BFMeta

计算机第一次尝试，用一种永续的方式

让科技进行自我复制繁殖

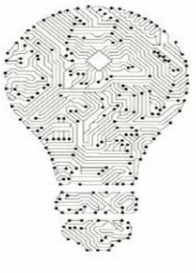
1. 摘要

元宇宙是一个基于web3.0技术体系和运行机制支撑的可信数字价值交互网络，是以公链为核心的全新web3.0数字生态。 它基于虚拟现实技术提供沉浸式体验，构建基于区块链技术的新型社会经济体系，使虚拟世界与现实世界在经济系统、社会系统、身份系统等方面紧密结合。 区块链底层操作系统是元宇宙经济系统构建的核心。BFMeta是建立在全球领先的开源BFMeta区块链操作系统之上的BFMeta元宇宙公链，继承了BFMeta公链底层高效、安全、高扩展性和高承载能力的优良性能，并在此基础上为 元宇宙的基本特征，它提供分布式存储、端到端通信、分布式数字身份、分布式信用系统、大规模消费级应用、虚拟现实技术接口、跨链兼容性等应用层面的突破和 不断创新，旨在成为全球超级元宇宙公链系统，支持全球50亿互联网用户步入新元宇宙世界。

2. 内容概述

BFMeta是一条建立在全球领先的开源Biochain Forest区块链操作系统之上的元宇宙超级公链。 通过其在区块链底层核心组件层面的一系列颠覆性创新，实现了对元宇宙经济体系建设的有力支撑。
BFMeta 的核心功能包括：
1）高扩展性
采用节能、安全、高效的共识算法，结合原生跨链、分布式存储、分布式计算等功能，可支持全球数十亿用户开发和体验Web3.0和元界应用。

2）移动端直连链



BFMeta打破了传统区块链技术必须在x86 PC上运行的限制，将区块链的应用范围延伸到移动终端，如Android终端、IOS终端、Windows终端、Linux、Unix等终端支持直接连接链，并且 终端就

是节点，节点就是服务。 因此，每个个体都可以通过移动终端直接进入元宇宙经济网络系统。

3）支持分布式数字身份DID

通过BFMeta分布式基础设施，可以实现一种新型的自我主权、可验证的分布式数字身份。 与平台控制数字身份的中心化平台不同，BFMeta上的数字身份掌握在用户手中。 BFMeta分布式数字身份的优势包括：安全性高，用户的身份信息不会泄露，信息由用户自己掌握； 自主可控，用户可以自主管理身份并控制其身份数据的共享；便携式，身份所有者可以在任何需要的地方使用它。

4）新增对Defi的支持

BFMeta既支持各类数字资产（Token）的DeFi，也支持数字产品DP的DeFi

DPFi是BFMeta针对数字产品DP的独特流动性协议，允许DP所有者以完全去信任的方式从点对点流动性提供者处获得担保股权贷款，增加其拥有的DP资产的流动性。DP流动性提供者使用DPFi来赚取有吸引力的回报，或者在贷款违约时以低于市场价值的价格获得DP。

5）跨链数字资产互操作

通过同构跨链技术以及异构跨链技术，实现数字资产和数字产品/NFT的发行和管理，支持平行链、进化链、子链、多链并发和跨链资产互操作性。

6）支持XR等数字交互技术

独特的移动端链技术可以接入任何XR硬件设备，实现元界沉浸体验与经济系统、价值流系统的高度融合，真正实现XR技术从虚拟现实游戏到元界社交经济的跨越。

7）完善的开发工具

拥有完整的开发文档、接口和SDK支持，拥有完善的区块链配置管理、服务大厅、行业协作和改造工具，可以满足全球开发者基于区块链底层的开发需求。

3、总体设计

3.1 开源移动区块链系统

传统的区块链技术应用一般适用于数据计算能力大、权益高的服务端。 随着区块链技术的逐步探索，发现目前的区块链应用以及区块链技术解决方案普遍存在一个问题：不支持移动端。 区块链应用上层都需要第三方服务节点提供移动服务。 对于基于去中心化设计、解决信用问题的区块链技术来说，需要引入第三方信用中介来提供终端服务，是违背设计初衷的。
未来的区块链需要支持移动终端。 首先，区块链是建立在互联网之上解决信用问题的新一代基础设施，它带来了进步。 如果基于区块链构建的应用仍然需要使用传统PC来进行访问，那么这在某种程度上是一种倒退。 如果区块链仅仅用在服务器端来解决机构间信用问题，将会在很大程度上限制区块链的发展。 如果区块链的应用不能面向最终用户，让最终用户感知到区块链的存在，那么这个应用与传统的中心化应用并没有本质上的区别。

当区块链不支持移动终端时，仍然可以在区块链上构建应用并提供移动终端服务，但这需要额外的第三方节点来提供区块链与移动终端之间的数据传输服务。 这种引入第三方节点的方式本质上引入了第三方中心信用，违背了区块链去中心化的设计初衷。 从某种意义上说，当用户无法直接参与区块链时，与不使用区块链没有本质区别，仍然依靠中心信用来提供服务，而这种形式的区块链在某种意义上只能称为分布式链数据库 。
因此，区块链必须支持移动终端，否则当用户感知不到区块链带来的积极意义时，就会很大程度上制约和限制区块链的发展。
然而，在做移动端区块链设计的过程中遇到了很多问题，比如：
(1)移动终端算力不足
(2) 移动网络不稳定，无法上网
(3)移动终端存储空间有限
因此，如何完善区块链的整体架构，成为解决移动区块链必须解决的挑战。

BFMeta打造了一个基于移动端的区块链系统——BFMS，全称为Bioforest Chain System

打破了传统区块链技术必须运行在x86 PC上的限制，将区块链的应用延伸到手机、平板电脑、智能穿戴等移动终端。 设备和物联网设备平台。
BFMS包括：核心应用组件、核心区块链底层技术组件、平台基础设施和开发者社区。
1）核心区块链底层技术组件：安全机制、区块链存储、链上双工通信网络、多链架构（跨链交易&区块链进化）等。
2）核心应用组件：可编程合约、可编程资产、应用框架核心协议、链上服务、API和可视化开发工具等。
3）平台基础设施：用于承载区块链系统运行过程的基本需求，包括Android、Windows、Linux、UNIX、macOS等操作系统。
4）开发者社区：BFMeta DAO治理、工具包下载、教程服务等。

接下来我们将详细介绍BFMeta移动区块链系统各项技术的创新点。
3.2 区块链存储机制
3.2.1 RSD存储机制
数据存储能力是区块链落地的基础。 由于区块链的特殊性，无法预先确定并要求参与节点提供大容量、高吞吐量的数据存储设备，因此与传统的中心化IT建设相比，在设计时需要添加以下考虑因素 。
1）存储容量
传统的IT建设可能需要提供大容量的磁盘阵列来增加容量或者将业务和历史数据分开以减轻存储压力和系统性能压力，但区块链网络无法通过上述两种方法来完成。
2) 吞吐量性能
在传统IT建设中，可能会要求节点采用高速、SSD等速度更快的存储设备，或者采用Raid1阵列磁盘并增加磁盘数量，或者将吞吐量压力分散到多个节点，或者将数据预加载到内存等。 可以有效提高吞吐量性能，但在区块链场景中不起作用，因为大多数参与节点不具备这些设备和条件。
3）RSD移动存储机构
传统IT建设中，系统架构是按照中心化服务器来设计的，并且为了考虑到终端的成本，终端不参与业务逻辑计算，终端也不存储业务数据 ，所以终端内几乎不需要移动存储，少量存储的也是不参与业务逻辑运算的用户文件和个人数据信息。 传统的存储设计方法无法应用于区块链场景。

由于现实困难，大多数区块链在存储设计上仍然遵循传统的IT建设思路，数据集中存储在参与节点中，终端参与者（如手机钱包）通过其他参与节点进行中继。

传统存储理念存在的问题

1）伪去中心化

由于终端无法直接接入区块链，实际的数据请求和业务流程都是由其他节点完成，通过参与节点是否作弊来保证数据的可信度，这违背了区块链去中心化的初衷 。

2）伪节点

如果终端无法接入区块链网络，或者接入区块链网络但数据不完整，则无法有效参与网络的治理（传统共识机制下）。 虽然它看起来像一个节点，在一定程度上可以通过桥接的方式完成间接通信，但它仍然不被认为是一个合格的参与节点。

3）无法参与共识

当终端网络和数据缺失时，将导致终端失去参与共识的基础。 无法参与共识往往意味着无法参与出块，也就意味着无法获得伴随出块而来的奖励，这对于贡献同等参与度的终端来说在某种意义上是不公平的。

BFMeta专有的RSD机制
我们重新设计了存储区块链数据的方式我们称之为关系对象存储（ROS）

对于BFMeta生态系统的未来发展，为了鼓励更多的参与者和端点访问BFMeta网络并获得公平的奖励，我们需要设计一个解决方案，考虑到不同端点面临的访问问题，并提供有效可靠的解决方案。 为此，我们首先提出了The R-Node（实时节点）、The S-Node（服务节点）、The D-Wallet（分布式钱包）的概念，以支持高性能网络节点和分布式服务节点 分别，并保证他们也能参与到共识机制中，我们重新设计了区块链的数据存储机制。 为了实现这些

理念并保证它们也能参与到共识机制中，我们重新设计了区块链的数据存储机制。

多维分片存储

BFMeta元宇宙数据存储不仅采用多磁盘，还采用了“多维切片扩展”的专利存储技术，使得数据能够在大容量存储的同时保持应用层的逻辑一致性。 数据可以存储海量，但可以保持应用层的逻辑统一性。 扩展技术使存储性能有了质的提升，为未来的综合商业应用奠定了坚实的基础。

内存图像存储

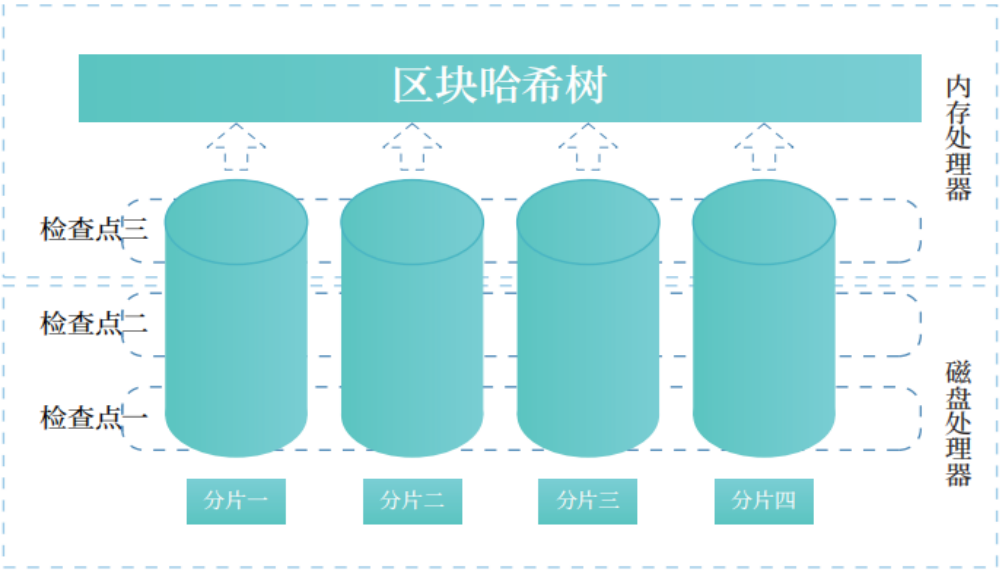
为了解决数据检索效率的问题，我们引入了目前常用的NoSQL数据库Mongodb，主要利用其在长链数据情况下的快速检索能力。 由于它的存储容量不小，不适合移动，我们对其进行了改造并与SQLite集成，让节点在保持轻量的情况下参与共识，为我们提供公平奖励的基础。

