V 1.0

Berzeck

Nuls.io

Module Specification

**NULS 2.0 Platform - Module Specification**

Table Of Contents

[1] Design Goals 3](#_Toc529076883)

[1.1] Development Flexibility 3](#_Toc529076884)

[1.2] Deployment Flexibility 3](#_Toc529076885)

[1.3] Standard framework to build all type of complex systems 4](#_Toc529076886)

[1.4] Interconnected ecosystem 5](#_Toc529076887)

[1.5] Complete module decoupling 6](#_Toc529076888)

[1.6] Near zero down time 6](#_Toc529076889)

[1.7] Improved security and quality assurance 6](#_Toc529076890)

[1.8] Publish - Subscription development pattern 6](#_Toc529076891)

[1.9] Dedicated user interfaces 7](#_Toc529076892)

[2] Architecture High Level Conceptual Design 8](#_Toc529076893)

[2.1] Overview 9](#_Toc529076894)

[2.2] Base libraries 9](#_Toc529076895)

[2.2] Nulstar 10](#_Toc529076896)

[2.3] NULS 2.0 10](#_Toc529076897)

[3] Base Libraries Detailed Design 11](#_Toc529076898)

[3.1] Service Base Library 11](#_Toc529076899)

1] Design Goals

1.1] Development Flexibility（开发灵活性）

The platform is language agnostic so modules can be developed using different programming languages. The decision is made by developers themselves, and the only requirements are that modules use the same transport mechanism (WebSockets/JSon) and that they send their API to the Manager module.

该平台与语言无关，因此可以使用不同的编程语言开发模块。具体方案由开发人员自己决定，唯一的要求是模块使用相同的传输机制（WebSockets / JSon），并且他们将API发送到Manager模块。

This will allow developers to focus on a single module instead of learning the entire infrastructure. By focusing on just the functionality they are interested in, the learning curve will be a lot lower, and they can become productive in a shorter time span.

这将允许开发人员专注于单个模块，而不是学习整个基础架构。 通过专注于他们感兴趣的功能，学习曲线将会低很多，并且可以在更短的时间完成工作，提高效率。

Multiple Development Languages（多种开发语言）

API

Manager

C ++

API

Account

C++

Storage

Java

Block

Go

Custom Module

Rust

API

API

1.2] Deployment Flexibility（部署灵活性）

Every module is effectively a separate application, so it is possible to put most of them in a separate environment according to specific needs of each company. For example, if a module in a specific application is performance intensive then it can be deployed in several servers running concurrently.

每个模块实际上都是一个单独的应用程序，因此可以根据每个公司的特定需求将大多数模块放在单独的环境中。

例如，如果特定应用程序中的模块性能密集，则可以将其部署在多个并发运行的服务器中。

Modules Can Run Multiple Instances On Different Servers

Server 3

Server 2

Server 1

Storage

Instance 2

Storage

Instance 1

Storage

Instance 3

Block

Instance 1

Manager

Instance 1

Account

Instance 1

1.3] Standard framework to build all type of complex systems（用标准框架构建所有类型的复杂系统）

The individual components will serve as a base to build all type of systems around NULS like exchanges, social platforms, auction sites, betting sites, explorers, etc. NULS stops being just a blockchain implementation and becomes a development platform on which several complex applications will be built on.

各个组件将作为构建NULS周围所有类型系统的基础，如交易所，社交平台，拍卖网站，博彩网站等。NULS不再只是一个区块链实施，而是成为一个开发平台，可以在上面创建复杂的应用程序。

For example, to build a block explorer only one module needs to be attached to the standard modules (Described in section 2) which will handle specific functions and expose the specific API to external applications. If no custom API is required then just the front end needs to be developed.

