# 测试依据

## 技术指标

根据研制总要求的内容，技术指标包括了功能要求和性能要求两个部分。原型版实现了技术指标的核心内容。部分技术指标以斜体表示，留待基本版阶段实现，这部分指标的测试项与测试用例在基本版阶段细化。其他指标在原型版阶段均已实现。

### 功能要求

**1．共性要求**

（1）支持飞腾、龙芯、申威等国产硬件平台，其中龙芯平台支持龙芯3A4000和龙芯3B4000，飞腾平台支持FT 2000客户机和FT 2000+服务器，*申威平台支持申威6B服务器，支持申威421或者4D客户机中的至少一款；*

（2）支持统信、银河麒麟、普华等《JY关键软硬件自主可控产品名录》中的JY通用操作系统中的至少2种，支持的每种JY通用操作系统至少支持两个版本，其中麒麟操作系统支持银河麒麟桌面操作系统（GF版）V10、银河麒麟高级服务器操作系统（GF版）V10两个版本，统信操作系统支持统信桌面操作系统（JY版）、统信服务器操作系统（JY版）两个版本，普华操作系统支持普华桌面操作系统V5.0、普华服务器操作系统V5.0两个版本。

**2．应用框架要求**

应用框架对JS应用所需通用基础API接口进行统一规范和封装，完整覆盖JS应用所需（包括但不限于）界面交互、数据传输、数据访问、文本解析、JY加解密服务、数据库访问、日志处理、图像处理、协议处理功能，支持C/S、B/S等JS应用常用架构。包括了Javascript客户端应用框架、C++服务端应用框架、java服务端应用框架等三个框架。

（1）Javascript客户端应用框架要求

1）界面交互能力，支持表单、图表、树、列表、编辑框、布局、数据绑定等常用界面控件；

2）界面集成能力，提供界面框架，包括个性化页面框架、信息浏览框架以及功能导航框架，支持在框架中集成web页面，支持以插件的方式将常用界面控件动态加载到界面框架；

3）界面驱动能力，支持数据驱动的界面展现，支持根据设备的屏幕大小自动调整界面的显示效果；

4）本地界面融合能力，支持将web页面封装成为本地窗口,并为web页面提供创建系统菜单及系统托盘的能力，支持在web页面里对封装本地窗口的位置、大小等进行显示控制；

5）窗口之间的信息交互能力，支持web页面窗口之间、web页面与本地窗口之间的信息交互；

6）调用本地shell命令与脚本的能力，支持在web页面里调用本地shell命令与脚本；

7）数据传输能力，支持在界面程序中通过javascript语言调用本地数据传输库，支持基于tcp、websocket、http等协议的数据传输；

8）数据访问能力，支持在界面程序中通过javascript语言进行持久化存储，以及对本地websql数据库进行访问；

9）文本解析能力，支持在界面程序中通过javascript语言调用本地文本解析库，支持对ini、xml、Json等常见文本格式进行解析；

10）日志处理能力，支持在界面程序中通过javascript语言调用本地日志处理库，支持将日志记录到本地文件；

11）图像处理能力，支持在界面程序中通过javascript和html语言进行常用的canvas图像处理；

12）协议处理能力，支持在界面程序中通过javascript语言调用本地协议处理库，支持对protobuf、xml、http等格式的数据协议进行处理；

13）规范统一的javascript高级API接口，对界面集成、数据传输、数据访问、文本解析、日志处理、图像处理、协议处理等方面的能力进行统一规范；

14）框架支持7\*24小时连续稳定运行。

（2）C++服务端应用框架

1）模块集成能力，提供插件集成框架，支持以插件的方式对软件模块进行集成，支持动态加载插件，支持对插件的生命周期进行管理；

2）多任务处理能力，提供任务处理插件，支持基于任务队列与多线程的方式对服务请求进行处理；

3）数据传输能力，提供数据传输插件，支持基于tcp协议、http协议的数据传输；

4）数据访问能力，提供数据访问插件，支持对sqllite等典型数据文件进行访问；

5）文本解析能力，提供文本解析插件，支持对ini、xml、Json等常见文本格式进行解析；

*6）JY加解密服务能力，提供JY加解密插件，支持对内存数据进行对称加密、非对称加密、摘要、签名、验证等JY加解密操作；*

*7）数据库访问能力，提供数据库访问插件，支持对达梦、金仓、神通等主流国产数据库提供接口一致的库访问能力，支持数据库连接池；*

8）日志处理能力，提供日志处理插件，支持将日志记录到本地文件，支持调试、告警、错误、正常等日志类型；

9）协议处理能力，提供协议处理插件，支持对protobuf、xml、http等格式的数据协议进行处理；

*10）运维监控能力，提供运维监控插件，支持对服务的运行状态进行监控；*

11）规范统一的C++高级API接口，对模块集成、多任务处理、数据传输、数据访问、文本解析、JY加解密、数据库访问、日志处理、协议处理、运维监控等方面的能力进行规范；

12）框架支持7\*24小时连续稳定运行。

（3）Java服务端应用框架

1）模块集成能力，支持以插件的方式对软件模块进行动态集成；

2）流程集成能力，支持通过注解、配置文件等方式进行流程定制，能动态集成事务处理、日志记录、异常处理、权限验证等流程节点；

3）数据传输能力，支持基于tcp协议、http协议的数据传输；

4）数据访问能力，支持对sqllite等典型数据文件进行访问；

5）文本解析能力，支持对ini、xml、Json等常见文本格式进行解析；

*6）数据库访问能力，支持对达梦、金仓、神通等主流国产数据库提供接口一致的库访问能力，支持数据库连接池；*

7）日志处理能力，支持将日志记录到本地文件；

8）协议处理能力，支持对protobuf、xml、http等格式的数据协议进行处理；

9）规范统一的配置和API接口，对模块集成、流程集成、数据传输、数据访问、文本解析、数据库访问、日志处理、协议处理、运维监控、模块扩展等方面的能力进行规范；

10）框架支持7\*24小时连续稳定运行。

**3．应用运行库要求**

应用运行库一方面可以对应用框架提供支撑，同时也可以直接对应用程序提供支撑。应用运行库主要包括图形界面、输入输出、文本处理、网络传输、安全处理、多任务处理等15类，涵盖了应用所需的基础支撑能力。每类应用运行库都需要提供不少于１个库产品，各类库的主要功能要求如下：

（1）图形界面库，提供列表框、文本编辑、滚动条、菜单等常用控件，支持C++、Java、JavaScript语言。

（2）输入输出库，提供基础的网络IO、文件IO与数据压缩的能力，提供支持异步IO模型的编程接口，支持C、C++、Java语言。

（3）文本处理库，提供ini、xml、Json等常用文本文件修改与解析能力，提供文本的正则表达式处理能力，支持C、C++、Java语言。

（4）网络传输库，支持基于tcp、udp、http、ftp等常见传输协议的网络传输能力，提供远程过程调用能力，提供新研可靠报文传输库，支持可靠的报文流传输，支持C、C++、Java语言。

（5）安全处理库，提供数据摘要、数字签名与验证、对称加密、非对称加密等能力，支持C、C++、Java语言。

（6）多任务处理库，提供跨平台的多任务处理和调度能力，支持C、C++、Java语言，新研消息队列线程池库，提供基于线程池和任务消息队列的多线程处理能力，支持C、C++语言。

（7）协议处理库，提供用户自定义数据结构基于protobuf协议的序列化与反序列化能力，提供xml、Json、http等常用协议的解析处理能力，提供自描述字典协议处理库，支持以键值对的方式对协议数据包进行动态修改，支持对协议元信息的解析，支持以二进制、xml、Json等格式进行协议的序列化和反序列化。

*（8）数据库访问库，提供达梦、金仓、神通等常用国产数据库以及文件数据库的C、C++、Java等语言的访问接口。*

（9）日志处理库，提供日志记录能力，支持多种日志级别与日志输出格式。支持C、C++、Java语言。

（10）配置解析库，提供ini、xml、json等格式配置文件的解析能力，支持C、C++、Java语言。

（11）图像处理库，提供png、jpg、bmp等多种图像格式的相互转换能力，支持图像的旋转、缩放、滤波、模糊、锐化等常见处理能力，支持C、C++语言。

（12）音频视频处理库，提供音视频的采集、分离、转换、编解码等能力，支持mpg、avi、flv等常见音视频格式。支持C、C++、Java语言。

（13）系统诊断库，提供应用程序内存检测、函数调用栈分析、本地调试等能力，支持对C/C++程序和Java程序的诊断。

（14）文件系统访问库，提供文件系统的文件及目录创建、更改、删除等功能的高级接口，支持C++和Java语言。

*（15）提供.net 4.5以上的国产平台适配优化版本，支持基于.net的典型windows服务程序的运行。*

（16）提供chromium v89以上的国产平台适配优化版本。

（17）*提供dpdk v19.11以上的国产平台适配优化版本，提供spdk v20.01以上的国产平台适配优化版本，*提供qt v4.8以上、v5.11以上的国产平台适配优化版本，*提供openmpi v4.1以上的国产平台适配优化版本。*

**4．运行环境要求**

运行环境要求含应用沙箱以及系统库、系统工具需求，支持JS应用二进制跨发行版兼容运行，支持跨硬件平台的JS应用源码级兼容，提供典型的运行时框架产品，运行环境还包括java、javascript等语言运行时。

（1）应用沙箱支持设备挂载、系统用户管理、本地库调用、进程管理等系统基本能力；

（2）应用沙箱支持跨主机网络通信，支持管道、消息队列、信号、信号量等应用间常见IPC通信能力，支持应用之间的内存映射文件共享、文件系统共享等常见共享操作；

（3）应用沙箱支持服务端程序和客户端程序的运行，支持客户端界面的展现；

（4）应用沙箱支持同一款CPU芯片下，基于运行时框架接口开发的应用和多版本库跨国产操作系统发行版的二进制兼容；

（5）应用沙箱支持当应用所需库与运行环境自带库出现版本冲突时，应用按需加载和调用自带的库版本；

（6）支持C/C++、Java、Javascript等类型的多个应用分别运行于不同的应用沙箱中，支持在不同硬件平台下JS应用源码级兼容，当多个应用的依赖库存在版本冲突时，支持各自加载所需的库版本，支持同一个应用的多个版本同机兼容运行；

（7）支持在应用沙箱内进行库的兼容性检查和确认，确保运行时框架内部包含的应用框架、应用运行库及系统库的静态依赖关系自洽无冲突；

（8）提供jdk的v8、v11的国产平台适配优化版本，提供javascript v8 8.9以上的国产平台适配优化版本，提供python语言运行时v3.8以上的国产平台适配优化版本，*提供luajit的v2.1以上的国产平台适配优化版本；*

（9）提供QT运行时框架，集成和封装了QT库、系统库和系统工具，配合应用沙箱，支持常见QT界面应用程序的运行；

（10）提供Javascript运行时框架，集成和封装了Javascript客户端应用框架、Javascript基础库、系统库和系统工具，配合应用沙箱，支持Javascript界面应用程序的运行；

（11）提供C++运行时框架，集成和封装了C++服务端应用框架、C++基础库、系统库和系统工具，配合应用沙箱，支持C++服务端应用程序的运行；

（12）提供Java运行时框架，集成和封装了Java服务端应用框架、Java基础库、系统库和系统工具，配合应用沙箱，支持Java服务端应用程序的运行。

**5．配套工具要求**

（1）规范运行时框架接口，包括应用框架、基础库的API接口，API接口的内容与国产芯片技术无关，支持跨国产平台的应用源码兼容；

（2）支持对运行时框架依赖的操作系统镜像、基础库进行集中管理，支持对运行时框架的安装包进行集中管理，支持对运行时框架的安装包进行版本管理；

*（3）支持对运行时框架和基础库的发布共享与技术交流，支持对提交共享的基础库进行测试认定；*

（4）支持基于基础库进行运行时框架的按需构建与封装，支持对运行时框架的功能进行验证；

（5）支持对应用和运行时框架进行安装、升级和卸载，支持对安装信息进行管理，*支持对应用启动的快捷方式进行管理；*

（6）支持应用基于运行时框架的启动和运行，支持多个运行时框架产品同机运行，支持同一个运行时框架产品的多版本同机运行；

*（7）支持对应用沙箱进行监视，包括应用沙箱中运行的进程、应用沙箱资源占用、进程运行时长、运行时库的使用情况、系统日志等信息；*

*（8）支持对应用沙箱进行控制，包括停止应用沙箱中运行的进程、终止正在运行的应用沙箱、保存应用运行状态等操作；*

*（9）支持基于JY通用集成开发环境及QTCreator等通用集成开发环境，进行运行时框架应用的开发。*

### 性能要求

（1）运行时框架下SPECjvm、Octane、*L2fwd*典型benchmark效率（分值/CPU整数峰值）至少在一款国产硬件平台下达到同期主流Intel平台的80%以上，Intel平台选型确定为Intel金牌5115（配置：10核CPU，2.40GHz，2019年主流机型）；

（2）应用沙箱的UnixBench效率达到同机docker容器的80%以上。

# 测试用例描述

## 共性要求/GN\_GXC

### 龙芯平台测试

1. 龙芯平台支持龙芯3A4000测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 龙芯平台支持龙芯3A4000/GN\_GXC\_LX3A | | | | |
| 用例说明 | | **测试目的：**测试运行时框架的原型版技术指标在龙芯3A4000上的支持情况。  **测试方法：**基于龙芯3A4000，分别安装麒麟、统信桌面操作系统，执行所有原型版测试用例；  **合格判据：**所有原型版测试用例均在龙芯3A4000上通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 龙芯3A4000计算机就绪，部署好麒麟桌面操作系统。 | 无 | 在龙芯3A4000、银河麒麟桌面操作系统（GF版）V10上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与结果一致 | - |
| 步骤 2 | 龙芯3A4000计算机就绪，部署好统信桌面操作系统。 | 无 | 在龙芯3A4000、统信桌面操作系统（JY版）上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与结果一致 | - |

1. 龙芯平台支持龙芯3B4000测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 龙芯平台支持龙芯3B4000/GN\_GXC\_LX3B | | | | |
| 用例说明 | | **测试目的：**测试运行时框架的原型版技术指标在龙芯3B4000上的支持情况。  **测试方法：**基于龙芯3B4000，分别安装麒麟、统信服务器操作系统，执行所有原型版测试用例；  **合格判据：**所有原型版测试用例均在龙芯3B4000上通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 龙芯3B4000计算机就绪，部署好麒麟服务器操作系统。 | 无 | 在龙芯3B4000、银河麒麟服务器操作系统（GF版）V10上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与预期结果一致 | - |
| 步骤 2 | 龙芯3B4000计算机就绪，部署好统信服务器操作系统。 | 无 | 在龙芯3B4000、统信服务器操作系统（JY版）上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与预期结果一致 | - |

### 飞腾平台测试

1. 飞腾平台支持FT 2000客户机测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 飞腾平台支持FT 2000客户机/GN\_GXC\_FTKHJ | | | | |
| 用例说明 | | **测试目的：**测试运行时框架的原型版技术指标在FT 2000客户机上的支持情况。  **测试方法：**基于FT 2000客户机，分别安装麒麟、统信桌面操作系统，执行所有原型版测试用例；  **合格判据：**所有原型版测试用例均在FT 2000客户机上通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | FT 2000客户机计算机就绪，部署好麒麟桌面操作系统。 | 无 | 在FT 2000客户机、银河麒麟桌面操作系统（GF版）V10上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与结果一致 | - |
| 步骤 2 | FT 2000客户机计算机就绪，部署好统信桌面操作系统。 | 无 | 在FT 2000客户机、统信桌面操作系统（JY版）上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与结果一致 | - |

1. 飞腾平台支持FT 2000+服务器测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 飞腾平台支持FT 2000+服务器/GN\_GXC\_FTFWQ | | | | |
| 用例说明 | | **测试目的：**测试运行时框架的原型版技术指标在FT 2000+服务器上的支持情况。  **测试方法：**基于FT 2000+服务器，分别安装麒麟、统信服务器操作系统，执行所有原型版测试用例；  **合格判据：**所有原型版测试用例均在FT 2000+服务器上通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | FT 2000+服务器计算机就绪，部署好麒麟服务器操作系统。 | 无 | 在FT 2000+服务器、银河麒麟服务器操作系统（GF版）V10上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与预期结果一致 | - |
| 步骤 2 | FT 2000+服务器计算机就绪，部署好统信服务器操作系统。 | 无 | 在FT 2000+服务器、统信服务器操作系统（JY版）上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与预期结果一致 | - |

### 申威平台测试

1. 申威平台支持申威421客户机测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 申威平台支持申威421客户机/GN\_GXC\_SWKHJ | | | | |
| 用例说明 | | **测试目的：**测试运行时框架的原型版技术指标在申威421客户机上的支持情况。  **测试方法：**基于申威421客户机，安装银河麒麟桌面操作系统、统信桌面操作系统，执行部分关键原型版测试用例；  **合格判据：**所有原型版测试用例均在申威421客户机上通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 申威421客户机就绪，部署好统信桌面操作系统。 | 无 | 在申威421客户机、银河麒麟客户机操作系统（GF版）V10上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与预期结果一致 | - |
| 步骤 1 | 申威421客户机就绪，部署好统信桌面操作系统。 | 无 | 在申威421客户机、统信客户机操作系统（JY版）上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与预期结果一致 | - |

1. 申威平台支持申威6B服务器测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 申威平台支持申威421客户机/GN\_GXC\_SWKHJ | | | | |
| 用例说明 | | **测试目的：**测试运行时框架的原型版技术指标在申威421客户机上的支持情况。  **测试方法：**基于申威6B服务器，安装银河麒麟服务器操作系统、统信服务器操作系统，执行部分关键原型版测试用例；  **合格判据：**所有原型版测试用例均在申威6B服务器上通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 申威421客户机就绪，部署好统信桌面操作系统。 | 无 | 在申威6B服务器、银河麒麟服务器操作系统（GF版）V10上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与预期结果一致 | - |
| 步骤 1 | 申威421客户机就绪，部署好统信桌面操作系统。 | 无 | 在申威6B服务器、统信服务器操作系统（JY版）上，运行5.2~5.5所有测试项和测试用例。 | 所有测试用例通过。 | 与预期结果一致 | - |

## 应用框架/GN\_ YYKJC

### JavaScript客户端应用框架测试（贺荣）

1. 界面交互功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 界面交互功能/GN\_YYKJ\_JS\_JMJH | | | | |
| 用例说明 | | **测试目的：**测试软件是否支持常用界面控件。  **测试方法：**通过访问已经集成好的常用界面控件库的页面，判断常用界面控件是否在框架中能够正常显示。  **合格判据：**常用界面控件能够正常在框架中显示。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录的用户已成功安装“常用界面控件”应用。 | 用户名、密码等登录信息。 | 用户登录JavaScript客户端应用框架。 | 成功访问JavaScript客户端应用框架。 | 与结果一致 | - |
| 步骤 2 | 用户成功访问JavaScript客户端应用框架。 | 无 | 用户点击“常用界面控件”应用。 | 成功进入”常用界面控件”应用，可以成功展示常用界面控件的展示效果，包括表单、图表、树、列表、编辑、布局、数据绑定等常用界面控件。 | 与结果一致 | - |

1. 界面集成功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 界面集成功能/GN\_YYKJ\_JS\_JMJC | | | | |
| 用例说明 | | **测试目的：** 测试软件的界面集成能力。  1.提供界面框架，包括个性化页面框架、综合信息浏览框架以及功能导航框架。  2.支持在框架中集成web页面。  3.支持以插件的方式将常用界面控件动态加载到界面框架。  **测试方法：**  1.通过功能导航框架的导航，找到个性化页面框架的入口，进入到首页配置界面。  2.在个性化设置框架的消息订阅列表中找到web页面插件和控件展示插件，拖动添加并保存设置。  3.回到综合信息浏览框架，可以看到在个性化页面框架的设置生效，综合信息浏览框架中成功集成了web页面和插件展示应用。插件展示应用动态展示了常用的图标控件。  **合格判据：**  1.提供界面框架，可以通过功能导航框架进入个性化页面框架的消息订阅列表中，选择插件集成。集成后在综合信息浏览框架中看到集成对应的插件展示效果。  2.支持在框架中集成web页面，可以在综合信息浏览框架中看到集成了web页面。  3.支持以插件的方式将常用界面控件动态加载到界面框架，可以在综合信息浏览框架中看到集成的控件展示插件中动态展示了页面的内容。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行。框架中存在以常用控件开发的插件小程序。 | 用户名、密码等登录信息。 | 用户登录JavaScript客户端应用框架。 | 成功访问JavaScript客户端应用框架。 | 与评价结果一致 | - |
| 步骤 2 | 成功访问JavaScript客户端应用框架。 | 无 | 点击设置。 | 成功展示功能导航。 | 成功展示功能导航 | - |
| 步骤 3 | JavaScript客户端应用框架成功展示功能导航框架。 | 无 | 在功能导航中找到个性化页面框架的首页配置入口，并点击进入。 | 成功进入首页配置页面，点击消息订阅列表查看已有的小程序插件和用户的个性化设置信息。 | 与预期结果一致 | - |
| 步骤 4 | JavaScript客户端应用框架的个性化页面框架成功展示消息订阅列表。 | 无 | 在消息订阅列表中找到“web页面”和“控件展示”放置在显示区域，并提交保存。 | 成功添加插件小程序并保存设置。 | 与预期结果一致 | - |
| 步骤 5 | JavaScript客户端应用框架的个性化页面框架成功添加插件并保存设置。 | 无 | 重新打开综合信息浏览框架页面。 | 成功看到综合信息浏览框架集成并展示了web页面插件小程序和控件展示插件小程序。 | 与预期结果一致 | - |
| 步骤 5 | JavaScript客户端应用框架的综合信息浏览框架展示了web页面插件和控件展示插件。 | 无 | 查看综合信息浏览框架。 | 综合信息浏览框架能够正常显示“web页面”和“控件展示”插件小程序中的内容。 | 与预期结果一致 |  |

1. 界面驱动功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 界面驱动功能/ GN\_YYKJ\_JS\_JMQD | | | | |
| 用例说明 | | **测试目的：** 测试软件是否支持界面驱动功能。  **测试方法：**添加数据驱动界面插件小程序，确定好数据源后，软件可以实现界面驱动功能以及界面显示自适应。  **合格判据：** 数据驱动界面能够动态展现接收到的数据。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，通知通告数据接口程序运行正常。 | 用户名、密码等登录信息。 | 管理员访问小程序订阅设置。 | 成功访问小程序订阅设置。 | 与结果一致 | - |
| 步骤 2 | 成功访问小程序模板管理页面 | 输入必填项 | 点击添加频道按钮，新增频道名称为通知通告，组件名称为itemCardNotification的模板，点击确认 | 保存成功 | 与结果一致 | - |
| 步骤 3 | 成功进入小程序分类管理界面 | 无 | 点击添加小程序订阅分类，输入必填项。点击确认。 | 保存成功 | 与结果一致 | - |
| 步骤 4 | 通知通告数据接口程序运行正常，成功访问小程序订阅管理界面。 | 通知通告数据接口地址。 | 在小程序订阅管理页面，点击添加，输入订阅名称和显示标题均为通知通告，类型字段为远程接口，订阅分类选为步骤3设置的分类名称，订阅频道为通知通告，接口地址为通知通告数据接口地址，点击确认。 | 保存成功。 | 与结果一致 | - |
| 步骤 5 | JavaScript客户端应用框架正常运行，成功在小程序订阅管理页面新增应用插件小程序。 | 用户名、密码等登录信息。 | 普通用户登录访问JavaScript客户端应用框架。 | 登录成功。 | 与结果一致 | - |
| 步骤 6 | 成功登录JavaScript客户端应用框架，功能导航框架成功展示导航。 | 无 | 在功能导航中找到个性化页面框架的首页配置入口，并点击进入。 | 成功进入首页配置页面，点击消息订阅列表查看已有的小程序插件和用户的个性化设置信息。 | 与结果一致 |  |
| 步骤 7 | 成功进入个性化页面框架的首页配置页面。 | 无 | 在消息订阅列表中，选择刚刚新增的插件小程序放置在显示区域，点击保存进行控件位置的存储。 | 保存成功。 | 与结果一致 |  |
| 步骤 8 | 个性化页面框架的首页配置保存成功。 | 无 | 回到综合信息浏览框架综合首页。 | 综合首页显示刚刚新增的通知通告插件小程序，小程序中展示通知通告的最新数据。 | 与结果一致 |  |
| 步骤 9 | 个性化页面框架的首页配置保存成功，成功关闭JavaScript客户端应用框架。 | 无 | 将屏幕的分辨率分别调整为“1600\*900”、“1920\*1080”，刷新页面，重新打开JavaScript客户端应用框架。 | 综合首页根据设备的分辨率自动调整界面的显示效果。 | 与结果一致 |  |

1. 本地界面融合功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 本地界面融合功能/ GN\_YYKJ\_JS\_JMRH | | | | |
| 用例说明 | | **测试目的：** 测试软件的本地界面融合功能。  1.支持将web页面封装成为本地窗口，可以创建系统菜单及系统托盘。  2.支持在web页面里对封装本地窗口的位置、大小等进行显示控制  **测试方法：** 登录JavaScript客户端应用框架后可以创建系统菜单及系统托盘，并对窗口位置大小等显示进行控制。  **合格判据：** 成功创建系统菜单及系统托盘，并对窗口的位置大小进行控制。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行。 | 用户名、密码等登录信息。 | 用户访问JavaScript客户端应用框架。 | 成功访问JavaScript客户端应用框架，自动创建系统托盘窗口和系统菜单。 | 与预期结果一致 | - |
| 步骤 2 | 用户成功登录JavaScript客户端应用框架。 | 无 | 点击JavaScript客户端应用框架上的“最小化”图标。 | JavaScript客户端应用框架窗口最小化成功。 | 与预期结果一致 |  |
| 步骤 3 | 用户成功登录JavaScript客户端应用框架。 | 无 | 在JavaScript客户端应用框架最小化窗口，在窗口内的web页面可拖拽区域，执行拖拽操作。 | 成功改变JavaScript客户端应用框架最小化显示窗口位置。 | 与预期结果一致 |  |

1. 窗口之间的信息交互功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 窗口之间的信息交互功能/ GN\_YYKJ\_JS\_XXJH | | | | |
| 用例说明 | | **测试目的：** 测试软件的窗口之间的信息交互功能。  1. 支持web页面窗口之间进行信息交互  2. 支持web页面与本地窗口之间进行信息交互  **测试方法：** 访问JavaScript客户端应用框架，打开2个web应用和一个本地应用，其中一个web应用发送的信息可以被另外两个应用接收并展示出来。  **合格判据：** 未发送信息的web窗口和本地窗口可以展示发送信息的web窗户口发送的信息。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户成功安装相关的“web信息发送”、“web信息接收”、“本地信息接收”应用。 | 用户名、密码等登录信息。 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 成功访问框架，并查看安装列表 | - |
| 步骤 2 | 在已安装的应用列表，可以看到相关的“web信息发送”、“web信息接收”、“本地信息接收”应用。 | 无 | 分别启动相关的3个“web信息发送”、“web信息接收”、“本地信息接收”应用。 | 成功打开2个web页应用，和一个本地应用。 | 与预期结果一致 | - |
| 步骤 3 | 成功打开2个web页应用，和一个本地应用 | 无 | 在“web信息发送”应用中输入一段文字信息，点击“发送消息”。 | “web信息接收”、“本地信息接收”应用成功显示刚刚发送的信息。 | 与结果一致 | - |

1. 调用本地shell命令功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 调用本地shell命令功能/ GN\_YYKJ\_JS\_DYML | | | | |
| 用例说明 | | **测试目的：** 测试软件是否支持调用本地shell命令功能。  **测试方法：** 通过打开Shell调用应用，展示调用shell脚本获取到的信息。  **合格判据：** 支持调用本地shell命令功能，Shell调用应用显示调用shell脚本的返回信息。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户已成功安装“Shell调用应用”。 | 用户名、密码等登录信息。 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 与预期结果一致 | - |
| 步骤 2 | 在已安装的应用列表，可以看到“Shell调用”应用。 | 无 | 在已安装的应用列表，打开“Shell调用”应用 | 成功打开“Shell调用”应用，并显示Shell调用应用调用shell脚本的返回信息。 | 与结果一致 | - |

1. 数据传输功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 数据传输功能/ GN\_YYKJ\_JS\_SJCS | | | | |
| 用例说明 | | **测试目的：** 测试软件的数据传输能力  1. 支持基于websocket、http等协议的数据传输（用例中缺少了TCP）  **测试方法：**打开应用了本地数据传输库的程序，打开发送端发送数据。  1.支持在界面程序中通过javascript语言调用本地数据传输库进行基于websocket协议的数据传输，并在接收端接收显示信息。  2.支持在界面程序中通过javascript语言调用本地数据传输库进行基于http协议的数据传输，并在接收端接收显示信息。  **合格判据：**  1.JavaScript客户端应用框架使用websocket协议对数据进行传输的能力，可以发送并接收显示websocket协议传送的信息。  2.JavaScript客户端应用框架使用http协议对数据进行传输的能力，可以发送并接收显示http协议传送的信息 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户已成功安装“信息发送”、“信息接收”应用。 | 用户名、密码等登录信息。 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 与预期结果一致 | - |
| 步骤 2 | 在已安装的应用列表，可以看到“信息发送”、“信息接收”应用。 | 无 | 在已安装的应用列表中打开“信息发送”、“信息接收”应用。 | 成功打开“信息发送”、“信息接收”应用。 | 与预期结果一致 | - |
| 步骤 3 | “信息发送”、“信息接收”应用成功打开。 | 自定义信息。 | 点击“信息发送”应用页面中的“websocket协议发送信息”按钮，使用websocket协议对输入的自定义信息数据报文进行传输。 | “信息接收”应用页面展示“信息发送”应用页面中使用websocket协议传送来的数据信息。 | 与预期结果一致 | - |
| 步骤 4 | “信息发送”、“信息接收”应用成功打开。 | 自定义信息。 | 点击“信息发送”应用页面中的“http协议发送信息”按钮，使用http协议对输入的自定义信息数据报文进行传输。 | “信息接收”应用页面展示“信息发送”应用页面中使用http协议传送来的数据信息。 | 与预期结果一致 |  |

1. 数据访问功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 数据访问功能/GN\_YYKJ\_JS\_SJFW | | | | |
| 用例说明 | | **测试目的：** 测试软件是否支持数据访问功能，包括用javascript语言进行持久化存储以及访问websql数据库。  **测试方法：** 在应用中对数据数据进行持久化存储，对websql进行数据新增，关闭应用并重新打开后仍能获取到数据。  **合格判据：** 支持数据访问功能，可以用javascript语言对数据进行持久化存储，可以访问websql。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户已成功安装数据访问“数据访问”应用。 | 用户名、密码等登录信息。 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 与预期结果一致 | - |
| 步骤 2 | 在已安装的应用列表中可以看到“数据访问用。 | 无 | 在已安装的应用列表，打开“数据访问”应用。 | 成功打开“数据访问”应用。 | 与预期结果一致 | - |
| 步骤 3 | “数据访问”应用已经成功打开。 | 自定义信息。 | 在“数据访问”应用页面的“持久化”输入框中输入信息并保存。 | 提示保存成功。 | 与结果一致 | - |
| 步骤 4 | “数据访问”应用已经成功打开。 | 自定义信息。 | 在“数据访问”应用页面的“存储到websql”输入框中输入信息，点击“新增”进行新增数据。 | 提示新增成功。 | 与结果一致 | - |
| 步骤 5 | “数据访问”应用已经成功打开。 | 无 | 点击关闭按钮，关闭应用。 | 成功关闭“数据访问”应用。 | 成功关闭 |  |
| 步骤 6 | “数据访问”应用已经关闭，在已安装的应用列表中可以看到“数据传输”应用。 | 无 | 重新打开“数据访问”应用。 | 成功打开“数据访问”应用。 | 成功打开 |  |
| 步骤 7 | “数据访问”应用已经成功打开。 | 无 | 点击“获取持久化信息”。 | 成功展示步骤3中“持久化”输入框中输入的信息。 | 与结果一致 |  |
| 步骤 8 | “数据访问”应用已经成功打开。 | 无 | 点击“获取数据库信息”去访问本地websql。 | 成功获取到存储到websql输入框中输入信息以及历史输入的信息。 | 与结果一致 |  |

1. 文本解析功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 文本解析功能/ GN\_YYKJ\_JS\_WBJX | | | | |
| 用例说明 | | **测试目的：**提供文本解析能力，支持在界面程序中通过javascript语言调用本地文本解析库对ini、xml、Json等常见文本格式进行解析。  **测试方法：**将ini、xml、json等文件上传到调用本地文本解析库的应用中，应用可以对文件进行解析。  **合格判据：**程序可以正常调用本地文本解析库，对ini，xml，json等常见文本格式的文件进行解析。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户已成功安装“文本解析”应用。 | 用户名、密码等登录信息。 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 成功访问 | - |
| 步骤 2 | 在已安装的应用列表中可以看到“文本解析”应用。 | 无 | 在已安装的应用列表，打开“文本解析”应用。 | 成功打开“文本解析”应用。 | 成功打开应用 | - |
| 步骤 3 | “文本解析”应用已成功打开，已有相关格式文本文件。 | json文件。 | 在“文本解析”应用页面中，点击文件选择输入框并选择json文件，并点击上传。 | 提示上传成功，页面上成功展示解析后的json的内容。 | 成功上传信息 | - |
| 步骤 4 | “文本解析”应用已成功打开，已有相关格式文本文件。 | xml文件。 | 在“文本解析”应用页面中，点击文件选择输入框并选择xml文件，并点击上传。 | 提示上传成功，页面上成功展示解析后的xml的内容。 | 成功上传信息 | - |
| 步骤  5 | “文本解析”应用已成功打开，已有相关格式文本文件。 | ini文件。 | 在“文本解析”应用页面中，点击文件选择输入框并选择ini文件，并点击上传。 | 提示上传成功，页面上成功展示解析后的ini的内容。 | 与预期结果一致 |  |

1. 日志处理功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 日志处理功能/ GN\_YYKJ\_JS\_RZCL | | | | |
| 用例说明 | | **测试目的：** 测试软件是否具有日志处理能力，使界面程序可以通过javascript语言调用本地日志处理库将日志记录到本地文件。  **测试方法：**打开一个使用javascript语言调用本地日志处理库的应用，将记录的日志输出到本地文件。  **合格判据：**程序可以正常调用本地日志处理库，将日志记录到本能地文件。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户已成功“日志处理”应用。 | 用户名、密码等登录信息。 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 与预期结果一致 | - |
| 步骤 2 | 在已安装的应用列表中可以看到“日志处理”应用。 | 无 | 在已安装的应用列表，打开“日志处理”应用。 | 成功打开“日志处理”应用。 | 与预期结果一致 | - |
| 步骤 3 | “日志处理”应用已成功打开。 | 无 | 在“日志处理”应用页面中，点击“日志输出”按钮，将日志输出。 | 页面上展示成功输出日志的路径，访问日志路径，可以看到日志文件中有输出的日志内容。 | 与预期结果一致 | - |

1. 图像处理功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 图像处理功能/ GN\_YYKJ\_JS\_TXCL | | | | |
| 用例说明 | | **测试目的：** 测试软件是否支持图像处理功能，支持界面程序中通过javascript和html语言进行常用的canvas图像处理。  **测试方法：**打开一个使用javascript语言和html语言进行canvas绘图的界面应用程序，可以进行正常绘图。  **合格判据：**具备图像处理能力，界面程序可以通过javascript和html语言进行常用的canvas图像处理。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户已成功安装“图像处理”应用。 | 用户名、密码等登录信息。 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 与结果一致 | - |
| 步骤 2 | 在已安装的应用列表中可以看到“图像处理”应用。 | 无 | 在已安装的应用列表，打开“图像处理”应用。 | 成功打开“图像处理”应用。 | 与结果一致 | - |
| 步骤 3 | “图像处理”应用已成功打开。 | 无 | 在“图像处理”应用页面中，使用鼠标在空白画布上绘图。 | 成功使用鼠标在画板上绘图。 | 与结果一致 | - |

1. 协议处理功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 协议处理功能/ GN\_YYKJ\_JS\_XYCL | | | | |
| 用例说明 | | **测试目的：** 测试软件是具备协议处理能力，支持在界面程序中通过javascript语言调用本地协议处理库，支持对xml、http等格式的数据协议进行处理。（△△△（同步在需求中补充用例、用例与建模）用例中缺少了protobuf）  **测试方法：**打开应用了本地协议处理库的程序，发送相关协议封装的数据并进行协议解析。  1支持在界面程序中通过javascript语言调用本地协议处理库对xml协议的数据进行解析，并显示参数信息。  2支持在界面程序中通过javascript语言调用本地协议处理库对http协议的数据进行解析，并显示参数信息。  **合格判据：**  1 JavaScript客户端应用框架提供通过javascript语言调用本地协议处理库对xml协议进行处理的能力，可以显示xml协议中的参数。  2 JavaScript客户端应用框架提供通过javascript语言调用本地协议处理库对http协议进行处理的能力，可以显示http协议中的参数 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户已成功安装“协议处理”应用。 | 用户名、密码等登录信息。 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 与预期结果一致 | - |
| 步骤 2 | 在已安装的应用列表中可以看到“协议处理”应用。 | 无 | 打开“协议处理”应用。 | 成功打开应用。 | 与预期结果一致 | - |
| 步骤 3 | “协议处理”应用已成功打开。 | 无 | 点击“发送http请求”。 | 应用页面上显示本调用http请求结果。 | 与预期结果一致 |  |
| 步骤 4 | “协议处理”应用已成功打开。 | 无 | 在“协议处理”应用页面中，点击文件选择输入框并选择xml文件，并点击上传。 | 提示上传成功，页面上成功展示解析后的xml的内容。 | 与预期结果一致 |  |

1. javascript高级API接口测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | javascript高级API接口/GN\_YYKJ\_JS\_GJJK | | | | |
| 用例说明 | | **测试目的：** 规范统一的javascript高级API接口，对界面集成、数据传输、数据访问、文本解析、日志处理、图像处理、协议处理等方面的能力进行统一规范  **测试方法：** 检查javascript规范，查看是否对界面集成、数据传输、数据访问、文本解析、日志处理、图像处理、协议处理等方面的能力进行统一规范。  **合格判据：** 规范中对界面集成、数据传输、数据访问、文本解析、日志处理、图像处理、协议处理等方面的能力进行了统一规范。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已经提供javascript规范 | 无 | 检查javascript规范 | javascript规范中对对界面集成、数据传输、数据访问、文本解析、日志处理、图像处理、协议处理等方面的能力进行了统一规范 | 与结果一致 | - |

1. 稳定运行功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 稳定运行/GN\_YYKJ\_JS\_WDYX | | | | |
| 用例说明 | | **测试目的：**测试框架支持7\*24小时连续稳定运行。  **测试方法：** 启动并访问javascript客户端应用框架可以正常访问，在7\*24小时后再次访问，可以正常访问。  **合格判据：**框架支持7\*24小时连续稳定运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行。 | 用户名、密码等登录信息。 | 用户访问JavaScript客户端应用框架。 | 成功访问JavaScript客户端应用框架。 | 与预期结果一致 | - |
| 步骤 2 | 已经持续运行了很长时间 | 无 | 在7\*24小时后再次访问javascript客户端应用框架。 | 成功访问JavaScript客户端应用框架。 | 与预期结果一致 | - |

### C++服务端应用框架测试（李磊）

1. 模块集成功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 模块集成功能/ GN\_YYKJ\_CPP\_MKJC | | | | |
| 用例说明 | | **测试目的：** 测试插件集成框架的模块集成能力。  1.支持以插件的方式对软件模块进行集成，支持动态加载插件。  2.支持对插件的生命周期进行管理。  **测试方法：**  1.模块集成：C++服务端应用框架能都发现模块目录下的指定格式的应用模块插件，并将其加载至C++服务端应用框架中。  2.生命周期管理：C++服务端应用框架可以对应用模块插件生命周期管理，可对应用模块插件进行加载和卸载。  3.服务注册与发现：应用模块插件可向C++服务端应用框架注册服务，亦可以从C++服务端应用框架发现所需要的服务。  **合格判据：**  1.C++服务端应用框架可将相关的用户插件模块集成至C++服务端应用框架。  2.C++服务端应用框架可以对应用模块插件生命周期管理，可对应用模块插件进行加载和卸载。  3.应用模块插件可向C++服务端应用框架注册服务，亦可以从C++服务端应用框架发现所需要的服务。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经布置，并且仓库中已包含C++服务端应用框架镜像。  2.应用沙箱工具已正常运行。 | 应用沙箱命令行命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取C++服务端应用框架镜像（sandboxCli pull cpp\_rtf\_xx）；拉取完毕通过应用沙箱命令（sandboxCli images）查看拉取镜像；确定拉取成功后通过应用沙箱命令（sandboxCli start cpp\_rtf\_xx）启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 应用沙箱工具已正常运行。 | 应用沙箱命令行命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，将应用模块插件目录拷贝至C++服务端应用框架指定目录（/opt/pluginPlatform/）下，并在应用启动配置中添加需要依赖的C++服务端应用框架模块，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 与预期结果一致 | - |
| 步骤 3 | 1.应用模块插件实现C++服务端应用框架相关接口，并具备相关配置文件。  2.应用沙箱工具已正常运行。 | 进入相关目录，启动用户程序。 | C++服务端应用框架发现应用启动配置项中的所需要的模块插件，并将其加载至C++应用端服务框架中，可通过相关日志查看已加载至C++服务端应用框架中的模块插件。 | C++服务端应用框架成功加载用户应用模块插件。 | 与预期结果一致 | - |
| 步骤 4 | 应用沙箱工具已正常运行。 | 用户程序输入。 | C++服务端应用框架通过调用应用模块插件继承自C++服务端应用框架的纯虚函数（start，stop）对应用模块框架进行生命周期管理。可通过日志查看已加载至C++服务端应用框架中的模块插件的加载卸载。 | 可成功通过start函数和stop对用户模块插件进行生命周期管理。 | 与预期结果一致 | - |
| 步骤5 | 应用模块插件实现C++服务端应用框架相关接口，并具备相关配置文件。 | 用户程序输入。 | 应用模块插件通过C++服务端应用框架提供的接口向C++服务端应用模块注册相关服务，或通过相关接口发现C++服务端应用框架相关服务。 | 应用模块插件成功发现所需应用框架中的相关服务。 | 与预期结果一致 |  |