检查点存储

在RSD机制中，两种数据库各有分工，以及修改后的一种； 对于需要参与共识的移动端，需要在本地存储部分完整的区块数据，并通过这部分数据参与共识机制。 为了最大限度地减少终端存储的数据量和同步长度，我们建立了“关键检查点存储”专利技术。 终端不再需要业务计算，在同步过程中消耗额外的计算时间，而只需要存储检查点之后的区块数据。 同时提供业务节点故障时的快速启动能力。 对于关键检查点的建立，我们使用与区块相同的共识机制来完成。

哈希树存储

我们使用 Mongodb 存储区块哈希树，以提供共识机制执行过程中分叉问题的快速识别。 哈希树存储的目的是让业务在计算过程中丢弃负担数据，从而可以直接计算而不丢失历史信息。



3.2.2 私有数据存储

用照片和视频来记录生活几乎是现代社会人们工作生活的必备部分之一，而这些海量的照片数据我们往往无法完全用便携式设备来携带，我们一般会使用云设备来存储这些照片和视频 但我们的一些照片和视频可能涉及到我们工作的机密和生活的隐私，让我们不太放心地存储在公共场合，在这种背景下，一些解决方案提出使用离线加密工具来加密 照片和视频，但这反过来又增加了在特定范围内共享这些数据的复杂性。 另外，由于这些工具仍然由原始组织提供，数据仍然由原始组织存储，因此在一定程度上，人们无法完全相信这些工具是否真的被加密，或者是否秘密存储了额外的副本 加密之前，这使得当有真正的私人照片和视频存储需求时，私人存储服务很难推广。 目前还没有真正可信的工具或平台来存储和共享私人照片和视频，而其他数据如财务数据、日记、备忘录、合同等具有同类需求的数据也面临着同样的问题。 因此，如何在不增加存储和共享复杂性的情况下安全地存储和共享私人照片和视频是一个紧迫的问题。

BFMeta提供了一种安全存储和共享隐私数据的方法及其装置，通过创建群组信息和权限，获取区块链分配给群组的群组号，并为群组号分配成员账户地址，加密群组公钥 成员使用环签名，并将加密后的数据传输到区块链交易中，完成群组权限的创建和分配，并且在存储私有数据时，获取群组的环签名字符串，并对要存储的数据进行加密 使用所述环签名字符串，将加密后的数据转换成区块链交易并提交到区块链上，在读取数据时，使用访问者的私钥对数据进行解密，并将解密后的数据按照原始数据进行恢复 数据类型，实现数据私有存储和有限共享的作用。 解决了数据泄露和数据保管人的守护和窃取问题。

3.2.3 分布式存储

在这个信息技术时代，数据无处不在，它已经成为我们现代生活和工作的基础，但随着信息的发展，也出现了很多负面事件，比如无良服务提供商出卖用户隐私 专业数据窃取者、专门窃取各大信息平台用户数据的黑客团伙等等。 面对这样的数据危机，世界上有相当多的方案提供数据安全保护，比如加密存储、托管存储等； 在加密存储程序中，由于数据的不泄露取决于私钥的不泄露，而私钥来自于服务提供商，因此服务提供商在提供私钥服务的过程中严格不泄露私钥 关键是新的数据安全问题； 在数据托管存储方案中，数据的安全取决于托管存储 在数据托管存储方案中，数据的安全取决于托管方的技术实力和道德品质，而数据托管屡屡发生的数据泄露事件 平台的存在让人们对托管方技术实力的信任度逐渐降低，而托管方时不时出现的守护、窃取问题更是增添了人们的担忧。 尽管数据泄露事件不断，人们仍然继续使用这些服务的原因是个人无法提供如此大的存储容量，也无法维护一个可以随时随地提供服务的数据服务设备。 那么如何提供一种不需要专人维护、不需要第三方托管、能够提供超大规模存储容量的存储解决方案就成为一个迫切需要解决的问题。

BFMeta构建了一种去中心化的分布式数据存储方式，通过在区块链上创建账户并保存私钥，填写上传资源的描述信息，将准备上传的资源作为附件附加到区块链交易中，完成上传 通过处理预留的区块链交易并将交易放入区块来获取资源； 启动所述区块链中的节点，获取本地文件分布表，根据文件分布表检测区块链网络，将网络上比本地稀缺性更高的数据资源同步到本地文件存储库， 每天重复上述步骤同步资源，完成节点资源的同步； 需要下载资源的账户通过查找网络中的数据资源来选择需要下载的数据资源，解决了去中心化环境下数据无限可靠存储的问题。 这解决了去中心化环境中无限且可靠的数据存储问题。

3.3 区块链网络机制

3.3.1 全链路双工通信网络

区块链网络是建立整个BFMeta的基础，传统的区块链网络基本采用socket方式。

(1)Socket具有以下特点

A。 丰富的组件库，支持很早以前的编程语言，例如20世纪60年代和1970年代的cobol和c++。

b. 逻辑简单，开发方便，只需要关心传输的内容，而不需要关心底层的通信逻辑。

C。 不同的区域网络不能直接互操作，需要额外的组件来支持，例如GRPC。

(2) Socket问题
A。 网络连接的自主控制能力弱，意味着数据是实时的，传输的效率难以控制。
b. 基于Socket的应用程序无法直接与浏览器通信，这意味着以Webkit为核心的应用程序无法直接访问区块链网络。
C。 服务端主动沟通能力较弱，难以建立有效的实时推送机制

更高标准的BFMeta网络

为了实现移动终端直接参与共识机制，我们重新设计了通信域。 我们重新设计了区块链P2P网络，我们称之为——全链路双工通信。对于BFMeta生态系统未来的发展，我们需要将其设计成任何端点类型都可以轻松访问我们，而目前大多数区块链网络仍然需要中心化服务器（例如手机钱包）来提供外部服务，因此我们 对BFMeta提出了更高的设计要求——我们重新设计了P2P网络，因为没有现成的P2P网络来支持这一点。
(1)WebSocket机制介绍
针对实时性要求较高的场景，我们引入了WebSocket机制。 它为我们提供高可靠、高性能的BFMeta提供了基础。
(2)“HTTP协议”与WebSocket协议的结合
此外，我们还引入了互联网上使用最广泛的协议“HTTP协议”（后续我们将逐步升级为HTTPS），它与WebSocket协议相结合，让BFMeta的网络能力不仅可以在节点之间提供高效的互操作性， 而且跨BFMeta的网络能力不仅提供了节点之间的高效互操作，还提供了跨区域网络和终端类型的有效互操作，支持NAAS，为开发真正的分布式应用DAPP提供了基础。

3.3.2 节点寻址

随着比特币的出现，区块链技术越来越被认为是建立在互联网之上的新基础设施层。 未来，这个设施中将会有无数的网络节点，它们是支持区块链上业务运行的重要组成部分，而为业务指示具体的运行节点是区块链网络中必备的能力之一。 但在实际的业务运营环境中，由于网络运营商IP分配的限制、业务用途的变化、服务器规模调整等原因，我们需要对业务的实际运行节点进行调整，而本次调整后， 导致区块链上能找到我们业务的节点再也找不到我们了。
如果我们依靠第三方来找到我们，就会打破区块链的点对点模式，降低安全性； 那么如何在区块链网络内实现不依赖任何其他第三方的动态寻址能力，让业务节点无论如何变化都能保持业务连续性，始终让业务节点只需要记住一个名字就可以随时找到 我们，成为一个亟待解决的问题。

BFMeta构建了一种基于区块链的动态寻址方法及其系统，通过为区块链定义一个名称协议，然后通过转移区域的所有权和管理者来处理业务，当需要寻址时，通过更新业务所在的区域 节点定位并解析对应的区域位置，然后通过该区域解析出发起业务的区域地址，业务节点可以通过协议管理器中的链间协议模块和链内协议模块来处理数据 区块链节点的组成，通过区域管理器中的名称管理模块、归属管理模块、子域管理模块和位置解析模块的配合，可以对业务节点进行动态寻址，保持了业务的连续性，提高了业务的连续性。商业。 保持业务连续性，提高区块链中业务节点的处理性能。

3.3.3 蓝牙、NFC、AIRDROP网络传输

区块链是一种新型的分布式基础设施和计算方法，它利用区块链数据结构来验证和存储数据，利用分布式节点共识算法来生成和更新数据，利用密码学来保证数据传输和访问的安全，利用由自动化脚本代码组成的智能合约来编程 并操纵数据。

传统的数据连接方式包括蓝牙连接、NFC连接、AIRDROP连接等，但它们只能存在于发送数据请求的设备和作为特定请求的目标设备之间，而附近的其他设备不可能存在 设备与第三方设备建立连接并发送数据。
因此，在发送数据请求的设备以外的设备与第三方设备之间建立连接并成功发送数据是区块链领域的一个重要研究方向。
BFMeta构建基于蓝牙、NFC、AIRDROP的区块链网络传输技术，构建基于蓝牙、NFC、AIRDROP传输的区块链网络传输通道，该区块链网络上的任意节点与需要发送消息的节点相连 通过建立蓝牙、NFC、AIRDROP的拓扑组合向其他设备发送数据，但不能。当该节点收到发送数据的请求时，该节点根据发送目标地址和本地拓扑图计算出最优的数据发送路径并发送；当数据最终到达发送目标时，发送数据，除了发送数据请求的设备之外的其他设备和第三方设备之间可以建立连接并成功发送数据，解决了两个设备同时发送数据的问题。 在没有互联网的环境中不直接连接。 这解决了两个不直接连接的设备无法交换数据的问题。

3.4 区块链演进

随着比特币的出现，区块链技术的应用场景越来越广泛。 现有的区块链大多不支持主链和进化链的结构，少数支持进化链的实际上是同一节点的不同数据。 这种区块链结构存在一些问题：比如，缺乏进化链结构，很可能会导致未来所有的业务都放在主链上，从而导致所有的交易瓶颈都积压在主链上，从而导致 分配给每个业务的实际绩效低； 而同节点的进化链结构由于实际的存储和处理都在同一

个节点，这将导致未来节点变得越来越海量，而存储和交易这些问题在区块链出现时会变得尤为突出 承载着更大的业务量。
BFMeta构建了一套完整的独立进化链运行结构，即主链对节点进行与自身一模一样的精确复制，但不复制数据和创世块，进而生成具有依赖关系的创世块 根据特定的业务规则将其存储在主链上，之后将所有此类业务发送到进化链进行处理，进化链也部署在独立的节点上独立运行。 这直接将部分业务计算和存储消耗分摊到几个不同的节点上，从而大大提高了业务处理能力，为未来全网计算能力交易总量的无限扩展提供了基础。
通过BFMeta进化链的授权，企业可以根据自己的业务需求和应用场景，快速定制开发不同的业务进化链，保证企业多项业务的协同发展。 对于不需要完全上链的业务，企业也可以选择性上链，在BFMeta Biochain主链上自主开发DAPP或嵌入式开发。
BFMeta上的每一条进化链都可以与主链、平行链和其他进化链互操作，并结合BFMeta独特的跨链技术，实现各生物链之间的跨链资产交易。 这些生物链共同构成了一个活生生的生物质——BFMeta多链生态群。 该小组实现了每条进化链相互独立又与主链互联，保证每条链在生态土壤（BFMS移动区块链架构）中蓬勃发展。 同时，这个群体将在全世界开发者和用户的参与下不断演化、迭代和更新，以解决社会问题。

