|  |
| --- |
| 软件工程实验I组 |
| 软件需求规格说明书 |
| 对Node-RED工具的消息聚集和图形可视化拓展 |

|  |
| --- |
| 暴明坤、夏欣怡、胡俊涛、张雨濛、叶柏威、刘子渊  2020-3-25 |

版本变更历史

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 编制人 | 修改说明 | 审核人 | 版本说明 |
| V2.0.1 | 2020/4/1 | 刘子渊 | 修改用户部分内容 | 张雨濛 | 二稿 |
| V2.0.0 | 2020/4/1 | 胡俊涛 | 细化项目功能需求部分 | 刘子渊 | 二稿 |
| V1.1.0 | 2020/3/26 | 刘子渊 | 增加钉钉部分需求说明 | 暴明坤 | 一稿 |
| V1.0.0 | 2020/3/25 | 项目组全体成员 | 扩充项目需求规格说明书内容 | 暴明坤 | 一稿 |
| V0.1.0 | 2020/3/22 | 张雨濛 | 搭建需求规格说明书框架 | 项目组全体成员 | 初稿 |

目录

1 引言 4

1.1 目的 4

1.2 软件需求分析作用及目标 4

1.2.1 软件需求分析作用 4

1.2.2 软件需求分析目标 4

1.3 项目背景 5

1.4 项目目标 5

1.5 术语和缩略词 6

1.6 文档概述 7

2 总体概述 7

2.1 项目流程 7

2.2 产品功能 7

2.3 工作重点 7

2.4 用户类及其特点 8

2.4.1 用户 8

2.4.2 二次开发者 8

2.4.3 npm仓库 8

2.5 前提和约束 9

3 需求分析 9

3.1 业务需求 9

3.1.1 实现Node-RED本土化拓展 9

3.1.2 实现Node-RED可视化拓展 9

3.2 功能需求 10

3.2.1 创建流程 11

3.2.2 安装拓展包 12

3.2.3 部署流程 13

3.2.4 触发事件 14

3.2.5 监控流程 15

3.2.6 开发拓展包 16

3.2.7 发布拓展包 17

3.3 拓展需求 18

3.3.1 创建飞书配置节点 19

3.3.2 创建飞书获得群列表节点 20

3.3.3 创建飞书获得群信息节点 21

3.3.4 创建飞书获取文件节点 22

3.3.5 创建飞书获得接收消息节点 22

3.3.6 创建飞书获得上传图片节点 23

3.3.7 创建飞书获取图片节点 24

3.3.8 创建飞书目标节点 25

3.3.9 创建飞书添加目标节点 25

3.3.10 创建飞书删除目标节点 26

3.3.11 创建飞书发送消息节点 27

3.3.12 创建钉钉配置节点 28

3.3.13 创建钉钉获得上传图片节点 29

3.3.14 创建钉钉发送消息节点 29

3.3.15 创建钉钉获得接收消息节点 30

3.3.16 创建并配置图表渲染节点 31

3.3.17 创建并配置富文本节点 32

3.4 非功能性需求 33

3.4.1 兼容性 33

3.4.2 高效性 33

3.4.3 可靠性 34

3.4.4 拓展性 34

3.4.5 易用性 34

3.5 软件故障需求 35

3.5.1 需求故障 35

3.5.2 输入/输出故障 35

3.5.3 接口故障 35

3.5.4 数据故障 35

3.6 依从标准 35

4 运行要求 36

4.1 硬件要求 36

4.2 软件要求 36

5 参考资料 36

# 引言

## 目的

**本需求文档的编写目的是为了协调组内成员开展后期的工作，以指导后期的开发、测试等工作。通过此文档，确保规范化的软件开发流程，使之成为后期工作的指导性文件。**

## **软件需求分析作用及目标**

* + 1. 软件需求分析作用

软件需求分析是通过研究用户的需求，对软件预期实现的业务目标进行规划，确认软件功能，建立确认和验证的基本依据。

作为一个项目的开端，软件需求分析是项目实施的关键节点。现实情况是，大量存在不完整性、不正确性的软件都是源自于不恰当的软件需求分析，而需求分析对于软件的功能完整性尤为重要。软件需求分析的正确与完整，是软件达到预期的前提。

