**专利技术交底书**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **企业名称：** | 申请人为单位时填写 | | **统一社会信用代码：** |  |
| **申请人姓名：** | 詹志 | | **申请人身份证号：** | 申请人为自然人时填写 |
| **专利名称：** | 一种网络靶场中跨服务统一拓扑数据结构和操作算法的架构设计和实现 | | | |
| **通讯地址:** |  | | **邮编:** |  |
| **是否同步申请实质审查：** | 是/否 | | **是否申请**  **提前公布：** | 是/否 |
| **技术联系人：** | **姓名：詹志**  **电话：13361875998**  **E-mail：1782865441@qq.com** | | | |
| **发明人（可以写多个）：** | | 詹志 | | |
| **第一发明人的身份证号码：** | | 36250219990210463X | | |

|  |
| --- |
| **一、技术领域 ：**本发明或实用新型属于何技术领域，可以在哪些领域应用。 |
| 计算机、软件领域、网络安全 |
| **二、背景技术：**（即现有技术）1、介绍所属技术领域的现有技术，尤其是与本发明欲改进的核心技术有关的技术现状。2、可从大的技术背景（该技术领域的总体状况）和小的技术背景（与本发明改进的具体技术密切相关的技术状况）两个方面进行介绍。3、如果现有技术出自专利文献、期刊、书籍等，则提供出处。现有技术有相关附图的，最好一并提供并结合附图说明。 |
| 网络靶场是一种基于虚拟化技术，对真实网络环境的仿真，作为支撑网络空间安全技术验证，网络武器装备试验，攻防对抗演练和网络风险评估的平台。  在大型网络靶场的实现中，靶场平台会被分拆成许多微服务，如提供虚拟化资源管理能力的云平台服务、提供拓扑实例构建能力的拓扑网络构建服务、提供采集能力的采集服务、提供拓扑信息可视化能力的可视化服务。各个微服务都需要根据当前拓扑实例数据，执行自己的业务操作。例如可视化服务需要查询拓扑实例信息用于界面展示拓扑信息、采集服务需要查询拓扑实例信息用于获取配置的采集信息等。这些服务会按照服务内部对拓扑实例的理解，对从拓扑网络构建服务开放的接口中获取拓扑实例数据，进行解析；或者要求平台拓扑网络构建服务按照其他服务内部需要的数据格式去定制对应的拓扑数据接口。 |
| |  | | --- | | **三、现有技术存在的问题：**1、客观评价现有技术的缺点，会带来哪些问题（这些缺点是针对本发明的优点来说的，本发明正是要解决这些问题和缺点，本发明不能解决的问题和缺点不必写）。2、如果找不到对比技术方案及其缺点，可用反推法，根据本发明的优点来找出对应的缺点。3、缺点可以是成本高、结构复杂、性能差、工艺繁琐等类似问题。4、针对前面现有技术的所有缺点，逐一正面描述本发明所要解决的技术问题。 | | 大型网络靶场实现中，整个靶场平台被拆分成多个微服务。这些服务之间需要共享的核心数据就是拓扑实例的数据。为了获取到拓扑实例数据，这些服务会按照服务内部对拓扑实例的理解，对从平台拓扑网络构建服务开放的接口中获取拓扑实例数据，进行解析；或者要求平台拓扑网络构建服务按照其他服务内部需要的数据格式去定制对应的拓扑数据接口。这样的做法存在下面这些问题：   1. 拓扑网络构建服务需要针对不同的服务单独定制不同的接口，工作量大，对接时间长。 2. 如果拓扑网络构建服务针对不同的服务定制相应接口的能力不足，那其他服务需要在服务内部重新定义实现拓扑数据的格式和操作算法，工作量大，对接时间长。 3. 如果在拓扑网络构建服务内部针对不同服务定制众多业务接口，那拓扑网络构建服务会耦合大量其他服务对拓扑数据的操作，这些操作的业务含义对于平台拓扑网络构建服务不明确，维护很困难。 | |
| |  | | --- | | **四、本发明或实用新型采用的技术方案：**本部分是专利文件中最重要的部分，越详细越好。  1、阐明本发明是通过什么样的技术方案来解决其技术问题的（即实现发明目的），不能只有原理，也不能只作功能介绍，应当详细描述本发明的各个发明改进点及相应的技术方案。  2、技术方案应当清楚、完整地描述本发明的技术特征（如构造、组织、形状等）以及作用、原理，以使本领域普通技术人员能够实施本发明为准。  3、对于不同类型的发明创造需采用不同的描述方式来说明其技术方案。例如：对于设备发明，应当具体说明其零部件的结构及其连接关系，必要时结合附图加以说明；对于方法发明，应当具体说明其工艺方法、工艺流程和条件（如时间、压力、温度、浓度）；涉及机电一体化的发明，对于其中与电路有关的内容应当提供电路图、原理框图、流程图或时序图，并结合附图进行具体说明；对于涉及软件、业务（商务）方法的专利申请，除提供流程图外，还应提供相关的系统模块框图，并按照流程的时间顺序，以自然语言对各步骤进行描述。  