3.5 跨链交易

3.5.1 跨链网络互联

随着比特币的出现，区块链技术的应用场景越来越广泛。 目前的区块链设计中，各类链都是独立运行的，A链和B链不交叉，不互通，尽管采用了侧链式的技术来实现链间逻辑互通。 ，两条链之间实际上并没有真正的数据通信，而是利用第三条C链分别与A、B建立关系来实现。这种设计有两个问题，一是资源浪费，二是效率低下。
BFMeta构建了跨链网络互联方法，主要用于解决不同链之间的互操作问题。 这里所指的不同链包括两种完全不同类型的链，以及同一类型链的主链、进化链、侧链； 这里所说的互操作是指网络级的互操作。
跨链网络互联的核心设计思想是为不同的链创建一个通用的网络层适配器，并在适配器中建立每个链的处理逻辑。 这个网络适配器允许同一条链的节点之间保持现有的工作不变，但当它明确指定需要与其他链互操作时，它直接将数据直接发送到其他链的节点。 网络适配器除了显式指定数据发送方向之外，还可以起到无形的网络高速公路的作用，因为当第一条链的第一个节点需要向第一条链的第二个节点发送数据时，可能会发生： 如果这两个节点在不同的网络中不能直接互操作，那么利用本发明，第一链的第一节点可以借助

本发明，第一链的第一节点可以向第一链的第二节点发送数据。 第一链的节点可以借助与第一链的第二节点有公共连接的第二链的节点来中继数据，并且在第一链的处理逻辑中不关心第二链的存在链。

BFMeta不仅实现了跨链网络互联，还解决了网络互通概率低、效率差、设备重复浪费等问题。

3.5.2 跨链解耦

区块链是未来信用时代最重要的基础设施，而这个基础设施将由许多区块链组成，那么许多区块链之间的跨链数据交互和跨链资产转移将是必要且重要的一环。 单个区块链原本是独立的、完全自治的，但一旦涉及到跨链，可能就需要依赖其他链的可靠性。 链的话，会降低对方链的跨链数据或资产的可靠性，如果依赖于对方链，也会降低自身的可靠性。 区块链跨链交互过程中可靠性的权衡似乎是一个无法解决的问题，未来必然会有无数的区块链，跨链的需求变得刚性，那么如何保证 在不降低自身可靠性的情况下，跨链交互数据或资产的可靠性成为亟待解决的问题。

为了解决跨链可靠性与自身可靠性冲突的问题，BFMeta引入了链内授权机制，即根据链内跨链数据被确认的区块数量来确认是否允许透支。 透支期间，等待跨链网络可靠性恢复，如果透支超过等待时间，则禁止该链路下的所有交易，但自链的交易如果透支超过等待时间 等待时间，则该链路下的所有交易将被禁用，但自己链上的交易将继续正常处理，直到跨链网络恢复确认当前透支交易，然后恢复该链路下的交易。

BFMeta的链内授权机制是在对方宕机或消失时假设对超过1个区块已确认的数据进行可信处理，并对假设可信透支的资产进行负记账，并列出未偿还的资产 债务。 等待对方重建完毕，未偿债务大于0，开始等待对方买单。 当对方还没有完成买单时，对方添加的所有新的跨链交易都会在买单交易后排队，当对方一一确认该单据后，才开始处理新的交易。 这样，无论对方状态是否正常，都不会影响自己过去的验证和未来的交易，同时也不会降低跨链数据的可靠性。 若仍需要继续跨链交易，只需清除之前所有未结账单列表即可。 即可以保证两者的可靠性，又可以释放两者的相互依赖，解决跨链之间的耦合。

3.5.3 跨链资产互换

数字资产是区块链的重要组成部分，不同的区块链往往承载着不同的数字资产。 在实际业务场景中，经常需要在不同区块链之间交换资产，而这种交换往往成为一个难题； 为了尽量减少链间的资产交换，目前常见的解决方案是建立第三条区块链，分别与需要交换资产的各方建立交换关系，然后才间接交换资产。 为了尽量减少链间的资产交换，目前常见的解决方案是建立第三条区块链，分别与需

要交换资产的各方建立交换关系，然后才能间接进行资产交换，这增加了交换的步骤，延长了交易的时间，增加了相关业务的复杂度，并且由于增加了第三条链而阻碍了上层应用的进一步开发，并且在实际操作中三条链一致稳定工作的概率远低于第一条链 原有两条链，所以目前几乎没有真正的应用落地； 后来一些新的链，为了最大程度避免类似的问题，往往会采用某种最佳实践标准，而该标准也导致了一组整体结构几乎相同的链，本来就应该有一个 更好的方式来交换和流通这些链上的资产，但目前还没有出现； 那么如何设计一种不依赖第三条链的不同区块链之间的资产交换就成为一个迫切需要解决的问题。

BFMeta构建了一种在多个区块链之间流通资产的方法，包括跨链交换新资产、转移的新资产在本地链中流通以及转移的新资产再次转移回原链的步骤。 新资产转回原链还包括建立连接、确认交易和验证资产等子步骤。

BFMeta跨链交易的核心是在不同区块链之间建立微资产发行通道

当需要进行跨链资产交易时，交易对手资产将作为本链的新资产发行，并冻结本链的资产，从而使其他链的资产可以在原链和链上流通。 新链实现真正的跨链资产流通，同时可以保持原有资产总量不变。

1）资产审批人的核心功能是确认资产的有效性并管理资产总量。资产审批者有两个核心模块，资产检测模块和资产核算模块。 资产检测模块用于根据本地配置验证资产的可用性和真实性； 资产记账管理模块用于管理本链所有跨链资产及其分配和使用情况。

2）交易管理器的核心作用是帮助两条链达到共同的资产跨链状态。 交易管理器有两个核心模块，资产凭证管理模块和交易状态同步管理模块。 资产凭证管理模块用于确认和保存跨链资产的来源；交易状态膜管理模块用于协同双方完成新资产的发行确认。

3）冷冻库的销毁是为了保证资产到其他链后能够维持全局的资产总量。 销毁冷冻柜的核心功能是保持链条的全球总量完整。 销毁冷冻库中有两个核心模块，资产冻结模块和资产销毁模块。 资产冻结模块用于当新的外部资产入链时冻结链上的资产，以保证链上的资产总量在出链后保持原来的状态； 资产销毁模块用于在外部资产转回时进行销毁，同时释放链上资产，以保证外部资产转出时链上资产总量保持原状态。

3.6 三层区块链架构

随着比特币的出现，区块链技术越来越被认可。 目前的区块链结构不方便用于数据量巨大的业务场景和业务授权，尤其是检索和验证，往往需要花费大量的时间来检索，而且很多都是在做无用的检索。 目前的区块链架构是将这些数据承载在一个区块链上，因此验证和检索彩票的速度会很慢，因为它需要检索全国所有网点、所有彩票、所有发行的数据才能将信息获取到。 进行验证并检索。

BFMeta提供三层区块链架构，数据检索和验证速度快，使用简单快捷，解决繁琐的检索和验证。 它包括。
（1）构建产品级发行授权链，发行链对每个产品进行签名和授权。
（2）构建参与者授权的授权链，发行链将相应产品的生成权授权给授权链中的参与者； 授权链对每个参与者进行签名验证。
(3)构建记录实际生产运营数据的生产链，并将签约参与者的生产运营情况承载在生产链中。
通过三层区块链分离不同步骤的技术职责，从海量数据中取出正确性的验证和检索，从而显着提高区块链在海量数据应用上验证和检索时的性能，提高检索速度 和验证，并使其使用起来简单快捷。

3.7 大块

基于生物链森林系统的区块链能够锻造出包含丰富信息的区块，这得益于我们使用的记忆图像式存储、事件多进程处理、矩阵广播等算法，使得海量事件能够在一个处理过程中进行处理。 时间很短，伪造者会在这个过程中对每个事件进行签名。 锻造成功后，我们将区块头广播到区块链网络，其他节点收到并验证区块头信息后进入同步流程。 由于大块的同步会消耗较多的流量和算力，因此我们设计了一些策略来保证节点同步时的稳定性，遵循先请求先同步的原则，在保证节点稳定性的同时提供高效的服务 到外面的世界。 当某些节点已经同步到区块时，它们也会向其他节点广播，以便其他节点也可以共享其资源。

3.8 地址私钥管理机制-我的秘密

我们重新设计了地址私钥的管理机制，我们称之为My Secret。
私钥是保证用户权益和公平的底线。 在大多数区块链中，每个用户都有一对公钥和私钥，而由于私钥字符串不规则且较长，导致几乎没有用户会直接记住这个私钥，更多的时候是保存在相册中 图片二维码的形式，或者直接由第三方钱包服务商统一存储，可以获得一些直接的好处。
（1）以二维码的形式方便用户传输和保存。
（2）第三方钱包服务商统一存储的优点是用户只需记住服务商设置的密码即可。 除了这些好处之外，还存在一些隐患：

A。 图像很容易丢失。
b. 第三方服务商可能面临安全漏洞、倒闭、监管盗窃等问题，本质上是用钱包服务商的信用换取钱包密钥，违背了区块链去中心化、去信用中介的初衷。

为了在密钥安全的基础上提供便捷的密钥管理，我们设计了“MySecret”，它使用自定义的长秘密消息作为种子（可以是喜欢的台词、歌曲、诗歌），然后密钥将 被存储在密钥中。 密钥经过加密并放入链上密钥保险箱中。 解决了传统密码位数少、强度低、易被破解、原始密钥字符杂乱无意义、难以记忆、第三方中心托管机构（如blockchain.info）等一系列问题。 不可靠，让用户真正可以在去中心化环境下安全、便捷地使用密钥，而无需借助其他第三方。

3.9 自动升级

在Web 2.0时代，应用程序由于某些bug或新功能而升级和更新是很常见的。 和所有应用一样，区块链也需要升级才能跟上时代的步伐。 然而，区块链升级比常规软件升级困难得多：
A。 传统区块链的升级需要对网络进行分叉，比如以太坊将共识机制从PoW切换到PoS，这只能通过硬分叉来实现。
b. 同时，升级工作需要数月至数年的准备工作才能完成。
BFMeta 彻底改变了这一过程，使区块链能够自我升级，而无需分叉链。 这些无分叉升级是通过BFMeta公开透明、人人参与的链上治理来实现的。 凭借此功能，BFMeta使项目能够保持敏捷、适应技术并不断发展。 它还显着降低了与有争议的硬分叉相关的风险。

3.10 分叉合并

在Bioclinic区块链中，当具有不同哈希值的合法区块出现在同一高度时，区块链将创建一个临时分叉，此时网络能够快速识别并基于共识规则执行回滚合并。 回滚区块中已确认的交易也会重新广播到网络。 区块分叉共识规则按照区块参与>区块内累计费用>区块签名来选择要应用的区块。 参与度由区块内确认的交易决定，计算公式为：区块内权益变化*共识权益权重+区块内确认的交易数量*共识交易权重，其中共识权重在创世记中指定 堵塞。