例如，要构建区块资源管理器，只需要将一个模块连接到标准模块（在第2节中描述），该模块将处理特定功能并将特定API公开给外部应用程序。 如果不需要自定义API，则只需要开发前端。

NULS 2.0

Account

Service Base Lib

Transaction

Service Base Lib

Block

Service Base Lib

Consensus

Service Base Lib

Smart Contracts

Service Base Lib

Cross Chain

Service Base Lib

BLOCK EXPLORER FRONT END

CUSTOM MODULES

NULSTAR

Connector

Service Base Lib

Manager

Service Base Lib

Controller

Service Base Lib

Storage

Service Base Lib

Block Explorer

Server Base Lib

Explorer Front End

Web Apps Base Lib

1.4] Interconnected ecosystem （互联生态系统）

The systems built around the platform will be able to connect between each other, allowing the ecosystem to improve at lot faster and at lower costs than other cryptocurrencies. This will be possible thanks to sharing the same message protocol and also some simple API conventions.

围绕该平台构建的系统将能够彼此连接，从而允许生态系统以比其他加密货币更快且更低成本的方式进行改进。 这可以归功于共享相同的消息协议以及一些简单的API约定。

For example, if an explorer and an exchange exist then it would be a lot easier to build an application that offers functionality of both of these systems, just subscribe to the relevant functions and present the result in a frontend page; also this system could inherit the API of both base systems automatically and add its own functions allowing it to be used as base for other more complex systems and so on.

例如，如果已经存在一个explorer系统和一个exchange系统，那么构建一个提供这两个系统功能的应用程序会更容易，只需订阅相关函数并将结果呈现在前端页面中; 此系统还可以自动继承两个基本系统的API，并添加自己的功能，使其可以用作其他更复杂系统的基础等等。

ADVANCED EXCHANGE API = EXCHANGE API + EXPLORER API + CUSTOM METHODS

EXPLORER API

EXCHANGE API

COMPLEX SYSTEM

ADVANCED EXCHANGE

Explorer

Exchange

COMPLEX SYSTEM API = ADVANCED EXCHANGE API + CUSTOM METHODS

1.5] Complete module decoupling（完整的模块解耦）

The architecture guarantees that there are no hard dependencies between modules, so it is very easy to add new services runtime to the platform. If a module can’t find any service that can provide what it needs then it can gracefully report to other components about this situation without crashing the whole application.

该体系结构保证模块之间没有硬依赖关系，因此很容易向平台添加新的服务。如果模块需要的服务没有任何模块提供，那么它可以向其他组件报告此情况，而不会导致整个应用程序崩溃。

1.6] Near zero down time （停机时间接近零）

Due to its architecture, it is a lot easier to provide continuous uptime service. If a module crashes then it can gracefully be deployed again automatically using standard cloud tools. The messages that had to be delivered to it, are just put on hold for few seconds until another instance is deployed. If a module is critical then multiple working units in parallel could be deployed and the Manager would act as load balancer.

由于其架构，可以非常容易做到持续提供服务。如果模块崩溃，则可以使用标准云工具优雅地再次自动部署。传递给它的消息将被暂停几秒钟，直到部署另一个实例。如果模块很关键，则可以部署多个并行工作单元，Manager将充当负载均衡器。

Even upgrades could be done without restarting the whole infrastructure because each module can be replaced individually and the upgrades can be done in steps. Also, two versions of the same module could be deployed so that an enterprise can create a rule where it uses the new version just 5% of time to test real dynamics. This is done without risking the whole set of incoming commands, and of course if the upgrade fails then the previous version is restored gracefully and automatically.

甚至升级也可以在不重新启动整个基础架构的情况下完成，因为每个模块都可以单独更换，升级可以分步完成。 此外，可以部署相同模块的两个版本，以便可以创建规则，使用新版本只需5％的时间来测试真实的动态。 这样做不会有所有接收到的命令的风险（？？？），当然如果升级失败，则会自动恢复以前的版本。

1.7] Improved security and quality assurance （提高安全性和质量保证）

As the modules are compact and decoupled pieces of software, code reviews and tests can be done in an efficient manner. Every change just affects the integrity of the module being developed and the rest of infrastructure is not affected. It also allows for the flexibility to give different levels of testing to each module.

