

介绍

ZSERVER4D是一套高级通讯系统的地基平台，它偏向于开发工艺和多平台支持。

功能

支持运行平台Android,IOS,Win32/64,Linux,OSX,物联网IOT(任意版本的linux均能支持，包括树莓1-3代，香橙，高通，三星，小序列cpu mips linux)

支持编译器：FPC3.0.4以及DelphiXE10.2和以后的版本

并行计算支持HPC服务器，并行深度参数服务器可配置

良好支持轻量云主机，腾讯云，阿里云，亚马逊云，均有数百台使用ZServer4D的服务器在运行中

支持内置的Pascal语系的内网穿透稳定核心库XNat(直接内核支持，非外部支持)

支持基于FRP的内网穿透(外部shell方式支持)，在公司或家里自己架设宅服 [宅服架设说明](#)

ZServer4D的前后台均支持苹果要求的IPV6审核条件，支持AAAA,A记录秒切，支持所有IPV6的云主机

内置高级加密系统，一万在线客户端会有一万把密钥，并且能动态定时更换密钥（请参考ZServer4D的附属开源项目 <https://github.com/PassByYou888/CoreCipher>）

支持去中心化网络群集，支持去中心化网络群集一键对接

内置抗量子密码支持 <https://en.wikipedia.org/wiki/SHA-3>

支持了5大美国国家标准技术研究所(NIST)高级加密标准算法

- rc6加密，通讯协议支持 <https://en.wikipedia.org/wiki/RC6>
- Twofish加密，通讯协议支持 <https://en.wikipedia.org/wiki/Twofish>
- Serpent加密，通讯协议支持 [https://en.wikipedia.org/wiki/Serpent_\(cipher\)](https://en.wikipedia.org/wiki/Serpent_(cipher))
- Mars加密，通讯协议支持 [https://en.wikipedia.org/wiki/MARS_\(cipher\)](https://en.wikipedia.org/wiki/MARS_(cipher))
- Rijndael加密，通讯协议支持 https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

架构设计可以轻松实现IP池和入口网络秒切，非常利于在国内商业环境中防止对手DDos攻击

全面支持Linux服务器开发(fpc方向)

内置NoSQL并行化内核，良好支持大数据，良好支持聚类分析，支持分布式数据库负载，支持分布式数据查询结果汇集（NoSQL技术体系从11月初开始一直处于整理中，工程较大，可能短期不能完成，但是未来会以开源形式为Delphi国内带来前沿的数据库支持体系）

开发平台支持

- Delphi及IDE要求：Delphi Rad studio XE10.2.1 or Last
- FPC编译器支持:FPC3.0.4 or last,可参看本项目随附的[IOT入手指南](#)将FPC升级至github最新的版本
- CodeTyphon 6.0 or last（尽量使用Online更新到最新的Cross工具链+相关库）

平台支持，test with Delphi 10.2 upate 1 Tokyo and FPC 3.0.4

- Windows: delphi-CrossSocket(C/S OK), delphi-DIOCP(C/S OK), delphi-ICS(C/S OK), delphi-Indy(C/S OK),delphi+fpc Synapse(C/S OK)
- Android:Indy(C/S OK), CrossSocket(Only Client)
- IOS Device: Indy(C/S OK), CrossSocket(Only Client)
- IOS Simulaor: n/a
- OSX: Indy(C/S OK), ICS(未测试), CrossSocket(C/S OK)
- Ubuntu16.04 x64 server: Indy(C/S OK), CrossSocket(C/S OK)
- Ubuntu18.04 x86+x64 Desktop:only fpc3.0.4 Synapse(C/S OK)
- Ubuntu18.04 x86+x64 Server:only fpc3.0.4 Synapse(C/S OK)
- Ubuntu18.04 arm32+arm neon Server:only fpc3.0.4 Synapse(C/S OK)
- Ubuntu18.04 arm32+arm neon desktop:only fpc3.0.4 compile ok,no test on run.
- Ubuntu16.04 Mate arm32 desktop:only fpc3.0.4 compile ok, test passed
- Raspberry Pi 3 Debian linux armv7 desktop,only fpc 3.0.4,test passed.
- wince(arm eabi hard float),windows 10 IOT,only fpc 3.3.1,test passed.