3.11 分布式计算

随着互联网时代的发展，人们对信息技术的要求越来越高，越来越多的场景需要使用计算机进行计算，而单台计算机的计算能力总是

有限的，在实际应用中经常使用 集群的形式进行计算，但这往往需要构建集群的机构或个人具备一定的初始经济实力，利用经济实力转向集群的规模，从而提高整体的计算能力，但并不是所有的组织都具备一定的计算能力。 业务初期的经济实力规模，另外即使建设了一定规模的集群计算能力，但并不总是需要这么大的计算能力，往往在一些突发的业务中只需要非常高的峰值计算需求， 在大多数时候都只是低负载的计算需求，所以在实际的业务场景中，构建大规模的计算能力因此构建大规模的计算能力来满足偶尔的峰值计算并不是一个经济高效的解决方案 真实业务场景的需求。 目前市场上有一些解决方案可以解决这个问题，比如云计算，需求方可以根据业务和性能需求随时增加或减少云服务器，这种方式在一定程度上解决了问题 初始投入大，性能可以灵活配置，但仍然只是简化了构建方便灵活的计算能力的问题，并没有解决根据计算能力的需求灵活配置计算资源的问题。 那么如何在无需一次性大规模投入的情况下提供真正灵活的计算资源配置和弹性的计算资源分配方案就成为亟待解决的问题。

BFMeta发明了一种基于区块链的分布式计算方法，包括计算任务定义、计算任务分发和计算任务执行，其中计算任务定义包括任务信息录入和可处理任务类型注册，计算任务分发包括数据拆解和节点连接、计算任务执行 包括任务信息获取和任务执行，本发明还公开了一种基于区块链的分布式计算系统，包括计算任务定义模块、计算任务分发模块和计算任务执行模块，计算任务定义 模块分为任务信息录入子模块和可处理任务类型注册子模块，计算任务分发模块分为数据拆解子模块和节点连接子模块，计算任务执行模块分为分为任务信息获取子模块和任务执行子模块。 本发明的有益效果是实现了去中心化的弹性分布式计算，解决了大规模分布式计算和闲置资源浪费的问题。

BFM硬件挖矿

升级计划

BFM硬件矿机挖矿

“BITWORLD”推出自研发两款专业主力矿机BITWORLD-90T 和BITWORLD-120T系列矿机，宣布进入市场，目前已被市场 预定2000台，BITWORLD承诺到2026年逐步投入5000万美元 在中东、亚太，电能、电力设备等政策稳定的地区布局BTC、 ETH挖矿产业。

4. 共识协议

4.1 BFMeta共识算法

4.1.1 矿工协议

4.1.1.1 授权创建协议

授权创世协议，是指创世块对应的协议，由创世块的生成者签名生成。 授权协议主要是标记这个节点可以获得的能力，包括最大节点数、最大TPS、允许处理的事件范围、允许流通的资产等。 同步区块时不验证授权，而伪造区块时验证授权。这也意味着该授权文件将能够定制节点的功能。 在公链中，我们默认为所有节点提供完全授权。

4.1.1.2 创世基础协议

Genesis基础协议是指这条链的基本信息，包括链MAGIC、链名称、主权益名称、Genesis地址等。

4.1.1.3共识激励协议

共识激励协议，是通过创世块定义BFMeta网络中矿工（或节点）的分配规则和矿工为其工作获得的激励。

BFMeta在共识激励协议中引入锻造矿卡NFT和收益矿卡NFT，旨在公平和激励地分配区块奖励和手续费收益。

BFMeta全网共计锻造矿卡NFT1000张，收益矿卡NFT10000张。

锻造矿卡NFT

锻造矿卡NFT用于决定参与者是否有资格参与区块锻造。

1)入围区块锻造资格：每轮中，将根据参与者持有的锻造矿卡NFT数量决定其是否入围区块锻造。持有更多锻造矿卡NFT的参与者将有更高的几率被选中进行区块锻造。

2)奖励组成：成功锻造区块的矿工(或节点)将获得两部分矿工奖励

a. 区块基准奖励：这是矿工在成功挖掘一个新区块后所获得的新生成的数字权益(Token)。

b. 交易手续费：这是矿工将交易包含在新区块中的奖励。交易发起者在提交每笔交易时，通常包含一小部分费用(即手续费或gas费)作为激励，矿工将这些交易添加到区块中并获得所有交易的手续费总和。

矿工奖励的主要作用是激励矿工投入计算资源来验证和添加新区块，从而维护区块链网络的安全性和稳定性。通过这种奖励机制，BFMeta确保了分布式记账系统的持续运作和去中心化特性。

收益矿卡NFT

在BFMeta中，收益矿卡NFT的持有者仅参与收益分配，不参与矿工竞选，具体机制如下：

1）收益分配：每轮结束时，根据收益矿卡NFT持有者的数量进行收益分配。持有更多收益矿卡NFT的将获得更多的收益分配份额。

2)收益计算：在轮末时统一计算收益，并按照收益矿卡NFT持有者的数量和比例分配。
3)奖励来源：每轮收益矿卡NFT总奖励额度来自本轮所有区块的交易手续费之和，奖励由共识机制支付，并计入总发行量内。
通过上述奖励分配机制，BFMeta不仅激励参与者持有和使用锻造矿卡NFT和收益矿卡NFT，还确保了网络的安全性和活跃度。持有者不仅可以通过积极参与打块获得基础奖励，还可以通过持有收益矿卡NFT获得额外的分红收益，从而实现更高的整体收益。这一机制设计有助于BFMeta生态系统的稳定和可持续发展。

4.1.1.4区块锻造协议

区块锻造协议是指锻造间隔和每轮的区块数量。BFMeta的锻造间隔为15秒，一轮为50个区块。每轮的最后一个区块是回合结束区块。在回合结束块中，我们计算算法为每轮生成所需的检查点数据。

4.1.1.5合约执行协议

合约执行协议意味着这条链上的一些事件参数将由创世区块的协议决定。例如发行资产所需的最低权益值、注册链、可发放的最大奖励事件数量等配置。

4.1.1.6事件处理协议

事件处理协议是指关于该链上事件的处理能力的协议，包括最大有效性、最大TPS、单个事件的最大大小以及每字节消耗的最小处理费用。

4.1.1.7算法协议证明

算法证明协议规定了TPOW参数的规则。TPOW的难度越高，单位块处理的事件就越困难。TPOW的难度可以通过增加地址参与来降低，从而每单位块处理更多的事件。

4.1.1.8网络通信协议

该链的区块链共识端口在此定义。其他端口在配置文件中配置。

4.2 区块锻造者（矿工节点）轮换

4.2.1 多节点、多进程出块方式

我们改进了竞争打块的机制，我们称之为——CABP（Competitive Accounting Based on Participation based 竞争性记账机制）。 区块是组成区块链数据的基本单位，区块生成策略将直接影响区块链网络的性能以及参与节点的权益。
传统区块链的区块撞块有一些特点。
A。 数据完整的参与节点只能参与区块命中。
b. 算力最强或权益最高的参与节点只能参与区块命中。
C。 在大多数共识机制中，只有出块才能获得收益。

这些功能带来了一些直接的问题。

A。 从数据完整性方面解决数据可靠性问题，也制约了轻量级节点的参与。
b. 单纯以算力和股权作为撞块的基础，会导致长期的发展阻碍，并且会出现明显的资源集中和分层现象。
C。 事实上，获得收入的唯一方法是参与封锁，这将阻止节点以其他方式为参与做出贡献。

通过总结传统封锁方式带来的问题，我们基于更长远的考虑改进了封锁机制，算力和公平性不再是唯一的判断标准，我们引入了更多的维度，比如 稳定性、活跃度和交易量，让各类参与者都可以参与出块过程，这将有利于BFMeta吸引各类玩家，从而丰富参与者生态。 我们改进了玩区块奖励的机制，我们称之为——BPIM（基于参与激励机制）。
奖励是维持区块链网络所必需的。 奖励机制设计的科学性会促进区块链网络的繁荣，反过来又会制约区块链网络的发展。
传统区块链网络中的奖励方式大致有两种：
A。 通过竞争击中区块来获得区块奖励。
b. 通过参与交易收取交易费用。

这两种方式逻辑简单，易于实现，但在后期开发中存在一些问题：
A。 竞争能力低的参与节点无法获得区块奖励，这对于为其他参与做出贡献的参与节点可能不公平。
b. 按费用进行交易会带来费用歧视，出块节点可能会优先处理高费用的交易，以保证自己的收益。

为了解决传统奖励机制带来的问题，BFMeta引入了多维奖励机制，我们设计了R-Node（实时节点）、S-Node（服务节点）和D-Wallet（分布式 钱包）。 R节点和S节点可以根据网络环境或参与者的意愿相互切换或同时工作，从而使不同维度的贡献者获得奖励。 R节点和S节点可以根据网络环境或参与者的意愿相互切换或同时工作，从而使不同参与维度的贡献者获得奖励。
在奖励分配中，通过权益获得的奖励和提供的参与度都会在区块被打出时进行奖励和分配（当区块包含费用时也会被分配）。 每个服务节点都会增加服务节点的权重以获得奖励，从而在服务节点较少时鼓励服务节点的访问，在服务节点足够多时鼓励实时节点提供更高效的服务节点 ，从而通过多维度的奖励机制从多个维度动态平衡BFMeta生物链森林网络。

块锻造代码示例（部分）：

```
//发起方
```

```
import{BFMetaTrMaker}from"@BFMeta /transaction-maker-api";
import{PARENT_ASSET_TYPE}from"@ BFMeta /transaction-maker-core";
constconfig:TransactionMaker.Api.ConfigOptions={  ips:["127.0.0.1:8888"],};
const BFMTrMaker=new BFMetaTrMaker (config);
constresult=await BFMetaTrMaker.transaction.sendTransferAny({     senderId:"A",
fee:"300000",
signature:"54c12141797437d902b11f16367e47287b5c41962af71e92d72ac91f2f14aa54a5802b02508bfa172ca1aed0dd9ca7e13167cdf33238536caa3d122f6c6cf407",
remark:{
pic:"blob+sha256+hex://0fe2a72a1788a19543b7c7eca8900d471ba79182b0449ed5eb04f094ab93a8ef?size=488599",  },
amount:"5000000",
recipientId:"B",
},ipInfo);
console.log(result);
//在节点发起方我们定义了一笔交易，发起者为A，接收者为BsendTransferAny代表这是一笔转账交易，同时我们在交易体中附上了基于BFMeta底层的内容上链设计
//blob+sha256+hex代表底层约定俗成的格式，后面的字符串代表hash值，最后的size表明这个内容的长度
// 节点接收方

duplexHander.on("onNewTransaction",(tr)=>{
//这里验证这笔交易
await verifyTransaction(tr);
//在验证交易合法后，节点会接收到传输的内容
awai ttransferBlob(tr);
//最后将广播至其他节点
await broadcast(tr);})
generateBlockGenerator.on("generateblock",(block)=>{
//这里是锻造区块、处理交易
await dealTransaction(block.tr);
...锻造区块
//最后将有关的内容存储至节点中
await saveBlob()
```

```
});
//关于存储的位置，可以是节点本身，也可以是文件服务器，多个节点共享资源。


replayBlock.on("replayBlock",(block)=>{
//同步的节点将下载交易
await downloadTransaction();
//以及有关的内容,若内容验证失败，则代表内容被篡改了，无法同步本区块。
await downloadBlob();
//储存
awai tsaveBlock()
})
```