4、所有附图应当有相应的文字描述，图中的关键词或注释尽量用中文。所有英文缩写（除了普遍公认有确定含义的）应有中文注释。注意，附图尽可能提供可编辑格式的版本以便代理人调整，如Visio、CAD、PPT等。 | | 本发明提出大型网络靶场中一种软件架构的设计和实现方式，通过在多个微服务之间开放标准的拓扑标准操作SDK和拓扑数据SDK，实现拓扑数据格式的统一化和拓扑标准操作算法的统一化。在这个过程中实现拓扑数据的跨服务存放，不同服务独立存储自身服务相关的部分拓扑数据，拓扑标准操作算法支持自动路由实现跨服务操作拓扑数据。  整个架构设计和实现以下图为例：     1. 由拓扑网络构建服务更新维护拓扑数据SDK和拓扑标准操作算法SDK并上传到nexus repository中，其他服务如可视化服务和采集服务从nexus repository中拉取拓扑数据SDK和拓扑标准操作算法SDK。   拓扑数据SDK是定义了拓扑的数据结构，拓扑操作SDK是定义了操作拓扑数据结构的算法。  拓扑数据SDK由描述整个拓扑数据结构的数据类组成。详细数据类（每个数据类对应数据库中一张表，每个数据类都有其唯一归属的微服务，即其对应的数据库表存储所在的微服务。数据类和归属的微服务关系以getService()方法形式定义在数据类对应的MapperProxy实例类的代码中，MapperProxy实例类具体实现详见6.4.1.2）见下面表格：   |  |  | | --- | --- | | 数据类 | 含义 | | SceneModel | 拓扑全局信息数据类，用于描述记录拓扑的全局配置信息 | | - SceneElementModel | 拓扑节点数据类，用于描述拓扑节点的基本属性如节点id、名称、坐标 | | - - SceneElementPortModel | 拓扑节点端口数据类，用于描述拓扑节点的端口信息，如端口名称，端口ip，vlan值等端口级别的配置信息 | | - SceneElementConnectionModel | 拓扑连线数据类，用于描述拓扑节点连线的信息，记录连线的开始节点和连线的结束节点，以及连线的一些配置信息 | | CloudSceneModel | 云平台服务相关的拓扑全局配置信息，如流量镜像等 | | - CloudSceneElementModel | 云平台服务相关的拓扑节点信息，如节点对应的平台虚拟资源。 | | -- CloudSceneElementPortModel | 云平台服务相关的拓扑端口信息，如端口接入的网络资源信息以及端口对应的网卡id、名称 | | - CloudSceneNetworkModel | 云平台服务相关的拓扑网络信息 | | CollectionSceneModel | 采集服务相关的拓扑全局配置信息，如是否开启采集 | | - CollectionSceneElementModel | 采集服务相关的拓扑节点信息，维护了每个节点的采集项 |   以上拓扑数据SDK只列举和两个服务（云平台和采集服务）相关的拓扑数据类，有其他更多服务相关拓扑数据类可以再额外添加。  本项目使用gradle做构建工具，在拓扑数据SDK模块的build.gradle文件中添加如下内容，执行gradle的发布操作，即可将当前模块发布到项目的maven仓库中。  publishing {  publications {  maven(MavenPublication){  from components.java  }  }  repositories {  maven {  name = 'remote'  allowInsecureProtocol = true  url 'http://<netxus服务ip>/repository/<目录地址>/'  credentials {  username = <netxus账号>  password = <netxus密码>  }  }  }  }   1. 靶场平台所有微服务启动时及启动完成后每分钟向注册中心注册自身的服务标识，以及远程方法调用地址，并拉取所有已注册的信息。   注册格式例如下：  {  "app\_code": "topo\_constructor",  "rpc\_host": "http://172.0.0.1:8081/proxy/call/service"  }   1. 当某用户在可视化服务接口查看某拓扑详情信息，需要传入具体的拓扑实例id 2. 可视化服务接收到请求的参数（拓扑实例id） 3. 可视化服务可以直接使用从nexus repository获取的拓扑标准算法SDK中的SceneHandler构造函数加载拓扑数据。   拓扑标准操作算法指的是拓扑数据的统一查询、遍历、创建、更新算法。下面代码为例：  // 获取当前spring容器内注册的BaseMapperInterface抽象类的实现类实例，SceneHandler只依赖于BaseMapperInterface抽象类接口，不依赖于具体实例。  @Override public BaseMapperInterface<SceneModel> getSceneMapper() {  return (BaseMapperInterface<SceneModel>) BeanCommon.getBean(SceneSdkMapperConstant.SCENE\_MAPPER); }  @Override public BaseMapperInterface<SceneAreaModel> getSceneAreaMapper() {  return (BaseMapperInterface<SceneAreaModel>) BeanCommon.getBean(SceneSdkMapperConstant.SCENE\_AREA\_MAPPER); }  @Override public BaseMapperInterface<SceneElementModel> getSceneElementMapper() {  return (BaseMapperInterface<SceneElementModel>) BeanCommon.getBean(SceneSdkMapperConstant.SCENE\_ELEMENT\_MAPPER); }  public SceneHandler(String sceneId, String areaId, SceneContextBO context) {  init(sceneId, areaId, false, context);  }  public void save() {  boolean isCreate = StringUtils.isEmpty(sceneId);  saveScene();   Map<String, A0> existAreaMap = isCreate ? new HashMap<>() : readAreas().stream()  .collect(Collectors.toMap(BaseSceneAreaModel::getNodeId, x -> x));  var filterAreaIds = new String[0];  if (isAreaMode) {  String currentAreaId = areaId == null ? StringUtils.EMPTY : areaId;  filterAreaIds = ArrayUtils.addAll(existAreaMap.values().stream().filter(BaseSceneAreaModel::getOpen)  .map(BaseSceneAreaModel::getNodeId).toArray(String[]::new), currentAreaId);  }  ...具体保存还有很多实例逻辑，涉及到很多数据类}  private boolean saveScene() {  // 保存SceneModel数据类的代码  S0 sceneObj = scene.toModel();  var result = getSceneMapper().save(sceneObj);  sceneId = sceneObj.getId();  scene.fromModel(sceneObj);   plugins.forEach(plugin -> plugin.setSourceSceneId(sceneId));   return result; }  上面图片中代码是两个函数的一部分。一个是SceneHandler构造函数，是拓扑数据的统一查询算法的实现。一个是save方法，是拓扑数据的统一更新算法的实现部分。其他服务通过引入拓扑标准算法SDK jar包，就可以直接使用拓扑标准算法SDK jar包中定义的各种拓扑操作算法，跨服务操作拓扑数据。   1. 实际加载拓扑数据时，当前服务会根据注入的数据库查询接口类的实例（即6.4.1.2描述的MapperProxy）不同，而执行不同的操作。对于数据表实际存储在当前服务，注入的是mybatis 基于JDK动态代理生成的本地数据库查询类实例；对于数据表不是存储在当前服务，注入的是当前服务内部通过自定义的配置类注册数据库查询接口类的实例，它会根据每个数据库查询接口类实例中getService方法获取其归属的微服务以及从注册中心获取的服务标识和远程调用信息，来请求对应服务地址，基于json的远程方法调用在其他服务内部执行实际数据库操作。在这个过程中采用seata实现跨服务的事务，保证跨服务拓扑数据的一致性。   具体执行流程如下图所示：     * 1. 拓扑操作算法实现时只依赖于抽象拓扑查询接口BaseMapperInterface，拓扑数据中的本地存储数据由本服务的mybatis框架管理的Mapper实现处理；非本地存储的数据，由自定义的MapperProxy处理。   