CPU架构支持，test with Delphi 10.2 upate 1 Tokyo and FPC 3.0.4

- MIPS(fpc-little endian), soft float, test pass on QEMU
- intel X86(fpc-x86), soft float
- intel X86(delphi+fpc), hard float,80386,PENTIUM,PENTIUM2,PENTIUM3,PENTIUM4,PENTIUMMM,COREI,COREAVX,COREAVX2
- intel X64(fpc-x86_64), soft float
- intel X64(delphi+fpc), hard float,ATHLON64,COREI,COREAVX,COREAVX2
- ARM(fpc-arm32-eabi,soft float):ARMV3,ARMV4,ARMV4T,ARMV5,ARMV5T,ARMV5TE,ARMV5TEJ
- ARM(fpc-arm32-eabi,hard float):ARMV6,ARMV6K,ARMV6T2,ARMV6Z,ARMV6M,ARMV7,ARMV7A,ARMV7R,ARMV7M,ARMV7EM
- ARM(fpc-arm64-eabi,hard float):ARMV8, aarch64

文档

编译指南 [CodeTyphon多架构及多平台开发陷阱](#) 在Lazarus或则CodeTyphon编译时出现缺失mtprocs库的解决办法

日常问题 [库说明](#)

[IOT完全攻略](#)

[关于XNAT内网穿透库 宅服架设\(FRP外壳支持\)](#)

[BigStream机制详解](#) [多媒体通讯CompleteBuffer BatchStream机制详解](#) [HPC服务器的工作机制详解](#) [延迟反馈机制详解](#) [序列化的命令队列机制详解](#)

[双通道机制详解](#) [p2pVM隧道技术](#) [p2pVM第二篇机理说明](#)

[云服务器框架](#) [怎样开发基于ZS的底层通讯IO接口](#) [console模式的后台程序开发](#)

[部署Ubuntu服务器的开发环境\(delphi方向\)](#) [Linux桌面开发指南\(fpc方向\)](#)

[云调度服务器用法详解](#)

[百度翻译服务后台\(支持Ubuntu16.04LTS服务器\)](#)

[百度翻译服务API\(支持Ubuntu16.04LTS服务器\)](#)

通讯接口支持(开发平台需求 Delphi Rad studio 10.2或则更高版本，低版本不支持)

1.indy 阻塞模式的通讯组件，已在ZServer4D内部集成(客户端兼容性好，服务器质量差强人意)

(open source) <http://www.indyproject.org/>

2.CrossSocket 异步式通讯组件，已在ZServer4D内部集成(服务器，客户端两边的质量都极好)

(open source) <https://github.com/winddriver/Delphi-Cross-Socket>

3.ICS异步式通讯组件，已在ZServer4D内部集成(质量很好)

(open source) <http://www.overbyte.be>

4.DIOCP 国人所开发的稳定DIOCP通讯库(服务器端的质量极好)

(Open source) <https://github.com/ymofen/diopc-v5>

通讯接口支持(FreePascal 3.0.4 or last with Lazarus，低版本不支持)

1.synapse4(open source) 已经在ZServer4D内部集成，主要支持fpc，同时也兼容delphi(客户端的兼容性好，服务器端质量很好)

synapse是支持ssl的优秀开源项目

在ZServer4D中使用Synapse的最大连接数被限制为100.

关于物联网IoT平台

ZServer4D对IoT平台的开发要求必须使用FPC编译器，ZServer4D对物联网的支持的标准系统包含一切Linux系统，要求最低FPC编译器版本为

3.0.4（需要和它对应的RT内核库）

关于IoT平台的开发测试机：本文提及到的IOT开发板都可以通过网购获取，自己动手diy Linux需要一定的耐心，懒人建议使用CodeTyphon，或则直接apt安装内置的fpc+Lazarus

关于处理机架构和大小端字节序

早期的PPC处理器架构都是大端字节序，这也造成了，早期的网络通讯标准，都是大端，它一直在影响我们使用。但是后来，到现在，大端字节序已经慢慢消失，主流的Intel处理器架构，包括ARM，X86，现在都采用了小端字节序。因此，在ZServer中，所有的二进制收发，都是以小端字节序工作的。假如你在后台需要处理大端字节序，使用外部自定义协议模式即可。

大端字节序的典型场景：比如在Indy的通讯接口中，我们发送Integer时，如果打开转换参数，它会被转换成大端字节序。

关于内存泄漏

ZServer4D内置的服务器有：Indy, ICS, CrossSocket, DIOCP, Synapse所有的服务器均无内存泄漏

ZServer4D内置的客户端接口，某些库采用的是用完抛弃的设计方式，这是针对应用程序使用的客户端库，并不是后台使用，这会有少量内存泄漏，它们是：indy, DIOCP(客户端)

有内存泄漏行为的客户端接口

- TCommunicationFramework_Client_Indy，用完抛弃
- TCommunicationFramework_Client_DIOCP，用完抛弃