4.2.2 共识机制

4.2.2.1 共识激励机制

共识激励机制是对向持有锻造矿卡NFT并对BFMeta网络提供算术、网络、存储、验证、打包服务的矿工，以及所有持有收益矿卡NFT的地址账户发放的激励。目前，BFMeta共识激励机制中成功锻造区块的激励可以分为三个部分。

1）区块基准奖励：矿工在成功挖掘一个新区块后所获得的新生成的数字权益(Token)

2）交易手续费：当前锻造区块中所有交易产生的手续费-此手续费由交易发起者支付给矿工

3）分红奖励：按照共识激励协议，持有收益矿卡NFT的地址可获得此奖励。奖励值为本轮所有区块的交易手续费之和，奖励的权益由共识机制支付，并计入权益总发行量内。

4.2.2.2 锻造区块获得的总激励

在BFMeta网络中，每成功锻造一个区块后，底层都会计算其应获得的总激励，如下

总激励=每个区块基准奖励+sum(每个区块的交易费用)

4.2.2.3 收益分红总激励

在BFMeta网络中，按照共识激励协议，在每轮区块结束时，由轮末块统计本轮所有区块的交易手续费之和，然后按照收益矿卡NFT持有数量进行收益分配。持有更多收益矿卡NFT的地址将获得更多的收益分配份额。收益奖励由共识机制支付，并计入总发行量内。

4.2.3 分布式事务同步

区块链是未来元宇宙中数字世界不可或缺的重要基础设施，而在这个基础设施之上，将承载数百条线、数千个行业的垂直应

用，很多应用会带来海量用户，海量用户 需要区块链的支持来提供海量的交易处理能力。 区块链特殊的链式区块结构决定了同一时间只有一个区块可以成为有效区块，而该区块包含了单位时间内的交易，这也制约了同一时间只能处理一批交易时间，严重制约了区块链性能的提升。 目前业界已经采取了一些方法来解决这个问题，比如去除区块结构，但是去除区块结构会导致交易的可靠性大幅下降，这是一个巨大的成本成本。 如何在不降低交易可靠性的情况下突破链式区块结构带来的性能限制成为亟待解决的问题。

BFMeta首创了区块链分布式交易同步处理的方法，使得解决这个问题成为可能：获取已完成共识的节点列表并统计最大约定的协议版本，计算参与第二共识节点的交易范围 并发送，检查共识状态、交易适用范围、拜占庭一致性问题，节点收到交易时检查交易范围和出块时间，将交易处理结果放入新块，等待 节点的出块时间向公众公布结果，并完成交易的并行处理。 我们还提供了区块链的分布式交易同步处理系统，包括第二共识管理器、交易管理器、交易管理器、区块锻造器等，各个组件依次连接，解决了区块链多个节点（非节点）并行处理交易的问题。 -同时协作打块节点），从而提高性能。

4.3 代币销毁

BFMeta提供代币销毁的机制，在BFMeta上的主资产BFM以及通过合约发行的其他资产，都可以进行销毁。持有指定资产的地址，可以通过发行资产销毁事件，销毁一定量的资产，并且按照比例得到发行这笔资产时质押的主权益。

比例计算公式如下：

得到的主权益X=发行时质押的主权益Y*销毁的资产A数量N/资产A的总流通量M

5. 可编程合约

5.1 智能合约

BFMeta构建了一种直接从移动设备创建智能合约的方法。

交易双方将部分资产以智能合约的形式冻结到智能合约中，提交到区块链进行全网存证，达成全网共识时合约生效。 合同生效后，双方只需签署冻结协议的补充分配协议即可在该资产金额内进行转让。 由于此时总额尚未发生变化，因此补充协议只需双方签字确认即可，不再需要等待全网确认。 这种方式提高了两个经常转账的账户之间的转账速度，除了第一次冻结和最后一次解冻为正常速度外，其他时间转账速度都是瞬时的，而且费用很低。

移动区块链上的智能合约应用兼具高效、安全、简单和经济实用，更符合未来规模落地应用的要求。

BFMeta的智能合约是下一代区块链智能合约，不像目前的智能合约依赖于节点中虚拟机的执行，从而衍生出各种反直觉的限制，勉强维持解决方案的可行性； 并在每个节点上重复运行一个代码，这是针对当今能源有限的现实的一种落后设计。

分布式网络，依靠链上通信来分布式执行智能合约，最终只将经过多重签名和正确账本确认的结果存储在链上。
本质上，BFMeta的智能合约定义的并不是传统的合约虚拟机，而是一个多重签名的变量列表：它也是合约涉及的所有地址都同意的合约执行结果，以及总的合约执行结果。 账簿变更无错误会计。 这意味着区块链在同步时不需要重复执行合约，而只需要确保所有涉及的地址都同意合约结果即可。 这甚至不需要依赖代码来执行合约； 现实生活中的人们也可以就合约结果达成一致并将结果上传到链上。 这为区块链账本操作开辟了更多可能性。
这也意味着任何编程语言都可以用于合约开发，只需基于链上的分布式网络将合约涉及的地址链接起来，根据其内容执行并签名提交结果即可。 当前区块链无法采用这样的设计的根本原因在于BFMeta将分布式网络深度融入到区块链中，依靠高可用的分布式网络来获得链下即链上的效果。 所以我们可以将智能合约分发到各个节点（包括移动节点）分别执行自己的部分代码，最后进行聚合。
理论上，这种智能合约模型可以用任何编程语言执行，但它涉及到合约的幂等性，以便让任何节点都能够验证其代码执行过程。 因此，我们推荐的编程语言是 Rust/Typescript。 使用 Rust 是因为它可以将 WASM 编译得足够小、足够高，以确保所有节点都能幂等地验证合约结果。 在满足条件的环境下，也可以编译为Native，用于一些高性能场景。 这意味着开发者的编程语言中可用的任何功能和功能都可以使用，而对于执行者来说，只需选择一个满足其需求的合约，同时选择性能更好的合约即可。
我们用一个具体的场景来描述一下这个合约方案与传统合约的不同之处：假设有一个“图像数据采集合约”，需要采集指定数量的地理区域的照片，入围的提交地址获得奖励。 那么在传统区块链中，仅依靠合约代码，无法判断照片是否满足链上的地域要求和质量要求； 而在BFMeta中，依靠分布式网络收集图像数据后，决策者使用自己的关键工具在本地审查图像数据并手动选择入围照片，最终确定合同并上传到链上。 如果贡献者公布了自己的审核标准，那么他们也可以让其他地址互相推送审核，贡献者最终只需要选择自己想要的（类似于切蛋糕机和切蛋糕机分离），这样 进一步保证公平。

5.2 数字产品（DP/NFT）

DP（Digital Products）是非同质数字资产类型的缩写，是存储在区块链上的数字产品的签名证明，具有唯一性、防篡改、不可拆卸的特点。
BFMeta提供了一种标记数字产品所有权的方式。 DP作品的每一次流通和所有权变更都记录在区块链上，并生成唯一的数字证书，使其不可拆卸、不可复制、不可篡改。

DP因其唯一性、不可分割性、不可篡改性和可复制性，对于构建元宇宙具有重要意义。 可用于记录和交易数字资产，如数字作品、艺术品、产权证书、门票、游戏道具等。 同时，DP改变了传统的虚拟物品交易模式，用户可以像在现实世界中一样直接生产和交易虚拟物品。
BFMeta通过DP将现实世界的各种资产与数字世界连接起来，不断丰富元宇宙的生态多样性，从而不断拓展元宇宙的想象边界。

5.3 DeFi 支持
DeFi（Decentralized Finance），基于区块链技术和加密货币创建的去中心化金融系统，具有去中心化、公开透明、可靠、公平、安全的特点。 它已经能够使得元宇宙中虚拟身份的归属、流通、价值实现和认证成为可能。
BFMeta既支持各种数字资产（Token）DeFi，也支持数字产品DeFi。 DPFi 是 BFMeta数字产品 DP 的流动性协议，允许 DP 所有者以完全去信任的方式从点对点流动性提供者处获得担保股权贷款，增加其拥有的 DP 资产的流动性。 DP流动性提供者使用DPFi来赚取有吸引力的回报，或者在贷款违约的情况下有机会以低于市场价值的价格收购DP。

6. 可编程数字资产发行
6.1 销毁发行（通缩机制）
BFMeta构建了基于区块链代币销毁的资产发行方法、系统和装置。

6.2 去中心化资产交易所
资产是社会活动中生产生活的重要组成部分，既是生产的必需要素，又是社会发展的重要驱动因素，其在社会中的流通效率在很大程度上影响着社会发展的前进速度。
如何加快资产的流通效率，加快社会的发展，是相关机构和个人整个社会活动的一致努力。
传统方式下，资产的流通往往依赖于一个中央机构，比如房屋中介、公证处、物业管理中心、知识产权交易所等，他们充当中介机构，为资产流通的双方提供服务，或者 由中介机构介入持有资产和资金来担保交易，或者由中介机构见证交易，这在一定程度上缓解了资产流通的问题，但并不能很好地解决任何一种中介机构的缺失，很大程度上会导致 导致资产流通失败，而中介机构的同时参与决定了资产流通完成效率低下；
但如果让双方直接交易，很有可能因为信任问题而无法达成协议。 那么，如何建立一种双方都信任且不受第三方效率影响的资产流通方式就成为亟待解决的问题。

BFMeta构建了一种去中心化的资产交换方式，发行方填写发行主体和发行资产的信息，发行方使用数字证书对交易进行签

名，将区块链交易提交到区块链上，并将交易放入区块链中。堵塞; 参与方使用私钥对身份特征信息进行加密并转化为区块链交易，完成实名认证； 当区块链提取出资产的信息后，发送方填写资产转移信息，并使用发送方的签名来签署并发送区块链交易； 交易对手接收到区块链交易，对其进行签名并发送给区块链进行处理，通过验证交易的正确性完成资产交换，实现了资产去中心化、快速流动的作用，解决了交易带来的信用风险。 中介和资产流通效率低下的问题。

7. 连锁服务

7.1 链域名-LNS

在Web 2.0中，由于一串数字IP地址不易记忆、不能显示地址组织的名称和性质等缺点，因此设计了域名，并使用DNS（Domain Name System）来映射 域名和IP地址相互关联，使人们更容易访问互联网，而无需记住机器可以直接读取的IP地址的数量。 IP 地址数字字符串。

在区块链系统中，节点IP也不容易记住。 我们是否可以像域名系统一样，为区块链上的每个节点设计一个对应的链域名？同时区块链可以解决Web 2.0互联网上各个网站的安全和隐私问题，那么我们是否可以在区块链上开发一个新的链上域名系统，实现链上网站的去中心化、分布式访问呢？

BFMeta构建了一个新的分布式位置名称服务LNS，称为“Location Name Service”。 组织/用户可以注册或购买LNS并为其创建相应的DWeb站点，人们可以轻松访问该站点。 LNS位置名称服务的出现，在区块链复杂的计算机语言和人类通用语言之间架起了一座桥梁，让人们通过输入人名/组织名称+.com/cn/org就可以轻松访问区块链网站，只需 就像过去的互联网一样。

7.2 分布式网络

BFMeta使用内部发明的点对点协议为组织和个人提供 DWeb 区块链网站建设。 这些 DWeb 站点可以像常规 Web 2.0 站点一样存储网页、图像、媒体、用户数据等。

在Web 2.0时代，托管网站传统上是由“服务器”完成的，服务器可以是集中式供应商计算机或云专用计算机。 帮助DWeb站点在线，甚至可以永久开放喜爱的站点，永久在线。