由于模块是紧凑且分离的软件，因此可以有效的完成代码审查和测试。每次更改只会影响正在开发的模块的完整性，其余的基础架构不会受到影响。它还允许灵活地为每个模块提供不同级别的测试。

For example, some critical modules can be audited by an external party without the need to review all platform. As these services may be executed in different servers, it is more difficult to compromise the whole platform

例如，一些关键模块可以由外部方审核，而无需审核所有平台。 由于这些服务可能在不同的服务器中执行，因此更难以破坏整个平台

1.8] Publish - Subscription development pattern （发布 - 订阅开发模式）

Most applications in crypto world are designed and developed using traditional techniques that are not well suited when internal states are changing constantly and you need to update your application accordingly. For example, when a node is downloading transactions there is a function which returns how many transactions have been processed. This function needs to be polled constantly to update the application, but polling is one of the most inefficient ways to transfer information.

加密世界中的大多数应用程序都是使用传统技术设计和开发的，当内部状态不断变化并且您需要相应地更新应用程序时，这些技术并不适用。

例如，当节点正在下载事务时，有一个函数可以返回已处理的事务数。 需要不断轮询此函数以更新应用程序，但轮询是传输信息的最低效方法之一。

With Publish - Subscription development pattern, this problem could be eliminated altogether because the function will update constantly at specific periods or events defined by the user of the function. When the application no longer needs the information it just ‘unsubscribes’ from the function.

使用Publish - Subscription开发模式，可以完全消除此问题，因为该函数将在特定时段/事件中被不断调用。当应用程序不再需要该信息时，它只是“取消订阅”该功能。

1.9] Dedicated user interfaces（专用的用户界面）

The user interfaces will be applications completely decoupled from the core architecture, and the only thing binding them is the transport protocol and an API. This is very important because user interfaces have very short development cycles, whereas the core has a long development cycle; multiple developers and designers can work in parallel without the need to coordinate with each other or even with the core development team. This property will allow the development of multiple interfaces to fulfill different needs since companies may need powerful and complex interfaces, while others light and novice-friendly ones, some may prefer web applications while others native ones.

用户界面将是完全与核心架构分离的应用程序，唯一绑定它们的是传输协议和API。这非常重要，因为用户界面的开发周期非常短，而核心的开发周期很长; 多个开发人员和设计人员可以并行工作，无需相互协调，甚至无需与核心开发团队协调。该属性将允许开发多个接口以满足不同的需求，因为有的人可能需要功能强大且复杂的接口，而有的人需要轻便且新手友好，有的人可能更喜欢Web应用程序而有的人喜欢本机应用程序。

2] Architecture High Level Conceptual Design（高层次的架构设计）

FIG 2.1

SYSTEMS

Connector

Service Base Lib

Manager

Service Base Lib

Controller

Service Base Lib

NULS 2.0

Tier 1

Tier 2

NULSTAR COMMON MODULES

Reporter

Service Base Lib

Auth

Service Base Lib

Updater

Service Base Lib

Grouper

Service Base Lib

Translator

Service Base Lib

Storage

Service Base Lib

Account

Service Base Lib

Transaction

Service Base Lib

Block

Service Base Lib

Consensus

Service Base Lib

Smart Contracts

Service Base Lib

Module 1

Service Base Lib

Module 2

Service Base Lib

Module N

Service Base Lib

Service Base Lib

. . . . . . . . . . .

Module N+1

Cross Chain

Service Base Lib

OFFICIAL APPLICATIONS

Nuls Control Center

Native Apps Base Lib

IOS, Android Wallet

Mobile Apps Base Lib

Explorer

Web Apps Base Lib

Web Light Wallet

Web Apps Base Lib

OTHER APPLICATIONS

Desktop App1

Native Apps Base Lib

Mobile App 1

Mobile Apps Base Lib

Web App 1

Web Apps Base Lib

Web App 2

Web Apps Base Lib

2.1] Overview（总体概览）

As shown in Fig. 2.1, all kind of complex systems could be built reusing NULS 2.0 components. All these systems will expose their APIs through the Connector module that interchanges messages using Json packets over WebSockets.

如图2.1所示，可以使用NULS 2.0组件构建所有类型的复杂系统。所有这些系统都将通过Connector模块公开其API，通过WebSockets使用Json数据包交换消息。

A configuration file called Modules.cfg should exist in every server instance where modules are deployed, it contains specific parameters and other directives that modules may require.