无内存泄漏行为的安全客户端

- TCommunicationFramework_Client_ICS，安全回收，无泄漏
- TCommunicationFramework_Client_CrossSocket，安全回收，无泄漏
- TCommunicationFramework_Client_Synapse，安全回收，无泄漏

在ZServer4D中所捆绑的类，包括编解码，链表，数据库，均无内存泄漏

关于压力测试

压力测试如果链接超过6万，Windows系统会自动关闭侦听端口，具体原因不详，压测请尽量保持在6万以内，超过6万服务器侦听端口会自动关闭，只需要将服务器重开一次即可

关于切入和使用

ZServer4D是系统化的生产工艺地基，它并不像VCL那样傻瓜，可以拿来就用，你需要自行动手进行需求的提炼，简单来说，你必须自己动手封装，然后再使用。ZServer4D有丰富Demo和文档提供技术参考。

最后一更新日志

大更新预告:下一次更新会增加沙箱服务器模型，该模型提供paas虚拟化服务器后台的技术体系支持

目前已经支持的同构虚拟系统：vmware,virtualBox,HyperV，目前已经支持异构模拟器：QEMU

大更新预告:下一次更新会增加delphi/fpc控件形式的开发工艺，正在开发中

大更新预告:下一次更新会新增内网穿透开发组件，我们不必搭建CS服务器，直接挂载访问XNatServer也可实现远程服务

2018-10-28 内核大修

- 保证机制不变的前提，大幅修改内核流程，几乎重做内核
- 本次更新后的内核工作流程的稳定性会有前所未有的提高
- 内核会区分使用带有异步性质的程序（多线程并发+原子锁+同步回调），和非异步程序（单线程往往有更高的稳定性和可维护性）
- CrossSocket服务器本次更新后，使用内核的工作模式为非异步程序，底层为异步，接口后，以会全部同步，然后以非异步方式处理
- CrossSocket客户端本次更新后，使用内核的工作模式为非异步程序，底层为异步，接口后，以会全部同步，然后以非异步方式处理
- DIOCP服务器本次更新后，使用内核的工作模式为异步程序
- DIOCP客户端本次更新后，使用内核的工作模式为异步程序

- ICS服务器本次更新后，使用内核的工作模式为异步程序
- ICS客户端本次更新后，使用内核的工作模式为非异步程序
- INDY服务器本次更新后，使用内核的工作模式为异步程序
- INDY客户端本次更新后，使用内核的工作模式为非异步程序
- Synapse服务器本次更新后，使用内核的工作模式为异步程序
- Synapse客户端本次更新后，使用内核的工作模式为非异步程序
- 优化数据吞吐能力
- 修复CrossSocket底层在EPOLL+KQUEUE中的发送异步回调事件的死锁问题
- 内置了TimeOut检查机制
- 以ab为主，用高达20亿次请求在EPOLL(linux)+KQUEUE(unix)+IOCP(win)压测XNAT通过，整个过程持续了1天

2018-10-23

小幅度更新

- 修复:CrossSocket接口在Linux发送大数据块卡死的问题
- 修复:在Linux环境部署XNAT公网服务器后，用AB严格测试，发现互斥锁会锁不住某些线程(rt底层会线程越权)，改用delphi内置的原子锁，解决该问题，经历15分钟，1000万请求，无任何报错。XNAT只在delphi测试通过。fpc未测试。
- 升级:XNAT支持负载均衡，由外网服务器自动化全权调度，内网穿透的使用无变化
- 演示:新增指令序列化，以及5种Stream收发的Demo
- 文档:新增指令序列化的机制详解
- 修复:DelayClose缺少时间参数的问题

2018-10-22

小幅度更新

- XNAT使用RD远程桌面速度太慢的解决方案：已将后置式的代理转发机制，改为了无延迟的直接转发，负载性能会下降，但是实时性更好
- 重做XNAT内置的IPV6地址生成方式：现在会雪崩式生成无重复的IPV6地址
- XNAT新增一个基于Mapping直接构建ServerFramework的接口以及Demo，都在XNAT的演示目录
- 过去，XNAT的Data交换是后置式的，现在更改为即时交换
- CompleteBuffer增加一个压缩开关，默认使用FastCompress函数(该函数位于MemorySteam64.pas)