用户可以在BFMeta获取LNS链域名并创建相应的DWeb站点，然后与任何其他用户共享DWeb链接。

7.3 双离线支付

互联网已成为我们现代日常生活的重要组成部分。 如果在某些特殊情况下无法上网怎么办？ 我们很可能无法订外卖、叫出租车、观看课堂视频、提交作业、与人沟通、与人交易等等。 无法以电子方式与人进行交易几乎阻碍了我们 80% 以上的日常活动。 那么有没有一种即使在网络离线的情况下也可

以使用的电子支付方式呢？ 目前已有银联卡推出的持卡零钱包、支付宝微信推出的线下支付等产品，但这些产品仍然需要商户联网才能使用，双方仍然不可能 完全脱离网络进行支付。 在当今互联网发达的时代，网络掉线的情况仍然时有发生，比如在飞机上、在远洋地区、在森林深处，停电断线、网络堵塞、光纤断线、网络拥塞等， 会导致网络不可用，因此如何在网络完全离线的情况下为交易双方提供支付服务成为一个迫切需要解决的问题。

BFMeta构建了基于区块链的离线交易方式，包括单边离线交易方式和双边离线交易方式。 在单方离线状态下，允许在线方向离线方提交支付请求； 在双边离线状态下，付款人向收款人开具不可撤销、不可否认、不可伪造的支付凭证，由收款人直接作为到账依据，当网络联网时，收款人从网络兑现该凭证。 已恢复。
BFMeta的线下交易系统，包括交易管理器、账户同步器、凭证管理器； 可以为移动终端设备提供良好的支持，即使网络不稳定也能提供应用层服务，实现离线支付的作用，解决断网时无法支付的问题。

7.4 链上红包
基于移动端“日常应用”的天然属性，BFMeta权益红利是区块链落地应用的重要创新，是帮助用户建立区块链认知的起点。数字钱包或者支付宝/微信红包上的书号变更是有本质区别的。 同时，在实物资产上链的场景下，链上红包还可以发送实物资产对应的数字资产。
7.5 服务市场
在服务市场上，BioChainlin提供各种有价值的冲浪市场（Web Application）和节点应用（Node Application），为开发者、不同类型的节点和用户提供全面、多元化的服务。
用户可以访问Surf Market上的权益链上网站，或者在节点搜索结果列表中选择喜欢的节点（例如EOW），然后下载安装使用。 区块链应用程序员还可以在“开发者社区”开发各种DApp、DWeb并部署智能合约。 在实体链场景下，数字商品发行者还可以在网上市场展示自己的产品、品牌和售后服务。

7.6 穿梭世界
在BFMeta上，企业可以根据自己的业务需求和应用场景定制开发平行链。 对于不需要完全上链的业务，企业也可以选择性上链，在BFMeta主链上自主开发DAPP或嵌入式开发。
BFMeta上的每条子链都会与其他平行链互操作，共同形成一个活生生的生物质。 这个群体将在全世界开发者和用户的参与下不断演化、迭代和更新，以解决社会问题。

对于企业来说，BFMeta呈现出三大功能：

1）赋能进化链发展
通过BFMeta的授权，企业可以根据自己的业务需求和应用场景定制开发许可链和平行公链。
2）链上实体资产及数字资产（包括数字商品、数字消费积分等）发行
BFMeta提供“数字资产锚定物理物体，链上数字孪生替代物理物体进行链上流通”，以及为实体企事业单位提供数字资产发行和管理服务。
3）链上应用开发和智能合约部署
开发者可以在BFMeta上开发各种DApp、DWeb并部署智能合约，共同构建可信的基础平台。

8. 接口文档

8.1 接口传入参数和返回参数说明

(1) 接口全名是接口的函数名，也是命令行调用时的全名。
(2) 接口缩写是命令行调用时的缩写名称。
(3)可调用方法是指接口允许被调用的方法。
（4）使用http时使用call方式，使用websocket时在/api前面添加该字符串。
(5) 请求和返回参数以 typescript 的语法描述，如果是为类型定义设计的，则以 <typeDefinition> 描述。

8.1.1 传递/输入参数示例

下面对“获取指定账户”接口的参数传递和输入进行说明。
- 接口全名：getAccountInfoAndAssets
- 接口缩写：ga
- 可调用方法：Http、Websocket、命令行、Grpc
- 调用方式：邮寄
- 接口url地址：/api/basic/getAccountInfoAndAssets
- 请求参数

```
interface GetAccountInfoAndAssets { /**account address */
address: string;}
```

- Return parameters

```
interface GetAccountInfoAndAssets extends RespCommonParam {
result: MemInfoModel.AccountInfoAndAsset;}
```

8.2 基本界面

本节将简单介绍BFMeta基本接口参数，更多详细的各参数使用及重新引用内容，请前往BFMeta开发者社区。

8.2.1 获取BFMin版本号

- 接口全名：getBFMinVersion

- 接口缩写：v
- 可调用方法：Http、Websocket、命令行
- 调用方式：获取
- 接口url地址：/api/basic/getBFMinVersion
- 请求参数：无

8.2.2 获取本地节点当前最新区块

- 接口全名：getLastBlock
- 接口缩写：glb
- 可调用方法：Http、Websocket、命令行、Grpc
- 调用方式：获取
- 接口url地址：/api/basic/getLastBlock
- 请求参数：无

8.2.3 获取指定块

- 接口全名：getBlock
- 接口缩写：gb
- 可调用方法：Http、Websocket、命令行、Grpc
- 调用方式：邮寄
- 接口url地址：/api/basic/getBlock
- 请求参数。

```
interface GetBlock {/**block signature */
signature?: string;/**block height */
height?: number;/**view the page (20 records per page) */
page?: number;}
```

8.2.4 获取指定事件

- 接口全名：getTransactions
- 接口缩写：gt
- 可调用方法：Http、Websocket、命令行、Grpc
- 调用方式：邮寄
- 接口url地址：/api/basic/getTransactions
- 请求参数。

```
interface GetTransactions {/**event id */
signature?: string;/**event block height */
height?: number;/**The minimum height of the block the event belongs to, can be used with maxHeight to query events in a block. */
minHeight?: number;/**the highest height of the block the event belongs to, can be used with minHeight to query the events of a period */
maxHeight?: number;/**event initiator */
senderId?: string;/**event recipient */
```

```
recipientId?: string;/**Event type, if not passed in then event type is not filtered, please refer to */type?: string[];/**The index value of the event, you can query the event based on the index value of the event. The index value may be a value such as assetType or signature or username, and it is recommended to use it in parallel with other conditions to find precisely. */
storageValue?: string;/**View the page (20 records per page)  */
page?: number;}
```

8.2.5 获取账户最后一笔交易

- 接口全名：getAccountLastTransaction
- 接口缩写：无
- 可调用模式：Http、Websocket
- 调用方式：邮寄
- 接口url地址：/api/basic/getAccountLastTransaction
- 描述：该接口用于获取指定地址的大概余额。 根据accountType 的返回参数 transactionAssetChanges 获取assetBalance作为余额
- 请求参数。

```
interface GetAccountInfoAndAssets {/**account address */
address: string;/**asset type */
assetType: string;}
```

8.2.6 创建账户

- 接口全名：createAccount
- 接口缩写：ca
- 可调用方法：Http、Websocket、命令行、Grpc
- 调用方式：邮寄
- 接口url地址：/api/basic/createAccount
- 请求参数。

```
interface CreateAccount {/**account key */
secret: string;}
```

8.2.7 获取节点状态

- 接口全名：getBlockChainStatus
- 接口缩写：gbc
- 可调用方法：Http、Websocket、命令行、Grpc
- 调用方式：获取
- 接口url地址：/api/basic/getBlockChainStatus
- 请求参数：无

8.2.8 根据交易类型获取账户最近一笔交易

- 接口全名：getAccountLastTypeTransaction

- 接口缩写：galtt
- 可调用方法：Http、Websocket、命令行
- 调用方式：邮寄
- 接口url地址：/api/basic/getAccountLastTypeTransaction
- 请求参数。

```
interface GetAccountLastTypeTransaction{/**account address */
address: string;/**transaction type */
transactionType: string;}
```

8.2.9 获取事件类型

- 接口全名：getTransactionType
- 接口缩写：无
- 可调用模式：Http、Websocket
- 调用方式：邮寄
- 接口url地址：/api/basic/getTransactionType
- 请求参数。

```
interface GetTransactionType{/**event base type */
baseType: {
    //equity transfer
    TRANSFER_ASSET = "AST-01",
    //secondary password
    SIGNATURE = "BSE-01",
    //register forger
    DELEGATE = "BSE-02",
    //govern voting
    VOTE = "BSE-03",
    //set username
    USERNAME = "BSE-04",
    //start receiving votes
    ACCEPT_VOTE = "BSE-05",
    //stop receiving tickets
    REJECT_VOTE = "BSE-06",
    //create DAPPID
    DAPP = "WOD-00",
    //DAPPID payment
    DAPP_PURCHASING = "WOD-01",
    //data storage certificate
    MARK = "EXT-00",
   //create equity
    ISSUE_ASSET = "AST-00",
```

```
  //destroy equity
  DESTORY_ASSET = "AST-02",
  //initiate a gift of equity
  GIFT_ASSET = "AST-03",
  //accepting a gift of equity
  GRAB_ASSET = "AST-04"};}
```

8.3 事件类接口使用说明

8.3.1 转移事件

8.3.1.1 创建传输事件

- 接口全名：trTransferAsset
- 可调用方法：Http、Websocket、命令行
- 调用方式：邮寄
- 接口url地址：/api/transaction/trTransferAsset
- 请求参数。

```
interface TrTransferAsset extends TrCommonParam {/**Number of transferred equitys, 0-9 and without decimal point, must be greater than 0 */
amount: string;/**The type of equity to be transferred, in uppercase letters, 3-5 characters */
assetType?: string;/**The name of the chain to which the equity is transferred, in lowercase letters, 3-8 characters */
sourceChainName?: string;/**Network identifier of the chain to which the equity is transferred, in uppercase letters or numbers, 5 characters, last bit is a check digit */
sourceChainMagic?: string;/**The address of the receiving account of the event, base58 encoded hexadecimal string */
recipientId: string;}
```

8.3.1.2 创建传输事件（使用安全密钥）

- 接口全名：trTransferAssetWithSign
- 可调用方法：Http、Websocket、命令行
- 调用方式：邮寄
- 接口url地址：/api/transaction/trTransferAssetWithSign
- 请求参数。

```
interface TrTransferAssetWithSign {/**buffer generated by event body without signature, generated by TrTransferAsset */
buffer: string;/**event signature */
signature: string;}
nterface SendTrCommonParam {/**Buffer to be signed, converted to base64 string */
buffer: Buffer;/**signature of the transaction */
signature: string;/**the security signature of the transaction */
signSignature?: string;
interface TrSignature extends TrCommonParam {/**new security password */
newSecondSecret: string;}
interface TrSignatureWithSign {/**buffer generated by event body without signature, generated by trSignature */
buffer: string;/**event signature */
signature: string;}
```

```
interface SendTrCommonParam {/**Buffer to be signed, converted to base64 string */
buffer: Buffer;/ */
buffer: Buffer;/**signature of the transaction */
signature: string;
 /**the security signature of the transaction */
signSignature?: string;}
```