名为Modules.cfg的配置文件应存在于部署模块的每个服务器实例中，它包含特定参数和模块可能需要的其他指令。

In turn, modules can be composed by smaller components developed as plugins; for example, Consensus module can have multiple components, one for each consensus mechanism like PoC, PoS, PoW; specific components should be activated when starting the application using Modules.cfg configuration file.

反过来，模块可以由较小组件组成（插件）。例如，共识模块可以有多个组件，每个组件对应一个共识机制，如PoC，PoS，PoW，使用Modules.cfg配置文件启动应用程序时，激活特定组件。

External applications like GUI, explorer, light wallet and other applications should establish connection to the Connector module to request any information they need to operate normally. Each one of these applications plus NULS 2.0 modules could be grouped in a single package for the convenience of the end user.

GUI，资源管理器，轻型钱包和其他应用程序等外部应用程序，都应与Connector模块建立连接，以请求正常运行所需的任何信息。为了方便最终用户，这些应用程序中的每一个，加上NULS 2.0所有模块都可以组合在一个包中。

For example, the desktop wallet is just NULS 2.0 components plus NULS Control Center clustered in one single convenient package:

例如，桌面钱包只是NULS 2.0组件 + NULS Control Center，组合在一个单独的软件包中：

SYSTEM – DESKTOP WALLET

.

.

.

.

.

Nuls Control Center

Native Apps Base Lib

NULSTAR

Connector

Service Base Lib

NULS 2.0

OFFICIAL APPLICATION

2.2] Base libraries（基础库）

Base libraries (depicted with gray boxes in Fig 2.1) provide the basic standard functionality that allow modules and external applications to integrate to the platform easily and seamlessly. These libraries should be ported to multiple languages to facilitate module and application development for interested parties.

基础库（在图2.1中用灰色框表示）提供了基本的标准功能，允许模块和外部应用程序轻松无缝地集成到平台。 应将这些库移植到多种语言，以便为感兴趣的各方开发模块和应用程序。

These libraries are:

这些库是：

* Service Base Library（服务基础库）:

All modules should inherit this library, which is in charge of providing the common tools to interact with the rest of the infrastructure seamlessly, like connection routines, sending/receiving Json packages, etc.

所有模块都应继承此库，该库负责提供与其他基础架构无缝交互的通用工具，如连接路由，发送/接收Json包等。

* Native Apps Base Library（本地应用基础库）:

This library allows the development of external native desktop applications including GUI interfaces, abstracting the platform implementation details from the developer.

该库允许开发外部本机桌面应用程序，包括GUI界面，从开发人员中抽象出平台实现细节。

* Mobile Apps Base Library（移动应用基础库）:

Designed for mobile and tablet platforms

专为移动和平板电脑平台设计

* Web Apps Base Library（网页应用基础库）:

Provide the necessary methods to build web frontend to all kind of applications.

提供必要的方法来为上面所有类型的应用程序构建Web前端

2.2] Nulstar

Nulstar is formed by a group of modules that can be used to develop any kind of complex systems including NULS 2.0. All these systems will expose their APIs through the Connector module that interchanges messages using Json format over WebSockets.

Nulstar由一组模块组成，可用于开发任何类型的复杂系统，包括NULS 2.0。 所有这些系统都将通过Connector模块公开其API，该模块通过WebSockets使用Json格式交换消息。

Tier 1 consists of the minimum set of modules required to operate the underlying infrastructure.

Manager, Storage, Controller, Connector

第1层包含运行底层基础架构所需的最小模块集。（Manager, Storage, Controller, Connector）

Tier 2 represents those modules, which are optional, that provide many common functions that several systems require in a systematic and standard way.

第2层代表那些可选的模块，它们以系统和标准的方式提供若干系统所需的许多常用功能。

Reporter, Auth, Udater, Grouper, Translator

2.3] NULS 2.0

NULS 2.0 consists of the modules provided by Nulstar Tier 1 plus the specific ones that implements the NULS blockchain protocol, in a later phase Tier 2 modules will also be provided.