2. 系统启动时，本地数据库接口Mapper会由mybatis框架使用JDK动态代理技术生成实际执行数据库操作的MapperProxy实例，并注册到spring容器中供拓扑标准操作算法使用。   预先定义完成的非本地数据库接口MapperProxy实例会通过Spring Config配置类声明注册到spring容器中供拓扑标准操作算法使用，下面是一个配置类代码简单展示：  @Configuration  public class MapperProxyBeanConfig {  /\*\*  \* 获取场景mapperproxy  \*  \* @return 场景mapperproxy  \*/  @Bean("sceneMapper")  public SceneMapperProxy getSceneMapperProxy() {  return new SceneMapperProxy();  }  /\*\*  \* 获取场景节点mapperproxy  \*  \* @return 场景节点mapperproxy  \*/  @Bean("sceneElementMapper")  public SceneElementMapperProxy getSceneElementMapperProxy() {  return new SceneElementMapperProxy();  }  /\*\*  \* 获取场景节点端口mapperproxy  \*  \* @return 场景节点端口mapperproxy  \*/  @Bean("sceneElementPortMapper")  public SceneElementPortMapperProxy getSceneElementPortMapperProxy() {  return new SceneElementPortMapperProxy();  }  } ......后续还有很多通过当前配置类注册到Spring容器的MapperProxy实例。   * 1. 系统运行时，拓扑标准操作算法的实现类SceneHandler创建类的实例，进行类初始化时，会从Spring容器中获取到依赖的BaseMapperInterface抽象类的具体实例。   2. 系统运行时，操作当前系统本地存储数据时，直接通过mybatis动态生成MapperProxy实例连接本地数据库，进行操作；操作非本地存储数据时，通过预先定义完成的MapperProxy调用基于json的远程方法调用客户端，通过当前数据类对应存储的服务地址，调用到对应服务地址的远程方法调用服务端，最终由其他服务内部的mybatis动态生成MapperProxy实例进行数据操作。      1. 基于json的远程方法调用客户端实现         1. 构造请求参数对象，保留请求参数的值信息和类型信息  |  |  |  | | --- | --- | --- | | 字段 | 类型 | 含义 | | clazz | 字符串 | spring容器中对应的业务bean名称 | | method | 字符串 | 调用方法名称 | | params | 字符串数组 | 将所有方法调用的参数转换为json字符串，记录参数值 | | paramTypes | 字符串数组 | 将所有方法调用的参数类型转换为字符串，记录参数值的类型 |   如上表，请求参数对象含有四个字段，clazz字段为spring容器中对应的业务bean的名称，这个是每个MapperProxy固定的属性。method字段为调用方法名称，这个由当前调用的方法决定。params字段为当前调用方法的参数值转换的json字符串数组。paramsType是对应参数的参数类型，因为在网络传输中，仅使用字符串传递参数值，其参数值的类型会丢失掉，从而服务端仅根据参数值，无法还原实际的参数信息。下面详细介绍一个java中使用字符串能完整保留参数类型信息的算法：   * + - * 1. 初始化使用typeName变量记录当前参数类型信息，变量的值初始为空字符串。         2. 开始判断输入的参数类型信息         3. 如果参数类型是Class类，则直接使用Class.getName()获取参数类型名称，记为className，typeName = typeName + className。         4. 如果参数类型不是Class类，那参数类型就是参数化类型类（ParameterizedType)   获取当前参数化类型的原始类型（ParameterizedType.getRawType)。  将原始类型作为参数递归执行步骤6.4.1.1.2及其后续步骤  获取当前参数化类型的泛型参数数组（ParameterizedType.getActualTypeArguments）。  typeName = typeName + “<”。  