8.3.3 设置用户名事件
8.3.3.1 创建设置用户名事件
- 接口全名：trUsername
- 可调用方法：Http、Websocket、命令行
- 调用方式：邮寄
- 接口url地址：/api/transaction/trUsername
- 请求参数

```
interface TrUsername extends TrCommonParam {/**username string, upper and lower case letters, numbers, underscores, 1-20 characters, cannot contain the name of the current chain */
alias: string;}
interface TrUsernameWithSign {/**buffer generated by event body without signature, generated by trUsername */
buffer: string;/**event signature */
signature: string;}
nterface SendTrCommonParam {/**buffer to be signed, converted to base64 string */
buffer: Buffer;/**signature of the transaction */
signature: string;
 /**the security signature of the transaction */
signSignature?: string;}
interface TrDelegate extends TrCommonParam {}
interface TrDelegateWithSign {/**buffer generated by the event body without signature, generated by trDelegate */
buffer: string;/**event signature */
signature: string;}
```

```
interface SendTrCommonParam {/**buffer to be signed, converted to base64 string */
buffer: Buffer;/**signature of the transaction */
signature: string;
 /**signature of the transaction */
signSignature?: string;}
```

8.3.5 接收轮询事件

8.3.5.1 创建接收投票事件
- 接口全名：trAcceptVote
- 可调用方法：Http、Websocket、命令行
- 调用方式：邮寄
- 接口url地址：/api/transaction/trAcceptVote
- 请求参数。

```
interface TrAcceptVote extends TrCommonParam {}
interface TrAcceptVoteWithSign {/**buffer generated by event body without signature, generated by trAcceptVote */
buffer: string;/ **event signature */
signainterface SendTrCommonParam {/**Convert the buffer that needs a signature into a base64 string. */
buffer: Buffer;/**signature of the transaction */
signature: string;
 /**the security signature of the transaction */
signSignature?: string;}
interface TrRejectVote extends TrCommonParam {}
interface TrRejectVoteWithSign {/**buffer generated from event body without signature, generated by trRejectVote */
buffer: string;/**event signature */
signature: string;}
interface SendTrCommonParam {/**Convert the buffer that needs a signature into a base64 string. */
buffer: Buffer;/**signature of the transaction */
signature: string;
 /**the security signature of the transaction */
signSignature?: string;}
interfaceTrVote extends TrCommonParam {/**the number of equity cast, 0-9 and without decimal points, 0 allowed */
equity: string;/**the address of the receiving account for the event, base58 encoded hexadecimal string */
interface TrVoteWithSign {/****buffer generated from event body without signature, generated by trVote */
buffer: string;/**event signature */
signature: string;}
interface SendTrCommonParam {/**Convert the buffer that needs a signature into a base64 string. */
buffer: Buffer;/**signature of the transaction */
signature: string;
 /**the security signature of the transaction */
signSignature?: string;}
interface TrDapp extends TrCommonParam {
  /**dappid without checksum, upper case or numeric, 7 characters */
  newDappid: string;
  /**The type of the dappid, can only be 0 or 1, 0 means the dappid is paid, 1 means the dappid is free. */
  type: number;
  /**The number of benefits needed to purchase the right to use the dappid (must be carried if the dappid is a paid app, no need to carry it if it is a free app), 0-9 and no decimal points, must be greater than 0. */
  amount: string;
  /**the recipient account address of the event, base58 encoded hexadecimal string */
  recipientId?: string;}
interface TrDappWithSign {
  /**buffer generated from event body without signature, generated by trDapp */
  buffer: string;
```

```
  /**event signature */
  signature: string;}
interface SendTrCommonParam {
  /**Convert the buffer that needs a signature into a base64 string. */
  buffer: string;
  /**event signature */
  signature: string;
  /**the security signature of the transaction */
  signSignature?: string;}
```

```
interface TrDappPurchasing extends TrCommonParam {
  /**the recipient account address of the event, base58 encoded hexadecimal string */
  recipientId: string;
  /**the dappid to which the certificate belongs, an uppercase acquisition array, 8 characters
*/
  dappid: string;
  /**The type of the dappid, can only be 0 or 1, 0 means the dappid is a paid type, 1 means
the dappid is a free type. */
  type: number;
  /**the number of dappid purchase assets */
  purchaseAsset: number;}
interface TrDappPurchasingWithSign {
  /**buffer generated from event body without signature, generated by trDappPurchasing
*/
  buffer: string;
  /**event signature */
  signature: string;}
nterface SendTrCommonParam {
  /**Convert the buffer that needs a signature into a base64 string. */
  buffer: string;
  /**event signature */
  signature: string;
  /**the security signature of the transaction */
  signSignature?: string;}
interface TrMark extends TrCommonParam {
  /**the contents of the certificate, as an arbitrary string */
  content: string;
  /**the type of the certificate, as an arbitrary string, used to distinguish the certificate */
  action: string;
  /**the dappid of the certificate, in uppercase letters, 8 characters */
  dappid: string;
  /**The type of dappid, can only be 0 or 1. 0 means dappid is paid type. 1 means dappid is
free type. */
  type: number;
  /**number of equity spent to purchase dappid */
  purchaseAsset?: number;}
interface TrMarkWithSign {
```

```
  /**buffer generated from event body without signature, generated by trMark */
  buffer: string;
  /**event signature */
  signature: string;}
interface SendTrCommonParam {
  /**Convert the buffer that needs a signature into a base64 string. */
  buffer: string;
  /**event signature */
  signature: string;
  /**the security signature of the transaction */
  signSignature?: string;}
```

```
nterface TrIssueAsset extends TrCommonParam{
    /**the name of the issued asset, in uppercase letters, 3-5 characters */
    assetType: string;
    /**The total number of new equitys issued, the number of equitys consists of ten numbers from 0-9, the number of equitys does not contain a decimal point and must be greater than 0. */
    expectedIssuedAssets: string;
    /**The address of the creation account of the new entitlement, base58 encoded hexadecimal string, this address must be given to the originating account of this event to transfer the master entitlement of this chain */
    recipientId: string;}
interface TrIssueAssetWithSign {
  /**buffer generated from event body without signature, generated by trMark */
  buffer: string;
  /**event signature */
  signature: string;}
interface SendTrCommonParam {
  /**Convert the buffer that needs a signature into a base64 string. */
  buffer: string;
  /**event signature */
  signature: string;
  /**the security signature of the transaction */
  signSignature?: string;}
interface TrDestroyAsset extends TrCommonParam{
    /**The number of equitys to be destroyed, 0-9 and without decimal points, must be greater than 0. */
    amount: string;
    /**the name of the destroyed asset, in uppercase letters, 3-5 characters */
    assetType: string;
    /**the issuing account address of the equity, base58 encoded hexadecimal string */
    recipientId: string;}
```

```
interface TrDestroyAssetWithSign {
  /**buffer generated from event body without signature, generated by trMark */
```

```
  buffer: string;
  /**event signature */
  signature: string;}
interface SendTrCommonParam {
  /**Convert the buffer that needs a signature into a base64 string. */
  buffer: string;
  /**event signature */
  signature: string;
  /**the security signature of the transaction */
  signSignature?: string;}
```

```
interface TrToExchangeAsset extends TrCommonParam{
    /**the network identifier of the source chain of equitys to be exchanged, consisting of upper case letters or numbers, 5 characters, the last bit is a check digit */
    toExchangeSource: string;
    /**the network identifier of the equity source chain to be exchanged, in uppercase letters or numbers, 5 characters, the last bit is a check digit */
    beExchangeSource: string;
    /**the name of the equity source chain to be exchanged, in lowercase letters, 3-8 digits */
    toExchangeChainName: string;
    /**the name of the source chain of the equity being exchanged, in lowercase letters, 3-8 bits */
    beExchangeChainName: string;
    /**the name of the equity to be exchanged, in uppercase, 3-5 characters */
    toExchangeAsset: string;
    /**the name of the equity being exchanged, in uppercase, 3-5 characters */
    beExchangeAsset: string;
    /**The number of equitys to be exchanged, 0-9 and without decimal points, must be greater than 0. */
    toExchangeNumber: string;
    /**used as the denominator for the exchange ratio with equity, a positive integer. exchangedEquity = exchangedEquity *exchange ratio */
    prevWeight: string;
    /**the numerator of the exchange ratio with equity, a positive integer. exchanged equity = exchanged equity * exchange ratio */
    nextWeight: string;
    /**cryptographic key set: if the key is filled, the event receiving the equity exchange must carry a signature pair generated by some key.  */
    ciphertexts?: string[];}
interface TrToExchangeAssetWithSign {
  /**buffer generated from event body without signature, generated by trMark */
  buffer: string;
  /**event signature */
  signature: string;}
interface SendTrCommonParam {
  /**Convert the buffer that needs a signature into a base64 string. */
  buffer: string;
```

```
  /**event signature */
  signature: string;
  /**the security signature of the transaction */
  signSignature?: string;}
```

```
interface TrBeExchangeAsset extends TrCommonParam{
    /**to event signature, 128-byte hexadecimal string */
    transactionSignature: string;
    /**To exchange the number of equitys, the number of equitys consists of ten numbers from 0-9, the number of equitys does not contain a decimal point and must be greater than 0. */
    beExchangeNumber: string;
    /**The number of exchanged equitys consists of ten numbers from 0-9. the number of equitys does not contain a decimal point and must be greater than 0. */
    toExchangeNumber: string;
    /**encryption key: if the key is filled in for an equity exchange event, it must carry the key specified for an equity exchange event to generate a key signature pair. */
    ciphertext?: string;
    /**to event's originating account address, base58 encoded hexadecimal string */
    recipientId: string;}
interface TrBeExchangeAssetWithSign {
  /**buffer generated from event body without signature, generated by trMark */
  buffer: string;
  /**event signature */
  signature: string;}
interface SendTrCommonParam {
  /**Convert the buffer that needs a signature into a base64 string. */
  buffer: string;
  /**event signature */
  signature: string;
  /**the security signature of the transaction */
  signSignature?: string;}
```