* + 1. 软件需求分析目标

软件需求分析有如下目标：

1) 全面描述软件的预期实现功能，用以帮助用户判断所实现功能的正确性、一致性、完整性；

2) 了解并正确地描述软件实现所需要的全部信息，作为软件设计、确认、验证的基准；

3) 为项目管理人员后续的成本规划和人力分配提供依据；

软件需求分析应当尽可能提供软件实现所需要的信息，减少软件开发人员、软件测试人员和需求方的接触，因此软件需求分析要达到正确、一致、完整、可检验、可追踪和可修改。

## 项目背景

物联网（Internet of Things，简称IoT）是上个世纪就已经开始发展的互联网趋势，其主要目标是让尽可能多的设备连入互联网，让更多的数据通过网络连接流动，而不是低效的人-人、人-电脑传递。为解决传统命令式编程（Imperative Programming）不能适应物联网开发过程中需求的复杂性和灵活性，IBM公司带来了基于流编程（Flow-Based Programming）范式的事件驱动（event-driven）流程编排工具Node-RED，其将每个数据处理的步骤抽象为节点（node），通过拖拽创建节点、改变节点位置，以及连线表示节点之间的数据流动方向，给用户以方便且动态地编排并部署工作流（flow）的可能，而部署后的工作流则由事件触发执行。而Node-RED工具另一个顺应物联网开发需求的特性是模块化，其通过良好的代码结构设计实现了节点、流程与工具本身的解耦，并集成了可供开发人员共享节点和流程的社区，因此用户可以一键下载并热插拔节点和流程，尽可能地复用他人已经实现的数据处理需求。

目前，Node-RED在国外社区较为繁荣，但一些实用功能并没有国内开发者进行本土化，比如消息聚集拓展包等。消息聚集拓展包可以通过即时通信软件如Telegram、Facebook Messenger、Slack等提供的机器人机制，将流程执行中的信息实时主动地反馈给用户，也可以让用户从移动端发消息给Node-RED作为事件来触发流程的执行，极大地拓展和丰富了Node-RED的使用场景。而国内即时通信软件，如阿里的钉钉，字节跳动的飞书等均未被Node-RED支持。除此以外，在这个愈发强调数据可视化的时代，现有的 Node-RED 插件中缺少通用的内容渲染模块，该模块的缺失使Node-RED在很多情况下使用受限。

## 项目目标

本项目计划完成两个方面的内容，一是借助本土办公通信软件例如飞书或钉钉等，开发Node-RED工具的本土化消息聚集拓展包，方便国内物联网开发人员的使用，填补Node-RED在中国本土化的空白；二则是为满足Node-RED中数据可视化的要求，实现推送消息的图文并茂，开发一个将数据绘制为图表和进行Markdown模板渲染的可视化拓展包。最终通过一些使用场景示例的搭建展示这两个拓展包的实用价值。

## 术语和缩略词

关于本项目中的一些专业术语如下表1所示。

表 1 专业术语表

|  |  |  |
| --- | --- | --- |
| 编号 | 术语/缩略语 | 描述 |
| 1 | JavaScript | JavaScript（简称JS）是一种基于原型编程、解释型或即时编译型的编程语言，支持面向对象、命令式和函数式编程范式，最开始用作开发Web页面的脚本语言，后来也运用到一些非浏览器环境中。 |
| 2 | HTML | 超文本标记语言（HyperText Markup Language）的缩写，HTML文本是由结构嵌套的HTML标签组成的描述性文本，HTML标签可以说明文字、图形、动画、声音、表格、链接等，可以引用其他位置上的资源，是目前用于网页内容编写的唯一语言。 |
| 3 | Node.js | Node.js是基于Chrome V8引擎开发、跨平台的JavaScript运行环境，采用事件驱动、非阻塞式I/O的模型，可以让JavaScript运行在服务器端。 |
| 4 | node | Node-RED流程中的一个节点，它们可以在流程执行时实例化，并接收、处理和发送消息。编码上node由一组定义节点功能的JavaScript文件和用于节点配置编辑页面及帮助文本页面的HTML文件构成。 |
| 5 | flow | Node-RED中的流程，一个流程由多个node组成，node之间有数据流的依赖关系，流程的起始节点一般为事件监听相关node。流程部署后，由事件（例如手动、定时器、HTTP请求等）触发流程的执行 |
| 6 | Markdown | Markdown是一种轻量级标记语言，它允许人们使用易读易写的纯文本格式编写文档，然后转换成有效的XHTML（或者HTML）文档。 |