如果泛型参数数组不为空，对于每一个泛型参数元素执行下面步骤  将泛型参数作为参数递归执行步骤6.4.1.1.2及其后续步骤  typeName = typeName + “,”。  typeName = typeName + “>”。  最终执行完成typeName即记录着参数类型信息。   * + - 1. 模板方法设计模式实现MapperProxy的通用远程方法调用逻辑     具体实现如上图所示   * + - * 1. 定义BaseMapperInterface接口类描述MapperProxy提供的基础的和数据库交互相关的增删改查能力。定义IProxy接口描述MapperProxy提供的跨服务调用操作能力。         2. ServiceProxy抽象类实现IProxy接口，提供了基础的基于json的远程方法调用组装参数和发起调用的模板实现。BaseMapperProxy抽象类继承了ServiceProxy抽象类同时实现了BaseMapperInterface接口，提供了跨服务进行数据库交互操作的能力。         3. 具体的业务表MapperProxy则只需要继承BaseMapperProxy抽象类，并定义自己的归属服务名称和在归属的服务中的对应的类的标识后，就能方便地对上层业务提供跨服务数据库交互操作的能力。     1. 基于json的远程方法调用服务端实现        1. 解析请求参数对象，还原请求参数的值信息和类型信息   解析请求参数对象主要是将请求对象中params数组中的字符串元素根据其类型信息转换成对应的java对象。可以从paramTypes数组中的对应位置获取到其以字符串形式描述的类型信息。下面详细描述如何从6.4.1.1中得到的类型信息字符串解析出实际的java类型信息。   * + - * 1. 记录输入的类型信息字符串为S,其长度为L,起始扫描位置为H，H初始值为0，中止符集合E包含字符’<’、’,’、’ ‘、’>’。         2. 开始获取原始类型信息，使用raw记录原始类型信息字符串   对于索引a取值从0每次加1递增到L-1，S.charAt(a)为对应索引位置的字符,记录为s。如果E包含了s，则返回子字符串raw=S.subString(0,a)  子字符串raw，通过JDK反射提供的Class.forName即可获取到raw字符串代表的类，记录为rawClass   * + - * 1. 开始获取参数类型信息，遍历字符串S中每个字符，开始的索引值为raw.length，使用数组Arg[]记录参数类型信息字符串。   如果raw.length等于s.length，则直接返回空的Arg[]数组。  判断S.charAt(raw.length)是否等于’<’，如果不等于，则说明S的格式异常，直接抛出异常，中止当前解析。  使用greaterMarkCout记录当前待处理的’<’符号数量，初始为0，使用startIndex记录当前获取到的参数类型信息开始索引，初始为raw.length+1  遍历字符串S中的每个字符，开始的索引值为raw.length+1，每次加1，最大为s.length-1，记录当前索引值为index\_a。  记录s.chartAt(index\_a)字符为c  如果c字符等于’<’，则greaterMarkCout = greaterMarkCout + 1  如果c字符等于’>’，则greaterMarkCout = greaterMarkCout - 1，如果greaterMarkCout小于0，则向Arg[]中添加元素S.subString(staertIndex, index\_a)，中止当前遍历，跳转到6.4.2.1.3.5。  如果c字符等于’,’，且如果greaterMarkCout 等于0，则向Arg[]添加元素S.subString(startIndex, index\_a)。startIndex = index\_a + 1。   * + - * 1. 判断Arg[]数组是否为空，如果为空，则返回rawClass         2. 如果Arg[]数组不为空，则对于Arg[]数组中每个元素递归执行6.4.2.1.1~6.4.2.1.5，使用一个类型数组type[]记录每个元素的执行结果。最终得到参数化类型ParameterizedType其原始类型即为rawClass，泛型参数类型为type[]。       1. 基于JDK反射库实现方法调用，获取方法执行返回值   经过步骤6.4.2.1，解析请求参数对象，可以获取到需要执行的方法归属类、执行的方法名称、以及执行方法所传递的参数。使用JDK反射库中Class类的getMethod方法即可获取到方法对象，并使用invoke方法执行方法调用获取到方法执行的返回值。   * + - 1. 处理方法副作用，构造返回对象   通常方法调用的执行不仅仅只会得到返回值，还会影响调用参数，即产生方法调用副作用。如保存数据时，通常会将保存后生成的主键id回填至保存的数据对象参数中。