NULS 2.0由Nulstar Tier 1提供的模块以及实现NULS区块链协议的特定模块组成，在后续版本中还将提供第2层模块。

The list of NULS 2.0 specific modules is:

NULS 2.0特定模块的列表是：

Account, Transaction, Block, Consensus, Smart Contracts, Cross Chain

3] Base Libraries Detailed Design（基础库详细设计）

3.1] Service Base Library（服务基础库）

Every module that needs to be integrated to the platform should inherit this library. This library serves as the most basic piece of software that developers need to use for developing modules in any language which this library is ported to, as it provides several common functions. Developers may also choose to perform these functions manually.

需要集成到平台的每个模块都应该继承此库。在任何语言的实现中都应该实现它。

Its functions are:

* Provide simple basic information about the module.  
  提供关于模块的简单基本信息
* Start at least on WebSocket server to receive incoming connections.  
  至少启动WebSocket服务器，接收传入连接
* Provide functionality to add more WebSocket servers as required.  
  根据需要启动更多WebSocket服务器
* Encode packets to be sent to Json specification.  
  编码使用Json规范的数据包
* Decode and process packets received from Json specification.  
  解码使用Json规范的数据包
* Manage a queue of pending requests awaiting responses.  
  管理等待响应的待处理请求队列
* Log critical events to hard disk.  
  将关键事件记录到硬盘
* Collect all methods with their respective parameters that will conform the module’s API.  
  扫描模块提供的API的方法以及参数
* Ability to add meta information each one of the API’s methods including a description, parameter validation information and event/period limitations.  
  能够为API的每个方法添加元信息，包括描述，参数验证信息和事件/期间限制
* Connect to Manager module and provide the module’s API, metainformation and connection information.  
  连接到Manager模块并提供模块的API，元信息和连接信息
* Connect to Storage module for logging purposes.  
  连接到存储模块以进行日志记录
* Send non-critical events logging information to Storage module.  
  发送非关键事件日志信息到存储模块
* Establish and negotiate WebSocket connections to other modules as required using the connection information returned from the Manager module.  
  根据从Manager模块返回的信息，依自己的需要建立与其他模块的WebSocket连接

3.1.1] Provide simple basic information about the module（提供有关模块的简单基本信息）

Five functions that return strings are responsible for this task:

返回string的五个函数负责此任务：

* fName() .- The name of the module.   
  fName（）.-模块的名称
* fVersion() .- Version of the module. It is composed by four sections: Major, minor, bug, build. Major number is reserved when the module suffered a massive overhaul or refactoring. Changes in minor version are performed when module offers new features. Bug version changes are given for minor features or for fixing issues. And last, the build version should be increased for a new compilation.

The convention used states that if a module increases the major version then it is not backwards compatible so special care must be taken when upgrading, the module is compatible otherwise.  
fVersion（）-模块版本。它由四个部分组成：Major，minor，bug，build。  
当模块遭受大规模的大修或重构时，保留主要编号。当模块提供新功能时，将执行次要版本的更改。针对次要功能或修复问题提供了Bug版本更改。最后，应该为新编译增加构建版本。  
使用的约定规定，如果模块增加主要版本，则它不向后兼容，因此在升级时必须特别小心，否则模块是兼容的

* fDomain() .- In order to avoid clashes with third party module names and Roles, a domain must be given that serves as a prefix, its preferable to be a unique string so an owned web domain name is suggested.  
  fDomain（）.-为了避免与第三方模块名称和角色冲突，必须给出一个充当前缀的域，它最好是一个唯一的字符串，因此建议拥有自己的Web域名。
* fApiRole() .- A string the represents the specific Role this module is able to offer. The convention states that the string should start with “Role\_”, similar Roles must offer similar API’s otherwise Manager module uses the domain to specify which module to use. More than one module can run concurrently offering the same Role, in this case Manager component is used as a load balancer.  
  fApiRole（）.-表示此模块能够提供的特定角色的字符串。约定规定字符串应以“Role\_”开头，类似的角色必须提供类似的API，否则Manager模块使用域来指定要使用的模块。 多个模块可以同时运行，提供相同的角色，在这种情况下，Manager组件用作负载均衡器。
* fApiVersion .- API version number comprised of three sections: Major, Minor, Bug. Major number is reserved when the API changes are not backward compatible. Minor version is changed when the API adds new methods or modify existing ones without breaking compatibility. Bug version is used for minor changes, like method description or for changes that are backward and forward compatible with other modules which has different Bug version number but equal Major and Minor version numbers.  
  fApiVersion .- API版本号由三部分组成：Major，Minor，Bug。 当API更改不向后兼容时，保留主编号。当API添加新方法或修改现有方法而不破坏兼容性时，将更改次要版本。 Bug版本用于微小更改，例如方法描述或与其他具有不同Bug版本号但具有相同的主要版本号和次要版本号的模块向后和向前兼容的更改