## 文档概述

文档用途：本文档主要介绍基于Node-RED的各模块需求分析和规格说明。

主要包含以下内容：

1) 软件需求说明书概述，主要包含本需求说明书的背景、目的和缩略语等；

2) 项目的整体描述，主要包含项目流程、产品功能和工作重点等；

3) 产品的业务需求、功能需求、扩展需求和非功能性需求；

4) 与此项目实现相关的运行环境的要求。

# 总体概述

## 项目流程

**用户进入Node-RED开发环境后，在编辑器中拖放编辑各种Node-RED节点，配置节点并连接成流，最终部署。完成Node-RED的开发。**

**这些Node-RED节点中，就包括本项目将要开发的Node-RED拓展节点。**

## **产品功能**

本项目将基于Node-RED平台，在该平台基础上拓展消息聚集和图形可视化功能。其中消息聚集部分是打通Node-RED平台与本土办公通信软件，如飞书和钉钉等，使得用户可以以低代码方式使用通信软件API。图形可视化部分提供将数据绘制为图表和进行Markdown模板渲染的可视化的功能。

## **工作重点**

基于本土办公软件开发消息聚集拓展包，同时开发用来可视化数据的图表绘制拓展包和基于Markdown的模板渲染拓展包。

## **用户类及其特点**

系统中有三类参与者，用户、二次开发者和npm仓库，其中同一个自然人可能同时具有用户和二次开发者的角色。

* + 1. 用户

用户即使用Node-RED构建物联网应用的人员，其主要行为有创建流程、安装拓展包、部署流程、触发事件和监控流程，触发事件和监控流程用例拓展了部署流程用例。在Node-RED中一个流程就是一个低代码、由众多子应用编排组成的物联网应用，用户通过拖拽交互式网页创建流程，完成后通过部署流程操作使Node-RED后台真实运行流程，监听事件的到来并处理。触发事件是通过网页上的按钮触发流程中某节点从而产生数据激活流程，监控流程则是将后台执行流程的一些信息返回至网页供用户监控。安装拓展包则可以复用其他人对Node-RED平台进行功能拓展的代码，避免了重复开发。

* + 1. 二次开发者

二次开发者即基于 Node-RED 框架进行功能性扩展的使用者。在Node-RED已有节点无法满足需求的情况下，二次开发者会基于Node-RED开发能够拓展其功能的拓展包，包中通常包含功能相关的一组节点。二次开发者有开发拓展包用例和发布拓展包用例，开发拓展包是二次开发者在其本地编写代码实现并测试新功能脚本的过程，发布拓展包则是其为了能够与他人分享劳动成果，将拓展包上传至npm仓库供他人下载的过程。

* + 1. npm仓库

npm仓库的主要作用是托管Node-RED相关的代码，其中既包括Node-RED框架代码，也包括二次开发者基于Node-RED框架开发的扩展功能的代码。用户和二次开发者主要从npm仓库中获取Node-RED框架代码和二次开发者的扩展功能代码，除此以外，二次开发者还可向npm仓库发布扩展功能代码供他人使用。

## **前提和约束**

本项目仅提供相对直观可用的消息聚集功能和图表绘制功能。并不保证拓展包中的节点包括开放平台中所有的API。同时也不保证包含所有的常见图表类型。

本项目中节点间交换的消息应该是常见的Node.js对象，不应使用过于特殊或复杂的对象类型。并尽可能使用可以JSON编码的对象类型。

本项目出于复杂度和Node-RED编程模型的考虑。不保证节点的状态可以持久化保存在磁盘上。用户如有需要可以使用Node-RED中数据库相关的节点，自行进行持久化设计。

