

data\_reader\_: DDS::DataReader\* 指针 (Subscriber)。

listener\_: MyDataReaderListener\* 指针 (Subscriber)。

·配置参数: 存储从配置解析模块接收的参数副本。

is\_initialized\_: bool 标记初始化状态。

·关键方法:

DDSManager(...): 构造函数，接收配置参数。

~DDSManager(): 析构函数，确保资源清理。

bool initialize(): 核心初始化方法，执行实体创建流程。

void shutdown(): 核心清理方法，删除实体并回收资源。

get\_participant(), get\_data\_writer(), get\_data\_reader(): 提供对创建实体的访问。

bool is\_initialized(): 查询初始化状态。

### 2.4.2 MyDataReaderListener 类

·职责：实现DDS：SimpleDataReaderlistener处理数据到达事件。

·关键属性:

callback\_: OnDataAvailableCallback (函数指针/std::function)，指向性能数据采集模块的处理函数。

·关键方法:

on\_process\_sample(...) (来自 SimpleDataReaderListener) 或on\_data\_available(...) (来自 DataReaderListener): 当数据可用时被ZRDDS调用。在此方法中，记录时间戳并调用 callback\_。

## 2.5 接口设计

### 2.5.1提供给其他模块的接口 (由DDSManager提供)

·构造函数:接收所有必要的配置参数。

DDSManager(const std::string& qos\_xml\_path, DDS::DomainId\_t domain\_id, const std::string& topic\_name, const std::string& type\_name, const std::string& role, const std::string& participant\_qos\_name, const std::string& data\_writer\_qos\_name, const std::string& data\_reader\_qos\_name, OnDataAvailableCallback callback);

·初始化接口:执行DDS实体创建流程。成功返回true，失败返回false。

bool initialize();

·清理接口:删除所有已创建的DDS实体，回收资源。

void shutdown();

·实体访问接口:

DDS::DomainParticipant\* get\_participant() const;

DDS::DataWriter\* get\_data\_writer() const; (仅Publisher有效)

DDS::DataReader\* get\_data\_reader() const; (仅Subscriber有效)

·状态查询接口:返回模块当前是否已成功初始化

bool is\_initialized() const;

### 2.5.2 依赖的外部接口 (由其他模块提供或ZRDDS库提供)

·配置解析模块:提供包含所有必要参数的结构体或变量。

·ZRDDS库接口:

DDS::DomainParticipantFactory::get\_instance()

DDS::DomainParticipantFactory::reload\_qos\_profiles()

DDS::DomainParticipantFactory::create\_participant\_with\_profile()

DDS::DomainParticipant::create\_topic()

DDS::DomainParticipant::create\_datawriter\_with\_topic\_and\_qos\_profile()

DDS::DomainParticipant::create\_datareader\_with\_topic\_and\_qos\_profile()

TestDataTypeSupport::register\_type() (由zrddsgen生成)

DDS::DomainParticipantFactory::delete\_contained\_entities()

DDS::DomainParticipantFactory::delete\_participant()

DDS::DomainParticipantFactory::finalize\_instance()

# 三、性能数据测试模块

## 3.1吞吐率测试

吞吐率计算公式：数据包大小\*包数量/总时间

吞吐率可分为两个层面：网络层，应用层

网络层的吞吐率计算不需要考虑序列化与反序列化的时间，而应用层则恰恰相反

·输入：size（数据包大小），samples（数据包数量），interval（发送间隔，单位ms），duration（测试持续时间，与测试层面有关）

·输出：总吞吐率、丢包率，吞吐率随时间的变化（时序图/统计）

## 3.2时延测试

计算公式：

时延 = 接收时间戳 - 发送时间戳

平均时延 = 所有时延之和 / 样本数量

·输入：size（数据包大小），samples（数据包数量），interval（发送间隔，单位ms），duration（测试持续时间，与测试层面有关）

·输出：平均时延、最小时延、最大时延，时延分布（可选：时延直方图或时序统计

## 3.3抖动测试

计算公式：

抖动 = |本次时延 - 上一次时延|

平均抖动 = 所有抖动值之和 / （样本数量 - 1）

·输入：size（数据包大小），samples（数据包数量），interval（发送间隔，单位ms），duration（测试持续时间，与测试层面有关）

## 3.4系统资源占用

计算公式：

CPU占用率 = 测试进程CPU时间 / 总CPU时间 \* 100%

内存占用 = 测试进程占用内存峰值与均值

网络带宽利用率 = 网络传输字节数 / 测试时长

·输入：duration（测试持续时间），monitor\_items（监控项：CPU、内存、网络等），test\_config（测试配置，如并发数、负载大小等）

·输出：CPU、内存、网络带宽等资源占用均值、峰值，资源占用随时间变化（时序图/统计），资源占用详细采样数据（可选）

# 四、统计分析模块

该模块主要功能是对性能数据测试/模块所采集的原始数据进行进一步的处理、聚合与分析，生成可解释性更强的统计指标和趋势信息。任务是将原始数据转化为结构化的、可对比的性能指标，并支持后续结果输出与可视化。

·输入：性能测试模块生成的 CSV 文件（包含的字段Bytes, Demand, Recovery Time, Sent Samples, Send Time(us), Packs/sec, MBits/sec, Rec Samples, Lost Samples, Rec Time(us), Packs/sec, MBits/sec 等 ）

·输出：

1. 吞吐率：平均值、峰值、标准差，以及随时间变化的趋势曲线
2. 时延：平均值、最大值、最小值、百分位数统计（如P50、P90、P99），时延分布直方图
3. 抖动：平均抖动值、抖动方差、时序波动曲线
4. 系统资源：CPU、内存、网络带宽等的均值、峰值，随时间变化曲线，资源利用率与负载的相关性分析
5. 统计报告：包括表格化的指标结果、关键性能指标（KPI）对比，以及可选的异常检测结果（如突增、突降告警）

# 五、结构输出模块

该模块负责将统计分析模块的结果转换为多种结构化的输出形式，保证结果可读性与可扩展性，同时方便后续的报告生成与外部系统对接。模块需要支持文本化输出、可视化图表生成，以及结构化文件导出。

·输入：

分析结果（吞吐率、时延、抖动、系统资源统计指标与趋势数据）

输出配置（输出格式：JSON、CSV、Excel、PDF，图表类型配置：折线图、柱状图、直方图，输出目录路径）

展示需求（是否生成可视化图表，是否导出对比分析，是否生成完整报告）

·输出：

1.报表文件：

JSON/CSV/Excel 格式的结构化数据文件，包含各项统计指标与详细采样数据

PDF 格式的性能报告，集成文字说明、统计表格和可视化图表

2.可视化图表：

吞吐率、时延、抖动、系统资源随时间变化的折线图

时延分布直方图、抖动分布图、CPU/内存/网络占用对比柱状图

KPI总览图表，展示核心指标对比与趋势

# 六、总控模块