1. 多任务处理功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 多任务处理功能/ GN\_YYKJ\_CPP\_RWCL | | | | |
| 用例说明 | | **测试目的：** 测试C++服务端应用框架的多任务处理能力。  1.支持基于任务队列对服务请求进行处理。  2.支持基于多线程的方式对服务请求进行处理。  **测试方法：**  设计任务请求插件和任务处理插件，其中任务请求插件向任务处理插件发送任务请求，任务处理插件对任务请求进行处理并输出结果以及处理线程标识。  **合格判据：**  任务请求正常处理，输出结果表示不止一个线程处理。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经就绪。  2.应用沙箱工具已正常运行。 | 应用沙箱命令行命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取、查看和启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 启动本地C++服务端应用框架成功。 | 应用沙箱命令行命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，拷贝应用模块插件目录，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 成功进入C++服务端应用框架容器 | - |
| 步骤 3 | 在应用框架目录下成功生成启动应用相关目录。 | 进入相关目录，启动用户程序。 | C++服务端应用框架启动，已成功加载C++应用插件。 | 可通过相关日志查看到C++服务端应用框架已成功加载用户应用插件。 | 与预期结果一致 | - |
| 步骤 4 | 任务处理插件实现C++服务端应用框架相关接口。 | 任务请求插件，任务处理插件，任务处理请求。 | 任务处理插件通过之前获取的多任务模块接口，通过相关接口注册任务处理函数。  任务请求插件向任务处理插件发送任务处理请求。 | 任务请求插件成功向任务处理插件发送任务处理请求。 | 与预期结果一致 | - |
| 步骤 5 | 任务处理请求发送成功 | 无 | 任务处理插件将任务处理完成后返回处理结果。可通过相关日志查看任务处理结果，并输出任务处理线程的标识。 | 接收端处理完成后返回处理结果。 | 与预期结果一致 | - |

1. 数据传输功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 数据传输功能/ GN\_YYKJ\_CPP\_SJCS | | | | |
| 用例说明 | | **测试目的：** 提供数据传输插件。  1.支持基于tcp协议的数据传输。  2.支持基于http协议的数据传输。  **测试方法：**  通过设计发送端和接收端插件，实现对tcp、http协议数据的发送与接收。  **合格判据：**  tcp、http接收端插件可以接收和打印发送端发送的数据。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经就绪。  2.应用沙箱工具已正常运行。 | 应用沙箱命令行命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取、查看和启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 启动本地C++服务端应用框架成功。 | 应用沙箱命令行命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，拷贝应用模块插件目录，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 成功进入C++服务端应用框架容器 | - |
| 步骤 3 | 在应用框架目录下成功生成启动应用相关目录。 | 进入相关目录，启动用户程序。 | C++服务端应用框架启动，已成功加载C++应用插件。 | 可通过相关日志查看到C++服务端应用框架已成功加载用户应用插件。 | 与预期结果一致 | - |
| 步骤 4 | 应用模块插件实现C++服务端应用框架相关接口。 | 用户程序输入。数据传输请求，数据处理程序。 | 用户软件模块通过之前获取的数据传输模块接口，启动tcp协议的接收端和发送端，用户软件模块发送端通过传递参数以tcp协议对数据报文进行传输，用户软件模块接收端接收到报文之后向发送端返回接收结果。可通过相关日志查看任务处理结果。 | 发送端成功向接收端发送数据传输请求，接收端处理完成后返回处理结果。 | 与预期结果一致 | - |
| 步骤 5 | 应用模块插件实现C++服务端应用框架相关接口。 | 用户程序输入。数据传输请求，数据处理程序。 | 用户软件模块通过之前获取的数据传输模块接口，启动http协议的接收端和发送端，用户软件模块发送端通过传递参数以http协议对数据报文进行传输，用户软件模块接收端接收到报文之后向发送端返回接收结果。可通过相关日志查看任务处理结果。 | 发送端成功向接收端发送数据传输请求，接收端处理完成后返回处理结果。 | 与预期结果一致 | - |

1. 数据访问功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 数据访问功能测试/ GN\_YYKJ\_CPP\_SJFW | | | | |
| 用例说明 | | **测试目的：** 提供数据访问插件：支持对sqllite等典型数据文件进行访问。  **测试方法：** 支持对sqllite等典型数据文件进行访问：用户软件模块可通过C++服务端应用框架提供的数据访问模块对sqlite数据文件进行访问，对其进行增删查改等操作。  **合格判据：** 通过对数据访问模块的调用可以成功对sqlite等文件进行增删查改等操作。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经就绪。  2.应用沙箱工具已正常运行。 | 应用沙箱命令行命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取、查看和启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 启动本地C++服务端应用框架成功。 | 应用沙箱命令行命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，拷贝应用模块插件目录，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 与预期结果一致 | - |
| 步骤 3 | 在应用框架目录下成功生成启动应用相关目录。 | 进入相关目录，启动用户程序。 | C++服务端应用框架启动，已成功加载C++应用插件。 | 可通过相关日志查看到C++服务端应用框架已成功加载用户应用插件。 | 与预期结果一致 | - |
| 步骤 4 | 1. Sqlite相关文件。 | Sqlite相关文件。 | 用户软件模块通过之前获取的数据访问模块接口，通过加载公共运行库中的sqlite库，用户软件模块加载指定sqlite文件，并对其进行解析。 | 数据访问模块加载sqlite文件成功。 | 加载文件成功 | - |
| 步骤 5 | 1. 1.Sqlite相关文件。 2. 2.新增的数据项。 | Sqlite相关文件。 | 通过sqlite处理库在sqlite文件中新增指定数据项。可通过相关日志查看任务处理结果。 | 在sqlite文件中新增数据项成功。 | 与预期结果一致 | - |
| 步骤 6 | 1. 1.Sqlite相关文件。 2. 2.删除的数据项。 | Sqlite相关文件。 | 通过sqlite处理库在sqlite文件中删除指定数据项。可通过相关日志查看任务处理结果。 | 在sqlite文件中删除数据项成功。 | 与预期结果一致 | - |
| 步骤 7 | 1. 1.Sqlite相关文件。 2. 2.查找的数据项。 | Sqlite相关文件。 | 通过sqlite处理库在sqlite文件中查找指定数据项。可通过相关日志查看任务处理结果。 | 在sqlite文件中查找数据项成功。 | 与结果一致 | - |
| 步骤 8 | 1. 1.Sqlite相关文件。 2. 2.修改的数据项。 | Sqlite相关文件。 | 通过sqlite处理库在sqlite文件中修改指定数据项。可通过相关日志查看任务处理结果。 | 在sqlite文件中修改数据项成功。 | 与结果一致 | - |

1. 文本解析功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 文本解析功能/ GN\_YYKJ\_CPP\_WBJX | | | | |
| 用例说明 | | **测试目的：**提供文本解析插件：支持对ini、xml、Json等常见文本格式进行解析。  **测试方法：**  1. 用户软件模块可通过C++服务端应用框架提供的文本解析模块对ini格式的文本进行处理。  2. 用户软件模块可通过C++服务端应用框架提供的文本解析模块对xml格式的文本进行处理。  3. 用户软件模块可通过C++服务端应用框架提供的文本解析模块对json格式的文本进行处理。  **合格判据：**客户程序可以正常调用文本解析插件，对ini，xml，Json等常见文本格式的文件进行解析和操作。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经就绪。  2.应用沙箱工具已正常运行。 | 应用沙箱命令行命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取、查看和启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 启动本地C++服务端应用框架成功。 | 应用沙箱命令行命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，拷贝应用模块插件目录，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 与结果一致 | - |
| 步骤 3 | 在应用框架目录下成功生成启动应用相关目录。 | 进入相关目录，启动用户程序。 | C++服务端应用框架启动，已成功加载C++应用插件。 | 可通过相关日志查看到C++服务端应用框架已成功加载用户应用插件。 | 与结果一致 | - |
| 步骤 4 | 相关格式文本文件。 | ini文件。 | 通过文本处理插件相关接口对ini文件中指定数据项进行增删查改等操作。可通过相关日志查看任务处理结果。 | 应用框架文本处理插件对ini文件中相关数据项成功进行增删查改等操作。 | 与预期结果一致 | - |
| 步骤 5 | 相关格式文本文件。 | xml文件。 | 通过文本处理插件相关接口对xml文件中指定数据项进行增删查改等操作。可通过相关日志查看任务处理结果。 | 应用框架文本处理插件对xml文件中相关数据项成功进行增删查改等操作。 | 与预期结果一致 |  |
| 步骤 6 | 相关格式文本文件。 | json文件。 | 通过文本处理插件相关接口对json文件中指定数据项进行增删查改等操作。可通过相关日志查看任务处理结果。 | 应用框架文本处理插件对json文件中相关数据项成功进行增删查改等操作。 | 与预期结果一致 |  |

1. JY加解密服务测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JY加解密功能/ GN\_YYKJ\_CPP\_JYJJ | | | | |
| 用例说明 | | **测试目的：**提供加解密插件：提供JY加解密插件，支持对内存数据进行对称加密、非对称加密、摘要、签名、验证等JY加解密操作。  **测试方法：**  1. 应用程序调用Crypto库生成RSA公私钥接口，能生成公私钥。  2. 应用程序使用已生成的公私钥,调用Crypto库公钥加密和私钥解密接口，能对明文进行正确的加解密操作。  3. 应用程序调用Crypto库对文件签名接口和验证文件签名接口，能对文件进行正确的签名操作，能验证文件的签名。  **合格判据：**客户程序可以正常调用JY加解密插件，支持对内存数据进行对称加密、非对称加密、摘要、签名、验证等JY加解密操作。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经就绪。  2.应用沙箱工具已正常运行。 | 应用沙箱命令行命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取、查看和启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 启动本地C++服务端应用框架成功。 | 应用沙箱命令行命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，拷贝应用模块插件目录，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 与结果一致 | - |
| 步骤 3 | 在应用框架目录下成功生成启动应用相关目录。 | 进入相关目录，启动用户程序。 | C++服务端应用框架启动，已成功加载C++应用插件。 | 可通过相关日志查看到C++服务端应用框架已成功加载用户应用插件。 | 与结果一致 | - |
| 步骤 4 | Crypto库能被正常调用。 | Crypto动态库文件。 | 应用程序调用crypto库生成RSA公私钥接口，生成公私钥。 | 正常生成公，私钥。 | 与预期结果一致 | - |
| 步骤 5 | Crypto库能被正常调用。 | 提供被加解密的明文。 | 应用程序使用已生成的公私钥,调用Crypto库公钥加密和私钥解密接口，能对明文进行正确的加解密操作。 | 对明文能够进行正确的加解密操作。 | 与预期结果一致 |  |
| 步骤 6 | Crypto库能被正常调用。 | 提供被加解密的文件。 | 应用程序调用Crypto库对文件签名接口和验证文件签名接口，能对文件进行正确的签名操作，能验证文件的签名。 | 能对文件进行正确的签名操作，能验证文件的签名。 | 与预期结果一致 |  |
| 步骤 7 | Crypto库能被正常调用。 | 提供被加解密的明文文件。 | 应用程序调用Crypto库base64加密和解密接口，能对明文进行正确的加解密操作。 | 能对明文进行正确的加解密操作。 | 与预期结果一致 |  |
| 步骤 8 | Crypto库能被正常调用。 | 提供被加解密的明文文件。 | 应用程序调用Crypto库十六进制加密和解密接口，能对明文进行正确的加解密操作。 | 能对明文进行正确的加解密操作。 | 与预期结果一致 |  |
| 步骤 9 | Crypto库能被正常调用。 | 提供被加解密的明文文件。 | 应用程序调用Crypto库gzip相关接口，能对文件进行正确的压缩和解压操作。 | 能对文件进行正确的压缩和解压操作。 | 与预期结果一致 |  |
| 步骤 10 | Crypto库能被正常调用。 | 提供被调用的数据。 | 应用进程调用JY加解密库摘要接口，针对同一段数据多次调用该接口，可以得到同样的摘要。 | 针对同一段数据多次调用该接口，可以得到同样的摘要。 | 与预期结果一致 |  |

1. 数据库访问能力测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 数据库访问功能/ GN\_YYKJ\_CPP\_SJKF | | | | |
| 用例说明 | | **测试目的：**提供数据库访问插件，支持对达梦、金仓、神通等主流国产数据库提供接口一致的库访问能力，支持数据库连接池。  **测试方法：**  1. 用户软件模块可通过C++服务端应用框架提供的统一接口，对达梦数据库进行访问，操作。  2. 用户软件模块可通过C++服务端应用框架提供的统一接口，对金仓数据库进行访问，操作。  3. 用户软件模块可通过C++服务端应用框架提供的统一接口，对神通数据库进行访问，操作。  **合格判据：**提供达梦、金仓、神通等常用国产数据库以及文件数据库的C、C++等语言的访问接口。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经就绪。  2.应用沙箱工具已正常运行。 | 应用沙箱命令行命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取、查看和启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 启动本地C++服务端应用框架成功。 | 应用沙箱命令行命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，拷贝应用模块插件目录，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 与结果一致 | - |
| 步骤 3 | 在应用框架目录下成功生成启动应用相关目录。 | 进入相关目录，启动用户程序。 | C++服务端应用框架启动，已成功加载C++应用插件。 | 可通过相关日志查看到C++服务端应用框架已成功加载用户应用插件。 | 与结果一致 | - |
| 步骤 4 | 金仓数据启动成功。 | Unixodbc以及金仓数据库驱动程序启动成功。 | 通过c++服务端应用框架接口注册， 连接金仓数据库。对数据库进行增删改查操作。设置数据库连接参数，数据库连接池初始化成功。 | 连接金仓数据库成功，可以正确创建数据表， 可以对数据库进行正删改查操作。 可以对数据库连接池初始化成功。 | 与预期结果一致 | - |
| 步骤 5 | 达梦数据启动成功。 | Unixodbc以及达梦数据库驱动程序启动成功。 | 通过c++服务端应用框架接口注册， 连接达梦数据库。对数据库进行增删改查操作。设置数据库连接参数，数据库连接池初始化成功。 | 连接达梦数据库成功，可以正确创建数据表， 可以对数据库进行正删改查操作。 可以对数据库连接池初始化成功。 | 与预期结果一致 |  |
| 步骤 6 | 神通数据库启动成功。 | Unixodbc以及神通数据库驱动程序启动成功。 | 通过c++服务端应用框架接口注册， 连接神通数据库。对数据库进行增删改查操作。设置数据库连接参数，数据库连接池初始化成功。 | 连接神通数据库成功，可以正确创建数据表， 可以对数据库进行正删改查操作。 可以对数据库连接池初始化成功。 | 与预期结果一致 |  |

1. 日志处理功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 日志处理功能/ GN\_YYKJ\_CPP\_RZCL | | | | |
| 用例说明 | | **测试目的：** 提供日志处理插件，支持将日志记录到本地文件，支持调试、告警、错误、正常等日志类型。  **测试方法：**  1.用户软件模块可通过C++服务端应用框架提供的日志处理模块进行日志记录。  2.用户软件模块可通过C++服务端应用框架提供的日志处理模块对调试级日志进行记录。  3.用户软件模块可通过C++服务端应用框架提供的日志处理模块对警告级日志进行记录。  4.用户软件模块可通过C++服务端应用框架提供的日志处理模块对错误级日志进行记录。  5.用户软件模块可通过C++服务端应用框架提供的日志处理模块对信息级日志进行记录。  **合格判据：** 客户程序可以成功调用日志功能模块，支持将日志记录到本地文件，支持调试、告警、错误、信息等日志类型。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经就绪。  2.应用沙箱工具已正常运行。 | 应用沙箱命令行命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取、查看和启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 启动本地C++服务端应用框架成功。 | 应用沙箱命令行命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，拷贝应用模块插件目录，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 与预期结果一致 | - |
| 步骤 3 | 在应用框架目录下成功生成启动应用相关目录。 | 进入相关目录，启动用户程序。 | C++服务端应用框架启动，已成功加载C++应用插件。 | 可通过相关日志查看到C++服务端应用框架已成功加载用户应用插件。 | 与结果一致 | - |
| 步骤 4 | 已成功加载C++应用插件。 | 日志文件信息。 | 通过日志处理插件相关接口新增日志文件。 | 成功生成日志文件。 | 与结果一致 | - |
| 步骤 5 | 日志文件已存在。 | 调试级日志信息。 | 通过日志处理插件相关接口在日志文件新增调试级日志。 | 日志文件成功记录调试级日志。 | 日志记录成功 |  |
| 步骤 6 | 日志文件已存在。 | 警告级日志信息。 | 通过日志处理插件相关接口在日志文件新增警告级日志。 | 日志文件成功记录警告级日志。 | 与预期结果一致 |  |
| 步骤 7 | 日志文件已存在。 | 错误级日志信息。 | 通过日志处理插件相关接口在日志文件新增错误级日志。 | 日志文件成功记录错误级日志。 | 与预期结果一致 |  |
| 步骤 8 | 日志文件已存在。 | 信息级日志信息 | 通过日志处理插件相关接口在日志文件新增信息级日志。 | 日志文件成功记录信息级日志。 | 与预期结果一致 |  |

1. 协议处理功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 协议处理功能/ GN\_YYKJ\_CPP\_XYCL | | | | |
| 用例说明 | | **测试目的：** 提供协议处理插件，支持对protobuf、xml、http等格式的数据协议进行处理。  **测试方法：**  1.用户软件模块可通过C++服务端应用框架提供的协议处理模块对protobuf格式的数据协议进行处理。  2.用户软件模块可通过C++服务端应用框架提供的协议处理模块对xml格式的数据协议进行处理。  3.用户软件模块可通过C++服务端应用框架提供的协议处理模块对http格式的数据协议进行处理。  **合格判据：** 客户程序可以正常调用协议处理插件的API接口，成功对protobuf、xml、http等格式的数据协议进行处理。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经就绪。  2.应用沙箱工具已正常运行。 | 应用沙箱命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取、查看和启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 启动本地C++服务端应用框架成功。 | 应用沙箱命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，拷贝应用模块插件目录，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 与结果一致 | - |
| 步骤 3 | 在应用框架目录下成功生成启动应用相关目录。 | 进入相关目录，启动用户程序。 | C++服务端应用框架启动，已成功加载C++应用插件。 | 可通过相关日志查看到C++服务端应用框架已成功加载用户应用插件。 | 与结果一致 | - |
| 步骤 4 | C++ protobuf协议数据发送端与接收端测试应用启动， protobuf格式数据准备好。 | protobuf格式数据。 | 发送端发送protobuf格式数据，接收端通过协议处理插件接收到protobuf格式数据，并对protobuf格式文件进行解析操作。 | 发送端成功发送protobuf格式协议数据，接收端接收并成功解析。 | 与预期结果一致 | - |
| 步骤5 | C++ xml协议数据发送端与接收端测试应用启动， xml格式数据准备好。 | xml格式数据。 | 发送端发送xml格式数据，接收端通过协议处理插件接收到xml格式数据，并对xml格式文件进行解析操作。 | 发送端成功发送xml格式协议数据，接收端接收并成功解析。 | 与结果一致 |  |
| 步骤6 | C++ http协议数据发送端与接收端测试应用启动， http格式数据准备好。 | http格式数据。 | 发送端发送http格式数据，接收端通过协议处理插件接收到http格式数据，并对http格式文件进行解析操作。 | 发送端成功发送xml格式协议数据，接收端接收并成功解析。 | 与预期结果一致 |  |

1. 运维监控能力测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 运维监控功能/GN\_YYKJ\_CPP\_YWJK | | | | |
| 用例说明 | | **测试目的：** 提供运维监控插件，支持各个插件进行监控， 收集插件运行状态信息。  **测试方法：**  1.用户软件模块可通过C++服务端应用框架提供的http 协议获取c++服务端应用框架中各插件的运行状态信息。  2.用户软件模块可通过C++服务端应用框架提供的http 协议设置c++服务端应用框架中各插件的监控命令。  3.用户软件模块可通过C++服务端应用框架提供的http 协议获取c++服务端应用框架中各种处理的接口。  **合格判据：** 客户程序可以正常调用协议获取插件的运行状态信息。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经就绪。  2.应用沙箱工具已正常运行。 | 应用沙箱命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取、查看和启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 启动本地C++服务端应用框架成功。 | 应用沙箱命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，拷贝应用模块插件目录，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 与结果一致 | - |
| 步骤 3 | 在应用框架目录下成功生成启动应用相关目录。 | 进入相关目录，启动用户程序。 | C++服务端应用框架启动，已成功加载C++应用插件。 | 可通过相关日志查看到C++服务端应用框架已成功加载用户应用插件。 | 与结果一致 | - |
| 步骤 4 | C++ 运维监控插件， http server插件正常启动，运行。 | 输入http server默认url 网址。 | 通过默认url网址获取运维监控插件的相关url。 | 通过默认url成功获取运维监控模块的对应url。 | 与预期结果一致 | - |
| 步骤5 | C++ 运维监控url成功获取。 | 输入获取运维监控运行状态的url。 | 发送端发送json格式的url获取运维监控插件的监控数据。 | 发送端成功发送url格式协议数据，获取到运维监控插件的运行状态数据。 | 与结果一致 |  |
| 步骤6 | 通过控制url 向运维插件发送控制命令。 | http格式数据。 | 通过控制url发送json格式的控制数据给监控插件， 控制被监控插件做相应的控制操作。 | 成功发送json格式协议数据，触发被监控插件做相应的控制操作。 | 与预期结果一致 |  |

1. C++高级API接口测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | C++高级API接口/GN\_YYKJ\_CPP\_GJJK | | | | |
| 用例说明 | | **测试目的：** 规范统一的C++高级API接口，对模块集成、多任务处理、数据传输、数据访问、文本解析、JY加解密、数据库访问、日志处理、协议处理、运维监控等方面的能力进行统一规范  **测试方法：** 检查C++规范，查看是否对模块集成、多任务处理、数据传输、数据访问、文本解析、JY加解密、数据库访问、日志处理、协议处理、运维监控等方面的能力进行统一规范。  **合格判据：** 规范中对模块集成、多任务处理、数据传输、数据访问、文本解析、JY加解密、数据库访问、日志处理、协议处理、运维监控等方面的能力进行了统一规范。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已经提供C++规范 | 无 | 检查C++规范 | C++规范中对模块集成、多任务处理、数据传输、数据访问、文本解析、JY加解密、数据库访问、日志处理、协议处理、运维监控等方面的能力进行了统一规范 | 与结果一致 | - |

1. 稳定运行功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 稳定运行功能/ GN\_YYKJ\_CPP\_WDYX | | | | |
| 用例说明 | | **测试目的：**  支持C++运行时框架在7\*24小时的连续运行过程中，可以持续提供稳定的服务；保证系统长期运行的稳定性和健壮性。  **测试方法：** 编写模拟C++客户端稳定性测试程序不断发送网路调用请求，编写C++服务端稳定性测试程序，基于数据传输、协议处理与多任务处理等插件不断处理客户端发送来的请求，确保其长期运行不宕机，内存无泄漏。  **合格判据：**C++服务端稳定性测试程序不崩溃，内存占用没有直线上升，程序持续输出任务处理信息。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.软件仓库已经就绪。  2.应用沙箱工具已正常运行。 | 应用沙箱命令行命令pull，images，start。 | 通过应用沙箱命令（sandboxCli）拉取、查看和启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。 | 与预期结果一致 | - |
| 步骤 2 | 启动本地C++服务端应用框架成功。 | 应用沙箱命令行命令start，cp。 | 通过tty模式进入C++服务端应用框架镜像，拷贝应用模块插件目录，并生成启动应用相关目录。 | 成功进入C++服务端应用框架容器中，在应用框架目录下成功生成启动应用相关目录。 | 与预期结果一致 | - |
| 步骤 3 | 在应用框架目录下成功生成启动服务端稳定性测试程序相关目录。 | 进入相关目录，启动服务端稳定性测试程序。 | C++服务端稳定性测试程序启动。 | 可通过相关日志查看到C++服务端稳定性测试程序启动。 | 与预期结果一致 | - |
| 步骤4 | 客户端测试程序已经就绪。 | 客户端稳定性测试程序。 | 运行客户端稳定性测试程序，调用数据传输、协议处理与多任务处理运行时框架 API接口，反复处理网络请求，查看系统内存占用率等信息。 | 内存占用没有直线上升，服务端稳定性测试程序持续输出任务处理信息。 | 与预期结果一致 | - |
| 步骤 5 | C++服务端稳定性测试程序一直在运行。 | 客户端稳定性测试程序。 | 反复执行步骤4。 | C++服务端稳定性测试程序不崩溃，内存占用没有直线上升，程序持续输出任务处理信息。 | 与预期结果一致 | - |

### Java服务端应用框架测试（王强）

1. 模块集成功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 模块集成测试/ GN\_YYKJ\_JAVA\_MKJC | | | | |
| 用例说明 | | **测试目的:** 支持以插件的方式对软件模块进行动态集成。  **测试方法:** 提供Spring相关的Starter插件和基础库，通过加载到应用运行环境进行模块化集成。  **合格判据:** 能够根据应用需要从运行时框架中提取相应的应用模块，组成满足应用需要的运行时环境。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.Java服务端 应用框架已经在运行时框架中具备；  2.在开发机上部署沙箱工具和服务。 | 测试应用所需要的功能模块。 | 选择SpringCloud应用所需要的Spring Starter和基础库，把它们定义在应用工程pom.xml中。并在沙箱中对应用进行编译。 | 测试应用能够成功编译。 | 与预期结果一致 | - |
| 步骤 2 | 应用沙箱工具已正常运行。 | 应用沙箱命令行命令启动应用。 | 修改应用的配置文件，将测试应用在沙箱中启动。 | 测试应用能够成功运行，相关联的软件模块能够分别启动没有异常。 | 与预期结果一致 | - |
| 步骤 3 | 应用已经正常启动。 | 根据应用的设计需要进行相关模块检测。 | 查看日志检测应用使用应用框架所提供的各个模块启动状态，逐步测试每个模块的正确性。 | 应用动态集成的模块能够正常工作。 | 与预期结果一致 | - |

1. 流程集成功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 流程集成功能/GN\_YYKJ\_JAVA\_LCJC | | | | |
| 用例说明 | | **测试目的:** 测试运行时框架持通过注解、配置文件等方式进行流程定制，能动态集成事务处理、日志记录、异常处理、权限验证等流程节点。  **测试方法:** 基于运行时框架提供的Spring应用框架搭建测试应用，测试其通过配置或者注解的方式能够进行事务处理，日志处理，异常处理和权限验证等操作。  **合格判据:** 通过运行时框架提供的应用运行框架和基础库，能够实现测试应用的流程集成。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 运行时框架已经提供。 | 应用沙箱命令行命令启动应用命令。 | 修改spring配置文件，开启事务，看到相关的数据库访问操作已经在事务中管理。 | 事务的开启可以通过配置来实现。 | 与结果一致 | - |
| 步骤 2 | 测试应用成功运行。 | 配置修改。 | 修改spring配置文件，重启应用，调整日志的输出级别和其他特性。 | 日志的调整可以通过配置来实现。 | 与结果一致 | - |
| 步骤 3 | 测试应用成功运行。 | 修改配置开关。 | 修改spring配置文件关于异常处理部分，重启应用，可以打印显示某些异常信息，并进行特殊处理。 | 通过配置开关能够对异常进行特殊处理。 | 与预期结果一致 | - |
| 步骤 4 | 测试应用成功运行。 | 权限认证配置修改。 | 开启权限认证配置，重启测试应用，使满足指定条件的服务方法调用时必须经过权限认证。 | 相关的服务方法可以通过配置文件开启权限验证。 | 与预期结果一致 | - |

1. 数据传输功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 数据传输功能/GN\_YYKJ\_JAVA\_SJCS | | | | |
| 用例说明 | | **测试目的:** 支持基于tcp协议、http协议的数据传输。  **测试方法:**  1.支持基于tcp协议的数据传输：用户软件模块可通过Java服务端应用框架提供的数据传输基础库以tcp协议进行数据传输。  2.支持基于http协议的数据传输：用户软件模块可通过Java服务端应用框架提供的数据传输基础库以http协议进行数据传输。  **合格判据:**  1. JAVA服务端应用框架可以支持tcp协议对数据进行传输。  2. JAVA服务端应用框架可以支持http协议对数据进行传输。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.Java运行时框架及其内部的应用框架已经具备；  2.应用沙箱工具和服务已经在测试机上安装。 | 在应用框架中确定提供tcp服务的模块。 | 测试应用集成基于tcp服务的模块，向外提供一个tcp的服务。将测试应用进行编译。 | 测试应用能够正确集成tcp服务模块，能够正确编译。 | 与预期结果一致 | - |
| 步骤 2 | 测试应用已经正确编译。 | 应用沙箱命令启动测试应用。 | 配置相关的tcp端口和服务信息，使用沙箱正确启动测试应用。 | 成功在沙箱中启动基于应用框架的测试应用，并且检测提供的tcp服务正常。 | 成功在沙箱中启动基于应用框架的测试应用 | - |
| 步骤 3 | 1.Java运行时框架及其内部的应用框架已经具备；  2.应用沙箱工具和服务已经在测试机上安装。 | 在应用框架中确定提供http服务的模块。 | 测试应用集成基于http服务的模块，向外提供一个http的Restful服务。将测试应用进行编译。 | 测试应用能够正确集成http服务模块，能够正确编译。 | 与预期结果一致 | - |
| 步骤 4 | 测试应用已经正确编译。 | 应用沙箱命令启动测试应用。 | 配置相关的http端口和服务信息，使用沙箱正确启动测试应用，进行调用检测http的服务。 | 成功在沙箱中启动基于应用框架的测试应用，并且检测提供的http服务正常。 | 与结果一致 | - |

1. 数据访问功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 数据访问功能/GN\_YYKJ\_JAVA\_SJFW | | | | |
| 用例说明 | | **测试目的:** 支持对SQLite等典型数据文件进行访问。  **测试方法:** 支持对SQLite等典型数据文件进行访问：用户软件模块可通过Java服务端应用框架提供的数据访问模块对SQLite数据文件进行访问，对其进行增删查改等操作。  **合格判据:** 通过对数据访问模块的调用可以成功对SQLite数据库文件进行增删查改等操作。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Sqlite数据库已经启动；  应用框架提供数据库访问的ORMapping处理库。 | 在测试应用中集成数据库操作的库和操作脚本。 | 在测试应用中使用应用框架提供的对象关系映射库MyBatis操作，将SQLite操作的相关语句映射成与具体数据库无关的Java操作对象。  然后在沙箱中对应用进行编译。 | 使用对象关系配置可以将Java对象与具体的增删改查等SQLite语句映射。能够保证应用能够正确在沙箱中编译成功。 | 与预期结果一致 | - |
| 步骤 2 | 测试应用形成沙箱镜像。 | 启动应用的沙箱操作命令。 | 使用沙箱命令在测试机器上把数据库操作的测试应用在沙箱中启动。 | 测试应用在沙箱中正确启动。 | 测试应用正确启动 |  |
| 步骤 3 | 提供插入相关的SQLLite映射语句。 | 调用插入操作方法。 | 通过Java对象的插入方法处理在SQLite文件中新增指定数据项。可通过相关日志查看任务处理结果。 | 相关记录在SQLite数据库中插入。 | 与预期结果一致 | - |
| 步骤 3 | 提供删除相关的SQLLite映射语句。 | 调用删除操作方法。 | 通过Java对象的删除方法处理在SQLite文件中删除指定数据项。可通过相关日志查看任务处理结果。 | 相关记录在SQLite数据库中删除。 | 与预期结果一致 |  |
| 步骤 4 | 提供修改相关的SQLLite映射语句。 | 调用修改操作方法。 | 通过Java对象的修改方法处理在SQLite文件中修改指定数据项。可通过相关日志查看任务处理结果。 | 相关记录在SQLite数据库中变更。 | 与预期结果一致 |  |
| 步骤 5 | 提供查询相关的SQLLite映射语句。 | 调用查询操作方法 | 通过Java对象的查询方法处理在SQLite文件中查询指定数据项。可通过相关日志查看任务处理结果。 | 在SQLite数据库中相关记录能够被检索和日志打印界面显示。 | 与预期结果一致 |  |

1. 文本解析功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 文本解析功能/GN\_YYKJ\_JAVA\_WBJX | | | | |
| 用例说明 | | **测试目的:**提供文本解析基础库，支持对ini、xml、Json等常见文本格式进行解析。  **测试方法:**  1. 测试应用可通过Java服务端应用框架提供的文本解析库对ini格式的文本进行处理。  2. 用户软件模块可通过Java服务端应用框架提供的文本解析库对xml格式的文本进行处理。  3. 用户软件模块可通过 Java服务端应用框架提供的文本解库块对Json格式的文本进行处理。  **合格判据:** 客户程序可以正常调用文本解析插件，对ini，xml，Json等常见文本格式的文件进行解析和操作。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.Nexus仓库已经布置，并且仓库中已包含Java服务端应用框架镜像；  2.应用沙箱工具已正常运行 | 应用沙箱命令行命令进行测试应用的编译。 | 在测试应用工程中集成文本文件解析的相关的基础库，这些基础库通过应用框架提供。提供测试方法使用对应的库分别完成对ini, xml和json文件内指定数据项的增删改和文件的查询操作。对应用进行编译构建，形成沙箱镜像。 | 应用框架能够提供文本文件解析所需要的基础库和配置能力，测试应用能够正确在沙箱中进行编译。 | 与预期结果一致 | - |
| 步骤 2 | 在测试机上应用沙箱工具已正常运行。 | 沙箱启动命令。 | 配置相关的文本文件目录，使用沙箱命令启动测试应用程序。 | 应用能够正确启动，并可以支持执行下述的步骤。 | 应用正确启动 | - |
| 步骤 3 | 相关格式文本文件。 | ini文件。 | 测试程序中调用通过文本处理库相关接口对ini文件中指定数据项进行增删查改等操作。可通过相关日志查看任务处理结果。 | 应用框架文本处理插件对ini文件中相关数据项成功进行增删查改等操作。 | 与预期结果一致 | - |
| 步骤 4 | 相关格式文本文件。 | xml文件。 | 测试程序中调用通过文本处理xmlParser库相关接口对xml文件中指定数据项进行增删查改等操作。可通过相关日志查看任务处理结果。 | 应用框架文本处理插件对xml文件中相关数据项成功进行增删查改等操作。 | 与结果一致 |  |
| 步骤 5 | 相关格式文本文件。 | json文件 | 测试程序中调用通过文本处理Json库相关接口对json文件中指定数据项进行增删查改等操作。可通过相关日志查看任务处理结果。 | 应用框架文本处理插件对json文件中相关数据项成功进行增删查改等操作。 | 与预期结果一致 |  |

1. 数据库访问功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 数据库访问功能/GN\_YYKJ\_JAVA\_SJFW | | | | |
| 用例说明 | | **测试目的:** 支持对达梦、金仓、神通等主流国产数据库提供接口一致的库访问能力，支持数据库连接池。  **测试方法:** 统一使用Java服务端应用框架所提供的数据访问库包括MyBatis和DBUtils等工具，编写完整的数据库增删改查的操作程序；切换不同的国产数据库数据源，测试相关的数据库操作可以在不同的国产数据源中都能正确执行成功。  **合格判据:** 使用应用框架提供的基础库开发的测试应用可以在不同的国产数据源中都能正确执行成功。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 相关的国产数据库已经安装运行成功。 | 建表SQL语句。 | 准备工作，在不同的国产数据库中创建需要的数据库表结构。 | 在相应的数据库中能够成功创建出库表结构。 | 与预期结果一致 | - |
| 步骤 2 | 准备好测试应用，在测试机器上安装好沙箱工具，保持测试机与数据库网络通畅。 | 调用沙箱命令。 | 在外部配置达梦数据源，使用沙箱命令启动基于应用沙箱的测试应用，使应用的数据源切换到达梦数据库。 | 应用启动成功，应用连接达梦数据库成功。 | 应用启动成功 | - |
| 步骤 3 | 测试应用正常启动。 | 运行应用的测试脚本。 | 在测试应用中批量运行对数据库进行增删改查操作的测试方法，确保各个方法运行正常。 | 测试方法运行正常，相应的数据库操作体现到达梦数据库内的记录修改中。 | 与预期结果一致 | - |
| 步骤 4 | 测试机与数据库网络畅通。 | 调用沙箱命令。 | 在外部配置金仓数据源，使用沙箱命令启动基于应用沙箱的测试应用，使应用的数据源切换到金仓数据库。 | 应用启动成功，应用连接金仓数据库成功。 | 与结果一致 | - |
| 步骤 5 | 测试应用正常启动。 | 运行应用的测试脚本。 | 在测试应用中批量运行对数据库进行增删改查操作的测试方法，确保各个方法运行正常。 | 测试方法运行正常，相应的数据库操作体现到金仓数据库内的记录修改中。 | 与结果一致 | - |
| 步骤 6 | 测试机与数据库网络畅通。 | 调用沙箱命令。 | 在外部配置神通数据源，使用沙箱命令启动基于应用沙箱的测试应用，使应用的数据源切换到神通数据库。 | 应用启动成功，应用连接神通数据库成功。 | 与预期结果一致 | - |
| 步骤 7 | 测试应用正常启动。 | 运行应用的测试脚本。 | 在测试应用中批量运行对数据库进行增删改查操作的测试方法，确保各个方法运行正常。 | 测试方法运行正常，相应的数据库操作体现到神通数据库内的记录修改中。 | 与预期结果一致 | - |
| 步骤8 | 配置Druid数据库连接池 | 运行连接池测试脚本 | 在测试应用中从连接池中获取数据连接，批量运行对数据库进行增删改查操作的测试方法，确保各个方法运行正常。 | 测试方法运行正常，相应的数据库操作体现到Sqlite库内的记录修改中。 | 与预期结果一致 | - |

1. 日志处理功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 日志处理功能/GN\_YYKJ\_JAVA\_RZCL | | | | |
| 用例说明 | | **测试目的:** 支持将日志记录到本地文件。  **测试方法:** 用户应用通过使用运行时框架提供的日志库生成日志；用户通过日志配置文件调整日志的输出级别，文件位置和文件大小。  **合格判据:** 客户程序可以成功调用日志库，支持将日志记录到本地文件，支持调试、告警、错误、信息等日志类型。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 提供支持日志测试的用户应用。 | 应用沙箱命令行命令启动应用命令。 | 通过应用沙箱命令启动带有日志输出的测试应用，该应用依赖运行时框架。 | 应用启动成功，相关的日志文件和日志信息能够生成。 | 应用启动成功。 | - |
| 步骤 2 | 应用成功运行。 | 读取应用相关的配置文件。 | 查看应用的日志生成文件，看到指定目录下的日志文件已经生成，并切输出了不同级别的日志信息。 | 可以查看到相关的日志。 | 与预期结果一致。 | - |
| 步骤 3 | 应用成功运行，且已经输出日志信息。 | 调整日志配置文件 | 修改配置文件中的输出日志文件名称；输出日志的级别从DEBUG到TRACE。 | 相关的配置文件设置成功。 | 配置文件设置成功。 | - |
| 步骤 4 | 配置文件已经修改成功。 | 停止应用再次启动。 | 使用新的日志配置文件重新启动应用。 | 应用重启成功，相关的日志生成。 | 应用重启成功。 | - |
| 步骤 5 | 应用重启成功。 | 查看日志文件。 | 在新配置的目录下查看日志文件和日志内容。 | 能够找到日志文件，并且日志信息按照配置打印相应级别的日志。 | 与预期结果一致。 |  |

1. 协议处理功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 协议处理功能/GN\_YYKJ\_JAVA\_XYCL | | | | |
| 用例说明 | | **测试目的:** 支持对protobuf、xml、http等格式的数据协议进行处理。  **测试方法:**  1. 用户软件模块可通过Java服务端应用框架提供的grpc处理基础库对protobuf格式的数据协议进行处理。  2. 用户软件模块可通过Java服务端应用框架提供的xmlparser处理基础库对xml格式的数据协议进行处理。  3. 用户软件模块可通过Java服务端应用框架提供的servlet/http接口对http/html格式的数据协议进行处理  **合格判据:** 测试应用程序可以正常调用协议处理基础库的API接口，成功对protobuf、xml、http等格式的数据协议进行处理。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.Nexus仓库已经布置，并且仓库中已包含Java服务端应用框架镜像。  2.应用沙箱工具已正常运行。 | 应用沙箱命令行命令进行测试应用的编译。 | 在测试应用工程中集成不同格式数据协议处理的基础库，这些基础库可以通过应用框架提供。使用对应的库分别完成对protobuf, xml和http数据格式的内容进行解析和构造处理。对测试应用进行编译构建，形成沙箱镜像。 | 应用框架能够提供相应数据协议处理所需要的基础库和配置能力，测试应用能够正确在沙箱中进行编译。 | 与结果一致 | - |
| 步骤 2 | 在测试机上应用沙箱工具已正常运行。 | 沙箱启动命令。 | 配置相关的协议文件的目录，使用沙箱命令启动测试应用程序。 | 应用能够正确启动，并可以支持执行下述的步骤 | 与预期结果一致 | - |
| 步骤 3 | Protobuf格式协议的发送方和接收方程序准备好 | protobuf格式的协议数据。 | 测试应用软件通过协议处理库加载protobuf格式请求内容，在日志中显示，并构成protobuf格式的响应内容。 | 成功加载和解析protobuf格式的内容，并生成相应的应答格式内容。 | 与预期结果一致 | - |
| 步骤4 | xml格式协议的发送方和接收方程序准备好 | xml格式格式的协议数据。 | 测试应用软件通过协议处理基础库加载xml格式协议数据，并对xml格式协议数据进行增删查改等操作。 | 成功加载xml格式文件，并对其进行相关操作成功。 | 与预期结果一致 |  |
| 步骤5 | http格式协议的发送方和接收方程序准备好 | http格式格式的协议数据。 | 用户软件通过协议处理插件加载http/html格式协议数据，并对http格式协议数据进行请求读取和页面生成操作。 | 成功读取和生成http格式页面文件，并对其进行相关操作成功。 | 与预期结果一致 |  |

1. 配置和API接口功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 配置和API接口功能/GN\_YYKJ\_JAVA\_GJJK | | | | |
| 用例说明 | | **测试目的：** 规范统一的配置和API接口，对模块集成、流程集成、数据传输、数据访问、文本解析、数据库访问、日志处理、协议处理、运维监控、模块扩展等方面的能力进行规范  **测试方法：** 检查JAVA规范，查看是否对对模块集成、流程集成、数据传输、数据访问、文本解析、数据库访问、日志处理、协议处理、运维监控、模块扩展等方面的能力进行统一规范。  **合格判据：** 规范中对模块集成、流程集成、数据传输、数据访问、文本解析、数据库访问、日志处理、协议处理、运维监控、模块扩展等方面的能力进行了统一规范。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已经提供JAVA规范 | 无 | 检查JAVA规范 | JAVA规范中对模块集成、流程集成、数据传输、数据访问、文本解析、数据库访问、日志处理、协议处理、运维监控、模块扩展等方面的能力进行了统一规范 | 与结果一致 | - |

