F-S-08-01： API 接口日志未生成

典型现象

- 调用 WMS/WCS 接口后，在 API 日志管理界面查不到请求记录。
- 排查问题时无法追溯请求报文和响应内容。
- 仅部分接口或特定时间段日志缺失。

可能原因

1. **日志开关未开启**：宽海 WMS 的 API 日志功能在配置中被禁用（如 appsettings.json:LogDeubgEnable/LogAOPEnable=false ）。
2. **磁盘空间不足**：日志存储磁盘已满，新的日志无法写入。
3. **日志文件权限错误**：运行服务的用户对日志目录没有写入权限。
4. **异步队列堆积丢失**：API 日志通过异步队列写入，队列满或消费者异常导致丢弃。
5. **日志级别过滤**：API 日志的打印级别设置为 DEBUG，而生产环境日志级别为 INFO。
6. **响应内容超出**：接口调用后返回的内容太大记录到数据库失败。

排查思路

1. **检查日志配置**：登录 WMS 后台管理界面，查看“系统配置”中 API 日志开关是否开启。也可直接检查配置文件（如 appsettings.json:LogDeubgEnable/LogAOPEnable=false 的值）。
2. **验证磁盘空间**：登录服务器执行 df -h ，检查日志所在分区（如 /var/log ）使用率。若使用率 100%，清理旧日志或扩容。
3. **检查目录权限**：使用 ls -ld /path/to/logdir 查看权限，确保运行用户（如 wms ）有 wx 权限。可通过 chmod 或 chown 修复。
4. **测试手动写入**：在服务器上切换到服务用户，执行 echo "test" >> /path/to/logdir/api.log ，确认能否成功写入。
5. **检查日志级别**：确认生产环境的日志级别为 INFO 或 WARN，且 API 日志输出级别不低于该级别。
6. **测试手动调用**：使用Postman手动调用接口，查看返回内容是否超出数据库字段最大值。

保养提示

- 配置日志轮转策略（如按天切割，保留最近 30 天），并设置磁盘使用率告警（超过 80% 时预警）。
- 关键接口的请求响应单独记录到数据库表中，减少对文件系统的依赖。
- 定期抽查 API 日志的完整性，通过与网关访问日志比对，发现缺失时及时告警。

F-S-08-02：任务异常字段为空但实际有异常

典型现象

- WMS 任务管理界面中，某个任务的“异常字段”列显示为空，但任务明显卡住或执行错误。
- 用户无法根据前端提示快速定位问题原因。
- 后端日志中有详细错误，但未回写到任务表。

可能原因

1. **异常未捕获**：代码中执行任务时发生异常，但未捕获并写入任务的异常字段。
2. **字段映射错误**：异常信息写入了数据库的其他字段（如 ExceptionMessage ），但前端读取的是 Message 。
3. **事务回滚导致字段未保存**：任务状态更新和异常字段写入在同一事务中，事务回滚时异常信息丢失。
4. **异步处理丢失**：任务异常通过异步线程记录，但异步任务未执行或执行失败。
5. **异常信息太大**：异常捕获后更新到任务“异常字段”时，因超出数据库“异常字段”最大值导致更新失败。
6. **异常字段被后续流程覆盖或清空**：任务重试、状态更新、定时清理等逻辑中，可能会重置异常信息字段，但保留了异常状态。
7. **异常发生在字段赋值之后**：先写了异常字段，之后又抛出另一个异常，导致最终状态是异常，但异常字段被覆盖或未更新。

排查思路

1. **查看后端日志**：根据任务 ID 搜索日志，查找是否有 ERROR 或 Exception 记录。确认异常是否存在。
2. **检查数据库表**：直接查询 Dt_Task 表，查看该任务的 ExceptionMessage 、 Remark 等可能存储异常的字段，确认是否有值。
3. **审查代码**：定位任务执行的代码块，检查 catch 块中是否调用了 task.ExceptionMessage=e.Message 并执行了 save() 。
4. **检查事务边界**：确认任务状态更新和异常字段写入是否在同一事务中，且异常发生后事务是否被标记为 rollback-only 导致提交失败。
5. **验证前端映射**：查看前端代码中“异常字段”读取的是哪个 JSON属性，与后端返回的字段名对比。
6. **手动更新字段**：复制异常信息手动更新 Dt_Task 表中存储异常的字段，确认是否能更新成功。
7. **重现并追踪**：找一条“字段为空但有异常”的任务 ID，查看该任务的完整处理日志，找到它被标记为异常的那一步，看此时异常信息有没有被正确写入对象/变量。
8. **检查代码逻辑**：搜索状态变为 异常 的所有代码位置，检查每个位置是否有同步写入异常信息，特别是在重试、状态机切换、回调处理等复杂逻辑中。
9. **检查是否有定时任务或清理逻辑**：是否有后台任务会清空异常信息，比如为了“重试”而清空前一次错误。

保养提示

- 统一封装任务执行器，在 catch 块中强制捕获异常并持久化到数据库，无论事务是否回滚（可使用 @Transactional(propagation = REQUIRES_NEW) ）。

- 定期对比后端错误日志与任务表中异常字段的空置率，发现异常未记录时自动告警。
- 提供“补充异常信息”的管理功能，允许运维人员手动将日志中的错误关联到任务记录。

F-S-08-03：日志出现大量 WARN_ERROR

典型现象

- 应用日志文件中 WARN 或 ERROR 级别日志快速增加，占用大量磁盘空间。
- 日志中重复输出相同的错误信息，但业务表面正常。
- 排查问题时被大量无用错误干扰。

可能原因

1. **业务异常未正确处理**：某些正常业务分支（如数据校验失败）也打印 ERROR 日志，且频率高。
2. **依赖服务不稳定**：调用的外部接口或数据库偶尔超时，重试机制打印 WARN 日志，虽最终成功但日志量大。
3. **日志级别配置错误**：生产环境配置文件误将某个包的日志级别设为 DEBUG 或 TRACE，导致大量细粒度日志输出。
4. **死循环或高频轮询**：代码中存在 while(true) 且每次循环都打印 ERROR，或轮询任务间隔过短。

排查思路

1. **统计错误模式**：使用 grep 、 awk 或日志分析工具（如 ELK）统计前 10 个最频繁的错误日志内容。
2. **分析错误根源**：选择出现频率最高的错误，查看堆栈或错误信息，定位代码位置。
3. **调整日志级别**：临时将对应包的日志级别提升到 WARN 或 ERROR，观察是否减少。若减少，说明原级别设置过低。
4. **优化业务代码**：对于因数据校验打印 ERROR 的情况，改为打印 INFO 或 DEBUG，或降低阈值。
5. **检查轮询任务**：搜索 log.error 出现在循环中的代码，增加间隔时间或条件判断。
6. **检查异步队列**：查看是否有 AsyncAppender 丢弃日志的 WARN。适当增大队列大小或减少日志产出。

保养提示

- 建立“日志规范”：明确 ERROR 仅用于需要人工介入的故障， WARN 用于可自动恢复的临时问题。
- 部署日志聚合系统（如 ELK），设置“ERROR 频率告警”，当某类 ERROR 在 5 分钟内出现 100 次以上时通知开发介入。
- 定期审查高频日志，每周优化 1-2 个不必要的 ERROR/WARN。