NULS 2.0

NULSTAR

Storage

fName() -> “NULS Storage Module”

fVersion() -> “1.0.3.298”

fDomain() -> “nuls.io”

fApiRole() -> “Role\_StorageManager”

fApiVersion() -> “1.1.5”

Service Base Lib

3.1.2] Start at least on WebSocket server to receive incoming connections.（至少启动WebSocket服务器，接收传入连接）

By default a module needs to open a WebSocket server to process incoming connections using the port, the log level and the encryption type specified in the command line parameters when executing it (Refer to section 4), if the port is not available then it must try with the next port number and log this occurrence to Storage module, if this module is not available then it must be logged to hard disk as a critical issue.

默认情况下，模块需要打开WebSocket服务器，接收新连接，日志级别和命令行参数中指定的加密类型处理传入连接（请参阅第4节），如果端口不可用则必须尝试使用下一个端口号并将此事件记录到存储模块，如果此模块不可用，则必须作为一个非常严重的错误记录到硬盘。

The WebSocket port should only receive connection requests by modules inside the architecture, for external applications another WebSocket server should be opened by Connector module to accept external requests.

WebSocket端口应仅接收体系结构内的模块的连接请求，对于外部应用程序，连接器模块应打开另一个WebSocket服务器以接受外部请求。

Example: The Account module starts a Web Socket server at startup accepting incoming connections at port 7771.

示例：帐户模块在启动时启动Web Socket服务器，在端口7771监听传入连接。

NULS 2.0

NULSTAR

Web Socket

Port: 7771

Account

Service Base Lib

3.1.3] Provide functionality to add more WebSocket servers as required.（根据需要启动更多WebSocket服务器）

The implementation should have a provision to add more WebSocket servers for special cases controlled by the module.

应该有一个规则为特殊情况添加WebSocket服务器。

Example: The Connector module needs to open two more Web Socket servers that accepts incoming connections from external applications, one should be enabled just for privileged users and other for the rest.

示例：连接器模块需要为外部应用打开另外两个Web Socket服务器，一个仅为管理员启用，一个为其他用户启用。

NULS 2.0

NULSTAR

Connector

Service Base Lib

Web Socket

Port: 7772

Web Socket for public connections

Port: 7790

Web Socket for admin connections

Port: 7791

3.1.4] Encode packets to be sent to Json specification.（编码使用Json规范的数据包）

Eight types of Messages are currently defined that must be implemented as classes or structures (depending on implementation language):  
目前定义了八种类型的消息，必须作为类或结构实现（取决于实现语言）：

* NegotiateConnection
* NegotiateConnectionResponse
* Request
* Unsubscribe
* Response
* Ack
* RegisterCompoundMethod
* UnregisterCompoundMethod.

Please refer to Nulstar - Documentation - Message Protocol document for specific field details.

有关具体的字段详细信息，请参阅Nulstar - Documentation - Message Protocol

When a module needs to send a Message, it must instance the respective class or structure and fill the values with appropriate values, the resulting object must be passed to another called PacketProcessor that is in charge of getting values and encoding them to the selected format (JSon in this case). Several formats should be added in the future for different use cases like XML or binary.