1. 稳定运行功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 稳定运行功能/GN\_YYKJ\_JAVA\_WDYX | | | | |
| 用例说明 | | **测试目的:** JAVA运行时框架程序连续稳定运行7\*24小时  **测试方法:** 保持基于Java服务端应用框架的测试应用服务长期运行,期间不间断地调用定时任务调用服务端提供的服务，确保服务运行期间每次调用都能成功。  **合格判据:** Java 运行时框架程序可以连续稳定运行7\*24小时，相关的服务能够稳定提供服务。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 测试机器上已经安装应用沙箱工具并启动沙箱服务。 | 应用沙箱命令行命令。 | 启动基于运行时应用框架的测试应用服务，保持长时间运行。 | 应用能够正确启动，所提供的服务正常。 | 应用正确启动 | - |
| 步骤 2 | Java运行时框架程序已经正确安装，正常运行，网络畅通。 | 稳定性测试程序。 | 启动基于crontab的定期探测程序。反复调用Java运行时框架提供的服务。 | 探测程序可以正常调用Java运行时框架提供的服务。 | 与预期结果一致 | - |
| 步骤 3 | 测试机器本身稳定运行。 | 稳定性测试程序。 | 定期查看服务器的运行日志和探测程序的执行结果。 | 确保再相当长的一段时间内，相关的日志有稳定和均匀的调用结果日志生成。 | 与预期结果一致 | - |

## 应用运行库/GN\_ YYXKC

### 图形界面库测试（王峰、（王强）、贺荣）

1. WxWidget库测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | WxWidget库/ GN\_YYXKC \_TXJM \_WK | | | | |
| 用例说明 | | **测试目的:**提供列表框、文本编辑、滚动条、菜单、按钮等常用控件。  **测试方法:** 执行WxWidgets应用程序，显示常用控件。  **合格判据:** WxWidgets库常用的界面控件正常显示及操作。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 沙箱工具正常安装，应用程序正常安装。 | 沙箱工具命令。 | 使用沙箱工具启动应用程序。 | 应用程序能够正常启动，常用控件界面显示正常。 | 与预期结果一致 | - |
| 步骤 2 | 测试应用已启动。 | 任意测试数据。 | 选择列表框中的某一项，通过键盘输入数据。 | 能够正常输入。 | 与预期结果一致 | - |
| 步骤 3 | 测试应用已启动。 | 任意测试数据。 | 选择文本框， 通过键盘输入数据。 | 能够正常输入。 | 与预期结果一致 | - |
| 步骤 4 | 测试应用已启动 | 无 | 查看滚动条状态，用鼠标拖拽滚动条。 | 窗口内容跟随变化。 | 与预期结果一致 | - |
| 步骤 5 | 测试应用已启动。 | 无 | 点击菜单项。 | 不同菜单项能实现不同功能。 | 与预期结果一致 | - |
| 步骤 6 | 测试应用已启动。 | 无 | 点击不同按钮。 | 能够实现不同功能。 | 与预期结果一致 | - |

1. SWT库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | SWT库的功能测试/GN\_YYXK\_TXJM \_SK | | | | |
| 用例说明 | | **测试目的:**支持Java语言的图形界面显示和处理能力。  **测试方法:** 通过应用界面测试正确显示窗体的按钮，标签，文本框，选择框，下拉框，菜单栏等用户组件；确保这些组件在一个框架容器中；支持常用的流式布局或网格布局，对组件进行布局管理；显示菜单栏、工具栏、弹出菜单等界面中间容器；支持各个组件的事件处理。  **合格判据:** Swing库能够正确在银河麒麟和深度操作系统、国产平台运行；显示组件、并支持布局摆放和正确做出事件相应处理。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 国产操作系统(桌面版)提供图形界面支持。 | 无 | 进入SWT应用程序，应用在窗体显示的按钮，标签，文本框，选择框，下拉框等用户组件，并确保这些组件在一个框架容器中。 | 正确的显示各个组件，并能进行相应的输入操作。 | 与预期结果一致 | - |
| 步骤 2 | 测试界面使用布局容器，对组件布局设计。 | 无 | 应用使用面板容器对用户组件进行布局管理，支持常用的流式布局或网格布局，检测布局不会因窗体的缩放而改变。 | 组件布局保持既定风格，显示不会错乱。 | 与预期结果一致 | - |
| 步骤 3 | 测试中提供菜单栏、工具栏、弹出菜单特殊窗体。 | 无 | 点击应用界面中的菜单栏、工具栏、弹出菜单特殊窗体，确保展现和功能正确，如在输入框总产生正确提示或弹出相应的对话框。 | 正确的产生提示信息或弹出窗口。 | 正确产生提示信息 | - |
| 步骤 4 | 测试应用中对相关的组件设置相应的事件。 | 无 | 检测相关组件事件处理能力，测试中检测首先给事件源注册监听器；然后使组件接受外部作用，事件被触发；之后组件产生一个响应的事件对象，并把此对象传递给与之关联的事件处理器；最后事件处理器启动，并执行相关的代码来处理该事件；查看相应组件的事件相应是否正确。 | 触发后，相关的事件被触发执行。 | 事件被触发执行 | - |

1. VUE库测试用例（贺荣）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | VUE库/ GN\_YYXK\_TXJM \_VK | | | | |
| 用例说明 | | **测试目的：** 测试软件支持使用VUE库的能力。（需要覆盖：提供列表框、文本编辑、滚动条、菜单等常用控件）  **测试方法：** 通过访问应用了VUE库中的图形组件应用，可以正常展示应用界面，并可以进行数据更新，来验证使用VUE库的能力。  **合格判据：** 软件支持使用VUE库绘制图形界面的能力。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户已成功安装VUE库应用。 | 用户名、密码等登录信息 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 与预期结果一致 | - |
| 步骤 2 | 在已安装的应用列表中可以看到VUE库应用 | 无 | 在已安装的应用列表，打开VUE库应用 | 成功打开VUE库应用，成功展示VUE库中的内容。 | 与预期结果一致 | - |

1. QT音视频播放测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT音视频播放/ GN\_YYXK\_TXJM\_QTK\_YSBF | | | | |
| 用例说明 | | **测试目的**: Qt能成功运行示例程序，显示音视频的播放。  **测试方法:** Qt服务对音频文件和视频文件进行解析，播放音视频。  **合格判据:** Qt服务可以运行QtMultimediaText示例程序且运行无报错。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Qt库及示例程序正常安装。 | 无 | 运行QtMultimediaText程序，运行成功弹出“音视频”窗口。 | 窗口上方存在视频播放小窗，存在“音频测试”按钮。 | 与结果一致 | - |
| 步骤 2 | 系统音频播放输出设备能够使用正常。 | 被测视频和音频 | 在窗口中播放视频和音频。 | 视频播放小窗可以正常播放视频，视频无花屏无卡顿现象；点击“音频播放”按钮，音频播放正常无杂音。 | 与结果一致 | - |

1. QT用户界面测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Qt用户界面/GN\_YYXK\_TXJM\_QTK\_YHJM | | | | |
| 用例说明 | | **测试目的:** Qt软件能成功运行实例程序，用户图形界面布局合理，可以实现用户交互体验。  **测试方法:** 对QT用户界面的标签类显示、下拉框测试、按钮测试、进度条测试选择框的基础功能进行交互性测试。  **合格判据:** Qt服务可以运行QtGuiText示例程序且运行无报错。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Qt库及示例程序正常安装。 | 无 | 运行QtGuiText程序，运行成功弹出“用户界面”窗口。 | 窗口界面布局合理，图片显示清晰不模糊，文字显示正常无乱码。 | 与预期结果一致 | - |
| 步骤 2 | QtGuiText程序成功运行。 | 无 | 点击“下拉框测试”。 | 右侧的下拉框按钮，下拉框显示“10、20、50、100”4个数值选项。 | 与预期结果一致 | - |
| 步骤 3 | QtGuiText程序成功运行，已点击“下拉框测试”。 | 无 | 分别点击下拉框数值。 | 可成功点击并切换数值。 | 与预期结果一致 | - |
| 步骤 4 | QtGuiText程序成功运行 | 无 | 点击“进度条测试”。 | 右侧的“进度条”也会随之改变相应的数值进度。 | 与预期结果一致 | - |
| 步骤 5 | QtGuiText程序成功运行，已点击“进度条测试”。 | 无 | 分别点击“按钮测试”，右侧的“10%、20%、50%、100%”按钮。 | 点击成功并且“进度条测试”右侧的“进度条”也会随之改变相应的数值进度。 | 与预期结果一致 | - |
| 步骤 6 | QtGuiText程序成功运行。 | 无 | 点击“单选框测试” 。 | 按钮可成功点亮，再次点击按钮亮度消失，遍历操作“复选框测试1、复选框测试2、复选框测试3”均可成功。 | 与预期结果一致 | - |

1. QT网络传输测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT网络传输/ GN\_YYXK\_TXJM\_QTK\_WLCS | | | | |
| 用例说明 | | **测试目的:** Qt软件能成功运行实例程序，通过TCP、UDP和HTTP等网络传输协议实现网络通信与数据传输。  **测试方法:** 1. Qt服务可以运行tcp服务端和tcp客户端的示例程序，实现tcp协议可靠性连接；  2. Qt服务可以运行udp接收端和udp发送端的示例程序，实现udp协议的数据发送与接收；  3. Qt服务可以运行http文本传输的示例程序，实现超文本协议数据传输。  **合格判据:** Qt服务可以运行tcp、udp、http示例程序且运行无报错。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Qt库、示例程序正常安装。 | 无 | 运行tcpserver服务端程序。 | tcp服务端运行成功；弹出tcp服务端窗口，显示“等待连接”提示。 | tcp服务端运行成功 | - |
| 步骤 2 | Qt库、示例程序正常安装。 | 无 | 运行tcpclient客户端程序。 | udp客户端运行成功，弹出udp客户端窗口，显示“主机”、“端口”等输入信息框。 | udp客户端运行成功 | - |
| 步骤 3 | tcpserver服务端、客户端程序已运行。 | IP地址：“127.0.0.1”端口：“6666”。 | 机框输入：“127.0.0.1”端口框输入：“6666”，点击“连接”按钮。 | tcp服务端和tcp客户端连接成功，显示“数据测试成功”。 | 与预期结果一致 | - |
| 步骤 4 | Qt库、示例程序正常安装。 | 无 | 运行udpreceiver接收端程序。 | udp接收端运行成功，弹出udp接收端窗口，显示“等待接收数据”提示。 | 与预期结果一致 | - |
| 步骤 5 | Qt库、示例程序正常安装。 | 无 | 运行udpsender发送端程序。 | udp发送端运行成功，弹出udp发送端窗口，显示“进行广播”按钮。 | 与结果一致 | - |
| 步骤 6 | udpreceiver接收端、发送端程序已运行。 | 无 | 点击“进行广播”按钮。 | udp发送端进行数据广播发送，udp接收端能成功接收数据，显示“UDP接收数据成功”。 | 与结果一致 | - |
| 步骤 7 | Qt服务测试http超文本传输协议系统网络连接正常。 | 被测试外网地址 | 使用ping命令测试任意外网地址。 | 系统网络连接正常。 | 与预期结果一致 | - |
| 步骤 8 | Qt库、示例程序正常安装。 | 无 | 运行HTTPtext程序。 | 成功弹出http服务窗口。 | 与预期结果一致 | - |
| 步骤 9 | HTTPtext程序已运行。 | 无 | 点击“http测试”按钮。 | 文字框中能接收到网站信息。 | 与预期结果一致 | - |

1. QT库-OpenGL图形绘制测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT库-OpenGL图形绘制测试/ GN\_YYXK\_TXJM\_QTK\_ TXHZ | | | | |
| 用例说明 | | **测试目的:** 对QWidget及QGLWidget进行相同的绘制操作并进行比较，验证成功实现OpenGL图形加速功能。  **测试方法:**  1.QWidget绘制操作；  2.QGLWidget绘制操作；  3.QWidget及QGLWidget并排展示绘制内容。  **合格判据:**  1.测试过程覆盖飞腾平台和龙芯平台。  2.测试过程和项目完整。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | qt4-default (>=4:4.8.7.1)已安装、qtcreator已安装。 | 无 | 运行“2dpainting”示例程序。 | 出现程序窗口。 | 出现程序窗口 | - |
| 步骤 2 | 已运行“2dpainting”示例程序。 | 无 | 程序调用QWidget及QGLWidget绘制图形。 | QWidget、QGLWidget分列窗口的两端，QWidget绘制内容具有抗锯齿效果。 | 与预期结果一致 | - |

1. QT库-文本通信测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT库-文本通信测试/ GN\_YYXK\_TXJM\_QTK\_ WBTX | | | | |
| 用例说明 | | **测试目的:** 通过DBUS方式，实现纯文本聊天的功能。  **测试方法:** 完成一对一文本报文传输功能。  **合格判据:**  1.测试过程覆盖飞腾平台和龙芯平台。  2.测试过程和项目完整。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 通信双方客户机上已安装qt4-default (>=4:4.8.7.1)，并已启动，通信双方客户机上已安装qtcreator，并已启动，通信网络畅通。 | 无 | 运行“chat”示例程序。 | 运行成功。 | 运行示例成功 | - |
| 步骤 2 | 已运行“chat”示例程序。 | 昵称信息。 | 在程序窗口输入昵称并继续。 | 输入成功 | 输入成功 | - |
| 步骤 3 | 已运行“chat”示例程序并输入昵称。 | 聊天信息。 | 再次运行本程序，重复上述过程，输入文本信息并发送。（多次重复运行本程序，在各聊天窗口输入文本信息，查看聊天窗口信息）。 | 在双方聊天记录窗口出现发送信息。 | 与预期结果一致 | - |

1. QT库-文本通信测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT库-Phonon多媒体测试/ GN\_YYXK\_TXJM\_QTK\_ DMT | | | | |
| 用例说明 | | **测试目的:** 实现多媒体文件播放和控制功能。  **测试方法:** 能够打开或播放音频文件，或异常终止能够返回错误信息。  **合格判据:**  1.测试过程覆盖飞腾平台和龙芯平台。  2.测试过程和项目完整。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | qt4-default (>=4:4.8.7.1)已安装、libphonon-dev、phonon-backend-gstreame已安装、qtcreator已安装。 | 无 | 运行“qmusicplayer”示例程序。 | 出现播放器窗口。 | 与预期结果一致 | - |
| 步骤 2 | 已运行“qmusicplayer”示例程序。 | 待播放音频文件。 | 添加需要播放的音频文件。 | 播放列表中出现文件有关信息。 | 与预期结果一致 | - |
| 步骤 3 | 已添加音频文件。 | 无 | 选择音频文件，点击播放按钮开始播放。 | 播放成功。 | 与预期结果一致 | - |
| 步骤 4 | 已添加音频文件。 | 无 | 点击播放器播放、停止、暂停等。 | 播放成功、播放停止、播放暂停。 | 与预期结果一致 | - |

### 文本处理库测试（余永华、（王强））

1. ini文本文件处理库测试用例（余永华）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Ini配置文件解析处理库/ GN\_YYXKC \_WBCL\_IK | | | | |
| 用例说明 | | **测试目的:**C语言配置文件解析库 — iniparser可以方便的对配置文件进行解析、添加、修改、删除等操作。  **测试方法:** 打开配置文件，进行读取解析，增加配置项，修改配置项，删除配置项，保存修改后的内容到新的文件中  **合格判据:** 能够正确解析、保存配置文件。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | iniparser库已部署。 | 进入测试程序文件夹执行命令./iniparser\_test，以及待解析的ini文件。 | 启动ini文件处理程序，加载ini配置文件，调用iniparser\_getstring()读取配置项，并分别调用iniparser\_set()新增或者修改ini配置项，iniparser\_unset()删除配置项。 | 能够解析、添加、修改、删除ini配置文件。 | 与预期结果一致 | - |
| 步骤 2 | 已调用ini文件处理程序。 | 无 | 调用iniparser\_dumpsection\_ini()保存ini到新的文件。 | 能够保存ini到新的文件。 | 与预期结果一致 | - |
| 步骤 3 | 已调用ini文件处理程序。 | 无 | 程序修改的项是否修改成功。 | 程序修改ini的项，与修改后的ini文件一致。 | 与预期结果一致 | - |

1. 快速文本处理库测试用例（余永华）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 快速文本处理库/ GN\_YYXK \_WBCL\_KSWB | | | | |
| 用例说明 | | **测试目的:** 验证快速文本处理库安装与常用接口调用是否正常；  **测试方法:** 编写测试用例测试程序，对json数据的封装、解析、修改、存储等接口进行调用；  **合格判据:** 应用程序成功执行，并且能返回正确结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署快速文本处理库。 | 无 | 执行json数据封装程序。 | 成功执行json数据封装程序。 | 与预期结果一致 | - |
| 步骤 2 | 已执行json数据封装程序。 | 待封装的json数据。 | 创建Json数据的头结点，并向其中添加Json数据，然后调用cJson\_Print()输出Json数据。 | 能正确封装Json数据和打印数据。 | 封装和打印数据 | - |
| 步骤 3 | 已正确部署快速文本处理库。 | 无 | 执行json数据解析程序。 | 成功执行json数据解析程序。 | 与结果一致 | - |
| 步骤 4 | 已执行json数据解析程序。 | 待解析的字符串。 | 调用cJSON\_Parse()解析字符串类型Json数据，根据键值对的名称从列表中取出对应的值。 | 能正确解析字符串类型Json数据并输出。 | 与结果一致 | - |
| 步骤 5 | 已正确部署快速文本处理库。 | 无 | 执行xml文档解析程序。 | 成功执行xml文档解析程序。 | 成功执行文件解析程序 | - |
| 步骤 6 | 已执行xml文档解析程序。 | 待解析的xml文档 | 调用xmlParseFile()解析xml文档，然后调用xmlNodeListGetString()获取节点内容，并输出。 | 正确解析xml文档并输出。 | 输出解析的文档 | - |
| 步骤 7 | 已正确部署快速文本处理库。 | 无 | 执行xml文档查询程序。 | 成功执行xml文档查询程序。 | 与预期结果一致 | - |
| 步骤 8 | 已执行xml文档查询程序。 | 待查询的xml文档 | 打开xml文档，使用Xpath表达式过滤xml文档内容的节点集，得到需要查询的信息并输出。 | 成功使用XPath表达式在复杂的xml文档中查询所需要的信息。 | 与预期结果一致 | - |
| 步骤9 | 已正确部署快速文本处理库。 | 无 | 执行xml文档修改程序。 | 成功执行xml文档修改程序。 | 与结果一致 | - |
| 步骤10 | 已执行xml文档修改程序。 | 待修改的xml文档 | 调用xmlNewTextChild()在指定节点指针上添加一个子元素，然后调用xmlSaveFormatFile()将修改后的文档写入到磁盘中。 | 能正确在xml文档中新增或删除指定元素。 | 与结果一致 | - |
| 步骤11 | 已正确部署快速文本处理库。 | 无 | 执行xml文档创建程序。 | 成功执行xml文档创建程序。 | 与预期结果一致 | - |
| 步骤12 | 已执行xml文档创建程序。 | 无 | 调用xmlNewDoc()创建xml文档，为设置文档的根结点，然后调用xmlAddChild()添加子节点，并设置子节点的内容和属性，最后调用xmlSaveFile()将xml文档存入文件。 | 成功创建xml文档。 | 成功创建文档 | - |
| 步骤13 | 已正确部署快速文本处理库。 | 无 | 执行xml文档编码修改程序。 | 成功执行xml文档编码修改程序。 | 与预期结果一致 | - |
| 步骤 14 | 已执行xml文档编码修改程序。 | 待转换的xml文档。 | 调用xmlCharEncodingHandlerPtr()创建编码处理器指针，调用xmlFindCharEncodingHandler()查找libxml2中指定的编码处理器，调用编码处理器的转换函数将数据转换成指定格式。 | 成功将xml文档转换成需要的编码格式。 | 与结果一致 | - |

1. 正则表达式文本处理库测试用例（余永华）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 正则表达式文本处理库/ GN\_YYXK \_WBCL\_ZZWB | | | | |
| 用例说明 | | **测试目的:** 验证正则表达式文本处理库安装与常用接口调用是否正常；  **测试方法:** 编写测试用例测试程序，对正则表达式库的版本确定、文本匹配等功能和接口进行调用；  **合格判据:** 应用程序成功执行，并且能返回正确结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署正则表达式文本处理库。 | 无 | 执行正则表达式文本处理库测试程序。 | 成功执行正则表达式文本处理库测试程序。 | 成功执行测试程序 | - |
| 步骤 2 | 已执行正则表达式文本处理库测试程序。 | 无 | 调用pcre\_version()获取库版本信息。 | 能正确获取并输出调用的库版本信息。 | 与结果一致 | - |
| 步骤 3 | 已执行正则表达式文本处理库测试程序。 | 待翻译的正则表达式字符串。 | 调用pcre\_compile()将字符串类型的正则表达式编译为库内部结构，加快匹配多个字符串时的匹配速度。 | 能成功执行并返回成功。 | 与预期结果一致 | - |
| 步骤 4 | 已执行正则表达式文本处理库测试程序。 | 无 | 调用pcre\_fullinfo()获取已编译好的模式信息。 | 能正确获取并输出编译好的模式信息。 | 获取并输出编译好的模式信息 | - |
| 步骤 5 | 已执行正则表达式文本处理库测试程序。 | 待匹配的字符串。 | 调用pcre\_exec()将字符串与编译好的模式进行匹配，返回匹配串的便宜位置。 | 能正确返回字符串的匹配结果。 | 与结果一致 | - |
| 步骤 6 | 已执行正则表达式文本处理库测试程序。 | 无 | 调用pcre\_free()释放不再需要已编译过的正则表达式，防止产生内存泄露。 | 成功释放不再需要的内存。 | 成功释放内存 | - |

1. pdf文本处理库测试用例（余永华）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | pdf文本处理库的功能测试/GN\_YYXK \_WBCL\_WBCL | | | | |
| 用例说明 | | **测试目的:**支持文本处理，提供加速pdf格式的文本交换和处理。  **测试方法:** 测试通过Pdf库能够生成的指定目录和名称的Pdf文档，测试可以按照指定的格式输出文本或者表格等到Pdf文档中。  **合格判据:** 能够在指定目录创建并打开Pdf文件；可以正确的向文件中添加文本或者表格等格式内容；能够正确地关闭Pdf文件。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已执行测试程序。 | 要写入的字符串。 | 打开测试程序，创建文档对象（Document）和与之对应的书写器（Writer），测试通过PdfWriter书写器对象把文档书写到磁盘中。 | 正确创建书写器，并能够向磁盘书写文件。 | 与结果一致 | - |
| 步骤2 | 已执行测试程序。 | 指定文件名和目录。 | 定义输出文件在磁盘中的目录和文件名称，打开该文件，并向其中书写内容。 | 能够将pdf文件输出到指定目录。 | 与预期结果一致 | - |
| 步骤 3 | 已执行测试程序。 | 输入相关的文字内容。 | 测试向文件中输入普通文字和表格。 | 查看文件内容满足设计要求。 | 与结果一致 | - |
| 步骤4 | 已执行测试程序。 | 无 | 关闭Pdf文件，查看和检查文件内容是否正确。 | 能够正确关闭文件，并切文件内容与期望结果一致。 | 与结果一致 | - |

1. JAVA其他文本处理库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JAVA其他文本处理库的功能测试/GN\_YYXK \_WBCL\_QITA | | | | |
| 用例说明 | | **测试目的:**支持文本处理，提供ini、xml。  **测试方法:** 测试通过ini和xml格式的文件的读取和增删改测试相关文件的处理能力。  **合格判据:** 能够在指定目录创建并打开ini,xml文件；可以正确的向文件中添加或修改内容。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已执行测试程序。 | 指定ini文件名和目录 | 打开测试程序，读取ini文件，查看相应的条目读取是否正确。 | 正确读取ini文件。 | 与结果一致 | - |
| 步骤2 | 已执行测试程序。 | 对ini文件增改删条目 | 测试向相应的条目增加数据，变更条目，删除原始的条目 | 正确的操作ini文件。 | 与预期结果一致 | - |
| 步骤 3 | 已执行测试程序。 | 指定xml文件名和目录 | 读取xml文件，获取element内容 | 正确读取xml文件。 | 与结果一致 | - |
| 步骤4 | 已执行测试程序。 | 无 | 对xml文件里的element单元进行添加、变更和删除操作。 | 正确的操作xml文件。 | 与结果一致 | - |
| 步骤5 | 已执行测试程序。 | 无 | 对串的查找比较使用正则表达式方法进行处理，查找模式匹配的串 | 正确的应用正则表达式 | 与结果一致 | - |

1. jsoncpp库测试用例（余永华）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | jsoncpp库/ GN\_YYXK \_WBCL\_JSK | | | | |
| 用例说明 | | **测试目的:** 验证jsoncpp库安装与常用接口调用是否正常；  **测试方法:** 编写测试用例测试程序，对Json库的数据串解析、对象序列化等等接口进行调用；  **合格判据:** 应用程序成功执行，并且能返回正确结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署jsoncpp库 | 无 | 执行json数据解析程序。 | 成功执行json数据解析程序。 | 与预期结果一致 | - |
| 步骤 2 | 已执行测试程序。 | 待解析字符串 | 调用Json::Reader类中的parse()将字符串数据解析为Json数据流。 | 能正确将字符串解析为Json数据流。 | 与预期结果一致 | - |
| 步骤 3 | 已执行测试程序。 | 无 | 调用Json::Value类中的asString()、asInt()获取Json对象。 | 能获取并输出Json数据流中的Json对象。 | 与结果一致 | - |
| 步骤 4 | 已正确部署jsoncpp库。 | 无 | 执行json序列化程序。 | 成功执行json数据解析程序。 | 与预期结果一致 | - |
| 步骤 5 | 已执行测试程序。 | 待序列化json对象 | 调用Json::Value类中的toStyledString()将Json对象序列化为字符串。 | 正确构建Json对象序列化为字符串并输出。 | 与结果一致 | - |
| 步骤 6 | 已正确部署jsoncpp库。 | 无 | 执行json数据写入文件程序。 | 成功执行json数据解析程序。 | 与结果一致 | - |
| 步骤 7 | 已执行测试程序。 | 待转换json对象 | 调用Json::FastWriter类中的write()将Json对象写为不带格式的字符流，调用Json::StyledWriter类中的write()或者Json::Value类中的toStyledString()将Json对象写为带格式的字符流。 | 正确将Json对象写为带格式或不带格式的字符流。 | 与预期结果一致 | - |

1. jackson库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Jackson库的功能测试/GN\_YYXK \_WBCL\_JAK | | | | |
| 用例说明 | | **测试目的:**支持文本处理，提供Json数据格式的处理。  **测试方法:** 测试通过Jackson库能够对简单和负载的Java结构对象进行序列化和反序列化处理；测试使用Jackson提供的注解，方便对JSON序列化和反序列化进行控制；测试支持通用的配置处理，包括异常和错误抛出或忽略、属性名单引号、双引号或无符号。  **合格判据:** 能够对Java对象进行JSON格式的序列化和反序列化；支持注解方式定义序列化属性；支持通过属性配置实现一些特殊配置。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 测试程序中构造需要转换的Java类对象。 | 待转换的Java对象。 | 使用对象映射ObjectMapper处理器，将Java类对象转换成字符串，在终端输出，或者保存到文本或者字节流文件中，测试转换后的串是否符合json格式。 | 正确的将Java类对象装换为字符串，在终端输出。 | 与预期结果一致 | - |
| 步骤 2 | 已经成功将对象装换为文件或字节数组或者字符串。 | 待转换的信息。 | 使用对象映射ObjectMapper从byte[]、File、InputStream、字符串等数据的JSON反序列化，测试可以正确进行反序列化处理。 | 把转换后的字符串转换回Java类对象，比较原始对象是否相等。 | 与预期结果一致 | - |
| 步骤 3 | 测试应用中使用Jackson提供的注解和属性设置。 | 设置不同属性。 | 使用Jackson提供的注解，对JSON序列化和反序列化进行控制，在Java类对象特殊属性中加入相应的注解，运行序列化和反序列化处理，并控制异常和错误抛出的行为。 | 控制输出结果的行为和显示格式。 | 与预期结果一致 | - |

1. FastJson库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | FastJson库的功能测试/GN\_YYXK \_WBCL\_FAK | | | | |
| 用例说明 | | **测试目的:**支持文本处理，提供Json数据格式的处理。  **测试方法:** 测试通过FastJson库能够对简单和负载的Java结构对象进行序列化和反序列化处理；测试使用FastJson提供的注解，方便对JSON序列化和反序列化进行控制；测试支持通用的配置处理，包括异常和错误抛出或忽略、属性名单引号、双引号或无符号。  **合格判据:** 能够对Java对象进行JSON格式的序列化和反序列化；支持注解方式定义序列化属性；支持通过属性配置实现一些特殊配置。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 测试程序中构造需要转换的Java类对象。 | 待转换的Java对象。 | 使用JSON 对象处理器，将Java类对象转换成字符串，在终端输出，或者保存到文本或者字节流文件中，测试转换后的字符串是否符合json格式。 | 正确的将Java类对象装换为字符串，在终端输出。 | 与预期结果一致 | - |
| 步骤 2 | 已经成功将对象装换为文件或字节数组或者字符串。 | 待转换的信息。 | 使用JSON 对象从byte[]、File、InputStream、字符串等数据的JSON反序列化，测试可以正确进行反序列化处理。 | 把转换后的字符串转换回Java类对象，比较原始对象是否相等。 | 与结果一致 | - |
| 步骤 3 | 测试应用中使用FastJson提供的注解和属性设置。 | 设置不同属性。 | 使用FastJson提供的注解，方便对JSON序列化和反序列化进行控制，测试可以在Java类对象中特殊属性中加入相应的注解，运行序列化和反序列化处理  测试能够支持通用的配置处理，包括异常和错误抛出或忽略、属性名单引号、双引号或无符号。 | 控制输出结果的行为和显示格式。 | 与结果一致 | - |

1. jsoup测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Jsoup库的功能测试/ GN\_YYXK \_WBCL\_JSOK | | | | |
| 用例说明 | | **测试目的:**支持Java语言的 Html 文档解析和处理能力。  **测试方法:** 通过测试程序测试Html 文档的正确解析，确保能获取到 Html 文档中元素的值  **合格判据:** Java 语言能够正确解析 Html 文档，并可获取 Html 数据。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 测试程序引入 Jsoup 库和标准的 Html 文档。 | Html 文档路径。 | 使用 Jsoup 的将 Html 文档解析为 Document 类。 | 正确的将 Html 文档解析为 Document 类。 | 与预期结果一致 | - |
| 步骤 2 | 已将 Html 文档解析为 Document 类。 | 无 | 使用document.getElementsByTag("body")获取元素对象Elements，通过Elements.text()获取元素的内容。 | 正确获取元素的内容。 | 获取元素内容 | - |

### 网络传输库测试（张金辉、（王强））

1. 可靠报文传输库测试用例（张金辉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 可靠报文传输库/ GN\_YYXKC \_WLCS\_BWCS | | | | |
| 用例说明 | | **测试目的:**测试国产平台下可靠报文传输库的能力是否正常。  **测试方法:** 初始化客户端和服务端，并在客户端进行同步、异步、投递三种方式的数据发送，确定数据都能到达服务端，最后结束化客户端和服务端；  **合格判据:** 所有报文传输功能都能正常完成。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 应用运行库已部署在通信双方，网络畅通。 | 服务端ip地址和端口，服务端回调通知函数。 | 服务端进程启动，调用LT\_Initialize函数初始化，注册会话建立、消息到达等回调函数,调用LT\_CreateService创建服务。 | 服务端创建服务成功。 | 创建服务成功 | - |
| 步骤 2 | 服务端创建服务成功。 | 服务端ip地址和端口，客户端回调通知函数。 | 客户端进程启动，调用LT\_Initialize函数初始化，注册会话建立、消息到达等回调函数,调用LT\_CreateSession创建会话。 | 客户端与服务端创建会话成功。 | 与预期结果一致 | - |
| 步骤 3 | 客户端与服务端创建会话成功。 | 同步报文。 | 客户端通过LT\_SynSendMsg同步发送消息，服务端的LT\_OnMsgReceive回调函数触发，处理完并产生回答消息，通过LT\_SendRes将回答消息发送给客户端。 | 客户端LT\_SynSendMsg返回成功。 | 与结果一致 | - |
| 步骤 4 | 客户端与服务端创建会话成功。 | 异步报文。 | 客户端通过LT\_AsynSendMsg异步发送消息并返回，服务端的LT\_OnMsgReceive回调函数触发，处理完并产生回答消息，通过LT\_SendRes将回答消息发送给客户端，客户端的LT\_OnResReceive回调函数被触发。 | 客户端的LT\_OnResReceive回调函数被触发。 | 与结果一致 | - |
| 步骤 5 | 客户端与服务端创建会话成功。 | 投递报文。 | 客户端通过LT\_PostMsg投递消息并返回，服务端的LT\_OnMsgReceive回调函数触发，处理完并产生回答消息，通过LT\_SendRes将回答消息发送给客户端，客户端的LT\_OnResReceive回调函数被触发。 | 客户端的LT\_OnResReceive回调函数被触发。 | 与预期结果一致 | - |
| 步骤 6 | 客户端与服务端创建会话成功。 | 无 | 客户端通过LT\_CloseHandle关闭与服务端会话，服务端的LT\_OnDisconnect回调函数触发，关闭本地会话。 | 服务端的LT\_OnDisconnect回调函数触发。 | 与预期结果一致 | - |

1. http报文传输库测试用例（张金辉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | http报文传输库/ GN\_YYXKC \_WLCS\_BWK | | | | |
| 用例说明 | | **测试目的:** http报文传输库作为是一个多协议的便于客户端使用的URL传输库，基于C语言，提供C语言的API接口，支持http及其他协议。  **测试方法:** 运行程序调用http报文传输库，访问http服务端，把服务端返回的报文，接收并打印到终端。  **合格判据:** 能够访问http服务器，并把报文显示到终端。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | libcurl库已部署。 | 进入测试程序文件夹，执行./lib1500 ip（http服务器地址）。 | 启动测试程序，并调用curl\_global\_init()，全局基础信息初始化libcurl，并调用curl\_easy\_init()创建一个简单的句柄。 | 成功启动测试程序。 | 成功启动测试程序 | - |
| 步骤 2 | 测试程序已启动。 | 无 | 调用curl\_easy\_setopt()设置回调函数，进行接收数据 ，curl\_easy\_setopt设置句柄属性，调用curl\_easy\_perform()提交数据，利用回调获取结果。 | 利用libcurl访问http服务，无法访问的地址给出错误提示，可以访问的地址展示出返回报文。 | 与预期结果一致 | - |
| 步骤 3 | 测试程序已启动。 | 无 | 调用curl\_easy\_cleanup()销毁句柄和调用curl\_global\_cleanup()销毁初始化libcurl信息。 | 成功销毁相关信息。 | 与预期结果一致 | - |
| 步骤4 |  | 执行./ftpupload，本示例运用了ftp服务 | 程序启动，curl\_global\_init()，全局基础信息初始化libcurl，并调用-curl\_easy\_init()创建一个简单的句柄 | 成功启动测试程序。 |  | - |
| 步骤5 | 测试程序已启动。 |  | 调用curl\_easy\_setopt()设置回调函数，会载入ftp服务地址和账密信息和需要上传的文件信息 ，curl\_easy\_setopt设置句柄属性，调用curl\_easy\_perform()提交数据，利用回调获取结果。 | 利用libcurl访问http服务，无法访问的地址给出错误提示，可以访问的地址展示出返回报文 |  | - |
| 步骤6 | 检查结果正确性 | 登录ftp服务器 | 检查上传文件的结果是否正确 |  |  | - |

1. libevent库测试用例（张金辉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | libevent库/ GN\_YYXKC\_WLCS\_LK | | | | |
| 用例说明 | | **测试目的:**libevent是一个用来开发可扩展的网络服务器的事件通知函数库。当一个文件描述符上的特定事件发生或是一个超时时间到达后，libevent API提供一种执行回调函数的机制。  **测试方法:** 测试运用libevent库编写的服务端，监听客户端，并向客户端收发消息，支持多个客户端。测试运用libevent库编写的客户端，连接服务端，并接收和回复服务端消息。  **合格判据:** 开启至少三个客户端进程，与服务端进行数据传输，确保数据准确性。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | libevent库已部署。 | 在测试程序文件夹中，执行./server命令。本示例采用tcp服务 | 启动服务端程序，开始监听客户端，程序中调用event\_base\_new()初始化event事件，并进行ip和端口监听，收到连接后，调用conn\_eventcb()处理客户发来的连接。然后调用conn\_readcb()来向客户端收发信息。 | 启动服务端监听，并能接收客户端发送的数据。 | 与预期结果一致 | - |
| 步骤 2 | 已启动服务端程序。 | 在测试程序文件夹中，./client命令 | 启动多个客户端程序，向服务端发送数据，客户端调用bufferevent\_socket\_connect()连接服务端，调用conn\_eventcb()处理与服务端连接，调用conn\_readcb()向服务端收发信息。 | 启动多个客户端，连接服务端，并向服务端发送数据，具有接收数据功能。 | 与预期结果一致 | - |
| 步骤 3 | 已完成数据收发。 | 校验数据。 | 检查服务端和客户端数据。 | 看服务端和客户端收发数据是否一致。 | 与预期结果一致 | - |
| 步骤4 | 启动udp服务端 | 执行./server\_udp | 调用event\_init()初始化event事件，调用bind\_socket()绑定UDP服务，调用event\_set()设置事件选项，调用event\_add()将event注册到event\_base的已注册事件链表中 | 启动udp服务端 | 服务端启动成功 | - |
| 步骤5 | 启动udp客户端 | 执行./client\_udp | 调用socket()设置udp句柄，配置好服务器地址，调用sendto()向udp服务端发送消息 | 启动udp客户端 | 客户端启动成功 | - |
| 步骤6 | 服务端验证消息接收结果 | 检测数据准确性 |  | 看结果是不是：i am from client！ | 与预期结果一致 | - |

1. websocket库测试用例（张金辉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | websocket库/ GN\_YYXK C\_WLCS\_WK | | | | |
| 用例说明 | | **测试目的:**WebSocket是一种在单个TCP连接上进行全双工通信的协议。允许服务端主动向客户端推送数据。在WebSocket API中，浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。  **测试方法:** 运用websocket库编写的服务端，提供http服务，从浏览器访问，可以把服务器中的资源展示到页面上。  **合格判据:** 浏览器访问WebSocket提供的api都能成功。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | libwebsockett库已部署，并且把测试资源拷贝到相应目录下。 | 在测试程序文件夹中，执行./ws\_server。 | 启动服务端程序，调用lws\_create\_context()创建服务器句柄，启动服务，并通过回调函数处理客户端发来的请求。 | 启动websocket服务端监听。 | 与预期结果一致 | - |
| 步骤 2 | 已启动服务端程序。 | 在测试程序文件夹中，执行./ws\_client。 | 启动服务端程序，调用lws\_client\_connect创建一个客户端连接，之后调用pthread\_routine()向服务端发送消息，并调用lws\_service()启动服务接收服务端发送的消息。 | 接收来自浏览器发送来的消息，并成功回应。 | 与预期结果一致 | - |

1. httpclient库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | HttpClient库的功能测试/GN\_YYXK \_WLCS\_HCK | | | | |
| 用例说明 | | **测试目的:** 提供客户端调用处理和登录安全处理能力。  **测试方法:** 测试通过Post/Get方式向服务端发送请求，并获取返回信息；测试同步和异步方式的请求。  **合格判据:** 检测Post和Get请求是否全覆盖；检测同步和异步请求场景；测试请求对象的构建和返回内容的解析；库的运行覆盖能够覆盖指标中的国产平台和国产操作系统。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 提供服务端测试http服务，并保证网路畅通。 | 连接池、线程池等配置信息。 | 进入测试程序后，客户端进程初始化，并配置连接池，线程池和连接超时等信息。 | 输出相关操作的提示日志。 | 输出日志内容 | - |
| 步骤 2 | 已启动测试程序。 | 无 | 创建请求方法的实例，并指定请求 URL。针对发送 GET 请求，创建 HttpGet 对象；针对发送 POST 请求，创建 HttpPost 对象。 | 提示相关的GET/POST对象创建成功。 | 与预期结果一致 | - |
| 步骤 3 | 已创建请求方法的实例。 | 设置不同的请求参数。 | 针对发送请求参数，可调用 HttpGet、HttpPost 共同的 setParams(HttpParams params) 方法来添加请求参数；对于 HttpPost 对象而言，也可调用 setEntity(HttpEntity entity) 方法来设置请求参数。 | 请求参数设置成功。 | 与结果一致 | - |
| 步骤 4 | 已创建请求方法的实例。 | 无 | 调用 HttpClient 对象的 execute(HttpUriRequest request) 发送请求。 | 返回请求的应答对象。 | 与预期结果一致 | - |
| 步骤 5 | 已创建请求方法的实例。 | 无 | 调用上一步中返回的HttpResponse 的 getAllHeaders()、getHeaders(String name) 等方法获取服务器的响应头；调用 HttpResponse 的 getEntity() 方法获取 HttpEntity 对象，该对象包装了服务器的响应内容。程序通过该对象获取服务器的响应内容。 | 获取应答对象内的各种信息。 | 与预期结果一致 | - |
| 步骤 6 | 已创建请求方法的实例。 | 无 | 释放连接。测试无论执行方法是否成功，都能够释放连接。 | 连接被成功释放。 | 与预期结果一致 | - |