# 需求分析

## 业务需求

* + 1. 实现Node-RED本土化拓展

**基于Node-RED平台，以及国内本土办公软件如飞书和钉钉的开发平台官方文档，开发消息聚集拓展包，打通Node-RED在本土使用的壁垒，方便国内物联网开发人员的使用。**

* + 1. 实现Node-RED可视化拓展

**基于Node-RED平台，开发用于可视化数据的图表绘制拓展包和基于Markdown的模板渲染拓展包，填补Node-RED当前在数据可视化方面的空白，帮助快速解读在Node-RED平台中产生的数据。**

## **功能需求**

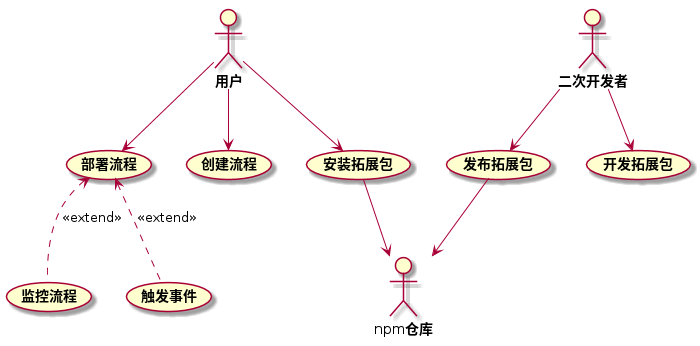


图1 Node-RED用例图

本项目组将在此部分对Node-RED工具的整体功能性需求进行用例建模，用例图如图1所示。

下面给出了Node-RED平台的用例说明表，如表2所示。

表2 Node-RED平台用例说明表

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Use Case ID | Brief Description | Primary Actor |
| 创建流程 | UC001 | 该用例描述用户在Node-RED工具中使用交互式面板创建流程的过程。一个流程由若干个节点组成，节点之间由连线表示数据依赖关系（即一个节点的输出是另外一个节点的输入）。 | 用户 |
| 安装拓展包 | UC002 | 该用例描述用户在Node-RED中安装来自开源社区的拓展包，为工具添加具有更多功能的节点的过程。拓展包以npm package的形式保存在npm的仓库中，包含了若干节点的HTML和JavaScript文件，安装之后节点会出现在网页的节点面板中供用户使用 | 用户 |
| 部署流程 | UC003 | 该用例描述了用户在创建或者修改流程之后，将流程部署到Node-RED服务端环境中的过程。流程在网页端的创建只是对其的定义，还需要部署到服务端环境中以实例化流程以及其中的节点，这时节点才会监听事件发生并触发流程。 | 用户 |
| 触发事件 | UC004 | 该用例描述了用户在部署流程后，通过手动触发流程中的输入节点进而触发流程的执行的过程。流程的输入节点不仅可以是各种来自网络或者系统的事件，也可以由用户手动触发来完成特定操作或者用于测试功能等。 | 用户 |
| 监控流程 | UC005 | 该用例描述了用户在部署流程后，为获取流程被触发执行而进行流程监控的过程。监控流程主要通过debug节点，将debug节点连接在需要查看执行信息的节点之后，即可将前继节点的输出消息反馈给网页端进行查看 | 用户 |
| 开发拓展包 | UC006 | 该用例描述了二次开发人员基于Node-RED工具，使用JavaScript和HTML编程语言开发新功能的拓展节点并测试的过程，功能代码位于Node-RED服务端环境中，功能测试则通过网页端使用节点。 | 二次开发者 |
| 发布拓展包 | UC007 | 该用例描述了二次开发者在完成开发和测试新功能的Node-RED拓展包，为与他人共享上传发布至npm仓库的过程。 | 二次开发者 |

对上述用例的详细描述见表3-表9。

* + 1. 创建流程

该用例的详细描述如表3所示。

表3 创建流程用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC001 | |
| Use Case Name | 创建流程 | |
| Brief Description | 该用例描述用户在Node-RED工具中使用交互式面板创建流程的过程。一个流程由若干个节点组成，节点之间由连线表示数据依赖关系（即一个节点的输出是另外一个节点的输入）。 | |
| Precondition | Node-RED网页端处于运行状态 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户点击菜单栏中的“创建流程”按钮，显示新的流程面板 |
| Step 2 | 用户从节点面板中拖动节点到流程面板以添加新的节点 |
| Step 3 | 用户双击节点，显示节点配置面板，对参数配置进行调整 |
| Step 4 | 重复step 2、step 3若干次之后，用户将所需的所有节点添加并配置完成 |
| Step 5 | 用户将一个节点的输入/输出点通过拖拽连线连接至其他节点的输出/输入点，表示节点之间数据依赖关系 |
| Step 6 | 重复step 5若干次之后，用户完成了所需要的节点功能组织，即完成流程的创建 |
| Postcondition | 流程以JSON数据格式保存在前端的JavaScript程序中 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 安装拓展包