当模块需要发送消息时，它必须实例化相应的类或结构并用适当的值填充值，生成的对象必须被PacketProcessor编码为所选格式（JSon）。 对于XML或二进制等其他格式，将来会添加。

Example: The object that holds NegotiateConnection data is passed to PacketProccesor which encodes the values to Json format

示例：将包含NegotiateConnection数据的对象传递给PacketProccesor，后者将值编码为Json格式

NULS 2.0

NULSTAR

Storage

Service Base Lib

Object NegotiateConnection : Message

ProtocolVersion: “1.0”

CompressionRate: “0”

String: NegotiateConnection

{

“NegotiateConnectioin”: {

“ProtocolVersion”: “1.0”,

“CompressionRate”: “0:”

}

}

PacketProcessor

Encode Message

to JSon

3.1.5] Decode and process packets received from Json specification.（解码使用Json规范的数据包）

Packets received should be decoded and processed depending on the Message type:

收到的数据包应根据消息类型进行解码和处理：  
（备注：下面不单独翻译，所有内容都在Message Protocol中，并且下面的内容有些是过期的，并不正确。）

* NegotiateConnection: The module should send back a Message of type NegotiateConnectionResponse, the filed NegotiationStatus should be set to “1” if the protocol is compatible and if it is able to send compressed packets with the specified compression level using **zlib** algorithm, “0” otherwise. This message type could be extended in the future with user authentication mechanism.
* NegotiateConnectionResponse: When the module receives a Message of this type, it should check the NegotiateConnection field, and send a Log Request to Storage module (See section 3.1.12 for further details). If negotiation is successful then the connection must be tagged as such so further requests/responses could be processed, otherwise 2 more tries with 10 second interval must be performed, if these tries fail then it must report to the Manager that the module can’t operate normally.
* Request: This message should be decoded and the module must process the methods specified in the RequestMethods array field with their respective parameters. A Response Message should be crafted with the required specific information. If one method fails then the Response should be considered unsuccessful as a whole even if some methods processed successfully. If Event or Period intervals are declared, methods should be processed again resulting in a new Response Message at each Event/Period.
* Response: How to process this message is defined by each module, if no more Responses are expected then PendingMessages queue should eliminate the respective entry (See section 3.1.6 for further details)
* Notification: How to process this message is defined by each module.
* RegisterCompoundMethod: The module should be able to create a new virtual method with the specified name and parameter aliases, then an API Registration Request should be sent to the Manager module to add the virtual function (Check 3.1.10 for information about crafting such request)
* UnregisterRegisterCompoundMethod: The module should be able to remove the specified virtual method, then an API Registration Request should be sent to the Manager module to add the virtual function (Check 3.1.10 for information about crafting such request).

3.1.6] Manage a queue of pending requests awaiting responses（管理等待响应的待处理请求队列）

A queue of requests awaiting response should be created to keep track on which Request messages didn’t receive a proper response, the queue must record the request id, a reference to the caller object and time of last response (this is the case when the request is sent with Event or Period more than zero so more Response Messages are expected). When no more Response Messages are expected then the appropriate record should be eliminated from the queue.

应该创建一个等待响应的请求队列来跟踪那些没有收到正确的响应的请求，队列必须记录请求id，对调用者对象的引用和最后一次响应的时间（如果发送请求的事件或时间段大于零，会需要更多响应消息。译者注：简单的说就是一个请求需要多次相同的响应）。如果不再需要响应消息，则应从队列中删除。

3.1.7] Log critical events to hard disk（将关键事件记录到硬盘）

When a module is unable to perform its functions properly and is unable to send a Log request to Storage Module then it should append a Critical error type to a file named Error.log with the following format:

当模块无法正常执行其功能并且无法向存储模块发送日志请求时，它应该将严重错误类型附加到名为Error.log的文件，格式如下：

Date|Time|Time Zone|EventLevel|Id|Critical error description

3.1.8] Collect all methods with their respective parameters that will conform the API（扫描模块提供的API的方法以及参数）

The base class should be able to collect the module methods that will conform the API using introspection techniques, each method should be tagged with one of the three specified types:

基类应该能够扫描模块提供的API，每个方法都应该用三种指定类型之一标记：

* Public: The method should be exposed to internal modules and external applications as well.  
  公共：该方法也应该暴露给内部模块和外部应用程序
* Admin: Means that the method should be exposed to internal modules and external applications as well but only through the Admin Web Socket server specified in the Connector module.  
  Admin：表示该方法也应该暴露给内部模块和外部应用程序，但只能通过Connector模块中指定的Admin Web Socket服务器
* Protected: The method should be exposed only to internal modules.  
  受保护：该方法应仅暴露给内部模块