1. netty库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Netty库的功能测试/GN\_YYXK \_WLCS\_NEK | | | | |
| 用例说明 | | **测试目的:** 提供特性通信功能处理和异步操作处理能力，支持客户端调用和服务端的通信服务。  **测试方法:** 测试应用启动服务端，在指定端口提供web服务；测试应用启动客户端，向服务端发送和接受请求；测试消息传输过程中的拆包和粘包处理；测试心跳和空闲处理。  **合格判据:** 服务端和客户端能力是否全覆盖；请求和应答消息的内容的责任链式处理正确；支持心跳和空闲检测能力；库的运行覆盖能够覆盖指标中的国产平台和国产操作系统。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Netty客户端和服务端框架库已经成功移植到国产平台。 | 线程组、IO模型等配置信息。 | 启动测试程序，进入测试程序后，服务端启动，配置线程组、IO模型、处理逻辑、绑定端口，在指定端口提供web服务。 | 服务端能够正确启动，在相关的端口提供服务。 | 与预期结果一致 | - |
| 步骤 2 | 已测试程序启动服务端。 | 请求地址和请求消息。 | 客户端启动，向服务端发送消息和接收消息，验证服务端和客户端双向通信。 | 发送请求被服务端接收，并由服务端返回期望的应答结果。 | 与预期结果一致 | - |
| 步骤 3 | 编写责任链处理事件。 | 构造好顺序的责任链。 | 通过责任链设计模式来组织代码逻辑，使用pipeline和channelHandler。 | 在每个责任链处理环节都有正确的处理。 | 结果一致 | - |
| 步骤 4 | 设计好执行调度处理逻辑。 | 无 | 重新连接，调 workerGroup 的 schedule 方法实现定时任务进行断网重连。 | 定时任务可以连接并执行。 | 与预期结果一致 | - |
| 步骤 5 | 构造数据传输载体的内容。 | 构造ByteBuffer并填充数据。 | 使用的ByteBuffer作为数据传输的载体，调用相关的API。 | 可以构造ByteBuffer和获取相关内容。 | 与预期结果一致 | - |
| 步骤 6 | 已启动服务端和客户端测试程序。 | 设置长度域拆包器。 | 拆包和粘包处理，使用基于长度域拆包器LengthFieldBasedFrameDecoder,添加到客户端和服务端pipeline中。 | 解决拆包和粘包问题。 | 结果一致 | - |
| 步骤 7 | 编写心跳检查的程序逻辑。 | 心跳周期。 | 心跳与空闲检测，Netty自带的IdleStateHandler让客户端每隔一段时间打个招呼，表示它还活着。 | 能够间歇性打印心跳处理的日志。 | 与预期结果一致 | - |
| 步骤8 | 编写基于UPD协议的Netty客户端和服务端程序 | 无 | 启动UDP服务端，查看服务是否启动 | UDP服务正确启动 | 与预期结果一致 |  |
| 步骤9 | 已完成步骤8 | 无 | 启动运行UDP客户端，向服务端进行通信，有相应的通信日志在服务端和客户端分别打印 | 客户端正确启动，并且相应的通信日志正确打印 | 与预期结果一致 |  |

### 安全处理库测试（魏冠杰、（王强））

1. JY加解密库测试用例（魏冠杰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JY加解密库/ GN\_YYXKC \_AQCL\_JYJM | | | | |
| 用例说明 | | **测试目的:**测试国产平台下JY加解密库接口与功能的正确性。  **测试方法:** 初始化库后，分别对数据的对称加解密、非对称加解密、数字签名与验证、数字信封与验证进行测试，最后结束化库；  **合格判据:** 返回正确码并输出操作结果，包括密文、明文、签名等信息。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已部署端密码设备模拟软件和JY加解密库并进行正确配置。 | 无 | 测试程序调用CSH\_Initialize初始化加解密库。 | 返回结果提示库初始化成功。 | 结果一致 | - |
| 步骤 2 | 库初始化成功。 | 数据明文。 | 测试程序初始化CSH\_OP结构为对称加密，并调用CSH\_Op\_M，对数据进行加密，然后初始化CSH\_OP结构为对称解密，并调用CSH\_Op\_M，对数据进行解密。 | 返回结果提示数据对称加密和解密均成功，输出密文，然后还原成为明文。 | 结果一致 | - |
| 步骤 3 | 库初始化成功。 | 数据明文。 | 测试程序初始化CSH\_OP结构为非对称加密，并调用CSH\_Op\_M，对数据进行加密，然后初始化CSH\_OP结构为非对称解密，并调用CSH\_Op\_M，对数据进行解密。 | 返回结果提示数据非对称加密和解密均成功，输出密文，然后还原成为明文。 | 与预期结果一致 | - |
| 步骤 4 | 库初始化成功。 | 数据明文。 | 测试程序初始化CSH\_OP结构为签名，并调用CSH\_Op\_M，对数据进行签名，然后初始化CSH\_OP结构为验证，并调用CSH\_Op\_M，对数据进行验证。 | 返回结果提示数据签名和验证均成功，输出带签名信息的明文，然后验证该签名正确。 | 与预期结果一致 | - |
| 步骤 5 | 库初始化成功。 | 数据明文。 | 测试程序初始化CSH\_OP结构为数字信封，，并调用CSH\_Op\_M，对数据进行数字信封算法处理，然后初始化CSH\_OP结构为验证，并调用CSH\_Op\_M，对数据进行验证。 | 返回结果提示数字信封和验证均成功，输出带数字信封处理信息的明文，然后验证该数字信封正确。 | 与预期结果一致 | - |
| 步骤 6 | 库初始化成功。 | 无 | 测试程序调用结束化函数CSH\_Finalize。 | 返回成功结束化的信息。 | 与预期结果一致 | - |

1. codec库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Codec库的功能测试/GN\_YYXK \_AQCL\_COK | | | | |
| 用例说明 | | **测试目的:** 提供对数据密文的处理。  **测试方法:** 对明文进行MD5加密签名；对明文进行SHA1加密；对明文进行Base64加密；对已经加密过得Base64串进行解密处理。  **合格判据:** 能够一致正确地进行MD5, SHA1加密；能够一致正确地进行Base64加密和解密。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已部署加解密库并进行正确配置。 | 无 | 启动测试程序，进入测试程序后，应用相应的codec库，启动解密库初始化接口，返回操作实例对象。 | 正确返回操作的实例对象。 | 与预期结果一致 | - |
| 步骤 2 | 已启动测试程序。 | 要处理。的明文 | 调用库中提供的MD5接口，对明文进行MD5加密签名。 | 正确地进行MD5加密处理。 | 与预期结果一致 | - |
| 步骤 3 | 已启动测试程序。 | 要处理的明文。 | 调用库中提供的SHA1接口，对明文进行SHA1加密。 | 正确地进行SHA1加密处理。 | 与预期结果一致 | - |
| 步骤 4 | 已启动测试程序。 | 要处理的明文。 | 调用库中提供的Base64接口，对明文进行Base64加密。 | 正确地进行Base64加密处理。 | 与预期结果一致 | - |
| 步骤 5 | 已启动测试程序。 | Base64加密过得串。 | 对已经加密过得Base64串，调用库中提供的Base64接口进行解密处理。 | 正确地进行Base64解密处理。 | 与预期结果一致 | - |
| 步骤 6 | 已启动测试程序。 | 重复调用。 | 重复上述的步骤，比对加解密中间过程的一致性。 | 多次运行结果保持一致。 | 与预期结果一致 | - |

1. crypto库测试用例（魏冠杰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | crypto库/ GN\_YYXKC \_AQCL\_CRK | | | | |
| 用例说明 | | **测试目的：**提供对文本和消息安全处理所需要的多种加解密算法的封装，提供对文本和消息安全处理所需要的多种加解密算法的封装，包括提供数据摘要、数字签名与验证、对称加密、非对称加密等能力。  **测试方法：**调用crypto各种加解密方法生成C++语言应用程序，执行应用程序，对各种加密方法进行预置的消息进行加解密操作并将结果输出终端  **合格判据：** 能对预置的功能进行操作验证，对预置的信息成功加解密。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已安装libcrypto++dev库，执行程序已经成功生成。 | 进入测试程序文件夹，执行命令./cryptest h。 | 查看应用程序的使用说明信息。 | 输出使用信息。 | 与预期结果一致 | - |
| 步骤 2 | 已安装libcrypto++dev库，执行程序已经成功生成。 | 进入测试程序文件夹，执行命令./cryptest g。 | 根据提示输入key长度1024，私钥文件名和公钥文件名。 | 成功在当前文件夹生成公私钥文件。 | 与预期结果一致 | 此处公私钥文件名和seed需要记录，方便后边用例使用信息解密的时候使用 |
| 步骤 3 | 已安装libcrypto++dev库，执行程序已经成功生成。 | 进入测试程序文件夹，执行命令./cryptest r。 | 根据提示输入私钥文件名，公钥文件名，seed随机值，和要加密的信息，必须使用之前生成公私钥文件名和seed值。验证非对称加密算法RSA加解密能力。 | 生成加密内容并输出加密和解密成功后的信息，检验库的非对称加密算法RSA加解密能力。 | 与预期结果一致，验证库有非对称加密算法RSA加解密能力。 | 必须使用之前生成公私钥文件名 |
| 步骤 4 | 已安装libcrypto++dev库，执行程序已经成功生成。 | 进入测试程序文件夹，执行命令./cryptest v。 | 启动测试程序，调用Crypto库的非对称加密算法RSA接口进行数据操作，能进行正确的对称加密和解密，能验证库的对称加解密能力 | 对预设信息成功使用所说非对称加解密算法进行数据的加密和解密，并在终端输出提示信息。验证库有非对称加解密能力。 | 与预期结果一致  ，验证库有非对称加解密能力。 | - |
| 步骤 5 | 已执行步骤4操作。 | 无 | 启动测试程序，调用Crypto库的对称加密算法接口（比如DES、AES算法）进行数据操作，能进行正确的对称加密和解密，能验证库的对称加解密能力 | 对预设信息成功使用所说对称加解密算法进行数据的加密和解密，并在终端输出提示信息。验证库有对称加解密能力。 | 与预期结果一致  ，验证库有对称加解密能力。 | - |
| 步骤 6 | 已执行步骤4操作。 | 无 | 启动测试程序，调用Crypto库签名及验证算法DSA接口对数据文件签名和验证，能进行正确的签名和验证操作，能验证文件的签名和验证能力 | 预设信息能成功使用数据签名算法并测试验证在终端打印输出，验证库的数据签名和验证能力。 | 与预期结果一致  ，验证库有数据签名和验证能力。 | - |
| 步骤 7 | 已执行步骤4操作。 | 无 | 启动测试程序，调用Crypto库的数据摘要算法接口（比如CRC-32、MD2、MD5、Tiger、RIPEMD-128、RIPEMD-160、RIPEMD-256、RIPEMD-320）进行数据操作，能进行正确的数据摘要验证，能验证库的数据摘要能力 | 对预设信息成功使用所说数据摘要算法进行数据验证，并在终端输出信息。验证库有数据摘要能力。 | 与预期结果一致  ，验证库有数据摘要能力。 | - |
| 步骤 8 | 已执行步骤4操作。 | 无 | 调用Crypto库其他算法接口（比如base64加解密，十六进制加解密，gzip文件解压缩等），能进行正确使用对应算法操作，能验证库的其他算法能力 | 对预设信息成功进行内置其他算法调用操作，并在终端打印输出操作提示信息。验证库有其他算法能力 | 与预期结果一致，展示库的其他算法能力。 | - |

### 多任务处理库测试（魏冠杰、（王强））

1. 多任务队列线程池库测试用例（魏冠杰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 消息队列线程池库/ GN\_YYXKC \_DRWK\_RWDL | | | | |
| 用例说明 | | **测试目的:**测试国产平台下多任务队列线程池库的接口和功能是否满足要求。  **测试方法:** 分别创建定时、银行、传统三种模式的线程池，对这三个线程池的任务队列中分派任务，观察任务是否被线程池正常执行；  **合格判据:** 定时、银行、传统三种模式下所有任务都能正常分派和执行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 多任务队列线程池库已部署好。 | 在指定执行路径下，执行./threadpooltest | 展示测试程序的使用参数，展示三种方式的测试使用方法 | 成功展示信息，-b表示银行模式方式测试方法，-tr表示传统方式测试方法，-ti表示定时方式测试方法 | 展示相应信息 | - |
| 步骤 2 | 多任务队列线程池库已部署好。 | 在指定执行路径下，执行./threadpooltest -b | 测试银行模式的线程池使用，程序会调用CreateBankPool，创建银行模式任务线程池。 | 创建完银行模式的线程池，等待下一步交互操作 | 等待终端交互 | - |
| 步骤 3 | 已执行2操作 | r | 测试程序调用PutMsg，往银行模式任务线程池任务队列中分派内置指定个数的任务。 | 分派任务并在终端展示调用操作信息 | 终端展示调用操作信息并等待下次交互 | - |
| 步骤 4 | 已执行3操作 | n次输入r | 测试程序调用PutMsg，会往银行模式任务线程池任务队列中分派内置指定个数\*n的任务。 | 分派任务并在终端n次展示调用操作信息 | 终端n次展示调用操作信息并等待下次交互 | - |
| 步骤 5 | 已执行4操作 | q | 测试程序调用DeletePool函数，销毁线程池对象。 | 销毁并退出程序 | 线程池对象成功销毁，退出程序 | - |
| 步骤 6 | 多任务队列线程池库已部署好。 | 在指定执行路径下，执行./threadpooltest -tr | 测试传统模式的线程池使用，程序会调用CreateTraditionalPool，创建传统模式任务线程池。 | 创建完传统模式的线程池，等待下一步交互操作 | 等待终端交互 | - |
| 步骤 7 | 已执行6操作 | r | 测试程序调用PutMsg，往传统模式任务线程池任务队列中分派内置指定个数的任务。 | 分派任务并在终端展示调用操作信息 | 终端展示调用操作信息并等待下次交互 | - |
| 步骤 8 | 已执行7操作 | n次输入r | 测试程序调用PutMsg，会往传统模式任务线程池任务队列中分派内置指定个数\*n的任务。 | 分派任务并在终端n次展示调用操作信息 | 终端n次展示调用操作信息并等待下次交互 | - |
| 步骤 9 | 已执行8操作 | q | 测试程序调用DeletePool函数，销毁线程池对象。 | 销毁并退出程序 | 线程池对象成功销毁，退出程序 | - |
| 步骤 10 | 多任务队列线程池库已部署好。 | 在指定执行路径下，执行./threadpooltest -ti | 测试定时模式的线程池使用，程序会调用testTimertaskPool，创建定时模式任务线程池。 | 创建完定时模式的线程池后，会按照设定任务进行工作，此处会打印信息 | 会定时打印信息 | - |
| 步骤 11 | 已执行10操作 | q | 测试程序调用DeleteTimerTaskPool函数，销毁线程池对象。 | 销毁并退出程序 | 线程池对象成功销毁，退出程序 | - |

1. Quartz库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Quartz库的功能测试/GN\_YYXK \_DRWL\_QK | | | | |
| 用例说明 | | **测试目的:** 支持对任务的调度和管理，支持Java语言。  **测试方法:** 能够定义定时任务；添加触发器和调度器；测试调度器到达定时执行周期后是否能够正确触发定时任务；测试定时任务与SpringBoot结合使用。  **合格判据:** 能够按照预定的执行时间和频率执行定时任务；b.能够与Springboot结合执行定时任务。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已启动测试应用。 | 设置重复执行周期。 | 应用使用触发器和调度器，从工厂中获取调度实例，创建任务实例，添加触发器，设置重复周期。 | 能够创建任务实例和触发器。 | 有创建成功相关信息输出 | - |
| 步骤 2 | 已启动测试应用。 | 关联任务。 | 将调度器与定时任务和触发器相关联，启动调度器，测试任务是否按照调度设置定期执行。 | 调度器能够执行，并且能够按周期执行任务。 | 与预期结果一致 | - |
| 步骤 4 | SpringBoot支持Quartz库。 | 标识相关的调度注解。 | 使用Springboot整合Quartz定时任务框架，测试相关的注解是否能够工作。 | 设置的任务可以按周期执行。 | 与预期结果一致 | - |
| 步骤 5 | SpringBoot支持相关的调度触发配置。 | 设置配置信息。 | 在SpringBoot中配置类，配置触发器与任务调度器，测试配置文件的属性修改是否可以影响定时任务的执行周期等。 | 可以按照配置的要求执行任务调度。 | 与预期结果一致 | - |

### 协议处理库测试（魏冠杰）

1. 自描述字典协议处理库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 自描述字典协议处理库/ GN\_YYXKC \_XYCL\_ZMSK | | | | |
| 用例说明 | | **测试目的:** 测试国产平台下自描述字典协议处理库的接口和功能是否满足要求。  **测试方法:** 创建自描述字典协议对象，扩展和修改字典对象内容，解析字典对象内容，序列化和反序列化字典对象；  **合格判据:** 创建、扩展、修改、解析、序列化和反序列化均成功。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 自描述字典协议处理库已部署好。 | 无 | 测试程序调用IM\_Create创建字典对象。 | 返回有效的字典对象句柄。 | 字典对象句柄有效 | - |
| 步骤 2 | 字典对象创建成功。 | 字典对象句柄。 | 测试程序调用IM\_SetStr、IM\_SetInt、IM\_SetData在字典对象中添加各类键值对数据。 | 返回结果提示键值对数据添加成功。 | 与结果一致 | - |
| 步骤 3 | 键值对数据添加成功。 | 字典对象句柄。 | 测试程序调用IM\_SetStr、IM\_SetInt、IM\_SetBool在字典对象中修改已经添加好的键值对数据。 | 返回结果提示键值对数据修改成功。 | 与预期结果一致 | - |
| 步骤 4 | 字典对象创建成功。 | 字典对象句柄。 | 测试程序调用IM\_GetAttrCount、IM\_GetAttrKeys、IM\_GetAttrType以及IM\_GetStr、IM\_GetInt等函数，对字典对象进行解析。 | 返回正确的字典对象内容。 | 与结果一致 | - |
| 步骤 5 | 字典对象创建成功。 | 字典对象句柄。 | 测试程序分别调用IM\_SerializeToXml、IM\_SerializeToBin、IM\_SerializeToJson函数，对字典对象进行序列化 | 返回结果提示序列化成功。 | 提示序列化成功 | - |
| 步骤 6 | 字典对象序列化成功。 | 字典对象序列化后的内存指针。 | 测试程序调用IM\_UnSerializeFromXml、IM\_UnSerializeFromBin、IM\_UnSerializeFromJson函数，对字典对象进行反序列化。 | 返回结果提示字典对象反序列化成功。 | 与预期结果一致 | - |
| 步骤 7 | 字典对象创建成功。 | 字典对象句柄。 | 测试程序调用IM\_Destroy函数，对字典对象进行销毁。 | 字典对象成功销毁。 | 与预期结果一致 | - |

1. 高性能紧凑协议处理库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 高性能紧凑协议处理库/ GN\_YYXKC \_XYCL\_GXNK | | | | |
| 用例说明 | | **测试目的:** 提供用户自定义数据结构基于protobuf协议的序列化与反序列化能力，对应用中的协议进行高效处理，支持不同的语言和类型。  **测试方法:** 使用C调用libprotobuf-c库，序列化和反序列化信息，使用反序列化后信息与序列化之前信息作比对，如果一致则测试成功，否则测试失败；使用C++语言调用libprotobuf库，控制台输入person的相关信息及当前时间写入到文本文件，之后执行另一个C++查看信息应用程序，从文件中读出之前存储的信息及信息存储的时间，信息与添加信息一致反序列化信息并打印，查看打印文件信息，信息中包含添加信息。  **合格判据:** 执行C应用程序，成功的序列化和反序列化结构体信息，反序列化后与序列化之前信息一致，说明C语言调用protobuf库用例测试成功；执行C++添加信息应用程序，控制台输入person的相关信息及当前时间写入到文本文件，执行C++查看信息应用程序，从文件中读出之前存储的信息及信息存储的时间，信息中包含之前添加数据。说明C++调用Protobuf库用例测试成功。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 使用构建生成的C语言protobuf-c\_test可执行程序。 | 进入测试程序文件夹，键入命令./protobuf-c\_test | 执行程序不报错，打印预期结果 | 输出test succeeded. | 与预期结果一致 | - |
| 步骤 2 | C++语言应用程序add\_person\_cpp已编译完成。 | 进入测试程序文件夹，执行./add\_person\_cpp 1111。 | 根据提示输入个人信息ID号、名字、邮箱、电话号等信息。程序内部会调用相应接口序列化成指定类型文件。 | 打印输入的信息，之后回车保存到文件。 | 与预期结果一致 | - |
| 步骤 3 | C++语言应用程序list\_people\_cpp应用程序已编译完成。 | 进入测试程序文件夹，执行./list\_people\_cpp 1111。 | 使用list\_people\_cpp程序，查看已经存储文件的信息。程序内部会调用相应接口反序列化成指定类型文件，还原成对应信息。 | 打印已经存储文件的信息。 | 与预期结果一致 | - |
| 步骤 4 | Java语言环境和测试程序已准备完毕。 | 启动执行Java环境下server端 | 启动Java程序服务端，等待响应Java程序客户端 | 已启动Java程序服务端 | 与预期结果一致 | - |
| 步骤 5 | 已执行步骤4 | 启动执行Java环境下clientr端 | 启动客户端，会远程调用对应的方法，并输出结果 | 成功与客户端调用并输出结果 | 与预期结果一致 | - |

1. http协议处理库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | http协议处理库/ GN\_YYXKC \_XYCL\_XYCL | | | | |
| 用例说明 | | **测试目的:** 提供对http协议的处理。  **测试方法:** 调用libhttp-parser库接口，对http请求报文集和返回报文集进行解析操作并打印结果。  **合格判据:** 解析请求数据集和返回数据集都无异常。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已安装libhttp-parser-dev开发库，并且测试应用程序已生成。 | 进入测试程序文件夹，键入命令./test\_fast。 | 扫描解析请求报文集和返回报文集中每一个可能的中断，解析器处理从套接字获取的的内容，并对数据进行全面解析。 | 能成功解析请求报文集和返回报文集数据中的头字段和值信息、内容长度信息、请求方法信息、响应状态码信息、HTTP版本信息、请求URL信息、消息体信息、传输编码信息、返回报文集异常等数据。 | 与预期结果一致 | - |

1. Thrift库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Thrift库/ GN\_YYXKC \_XYCL\_TK | | | | |
| 用例说明 | | **测试目的**: 提供对传输任务管理和传输协议管理的能力。  **测试方法**:调用thriftC++客户端，发送不同类型的数据至thriftC++服务端，服务端应用能够显示交互信息  **合格判据**:在上述各步骤中，服务端应用能够显示交互信息 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | C++客户端和C++服务端应用程序已成功生成。 | ./TestServer。 | 启动thrift C++服务端应用程序，等待服务端交互。 | 正常启动并提示服务端服务类型（simple），数据传输方式及数据格式(buffered/binary)和主机端口号9090信息，等待客户端响应 | 与预期结果一致 | - |
| 步骤 2 | C++服务端已启动，在另外一个窗口C++TestClient生成程序路径下。 | ./TestClient。 | 启动thrift C++客户端应用程序，与thriftC++客户端应用程序交互，客户端发送字符串类型（包括简单字符串和复杂字符串）、布尔类型（包括true和false）、字节流、整型，双精度浮点型、二进制、结构体、嵌套结构体、容器map、容器set、容器list、枚举类型、Typedef类型数据及多类型组合数据，和单种自定义异常、多个自定义异常等数据场景，观察客户端响应。 | thriftC++客户端应用程序能成功与thriftC++客户端应用程序交互，服务端能成功显示客户端交互信息。 | 与预期结果一致 | - |
| 步骤 3 | C语言客户端和C语言服务端应用程序已成功生成。 | ./test\_server。 | 启动thrift C服务端应用程序，等待thrift C客户端交互。 | 正常启动并提示服务端服务类型（simple），数据传输方式及数据格式(buffered/binary)和主机端口号9090等信息，等待客户端交互。 | 与预期结果一致 | - |
| 步骤 4 | C语言服务端已启动，在另外一个窗口C语言客户端test\_client生成程序路径下。 | ./test\_client。 | 启动thrift C客户端应用程序，与thriftC服务端应用程序交互，发送字符串、字节流、整型、浮点型、二进制，结构体，嵌套结构体、容器map、容器set、容器list、枚举、Typedef，组合数据、自定义异常数据等场景数据，观察服务端显示信息 | thriftC客户端应用程序与thriftC服务端应用程序成功交互，接收并显示交互信息 | 与预期结果一致 | - |
| 步骤  5 | JAVA服务端应用程序已成功生成。 | 运行沙箱执行Java服务端测试程序。 | JAVA服务端应用程序启动，等待客户端与之交互 | 成功启动JAVA服务端程序，等待客户端与之交互 | 与预期结果一致 | - |
| 步骤  6 | JAVA服务端应用程序已成功启动 | 在另外一个窗口下运行沙箱执行Java客户端端测试程序。 | JAVA客户端应用程序启动与JAVA服务端应用程序交互，客户端调用服务端方法，能成功调用并打印输出预期结果信息 | 成功启动JAVA服务端程序，等待客户端与之交互 | 与预期结果一致 | - |

### 数据库访问库测试（魏冠杰、（王强））

1. 对象关系映射数据库访问库测试用例（魏冠杰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 对象关系映射数据库访问库/ GN\_YYXKC \_SJFW\_DYGX | | | | |
| 用例说明 | | 测试目的: 提供基于对象关系映射文件的java对象访问数据库的能力。  测试方法: 配置映射文件，与Java对象进行SQL语句映射；配置通用的配置，与数据库和数据源向结合；应用进行对象的插入，变更，查询和删除的方法调用，相应的操作会映射到具体的SQL语句在数据库中执行；与Springboot结合使用的场景测试。  合格判据: 能够正确执行相关的SQL语句；能够获取和释放数据库连接；对象的操作能够反应的数据库的库表内容上。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 应用创建持久化Java类，并定义相关的Getter获取和Setter设置方法。 | 在映射文件中定义相关的SQL语句。 | 应用创建映射mapper文件，定义增删改查对应的SQL语句与持久化类的映射。 | 相关的文件能够正确定义，并能够符合MyBatis的加载要求。 | 与预期结果一致 | - |
| 步骤 2 | 确定相关的数据库服务和相关的连接信息。 | 数据源信息。 | 应用创建配置mybatis-config文件， 定义MyBatis与数据库和数据源的关联关系。 | 能够正确定义数据源信息，并且访问数据库没有问题 | 与结果一致 | - |
| 步骤 3 | 相关操作已经定义在操作对象中。 | 调用相关的操作对象。 | 应用进行对象的插入，变更，查询和删除的方法调用，相应的操作会映射到具体的SQL语句在数据库中执行。 | 对象的增删改查与数据库的操作联动。 | 与结果一致 | - |
| 步骤 4 | Springboot支持数据库操作配置。 | 配置相关的注解。 | 与SpringBoot结合，支持相关的配置，测试与数据源结合。 | 相关的配置正确，能够被Springboot加载。 | 与结果一致 | - |
| 步骤 5 | Springboot能够识别相关的MyBatis注解。 | 使用相关的注解。 | 与SpringBoot结合，测试支持相关的注解定义，代替SQL语句映射，测试对象的方法操作映射到相关的数据库操作。 | SpringBoot通过注解能够使用MyBatis的映射和操作处理。 | 与预期结果一致 | - |

1. 高性能数据库访问库测试用例（魏冠杰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 高性能数据库访问库/ GN\_YYXKC \_SJFW\_GXNK | | | | |
| 用例说明 | | **测试目的:** 测试高性能数据库访问库对达梦、金仓、神通等数据库的访问功能。  **测试方法:** 分别连接达梦、金仓、神通等数据库，测试数据库查询、修改、Blob读写等。  **合格判据:** 能够实现达梦、金仓、神通等数据库的正常查询、修改、 Blob读写等功能。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 高性能数据库访问库已正确部署。达梦、金仓、神通数据库已部署完毕。 | 启动环境方式连接池访问数据库程序 | 启动环境方式连接池访问数据库程序，连接池访问数据库 | 连接池访问达梦、金仓、神通能访问通过 | 与预期结果一致 | - |
| 步骤 2 | 已完成步骤1 | 启动驱动方式连接池访问数据库程序 | 启动驱动方式连接池访问数据库程序，连接池访问数据库 | 连接池访问达梦、金仓、神通能访问通过 | 与预期结果一致 | - |
| 步骤 3 | 已完成步骤1 | 执行金仓数据库对应程序 | 连接访问金仓数据库，能正确的对数据库内容进行修改和查询操作 | 正确的对数据库内容进行查询操作 | 与预期结果一致 | - |
| 步骤 4 | 已完成步骤1 | 执行金仓数据库对应程序 | 连接访问金仓数据库，能正确的对数据库BLOB类型内容进行插入和查询操作 | 正确的对数据库BLOB类型内容进行插入和查询操作 | 与预期结果一致 | - |
| 步骤 5 | 已完成步骤1 | 执行达梦数据库对应程序 | 连接访问达梦数据库，能正确的对数据库内容进行修改和查询操作 | 正确的对数据库内容进行查询操作 | 与预期结果一致 | - |
| 步骤 6 | 已完成步骤1 | 执行达梦数据库对应程序 | 连接访问达梦数据库，能正确的对数据库BLOB类型内容进行插入和查询操作 | 正确的对数据库BLOB类型内容进行插入和查询操作 | 与预期结果一致 | - |
| 步骤 7 | 已完成步骤1 | 执行神通数据库对应程序 | 连接访问神通数据库，能正确的对数据库内容进行修改和查询操作 | 正确的对数据库内容进行查询操作 | 与预期结果一致 | - |
| 步骤 8 | 已完成步骤1 | 执行神通数据库对应程序 | 连接访问神通数据库，能正确的对数据库BLOB类型内容进行插入和查询操作 | 正确的对数据库BLOB类型内容进行插入和查询操作 | 与预期结果一致 | - |

1. sqlite库测试用例（魏冠杰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | sqllite库/ GN\_YYXKC \_SJFW\_SK | | | | |
| 用例说明 | | **测试目的:** 提供sqlite数据库信息处理的能力。  **测试方法:** 测试库版本，测试库的常用操作。  **合格判据:** 数据库信息可以查询和子句查询。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已成功安装sqlite3库。 | sqlite3 –help。 | 查看sqlite3命令行操作提示信息。 | 输出操作提示信息。 | 与预期结果一致 | - |
| 步骤 2 | 已成功安装sqlite3库。 | sqlite3 –v。 | 命令行查看sqlite3版本信息。 | 输出sqlite3版本信息。 | 与预期结果一致 | - |
| 步骤 3 | 已成功安装sqlite3库。 | sqlite3回车，之后输入.help。 | 查看sqlite3命令行点命令可用清单信息。 | 先输出版本号信息，之后输出点命令的可用清单信息。 | 与预期结果一致 | - |
| 步骤 4 | 已成功安装sqlite3库，使用准备好的数据库。 | sqlite3 data\_src.db  之后输入.table查看数据库内存在表名。 | 打开已存在数据库（如果不存在会新建数据库），查看数据库内存在的表列表名称。 | 打开存在的数据库data\_src.db  查看存在表CVE-2020-9054。 | 与预期结果一致 | - |
| 步骤 5 | 已成功安装sqlite3库。 | sqlite3 data\_src.db  之后输入.dump。 | 打开已存在数据库，之后把全部表信息导出到终端。 | 全部整个数据库的sql语句信息输出到终端。 | 与预期结果一致 | - |
| 步骤 6 | 已成功安装sqlite3库，已进入data\_src.db数据库内。 | 输入.exit或者.quit  回车。 | 退出当前的数据库或者sqlite命令行终端。 | 成功退出当前的数据库或者sqlite命令行终端。 | 与预期结果一致 | - |
| 步骤 7 | 已成功安装sqlite3库 | sqlite3 data\_src.db  之后输入.dump。 | 打开已存在数据库，之后把全部表信息导出到终端。 | 整个数据库的sql语句信息输出到终端。 | 与预期结果一致 | - |
| 步骤 8 | 已成功安装sqlite3库。 | sqlite3 /tmp/data\_src.db ".dump" >/tmp/test.sql  之后执行  cat /tmp/test.sql。 | 导出全部表信息到文件（生成的sql语法文件任何数据库皆通用），之后查看文件内容。 | 导出成功，之后查看文件内容为整个数据库的sql语句信息。 | 与预期结果一致 | - |
| 步骤 9 | 已成功安装sqlite3库。 | sqlite3 data\_src.db  之后输入.header on  再输入.mode csv  格式转换完成后导出.output /tmp/test.csv select \*from"CVE-2020-9054";.output stdout  退出.exit之后查看文件test.csv。 | 导出成csv格式，之后查看内容。 | 导出csv格式成功，内容和select \* from "CVE-2020-9054"查看内容一致。 | 与预期结果一致 | - |
| 步骤 10 | 已成功安装sqlite3库，已命令行进入data\_src.db数据库。 | 输入.schema。 | 查看建表语句内容。 | 输出建表语句内容。 | 与预期结果一致 | - |
| 步骤 11 | 已成功安装sqlite3库，已命令行进入data\_src.db数据库。 | 输入.database。 | 查看附加的或者当前操作的数据库文件。 | 说明已附加的数据库文件的路径。 | 与预期结果一致 | - |
| 步骤 12 | 已成功安装sqlite3库，已命令行进入data\_src.db数据库。 | insert into "CVE-2020-9054" values('123456','snmp','1.1.1.1',80,'Italy','Failure','2021-10-07 20:51:10');  之后查看  select \* from "CVE-2020-9054"。 | 在已有的表中，按照格式添加数据，之后查询是否添加成功。 | 成功添加，查询数据库成功显示添加信息。 | 与预期结果一致 | - |
| 步骤 13 | 已成功安装sqlite3库，已命令行进入data\_src.db数据库。 | update "CVE-2020-9054" set port=8888 where id='123456';之后查看  select \* from "CVE-2020-9054"。 | 在已有的表中，修改数据，之后查询是否修改成功（需要保证where条件信息存在，否则无法删除）。 | 成功修改已有信息，查询数据库成功显示信息已修改。 | 与预期结果一致 | - |
| 步骤 14 | 已成功安装sqlite3库，已命令行进入data\_src.db数据库。 | delete from "CVE-2020-9054" where id='123456';之后查看  select \* from "CVE-2020-9054"。 | 在已有的表中，删除数据，之后查询是否删除成功（需要保证where条件信息存在，否则无法删除）。 | 成功删除已有信息，查询数据库成功显示信息已删除。 | 与预期结果一致 | - |
| 步骤 15 | 已成功安装sqlite3库，已命令行进入data\_src.db数据库。 | select \* from "CVE-2020-9054" where protocol='tcp' and port=80。 | 在已有的表中，组合查询数据。 | 成功显示满足条件信息。 | 与预期结果一致 | - |
| 步骤 16 | 已成功安装sqlite3库，已命令行进入data\_src.db数据库。 | select \* from "CVE-2020-9054" order by port。 | 查询信息，端口升序。 | 信息按照端口升序展示。 | 与预期结果一致 | - |
| 步骤 17 | 已成功安装sqlite3库，data\_src.db库文件已经准备好。 | cp -avf data\_src.db data\_src.db\_tmpbak  查看大小  ls -hl \*.db\*  操作消除空闲页sqlite3 data\_src.db “VACUUM;”  执行完  再次查看大小  最后还原文件  cp -avf data\_src.db\_tmpbak data\_src.db。 | 备份库文件，之后查看大小，操作数据库，消除空闲页，之后再次查看大小和之前的大小比对，查看操作效果，最后恢复原始数据库文件。 | 俩次大小比对不一致，第一次大小为4.4M，第二次大小为8.0k，消除空闲页操作效果明显。 | 与预期结果一致 | - |
| 步骤 18 | 已成功安装sqlite3库，已命令行进入data\_src.db数据库。 | ./sqlite\_c\_test /tmp/data\_src.db 'select \* from "CVE-2020-9054"。 | 执行C语言应用程序对sqlite数据库data\_src.db中表"CVE-2020-9054" 数据查询  （程序用法第二个参数为数据库名称 第三个参数为sql语句）。 | 执行终端成功输出sqlite数据库data\_src.db中表"CVE-2020-9054" 内容信息。 | 与预期结果一致 | - |
| 步骤 19 | 已成功安装sqlite3库。 | 命令行执行./sqlite\_cpp\_test | 执行C++语言应用程序，对sqlite数据库data\_src.db中表"CVE-2020-9054" 内容打印。 | 执行终端成功输出sqlite数据库data\_src.db中表"CVE-2020-9054" 内容信息。 | 与预期结果一致 | - |

1. DBUtils库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | DBUtils库的功能测试/GN\_YYXK \_SJFW\_DK | | | | |
| 用例说明 | | **测试目的:** 支持对数据库的连接和关闭等操作。  **测试方法:** 测试能够从数据源中获取连接；测试执行变更类的SQL语句；测试执行查询类的SQL语句，并对返回的结果集进行正确处理；测试关闭连接，将连接返回给数据源。  **合格判据:** 能够正确执行相关的SQL语句；能够获取和释放数据库连接；能够对查询结果进行正确的获取和显示。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 提供一个可供测试的数据库服务。 | 数据连接参数。 | 进入测试程序后，应用从数据源中获取一个数据库连接。 | 能够获得期望的数据库连接。 | 与预期结果一致 | - |
| 步骤 2 | 数据库里准备好相关的表结构。 | 调用数据库操作的相关SQL语句。 | 调用相关的变更sql语句操作的API，能够通过数据源或数据连接构建操作实例，支持对数据库的插入、变更和删除等操作。 | 数据库操作的语句能够正确执行。 | 与预期结果一致 | - |
| 步骤 3 | 数据库里准备好相关的操作数据。 | 查询语句。 | 调用操作实例，进行查询类语句的API操作，使用结果集合接口，对返回的结果集进行处理，提供数组、Map和JavaBean等记录转换方式。 | 查询语句正确执行，并且能够获取指定类型的记录信息。 | 与预期结果一致 | - |
| 步骤 4 | 已使用DBUtils连接数据库。 | 无 | 使用DBUtils关闭数据库连接，返回连接到数据源中。 | 数据库连接可以正确关闭。 | 数据库连接可以正确关闭 | - |

### 日志处理库测试（张金辉、（王强））

1. C++接口日志处理库测试用例（张金辉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | C++接口日志处理库/ GN\_YYXKC \_RZCL\_CJK | | | | |
| 用例说明 | | **测试目的:** 使用C++接口日志处理库，可以很便利地将日志或者跟踪调试信息写入字符流、内存字符串队列、文件。  **测试方法:** 基于C++接口日志处理库编写简单的日志程序，定义多种日志格式和布局，输出到日志文件中。  **合格判据:** 简要描述合格判据。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | log4cpp库已部署。 | 执行./log4cpp\_test。 | 启动日志程序，实例化一个layout 对象；初始化一个appender 对象；把layout对象附着在appender对象上；然后实例化一个category对象；把appender对象附到category上，设置category的优先级；调用info()、warn()和log()执行不同级别的日志输出。 | 启动日志程序向日志文件输出不同级别的日志。 | 与预期结果一致 | - |
| 步骤 2 | 已执行日志测试程序。 | 无 | 查看日志文件中的日志及格式是否与程序逻辑一致。 | 日志及格式与程序逻辑一致。 | 与预期结果一致 | - |

1. Java接口日志处理库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Java接口日志处理库/ GN\_YYXKC \_RZCL\_JJK | | | | |
| 用例说明 | | **测试目的:** 提供Java应用程序的日志处理能力；支持多种日志级别与日志输出格式。  **测试方法:** 根据不同的输出日志等级，打印日志信息；配置Appender日志输出目的地，负责把日志信息输出到指定地方，包括控制台、磁盘文件；修改配置文件：log4j.properties 或者 log4j.xml，测试不同的配置场景。  **合格判据:** 能够正确打印不同级别的日志信息；可以通过配置文件修改显示格式、日志级别，定义不同的输出目的地。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已部署java接口日志处理库与测试程序。 | 无 | 启动测试程序，进入测试程序后，创建Logger日志对象，负责捕捉日志记录信息。 | 日志对象创建成功。 | 日志对象创建成功 | - |
| 步骤 2 | 已启动测试程序。 | 不同级别的日志信息。 | 使用日志对象，根据不同的输出日志等级，打印日志信息。 | 按不同的日志级别打印日志信息。 | 与预期结果一致 | - |
| 步骤 3 | 已启动测试程序。 | 设置不同的日志输出目录。 | 配置Appender日志输出目录，负责把日志信息输出到指定地方，包括控制台、磁盘文件等。 | 能够支持不同的输出路径。 | 与预期结果一致 | - |
| 步骤 4 | 根据运行要求配置相关配置文件的目录。 | 配置文件修改信息。 | 修改配置文件：log4j.properties 或者 log4j.xml;测试通过配置修改不同的日志输出目录。 | 能够支持不同的输出路径。 | 与预期结果一致 | - |
| 步骤 5 | 已部署java接口日志处理库与测试程序。 | 配置文件修改信息。 | 测试通过配置修改日志的显示格式。 | 按指定格式输出日志。 | 输出指定格式日志 | - |
| 步骤 6 | 已部署java接口日志处理库与测试程序。 | 配置文件修改信息。 | 通过配置修改不同的日志输出级别。 | 按指定的日志基本输出日志。 | 与预期结果一致 | - |
| 步骤 7 | 已部署java接口日志处理库与测试程序。 | 配置文件修改信息。 | 通过配置滚动文件信息，防止单个日志文件过大或过期日志无法清除；通过配置日志的缓存来进行性能优化。 | 日志的尺寸和日志性能得到优化 | 与预期结果一致 | - |

1. slf4j库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | slf4j库的功能测试/GN\_YYXK \_RZCL\_SK | | | | |
| 用例说明 | | **测试目的:** 提提供Java应用程序的日志处理能力。  **测试方法:** 根据不同的输出日志等级，打印日志信息；使用slf4j的API就可以实现日志的不同级别的输出。  **合格判据:** 能够正确打印不同级别的日志信息；可以通过配置文件修改显示格式、日志级别，定义不同的输出目的地。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 后端日志库log4j库已经部署。 | 无 | 启动测试程序，进入测试程序后，应用根据具体的类创建日志工厂，使用测试日志工厂产生正确的日志实体。 | 成功创建日志工厂，并产生日志实体。 | 与结果一致 | - |
| 步骤 2 | 已启动测试程序。 | 各种级别的日志信息。 | 通过日志实体输出不同级别的日志信息，测试仅通过slf4j的API就可以实现日志的不同级别的输出。 | 仅通过slf4j的API就能实现日志输出。 | 与结果一致 | - |
| 步骤 3 | 已启动测试程序。 | 日志配置修改信息。 | 可以实现相应的日志输出和格式等配置信息与log4j的配置一致。 | 通过修改log4j的配置能够影响日志的输出。 | 与预期结果一致 | - |

### 配置解析库测试（张金辉、（王强））

1. jsoncpp库测试用例（张金辉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | jsoncpp库/ GN\_YYXKC \_PZJX\_JSK | | | | |
| 用例说明 | | **测试目的:** 验证jsoncpp库安装与常用接口调用是否正常；  **测试方法:** 编写测试用例测试程序，对Json库的数据串解析、对象序列化等等接口进行调用；  **合格判据:** 应用程序成功执行，并且能返回正确结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署jsoncpp库 | 无 | 执行json数据解析程序。 | 成功执行json数据解析程序。 | 与预期结果一致 | - |
| 步骤 2 | 已执行测试程序。 | 待解析字符串 | 调用Json::Reader类中的parse()将字符串数据解析为Json数据流。 | 能正确将字符串解析为Json数据流。 | 与预期结果一致 | - |
| 步骤 3 | 已执行测试程序。 | 无 | 调用Json::Value类中的asString()、asInt()获取Json对象。 | 能获取并输出Json数据流中的Json对象。 | 与结果一致 | - |
| 步骤 4 | 已正确部署jsoncpp库。 | 无 | 执行json序列化程序。 | 成功执行json数据解析程序。 | 与预期结果一致 | - |
| 步骤 5 | 已执行测试程序。 | 待序列化json对象 | 调用Json::Value类中的toStyledString()将Json对象序列化为字符串。 | 正确构建Json对象序列化为字符串并输出。 | 与结果一致 | - |
| 步骤 6 | 已正确部署jsoncpp库。 | 无 | 执行json数据写入文件程序。 | 成功执行json数据解析程序。 | 与结果一致 | - |
| 步骤 7 | 已执行测试程序。 | 待转换json对象 | 调用Json::FastWriter类中的write()将Json对象写为不带格式的字符流，调用Json::StyledWriter类中的write()或者Json::Value类中的toStyledString()将Json对象写为带格式的字符流。 | 正确将Json对象写为带格式或不带格式的字符流。 | 与预期结果一致 | - |