该用例的详细描述如表4所示。

表4 安装拓展包用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC002 | |
| Use Case Name | 安装拓展包 | |
| Brief Description | 该用例描述用户在Node-RED中安装来自开源社区的拓展包，为工具添加具有更多功能的节点的过程。拓展包以npm package的形式保存在npm仓库中，包含了若干节点的HTML和JavaScript文件，安装之后节点会出现在网页的节点面板中供用户使用 | |
| Precondition | Node-RED网页端、服务端处于运行状态 | |
| Primary Actor | 用户 | |
| Secondary Actors | npm仓库 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户点击菜单栏中的“管理拓展包”，打开拓展包面板 |
| Step 2 | Node-RED前端将“读取拓展包列表”请求发送给后端 |
| Step 3 | 后端读取npm仓库中相关的拓展包，将拓展包的元信息返回前端 |
| Step 4 | 前端将拓展包元信息渲染为列表 |
| Step 5 | 用户在拓展包面板上的列表中浏览、筛选并选择需要安装的拓展包 |
| Step 6 | 网页端将安装拓展包命令发送给服务端 |
| Step 7 | 服务端执行npm install命令安装对应的拓展包，并加载其所包含的代码，将新增的节点信息发送给网页端 |
| Step 8 | 网页端收到信息后添加入节点列表，并更新节点面板 |
| Postcondition | 节点面板中显示新增的拓展包中的节点，并且实现相应功能的代码被保存在服务端，当节点被部署时被读取执行 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | Step 1 | 用户登入Node-RED服务端运行的终端环境中 |
| Step 2 | 用户切换目录至$HOME/.node-red |
| Step 3 | 用户执行npm install加所要安装的拓展包名称，下载包至当前目录 |
| Step 4 | 服务端监听到目录文件的变化，加载新安装拓展包的代码，将新增的节点信息发送给网页端 |
| Step 5 | 网页端收到信息后添加入节点列表，并更新节点面板 |
| Postcondition | 节点面板中显示新增的拓展包中的节点，并且实现相应功能的代码被保存在服务端，当节点被部署时被读取执行 |
| Bounded Alternative Flows | 无 | |

* + 1. 部署流程

该用例的详细描述如表5所示。

表5 部署流程用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC003 | |
| Use Case Name | 部署流程 | |
| Brief Description | 该用例描述了用户在创建或者修改流程之后，将流程部署到Node-RED服务端环境中的过程。流程在网页端的创建只是对其的定义，还需要部署到服务端环境中以实例化流程以及其中的节点，这时节点才会监听事件发生并触发流程。 | |
| Precondition | Node-RED网页端、服务端处于运行状态，并且当前流程为新创建或经过更改 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | extend：触发事件、监控流程 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户点击页面顶端的“部署”按钮 |
| Step 2 | 网页端将当前面板中的流程以JSON数据格式发送到Node-RED服务端 |
| Step 3 | 服务端创建实现每个节点功能的程序对象，并根据JSON中的数据依赖配置节点之间的传输 |
| Postcondition | 流程中的输入节点处于监听事件（例如HTTP连接、手动触发、定时事件等）状态，当有事件到来时将其传递给后续连接的节点来触发流程的执行 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 触发事件

该用例的详细描述如表6所示。

表6 触发事件用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC004 | |
| Use Case Name | 触发事件 | |
| Brief Description | 该用例描述了用户在部署流程后，通过手动触发流程中的输入节点进而触发流程的执行的过程。流程的输入节点不仅可以是各种来自网络或者系统的事件，也可以由用户手动触发来完成特定操作或者用于测试功能等。 | |
| Precondition | Node-RED网页端、服务端处于运行状态，流程已经在服务端部署 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户点击inject节点的触发按钮 |
| Step 2 | 网页端将该节点的触发事件发送给服务端 |
| Step 3 | 服务端收到消息后将inject节点中预先配置的消息体作为数据传递给后续连接的节点触发流程执行 |
| Postcondition | inject节点的后续节点收到预先配置的消息体进行处理 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 监控流程