2018-10-20

小幅度更新

- 升级:合并了最近更新的DIOCP库
- 优化:重做地基库中的Base64编码
- 优化:内核的状态计数器不再直接使用inc操作，全部改用原子模式AtomInc，Delphi为原生的原子整数操作，因为在fpc不支持通用原子函数使用互斥锁解决
- 优化:Android平台不再考虑对雷电这类使用kvm加速模拟器支持，开发安卓系统，请使用真实设备
- 优化:调整了几处正常人类看起来不太顺眼的命名
- 工艺:在底层IO开发参考代码中编写了详细备注，看一眼即可明白接口含义，CommunicationFramework_Client_Reference.pas,CommunicationFramework_Server_Reference.pas
- 工艺:核心库的加密规则调整:不再使用固定加密算法，内核会随机使用一种加密算法进行工作，
- 工艺:ZDB新增对基础数据结构THashStringList(比THashVariantList更快)支持
- 工艺:ZDB改进网络服务的数据传输规则:凡是涉及到数据内容传输的地方，都会自动使用NIST认可的加密算法之一对其进行加密
- 工艺:将Examples中的所有的目录名全部改成英文
- 升级:本次升级已经测试通过若干已经上线项目，如不出意外，应该是大更新前最后一个小更新

2018-10-16

- 修复:修复zExpression无法对1.0e-2浮点识别的bug
- 修复:crossSocket接口反复释放的一个严重bug
- 修复:xNat内创穿透断线重连不稳定的bug，新版本的xNat公网服务器，只会在内网连接后，才会侦听端口，两端运行中现在会极其稳定
- 优化:xNat在手机也可以做内网穿透服务，优化了wiki或则4G,3G连接不稳定造成的通讯问题
- 安全:新增傻瓜化使用的抗量子密码支持(sha3)，在密码货币系统非常常见，经过验证已经与wiki一致，<https://en.wikipedia.org/wiki/SHA-3>
- 安全:因为sha3有大量的迭代计算，在摩尔定律使用秀儿算法是无法被破解的，同时sha3的计算性能也远远不如fastmd5，也许有数千倍差异，但是，使用sha3来存储和验证密码会万无一失

重新支持了5大美国国家标准技术研究所(NIST)高级加密标准算法，如下

- 安全:深度测试rc6加密, 通讯协议支持 <https://en.wikipedia.org/wiki/RC6>
- 安全:重做Twofish加密, 通讯协议支持 <https://en.wikipedia.org/wiki/Twofish>
- 安全:通讯协议支持Serpent加密 [https://en.wikipedia.org/wiki/Serpent_\(cipher\)](https://en.wikipedia.org/wiki/Serpent_(cipher))
- 安全:通讯协议支持Mars加密 [https://en.wikipedia.org/wiki/MARS_\(cipher\)](https://en.wikipedia.org/wiki/MARS_(cipher))
- 安全:通讯协议支持Rijndael加密 https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

2018-9-29

- 修复:在DataFrameEngine重做了Variant类型的读写支持, 统一多平台兼容性, 不再使用RT库自带的Variant写入方法
- 修复:FPC中Enum为4 byte定义会丢失符号位的问题
- 修复:Synapse的接口在连接失败时, 会尝试切换IPV4+IPV6重新连接
- 修复:Synapse客户端在连接失败时不返回状态
- 工艺:新增FPC泛型支持库, FPCGenericStructlist.pas
- 工艺:新增线性事件支持库, LinerAction.pas
- 工艺:新增查询服务器支持 (类似dns, 主要针对万物互联需求: 物联网设备太多, 调度服务器不够用, 可使用查询服务器分担), HostQuery.pas
- 工艺:兼容基于FPC对IOT的支持: 从底层到高级, 大规模统一调整命名, 此项调整会影响很多工程的代码细节

```
// 本项目中的回调分为3种
// call: 直接指针回调, fpc+delphi有效
// method: 方法回调, 会继承一个方法宿主的地址, fpc+delphi有效
// proc: 匿名过程回调, 只有delphi有效

// 如果本项调整对于改造现有工程有一定的工作量, 请使用字符串批量处理工具
// 在任何有回调重载的地方, 方法与函数, 均需要在后缀增加回调类型首字母说明

// 如
RunOp 变更为 RunOpP() // 后缀加P表示匿名类型回调
RunOp 变更为 RunOpM() // 后缀加M表示方法类型的回调
RunOp 变更为 RunOpC() // 后缀加C表示指针类型的回调

SendStreamCmd 变更为 SendStreamCmdP() // 后缀加P表示匿名类型回调
SendStreamCmd 变更为 SendStreamCmdM() // 后缀加M表示方法类型的回调
```