1. ini配置文件解析处理库测试用例（张金辉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | ini配置文件解析处理库/ GN\_YYXKC \_PZJX\_IK | | | | |
| 用例说明 | | **测试目的:**C语言配置文件解析库 — iniparser可以方便的对配置文件进行解析、添加、修改、删除等操作。  **测试方法:** 打开配置文件，进行读取解析，增加配置项，修改配置项，删除配置项，保存修改后的内容到新的文件中  **合格判据:** 能够正确解析、保存配置文件。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | iniparser库已部署。 | 进入测试程序文件夹执行命令./iniparser\_test，以及待解析的ini文件。 | 启动ini文件处理程序，加载ini配置文件，调用iniparser\_getstring()读取配置项，并分别调用iniparser\_set()新增或者修改ini配置项，iniparser\_unset()删除配置项。 | 能够解析、添加、修改、删除ini配置文件。 | 与预期结果一致 | - |
| 步骤 2 | 已调用ini文件处理程序。 | 无 | 调用iniparser\_dumpsection\_ini()保存ini到新的文件。 | 能够保存ini到新的文件。 | 与预期结果一致 | - |
| 步骤 3 | 已调用ini文件处理程序。 | 无 | 程序修改的项是否修改成功。 | 程序修改ini的项，与修改后的ini文件一致。 | 与预期结果一致 | - |

1. tinyxml库测试用例（张金辉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | tinyxml库/GN\_YYXKC \_PZJX\_TK | | | | |
| 用例说明 | | **测试目的:**TinyXML是目前非常流行的一款基于DOM模型的XML解析器，适合存储简单数据，配置文件，对象序列化等数据量不是很大的操作。支持对XML的读取和修改。  **测试方法:** 读取xml文件，解析文件内容，修改内容，并保存到新的文件。  **合格判据:** 修改后的内容，和新文件的内容一致。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | tinyxml库已部署，要解析的xml文件拷贝到相应目录。 | 进入测试程序文件夹执行命令./tinytest，以及待解析的xml文档。 | 程序启动，调用LoadFile()加载xml文件，然后调用RootElement()获取根节点，然后调用FirstChildElement()遍历子节点，FirstAttribute()遍历属性，并调用value()获取值内容。 | 成功解析出xml文档内容。 | 与预期结果一致 | - |
| 步骤 2 | 已执行tinyxml测试程序。 | 无 | 调用GetText()获取文本，调用saveFile()保存xml内容到文件。 | 成功保存xml到新的文件。 | 与预期结果一致 | - |
| 步骤 3 | 已执行tinyxml测试程序。 | 无 | 查看新文件的内容和程序作比较。 | 新文件修改的内容和程序修改的部分一致。 | 与预期结果一致 | - |

1. configuration库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Configuration库的功能测试/ GN\_YYXK \_PZJX\_CK | | | | |
| 用例说明 | | **测试目的:** 提供Java语言的配置解析能力。  **测试方法:** 解析配置文件（如 properties文件）的数据。  **合格判据:** 解析配置文件（如 properties文件）的数据正常。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 标准的 properties 配置文件。 | Properties 配置文件路径。 | 进入测试程序，设置 properties文件的路径；引入 properties 文件，解析并获取配置值。测试是否能获取到配置文件的配置值。 | 正常获取到 properties 文件的配置值。 | 与预期结果一致 | - |

### 图像处理库测试（张金辉、（王强））

1. C++简单图像处理库测试用例（张金辉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | C++简单图像处理库/ GN\_YYXKC \_TXCL\_CTK | | | | |
| 用例说明 | | **测试目的:**freeImage 是一个免费的、开源的，跨平台的基于C、C++语言图像处理库，支持图像基本信息处理，支持png、jpg、bmp等多种图像格式的相互转换能力，支持图像的旋转、滤波等常见处理能力。  **测试方法:** 读取图片文件，解析文件内容，修改内容，并保存成新格式的文件。  **合格判据:** 程序处理后生成的文件，和预期效果一致。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | freeimage库已部署，要解析的图像文件拷贝到相应目录。 | 进入测试程序文件夹执行命令./freeimage。 | 程序启动，初始化freeimage，加载图片文件，分别调用FreeImage\_GetWidth()获取图像宽度，FreeImage\_GetHeight()获取图像高度，  并调用FreeImage\_GetScanLine()扫描图像用来设置图像二维数组。 | 程序可以正常解析图像的基本信息。 | 与预期结果一致 | - |
| 步骤 2 | 已执行C++简单图像处理库测试程序。 | 无 | 调用FreeImage\_Save()保存图像或进行图像格式转换。 | 图像可以在png、jpg、bmp格式之间转换。 | 与预期结果一致 | - |
| 步骤 3 | 已执行C++简单图像处理库测试程序。 | 无 | 调用cpu\_rotate()进行旋转处理，调用cpu\_filter()进行滤波处理。 | 可以进行旋转、滤波操作，dst.bmp是经过旋转的图片，dst.png是滤波后的图片。 | 与预期结果一致 | - |

1. libopencv库测试用例（张金辉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | libopencv库/ GN\_YYXKC \_TXCL\_OCV | | | | |
| 用例说明 | | **测试目的**:libopencv是一个免费的、开源的，跨平台的基于C、C++语言图像处理库，支持图像基本信息处理，支持png、jpg、bmp等多种图像格式的相互转换能力，支持图像的旋转、滤波等常见处理能力。  **测试方法:** 读取图片文件，解析文件内容，修改内容，并保存成新格式的文件。  **合格判据:** 程序处理后生成的文件，和预期效果一致。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | libopencv库已部署，要解析的图像文件拷贝到相应目录。 | 进入测试程序文件夹执行命令./sharp。 | 程序启动，调用imread()加载图片文件，调用create()创建图像信息，调用sharpenImage0()进行图片锐化，  调用sharpenImage1()进行图片锐化，结果分别保存到当前目录为1.jpg和2.jpg，最后释放图片资源 | 程序可以正常解析图像的基本信息，并且1.jpg和2.jpg能明显看出锐化效果 | 与预期结果一致 | - |
| 步骤 2 | libopencv库已部署，要解析的图像文件拷贝到相应目录 | 进入测试程序文件夹执行命令./mohu。 | 程序启动，调用imread()加载图片文件，调用blur(）和GaussianBlur(）分别进行均值滤波和高斯滤波进行模糊处理，调用imwriter（）结果分别保存到blur.jpg和gaosi.jpg，最后释放图片资源 | blur.jpg和gaosi.jpg能明显看出模糊效果。 | 与预期结果一致 | - |

1. Java简单图像处理库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Java简单图像处理库的功能测试/ GN\_YYXK \_TXCL\_JTK | | | | |
| 用例说明 | | **测试目的:** 快速解析图片的基本信息（size, color space, ICC profile等）和元数据。  **测试方法:** 解析图片的基本信息；解析图片的元数据。  **合格判据:** 能够正常解析图片的基本信息和元数据。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Java简单图像处理库已部署。 | 测试的图片文件。 | 启动测试程序，设置图像文件的路径，使用使用Imaging.getImageInfo(imageFile)方法获取图片的基本信息。测试是否能获取图片的基本信息。 | 可正常获取图片的基本信息。 | 与预期结果一致 | - |
| 步骤 2 | 已启动Java简单图像处理库测试程序。 | 测试的图片文件。 | 使用Imaging.getMetadata(imageFile)方法获取图片的元数据。测试是否能获取图片的元数据。 | 可正常获取图片的元数据。 | 成功获取图片的元数据 | - |

1. Thumbnailtor库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Thumbnailtor的功能测试/ GN\_YYXK \_TXCL\_TK | | | | |
| 用例说明 | | **测试目的:** 提供图像调整大小功能和图像裁剪功能。  **测试方法:** 测试图像调整大小功能，测试图像裁剪功能  **合格判据:** 调整图像大小功能和图像裁剪功能正常。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Thumbnailtor库已部署。 | 要测试的图像文件路径，调整后图片文件的存放路径。 | 进入测试程序，使用Thumbnailtor提供的调整大小的方法调整图像文件的图像大小为原来的半，并保存到输入的存放路径，测试是否图像已调整并且存放路径里有已调整的图像文件。 | 正确将图像文件中的图像大小调整为原来的一半并保存到了存放路径。 | 与预期结果一致 | - |
| 步骤 2 | 已启动Thumbnailtor库测试程序。 | 要测试的图像文件路径，调整后图片文件的存放路径。 | 使用Thumbnailtor提供的图像裁剪的方法裁剪图片的右上角的图像，并保存到存放路径，测试是否存放路径下有已裁剪后的图像文件。 | 存放路径下有已裁剪后的图片文件。 | 成功存放已裁剪的图片文件 | - |

### 输入输出库测试（魏冠杰、（王强））

（JAVA的IO库：覆盖 网络IO、文件IO与数据压缩的能力，提供支持异步IO模型的编程接口）

1. 多模式输入输出库测试用例（魏冠杰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 多模式输入输出库/ GN\_YYXKC \_SRSC\_DMK | | | | |
| 用例说明 | | **测试目的:** 提供网络和低级I/O编程处理  **测试方法:** 调用asio库相关接口，实现tcp交互、udp交互及计时器功能。  **合格判据:** 实现tcp交互，udp交互和计时器功能。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已安装libasio-dev库，并且测试程序已编译完成。 | ./blocking\_tcp\_echo\_server 8090。 | 调用asio库tcp相关接口，实现的tcp服务端程序能正常运行。 | tcp服务端程序正常启动。 | 与预期结果一致 | 端口可自定义，大于1024，小于65535。 |
| 步骤 2 | 已安装libasio-dev库，并且测试程序已编译完成。 | 另一窗口执行  ./blocking\_tcp\_echo\_client 127.0.0.1 8090  之后输入asio tcp test! | 调用asio库tcp相关接口，实现的tcp客户端程序能和tcp服务端程序连接并成功交互。 | 成功连接tcp服务端，并且在交互后，服务端返回相同输入asio tcp test! | 与预期结果一致 | 需要和服务端启动的主机和端口一致。 |
| 步骤 3 | 已安装libasio-dev库，并且测试程序已编译完成。 | ./blocking\_udp\_echo\_server 8091。 | 调用asio库udp相关接口,实现的udp服务端程序能正常运行。 | udp服务端程序正常启动 | 与预期结果一致 | 端口可自定义，大于1024，小于65535 |
| 步骤 4 | 已安装libasio-dev库，并且测试程序已编译完成。 | 另一窗口执行  ./blocking\_udp\_echo\_server 8091  之后输入asio udp test! | 调用asio库udp相关接口，实现的udp客户端程序能和udp服务端程序连接并成功交互。 | 成功连接udp服务端，并且在交互后，服务端返回相同输入asio udp test! | 与预期结果一致 | 需要和服务端启动的主机和端口一致。 |
| 步骤 5 | 已安装libasio-dev库，并且测试程序已编译完成。 | 一个窗口启动  ./async\_tcp\_echo\_server 8092  另一窗口执行  ./blocking\_udp\_echo\_server 8092  之后输入  asio tcp async test! | 服务端使用调用asio库tcp异步接口实现，服务端启动后，启动udp客户端程序，客户端能和该服务端程序连接并进行交互。 | tcp客户端成功连接tcp异步服务端，并且在交互后，服务端返回相同输入asio tcp async test! | 与预期结果一致 | 端口可自定义，大于1024，小于65535。 |

1. 压缩输入输出库测试用例（魏冠杰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 压缩输入输出库/ GN\_YYXKC \_SRSC\_YSK | | | | |
| 用例说明 | | **测试目的:** 提供对文件压缩和解压的支持。  **测试方法:** 执行测试程序，测试程序中会对信息的压缩解压缩、.gz文件的操作、小缓存区调用deflate压缩函数和inflate解压缩函数、大缓存区调用deflate压缩函数和inflate解压缩函数、deflate压缩函数完整刷新及inflateSync解压函数、使用预设字典deflate压缩及inflate解压缩测试项，进行测试，执行程序后，输出预期信息。  **合格判据:** 执行程序后，输出预期信息。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确的安装zlib1g-dev库；已链接静态库libz.a编译生成example测试程序。 | 在测试程序文件夹执行命令./example。 | 静态编译后，对测试方法中要求的测试项进行测试。 | 正确输出预置的的信息。 | 与预期结果一致 | - |
| 步骤 2 | 已链接动态库-lz编译生成example 测试程序。 | 在测试程序文件夹执行命令./examplesh。 | 动态编译后，对测试方法中要求的测试项进行测试。 | 正确输出预置的的信息。 | 与结果一致 | - |
| 步骤 3 | 已生成example测试程序。 | 在测试程序文件夹执行命令./example64。 | 明确指定处理文件位移为64bit并且静态编译后，对测试方法中要求的测试项进行测试。 | 正确输出预置的的信息。 | 与结果一致 | - |
| 步骤 4 | 已生成example测试程序。 | 在测试程序文件夹执行命令./example64sh。 | 指定处理文件位移为64bit并且动态编译后，对测试方法中要求的测试项进行测试。 | 正确输出预置的的信息。 | 与预期结果一致 | - |

1. libuv库测试用例（魏冠杰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | libuv库/ GN\_YYXKC\_SRSC\_LK | | | | |
| 用例说明 | | **测试目的:** 测试libuv提供的异步输入输出处理、异步网络连接处理、进程间通信等处理。  **测试方法:** 调用libuv库相关处理接口，测试验证接口是否通过。  **合格判据:** 生成执行程序的源码确实进行了相关调用，执行程序执行达到预期。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已成功的安装libuv1-dev库，测试程序已编译完成。 | 在测试程序文件夹执行命令./run-tests –list。 | 看执行程序能调用的单元测试列表。 | 输出一系列单元测试可选的参数列表。 | 与预期结果一致 | - |
| 步骤 2 | 已成功的安装libuv1-dev库，测试程序已编译完成。 | ./run-tests fs\_chmod  （可对用例1中list列表中单独参数进行测试）。 | 对list列表中单独参数进行测试，成功输出预期结果。 | 对list列表中单独参数进行测试，成功输出ok等相关测试信息。 | 与预期结果一致 | - |
| 步骤 3 | 已成功的安装libuv1-dev库，测试程序已编译完成。 | ./run-tests。 | 启动测试程序，调用libuv库文件系统函数接口，能对文件进行正确的文件相关操作。 | 对libuv库文件接口进行正确的文件相关调用，输出正常。 | 与预期结果一致 | - |
| 步骤 4 | 已运行测试程序。 | 无 | 调用libuv库tcp网络编程函数接口，能正确的进行tcp网络编程操作。 | 对libuv库tcp网络编程函数接口进行正确的文件相关调用，输出正常。 | 与预期结果一致 | - |
| 步骤 5 | 已运行测试程序。 | 无 | 调用libuv库udp网络编程函数接口，能正确的进行udp网络编程操作。 | 对libuv库udp网络编程函数接口进行正确的文件相关调用，输出正常。 | 与预期结果一致 | - |
| 步骤 6 | 已运行测试程序。 | 无 | 调用libuv库dns网络编程函数接口，能正确的进行dns ip解析操作。 | 对libuv库dns网络编程函数接口进行正确的文件相关调用，输出正常。 | 与预期结果一致 | - |
| 步骤 7 | 已运行测试程序。 | 无 | 调用libuv库pipe网络编程函数接口，能正确的进行pipe网络编程操作。 | 对libuv库pipe网络编程函数接口进行正确的文件相关调用，输出正常。 | 与预期结果一致 | - |
| 步骤 8 | 已运行测试程序。 | 无 | 调用libuv库线程函数接口，能正确的进行线程操作。 | 对libuv库线程函数接口进行正确的文件相关调用，输出正常。 | 与预期结果一致 | - |
| 步骤 9 | 已运行测试程序。 | 无 | 调用libuv库线程池函数接口，能正确的进行线程池操作。 | 对libuv库线程池函数接口进行正确的文件相关调用，输出正常。 | 与预期结果一致 | - |
| 步骤 10 | 已运行测试程序。 | 无 | 调用libuv库子进程函数接口，能正确的进行子进程操作。 | 对libuv库子进程函数接口进行正确的文件相关调用，输出正常。 | 与预期结果一致 | - |
| 步骤 11 | 已运行测试程序。 | 无 | 调用libuv库其他部分相关函数接口，能正确的进行相关类型操作。 | 对libuv库其他部分相关接口进行正确的相关类型操作，输出正常。 | 与预期结果一致 | - |

1. guvaIO库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | GuavaIO库的功能测试/ GN\_YYXK\_SRSC\_GK | | | | |
| 用例说明 | | **测试目的:** 提供文件处理、流处理、编码修改等功能。  **测试方法:** 测试文件处理功能、流处理功能、编码处理功能。  **合格判据:** 文件处理功能、流处理功能、编码处理功能正常。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署GuavaIO库。 | 测试文件路径。 | 进入测试程序后，测试读取测试文件的内容，并输出文件的内容，测试是否可以正确读取测试文件并输出测试文件内容。 | 可正常读取测试文件并输出内容。 | 与预期结果一致 | - |
| 步骤 2 | 已启动GuavaIO库测试程序。 | 测试文件路径。 | 测试以文件输入流读取测试文件的内容并使用文件输出流将“test” 文本写入测试文件。 | 可正常读取测试文件内容并将“test” 写入测试文件。 | 与预期结果一致 | - |
| 步骤 3 | 编写责任链处理事件。 | 测试文件路径。 | 测试读取文件内容，并将内容以指定编码写入测试文件内。 | 可正常读取测试文件内容并修改编码后写入测试文件内。 | 与预期结果一致 | - |

### 音频视频处理库测试（余永华、（王强））

JAVA和C++的音视频处理：需要补充音视频的采集相关步骤，JAVA还有格式转换的步骤

指标：音视频的采集、分离、转换、编解码等能力，支持mpg、avi、flv等常见音视频格式，支持C、C++、JAVA语言

1. Jave视音频编解码库测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 视音频编解码库/ GN\_YYXKC \_YPSP\_BMZH | | | | |
| 用例说明 | | **测试目的:** 能够通过音频库对音频文件的编码进行设置，重新变成指定格式音频文件  **测试方法:** 编写测试用例测试程序，对视音频编解码库的文件进行音频编码、视频编码、音频解码、视频解码、格式转换等等接口进行调用。  **合格判据:** 应用程序成功执行，并且能返回正确结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署视音频编解码库和测试文件。 | 无 | 读取音视频文件，配置音视频编码属性 | 成功读取文件， 相关的音视频属性设置成功。 | 与预期结果一致 | - |
| 步骤 2 | 已完成步骤1 | 无 | 调用编码器接口，并设置编码后的目标文件，根据音视频属性集对源文件进行编码调用并生成目标文件 | 编码的目标文件正确成功生成。 | 与预期结果一致 | - |
| 步骤 3 | 已完成步骤1 | 无 | 打印并提取当前Encoder编码器支持的所有音频和视频的格式 | 确保支持mpg、avi、flv等常见音视频格式 | 与预期结果一致 |  |
| 步骤 4 | 已完成步骤1 | 无 | 分离原来的mp4音视频文件，产生独立的mp3文件 | 正确地进行分离 | 与预期结果一致 |  |

1. 视音频编解码库测试用例（余永华）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 视音频编解码库/ GN\_YYXKC \_YPSP\_YSPK | | | | |
| 用例说明 | | **测试目的:** 验证视音频编解码库安装与常用接口调用是否正常。  **测试方法:** 编写测试用例测试程序，对视音频编解码库的文件和文件夹处理、音频编码、视频编码、音频解码、视频解码、格式转换等等接口进行调用。  **合格判据:** 应用程序成功执行，并且能返回正确结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署视音频编解码库。 | 无 | 执行文件资源操作程序。 | 成功执行文件资源操作程序。 | 与预期结果一致 | - |
| 步骤 2 | 已执行文件资源操作程序。 | 音视频资源文件夹路径。 | 调用avio\_open\_dir()根据指定的路径打开文件夹，然后调用avio\_read\_dir()获取目录中子项信息，然后释放不再需要的已读文件对象，并关闭已打开文件夹。 | 初始化接口成功，正确读取文件夹信息并输出文件夹内文件列表的信息。 | 与预期结果一致 | - |
| 步骤 3 | 已正确部署视音频编解码库。 | 无 | 执行音频编码程序与视频编码程序。 | 成功执行音频编码程序与视频编码程序。 | 与预期结果一致 | - |
| 步骤 4 | 已执行音频编码程序与视频编码程序。 | 待重新编码文件。 | 调用avcodec\_find\_encoder()或avcodec\_find\_encoder\_by\_name()查找指定解码器，将AVFrame非压缩数据传给编码器，并调用avcodec\_receive\_packet()获取编码后的数据。 | 能使用指定编码器对音视频编码。 | 与预期结果一致 | - |
| 步骤 5 | 已正确部署视音频编解码库。 | 无 | 执行音频解码程序和视频解码程序。 | 成功执行音频解码程序和视频解码程序。 | 与预期结果一致 | - |
| 步骤 6 | 已执行音频解码程序和视频解码程序。 | 待解码文件 | 初始化音视频解码器，调用avcodec\_send\_packet()将AVPacket压缩数据给解码器，之后调用av\_get\_bytes\_per\_sample()获取数据并解码。 | 能使用指定编码器对音视频解码。 | 与预期结果一致 | - |
| 步骤 7 | 已正确部署视音频编解码库。 | 无 | 执行帧运动矢量数据获取程序。 | 成功执行帧运动矢量数据获取程序。 | 与预期结果一致 | - |
| 步骤 8 | 已执行帧运动矢量数据获取程序。 | 待解码文件。 | 调用av\_dict\_set()设置打开解码器的参数，调用av\_frame\_get\_side\_data()获取解码后帧的运动矢量。 | 获取并输出帧的运动矢量数据。 | 与预期结果一致 | - |
| 步骤 9 | 已正确部署视音频编解码库。 | 无 | 执行音视频数据解封装和解码程序。 | 成功执行音视频数据解封装和解码程序。 | 与预期结果一致 | - |
| 步骤 10 | 已执行音视频数据解封装和解码程序。 | 待解封装和解码文件。 | 调用avformat\_open\_input()打开音视频文件，并读取音视频数据，之后调用av\_read\_frame() 将文件内容分割成帧，并压缩数据给解码器 | 成功对音视频数据进行解封装和解码。 | 与预期结果一致 | - |
| 步骤 11 | 已正确部署视音频编解码库。 | 无 | 执行音频混音程序和视频滤镜添加程序。 | 成功执行音频混音程序和视频滤镜添加程序 | 与预期结果一致 | - |
| 步骤 12 | 已执行音频混音程序和视频滤镜添加程序。 | 待混音和添加滤镜文件。 | 调用avfilter\_graph\_alloc\_filter()创建过滤器，并初始化，然后对音视频数据进行解封装和解码。 | 成功对音视频文件混音和添加滤镜。 | 与预期结果一致 | - |
| 步骤 13 | 已正确部署视音频编解码库。 | 无 | 执行文件metadata信息获取程序。 | 成功执行文件metadata信息获取程序。 | 与预期结果一致 | - |
| 步骤 14 | 已执行文件metadata信息获取程序。 | 待获取metadata信息文件。 | 调用av\_dict\_get()获取解包后音视频文件的metadata信息。 | 能获取并输出文件的metadata信息并输出。 | 与预期结果一致 | - |
| 步骤 15 | 已正确部署视音频编解码库。 | 无 | 执行文件资源操作程序。 | 成功执行文件资源操作程序。 | 与预期结果一致 | - |
| 步骤 16 | 已执行文件资源操作程序。 | 待无编解码的转封装视音频文。件。 | 初始化文件并拷贝编码参数，然后调用avio\_open()打开输出文件，调用av\_interleaved\_write\_frame()转封装视音频文件。 | 能无编解码的转封装视音频文件。 | 与预期结果一致 | - |
| 步骤 17 | 已执行文件资源操作程序。 | 待转换参数的音频文件。 | 初始化文件后调用av\_opt\_set\_\*()设置输入和输出音频的信息，并调用swr\_convert()进行重采样转换。 | 能音频重采样，改变音频的相关参数：采样率、采样通道、采样格式等并保存到另一个文件中。 | 与预期结果一致 | - |
| 步骤 18 | 已执行文件资源操作程序。 | 待缩放视频文件。 | 初始化文件后调用sws\_scale()视频图像。 | 能对视频进行缩放。 | 与预期结果一致 | - |
| 步骤 19 | 已执行文件资源操作程序。 | 待转码音频文件。 | 调用音频编解码相关API把其他格式的音频文件比如mp3、wav等格式转换为aac格式。 | 能将提供音频转码为AAC格式。 | 与预期结果一致 | - |
| 步骤 20 | 已执行文件资源操作程序。 | 待转码视频文件。 | 调用视频编解码相关API把视频文件转换成制定格式。 | 能用编解码和滤波器将提供视频转码为指定格式。 | 与预期结果一致 | - |

1. 模块化音视频编解码框架库测试用例（余永华）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 模块化音视频编解码框架库/ GN\_YYXKC \_YPSP\_MKHYSP | | | | |
| 用例说明 | | **测试目的:** 验证模块化音视频编解码框架安装与常用接口调用是否正常。  **测试方法:** 编写测试用例测试程序，对模块化音视频编解码框架的版本获取、音视频播放、文件信息获取等等接口进行调用。  **合格判据:** 应用程序成功执行，并且能返回正确结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署模块化音视频编解码框架。 | 无 | 运行框架版本获取程序。 | 成功运行框架版本获取程序。 | 成功运行框架版本 | - |
| 步骤 2 | 已运行框架版本获取程序。 | 无 | 调用gst\_init ()初始化框架，并调用gst\_version ()获取框架版本。 | 框架初始化，获取并输出调用的框架版本信息。 | 与预期结果一致 | - |
| 步骤 3 | 已正确部署模块化音视频编解码框架。 | 无 | 运行管道播放视频程序。 | 成功运行管道播放视频程序。 | 与预期结果一致 | - |
| 步骤 4 | 已运行管道播放视频程序。 | 待播放视频。 | 调用gst\_parse\_launch()创建管道，启动管道后等待并处理管道上的消息。 | 创建管道，启动管道直接播放视频。 | 与预期结果一致 | - |
| 步骤 5 | 已正确部署模块化音视频编解码框架。 | 无 | 运行元件连接管道程序 | 成功运行元件连接管道程序。 | 与预期结果一致 | - |
| 步骤 6 | 已运行元件连接管道程序。 | 无 | 调用 gst\_element\_factory\_make()创建元件，将元件放入管道并连接元件。 | 创建元件，将元件放入管道并连接元件。 | 与预期结果一致 | - |
| 步骤 7 | 已正确部署模块化音视频编解码框架。 | 无 | 运行播放位置跳转程序。 | 成功运行播放位置跳转程序。 | 与结果一致 | - |
| 步骤 8 | 已运行播放位置跳转程序。 | 待播放视频 | 查询元件及当前位置，然后调用gst\_element\_seek\_simple()跳到指定位置。 | 查询元件，获取播放位置以及可跳播范围，执行跳播操作。 | 与结果一致 | - |
| 步骤 9 | 已正确部署模块化音视频编解码框架。 | 无 | 运行播放器绘制程序。 | 成功运行播放器绘制程序。 | 与预期结果一致 | - |
| 步骤 10 | 已运行播放器绘制程序。 | 待播放视频。 | 创建按钮，调用g\_signal\_connect()连接按钮信号与回调函数，并调用gst\_element\_set\_state()设置播放状态。 | 查询元件，获取播放位置以及可跳播范围，执行跳播操作。 | 与预期结果一致 | - |
| 步骤 11 | 已正确部署模块化音视频编解码框架。 | 无 | 运行文件信息获取程序。 | 成功运行文件信息获取程序。 | 运行文件信息并获取程序 | - |
| 步骤 12 | 已运行文件信息获取程序。 | 待获取信息文件 | 调用gst\_discoverer\_info\_get\_\*()系列接口获取文件数据。 | 快速找出文件或者文件的链接所包含的媒体具体信息或者是否能够播放该媒体。 | 与预期结果一致 | - |

### 系统诊断库测试（余永华）

1. 程序问题诊断处理库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 程序问题诊断处理库/ GN\_YYXKC \_XTZD\_CXZD | | | | |
| 用例说明 | | **测试目的:** 验证程序问题诊断处理库安装与常用接口调用是否正常。  **测试方法:** 编写测试用例测试程序，对程序问题诊断处理库的可执行程序诊断、未初始化内存诊断、内存读写越界诊断等功能与接口进行调用。  **合格判据:** 应用程序成功执行，并且能返回正确结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署程序问题诊断处理库。 | 命令行执行valgrind –version。 | 输出版本信息，验证是否安装成功。 | 正确输出版本信息。 | 与预期结果一致 | - |
| 步骤 2 | 已正确部署程序问题诊断处理库。 | 命令行执行valgrind ./valgrindTest0。 | 对C/C++可执行程序进行基础诊断。 | 正确输出诊断结果。 | 与预期结果一致 | - |
| 步骤 3 | 已正确部署程序问题诊断处理库。 | 命令行执行valgrind ./valgrindTest1。 | 对未初始化内存的C/C++程序进行诊断。 | 正确诊断未初始化内存。 | 与预期结果一致 | - |
| 步骤 4 | 已正确部署程序问题诊断处理库。 | 命令行执行valgrind ./valgrindTest2。 | 对内存读写越界的C/C++程序进行诊断。 | 正确诊断内存读写越界。 | 与预期结果一致 | - |
| 步骤 5 | 已正确部署程序问题诊断处理库。 | 命令行执行valgrind ./valgrindTest3 | 对内存覆盖的C/C++程序进行诊断。 | 正确诊断内存覆盖。 | 与预期结果一致 | - |
| 步骤 6 | 已正确部署程序问题诊断处理库。 | 命令行执行valgrind ./valgrindTest4。 | 对动态内存管理错误的C/C++程序进行诊断。 | 正确诊断动态内存管理错误。 | 与预期结果一致 | - |
| 步骤 7 | 已正确部署程序问题诊断处理库 | 命令行执行valgrind ./valgrindTest5。 | 对内存泄露的C/C++程序进行诊断。 | 正确内存泄露。 | 与预期结果一致 | - |

1. gdb库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | gdb库/ GN\_YYXKC\_XTZD\_GDK | | | | |
| 用例说明 | | **测试目的:** 验证gdb库安装与常用接口调用是否正常。  **测试方法:** 编写测试用例测试程序，对gdb库的断点设置、单步调试、变量名打印等功能与接口进行调用。  **合格判据:** 应用程序成功执行，并且能返回正确结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署gdb库。 | 命令行执行gdb buggy.out。 | 使用GDB工具进入程序诊断模式。 | 成功进入应用程序的gdb诊断模式。 | 与预期结果一致 | - |
| 步骤 2 | 已启动gdb程序。 | 命令行执行b main。 | 对函数设置断点，输出设置断点信息。 | 设置断点正常。 | 与预期结果一致 | - |
| 步骤 3 | 已设置断点。 | 命令行执行run。 | 运行程序，程序在函数断点处停止并等待gdb命令。 | 运行程序正常。 | 与预期结果一致 | - |
| 步骤 4 | 已运行程序。 | 命令行执行s。 | 单步进入，程序跟踪到函数内部，输出源码信息 | 单步进入正常。 | 与预期结果一致 | - |
| 步骤 5 | 已执行上一步操作。 | 命令行执行p element。 | 打印基本类型变量，输出变量名对应的变量。 | 打印基本类型变量正常。 | 与预期结果一致 | - |
| 步骤 6 | 已启动gdb程序。 | 命令行执行quit。 | 退出程序诊断模式。 | 退出程序诊断模式正常。 | 与预期结果一致 | - |

1. jdb库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | DBUtils库的功能测试/ GN\_YYXK\_XTZD\_JDK | | | | |
| 用例说明 | | **测试目的:** 提供Java 程序的调试能力。  **测试方法:** 调试 Java 程序。  **合格判据:** 调试 Java 程序正常。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK 适配正常并且 JDK 中包含 Jdb 命令。 | 编译后的 Java 程序的字节码，如 test.class。 | 使用 jdb test.class命令进入调试交互行。测试是否进入调试交互行。 | 正常进入调试交互行。 | 与预期结果一致 | - |
| 步骤 2 | 已启动jdb程序。 | 命令行执行stop in Test.main。 | 在调试交互行中输入打断点的命令（stop in Test.main）,测试是否成功打断点。 | 打断点功能正常。 | 与预期结果一致 | - |
| 步骤 3 | 已设置断点。 | 命令行执行threads。 | 在调试交互行输入列出线程的命令（threads），列出当前正在运行的线程。 | 正确列出正在运行的线程。 | 与预期结果一致 | - |
| 步骤 4 | 已设置断点。 | 命令行执行Step。 | 在调试交互行输入下一步命令（step），测试程序是否执行并且停在下一行。 | 下一步功能正常。 | 与预期结果一致 | - |
| 步骤 5 | 已执行上一步操作。 | 命令行执行locals。 | 在调试交互行输入列出当前堆栈帧中的所有局部变量的命令（locals），测试是否成功列出局部变量。 | 正确列出当前堆栈帧中的局部变量。 | 与预期结果一致 | - |
| 步骤 6 | 已执行上一步操作。 | 命令行执行print a/dump a/eval a。 | 在调试交互行输入打印当前堆栈帧中的所有局部变量值的命令（print a/dump a/eval a，其中a为变量名），测试是否成功输出局部变量的值。 | 正确列出当前堆栈帧中的局部变量的值，从而实现函数调用栈分析。 | 与预期结果一致 | - |
| 步骤 6 | 已执行上一步操作。 | 命令行执行jps。 | 另起一个调试交互行并输入列出当前系统中的java进程及其进程号的命令（jps），获取上一步java进程的进程号。 | 正确列出当前系统中的java进程的进程号。 | 与预期结果一致 | - |
| 步骤 7 | 已执行上一步操作。 | 命令行执行jmap [pid]。 | 在调试交互行输入用于打印指定Java进程的堆内存细节的命令(jmap [进程号]，如jmap 233065) ，成功打印上一步Java进程的堆内存细节。 | 正确打印上一步Java进程的堆内存细节，实现了应用程序内存检测。 | 与预期结果一致 | - |
| 步骤 8 | 已执行上一步操作。 | 命令行执行Cont。 | 在调试交互行中输入继续命令（cont），测试程序是否执行到下一个断点或正常退出。 | 继续命令功能正常。 | 与预期结果一致 | - |

1. JMH库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JMH库的功能测试/ GN\_YYXK\_XTZD\_JMK | | | | |
| 用例说明 | | **测试目的:** 支持设置基准测试项目、建立测试基准，运行基准测试的能力。  **测试方法:** 测试通过设置基准测试的项目、建立测试基准、运行基准测试。  **合格判据:** 基准测试程序可正常输出基准测试结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JMH库已成功部署。 | 目标测试类。 | 创建基准测试类,标记@BenchmarkMode(Mode.AverageTime)注解，在类中创建测试方法，使用@Benchmark 进行标注，方法内调用目标测试类程序。 | 基准测试类正常创建。 | 与预期结果一致 | - |
| 步骤 2 | JMH库已成功部署。 | 基准测试类。 | 创建测试类，创建测试方法，使用 Option 类设置启动参数，然后使用 Runner 类启动基准测试程序，测试是否输出基准测试结果。 | 正常输出基准测试结果。 | 与预期结果一致 | - |

### 文件系统访问库测试（王峰、余永华）

1. 统一文件系统访问库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 统一文件系统访问库/ GN\_YYXKC \_WJFW\_TYWJK | | | | |
| 用例说明 | | **测试目的:** 验证统一文件系统访问库安装与常用接口调用是否正常。  **测试方法:** 编写测试用例测试程序，对统一文件系统访问库的目录及文件创建、更改、删除等功能与接口进行调用。  **合格判据:** 应用程序成功执行，并且能返回正确结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署统一文件系统访问库。  服务端已正常开启统一文件系统服务 | 命令行执行nfs\_cfg\_sandbox.sh。 | 配置客户端的统一文件系统服务。 | 服务正常启动。 | 与预期结果一致 | - |
| 步骤 2 | 已正确部署统一文件系统访问库。 | 命令行执行test\_dir.sh。 | 测试目录创建、更改、删除等功能。 | 能正确实现目录创建、更改、删除功能。 | 与预期结果一致 | - |
| 步骤 3 | 已启动统一文件系统访问库测试程序。 | 命令行执行test\_file.sh。 | 测试文件创建、更改、删除等功能。 | 能正确实现文件创建、更改、删除功能。 | 与预期结果一致 | - |

1. VFS库测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | VFS库的功能测试/GN\_YYXK \_WJFW\_VK | | | | |
| 用例说明 | | **测试目的:**支持Java 语言的虚拟文件系统的处理。  **测试方法:** 测试通过 VFS库能够创建指定目录和删除指定目录,  测试可以创建某个特定的文件和删除文件。  **合格判据:** VFS库能够正确创建指定目录，删除指定目录，创建指定文件，删除指定文件。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已正确部署VFS库。 | 需创建的目录路径。 | 进入测试程序后，创建文件系统管理器对象，通过文件系统管理器对象解析所指定的目录获取文件对象创建目录。 | 正确创建目录。 | 与结果一致 | - |
| 步骤 2 | 已启动测试程序。 | 无 | 测试通过文件对象可以删除目录。 | 正确删除目录。 | 与预期结果一致 | - |
| 步骤 3 | 已启动测试程序。 | 需创建文件的路径。 | 通过文件系统管理器对象解析所指定的文件获取文件对象创建文件。 | 正确创建文件。 | 与预期结果一致 | - |
| 步骤 4 | 已启动测试程序。 | 无 | 测试通过文件对象可以删除文件。 | 正确删除文件。 | 成功删除文件 | - |
| 步骤 5 | 已启动测试程序。 | 无 | 创建要修改的目录，如/tmp/abc | 目录创建成功 | 与预期结果一致 |  |
| 步骤 6 | 已启动测试程序。 | 无 | 修改目录，将/tmp/abc修改为/tmp/cba | 目录修改成功 | 与预期结果一致 |  |

### 其他库测试（赵廉、倪宝珠）

1. .NET库序列化用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | .NET库序列化/ GN\_YYXK \_QTK\_DNK\_SER | | | | |
| 用例说明 | | **测试目的:**测试在.NET中运行库序列化应用程序。  **测试方法:** 利用dotnet运行测试用例。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装.NET Runtime | 无 | 在终端，进入程序文件夹 | 输出目录信息 | 与预期结果一致 | - |
| 步骤2 | 使用root用户 | 无 | dotnet bin/Debug/net5.0/SystemTextJsonHowTo.dll | 输出运行信息:  序列化运行结果 | 与预期结果一致 | - |

1. .NET库I/O支持用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | .NET库I/O支持/ GN\_YYXK \_QTK\_DNK\_IO | | | | |
| 用例说明 | | **测试目的:**测试在.NET中运行库I/O支持的应用程序。  **测试方法:** 利用dotnet运行测试用例。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装.NET Runtime | 无 | 在终端，进入程序文件夹 | 输出目录信息 | 与预期结果一致 | - |
| 步骤2 | 使用root用户 | 无 | dotnet bin/Debug/net5.0/FileReader.dll | 输出运行信息:  TestFile | 与预期结果一致 | - |

1. .NET库运算符用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | .NET库运算符/ GN\_YYXK \_QTK\_DNK\_OP | | | | |
| 用例说明 | | **测试目的:**测试在.NET中运行库运算符的应用程序。  **测试方法:** 利用dotnet运行测试用例。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装.NET Runtime | 无 | 在终端，进入程序文件夹 | 输出目录信息 | 与预期结果一致 | - |
| 步骤2 | 使用root用户 | 无 | dotnet bin/Debug/net5/lambda-expressions.dll | 输出运行信息:  lambda 引用的外部变量。 | 与预期结果一致 | - |

1. .NET库类型支持用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | .NET库类型支持/ GN\_YYXK \_QTK\_DNK\_TYPE | | | | |
| 用例说明 | | **测试目的:**测试在.NET中运行库类型支持的应用程序。  **测试方法:** 利用dotnet运行测试用例。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装.NET Runtime | 无 | 在终端，进入程序文件夹 | 输出目录信息 | 与预期结果一致 | - |
| 步骤2 | 使用root用户 | 无 | dotnet bin/Debug/net5/builtin-types.dll | 输出运行信息:  值类型和引用类型 | 与预期结果一致 | - |

1. QT4用户界面测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Qt4用户界面/GN\_YYXK \_QTK4\_QTK\_YHJM | | | | |
| 用例说明 | | **测试目的:** Qt软件能成功运行实例程序，用户图形界面布局合理，可以实现用户交互体验。  **测试方法:** 对QT用户界面的标签类显示、下拉框测试、按钮测试、进度条测试选择框的基础功能进行交互性测试。  **合格判据:** Qt服务可以运行QtGuiText示例程序且运行无报错。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Qt库及示例程序正常安装。 | 无 | 运行QtGuiText程序，运行成功弹出“用户界面”窗口。 | 窗口界面布局合理，图片显示清晰不模糊，文字显示正常无乱码。 | 与预期结果一致 | - |
| 步骤 2 | QtGuiText程序成功运行。 | 无 | 点击“下拉框测试”。 | 右侧的下拉框按钮，下拉框显示“10、20、50、100”4个数值选项。 | 与预期结果一致 | - |
| 步骤 3 | QtGuiText程序成功运行，已点击“下拉框测试”。 | 无 | 分别点击下拉框数值。 | 可成功点击并切换数值。 | 与预期结果一致 | - |
| 步骤 4 | QtGuiText程序成功运行 | 无 | 点击“进度条测试”。 | 右侧的“进度条”也会随之改变相应的数值进度。 | 与预期结果一致 | - |
| 步骤 5 | QtGuiText程序成功运行，已点击“进度条测试”。 | 无 | 分别点击“按钮测试”，右侧的“10%、20%、50%、100%”按钮。 | 点击成功并且“进度条测试”右侧的“进度条”也会随之改变相应的数值进度。 | 与预期结果一致 | - |
| 步骤 6 | QtGuiText程序成功运行。 | 无 | 点击“单选框测试” 。 | 按钮可成功点亮，再次点击按钮亮度消失，遍历操作“复选框测试1、复选框测试2、复选框测试3”均可成功。 | 与预期结果一致 | - |