该用例的详细描述如表7所示。

表7 监控流程用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC005 | |
| Use Case Name | 监控流程 | |
| Brief Description | 该用例描述了用户在部署流程后，为获取流程被触发执行而进行流程监控的过程。监控流程主要通过debug节点，将debug节点连接在需要查看执行信息的节点之后，即可将前继节点的输出消息反馈给网页端进行查看 | |
| Precondition | Node-RED网页端、服务端处于运行状态，流程已经在服务端部署 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开监控面板 |
| Step 2 | 网页端与服务端建立WebSocket连接 |
| Step 3 | 当服务端部署的流程被触发执行时，其中的debug节点将其输入数据经过JSON序列化后通过WebSocket传输给网页端 |
| Step 4 | 网页端将数据JSON反序列化后显示在监控面板中 |
| Postcondition | 无 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 开发拓展包

该用例的详细描述如表8所示。

表8 开发拓展包用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC006 | |
| Use Case Name | 开发拓展包 | |
| Brief Description | 该用例描述了二次开发人员基于Node-RED工具，使用JavaScript和HTML编程语言开发新功能的拓展节点并测试的过程，功能代码位于Node-RED服务端环境中，功能测试则通过网页端使用节点。 | |
| Precondition | Node-RED网页端、服务端处于运行状态 | |
| Primary Actor | 二次开发者 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 二次开发者在Node-RED服务端环境中以Node-RED SDK的规范编写功能拓展的相关代码 |
| Step 2 | 二次开发者切换工作目录到Node-RED用户目录（通常为~/.node-red） |
| Step 3 | 二次开发者执行npm install加拓展功能代码所在的位置安装本地拓展包 |
| Step 4 | Node-RED服务端监听到目录变动，读取新增包的元信息，并将更新后的节点列表发送给网页端 |
| Step 5 | 二次开发者进入网页端查看新增节点，并测试验证其功能 |
| Postcondition | 无 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 发布拓展包

该用例的详细描述如表9所示。

表9 发布拓展包用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC007 | |
| Use Case Name | 发布拓展包 | |
| Brief Description | 该用例描述了二次开发者在完成开发和测试新功能的Node-RED拓展包，为与他人共享上传发布至npm仓库的过程。 | |
| Precondition | Node-RED网页端、服务端处于运行状态 | |
| Primary Actor | 二次开发者 | |
| Secondary Actors | npm仓库 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 二次开发者修改拓展包代码中package.json配置文件以符合Node-RED规范，使得其他服务端在检索npm仓库时能够识别为Node-RED相关 |
| Step 2 | 二次开发者执行npm login登入自己的npm账户 |
| Step 3 | 二次开发者执行npm publish发布拓展包，程序将自动将代码打包并上传至npm仓库 |
| Postcondition | npm仓库中有该拓展包，并且可由其他用户通过下载安装 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