如果远程方法调用结果仅仅只有返回值，则不能很好得模拟本地方法调用，因此这里返回给远程方法调用客户端的除了本地方法调用的返回值外，还有本地方法调用完成后的方法参数。  远程方法调用客户端获取到方法调用的返回值时，还需要替换本身的方法调用参数值为服务端返回的方法调用参数值，来模拟方法副作用。   * 1. 发起拓扑操作的服务，向seata的事务管理者服务TC发起全局事务。   由于实际的拓扑数据操作是跨服务的，每个服务的本地数据库事务仅仅只能保证自身服务存储的部分拓扑数据的有效性、一致性和完整性，而不能保证完成的跨服务存储的拓扑数据的有效性、一致性、完整性。  因此采用seata分布式事务组件，来支持系统的分布式事务能力。事务管理着服务TC是seata提供的中心化的事务管理服务，维护全局事务和各个服务的事务的状态，并通知各个服务进行全局事务的提交或者回滚。各个业务服务则和TC进行注册/上报本服务的全局事务的执行状态和结果，并执行全局事务的提交或回滚。后续步骤更详细描述了各个业务服务和事务管理者服务TC之间的交互过程。   * 1. 所有的服务都向事务管理者服务TC获取当前拓扑操作的全局锁。   2. 各个服务将内部和拓扑操作相关的以及回滚的undoLog以本地事务的方式提交到本地数据库。   3. 各个服务将本地事务的执行状态都上报给事务管理者服务TC。   4. 事务管理者服务TC获取到所有服务上报的本地事务的执行状态都为成功后，给各个服务发送全局事务执行成功的指令。各个服务收到全局事务成功指令后，释放全局锁，以及删除6.7生成的用于回滚的undoLog。  1. 可视化服务获取到拓扑标准算法SDK以拓扑数据SDK格式解析好的拓扑数据，并在界面上给用户展示。 | |  | |
| **五、本发明或实用新型的关键改进点：**提炼出技术方案的关键改进点（即哪些改进点是重点想要保护的创新内容），列出1、2、3……以提醒代理人注意，便于代理人撰写权利要求书。 |
| 本发明提出大型网络靶场中一种软件架构的设计和实现方式，   1. 通过在多个微服务之间开放标准的拓扑标准操作SDK和拓扑数据SDK，实现拓扑数据格式的统一化和拓扑标准操作算法的统一化。 2. 在这个过程中实现拓扑数据的跨服务存放，不同服务独立存储自身服务相关的部分拓扑数据，拓扑标准操作算法支持自动路由实现跨服务操作拓扑数据。 |
| **六、本发明或实用新型取得的有益效果：**结合本发明创造，写明其与现有技术相比具有的优点：比如产率、质量、精度和效率的提高，能耗、原材料、工序的节省；加工、操作、控制、使用的简便等。注意要结合技术方案中的改进点说明，做到有理有据。 |
| 大型网络靶场实现中，整个靶场平台被拆分成多个微服务。这些服务之间需要共享的核心数据就是拓扑实例的数据。为了获取到拓扑实例数据，这些服务会按照服务内部对拓扑实例的理解，对从平台拓扑网络构建服务开放的接口中获取拓扑实例数据，进行解析；或者要求平台拓扑网络构建服务按照其他服务内部需要的数据格式去定制对应的拓扑数据接口。这样的做法存在下面这些问题：   1. 拓扑网络构建服务需要针对不同的服务单独定制不同的接口，工作量大，对接时间长。 2. 如果拓扑网络构建服务针对不同的服务定制相应接口的能力不足，那其他服务需要在服务内部重新定义实现拓扑数据的格式和操作算法，工作量大，对接时间长。 3. 如果在拓扑网络构建服务内部针对不同服务定制众多业务接口，那拓扑网络构建服务会耦合大量其他服务对拓扑数据的操作，这些操作的业务含义对于平台拓扑网络构建服务不明确，维护很困难。   **与原有技术相比，本发明**通过在多个微服务之间开放标准的拓扑标准操作SDK和拓扑数据SDK，实现拓扑数据格式的统一化和拓扑标准操作算法的统一化。同时在这个过程中实现拓扑数据的跨服务存放，不同服务独立存储自身服务相关的部分拓扑数据，拓扑标准操作算法支持自动路由实现跨服务操作拓扑数据。最终实现了网络靶场平台不同服务以来的拓扑数据的解耦，不同服务独立对归属本服务的部分拓扑数据通过统一的操作算法进行额外的内部业务操作，提升了网络靶场平台的可维护性。 |
| **七、本发明或实用新型的替代方案：**1、如有替代方案，请详尽写明，以提供足够多的具体实施方式，有助于使专利的保护范围更宽。2、替代方案可以是部分结构、器件、方法步骤的替代，也可以是完整技术方案的替代。 |
| **无** |

注意：

1.代理人并不是技术专家，交底书要使代理人能看懂，尤其是背景技术和详细技术方案，一定要写的全面、清楚。

2.英文缩写有中文译文，避免使用英文单词，最好在术语解释部分给出。

3.**全文对同一事物的叫法应统一，避免出现一种东西多种叫法。**

4.认为需要保密的地方可在交底书中注明，对代理人不必保密。

5.专利法规定：

1）专利必须是一个技术方案，应该阐述发明目的是通过什么技术方案来实现的，不能只有原理，也不能只做功能介绍；

2）专利必须充分公开，以本领域技术人员不需付出创造性劳动即可实现为准。