1. QT4网络传输测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT4网络传输/ GN\_YYXK \_QTK4\_QTK\_WLCS | | | | |
| 用例说明 | | **测试目的:** Qt软件能成功运行实例程序，通过TCP、UDP和HTTP等网络传输协议实现网络通信与数据传输。  **测试方法:** 1. Qt服务可以运行tcp服务端和tcp客户端的示例程序，实现tcp协议可靠性连接；  2. Qt服务可以运行udp接收端和udp发送端的示例程序，实现udp协议的数据发送与接收；  3. Qt服务可以运行http文本传输的示例程序，实现超文本协议数据传输。  **合格判据:** Qt服务可以运行tcp、udp、http示例程序且运行无报错。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Qt库、示例程序正常安装。 | 无 | 运行tcpserver服务端程序。 | tcp服务端运行成功；弹出tcp服务端窗口，显示“等待连接”提示。 | 与预期结果一致 | - |
| 步骤 2 | Qt库、示例程序正常安装。 | 无 | 运行tcpclient客户端程序。 | udp客户端运行成功，弹出udp客户端窗口，显示“主机”、“端口”等输入信息框。 | 与预期结果一致 | - |
| 步骤 3 | tcpserver服务端、客户端程序已运行。 | IP地址：“127.0.0.1”端口：“6666”。 | 机框输入：“127.0.0.1”端口框输入：“6666”，点击“连接”按钮。 | tcp服务端和tcp客户端连接成功，显示“数据测试成功”。 | 与预期结果一致 | - |
| 步骤 4 | Qt库、示例程序正常安装。 | 无 | 运行udpreceiver接收端程序。 | udp接收端运行成功，弹出udp接收端窗口，显示“等待接收数据”提示。 | 与预期结果一致 | - |
| 步骤 5 | Qt库、示例程序正常安装。 | 无 | 运行udpsender发送端程序。 | udp发送端运行成功，弹出udp发送端窗口，显示“进行广播”按钮。 | 与预期结果一致 | - |
| 步骤 6 | udpreceiver接收端、发送端程序已运行。 | 无 | 点击“进行广播”按钮。 | udp发送端进行数据广播发送，udp接收端能成功接收数据，显示“UDP接收数据成功”。 | 与预期结果一致 | - |
| 步骤 7 | Qt服务测试http超文本传输协议系统网络连接正常。 | 被测试外网地址 | 使用ping命令测试任意外网地址。 | 系统网络连接正常。 | 与预期结果一致 | - |
| 步骤 8 | Qt库、示例程序正常安装。 | 无 | 运行HTTPtext程序。 | 成功弹出http服务窗口。 | 与预期结果一致 | - |
| 步骤 9 | HTTPtext程序已运行。 | 无 | 点击“http测试”按钮。 | 文字框中能接收到网站信息。 | 与预期结果一致 | - |

1. QT4库-OpenGL图形绘制测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT4库-OpenGL图形绘制测试/ GN\_YYXK \_QTK4\_QTK\_ TXHZ | | | | |
| 用例说明 | | **测试目的:** 对QWidget及QGLWidget进行相同的绘制操作并进行比较，验证成功实现OpenGL图形加速功能。  **测试方法:**  1.QWidget绘制操作；  2.QGLWidget绘制操作；  3.QWidget及QGLWidget并排展示绘制内容。  **合格判据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | qt4已安装、qtcreator已安装。 | 无 | 运行“2dpainting”示例程序。 | 出现程序窗口。 | 与预期结果一致 | - |
| 步骤 2 | 已运行“2dpainting”示例程序。 | 无 | 程序调用QWidget及QGLWidget绘制图形。 | QWidget、QGLWidget分列窗口的两端，QWidget绘制内容具有抗锯齿效果。 | 与预期结果一致 | - |

1. QT4库-文本通信测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT4库-文本通信测试/ GN\_YYXK \_QTK4\_QTK\_ WBTX | | | | |
| 用例说明 | | **测试目的:** 通过DBUS方式，实现纯文本聊天的功能。  **测试方法:** 完成一对一文本报文传输功能。  **合格判据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 通信双方客户机上已安装qt4，并已启动，通信双方客户机上已安装qtcreator，并已启动，通信网络畅通。 | 无 | 运行“chat”示例程序。 | 运行成功。 | 与预期结果一致 | - |
| 步骤 2 | 已运行“chat”示例程序。 | 昵称信息。 | 在程序窗口输入昵称并继续。 | 输入成功 | 与预期结果一致 | - |
| 步骤 3 | 已运行“chat”示例程序并输入昵称。 | 聊天信息。 | 再次运行本程序，重复上述过程，输入文本信息并发送。（多次重复运行本程序，在各聊天窗口输入文本信息，查看聊天窗口信息）。 | 在双方聊天记录窗口出现发送信息。 | 与预期结果一致 | - |

1. QT5用户界面测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Qt5用户界面/GN\_YYXK \_QTK5\_QTK\_YHJM | | | | |
| 用例说明 | | **测试目的:** Qt软件能成功运行实例程序，用户图形界面布局合理，可以实现用户交互体验。  **测试方法:** 对QT用户界面的标签类显示、下拉框测试、按钮测试、进度条测试选择框的基础功能进行交互性测试。  **合格判据:** Qt服务可以运行QtGuiText示例程序且运行无报错。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Qt库及示例程序正常安装。 | 无 | 运行QtGuiText程序，运行成功弹出“用户界面”窗口。 | 窗口界面布局合理，图片显示清晰不模糊，文字显示正常无乱码。 | 与预期结果一致 | - |
| 步骤 2 | QtGuiText程序成功运行。 | 无 | 点击“下拉框测试”。 | 右侧的下拉框按钮，下拉框显示“10、20、50、100”4个数值选项。 | 与预期结果一致 | - |
| 步骤 3 | QtGuiText程序成功运行，已点击“下拉框测试”。 | 无 | 分别点击下拉框数值。 | 可成功点击并切换数值。 | 与预期结果一致 | - |
| 步骤 4 | QtGuiText程序成功运行 | 无 | 点击“进度条测试”。 | 右侧的“进度条”也会随之改变相应的数值进度。 | 与预期结果一致 | - |
| 步骤 5 | QtGuiText程序成功运行，已点击“进度条测试”。 | 无 | 分别点击“按钮测试”，右侧的“10%、20%、50%、100%”按钮。 | 点击成功并且“进度条测试”右侧的“进度条”也会随之改变相应的数值进度。 | 与预期结果一致 | - |
| 步骤 6 | QtGuiText程序成功运行。 | 无 | 点击“单选框测试” 。 | 按钮可成功点亮，再次点击按钮亮度消失，遍历操作“复选框测试1、复选框测试2、复选框测试3”均可成功。 | 与预期结果一致 | - |

1. QT5网络传输测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT5网络传输/ GN\_YYXK \_QTK5\_QTK\_WLCS | | | | |
| 用例说明 | | **测试目的:** Qt软件能成功运行实例程序，通过TCP、UDP和HTTP等网络传输协议实现网络通信与数据传输。  **测试方法:** 1. Qt服务可以运行tcp服务端和tcp客户端的示例程序，实现tcp协议可靠性连接；  2. Qt服务可以运行udp接收端和udp发送端的示例程序，实现udp协议的数据发送与接收；  3. Qt服务可以运行http文本传输的示例程序，实现超文本协议数据传输。  **合格判据:** Qt服务可以运行tcp、udp、http示例程序且运行无报错。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Qt库、示例程序正常安装。 | 无 | 运行tcpserver服务端程序。 | tcp服务端运行成功；弹出tcp服务端窗口，显示“等待连接”提示。 | 与预期结果一致 | - |
| 步骤 2 | Qt库、示例程序正常安装。 | 无 | 运行tcpclient客户端程序。 | udp客户端运行成功，弹出udp客户端窗口，显示“主机”、“端口”等输入信息框。 | 与预期结果一致 | - |
| 步骤 3 | tcpserver服务端、客户端程序已运行。 | IP地址：“127.0.0.1”端口：“6666”。 | 机框输入：“127.0.0.1”端口框输入：“6666”，点击“连接”按钮。 | tcp服务端和tcp客户端连接成功，显示“数据测试成功”。 | 与预期结果一致 | - |
| 步骤 4 | Qt库、示例程序正常安装。 | 无 | 运行udpreceiver接收端程序。 | udp接收端运行成功，弹出udp接收端窗口，显示“等待接收数据”提示。 | 与预期结果一致 | - |
| 步骤 5 | Qt库、示例程序正常安装。 | 无 | 运行udpsender发送端程序。 | udp发送端运行成功，弹出udp发送端窗口，显示“进行广播”按钮。 | 与预期结果一致 | - |
| 步骤 6 | udpreceiver接收端、发送端程序已运行。 | 无 | 点击“进行广播”按钮。 | udp发送端进行数据广播发送，udp接收端能成功接收数据，显示“UDP接收数据成功”。 | 与预期结果一致 | - |
| 步骤 7 | Qt服务测试http超文本传输协议系统网络连接正常。 | 被测试外网地址 | 使用ping命令测试任意外网地址。 | 系统网络连接正常。 | 与预期结果一致 | - |
| 步骤 8 | Qt库、示例程序正常安装。 | 无 | 运行HTTPtext程序。 | 成功弹出http服务窗口。 | 与预期结果一致 | - |
| 步骤 9 | HTTPtext程序已运行。 | 无 | 点击“http测试”按钮。 | 文字框中能接收到网站信息。 | 与预期结果一致 | - |

1. QT5库-OpenGL图形绘制测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT5库-OpenGL图形绘制测试/ GN\_YYXK \_QTK5\_QTK\_ TXHZ | | | | |
| 用例说明 | | **测试目的:** 对QWidget及QGLWidget进行相同的绘制操作并进行比较，验证成功实现OpenGL图形加速功能。  **测试方法:**  1.QWidget绘制操作；  2.QGLWidget绘制操作；  3.QWidget及QGLWidget并排展示绘制内容。  **合格判据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Qt5已安装、qtcreator已安装。 | 无 | 运行“2dpainting”示例程序。 | 出现程序窗口。 | 与预期结果一致 | - |
| 步骤 2 | 已运行“2dpainting”示例程序。 | 无 | 程序调用QWidget及QGLWidget绘制图形。 | QWidget、QGLWidget分列窗口的两端，QWidget绘制内容具有抗锯齿效果。 | 与预期结果一致 | - |

1. QT5库-文本通信测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT5库-文本通信测试/ GN\_YYXK \_QTK5\_QTK\_ WBTX | | | | |
| 用例说明 | | **测试目的:** 通过DBUS方式，实现纯文本聊天的功能。  **测试方法:** 完成一对一文本报文传输功能。  **合格判据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 通信双方客户机上已安装qt5，并已启动，通信双方客户机上已安装qtcreator，并已启动，通信网络畅通。 | 无 | 运行“chat”示例程序。 | 运行成功。 | 与预期结果一致 | - |
| 步骤 2 | 已运行“chat”示例程序。 | 昵称信息。 | 在程序窗口输入昵称并继续。 | 输入成功 | 与预期结果一致 | - |
| 步骤 3 | 已运行“chat”示例程序并输入昵称。 | 聊天信息。 | 再次运行本程序，重复上述过程，输入文本信息并发送。（多次重复运行本程序，在各聊天窗口输入文本信息，查看聊天窗口信息）。 | 在双方聊天记录窗口出现发送信息。 | 与预期结果一致 | - |

1. DPDK库环境抽象层功能用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | DPDK库环境抽象层功能/ GN\_YYXK \_QTK\_DK\_EAL | | | | |
| 用例说明 | | **测试目的:**测试DPDK库环境抽象层功能。  **测试方法:** 利用helloworld测试用例。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装dpdk | 无 | 在终端，进入程序文件夹，运行:  make | 输出运行信息,产生编译产物 | 与预期结果一致 | - |
| 步骤2 | 使用root用户 | 无 | 在终端，进入程序文件夹，运行:  ./build/helloworld | 输出EAL信息，以及hello from core 1  hello from core 2  … | 与预期结果一致 | - |

1. DPDK库二层转发用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | DPDK库二层转发功能/ GN\_YYXK \_QTK\_DK\_L2FWD | | | | |
| 用例说明 | | **测试目的:** 测试DPDK库二层转发功能。  **测试方法:** 利用l2fwd测试用例。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装dpdk | 无 | 在终端，进入程序文件夹，运行:  make | 输出运行信息,产生编译产物 | 与预期结果一致 | - |
| 步骤2 | 使用root用户.  连接的网络端口有数据包传输 | 无 | 在终端，进入程序文件夹，运行:  ./build/l2fwd -l 2-3 -- -p 0x3 | 输出每个端口收发统计 | 与预期结果一致 | - |

1. DPDK库用户工具使用用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | DPDK库用户工具使用用例//GN\_YYXK \_QTK\_DK\_UTIL | | | | |
| 用例说明 | | **测试目的:**测试DPDK库用户工具使用。  **测试方法:** DPDK库用户工具使用。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装dpdk | 无 | 在终端运行:  dpdk-devbind.py -s | 输出可用的设备 | 与预期结果一致 | - |
| 步骤2 | 使用root用户 | 无 | 在终端运行:  dpdk-devbind.py --bind=igb\_uio DEVICE1 | 绑定igb\_uio成功，  状态  Network devices using DPDK-compatible driver包括绑定的设备 | 与预期结果一致 | - |

1. SPDK库bdev用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | SPDK库bdev用例/ GN\_YYXK \_QTK\_SK\_BDEV | | | | |
| 用例说明 | | **测试目的:** 测试spdk套件bdev运行库是否能正常使用。  **测试方法:** 利用spdk套件bdev示例程序测试。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装spdk | 无 | 在终端运行$ sudo ./hello\_bdev -c bdev.conf | 输出包含运行信息:  hello\_bdev.c: 91:read\_complete: \*NOTICE\*: Read string from bdev : Hello World! | 与预期结果一致 | - |

1. SPDK库nvme识别用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | SPDK库nvme识别/ GN\_YYXK \_QTK\_SK\_NVMEIDF | | | | |
| 用例说明 | | **测试目的:** 测试spdk套件是否能正常识别nvme硬盘。  **测试方法:** 利用spdk套件identify程序测试。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装spdk | 无 | 在终端运行sudo ./identify | 输出包含类似如下信息:  NVMe Controller Controller Capabilities/Features Vendor ID: Subsystem Vendor ID: Serial Number: Model Number: | 与预期结果一致 | - |

1. spdk库nvme顺序读写用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | SPDK库nvme顺序读写/ GN\_YYXK \_QTK\_SK\_NVMESRW | | | | |
| 用例说明 | | **测试目的:** 测试spdk套件是否能正常进行nvme硬盘的顺序读写操作.  **测试方法:** 利用spdk套件arbitration程序测试。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装spdk | 无 | 在终端运行sudo ./arbitration -q 64 -s 131072 -w rw -M 50 -l 0 -t 10 -c 0xf -m 0 -a 0 -b 0 -n 100000 -i 0 | 输出包含类似如下信息:  Starting thread on core 0 with urgent priority queue  PLEXTOR PX-256M9PY + (P02028300042) core 0: 1286.30 IO/s 77.74 secs/100000 ios | 与预期结果一致 | - |

1. spdk库nvme随机读写用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | SPDK库nvme随机读写/ GN\_YYXK \_QTK\_SK\_NVMERRW | | | | |
| 用例说明 | | **测试目的:** 测试spdk套件是否能正常进行nvme随机读写操作。  **测试方法:** 利用spdk套件arbitration程序测试。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装spdk | 无 | 在终端运行sudo ./arbitration -q 64 -s 131072 -w randrw -M 50 -l 0 -t 10 -c 0xf -m 0 -a 0 -b 0 -n 100000 -i 0 | 输出包含类似运行信息:  PLEXTOR PX-256M9PY + (P02028300042 ) core 3: 530.20 IO/s 188.61 secs/100000 ios | 与预期结果一致 | - |

1. openmpi编译MPI应用用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | openmpi库代码编译 / GN\_YYXK \_QTK\_OMK\_MPIBY | | | | |
| 用例说明 | | **测试目的:** 利用OpenMPI库，将源代码编译。  **测试方法:** 在测试源码文件夹编译。  **判断依据:** 编译正常完成，无错误信息。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装openmpi-bin | 无 | 终端依次输入mpicc -v  mpic++ -v  mpifort -v  mpijavac –version | 输出编译器版本 | 与预期结果一致 | - |
| 步骤2 | 已安装openmpi-bin,已在测试文件夹目录 | 无 | 在终端运行make | 编译完成，无错误，生成可执行二进制文件 | 与预期结果一致 | - |

1. openmpi库获取安装信息用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | openmpi库获取安装信息用例/ GN\_YYXK\_QTK \_OMK\_OPMIINFO | | | | |
| 用例说明 | | **测试目的:** 获取openmpi ,orte的详细信息。  **测试方法:** 利用ompi获取输出信息。  **判断依据:** 正确输出安装组件信息。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已安装openmpi-bin | 无 | 终端输入dpkg -S /usr/bin/ompi\_info | 输出安装包名以及路径。 | 与预期结果一致 | - |
| 步骤2 | 已安装openmpi-bin | 无 | 在终端运行ompi\_info –all,  ompi\_info --all | 输出openmpi, orte信息，编译信息，编译器信息，mca组件信息 | 与预期结果一致 | - |

1. openmpi库mpi应用运行用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | openmpi库运行mpi应用用例/ GN\_YYXK\_QTK\_OMK\_MPIA | | | | |
| 用例说明 | | **测试目的:** 通过mpi编译的openmpi 应用正常运行，并能够支持的各种参数。  **测试方法:** 利用mpirun运行应用。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已在测试文件夹目录, 已成功编译应用 | 无 | 终端输入mpirun ring\_c | 输出运行结果,程序正常运行 | 与预期结果一致 | - |
| 步骤2 | 已在测试文件夹目录, 已成功编译应用 | 无 | 在终端运行mpirun -np 2 hello\_c | 输出输出运行信息:  Hello, world, I am 0 of 2, (Open MPI v4.1.0, package: Debian OpenMPI, ident: 4.1.0, repo rev: v4.1.0, Dec 18, 2020, 87)  Hello, world, I am 1 of 2, (Open MPI v4.1.0, package: Debian OpenMPI, ident: 4.1.0, repo rev: v4.1.0, Dec 18, 2020, 87) | 与预期结果一致 | - |

1. openmpi库运行集群内应用用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | openmpi库运行集群内应用用例/ GN\_YYXK\_QTK\_OMK\_MPICA | | | | |
| 用例说明 | | **测试目的:** 通过mpi编译的openmpi 应用正常运行，并能够支持的各种参数，能够在集群内的其余机器运行。  **测试方法:** 利用mpirun运行另一台主机的应用。  **判断依据:** 应用正常运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 已在测试文件夹目录, 已成功编译应用, 已配置集群 | 无 | 终端输入mpirun -H manager ./examples/ ring\_c | 输出运行结果，程序运行正常 | 与预期结果一致 | - |
| 步骤2 | 已在测试文件夹目录, 已成功编译应用，已配置集群 | 无 | 在终端运行mpirun -H manager:2 ./examples/ hello\_c | 输出运行信息:  Hello, world, I am 0 of 2, (Open MPI v4.1.0, package: Debian OpenMPI, ident: 4.1.0, repo rev: v4.1.0, Dec 18, 2020, 87)  Hello, world, I am 1 of 2, (Open MPI v4.1.0, package: Debian OpenMPI, ident: 4.1.0, repo rev: v4.1.0, Dec 18, 2020, 87) | 与预期结果一致 |  |

1. Chromium网页浏览基本功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Chromium网页浏览器基本功能测试/ GN\_YYXK \_QTK\_CHRK\_JBGN | | | | |
| 用例说明 | | **测试目的:** Chromium网页浏览器可正常完成浏览网址、切换标签页、阅读PDF文档、添加书签页等操作。  **测试方法:**  1.测试Chromium网页浏览器新建和关闭多个标签页功能。  2.测试Chromium网页浏览器正常访问网页（www.baidu.com）、播放视频功能（www.iqiyi.com）。  3.测试Chromium网页浏览器可添加网址到书签栏的功能。  4.测试Chromium网页浏览器正常使用插件功能。  **合格判据:**  1.检测网络环境是否稳定。  2.检测待浏览网页是否支持访问。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Chromium浏览器工具已安装及网络连接正常。 | 无 | 点击打开Chromium网页浏览器；点击新建标签页的‘+’号；点击标签页的‘x’。 | 弹出空白的新标签页；弹出新的空白标签页；正常关闭。 | 与预期结果一致 | - |
| 步骤 2 | 已打开chromium浏览器。 | 被访问网址。 | 输入访问网址，点击转到。 | 进入并加载该网址内容。例如输入www.iqiyi.com，进入爱奇艺官方网站，点击任一视频进行观看。 | 与预期结果一致 | - |
| 步骤 3 | 已打开网站。 | 无 | 点击网址输入框右侧的星号标志添加该网址为书签页。 | 打开主菜单-书签，可在书签栏中找到该网址。 | 与预期结果一致 | - |
| 步骤 4 | 已打开chromium浏览器。 | 网址：<http://www.gov.cn/zhengce/pdfFile/downloadFile.htm>。 | 输入网址<http://www.gov.cn/zhengce/pdfFile/downloadFile.htm>，点击任一PDF文档。 | 任一PDF文档在网页浏览器界面使用插件进行查看与下载。 | 与预期结果一致 | - |

1. Chromium网页浏览器的可靠性测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Chromium网页浏览器的可靠性测试/ GN\_YYXK \_QTK\_CHRK\_DX | | | | |
| 用例说明 | | **测试目的:** Chromium网页浏览器能够在一定的负载下可靠的进行使用，不会发生崩溃现象。  **测试方法:**  1.测试Chromium网页浏览器新建30+标签页时的稳定性。  2.测试Chromium网页浏览器连续长时间播放视频过程中的可靠性。  3.测试Chromium网页浏览器的基础内存耗用的合理性。  **合格判据:**  1.检测网络环境是否稳定。  2.检测待浏览网页是否支持访问。  3.检测是否有其余进程占用内存。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Chromium浏览器工具已安装及网络连接正常。 | 无 | 点击新建标签页的‘+’号。 | 弹出新的空白标签页。 | 与预期结果一致 | - |
| 步骤 2 | 已打开Chromium网页浏览器。 | 终端输入命令：free  -m、free -h。 | 在终端中输入命令:free  -m，打开Chromium浏览器，再次在终端输入输入命令:free –h。 | 查看物理内存并记录，计算内存耗用即增量。 | 与预期结果一致 | - |
| 步骤 3 | 已打开Chromium网页浏览器。 | 地址栏输入：www.iqiyi.com。 | 新标签页地址栏输入“www.iqiyi.com”，按下Enter键，选择界面任一视频连续播放12小时。 | 无卡顿、崩溃、闪退等现象。 | 无卡顿、崩溃、闪退 | - |

1. Chromium网页浏览测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Chromium网页浏览器的性能测试/ GN\_YYXK \_QTK\_CHRK\_NCS | | | | |
| 用例说明 | | **测试目的:** Chromium网页浏览器需要具有良好的性能表现，包括但不限于加载网页的速度、加载图片速度、网页的稳定性等。  **测试方法:**  1.测试Chromium网页浏览器打开常用网页加载时间。  2.测试Chromium网页浏览器对Acid3的支持情况。  3.测试Chromium网页浏览器对css的支持情况。  4.测试Chromium网页浏览器对HTML5的支持情况。  **合格判据:**  1.检测网络环境是否稳定。  2.检测待浏览网页是否支持访问。  3. 检测是否有影响测试结果的页面及进程。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Chromium浏览器工具已安装及网络连接正常。 | 任意常用网站。 | 打开一个常用网站（qq邮箱），按下f12，点击网络标签，并勾选右侧“禁用缓存”，按下f5刷新网页。 | 记录下方load时间。 | 与预期结果一致 | - |
| 步骤 2 | 已打开Chromium网页浏览器。 | 网址http://acid3.acidtests.org/。 | 点击“+”打开新的页签，输入网址http://acid3.acidtests.org/，回车。 | 得分结果是100/100 | 与预期结果一致 | - |
| 步骤 3 | 已打开Chromium网页浏览器。 | 网址http://www.howtocr eate.co.uk/csstest.html。 | 点击“+”打开新的页签，输入网址http://www.howtocr eate.co.uk/csstest.html，回车，鼠标右键点击页面。 | 保存此页面到本地。 | 与预期结果一致 | - |
| 步骤 4 | 已打开Chromium网页浏览器。 | 无 | 本地保存的网站地址-鼠标右键点击-选择-使用Chromium网页浏览器打开。 | 打开成功并在左上角显示时间。 | 与预期结果一致 | - |

1. Chromium网络和音视频测试测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Chromium网络和音视频测试测试/ GN\_YYXK \_QTK\_ WLYSP | | | | |
| 用例说明 | | **测试目的:** Chromium 网络和音视频的功能。  **测试方法:** html5test测试项。  **合格判据:**  1.测试过程覆盖飞腾平台和龙芯平台。  2.测试过程和项目完整。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 可以访问网站的环境，正确安装chromium浏览器后的环境。 | 打开Chromium网页浏览器。 | 打开浏览器访问http://html5test.com/。 | 访问网站后会进行自动化测试，比较结果和正常本地浏览器访问此网站得到的打钩结果一致。 | 与预期结果一致 | - |

1. Chromium加解密功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Chromium加解密功能测试/ GN\_YYXK \_QTK\_JJM | | | | |
| 用例说明 | | **测试目的:** Chromium加解密和v8引擎跑分。  **测试方法:** 每个测试项都有分数(Richards基准，DeltaBlue基准，Crypto基准，RayTrace基准，EarleyBoyer基准，RegExp基准，Splay基准，NavierStokes基准)。  **合格判据:**  1.测试过程覆盖飞腾平台和龙芯平台。  2.测试过程和项目完整。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 能够上外网的环境，正确安装chromium浏览器后的环境。 | 打开Chromium网页浏览器。 | 访问http://chrome.360.cn/test/v8/run.html 网站。 | 访问成功。 | 与预期结果一致 | - |
| 步骤 2 | 能够上外网的环境，正确安装chromium浏览器后的环境。 | 打开Chromium网页浏览器。 | 运行V8基准套件测试。 | 成功显示出各测试项分数。 | 与预期结果一致 | - |

1. Chromium网页浏览测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Chromium网页浏览器的性能测试/ GN\_YYXK \_QTK\_CHRK\_NCS | | | | |
| 用例说明 | | **测试目的:** 浏览器插件和UI界面测试。  **测试方法:** 插件的安装，卸载，使用的测试；chromium的所有UI界面测试。  **合格判据:**  1.测试过程覆盖飞腾平台和龙芯平台。  2.测试过程和项目完整。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 能够上网的环境，并准备相应插件，正确安装chromium浏览器的环境。 | 导入插件包。 | 正常安装、卸载谷歌浏览器SwitchyOmega插件。 | 正常安装，卸载使用谷歌浏览器SwitchyOmega插件。 | 与预期结果一致 | - |
| 步骤 2 | 能够上网的环境，并准备相应插件，正确安装chromium浏览器的环境。 | 导入插件包。 | 点击chromium的所有UI界面。 | 无报错。 | 无报错 | - |

## 运行环境/GN\_ YXHJC

### 应用沙箱测试（王乐、王峰、王强、赵廉）

1. 基本功能测试用例（王乐）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 基本功能/ GN\_YXHJC\_YYSX\_JB | | | | |
| 用例说明 | | **测试目的：** 测试沙箱对运行环境基本功能的支持。  1.测试验证应用沙箱能支持设备挂载。  2.测试验证应用沙箱能进行用户切换。  3.测试验证应用沙箱能进行本地库调用。  4.测试验证应用沙箱能进行进程管理。  **测试方法：** 通过修改沙箱内外的配置文件信息、切换沙箱启动用户身份以及在沙箱内进行命令处理等等的方式,对沙箱支持运行环境的基本功能进行测试。  **合格判据：**  1.应用沙箱可以正常运行。  2.应用沙箱能支持设备挂载。  3.应用沙箱能进行用户切换。  4.应用沙箱能进行本地库调用。  5.应用沙箱能进行进程管理。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.应用沙箱已正常部署；  2.宿主机已挂载好相应的设备  3.应用沙箱已安装好测试镜像。 | 1.修改沙箱config文件。  2.创建并启动一个完整的沙箱实例并指定-i -t的参数以及/bin/bash的执行程序。 | 1.修改config文件调整沙箱进程启动时自动挂载的设备。  2.重启沙箱服务进程使配置修改生效。  3.启动一个沙箱实例使其挂载上我们设置的设备，同时给此实例启动一个shell方便确认结果。 | 1.成功拉取C++服务端应用框架镜像到本地；可通过沙箱命令查看本地镜像；启动本地C++服务端应用框架成功。  2.沙箱实例启动成功后可以在宿主机ps查看到沙箱实例进程。  3.能在沙箱实例启动的shell中查看到需要挂载的设备。 | 与预期结果一致 |  |
| 步骤 2 | 1.应用沙箱已正常启动；  2.应用沙箱已安装好测试镜像；  3.宿主机已含有普通测试用户和root用户。 | 宿主机切换至root用户以及普通用户。 | 1. 切换到root用户，使沙箱实例启动时切换至root用户。 2. 切换到普通用户使沙箱实例启动时切换至普通用户。 | 1.操作2完成后可以观察到shell的执行用户为root用户。  1.操作4完成后可以观察到shell的执行用户为普通用户。 | 与预期结果一致 | - |
| 步骤 3 | 1.应用沙箱已正常启动；  2.应用沙箱已安装好测试镜像；  3.宿主机已安装好需要调用的本地库。 | 设置-v参数使沙箱实例在启动时将指定的宿主机路径挂载至沙箱的指定路径中。 | 1.创建一个完整的沙箱实例，并在-v参数后填入指定的本地库路径  2.启动镜像实例并指定-i -t参数及/bin/bash的执行程序。 | 可以在沙箱shell中查看到本地库已加载至沙箱的指定目录中。 | 与预期结果一致 | - |
| 步骤 4 | 1.应用沙箱已正常启动；  2.应用沙箱已安装好测试镜像。 | 1. 创建并启动一个沙箱实例。 2. 停止沙箱实例。 | 通过启停一个沙箱实例观察沙箱是否可以管理沙箱实例进程。 | 1.操作1完成后可以在宿主机使用ps命令查看到沙箱实例进程已经存在。  1.操作2完成后可以在宿主机使用ps命令查看到沙箱实例进行已经停止。 | 与预期结果一致 | - |

1. 常见共享功能测试用例（王乐）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 常见共享功能/ GN\_YXHJC\_YYSX\_CJGX | | | | |
| 用例说明 | | **测试目的：** 测试沙箱对常见共享功能的支持：  1.测试验证应用沙箱能支持跨主机网络通信。  2.测试验证应用沙箱能支持常见ipc通信。  3.测试验证应用沙箱能支持应用间的内存映射文件共享，文件系统共享。  **测试方法：** 在同一台计算机以及多台计算机上的应用沙箱里启动特定的测试程序，测试沙箱对常见共享功能的支持。  **合格判据：**  1.应用沙箱可以正常运行。  2.应用沙箱能支持跨主机网络通信。  3.应用沙箱能支持常见ipc通信。  4.应用沙箱能支持应用间的内存映射文件共享，文件系统共享。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.应用沙箱已在两台机器正常部署启动；  2.应用沙箱均已安装好测试镜像。 | 跨主机通信相关程序启动与配置信息。 | 1.两台机器分别启动测试实例并进入shell界面。  2.在其中一台机器上启动跨主机通信服务端。  3.在另一台机器上启动跨主机通信客户端。  4.客户端和服务端互发消息并确认。 | 1.两台机器上实例启动成功  2.测试程序服务端启动成功。  3.测试程序客户端连接成功。  4.客户端和服务端互发消息且确认成功。 | 与预期结果一致 |  |
| 步骤 2 | 1.应用沙箱已正常启动  2.应用沙箱已安装好测试镜像。 | ipc通信相关程序启动与配置信息。 | 1.启动两个测试实例  2.同时在两台机器上执行ipc测试程序。 | 1.沙箱实例启动正常。  2.两个实例中的测试程序启动正常。  3.管道、消息队列、信号、信号量测试正常。 | 与预期结果一致 | - |
| 步骤 3 | 1.应用沙箱已正常启动  2.应用沙箱已安装好测试镜像。 | 内存映射文件共享和文件系统共享相关程序启动与配置信息。 | 1.启动两个测试实例  2.同时在两台机器上执行内存映射文件共享和文件系统共享测试程序。 | 1.沙箱实例启动正常。  2.两个实例中的测试程序启动正常。  3.内存映射文件共享和文件系统共享测试成功。 | 与预期结果一致 | - |

1. 界面展现功能测试用例（王乐）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 界面展现功能/ GN\_YXHJC \_YYSX\_JMZX | | | | |
| 用例说明 | | **测试目的：**测试沙箱支持服务端程序和客户端程序的运行，支持客户端界面的展现。  **测试方法：**测试沙箱里运行系统自带后台服务、图形界面和文字界面应用的运行，验证其对服务端和客户端程序运行以及客户端界面展现能力的支持作用。  **合格判据：**服务端和客户端程序均能运行，客户端程序的图形界面和文字界面均能正常展示。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 应用沙箱已正常部署，运行时框架基础镜像准备好。 | 应用沙箱命令。 | 应用沙箱命令加载运行时框架基础镜像，并进入沙箱。 | 成功显示运行时框架基础镜像加载，并进入沙箱。 | 与预期结果一致 | - |
| 步骤 2 | 后台服务httpd已经在沙箱中安装好。 | 服务启动命令。 | 在沙箱里启动基础镜像自带的httpd服务，并在沙箱外用客户端连接它。 | 客户端能功访问httpd服务。 | 成功访问httpd服务 | - |
| 步骤 3 | 命令ls已经准备好。 | ls查看目录命令。 | 在沙箱里启动命令行，并用ls等命令查看目录信息。 | 成功显示沙箱中的目录信息。 | 与预期结果一致 | - |
| 步骤  4 | 界面应用已经在沙箱中安装好。 | 界面应用名称。 | 在沙箱里启动命令行，并执行界面应用。 | 应用的界面成功展现。 | 与预期结果一致 | - |

1. 支持C++应用和库跨系统兼容测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 支持C++应用和库跨系统兼容/ GN\_ YXHJC \_YYSX\_KBB | | | | |
| 用例说明 | | **测试目的：**支持同一款CPU芯片下，基于运行时框架接口开发的C++应用和多版本库跨国产操作系统发行版的二进制兼容。  **测试方法：** 在不同国产操作系统发行版运行同一C++应用镜像；在不同国产操作系统发行版运行不同版本的QT应用镜像。  **合格判据：** 同一C++应用镜像在不同国产操作系统发行版能够兼容运行；不同版本的QT应用在不同国产操作系统发行版能够兼容运行 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 沙箱工具正常安装，应用程序正常安装。 | 沙箱工具命令。 | 在安装了银河麒麟V10 GF版操作系统的飞腾客户机上，用沙箱工具运行C++应用程序。 | 能够正常运行。 | 与预期结果一致 | - |
| 步骤 2 | 沙箱工具正常安装，应用程序正常安装。 | 沙箱工具命令。 | 在安装了UOS V20 JY版操作系统的飞腾客户机上，用沙箱工具运行C++应用程序。 | 能够正常运行。 | 与预期结果一致 | - |
| 步骤 3 | 沙箱工具正常安装，应用程序正常安装。 | 沙箱工具命令。 | 在安装了银河麒麟V10 GF版操作系统的龙芯客户机上，用沙箱工具运行C++应用程序。 | 能够正常运行。 | 与预期结果一致 | - |
| 步骤 4 | 沙箱工具正常安装，应用程序正常安装。 | 沙箱工具命令。 | 在安装了UOS V20 JY版操作系统的龙芯客户机上，用沙箱工具运行C++应用程序。 | 能够正常运行。 | 与预期结果一致 |  |
| 步骤 5 | 沙箱工具正常安装，应用程序正常安装。 | 沙箱工具命令。 | 在安装了银河麒麟V10 GF版操作系统的飞腾客户机上，用沙箱工具运行两个版本的QT应用程序。 | 能够兼容运行。 | 与预期结果一致 |  |
| 步骤 6 | 沙箱工具正常安装，应用程序正常安装。 | 沙箱工具命令。 | 在安装了UOS V20 JY版操作系统的飞腾客户机上，用沙箱工具运行两个版本的QT应用程序。 | 能够兼容运行。 | 与预期结果一致 |  |
| 步骤 7 | 沙箱工具正常安装，应用程序正常安装。 | 沙箱工具命令。 | 在安装了银河麒麟V10 GF版操作系统的龙芯客户机上，用沙箱工具运行两个版本的QT应用程序。 | 能够兼容运行。 | 与预期结果一致 |  |
| 步骤 8 | 沙箱工具正常安装，应用程序正常安装。 | 沙箱工具命令。 | 在安装了UOS V20 JY版操作系统的龙芯客户机上，用沙箱工具运行两个版本的QT应用程序。 | 能够兼容运行。 | 与预期结果一致 | - |

1. 支持Java应用和库跨系统兼容测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 支持Java应用和库跨系统兼容测试/GN\_ YXHJ \_YYSX\_KBB | | | | |
| 用例说明 | | **测试目的：** 支持同一款CPU芯片下，基于运行时框架接口开发的Java应用和多版本库跨国产操作系统发行版的二进制兼容。  **测试方法：** 同一款CPU芯片下，测试同一个Java应用镜像在麒麟、UOS操作系统下均能正常运行，验证Java应用跨国产操作系统发行版的二进制兼容性；同一款CPU芯片下，测试带不同版本库的Java运行时框架在麒麟、UOS操作系统下均能正常运行，验证Java多版本库跨国产操作系统发行版的二进制兼容。  **合格判据：** 能够检测应用能够在同芯片主机机平台下跨操作系统的二进制兼容运行；能够检测在其他库版本的运行时框架下的应用也能实现跨操作系统二进制兼容运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 沙箱命令安装并启动，准备好相关镜像。 | 应用启动命令。 | 进入测试程序后，应用镜像在基于飞腾芯片+麒麟操作系统中，依托沙箱及运行时框架镜像成功启动。 | 应用能够启动，成功创建应用沙箱。 | 成功创建应用沙箱 | - |
| 步骤 2 | 成功创建应用沙箱。 | 输入服务测试地址信息。 | 测试应用服务启动日志显示正常，提供的服务运行正常。 | 测试相关的Java应用服务已经启动；相关的服务能够访问。 | 与预期结果一致 | - |
| 步骤 3 | 沙箱命令安装并启动，准备好相关镜像。 | 应用启动命令。 | 相同的应用镜像在飞腾芯片的UOS操作系统中，依托沙箱及运行时框架镜像成功启动。 | 应用能够启动，成功创建应用沙箱。 | 与预期结果一致 | - |
| 步骤 4 | 成功创建应用沙箱。 | 输入服务测试地址信息。 | 测试应用服务启动正常，提供的服务运行正常， 检验应用可以跨操作系统二进制兼容运行。 | 测试相关的Java应用服务已经启动；相关的服务能够访问。 | 与预期结果一致 | - |
| 步骤 5 | 沙箱命令安装并启动。 | 无 | 选择新的Java运行时框架，确保它与上述步骤的运行时框架提供不同版本的库 | 新的运行时框架，与旧的框架里的测试相关的基础库版本不同 | 与预期结果一致 | - |
| 步骤 6 | 准备好相关镜像。 | 应用启动命令。 | 在这个Java运行时框架下编译的应用，依托沙箱及框架发行版镜像在麒麟操作系统中成功启动。 | 应用能够启动，成功创建应用沙箱。 | 与预期结果一致 | - |
| 步骤 7 | 成功创建应用沙箱。 | 输入服务测试地址信息。 | 测试应用服务启动正常，提供的服务运行正常。 | 测试相关的Java应用服务已经启动；相关的服务能够访问。 | 与预期结果一致 | - |
| 步骤 8 | 沙箱命令安装并启动，准备好相关镜像。 | 应用启动命令。 | 在上述步骤中产生的应用，依托沙箱及框架发行版镜像在UOS操作系统中成功启动。 | 应用能够启动，成功创建应用沙箱。 | 与预期结果一致 | - |
| 步骤 9 | 成功创建应用沙箱。 | 输入服务测试地址信息。 | 测试，应用服务启动正常，提供的服务运行正常，检验应用在多版本库场景下也能跨操作系统二进制兼容运行。 | 测试相关的Java应用服务已经启动；相关的服务能够访问。 | 与预期结果一致 | - |

1. 支持C++应用库的加载和调用测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 支持C++应用库的加载和调用/ GN\_ YXHJC\_YYSX\_JZDY | | | | |
| 用例说明 | | **测试目的：**支持当C++应用所需库与运行环境自带库出现版本冲突时，应用按需加载和调用自带的库版本。  **测试方法：** 查看C++应用程序优先加载和调用自带的库版本信息，是否与应用程序预期的库版本一致。  **合格判据：** C++应用正确加载和调用自带的库版本，运行正常。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 沙箱工具正常安装，应用程序正常安装。 | 沙箱工具命令。 | 使用沙箱工具启动C++应用，该应用自带了一个库，该库版本与运行时框架带的库版本不同。 | 应用能够正常启动。 | 应用正常启动 | - |
| 步骤 2 | 应用能够正常启动。 | env命令。 | 进入C++应用沙箱，查看环境变量中的库搜索路径信息。 | 环境变量包含C++应用自带库所在路径。 | 与预期结果一致 | - |
| 步骤 3 | 环境变量包含C++应用自带库所在路径。 | ldd命令。 | 执行ldd命令，查看C++应用加载及调用的依赖库信息。 | C++应用加载及调用的是自带的库版本。 | 与预期结果一致 | - |