## **拓展需求**

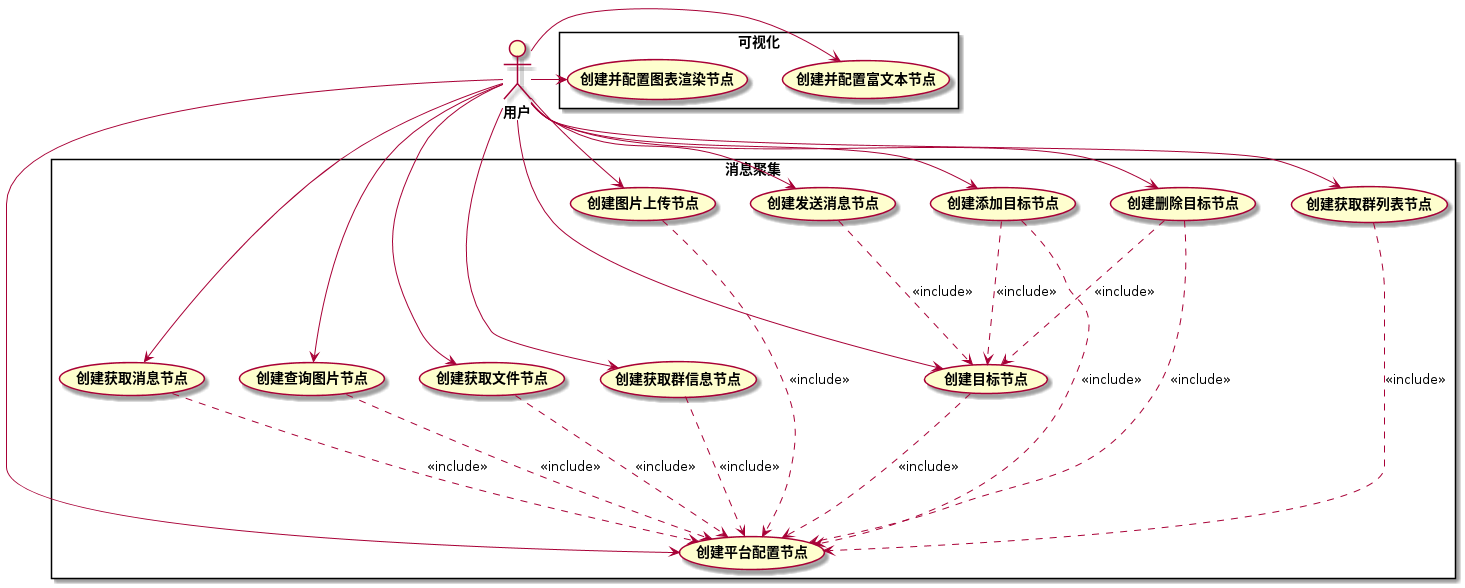


图2 Node-RED拓展用例图

Node-RED系统虽然功能强大，但是由于Node-RED最开始在IBM公司的主导下开发，所以其功能拓展多基于国外的各种社交软件和开放平台，并不适合在国内开发应用。所以为了解决这个问题，本项目计划基于Node-RED开发了基于工作软件API的消息聚集拓展包。同时考虑到在工作软件上进行展示的必要，我们还开发了各种图表的可视化应用以及基于markdown语法的模板拓展包。

本项目组在此部分将对Node-RED工具的拓展需求进行用例建模，用例图如图2所示。下面给出了拓展需求中的用例说明表，如表10所示。

表10 Node-RED平台拓展模块用例说明表

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Use Case ID | Brief Description | Primary Actor |
| 创建飞书配置节点 | UC008 | 该用例描述了用户在编辑器中添加飞书配置节点并进行配置。 | 用户 |
| 创建飞书获得群列表节点 | UC009 | 该用例描述了用户在编辑器中添加飞书获取群列表节点并进行配置。 | 用户 |
| 创建飞书获得群信息节点 | UC010 | 该用例描述了用户在编辑器中添加飞书获取群信息节点并进行配置。 | 用户 |
| 创建飞书获取文件节点 | UC011 | 该用例描述了用户在编辑器中添加飞书获取文件节点并进行配置。 | 用户 |
| 创建飞书获得接收消息节点 | UC012 | 该用例描述了用户在编辑器中添加飞书接收信息节点并进行配置。 | 用户 |
| 创建飞书获得上传图片节点 | UC013 | 该用例描述了用户在编辑器中添加飞书上传图片节点并进行配置。 | 用户 |
| 创建飞书获取图片节点 | UC014 | 该用例描述了用户在编辑器中添加飞书获取图片节点并进行配置。 | 用户 |
| 创建飞书目标节点 | UC015 | 该用例描述了用户在编辑器中添加飞书目标节点并进行配置。 | 用户 |
| 创建飞书添加目标节点 | UC016 | 该用例描述了用户在编辑器中添加飞书添加目标节点并进行配置。 | 用户 |
| 创建飞书删除目标节点 | UC017 | 该用例描述了用户在编辑器中添加飞书删除目标节点并进行配置。 | 用户 |
| 创建飞书发送消息节点 | UC018 | 该用例描述了用户在编辑器中添加飞书发送消息节点并进行配置。 | 用户 |
| 创建钉钉配置节点 | UC019 | 该用例描述了用户在编辑器中添加钉钉配置节点并进行配置。 | 用户 |
| 创建钉钉上传图片节点 | UC020 | 该用例描述了用户在编辑器中添加钉钉上传图片节点并进行配置。 | 用户 |
| 创建钉钉发送消息节点 | UC021 | 该用例描述了用户在编辑器中添加钉钉发送消息节点并进行配置。 | 用户 |
| 创建钉钉接收消息节点 | UC022 | 该用例描述了用户在编辑器中添加钉钉接收信息节点并进行配置。 | 用户 |
| 创建并配置图标渲染节点 | UC023 | 该用例描述了用户在编辑器中添加图表渲染节点并进行配置。 | 用户 |
| 创建并配置富文本节点 | UC024 | 该用例描述了用户在编辑器中添加富文本节点并进行配置。 | 用户 |