2018-9-21

兼容 fpc 3.0.0, 可正常编译, 建议使用fpc 3.0.4 or last

2018-9-18

新平台, 新平台测试通过, 在新平台树莓派3B+, 操作系统 Ubuntu16.04 Mate 下测试成功

新增几个技术文档

DoStatusIO.pas库

- 优化, 干掉后台线程刷新, 改用DoStatus方法替代线程, 最简单的使用方法:在你的主循环中, 加一句DoStatus不要给参数
- 优化, 减少对某些库的依赖性

其它优化

- 优化, FPC3.0.4编译出来的程序, 整体性能向前提升10%
- 优化, Delphi编译出来的程序, 整体性能向前提升10%
- 优化, 精细调整服务器主循环: 特别说明, 在Console模式下后台服务器程序中, 必须加上线程同步检查: CheckThreadSynchronize, 否则系统Console后台不能正常工作

2018-9-15

本次更新, 大幅提升底层库的稳定性

重大更新

- 新接口, 同时支持fpc+delphi:新增基于Synapse的通讯接口

- 内网穿透核心技术，同时支持fpc+delphi:新增内网穿透的开发库，由xNatService.pas,xNatClient.pas,xNatPhysics.pas三个小库组成，只需5行代码即可驱动，可使用ZServer支持的任意通讯接口工作。已通过5万ip/每分钟连续4小时的压力穿透测试。
- IoT物联网，只限fpc: 对IoT平台的支持，基于fpc在各平台完整支持了Synapse通讯接口，包括：ARM Linux(IoT需求)，Linux x86+x64，OSX x86+x64，Win x86+x64
- 稳定和安全，大规模取代底层库使用inline的机制

CommunicationFramework.pas库及周边支持

- 优化，稳定性提升，深度考虑安全性
- 安全，重做了p2pVM的验证系统（每个p2pVM握手时，都会用不同的验证方式）
- 安全，p2pVM第一次握手时，必须有验证码
- 工艺，新增2种协议模式：cpZServer(原来的通讯协议),cpCustom(外部自定义的通讯协议)
- 工艺，重做外部自定义通讯协议的开发工艺：开发自定义通讯协议时，不用再考虑同步异步问题
- 工艺，ProgressBackground全部统一替换成Progress
- 工艺，TCommunicationFrameworkServer服务器触发DoClientConnectAfter会区分协议，cpZServer,cpCustom会有各自处理机制
- 安全，在TCommunicationFramework中以性能换取了Progress的稳定性，客户端+服务器在高并发环境下不会再在这个地方出现异常报告了
- 周边：极小概率bug，修复CrossSocket的连接池释放时发生异常的问题(delphi)
- 周边：控制台模式服务器bug，修复ICS,Synapse,Indy服务器在Console应用模式中不触发线程同步的问题(delphi)
- 周边：小概率bug，修复ICS服务器使用StopService偶发性的出现卡死的bug
- 周边：合并了最新更新的CrossSocket内核
- 周边：优化，在MemoryStream64.pas库中对ZLib使用解压时，会预先分配内存或则文件空间，避免因为MemoryManager频繁Realloc造成性能耗损(在FPC方向程序中，性能可以相对提供10%)

TextParsing.pas库及其周边支持

- 大幅提升的解析性能，使用不变，相较以前，性能向前提升90%
- 新增可替代蚂蚁机制的文本探头技术
- 优化大规模解析程序的复杂度：降低50%
- 修复对123{abc}这种写法的误判行为

TextDataEngine.pas库

- 重做数据结构支持，使用与以前不变
- 新增了对THashStringList的内置支持(相对THashVariantList，性能更加优异)

ListEngine.pas库

- 安全：对已生成的Hash会使用安全校验措施(我实测Hash的事故率为0，但是有网友报告说Hash会莫名其妙报错，我现在加了一个安全措施，如果还遇到hash无法命中，请检查自己的bug)

百度翻译api服务器

- 由于百度不再提供免费翻译api，我们首次运行百度翻译api服务器时，会生成一个配置文件，它会指引你如何注册翻译api的账号

本次新增两套Demo

- ZServer4D对大规模后台验证服务器的工艺，比如你后台有Oracle这类大型数据库，该Demo详细描述了三方验证工艺
- XNat内网穿透库的Demo

[更多更新日志](#)

注意

REST,BAAS等等单项式的HTTP服务请自行在服务器开发和集成，ZServer4D不提供外部http支持

如果你在使用ZServer4D，并且对开发有疑问，请加群去寻找答案（请不要直接联系作者）

qq群490269542

请支持ZServer4D的后续开发 [支付宝转账](#)