1. 支持Java应用库的加载和调用测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 支持Java应用库的加载和调用测试/GN\_ YXHJ\_YYSX\_JZDY | | | | |
| 用例说明 | | **测试目的：** 支持当Java应用所需库与运行环境自带库出现版本冲突时，应用按需加载和调用自带的库版本。  **测试方法：** 通过在Java应用构建、打包和运行过程中，对自带与Java运行时框架中版本不一致的库的调用支持，验证Java应用按需加载和调用自带的库版本能力。  **合格判据：** 能够在构建、打包时使用自带的库；能够在运行时加载自带的库。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 编译脚本和工程文件已经定义。 | 编译应用脚本。 | 控制应用的工程脚本，对于库的选择使用指定目录下的自带库，而不是使用默认的Java运行时框架的库版本。 | 编译构建时能够使用自带的库。 | 与预期结果一致 | - |
| 步骤 2 | 沙箱镜像打包工具已经安装。 | 打包脚本。 | 执行打包操作，应用构建时按需打包所需要的依赖包，部分来自于Java运行时框架，部分来自于自带库。 | 能够将自带库和来自运行时框架的库一起打包到应用包中。 | 与预期结果一致 | - |
| 步骤 3 | 沙箱服务和沙箱工具已经具备。 | 已经安装的应用。 | 测试在应用在运行时，会加载相关的库，提供正确的服务，不会有库或类未找到异常。 | 应用能够正常运行，没有类加载错误。 | 与预期结果一致 | - |
| 步骤 4 | 应用能够正常运行，没有类加载错误。 | 库名称和版本。 | 检查在应用的运行环境里能够通过库名称识别或者日志显示应用运行时使用的库为自带的库。 | 检查相关的库已经被应用加载并使用。 | 与预期结果一致 | - |
| 步骤 5 | 检查相关的库已经被应用加载并使用。 | 库名称和版本。 | 检查应用在运行时，不会有多版本冲突问题。 | 确保对应重复的多版本库，只有一个库被加载。 | 与预期结果一致 | - |

1. 支持C++应用源码级兼容测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 支持C++应用源码级兼容/ GN\_YXHJC \_YYSX\_YYJR | | | | |
| 用例说明 | | **测试目的：**支持在不同硬件平台下C++JS应用源码级兼容。  **测试方法：**测试同一套C++应用源码在各国产平台下构建及运行。  **合格判据：**同一套源码在飞腾和龙芯平台上都能够构建及运行成功。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 通过浏览器进行登录访问应用远程构建系统。 | 远程构建信息。 | 登录应用远程构建系统，填写及设置待构建的应用程序名称、版本、构建后应用镜像名称、版本、应用依赖的运行时框架等信息。 | 远程构建信息正常填写及设置。 | 与预期结果一致 | - |
| 步骤 2 | 远程构建信息正常填写及设置。 | 待验证的应用压缩包名称。 | 选择应用工程本地上传选项，点击上传按钮，选择打包好的应用tar包。 | C++应用源码工程成功上传到远程服务器。 | 与结果一致 | - |
| 步骤 3 | C++应用源码工程成功上传到远程服务器。 | CPU架构。 | 勾选要构建的CPU架构（飞腾架构、龙芯架构），点击远程构建按钮，查看应用在各CPU架构平台构建的步骤信息。 | C++应用构建正确完成。 | C++应用构建正确完成 | - |
| 步骤 4 | C++应用构建正确完成。 | 部署信息。 | 将构建出的应用镜像分别安装在飞腾架构和龙芯架构客户机上，使用沙箱工具启动。 | C++应用能够正常运行。 | 与预期结果一致 | - |

1. 支持Java应用源码级兼容测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Java应用源码级兼容测试/GN\_YXHJ \_YYSX\_YYJR | | | | |
| 用例说明 | | **测试目的：**支持在不同硬件平台下Java JS应用源码级兼容。  **测试方法：**在不同硬件平台下使用同源的应用可以不做修改的编译运行；可以与运行框架结合选择相应平台下的依赖库。  **合格判据：**保证在不同硬件平台下应用源码（保持一份）能够编译成功，并且成功打包；保证在相应平台打包的应用可以在对应平台上运行。或者能够做到一份字节码应用，能够在各个国产平台上不需编译运行，做到字节码跨平台兼容。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已经部署沙箱工具和相关的编译打包工具。 | 带有源码的Java应用服务。 | 带有源码的Java应用服务在飞腾芯片平台上进行编译打包。 | 应用编译打包成功。 | 与预期结果一致 | - |
| 步骤 2 | 飞腾平台下应用编译打包成功。 | 相关的编译和打包脚本。 | 设置打包时依赖基于飞腾平台的Java运行时框架。 | Java运行时框架信息设置成功。 | 与预期结果一致 | - |
| 步骤 3 | 飞腾平台下应用、Java运行时框架镜像准备好。 | Java应用名称。 | 测试相关的应用包可以在飞腾平台上成功启动，并且各项服务运行正常。 | 应用能够启动各项服务正常。 | 与预期结果一致 | - |
| 步骤 4 | 龙芯平台下应用编译打包成功。 | 相关的编译和打包脚本。 | 把带有源码的Java应用服务在硬件龙芯平台上进行编译打包，在此过程中需保证源码不能进行修改。 | 应用编译打包成功。 | 应用编译打包成功 | - |
| 步骤 5 | 龙芯平台下应用、Java运行时框架镜像准备好。 | Java应用名称。 | 相关的应用包可以在龙芯平台上成功启动，并且各项服务运行正常。 | 通过上述过程验证应用在不同的硬件平台下能够支持源码级兼容。 | 与预期结果一致 | - |
| 步骤 6 | 申威平台下应用编译打包成功。 | 相关的编译和打包脚本。 | 把带有源码的Java应用服务在硬件申威平台上进行编译打包，在此过程中需保证源码不能进行修改。 | 应用编译打包成功。 | 应用编译打包成功 |  |
| 步骤 7 | 申威平台下应用、Java运行时框架镜像准备好。 | Java应用名称。 | 相关的应用包可以在申威平台上成功启动，并且各项服务运行正常。 | 通过上述过程验证应用在不同的硬件平台下能够支持源码级兼容。 | 与预期结果一致 |  |

1. C/C++类型应用运行支撑测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | C/C++类型应用运行支撑/ GN\_ YXHJC\_YYSX\_CYY | | | | |
| 用例说明 | | **测试目的：**多个应用的依赖库存在版本冲突时，支持各自加载所需的库版本，支持同一个C/C++应用的多个版本同机兼容运行。  **测试方法：** 同时运行不同版本的C++应用程序，测试应用沙箱的环境隔离功能。  **合格判据：** 不同版本的C++应用能够正常启动，分别加载各自所需的库版本。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 安装沙箱工具及C++应用程序两个版本。 | 沙箱命令、应用程序1.0名称。 | 使用沙箱工具启动版本1.0的C++应用程序。 | C++应用程序1.0能够正常启动。 | 与预期结果一致 | - |
| 步骤 2 | 应用程序存在配置冲突（如端口配置）时需要调整配置。 | 沙箱命令、应用程序1.1名称。 | 使用沙箱工具启动版本1.1的C++应用程序。 | C++应用程序1.1能够正常启动。 | 与结果一致 | - |
| 步骤 3 | C++应用程序1.0和1.1都能够正常启动。 | 应用依赖的库信息查看命令ldd。 | 使用沙箱工具进入版本1.0的C++应用沙箱，查看该应用依赖的库信息。 | 库版本为1.0。 | 与预期结果一致 | - |
| 步骤 4 | 库版本为1.0。 | 应用依赖的库信息查看命令ldd。 | 使用沙箱工具进入版本1.1的C++应用沙箱，查看该应用依赖的库信息。 | 库版本为1.1。 | 与预期结果一致 |  |
| 步骤 5 | 库版本为1.1。 | 对比两个应用程序依赖库的版本。 | 比较3和4依赖库信息。 | 两个不同版本的应用分别加载各自所需的库版本。 | 与结果一致 | - |

1. Java类型应用运行支撑测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Java类型应用运行支撑测试/GN\_ YXHJ \_YYSX\_JLXYY | | | | |
| 用例说明 | | **测试目的：**当多个Java应用的依赖库存在版本冲突时，支持各自加载所需的库版本，支持同一个应用的多个版本同机兼容运行。  **测试方法：**利用应用编译和沙箱运行测试不同Java应用各自加载所需要的库版本；在同一台机器上可以运行应用的不同版本，即同一个应用的多个版本同机兼容运行。  **合格判据：**同一个Java应用的不同版本使用各自的库加载运行；同一个Java应用的多个版本可以同机同时运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 相关的应用编译脚本和依赖库准备好。 | 编译打包命令。 | Java应用版本A加载自己的依赖库testa.jar，进行编译打包。 | 应用版本A能够成功编译打包。 | 与预期结果一致 | - |
| 步骤 2 | 相关的应用编译脚本和依赖库准备好。 | 编译打包命令。 | Java应用版本B加载自己的依赖库testb.jar，进行编译打包。 | 应用版本B能够成功编译打包。 | 与预期结果一致 | - |
| 步骤 3 | 应用版本A和B均能编译打包成功。 | 应用A和应用B的端口信息 | 修改配置，使应用版不B的服务端口不产生冲突 | 配置合适的服务端口。 | 配置合适的服务端口 | - |
| 步骤 4 | 应用成功编译生成镜像。 | 运行应用命令。 | 启动应用版本A在国产平台机器上运行，检测有运行时调用的是依赖库testa.jar的test()方法打印的信息，如： “这是testa.jar的方法”。 | 应用版本A运行正常并且打印期望的信息。 | 与预期结果一致 | - |
| 步骤 5 | 应用成功编译生成镜像。 | 运行应用命令。 | 支持应用版本B在相同的国产平台机器上同时运行，检测有运行时调用的是依赖库testb.jar的test()方法打印的信息，如： “这是testb.jar的方法”。 | 应用版本B运行正常并且打印期望的信息。 | 与预期结果一致 | - |

1. 支持应用沙箱内进行库的检查测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 支持应用沙箱内进行库的检查/ GN\_ YXHJ\_YYSX\_KJC | | | | |
| 用例说明 | | **测试目的：** 支持在应用沙箱内进行库的兼容性检查和确认。  **测试方法：** 在运行时框架应用沙箱内执行运行时框架、应用运行库及系统库的检查脚本。  **合格判据：** 运行时框架内部包含的应用框架、应用运行库及系统库的静态依赖关系自洽无冲突。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 安装QT4运行时框架、测试程序及沙箱工具。 | 沙箱工具命令、测试程序。 | 使用沙箱工具，基于测试程序交互式运行QT4应用框架。 | 测试程序交互式运行成功。 | 与结果一致 | - |
| 步骤 2 | 测试程序交互式运行成功。 | 应用框架检查脚本。 | 执行应用框架检查脚本，检查QT4应用框架静态依赖关系。 | QT4应用框架依赖关系自洽无冲突。 | 与预期结果一致 | - |
| 步骤 3 | QT4应用框架依赖关系自洽无冲突。 | 应用运行库检查脚本。 | 执行应用运行库检查脚本，检查应用运行库静态依赖关系。 | 应用运行库依赖关系自洽无冲突。 | 与预期结果一致 | - |
| 步骤 4 | 应用运行库依赖关系自洽无冲突。 | 系统库检查脚本。 | 执行系统库检查脚本，检查系统库静态依赖关系。 | 系统库依赖关系自洽无冲突。 | 与结果一致 | - |
| 步骤 5 | 安装QT5运行时框架。 | 沙箱工具命令。 | 使用沙箱工具，基于测试程序交互式运行QT5运行时框架。 | 测试程序交互式运行成功。 | 与结果一致 | - |
| 步骤 6 | 测试程序交互式运行成功。 | 应用框架检查脚本。 | 执行应用框架检查脚本，检查QT5应用框架静态依赖关系。 | QT5应用框架依赖关系自洽无冲突。 |  | - |
| 步骤 7 | QT5应用框架依赖关系自洽无冲突。 | 应用运行库检查脚本。 | 执行应用运行库检查脚本，检查应用运行库静态依赖关系。 | 应用运行库依赖关系自洽无冲突。 | 与预期结果一致 | - |
| 步骤 8 | 应用运行库依赖关系自洽无冲突。 | 系统库检查脚本。 | 执行系统库检查脚本，检查系统库静态依赖关系。 | 依赖关系自洽无冲突。 | 与预期结果一致 | - |

1. 应用沙箱性能测试用例（赵廉）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 应用沙箱的unixBench性能达到同机docker容器的80%/ GN\_YXHJ\_YYSX\_XN | | | | |
| 用例说明 | | **测试目的:**测试应用沙箱的性能。  **测试方法:**在应用沙箱内执行unixBench，结果与同机docker容器的unixBench结果相比；  **合格判据:** 比值〉=80%。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 应用沙箱和基础镜像在飞腾计算机上准备好。 | 基础镜像。 | 在飞腾计算机上运行应用沙箱，加载基础镜像，并安装和运行unixBench。 | 得到结果A。 | 可以得到结果A | - |
| 步骤 2 | Docker容器与环境和基础镜像在飞腾计算机上准备好。 | 基础镜像。 | 在飞腾计算机上运行docker，加载基础镜像，并安装和运行unixBench。 | 得到结果B。 | 可以得到结果B | - |
| 步骤 3 | 结果A和B均已得出。 | A、B。 | 计算A/B，得出结果C。 | C〉=80%。 | 与预期结果一致 | - |
| 步骤 4 | 其他平台准备好。 | 基础镜像。 | 在其他平台下测试，得出相应的A、B、C。 | C〉=80%。 | 与预期结果一致 |  |

### 语言运行时测试（赵廉、倪宝珠）

1. JDK8-对象测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK8-对象测试/ GN\_YXHJ \_YYXS\_JAVA\_DXCS8 | | | | |
| 用例说明 | | **测试目的：**验证JAVA可以使用基本数据类型，数组、字符串，创建一个新的类，并可以正常使用java面向对象中的继承、重载和多态。  **测试方法：**测试JAVA基本数据类型、数组、字符串，new一个新的类，并实现继承、重载和多态。  **合格判据：**  1.检测基本数据类型是否全覆盖。  2.测试数组、字符是否覆盖  3.测试类是否包含了继承，重载和多态功能 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK8版本。 | 运行Java编译器。 | 编译java文件，Java文件中包括Java基本数据类型、数组、字符串，new一个新的类，并实现继承、重载和多态等代码。 | java文件编译成class文件，没有bug。 | 与结果一致 | - |
| 步骤 2 | java文件编译成class文件。 | 执行Java class文件命令。 | 启动终端运行，执行查看结果。 | 输出结果无报错。 | 与预期结果一致 | - |

1. JDK8-语法测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK8-语法测试/ GN\_YXHJ \_YYXS\_JAVA\_ YFCS8 | | | | |
| 用例说明 | | **测试目的：** 验证JAVA可以使用循环和条件判断功能，条件判断为用户输入不同，得到结果不同。  **测试方法：** 编写程序对循环和条件判断功能进行测试。  **合格判据：**  1.检测循环功能包括for，while等  2.测试多种不同的输入，覆盖多种情况 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK8版本。 | 运行Java编译器。 | 编译java文件，Java文件中包括for循环和while条件判断功能相关的代码。 | java文件编译成class文件，没有bug。 | 与预期结果一致 | - |
| 步骤 2 | 命令行运行Java程序。 | 执行Java class文件命令。 | 启动终端运行，执行查看结果。 | 输出结果无报错。 | 输出结果无报错 | - |

1. JDK8-引包测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK8-引包测试/ GN\_YXHJ \_YYXS\_JAVA\_ YBCS8 | | | | |
| 用例说明 | | **测试目的：** 验证Java可以引入相关包，实现特定功能。  **测试方法：** 编写程序引入java.io、java.util、java.lang、java.text等基础功能包，并对引入的包的功能进行简单验证。  **合格判据：** 测试多种不同的包，尽量覆盖常用的使用场景 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK8版本。 | 运行Java编译器。 | 编译java文件，Java文件中包含java.io、java.util、java.lang、java.text等基础功能包的import代码。 | java文件编译成class文件，没有bug。 | 与预期结果一致 | - |
| 步骤 2 | 命令行运行Java程序。 | 执行Java class文件命令。 | 启动终端运行，执行查看结果。 | 输出结果无报错。 | 输出结果无报错 | - |

1. JDK8-TCP测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK8-TCP测试/ GN\_YXHJ \_YYXS\_JAVA\_ TCP8 | | | | |
| 用例说明 | | **测试目的：** 验证JAVA中的TCP通信功能。  **测试方法：** 通过JAVA中的ServerSocket类来实现对指定IP和指定端口的监听。  **合格判据：** 检测发送的信息的类型，包括字母，数字，特殊符号等。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK8版本。 | 编译、执行Java服务端程序命令。 | 执行服务端java程序，程序中含有打开和监听socket、被动建立连接等代码。 | 服务端java程序进入连接监听状态。 | 与结果一致 | - |
| 步骤 2 | 服务端java程序进入连接监听状态。 | 编译、执行Java客户端程序命令。 | 执行客户端java程序，程序中含有主动建立socket连接、发送请求等代码。 | 与服务端java程序建立连接。 | 与结果一致 | - |
| 步骤 3 | 与服务端java程序建立连接成功。 | 客户端向服务端发送tcp请求。 | 客户端向服务端基于tcp协议发送信息。 | 服务端收到客户端发送来的信息，在终端显示出来。 | 与预期结果一致 | - |
| 步骤 4 | 服务端收到tcp协议请求。 | 客户端退出。 | 关闭服务端和客户端连接。 | 连接正常关闭，无报错。 | 与预期结果一致 | - |

1. JDK8-UDP测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK8-UDP测试/ GN\_YXHJ \_YYXS\_JAVA\_ UDP8 | | | | |
| 用例说明 | | **测试目的：** 验证UDP编程中DatagramSocket监听指定的端口功能。  **测试方法：** 通过JAVA中的DatagramSocket类来实现对指定端口的监听。  **合格判据：** 检测发送的信息的类型，包括字母，数字，特殊符号等。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK8版本。 | 编译、执行Java服务端程序命令。 | 执行服务端java程序，程序中含有打开和监听DatagramSocket的代码。 | 进入UDP数据接收状态。 | 与预期结果一致 | - |
| 步骤 2 | 服务端java程序进入数据接收状态。 | 编译、执行Java客户端程序命令。 | 客户端执行java文件，程序中含有主动发送UDP报文代码。 | 向服务端进行UDP数据发送。 | 与预期结果一致 | - |
| 步骤 3 | 向服务端进行UDP数据发送。 | 无 | UDP服务端接收到客户端发来的信息并打印。 | UDP服务端收到客户端发送来的信息，在终端显示出来。 | 与预期结果一致 | - |
| 步骤 4 | UDP服务端在终端显示出收到的信息。 | 客户端退出，服务端退出。 | 关闭服务端和客户端。 | 无报错。 | 无报错 | - |

1. JDK8-多线程测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK8-多线程测试/ GN\_YXHJ \_YYXS\_JAVA\_ DXC8 | | | | |
| 用例说明 | | **测试目的：** 验证JAVA中的多线程功能。  **测试方法：** 使用Thread类创建新线程并start()启动。使用Executors类创建并使用线程池。  **合格判据：**  1.使用三种及以上方法实现多线程，尽可能多覆盖不同情况。  2.java程序互斥访问资源，互斥成功无卡死。  3.线程池创建并执行并行任务，无报错。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK8版本。 | 编译、执行Java服务端程序命令。 | 选择编译好的java文件并执行，代码中含有三种以上创建新线程的程序。 | 终端显示多线程创建成功，无报错。 | 与结果一致 | - |
| 步骤 2 | 多线程创建成功，无报错。 | 无 | Java多线程程序互斥访问同一个资源。 | 终端显示线程同步成功，无报错，无卡死现象。 | 与结果一致 |  |
| 步骤 3 | 线程同步成功，无报错。 | 无 | 执行创建并使用线程池代码。 | 终端显示线程池创建成功，同时执行并行任务，无报错。 | 与预期结果一致 |  |
| 步骤 4 | 线程池同时执行并行任务，无报错。 | 程序退出命令。 | 程序关闭。 | 无报错。 | 无报错 | - |

1. JDK8库-垃圾回收机制测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK8库-垃圾回收机制测试/ GN\_YXHJ \_YYXS\_JAVA\_ JDK8LJHS | | | | |
| 用例说明 | | **测试目的：** 验证JDK8的垃圾回收机制。  **测试方法：** gc正执行，输出gc相关信息且无OOM报错。  **合格判据：**  1.测试过程覆盖飞腾平台和龙芯平台。  2.测试过程和项目完整 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1. JDK8运行环境正常安装和部署；  2. jvm新生代内存设置为100M。 | 打开示例程序jvmTest。 | 运行“jvmTest”示例程序。 | jvmTest运行成功。 | 与预期结果一致 | - |
| 步骤 2 | jvmTest运行成功。 | gc回收命令。 | 执行启动参数：java -verbose：gc -Xms200M -Xmx200M -Xmn100M -XX：+PrintGCDetails -XX：SurvivorRatio=8 -XX：MaxTenuringThreshold=1。 | gc执行成功，程序未崩溃。 | 与预期结果一致 | - |

1. JDK-8性能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK性能测试/ GN\_YXHJ \_YYXS\_JAVA\_XN11 | | | | |
| 用例说明 | | **测试目的:**对JDK11的性能进行验证。  **测试方法:** 通过specJVM benchmark工具，分别运行在飞腾计算机和intel金牌5115计算机上，除以CPU峰值后，对比性能。  **合格判据:** 在飞腾计算机上测试结果除以CPU峰值后，与intel金牌5115计算机上测试结果除以CPU峰值的比，大于80% | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK 11已成功安装在飞腾计算机上。 | 无 | 在飞腾计算机上运行specJVM，得到结果A1。 | 成功运行specJVM，得到结果A1。 | 可得到结果A1 | - |
| 步骤 2 | SpecCPU已成功安装在飞腾计算机上。 | 无 | 在飞腾计算机上运行SpecCPU，得到结果B1。 | 成功运行SpecCPU，得到结果B1。 | 可得到结果B1 | - |
| 步骤 3 | JDK 11已成功安装在intel金牌5115计算机上。 | 无 | 在intel金牌5115计算机上运行specJVM，得到结果A2。 | 成功运行specJVM，得到结果A2。 | 可得到结果A2 | - |
| 步骤 4 | SpecCPU已成功安装在intel金牌5115计算机上。 | 无 | 在intel金牌5115计算机上运行SpecCPU，得到结果B2。 | 成功运行SpecCPU，得到结果B2。 | 可得到结果B2 | - |
| 步骤 5 | A1、B1、A2、B2均已测试出来。 | A1、B1、A2、B2。 | 计算（A1/B1）/（A2/B2），得到性能比。 | 性能比〉=80%。 | 与预期结果一致 |  |

1. JDK11-对象测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK11-对象测试/ GN\_YXHJ \_YYXS\_JAVA11\_DXCS11 | | | | |
| 用例说明 | | **测试目的：**验证JAVA可以使用基本数据类型，数组、字符串，创建一个新的类，并可以正常使用java面向对象中的继承、重载和多态。  **测试方法：**测试JAVA基本数据类型、数组、字符串，new一个新的类，并实现继承、重载和多态。  **合格判据：**  1.检测基本数据类型是否全覆盖。  2.测试数组、字符是否覆盖  3.测试类是否包含了继承，重载和多态功能 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK11版本。 | 运行Java编译器。 | 编译java文件，Java文件中包括Java基本数据类型、数组、字符串，new一个新的类，并实现继承、重载和多态等代码。 | java文件编译成class文件，没有bug。 | 与结果一致 | - |
| 步骤 2 | java文件编译成class文件。 | 执行Java class文件命令。 | 启动终端运行，执行查看结果。 | 输出结果无报错。 | 与预期结果一致 | - |

1. JDK11-语法测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK11-语法测试/ GN\_YXHJ \_YYXS\_JAVA11\_ YFCS11 | | | | |
| 用例说明 | | **测试目的：** 验证JAVA可以使用循环和条件判断功能，条件判断为用户输入不同，得到结果不同。  **测试方法：** 编写程序对循环和条件判断功能进行测试。  **合格判据：**  1.检测循环功能包括for，while等  2.测试多种不同的输入，覆盖多种情况 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK11版本。 | 运行Java编译器。 | 编译java文件，Java文件中包括for循环和while条件判断功能相关的代码。 | java文件编译成class文件，没有bug。 | 与预期结果一致 | - |
| 步骤 2 | 命令行运行Java程序。 | 执行Java class文件命令。 | 启动终端运行，执行查看结果。 | 输出结果无报错。 | 输出结果无报错 | - |

1. JDK11-引包测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK11-引包测试/ GN\_YXHJ \_YYXS\_JAVA11\_ YBCS11 | | | | |
| 用例说明 | | **测试目的：** 验证Java可以引入相关包，实现特定功能。  **测试方法：** 编写程序引入java.io、java.util、java.lang、java.text等基础功能包，并对引入的包的功能进行简单验证。  **合格判据：** 测试多种不同的包，尽量覆盖常用的使用场景 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK11版本。 | 运行Java编译器。 | 编译java文件，Java文件中包含java.io、java.util、java.lang、java.text等基础功能包的import代码。 | java文件编译成class文件，没有bug。 | 与预期结果一致 | - |
| 步骤 2 | 命令行运行Java程序。 | 执行Java class文件命令。 | 启动终端运行，执行查看结果。 | 输出结果无报错。 | 输出结果无报错 | - |

1. JDK11-TCP测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK11-TCP测试/ GN\_YXHJ \_YYXS\_JAVA11\_ TCP11 | | | | |
| 用例说明 | | **测试目的：** 验证JAVA中的TCP通信功能。  **测试方法：** 通过JAVA中的ServerSocket类来实现对指定IP和指定端口的监听。  **合格判据：** 检测发送的信息的类型，包括字母，数字，特殊符号等。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK11版本。 | 编译、执行Java服务端程序命令。 | 执行服务端java程序，程序中含有打开和监听socket、被动建立连接等代码。 | 服务端java程序进入连接监听状态。 | 与结果一致 | - |
| 步骤 2 | 服务端java程序进入连接监听状态。 | 编译、执行Java客户端程序命令。 | 执行客户端java程序，程序中含有主动建立socket连接、发送请求等代码。 | 与服务端java程序建立连接。 | 与结果一致 | - |
| 步骤 3 | 与服务端java程序建立连接成功。 | 客户端向服务端发送tcp请求。 | 客户端向服务端基于tcp协议发送信息。 | 服务端收到客户端发送来的信息，在终端显示出来。 | 与预期结果一致 | - |
| 步骤 4 | 服务端收到tcp协议请求。 | 客户端退出。 | 关闭服务端和客户端连接。 | 连接正常关闭，无报错。 | 与预期结果一致 | - |

1. JDK11-UDP测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK11-UDP测试/ GN\_YXHJ \_YYXS\_JAVA11\_ UDP11 | | | | |
| 用例说明 | | **测试目的：** 验证UDP编程中DatagramSocket监听指定的端口功能。  **测试方法：** 通过JAVA中的DatagramSocket类来实现对指定端口的监听。  **合格判据：** 检测发送的信息的类型，包括字母，数字，特殊符号等。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK11版本。 | 编译、执行Java服务端程序命令。 | 执行服务端java程序，程序中含有打开和监听DatagramSocket的代码。 | 进入UDP数据接收状态。 | 与预期结果一致 | - |
| 步骤 2 | 服务端java程序进入数据接收状态。 | 编译、执行Java客户端程序命令。 | 客户端执行java文件，程序中含有主动发送UDP报文代码。 | 向服务端进行UDP数据发送。 | 与预期结果一致 | - |
| 步骤 3 | 向服务端进行UDP数据发送。 | 无 | UDP服务端接收到客户端发来的信息并打印。 | UDP服务端收到客户端发送来的信息，在终端显示出来。 | 与预期结果一致 | - |
| 步骤 4 | UDP服务端在终端显示出收到的信息。 | 客户端退出，服务端退出。 | 关闭服务端和客户端。 | 无报错。 | 无报错 | - |

1. JDK11-多线程测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK11-多线程测试/ GN\_YXHJ \_YYXS\_JAVA11\_ DXC11 | | | | |
| 用例说明 | | **测试目的：** 验证JAVA中的多线程功能。  **测试方法：** 使用Thread类创建新线程并start()启动。使用Executors类创建并使用线程池。  **合格判据：**  1.使用三种及以上方法实现多线程，尽可能多覆盖不同情况。  2.java程序互斥访问资源，互斥成功无卡死。  3.线程池创建并执行并行任务，无报错。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK已经正常安装，并为JDK11版本。 | 编译、执行Java服务端程序命令。 | 选择编译好的java文件并执行，代码中含有三种以上创建新线程的程序。 | 终端显示多线程创建成功，无报错。 | 与结果一致 | - |
| 步骤 2 | 多线程创建成功，无报错。 | 无 | Java多线程程序互斥访问同一个资源。 | 终端显示线程同步成功，无报错，无卡死现象。 | 与结果一致 |  |
| 步骤 3 | 线程同步成功，无报错。 | 无 | 执行创建并使用线程池代码。 | 终端显示线程池创建成功，同时执行并行任务，无报错。 | 与预期结果一致 |  |
| 步骤 4 | 线程池同时执行并行任务，无报错。 | 程序退出命令。 | 程序关闭。 | 无报错。 | 无报错 | - |

1. JDK11库-垃圾回收机制测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK11库-垃圾回收机制测试/ GN\_YXHJ \_YYXS\_JAVA11\_ JDK11LJHS | | | | |
| 用例说明 | | **测试目的：** 验证JDK11的垃圾回收机制。  **测试方法：** gc正执行，输出gc相关信息且无OOM报错。  **合格判据：**测试过程和项目完整 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1. JDK11运行环境正常安装和部署；  2. jvm新生代内存设置为100M。 | 打开示例程序jvmTest。 | 运行“jvmTest”示例程序。 | jvmTest运行成功。 | 与预期结果一致 | - |
| 步骤 2 | jvmTest运行成功。 | gc回收命令。 | 执行启动参数：java -verbose：gc -Xms200M -Xmx200M -Xmn100M -XX：+PrintGCDetails -XX：SurvivorRatio=8 -XX：MaxTenuringThreshold=1。 | gc执行成功，程序未崩溃。 | 与预期结果一致 | - |

1. JDK-11性能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JDK性能测试/ GN\_YXHJ \_YYXS\_JAVA11\_XN11 | | | | |
| 用例说明 | | **测试目的:**对JDK11的性能进行验证。  **测试方法:** 通过specJBB benchmark工具，分别运行在飞腾计算机和intel金牌5115计算机上，除以CPU峰值后，对比性能。  **合格判据:** 在飞腾计算机上测试结果除以CPU峰值后，与intel金牌5115计算机上测试结果除以CPU峰值的比，大于80% | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JDK 11已成功安装在飞腾计算机上。 | 无 | 在飞腾计算机上运行specJBB，得到结果A1。 | 成功运行specJBB，得到结果A1。 | 可得到结果A1 | - |
| 步骤 2 | SpecCPU已成功安装在飞腾计算机上。 | 无 | 在飞腾计算机上运行SpecCPU，得到结果B1。 | 成功运行SpecCPU，得到结果B1。 | 可得到结果B1 | - |
| 步骤 3 | JDK 11已成功安装在intel金牌5115计算机上。 | 无 | 在intel金牌5115计算机上运行specJBB，得到结果A2。 | 成功运行specJBB，得到结果A2。 | 可得到结果A2 | - |
| 步骤 4 | SpecCPU已成功安装在intel金牌5115计算机上。 | 无 | 在intel金牌5115计算机上运行SpecCPU，得到结果B2。 | 成功运行SpecCPU，得到结果B2。 | 可得到结果B2 | - |
| 步骤 5 | A1、B1、A2、B2均已测试出来。 | A1、B1、A2、B2。 | 计算（A1/B1）/（A2/B2），得到性能比。 | 性能比〉=80%。 | 与预期结果一致 |  |

1. Luajit 基础功能用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Luajit 基础功能用例/ GN\_YXHJ\_YYXS\_LU\_LANG | | | | |
| 用例说明 | | **测试目的:** 测试luajit基础功能。  **测试方法:** 用luajit运行对应的测试用例。  **判断依据:** 测试用例运行通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 当前路径为测试文件夹，  已安装luajit | 无 | 终端输入:  luajit test.lua lang | 输出运行结果，程序运行正常。  输出:  passed | 与预期结果一致 |  |

1. Luajit FFI功能用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Luajit v2.1.0 ffi功能用例/ GN\_YXHJ\_YYXS\_LU\_LIBFFI | | | | |
| 用例说明 | | **测试目的:** 测试luajit ffi功能。  **测试方法:** 用luajit运行对应的测试用例。  **判断依据:** 测试用例运行通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 当前路径为测试文件夹，  已安装luajit | 无 | 终端输入:  luajit test.lua lib/ffi | 输出运行结果，程序运行正常。  输出:  passed | 与预期结果一致 |  |

1. Luajit数学库用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Luajit 数学库功能用例/ GN\_YXHJ\_YYXS\_LU\_LIBMATH | | | | |
| 用例说明 | | **测试目的:** 测试luajit 数学库功能。  **测试方法:** 用luajit运行对应的测试用例。  **判断依据:** 测试用例运行通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 当前路径为测试文件夹，  已安装luajit | 无 | 终端输入:  luajit test.lua lib/math | 输出运行结果，程序运行正常。  输出:  passed | 与预期结果一致 |  |

1. Luajit 字符串库功能用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Luajit 字符串库用例/ GN\_YXHJ\_YYXS\_LU\_LIBSTRING | | | | |
| 用例说明 | | **测试目的:** 测试luajit 字符串库功能。  **测试方法:** 用luajit运行对应的测试用例。  **判断依据:** 测试用例运行通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 当前路径为测试文件夹，  已安装luajit | 无 | 终端输入:  luajit test.lua lib/string | 输出运行结果，程序运行正常。  输出:  passed | 与预期结果一致 |  |

1. Luajit table库功能用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Luajit 字符串库用例/ GN\_YXHJ\_YYXS\_LU\_LIBTABLE | | | | |
| 用例说明 | | **测试目的:** 测试luajit table库功能。  **测试方法:** 用luajit运行对应的测试用例。  **判断依据:** 测试用例运行通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤1 | 当前路径为测试文件夹，  已安装luajit | 无 | 终端输入:  luajit test.lua lib/table | 输出运行结果，程序运行正常。  输出:  passed | 与预期结果一致 |  |

1. 浏览器简单调试运行JavaScript代码测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 浏览器简单调试运行JavaScript代码测试/ GN\_YXHJC \_YYXS\_JS\_JDTS8 | | | | |
| 用例说明 | | **测试目的：** 在浏览器中进行简单调试运行JavaScript代码。  **测试方法：** 在浏览器开发者工具进行调试运行JavaScript代码。  **合格判据：** 成功打开开发者工具，并在控制台输出代码运行结果。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 浏览器已成功安装，浏览器js引擎为8.9.255.24。 | 打开浏览器。 | 在网页中按“F12”键。 | 打开开发者工具。 | 与预期结果一致 | - |
| 步骤 2 | 打开开发者工具。 | 简单JavaScript代码。 | 打开控制台，在控制台窗口输入console.log("Hello JavaScript") ，按下“回车键”执行。 | 在控制台窗口输出结果：Hello JavaScript。 | 与结果一致 | - |
| 步骤 3 | 简单JavaScript代码执行成功。 | 无。 | 点击“Clear console”，退出。 | 可清空控制台窗口代码内容。 | 与预期结果一致 | - |

1. JavaScript对事件的反应测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JavaScript对事件的反应测试/ GN\_YXHJC \_YYXS\_JS\_SJFY8 | | | | |
| 用例说明 | | **测试目的：** 对用户在浏览器的操作进行反应。  **测试方法：** 在浏览器实现网页刷新、点击事件反应。  **合格判据：** 成功响应用户在浏览器网页的操作。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 浏览器已成功安装，浏览器js引擎为8.9.255.24。 | 打开浏览器。 | 将编辑好的JavaScript代码，放到HTML文件中，使用浏览器打开该文件。 | 成功打开HTML文件。 | 打开HTML文件 | - |
| 步骤 2 | 成功打开HTML文件。 | 按下“点击”按钮。 | 点击“点击事件”按钮。 | 检测到浏览器网页响应点击事件。 | 与预期结果一致 | - |
| 步骤 3 | 检测到浏览器网页响应点击事件。 | 按下“刷新”按钮。 | 点击浏览器主界面“刷新”按钮。 | 检测到浏览器网页响应刷新事件。 | 与预期结果一致 | - |

1. JavaScript解析器解释功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JavaScript解析器解释功能测试/ GN\_YXHJC \_YYXS\_JS\_ JSGN8 | | | | |
| 用例说明 | | **测试目的：** V8解释器解释功能测试。  **测试方法：** V8作为一个JS解释器，需要对V8解释器解释功能进行测试，对于常量，变量，数组，字符串，函数等正常解释执行。  **合格判据：** 正常解释和执行所有js语句。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.使用交叉编译完成v8，mjsunit。  2.将编译好的v8环境部署到目标机器。  3.在目标机器对解释器功能进行验证。 | 声明常量代码解释执行命令。 | 声明常量功能测试。 | 声明常量能正常解释，不报错 | 与预期结果一致 | - |
| 步骤 2 | 声明变量代码解释执行命令。 | 声明变量功能测试。 | 声明变量能正常解释，正常初始化，并能打印 | - |
| 步骤 3 | 声明字符串代码解释执行命令。 | 声明字符串类型功能测试。 | 能声明字符串类型，并正常解释，并能打印输出 | - |
| 步骤 4 | 声明函数代码解释执行命令。 | 声明函数类型功能测试。 | 能够声明函数，函数能正常调用 | - |
| 步骤 5 | 声明数组代码解释执行命令。 | 声明数组与初始化功能测试。 | 声明数组能正常解释，正常初始化，并能打印 | - |

1. JavaScript异常处理功能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JavaScrip异常处理功能测试/ GN\_YXHJC \_YYXS\_JS\_ YCCL8 | | | | |
| 用例说明 | | **测试目的：** v8解释器在对异常处理的能力。  **测试方法：** 测试v8解释器在对异常处理的能力,当程序产生越界,溢出等情况时,v8引擎本身不能崩溃,否则可能导致系统存在漏洞。  **合格判据：**测试过程中v8引擎不崩溃。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.使用交叉编译完成v8 v8\_fuzzer。  2.将编译好的环境部署到到目标机器。 | 随机输入数据。 | 对json库进行随机输入测试。 | 测试过程中v8引擎不崩溃。 | 与预期结果一致 | - |
| 步骤 2 | 随机输入数据。 | 对正则库进行随机输入测试。 | 测试过程中v8引擎不崩溃。 | 与预期结果一致 | - |
| 步骤 3 | 随机输入数据。 | 对多函数返回值进行随机返回测试。 | 测试过程中v8引擎不崩溃。 | 与预期结果一致 | - |

1. JavaScript性能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | JavaScrip性能测试/ GN\_YXHJC \_YYXS\_JS\_YZSR\_XN | | | | |
| 用例说明 | | **测试目的:**对v8引擎的性能进行验证。  **测试方法:** 通过Octane benchmark工具，分别运行在飞腾计算机和intel金牌5115计算机上，除以CPU峰值后，对比性能。  **合格判据:** 在飞腾计算机上测试结果除以CPU峰值后，与intel金牌5115计算机上测试结果除以CPU峰值的比，大于80% | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Chrome v8已成功安装在飞腾计算机上，版本为8.9.255.24。 | 无 | 在飞腾计算机上运行Octane，得到结果A1。 | 成功运行Octane，得到结果A1。 | 可得到结果A1 | - |
| 步骤 2 | SpecCPU已成功安装在飞腾计算机上。 | 无 | 在飞腾计算机上运行SpecCPU，得到结果B1。 | 成功运行SpecCPU，得到结果B1。 | 可得到结果B1 | - |
| 步骤 3 | Chrome v8已成功安装在intel金牌5115计算机上，版本为8.9.255.24。 | 无 | 在intel金牌5115计算机上运行Octane，得到结果A2。 | 成功运行Octane，得到结果A2。 | 可得到结果A2 | - |
| 步骤 4 | SpecCPU已成功安装在intel金牌5115计算机上。 | 无 | 在intel金牌5115计算机上运行SpecCPU，得到结果B2。 | 成功运行SpecCPU，得到结果B2。 | 可得到结果B2 | - |
| 步骤 5 | A1、B1、A2、B2均已测试出来。 | A1、B1、A2、B2 | 计算（A1/B1）/（A2/B2），得到性能比C。 | 性能比C〉=80%。 | 与预期结果一致 |  |

1. Python3.8语法运算测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Python3.8语法运算测试/ GN\_YXHJC\_YYXS\_ PYTH\_YFYS | | | | |
| 用例说明 | | **测试目的：** 可以实现python3.8语言适配，完成基本命令与逻辑运算。  **测试方法：** Python基本语法执行与运算：在python环境下，可以实现基本的输入输出等操作，以及基本的逻辑运算。  **合格判据：** Python软件可以实现基本命令与逻辑运算。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Python3.8版本工具安装并正常运行。 | 输入python3进入python3环境。 | 完成各种常用命令的使用。 | 常用命令能正常使用。 | 与预期结果一致 | - |
| 步骤 2 | 输入python3进入python3环境。 | 完成基本的逻辑运算。 | 完成逻辑运算。 | 与预期结果一致 | - |

1. Python3.8文件运行测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Python3.8文件运行测试/ GN\_YXHJC\_YYXS\_ PYTH\_WJYX | | | | |
| 用例说明 | | **测试目的：** 可以实现python3.8语言适配，实现python示例程序的运行。  **测试方法：** python示例程序：手动编写 .py文件，实现python文件的执行，并得到输出结果。  **合格判据：** Python示例程序或脚本可以实现对应功能。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Python3.8版本工具安装并正常运行。 | Python示例程序 | 用vim编写.py的示例程序。 | 可以实现文件的执行，及对应功能或输出相应结果。 | 与预期结果一致 | - |

1. Python3.8编译打包测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Python3.8编译打包测试/ GN\_YXHJC\_YYXS\_ PYTH\_BYDB | | | | |
| 用例说明 | | **测试目的：** 可以实现python3.8语言适配，可以成功将python脚本打包成可执行文件。  **测试方法：**实现将 .py文件打包成可以执行的可执行文件，并且能运行。  **合格判据：** Python示例程序或脚本可以实现对应功能。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | Python3.8版本工具安装并正常运行。 | Python示例程序。 | 编写python可打包的示例脚本。 | 打包脚本编写成功。 | 打包脚本编写成功 | - |
| 步骤 2 | 打包脚本编写成功。 | Python程序打包脚本。 | 使用PyInstaller等打包软件进行打包并执行。 | 打包成功，文件可执行。 | 与预期结果一致 | - |