3.1.9] Ability to add metadata information for each one of the API’s methods including a description, parameter validation information and event/period limitations  
（能够为API的每个方法添加元信息，包括描述，参数验证信息和事件/期间限制）

Optionally modules should send three more types of method metadata:

模块应该发送三种类型的方法元数据：

* Method Description: This provides the description and/or help text of the method.  
  方法描述：这提供了方法的描述和/或帮助文本。
* Parameter validation: Ranges for numbers and regular expressions for strings could be sent to Manager module so validation checks are performed even before a Request Message is received.  
  参数验证：字符串的数字和正则表达式的范围可以发送到Manager模块，因此即使在收到请求消息之前也会执行验证检查。
* Event/Period limitations: The minimum Period in seconds and number of Events that the method should send back Response Messages.  
  事件/期间限制：方法应发送回响应消息的最小周期（秒）和事件数。

3.1.10] Connect to Manager module and provide the module’s API, metainformation and connection information

When starting a module, a Request Message invoking RegisterAPI should be sent to Manger module with information about each method and their respective metadata, also a string array with required Roles with their minimum API version should be attached.

Example:

{

“Request”: {

“RequestID”: “sdj8jcf8899ekffEFefee”,

“RequestInternalID”: “348022847492”,

“RequestDate”: “2018-11-05”,

“RequestTime”: “03:00:00”,

“RequestTimeZone”: “-4”,

“SubscriptionEventCounter”: “0”,

“SubscriptionPeriod”: “0”,

“ResponseMaxSize: “0”,

“RequestMethods”: [

{

"RegisterAPI": {

"ServiceAPIVersion": "0.1.0",

"ServiceDomain": "nuls.io",

"ServiceName": "NULS Connector",

"ServiceRole": "Role\_ConnectionManager",

"ServiceVersion": "0.0.1.1",

“RoleDependencies”: [ { “Role\_StorageManager”, “0.1.0” } ],

“ServiceIP”: “130.34.32.44”,

“ServicePort”: “7775”,

"Methods": [

{

"MethodDescription": "Sets the maximum number of client connections that should be accepted.\n

Parameters:\n maxconnections [0- ]: Maximum connections allowed. 0 means no limit.",

"MethodMinEvent": "0",

"MethodMinPeriod": "0",

"MethodName": "setmaxconnections",

"MethodScope": "admin",

"Parameters": [

{

"ParameterName": "lMaxConnections",

"ParameterType": "int",

“ParameterRange”: “1, 50”,

“ParameterRangeMinIncluded”: “1”,

“ParameterRangeMaxIncluded”: “1”,

“ParameterRegularExpValidator”: “”

}

]

},

{

……

……

……

……

3.1.11] Connect to Storage module for logging purposes

This step should be performed automatically when registering the API if RoleDependencies includes the Storage module Role as a dependency (“Role\_StorageManager”)

3.1.12] Send non-critical events logging information to Storage module

When a module wants to log an event, a Request of type LogEvent must be sent to Storage module.

There are five levels of log events:

* CriticalError = 1
* ImportantError = 2
* Warning = 3
* Info = 4
* Everything = 5

Example:

{

“Request”: {

“RequestID”: “AEsdj8jcf88d3fEFefee”,

“RequestInternalID”: “348022847492”,

“RequestDate”: “2018-11-05”,

“RequestTime”: “03:00:00”,

“RequestTimeZone”: “-4”,

“SubscriptionEventCounter”: “0”,

“SubscriptionPeriod”: “0”,

“ResponseMaxSize: “0”,

“RequestMethods”: [

{

"LogEvent": {

"LogEventLevel": “4”,

“LogSourceMessageType”: “Request”

“LogSourceMessageID”: “”

"LogSourceModule": "NULS Connector",

"LogComments”: “Incoming connection accepted from IP: 190.190.1.1!”

}

}

]

}

}

3.1.13] Establish and negotiate WebSocket connections to other modules as required using the connection information returned from the Manager module

After registering the API, connection information from modules that offer the API Roles needed is received. (See section 4.2 for further details). The module should establish connections with these other modules.