8.4 节点管理界面使用说明

8.4.1 节点安全关闭

- 接口全名：safetyClose
- 接口缩写：sfc
- 可调用方法：Http、Websocket、命令行、Grpc
- 调用方式：邮寄
- 接口url地址：/api/system/safetyClose
- 请求参数。

```
interface SafetyClose {/**verifyType: 001 node owner verification, 002 administrator verification */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;/**Whether the machine needs to be shut down: true means shutdown and false means no shutdown */
```

```
isShutdown?: boolean;}

interface SetSystemKey {/**old password for the node */

systemKeyOld: string;/**new password for the node */

systemKeyNew: string;/**Whether to decrypt the new password in asymmetric way (true: use asymmetric way to decrypt, false: do not use asymmetric way to decrypt, plaintext transmission) */

newKeyDecryptEnable?: boolean;}

interface VerifySystemKey {/**node password */

systemKey: string;}

interface AddSystemAdmin {/**node password */

systemKey: string;/**node administrator address: please refer to <Node Administrator> for administrator description */

systemAdminAddress: string;}

interface GetSystemAdmin {/**node password */

systemKey: string;/**node administrator address: if there is an incoming address, then return the information of that administrator address; if there is no incoming, then return the information of all administrators. */

systemAdminAddress?: string;}

interface VerifySystemAdmin {/**encrypted administrator address */

cryptoAdminAddress: string;}

interface DelSystemAdmin {/**node password */

systemKey: string;/**node administrator address: please refer to <Node Administrator> for administrator description */

systemAdminAddress: string;}

interface ResetSystemAdmin {/**node password */

systemKey: string;/**node administrator address: please refer to <Node Administrator> for administrator description */

systemAdminAddresses: string[];}

interface BindingAccount {/**node password */

systemKey: string;/**trustee private key after encryption */

cryptoSecret: string;/**encrypted Trustee Security Key */

secondSecret?: string;}

interface GetSystemDelegate {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;}

interface GetInjectGenerators {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;}

interface GetSystemDelegateDetail {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;/**受托人地址 */

address: string;}

interface GetSystemNodeInfo {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
```

```
verifyKey: string;}

interface MiningMachineInfo {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;}

interface SetSystemConfig {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;/**configuration information: all parameters here can be empty */

config: AllPartial<Config.ConfigRevisable>;}

interface GetSystemConfigInfoDetail {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;}

interface GetRuntimeState {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;}

interface GetSystemMonitor {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;/**Specify the type of access, including the traffic of accessing IP, number of times, number of accessing interfaces, number of blocks, event data, etc. */

monitorType?: string;/**The number of queries, for example, limit=10, means that 10 data can be queried. */

limit?: number;/**Query start position, for example, offset = 0, means start querying from row 1. */

offset?: number;}

interface GetSystemLoggerType {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;}

interface GetSystemLoggerList {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;/**log type */

loggerType: string;}

interface GetSystemLoggerDetail {/**node administrator address: please refer to <Node Administrator> for administrator description */

verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */

verifyKey: string;/**log file name */

loggerName: string;/**the number of queries: e.g. limit=10, means 10 data can be queried. */

limit?: number;/**query start position, e.g. offset = 0, means the query starts from the 1st row. */
```

```
offset?: number;/**the string to search */
searchString?: string;/**the way to read the file */
readFileType?: {
    readFileAsync = 0,
    createReadStream = 1,};}
interface DelSystemLogger {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;/** log file name */
loggerName: string;}
interface GetEmailAddress {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;/**the email address to check */
emailAddress?: string;}
interface SetEmailAddress {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;/**mailbox receive address */
emailToAddress: string;/**mailbox send address */
emailFromAddress: string;/**mailbox configuration */
emailConfig: {
    /**mailbox configuration type: POP3/SMTP/IMAP */
    type: string;
    /**mailboxConfig host */
    host?: string;
    /**mailbox configuration port */
    port?: number;
    /**Whether to enable mailbox security control */
    secureConnection?: boolean;
    /**Whether to enable ssl */
    ssl?: boolean;
    /**Whether tls is enabled */
    tls?: boolean;
    /**relay mailbox configured information */
    auth?: {
        /**relay mailbox configured username */
        user: string;
        /**relay mailbox configured password */
        pass: string;
    };};}
interface VerifySystemSecret {/**trustee private key after encryption */
cryptoSecret: string;}
interface SetSystemWhiteList {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
```

```
verifyKey: string;/**white list */
whiteList: string[];}
interface GetSystemWhiteList {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;}
interface DelSystemWhiteList {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;/* *white list */
whiteList: string[];}
interface GetProcessNetwork {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;/**query number: for example, limit=10 means you can query 10 pieces of data.  */
limit?: number;/**query start position: for example, offset = 0 means the query starts from row 1. */
offset?: number;/**process type */
processType?: string;}
interface GetProcessCPU {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;/**query number: for example, limit=10 means you can query 10 pieces of data. */
limit?: number;/**query start position: for example, offset = 0 means the query starts from row 1. */
offset?: number;/**process type */
processType?: string;}
interface GetProcessMemory {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;/**query number: for example, limit=10 means you can query 10 pieces of data. */
limit?: number;/**query start position: for example, offset = 0 means the query starts from row 1. */
offset?: number;/**process type */
processType?: string;}
interface SystemStatus {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;}
interface SystemProcess {/**node administrator address: please refer to <Node Administrator> for administrator description */
verifyType: string;/**check value: depending on the authority of the node visitor, password check for node owner and address check for administrator */
verifyKey: string;}
```

9. 应用工具

9.1 即时通讯-秘密聊天

Secret Chat是BFMeta块链的去中心化移动社交工具，用于群组内私人消息交换。 通过将一组节点组成逻辑组或逻辑节点，构建了“虚拟组”的概念，当需要向组中的每个人发送消息时，只需向这个虚拟节点发送消息即可，实现多对多的私密消息发送和接收，构建区块链下群组私密消息交换的结构模型。
它是区块链中消息交换的一种快速、安全的结构模型，解决了区块链中点对点群发受限的问题。
不仅支持链上发送文字、图片、语音、视频，还支持通过“Square”向好友分享、见证生活。

9.2 上帝之眼

区块链技术本身很复杂，即使是技术大牛也需要花费大量时间学习和部署。 例如，目前市场主流的比特币和以太币，只有具备专业矿机部署能力的人才能参与链上共识并获得奖励。这对于小白用户来说是不友好的。
为了方便普通用户创建区块链和部署单个节点，BFMeta开发了可视化链管理工具——上帝之眼。 通过可视化、可定制的配置界面，每个用户都可以轻松定制构建自己的区块链并进行管理。

9.3 BFM-pay

BFMeta团队研发出BFM pay钱包，接通了生物链林所有链及主流区块链的协议，实现了跨链互通。

9.3.1. 第一层BFMeta链

在BFM pay钱包中，用户可以基于支持BFMeta生物链林的链出发转账、销毁、支付密码等常见事件，合法的事件将在短时间内上链。事件处理的tps可达5000以上，手续费低也是BFMeta的技术优势。

9.3.2 第二层跨链

通过BFM pay本公链可以跨越至其他主流区块链(BTC、ETH、BSC、TRON等)，在BFM Pay中可以锻造和赎回其他公有链的资产，并且享受更高的tps(100000)更低的手续费。当用户发起跨链操作时，系统将自动识别目标链的合约特征，通过以下方式降低Gas消耗：

- **批量交易验证：**将多笔跨链交易打包成单一证明，利用链上独有的事件机制将手续费成本降低。
- **合并交易验证：**基于X-pay发出的事件，在一定条件下，将允许进行合并交易操作，从而避免无法同时上链的风险。

9.3.3 第三层联盟链及私有链

BFMeta是基础生物链林而研发的链，同样具备生物链林的联盟链及私有链的功能。在这种基础上节点可提升至百万tps。

1. 共识机制高效化

公有链（如比特币、以太坊）依赖PoW/PoS等全局共识算法，需全网节点验证交易，而联盟链/私有链采用轻量级共识。例如，私有链通过简单加密技术，将交易确认时间压缩至毫秒级，理论TPS可达百万。这种有限节点参与的快速共识机制，直接降低网络通信开销与计算延迟。

2. 网络拓扑可控性

联盟链节点通常部署在可控内网或云服务器集群中，物理距离近且带宽稳定，节点间通信延迟低于50ms。相较之下，公有链的全球分布式节点因跨地域传输导致平均延迟超过300ms。可控网络环境使区块传播速度提升6-8倍，避免因网络拥堵造成的交易堆积。

3. 资源分配定制化

私有链可针对性优化硬件配置：采用SSD存储加速状态读写，定制GPU加速加密运算，而公有链受制于异构硬件环境，无法统一优化。

4. 交易逻辑简化

联盟链业务场景聚焦（如供应链金融、存证），交易脚本复杂度显著低于公有链智能合约。以太坊合约平均执行需200ms，而BFMeta链码通过预编译等设计，将单交易执行时间压缩至5ms以内，并行处理能力提升40倍。

10. 代币分配

BFM是BFMeta的原生权益资产。

BFM发行总量为1,000,000,000枚，其中1亿枚为挖矿前权益，剩下为挖矿产生。 预挖部分100%用于生态空投及私募。

11. 免责声明

本白皮书并非建议您购买任何 BFMeta，也不是您签订任何合同或购买时应参考的文件。 本白皮书不构成买卖要约，也不构成任何类型的合同或承诺。 BFMeta无意在任何国家或司法管辖区构成证券或任何其他受监管的产品。

本白皮书不是招股说明书或任何其他证券发行文件的基础，也不构成在任何国家或司法管辖区发行或招揽证券或任何其他受监管产品。 本白皮书未经任何国家或司法管辖区的任何监管机构审查。

您承认并同意BFMeta不具备以下功能：

1. 代表BFMeta或任何其他机构在任何司法管辖区的股权、控制权或义务，或参与或控制前述机构所作出的决定的应用的权利。
2. 代表任何类型的投资。
3. 代表具有内在价值或市场价格的任何有价证券。
4. 代表任何人有义务赎回或购买的任何商品或资产。

通过参与本计划，参与者承认他或她理解并同意这些条款和条件中规定的条款和条件，并自行承担潜在风险。
1. 市场风险：如果整个加密货币市场被高估，那么投资风险将会增加，参与者可能对本计划的价格增长抱有很高的期望，但这种期望可能无法实现。
2、系统性风险：指不可抗力因素，包括但不限于自然灾害、政治动荡等。
3、监管风险：加密货币的交易具有高度的不确定性，且由于加密货币交易领域缺乏强有力的监管，导致加密货币面临大幅上涨和下跌等风险，市场经验不足的个人参与者 可能难以抵御市场不稳定带来的资产冲击和心理压力。
4、项目风险：团队将不遗余力地实现白皮书中提到的目标，目前已拥有较为成熟的商业模式，但由于行业整体发展趋势不可预测，现有的商业模式可能会受到影响。 与市场需求不匹配，难以实现盈利。 同时，由于本白皮书可能会随着项目细节的实施而更新，如果本计划的参与者不能及时获取项目的更新细节，则会因信息不对称而导致参与者知识不足 ，从而影响项目的后续发展。
5、技术风险：该项目基于密码学算法，密码学的快速发展也带来了被破解的潜在风险； 区块链、分布式存储等技术支撑核心业务发展，团队无法完全保证技术落地； 项目更新过程中，可能会发现存在漏洞，可以通过发布更新来弥补，但无法保证漏洞造成的影响程度。
6、黑客攻击和犯罪风险：在安全方面，电子代币匿名且难以追踪，容易遭到黑客攻击或被不法分子利用，或可能涉及非法资产转移等犯罪行为。
7、政策风险：目前国际上对于区块链项目以及虚拟货币各方融资的监管政策尚不明确，存在一定的可能因政策原因给参与者造成损失。
8、未知风险：随着区块链技术的不断发展，可能会存在一些目前无法预测的风险。
本白皮书不声明或保证其中描述或传达的与本计划有关的信息、声明、意见或其他事项的正确性或完整性，也不声明或保证任何转发的结果或合理性。 - 外观或概念性陈述，以及缺乏陈述和保证的情况并不限于上述内容。 本白皮书中的任何内容均不构成或不应被视为构成对未来的任何承诺或陈述。
在适用法律允许的最大范围内，对于任何人根据本白皮书采取的任何行动所引起的或与之相关的任何损失或损害，无论是由于疏忽、违约还是疏忽，我们均不承担任何责任。 。
请参赛者参赛前充分了解团队背景和整体架构，理性参与。
BFMeta保留修改和变更的权利

备注：
BFMeta相关产品已完成上线:
1.浏览器: https://tracker.bfmeta.org/#/
2.BFM Pay钱包下载：
A.手机应用市场直接搜索下载dweb Browser,
B.在Browser浏览器里输入：bfmeta.io下载BFM Pay钱包

BFMeta

BFMeta

计算机第一次尝试，用一种永续的方式
让科技进行自我复制繁殖