## 配套工具/GN\_PTGJC

### 运行时SDK测试（贺荣、王峰、王强、刘嘉俊）

1. 运行时框架规范接口管理测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 运行时框架规范接口管理/GN\_PTGJC \_YXSSDK\_GFJK | | | | |
| 用例说明 | | **测试目的：** 提供应用框架、基础库的API接口，API接口的内容与国产芯片技术无关，支持跨国产平台的应用源码兼容。  **测试方法：** 通过查看头文件中的API接口确认与国产芯片技术无关。  **合格判据：** 头文件提供的API接口与国产芯片技术无关 。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已经安装QT4运行时框架镜像和沙箱工具。 | 沙箱工具命令。 | 使用沙箱工具进入QT4运行时框架交互式终端。 | 交互式运行成功。 | 交互式运行成功 | - |
| 步骤 2 | QT4运行时框架交互式运行成功。 | vim及QT4应用框架头文件。 | 使用vim查看QT4应用框架提供的头文件，检查API接口。 | API接口与国产芯片技术无关。 | 与结果一致 | - |
| 步骤 3 | QT4运行时框架交互式运行成功。 | vim及基础库头文件。 | 使用vim查看各类基础库提供的头文件， 检查API接口。 | API接口与国产芯片技术无关。 | 与结果一致 | - |
| 步骤 4 | 已经安装QT5运行时框架镜像和沙箱工具。 | 沙箱工具命令。 | 使用沙箱工具进入QT5运行时框架交互式终端。 | 交互式运行成功。 | 与预期结果一致 |  |
| 步骤 5 | QT5运行时框架交互式运行成功。 | vim及QT5应用框架头文件。 | 使用vim查看QT5应用框架提供的头文件，检查API接口。 | API接口与国产芯片技术无关。 | 与预期结果一致 |  |
| 步骤 6 | QT5运行时框架交互式运行成功。 | vim及基础库头文件。 | 使用vim查看各类基础库提供的头文件， 检查API接口。 | API接口与国产芯片技术无关。 | 与预期结果一致 |  |

1. QT运行时框架支撑测试用例（王峰）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT运行时框架支撑测试/GN\_PTGJC \_YXSSDK\_QTZC | | | | | |
| 用例说明 | | **测试目的：** 支持常见QT界面应用程序的运行。  **测试方法：** 使用沙箱工具运行QT4及QT5界面应用程序；  **合格判据：** QT4及QT5界面应用程序能够正常运行，能够正常输入及显示中文。 | | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | | 备注 |
| 步骤 1 | 已安装QT4界面应用及沙箱工具。 | 沙箱工具命令。 | 使用沙箱工具启动QT4界面应用程序。 | 应用界面图片、文字等能够正常显示。 | 应用界面信息显示正常 | - | |
| 步骤 2 | 中文字符串。 | 点击文本框，切换中文输入法，输入中文信息。 | 能够输入中文，并显示正常。 | 与预期结果一致 | - | |
| 步骤 3 | 已安装QT5界面应用及沙箱工具。 | 沙箱工具命令。 | 使用沙箱工具启动QT5界面应用程序。 | 应用界面图片、文字等能够正常显示。 | 与预期结果一致 | - | |
| 步骤 4 | 中文字符串。 | 点击文本框，切换中文输入法，输入中文信息。 | 能够输入中文，并显示正常。 | 与预期结果一致 | - | |

1. Javascript运行时框架支撑测试用例（贺荣）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Javascript运行时框架支撑 / GN\_ YXHJ \_YYSX\_JKJ | | | | |
| 用例说明 | | **测试目的：** 测试软件是否支持Javascript运行时框架支撑。  **测试方法：** 打开集成了Javascript基础库的应用，应用能正常使用并展示应用内容。  **合格判据：** 集成了Javascript基础库的应用可以正常展示与使用。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户已成功安装“日志处理”应用和本地应用 | 用户名、密码等登录信息 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 与预期结果一致 | - |
| 步骤 2 | 在已安装的应用列表，可以看到“日志处理”应用和本地应用 | 无 | “日志处理”应用和本地应用 | 成功打开“日志处理”应用和本地应用，并展示内容。 | 与结果一致 | - |

1. C++运行时框架支撑测试用例（刘嘉俊）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | C++运行时框架支撑/ GN\_ PTGJC \_YYSXSDK\_CKJ | | | | |
| 用例说明 | | **测试目的：** 验证支持C++服务端应用程序的运行完整性和健壮性。  **测试方法：**   1. 模块集成：C++服务端应用框架能发现模块目录下的指定格式的应用模块插件，并将其加载至C++服务端应用框架中。 2. 服务注册与发现：应用模块插件可向C++服务端应用框架发现和调用所需要的服务。   **合格判据：** C++服务端应用框架支持服务端应用成功启动。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | C++服务端应用框架，服务端应用已经部署到环境中。 | 软件仓库的地址。 | 通过应用沙箱命令（sandboxCli）拉取C++服务端应用框架镜像，并启动C++服务端应用框架镜像。 | 成功拉取C++服务端应用框架镜像到本地。 | 与预期结果一致 | - |
| 步骤 2 | 成功拉取C++服务端应用框架镜像到本地； | 本地应用框架指定目录 | 通过tty模式进入C++服务端应用框架镜像，将应用模块插件目录拷贝至C++服务端应用框架指定目录。 | 在应用框架目录下成功生成启动应用相关目录。 | 与预期结果一致 | - |
| 步骤 3 | 在应用框架目录下成功生成启动应用相关目录。 | 运行框架支撑测试程序。 | 1.测试程序成功初始化C++服务端应用框架。  2.测试程序获取C++服务端应用框架插件接口。  3.测试程序调用C++服务端应用框架插件接口。 | 启动测试程序，测试程序成功运行，成功调用插件接口。 | 与预期结果一致 | - |
| 步骤 4 | 测试程序成功运行。 | 1.运行框架支撑测试程序。 | 1.测试程序释放C++服务端应用框架中的相关功能。  2.测试程序成功退出。 | 测试程序成功退出，并且正常释放C++服务端应用框架中相关功能。 | 与预期结果一致 | - |

1. Java运行时框架支撑测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Java运行时框架支撑/ GN\_PTGJC\_YYSXSDK\_JAKJ | | | | |
| 用例说明 | | **测试目的：** 集成和封装Java服务端应用框架、Java基础库、系统库和系统工具，支持Java服务端应用程序的运行。  **测试方法：** 运行时框架封装Java服务端应用框架；运行时框架封装Java基础库；运行时框架封装操作系统系统库和系统工具；运行时框架支持Java服务端应用程序的运行。  **合格判据：** 同一个应用的不同版本使用各自的库加载运行；同一个应用的多个版本可以同机同时运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.安装好沙箱工具启动相关服务。  2.准备好运行时框架镜像。 | 沙箱启动命令。 | 通过沙箱进入Java服务端运行时框架，找到对应的基础运行环境，查看其所提供的系统库和系统工具，确保运行框架所包含的基础运行环境的存在。 | 能够进入运行时框架环境，查看相关的基础运行库，系统库和相关的几个系统命令(如：pwd, ping)。 | 与预期结果一致 | - |
| 步骤 2 | 运行框架准备完善。 | 沙箱运行命令。 | 通过沙箱进入Java服务端运行时框架，在JDK环境中查看相应的工具和基础库，如java/javac和tools.jar和jre/lib/rt.jar。 | 查看JDK的版本和相关的库包 | 与预期结果一致 | - |
| 步骤 3 | Java运行时框架依赖的maven仓库settings.xml文件已经存在。 | 沙箱运行命令。 | 通过沙箱进入Java服务端运行时框架，查看为支撑应用提供的Java依赖库,maven仓库的库文件和相关的设置文件settings.xml。 | 相关的依赖库和maven配置信息已经在运行时框架中存在。 | 与结果一致 | - |
| 步骤 4 | Java运行时框架依赖的Springboot环境已经准备好。 | 沙箱运行命令。 | 通过沙箱进入Java服务端运行时框架，查看为支撑应用提供的SpringBoot服务端应用框架依赖库。 | 可以查看到spring相关的库信息。 | 与预期结果一致 | - |
| 步骤 5 | Java运行时框架依赖的Springboot测试应用已经准备好。 | 沙箱运行命令、Java应用程序。 | 基于Java服务端运行时框架对基于Springboot应用框架构建的应用进行构建和编译打包，基于运行时框架运行应用，检测查看应用运行结果，证实应用可以基于运行时框架运行。 | 应用可以基于运行时框架运行。 | 应用基于运行时框架运行 | - |

### 封装验证系统测试（刘嘉俊、王强、贺荣）

1. QT运行时框架封装打包管理测试用例（刘嘉俊）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT运行时框架封装打包管理/GN\_PTGJ C\_FZYZ\_FZDB | | | | |
| 用例说明 | | **测试目的：** 支持基于基础库进行QT运行时框架的按需构建与封装。  **测试方法：** 基于老版本的QT运行时框架以及需要新增的基础库，构建新版本QT运行时框架。  **合格判据：** 能够基于基础库构建新的QT运行时框架。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已安装QT运行时框架。 | docker命令。 | 使用docker工具运行老版本QT运行时框架。 | 成功进入交互式终端。 | 与结果一致 | - |
| 步骤 2 | 成功进入交互式终端。 | 软件仓库配置信息。 | 修改软件仓库配置信息，更新软件包信息。 | 成功更新软件仓库配置信息。 | 与结果一致 | - |
| 步骤 3 | 成功更新软件仓库配置信息。 | 软件包A。 | 使用软件包管理工具安装远程软件包A。 | 成功安装远程软件包A。 | 与结果一致 | - |
| 步骤 4 | 成功更新软件仓库配置信息。 | 软件包B。 | 使用软件包管理工具安装本地软件包B | 成功安装本地软件包B。 | 与预期结果一致 | - |
| 步骤 5 | 成功安装远程软件包A和B。 | 软件包管理工具命令。 | 使用软件包管理工具查看已经安装的软件包信息。 | 确认软件包A、B已经成功安装。 | 与预期结果一致 | - |
| 步骤 6 | 确认成功安装远程软件包A和B。 | docker命令。 | 使用docker工具构建QT运行时框架。 | 成功构建新版本QT运行时框架。 | 与结果一致 | - |
| 步骤 7 | 成功构建新版本QT运行时框架。 | docker命令。 | 使用docker工具将新的QT运行时框架上传到镜像仓库。 | 成功上传到镜像仓库。 | 上传到镜像仓库 | - |

1. QT运行时框架功能验证管理测试用例（刘嘉俊）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT运行时框架功能验证管理/GN\_PTGJC \_FZYZ\_GNYZ | | | | |
| 用例说明 | | **测试目的：** 测试封装验证系统基于QT运行时框架的应用验证管理功能。  **测试方法：** 通过测试应用能否基于QT4及QT5运行时框架正常运行，验证QT运行时框架的封装是否正常。  **合格判据：** 构建出的QT4和QT5应用镜像能够在国产平台验证机上正常安装及运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 通过浏览器进行登录访问应用远程构建系统。 | 应用信息。 | 登录应用远程构建系统，填写及设置待构建的应用程序名称、版本、构建后应用镜像名称、版本、应用依赖的运行时框架等信息。 | 成功登录并设置应用信息。 | 成功登录并设置 | - |
| 步骤 2 | QT4应用已构建完成。 | 飞腾版QT4应用镜像、QT4运行时框架镜像。 | 选择飞腾架构，点击远程验证按钮，进入远程验证的终端显示界面，在该终端上使用沙箱工具安装QT4运行时框架镜像，安装QT4应用镜像，运行QT4应用。 | QT4运行时框架及应用能够正常安装及运行。 | 与结果一致 | - |
| 步骤 3 | QT5应用已构建完成。 | 飞腾版QT5应用镜像、QT5运行时框架镜像。 | 在该终端上使用沙箱工具安装QT5运行时框架镜像，安装QT5应用镜像，运行QT5应用。 | QT5运行时框架及应用能够正常安装及运行。 | 与预期结果一致 | - |
| 步骤 4 | QT4应用已构建完成。 | 龙芯版QT4应用镜像、QT4运行时框架镜像。 | 选择龙芯架构，点击远程验证按钮，进入远程验证的终端显示界面，在该终端上使用沙箱工具安装QT4运行时框架镜像，安装QT4应用镜像，运行QT4应用。 | QT4运行时框架及应用能够正常安装及运行。 | 与结果一致 |  |
| 步骤 5 | QT5应用已构建完成。 | 龙芯版QT5应用镜像、QT5运行时框架镜像。 | QT5运行时框架及应用能够正常安装及运行在该终端上使用沙箱工具安装QT5运行时框架镜像，安装QT5应用镜像，运行QT5应用。 | QT5运行时框架及应用能够正常安装及运行。 | QT5正常安装及运行 | - |

1. QT运行时框架插件测试用例（刘嘉俊）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | QT运行时框架构建打包插件/GN\_PTGJ \_FZYZ\_GJCJ | | | | |
| 用例说明 | | 测试目的： QT运行时框架构建打包插件支持基于基础库进行QT运行时框架应用的开发、构建与封装。  测试方法： 基于QT运行时框架构建打包插件开发的心qtc应用可本地构建，远程构建，git构建相关应用镜像。  合格判据： 构建应用镜像成功。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已安装QT运行时框架。 | docker命令。 | 使用docker工具运行QT运行时框架。 | 成功进入交互式终端。 | 与结果一致 | - |
| 步骤 2 | 成功进入交互式终端。 | Qtcreator启动命令 | 启动qtcreator。 | Qtcreator启动成功。 | 与结果一致 | - |
| 步骤 3 | Qtcreator启动成功 | 新建qtcreator插件工程操作 | 创建qtcreator插件工程。 | 创建qtcreator插件工程成功。 | 与结果一致 | - |
| 步骤 4 | 创建qtcreator插件工程成功。 | 打开基础库操作界面命令 | 对qtcreator插件工程所依赖的基础库进行管理 | 对qtcreator插件工程所依赖的基础库管理成功。 | 与预期结果一致 | - |
| 步骤 5 | 创建qtcreator插件工程成功。 | 打开远程构建界面。 | 修改打包构建相关脚本，点击本地打包按钮。 | 本地打包成功。 | 与预期结果一致 | - |
| 步骤 6 | 创建qtcreator插件工程成功。 | 打开远程构建界面。 | 用户填写远程构建所需要的相关信息，点击远程构建按钮，成功构建出应用镜像包 | 远程构建成功。 | 与结果一致 | - |
| 步骤 7 | 创建qtcreator插件工程成功。 | 打开远程构建界面。 | 用户填写git构建所需要的相关信息，点击git构建按钮，成功构建出应用镜像包； | Git构建成功。 | 与结果一致 | - |

1. C++运行时框架封装打包管理测试用例（刘嘉俊）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | C++运行时框架封装打包管理/GN\_PTGJC \_FZYZ\_FZDB | | | | |
| 用例说明 | | **测试目的：**验证C++运行时框架封装打包管理的完整性和正确性。  **测试方法：**基于老版本的C++运行时框架以及需要新增的基础插件A和B，构建新版本C++运行时框架。  **合格判据：** 新版本C++运行时框架按需构建成功。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 软件仓库已就绪，C++运行时框架镜像已生成。 | 运行时框架软件仓库地址。 | 通过应用沙箱命令拉取和启动C++服务端运行时框架镜像。 | 成功拉取C++服务端应用框架镜像到本地。 | 与结果一致 | - |
| 步骤 2 | 成功拉取C++服务端应用框架镜像到本地。 | 基础插件目录A。 | 通过tty模式进入C++服务端应用框架镜像，将一个基础插件目录A拷贝至C++服务端应用框架指定目录下。 | 将一个基础插件目录A拷贝至C++服务端应用框架指定目录下。 | 与预期结果一致 | - |
| 步骤 3 | 成功拉取C++服务端应用框架镜像到本地。 | 基础插件目录B。 | 通过tty模式进入C++服务端应用框架镜像，将一个基础插件目录B拷贝至C++服务端应用框架指定目录下。 | 将一个基础插件目录B拷贝至C++服务端应用框架指定目录下。 | 与预期结果一致 | - |
| 步骤 4 | 基础插件目录A和B都已拷贝到应用框架指定目录下。 | 应用框架插件配置文件。 | 修改应用框架插件配置文件，包含刚刚拷贝过来的A和B插件。 | 应用框架插件配置文件被成功修改。 | 与结果一致 |  |
| 步骤 5 | 应用框架插件配置文件被成功修改。 | 运行时框架软件仓库地址。 | 将新版本C++运行时框架上传到软件仓库。 | 显示上传成功。 | 上传成功 |  |

1. C++运行时框架功能验证管理测试用例（刘嘉俊）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | C++运行时框架功能验证管理/GN\_PTGJC \_FZYZ\_GNYZ | | | | |
| 用例说明 | | **测试目的：**支持对C++运行时框架的功能进行验证。  **测试方法：**通过测试应用能否基于C++运行时框架正常运行，验证C++运行时框架的封装是否正常。  **合格判据：** C++运行时框架功能全部验证通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 通过浏览器进行登录访问应用远程构建系统。 | 通过浏览器进行登录访问应用远程构建系统。 | 登录应用远程构建系统，填写及设置待构建的应用程序名称、版本、构建后应用镜像名称、版本、应用依赖的运行时框架等信息。 | 成功登录并设置应用信息。 | 成功登录并设置应用 |  |
| 步骤 2 | C++应用已构建完成。 | 飞腾版C++应用镜像、C++运行时框架镜像。 | 选择飞腾架构，点击远程验证按钮，进入远程验证的终端显示界面，在该终端上使用沙箱工具安装C++运行时框架镜像，安装C++应用镜像，运行C++应用。 | C++运行时框架及应用能够正常安装及运行。 | 与预期结果一致 | - |
| 步骤 3 | C++应用已构建完成。 | 龙芯版C++应用镜像、C++运行时框架镜像。 | 选择龙芯架构，点击远程验证按钮，进入远程验证的终端显示界面，在该终端上使用沙箱工具安装C++运行时框架镜像，安装C++应用镜像，运行C++应用。 | C++运行时框架及应用能够正常安装及运行。 | 与预期结果一致 |  |

1. C++运行时框架插件测试用例（刘嘉俊）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | C++运行时框架构建打包插件/GN\_PTGJ \_FZYZ\_GJCJ | | | | |
| 用例说明 | | 测试目的： QT运行时框架构建打包插件支持基于基础库进行C++运行时框架应用开发、构建与封装。  测试方法： 基于C++运行时框架构建打包插件开发的心qtc应用可本地构建，远程构建，git构建相关应用镜像。  合格判据： 构建应用镜像成功。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已安装C++运行时框架。 | docker命令。 | 使用docker工具运行C++运行时框架。 | 成功进入交互式终端。 | 与结果一致 | - |
| 步骤 2 | 成功进入交互式终端。 | Qtcreator启动命令 | 启动qtcreator。 | Qtcreator启动成功。 | 与结果一致 | - |
| 步骤 3 | Qtcreator启动成功 | 新建qtcreator插件工程操作 | 创建qtcreator插件工程。 | 创建qtcreator插件工程成功。 | 与结果一致 | - |
| 步骤 4 | 创建qtcreator插件工程成功。 | 打开基础库操作界面命令 | 对qtcreator插件工程所依赖的基础库进行管理 | 对qtcreator插件工程所依赖的基础库管理成功。 | 与预期结果一致 | - |
| 步骤 5 | 创建qtcreator插件工程成功。 | 打开远程构建界面。 | 修改打包构建相关脚本，点击本地打包按钮。 | 本地打包成功。 | 与预期结果一致 | - |
| 步骤 6 | 创建qtcreator插件工程成功。 | 打开远程构建界面。 | 用户填写远程构建所需要的相关信息，点击远程构建按钮，成功构建出应用镜像包 | 远程构建成功。 | 与结果一致 | - |
| 步骤 7 | 创建qtcreator插件工程成功。 | 打开远程构建界面。 | 用户填写git构建所需要的相关信息，点击git构建按钮，成功构建出应用镜像包； | Git构建成功。 | 与结果一致 | - |

1. Java运行时框架封装打包管理测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Java运行时框架封装打包管理测试/GN\_YXHJ \_YYSX\_JAKJ | | | | |
| 用例说明 | | **测试目的：** 支持基于基础库进行Java运行时框架的按需构建与封装。  **测试方法：** 基于Java运行时框架进行构建和封装；在运行时框架中添加或者减少Maven依赖库（基础库）。  **合格判据：** Java运行时框架能够提供应用所需要的基础库和编译工具；按需要增加基础库。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 根据相应的测试平台配置好maven的设置和网络环境。 | mvn命令。 | 在pom.xml文件中加入要下载的基础库的名称和版本，执行mvn下载命令。 | 需要的基础库下载到本地指定的maven仓库中。 | 与预期结果一致 | - |
| 步骤 2 | 基础库下载到本地maven仓库。 | 相关的制作运行时框架的目录。 | 将基础库挂载到要制作运行时框架的目录。 | 相应的基础库在指定目录中。 | 基础库在指定的目录 | - |
| 步骤 3 | 沙箱环境和命令已提前准备。 | 运行沙箱启动命令。 | 进入新的运行时框架构成的沙箱环境中，查看相应的库是否已经存在。 | 相应的库已经在沙箱中存在。 | 与预期结果一致 | - |
| 步骤 4 | 准备验证pom.xml脚本。 | 执行mvn命令。 | 挂载验证脚本，执行库依赖命令mvn dependency：tree。 | 显示所有的依赖包都能够找到。 | 显示所有依赖包 | - |

1. Java运行时框架功能验证管理测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Java运行时框架功能验证管理测试/GN\_PTGJ \_FZYZ\_GNYZ | | | | |
| 用例说明 | | **测试目的：** 支持对Java运行时框架的功能进行验证。  **测试方法：** 基于已经打包的应用对Java运行时框架的功能进行验证；基于远程验证界面工具进行多平台的验证。  **合格判据：** Java运行时框架能够提供应用所需要的应用成功运行的依赖环境；应用能够通过沙箱和运行时框架的结合进行正确的运行，通过功能运行检测。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 在飞腾平台验证机器上提前安装应用所依赖的运行时框架。 | 输入沙箱命令安装应用。 | 在飞腾平台验证机器上安装应用的包或者远程镜像，检测应用安装是否正常。 | 应用在指定的飞腾机器平台上安装成功。 | 与预期结果一致 | - |
| 步骤 2 | 应用在指定的飞腾机器平台上安装成功。 | 输入沙箱命令启动应用。 | 调整应用的配置信息，运行应用，验证应用的功能是否正常启用，完成应用的飞腾平台验证。 | 应用在指定的飞腾机器平台上运行成功。 | 应用在飞腾机器平台上运行成功 | - |
| 步骤 3 | 在龙芯平台验证机器上提前安装应用所依赖的运行时框架。 | 输入沙箱命令安装应用。 | 在龙芯平台验证机器上安装应用的包或者远程镜像，检测应用安装是否正常。 | 应用在指定的龙芯机器平台上安装成功。 | 与预期结果一致 |  |
| 步骤 4 | 应用在指定的龙芯机器平台上安装成功。 | 输入沙箱命令启动应用。 | 调整应用的配置信息，运行应用，验证应用的功能是否正常启用，完成应用的龙芯平台验证。 | 应用在指定的龙芯机器平台上运行成功。 | 与预期结果一致 | - |
| 步骤 5 | 在申威平台验证机器上提前安装应用所依赖的运行时框架。 | 输入沙箱命令安装应用。 | 在龙芯平台验证机器上安装应用的包或者远程镜像，检测应用安装是否正常。 | 应用在指定的申威机器平台上安装成功。 | 与预期结果一致 |  |
| 步骤 6 | 应用在指定的申威机器平台上安装成功。 | 输入沙箱命令启动应用。 | 调整应用的配置信息，运行应用，验证应用的功能是否正常启用，完成应用的龙芯平台验证。 | 应用在指定的申威机器平台上运行成功。 | 与预期结果一致 |  |

1. Javascript运行时框架封装打包管理测试用例（贺荣）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Javascript运行时框架封装打包管理/GN\_PTGJ \_FZYZ\_FZDB | | | | |
| 用例说明 | | **测试目的：** 测试软件是否支持Javascript运行时框架封装打包管理。  **测试方法：** 分别使用构建封装和打包命令，对Javascript运行时框架工程进行封装打包，并成功部署运行。  **合格判据：** 支持Javascript运行时框架的封装打包管理。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 提供Javascript客户端应用框架的可执行文件已提供、运行时框架镜像的构建脚本 | 执行镜像构建命令 | 执行镜像构建命令(docker)，使用新的可执行文件构建成运行时框架 | 能够构造成指定的运行时框架 | 与预期结果一致 | - |
| 步骤 2 | 沙箱环境和命令已提前准备 | 运行沙箱启动命令 | 进入新的运行时框架构成的沙箱环境中，查看相应的可执行文件是否已经存在 | 相应的可执行文件已经在沙箱中存在 | 可执行文件已经在沙箱中存在 | - |

1. Javascript运行时框架功能验证管理测试用例（贺荣）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | Javascript运行时框架功能验证管理/GN\_PTGJ \_FZYZ\_GNYZ | | | | |
| 用例说明 | | **测试目的：** 测试软件是否支持Javascript运行时框架功能验证管理。  **测试方法：** 登录Javascript运行时框架，可以正常实现浏览展示，个性化设置等框架的功能，可以访问应用的页面。  **合格判据：**功能正常实现， 支持Javascript运行时框架功能验证管理。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | JavaScript客户端应用框架正常运行，登录用户成功安装至少一个应用。 | 用户名、密码等登录信息 | 用户访问JavaScript客户端应用框架，查看已安装的应用列表。 | 成功访问JavaScript客户端应用框架，并成功查看已安装的应用列表。 | 与预期结果一致 | - |
| 步骤 2 | 成功登录JavaScript客户端应用框架并可以查看应用列表 | 无 | 查看框架展示的信息。 | Javascript运行时框架可以正常实现浏览展示，个性化设置等框架的功能。 | 与结果一致 | - |
| 步骤 3 | 成功登录JavaScript客户端应用框架并可以查看应用列表 | 无 | 点击已安装的应用。 | 可以成功访问应用页面。 | 访问应用页面 |  |

### 软件仓库系统测试（王峰、王强）

1. 操作系统镜像管理测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 操作系统镜像管理测试/GN\_PTGJ \_YJCK\_XTJX | | | | |
| 用例说明 | | **测试目的：**支持对操作系统镜像进行管理。  **测试方法：**在仓库管理界面中对操作系统镜像进行管理操作；客户端的用户可以访问和下载所需要的操作系统镜像。  **合格判据：**能够在仓库管理界面中对操作系统镜像进行管理操作；能够使客户端的用户访问和下载所需要的操作系统镜像。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 输入用户名和密码登录到仓库管理界面。 | 导航到操作系统库界面。 | 按国产平台名称分类不同的操作系统镜像，提供基于Web的检索界面。 | 能够查看不同平台的操作系统镜像信息。 | 查看平台操作系统镜像信息 | - |
| 步骤 2 | 能够查看到不同平台的操作系统镜像信息。 | 本地操作系统镜像。 | 在管理界面上传操作系统镜像，测试对新镜像进行的存储操作。 | 在软件仓库中上传操作系统镜像成功。 | 与预期结果一致 | - |
| 步骤 3 | 能够查看到不同平台的操作系统镜像信息。 | 待删除的镜像名称。 | 在管理界面删除操作系统镜像，测试旧镜像的删除操作。 | 在软件仓库中删除操作系统镜像成功。 | 与结果一致 | - |
| 步骤 4 | 准备好需要下载操作系统的国产机器。 | 下载镜像，并进行安装。 | 在测试机器上能够下载操作系统镜像。 | 镜像能够下载成功。 | 镜像下载成功 | - |
| 步骤 5 | 已经创建好用户并配置好权限。 | 用户进行访问仓库操作系统镜像操作。 | 测试仓库权限管理，新增和删除操作系统镜像需要进行权限控制。 | 验证只有登录后的授权用户才可以上传或修改镜像。 | 与预期结果一致 | - |

1. 基础库管理测试用例（王峰）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 基础库管理/GN\_PTGJC \_RJCK\_JCK | | | | |
| 用例说明 | | **测试目的：**支持对基础库的管理。  **测试方法：**在仓库管理界面中对基础库进行管理操作；客户端的用户可以访问和下载所需要的基础库。  **合格判据：**能够在仓库管理界面中对基础库进行管理操作；能够使客户端的用户访问和下载所需要的基础库。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 登录仓库管理界面。 | 导航到软件仓库界面。 | 按国产平台名称进行基础库分类，按照类别提供基于Web界面的访问功能。 | 能够查看到不同平台的基础库信息。 | 与预期结果一致 | - |
| 步骤 2 | 准备好需要安装基础库的国产机器和操作系统。 | 使用apt系统命令下载基础库，并进行安装。 | 在测试机器上配置软件仓库作为软件源，更新软件源信息，能够下载基础库，可以进行安装。 | 基础库能够下载和安装成功。 | 与预期结果一致 | - |

1. 安装包集中管理测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 安装包集中管理测试/GN\_PTGJ \_RJCK\_AZB | | | | |
| 用例说明 | | **测试目的：** 支持对运行时框架或应用的安装包进行集中管理。  **测试方法：** 在仓库管理界面中对安装包进行管理操作；客户端的用户可以访问和下载所需要的安装包。  **合格判据：** 能够在仓库管理界面中对安装包进行管理操作；能够使用沙箱工具命令对安装包进行下载和上传操作；能够使客户端的用户访问和下载所需要的安装包。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 输入用户名和密码登录到仓库管理界面。 | 导航到软件仓库运行时框架安装包界面。 | 按国产平台名称以及运行时框架类型，进行安装包分类，提供基于Web的检索界面。 | 能够查看到不同平台和运行时框架类型的安装包信息。 | 与预期结果一致 | - |
| 步骤 2 | 能够查看到运行时框架安装包信息。 | 运行时框架安装包。 | 在管理界面进行上传安装包操作，测试对新安装包进行的存储操作。 | 在软件仓库中上传运行时框架安装包成功。 | 与预期结果一致 | - |
| 步骤 3 | 能够查看到运行时框架安装包信息。 | 删除指定的运行时框架安装包。 | 在管理界面进行进行删除运行时框架安装包，测试对旧安装包的删除操作。 | 在软件仓库中删除运行时框架安装包成功。 | 与预期结果一致 | - |
| 步骤 4 | 准备好需要安安装包的国产机器和操作系统，及沙箱工具环境。 | 使用沙箱命令下载安装包，并进行安装。 | 在测试机器上能够下载安装包，它可以支持在相应的操作系统上安装或者已安装包的方式传递使用，操作过程中测试使用沙箱命令下载安装包。 | 运行时框架安装包能够下载和安装成功。 | 与预期结果一致 | - |

1. 安装包版本管理测试用例（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 安装包版本管理测试/GN\_PTGJ \_RJCK\_AZBB | | | | |
| 用例说明 | | **测试目的：** 支持对运行时框架和应用的安装包进行版本管理。  **测试方法：** 在仓库管理界面中对安装包的不同版本进行管理操作；客户端的用户可以访问和下载所需要的所需要版本的安装包。  **合格判据：** 能够在仓库管理界面中对多版本安装包进行管理操作；能够使用沙箱工具命令对指定版本安装包进行下载和上传操作；能够使客户端的用户访问和下载所需要的指定版本的安装包。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 输入用户名和密码登录到仓库管理界面。 | 导航到软件仓库运行时框架安装包界面。 | 按国产平台名称以及运行时框架类型，进行安装包分类，提供基于Web的检索界面。 | 能够查看到不同平台和运行时框架类型的安装包信息。 | 与预期结果一致 | - |
| 步骤 2 | 在本机上准备好沙箱命令，并能够连成功登录远程仓库。 | 输入应用包上传的命令。 | 测试使用沙箱命令上传运行时框架的不同版本安装包，通过指定tag确定版本信息。 | 运行时框架能成功上传到远程应用仓库并通过tag指定不同的版本。 | 与预期结果一致 | - |
| 步骤 3 | 运行时框架不同版本能成功上传到远程应用仓库。 | 删除指定的安装包 | 在管理界面进行进行删除指定版本的安装包，测试对旧版本安装包的删除操作。 | 在软件仓库中删除指定版本的安装包成功。 | 与预期结果一致 | - |
| 步骤 4 | 准备好需要安安装包的国产机器和操作系统，及沙箱工具环境。 | 使用沙箱命令下载运行时框架安装包，并进行安装。 | 在测试机器上能够下载指定版本的运行时框架安装包，它可以支持在相应的操作系统上安装或者已安装包的方式传递使用，操作过程中测试使用沙箱命令下载安装包。 | 对指定版本(tag)的安装包能够下载和安装成功。 | 与预期结果一致 | - |

### 运行管理工具测试（王强、王乐）

王乐还缺三个用例及测试程序

1. 支持对运行时框架和基础库的发布共享与技术交流（王强）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 运行时框架和库的交流共享/GN\_ PTGJC \_YXGJ \_JLGX | | | | |
| 用例说明 | | **测试目的：** 提供基于web或其他形式交流共享界面，使不同的人员能够发布框架和库的意见建议，并形成讨论。  **测试方法：** 测试是否具备相关的技术交流和共享的界面，能够区分不同的用户，发送不同题目的问题，并由不同的人进行讨论回复；  **合格判据：** 测试是否具备相关的技术交流和共享的能力。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.远程构建服务器的界面网站正确启动  2. 相关的登录账户已经创建 | 输入感兴趣的问题 | 输入感兴趣的问题题目，并附上详细的描述信息 | 问题能够成功发布 | 与结果一致 |  |
| 步骤 2 | 1.远程构建服务器的界面网站正确启动  2. 相关的登录账户已经创建 | 无 | 查找发布的问题，并浏览详细的信息，并进行变更、删除等操作 | 能够看到相关的发布者的问题 | 与结果一致 | - |
| 步骤 3 | 1.远程构建服务器的界面网站正确启动  2. 相关的登录账户已经创建 | 无 | 以不同的用户身份登录到网站，对别人的问题进行交流回复，并做出评价 | 能够成功回复问题，对自己的回复能够进行变更处理 | 与预期结果一致 | - |

1. 支持对提交共享的基础库进行测试认定

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 运行时框架和库的交流共享/GN\_ PTGJC \_YXGJ \_JLGX | | | | |
| 用例说明 | | **测试目的：** 支持对基础库进行测试认定，认定通过的库可以进入库清单中。  **测试方法：** 1.检查飞腾、龙芯、申威平台下的接口是否统一；2.在飞腾、龙芯、申威平台上对库进行编译，编译通过；  **合格判据：** 库在三个平台上接口统一，且能够在三个平台上编译通过。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.已经提供待检测库在三个平台上的源码 | 无 | 检查三个平台的接口 | 接口一致 | 与结果一致 | - |
| 步骤 2 | 1.已完成步骤1 | 无 | 在飞腾平台上进行编译 | 编译通过 | 与结果一致 | - |
| 步骤 3 | 1.已完成步骤1 | 无 | 在龙芯平台上进行编译 | 编译通过 | 与结果一致 | - |
| 步骤 4 | 1.已完成步骤1 | 无 | 在申威平台上进行编译 | 编译通过 | 与结果一致 | - |

1. 安装、升级、卸载管理测试用例（王乐）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 安装、升级、卸载管理/GN\_ PTGJC \_YXGJ \_ASXGL | | | | |
| 用例说明 | | **测试目的：** 测试配套工具支持对应用和运行时框架进行安装、升级和卸载。  **测试方法：** 通过基于load、rmi等命令分别对运行时框架的新老版本镜像进行安装、升级与卸载，并查看本地软件安装的情况确认是否成功；  **合格判据：** 成功支持应用和运行时框架的安装、升级和卸载。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.已部署好应用沙箱且正常启动。  2.已准备好运行时框架镜像和应用老版本镜像。 | load命令、运行时框架和应用老版本镜像。 | 1.执行沙箱客户端load命令加载运行时框架老版本镜像。  2.执行沙箱客户端load命令加载应用老版本镜像。 | 1.运行时框架镜像安装成功。  2.应用镜像安装成功。  3.使用images命令可以查看到已安装好的运行时框架镜像和应用老版本镜像。 | 与结果一致 |  |
| 步骤 2 | 1.已安装好运行时框架镜像和应用老版本镜像  3.已准备好运行时框架镜像和应用镜像的升级镜像。 | load命令、运行时框架和应用升级镜像。 | 1.执行沙箱客户端load命令加载运行时框架升级镜像  2.执行沙箱客户端load命令加载应用升级镜像。 | 1.运行时框架升级镜像安装成功。  2.应用升级镜像安装成功。  3.使用images命令可以查看到已安装好的运行时框架升级镜像和应用升级镜像。 | 与结果一致 | - |
| 步骤 3 | 1.已部署好应用沙箱且正常启动。  2.已安装好运行时框架镜像和应用镜像及升级镜像。 | rmi命令、运行时框架和应用升级镜像 | 1.执行沙箱客户端rmi命令删除应用镜像。  2.执行沙箱客户端rmi命令删除运行时框架镜像。  3.执行沙箱客户端rmi命令删除应用升级镜像   1. 执行沙箱客户端rmi命令删除运行时框架升级镜像。 | 1.应用镜像删除成功   1. 运行时框架镜像删除成功。   2.应用升级镜像删除成功。  3.运行时框架升级镜像删除成功。 | 与预期结果一致 | - |

1. 安装信息管理测试用例（王乐）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 安装信息管理/GN\_PTGJC\_ YXGJ \_AZXX | | | | |
| 用例说明 | | **测试目的：** 支持对安装信息进行管理。  **测试方法：** 基于load安装好运行时框架以及应用镜像后，基于images命令查看已经安装好的镜像信息。  **合格判据：** 能正常显示镜像的安装类型、安装时间等 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已部署好应用沙箱且正常启动，已准备好运行时框架镜像和应用镜像。 | load命令，运行时框架基础镜像和应用镜像。 | 执行沙箱客户端load命令加载运行时框架基础镜像和应用镜像。 | 安装镜像成功，应用镜像可以成功运行。 | 安装镜像成功，并运行 |  |
| 步骤 2 | images命令。 | 执行沙箱客户端images命令显示基础镜像和应用镜像的安装类型和安装时间。 | 显示安装类型和安装时间成功。 | 与预期结果一致 |  |

1. 应用基于运行时框架的启动和运行管理测试用例（王乐）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 应用基于运行时框架的启动和运行管理/GN\_PTGJC\_ YXGJ \_QDYX | | | | |
| 用例说明 | | **测试目的：** 应用支持基于运行时框架的启动和运行。  **测试方法：**基于load安装好运行时框架以及应用镜像后，基于create命令创建沙箱实例，基于start命令启动时沙箱，基于stop命令停止沙箱实例。  **合格判据：** 应用能支持基于运行时框架的启动和运行。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 已安装好运行时框架镜像与 应用镜像。 | load命令。 | 调用沙箱客户端的load命令加载基于运行时框架的应用镜像。 | 1.安装镜像成功  2.使用images命令能查看到安装成功的应用镜像。 | 与预期结果一致 |  |
| 步骤 2 | 已安装好运行时框架镜像和应用镜像。 | create命令。 | 执行沙箱客户端的create命令创建沙箱实例。 | 1.沙箱实例创建成功。  2.使用ps命令可以查看到新创建的沙箱实例。 | 与预期结果一致 |  |
| 步骤 3 | 已创建好沙箱实例。 | start命令、stop命令。 | 1.执行沙箱客户端的start命令启动时沙箱。  2.执行沙箱客户端的stop命令停止沙箱实例。  3.重复上述操作。 | 1. 沙箱实例正常启动。 2. 沙箱实例正常停止。 | 与预期结果一致 |  |

1. 多个运行时框架同机运行管理测试用例（王乐）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 多个运行时框架同机运行管理/G N\_PTGJC\_YXGJ\_TJYX | | | | |
| 用例说明 | | **测试目的：** 支持多个运行时框架产品同机运行。  **测试方法：**基于create创建好基于java运行时框架、C++运行时框架的应用实例后，分别基于start启动，成功则意味着多个运行时框架能同机运行。  **合格判据：** 多个运行时框架产品同机运行正常。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.已部署好应用沙箱且正常启动  2.已安装好java运行时框架和c++运行时框架及其相应的应用。 | create命令、java和C++运行时框架与应用镜像。 | 创建多个运行时框架的应用实例，包括两步：  1.执行沙箱客户端的create命令创建基于java运行时框架的沙箱应用实例  2.执行沙箱客户端的create命令创建基于c++运行时框架的沙箱应用实例。 | 1.基于java运行时框架的沙箱应用实例创建成功  2.基于c++运行时框架的沙箱应用实例创建成功  3.使用ps命令可以查看到新创建的沙箱实例。 | 与结果一致 |  |
| 步骤 2 | 1.已部署好应用沙箱且正常启动  2.已创建好基于c++和java运行时框架的沙箱实例。 | start命令、java和C++运行时框架与应用镜像。 | 运行多个运行时框架的沙箱应用实例，包括两步：  1.执行沙箱客户端的start命令,运行java运行时框架的应用实例。  2.执行沙箱客户端的start命令,运行c++运行时框架的应用实例。 | 1.Java运行时框架应用实例正常运行。  2.c++运行时框架应用实例正常运行。 | 与结果一致 |  |

1. 同一运行时框架多版本同机运行管理测试用例（王乐）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例名称/标识 | | 同一运行时框架多版本同机运行管理/GN\_PTGJC\_ YXGJ \_DBBTJYX | | | | |
| 用例说明 | | **测试目的：** 支持同一运行时框架产品的多版本同机运行。  **测试方法：**基于create创建好C++运行时框架1.0、2.0的应用实例后，分别基于start启动，成功则意味着多个运行时框架多版本能同机运行。  **合格判据：** 同一运行时框架产品的多版本同机运行正常。 | | | | |
| 步骤 | 前提和约束 | 输入 | 目的和动作 | 预期结果 | 评估准则 | 备注 |
| 步骤 1 | 1.已部署好应用沙箱且正常启动。  2.已安装好c++运行时框架的两个不同版本以及相应的应用。 | create命令、C++运行时框架1.0和2.0版本，以及相应的应用镜像。 | 创建同一运行时框架产品多个版本的应用实例：  1.执行沙箱客户端的create命令创建基于c++运行时框架1.0版本的沙箱应用实例。  2.执行沙箱客户端的create命令创建基于c++运行时框架2.0版本的沙箱应用实例。 | 1.基于c++运行时框架1.0的沙箱应用实例创建成功。  2.基于c++运行时框架2.0的沙箱应用实例创建成功。  3.使用ps命令可以查看到新创建的沙箱实例。 | 与预期结果一致 |  |
| 步骤 2 | 1.已部署好应用沙箱且正常启动。  2.已创建好基于c++ 1.0和2.0运行时框架的沙箱实例。 | start命令、C++运行时框架1.0和2.0版本，以及相应的应用镜像。 | 运行同一运行时框架产品多个版本的沙箱应用实例：  1.执行沙箱客户端的start命令,运行c++运行时框架1.0的应用实例。  2.执行沙箱客户端的start命令,运行c++运行时框架2.0的应用实例。 | 1.c++运行时框架1.0的应用实例正常运行。  2.c++运行时框架2.0的应用实例正常运行。 | 与预期结果一致 |  |

1. 支持对应用启动的快捷方式进行管理（王乐）
2. 支持对应用沙箱进行监视（王乐）
3. 支持对应用沙箱进行控制（王乐）