对上述用例的详细描述见表11-表27。

* + 1. 创建飞书配置节点

该用例的详细描述如表11所示。

表11 创建飞书配置节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC008 | |
| Use Case Name | 创建飞书配置节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书配置节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中拖放飞书配置节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置飞书配置节点的AppID 和 AppSecret |
| Postcondition | 当节点被部署时，连接飞书API并定时刷新tenant token |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建飞书获得群列表节点

该用例的详细描述如表12所示。

表12 创建飞书获得群列表节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC009 | |
| Use Case Name | 创建飞书获得群列表节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书获取群列表节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了飞书配置节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中拖放飞书获取群列表节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置获取群列表节点,选择使用的飞书配置节点 |
| Postcondition | 当节点被部署时，定时获取群列表或再有输入时获取群列表，并输出 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建飞书获得群信息节点

该用例的详细描述如表13所示。

表13 创建飞书获得群信息节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC010 | |
| Use Case Name | 创建飞书获得群信息节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书获取群信息节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了飞书配置节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中拖放飞书获取群信息节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置获取群列表节点,选择使用的飞书配置节点 |
| Postcondition | 当节点被部署时，根据输入的群ID查找群信息，并输出 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建飞书获取文件节点

该用例的详细描述如表14所示。

表14 创建飞书获取文件节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC011 | |
| Use Case Name | 创建飞书获取文件节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书获取文件节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了飞书配置节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中拖放飞书获取文件节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置获取文件节点,选择使用的飞书配置节点 |
| Postcondition | 当节点被部署时，根据输入的文件key查询文件内容，并输出 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建飞书获得接收消息节点

该用例的详细描述如表15所示。

表15 创建飞书获得接收消息节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC012 | |
| Use Case Name | 创建飞书获得接收消息节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书接收信息节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了飞书配置节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中拖放飞书接收消息节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置接收消息节点,选择使用的飞书配置节点 |
| Postcondition | 当节点被部署时，利用HTTPS协议从飞书平台接收消息，输出到流中 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建飞书获得上传图片节点

该用例的详细描述如表16所示。

表16 创建飞书获得上传图片节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC013 | |
| Use Case Name | 创建飞书获得上传图片节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书上传图片节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了飞书配置节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中拖放飞书上传图片节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置飞书上传图片节点,选择使用的飞书配置节点 |
| Postcondition | 当节点被部署时，根据输入的图片信息，上传图片并返回图片key，并输出 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建飞书获取图片节点

该用例的详细描述如表17所示。

表17 创建飞书获取图片节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC014 | |
| Use Case Name | 创建飞书获取图片节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书获取图片节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了飞书配置节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中拖放飞书获取图片节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置获取图片节点,选择使用的飞书配置节点 |
| Postcondition | 当节点被部署时，根据输入的图片key查询图片内容，并输出 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建飞书目标节点

该用例的详细描述如表18所示。

表18 创建飞书目标节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC015 | |
| Use Case Name | 创建飞书目标节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书目标节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了飞书配置节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从侧边栏中添加飞书目标节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置飞书目标节点,选择使用的飞书配置节点 |
| Postcondition | 当节点被部署时，储存并维护一组消息发送目标 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建飞书添加目标节点

该用例的详细描述如表19所示。

表19 创建飞书添加目标节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC016 | |
| Use Case Name | 创建飞书添加目标节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书添加目标节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了飞书配置节点和飞书目标节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户在编辑器中拖放飞书添加目标节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置飞书添加目标节点,选择使用的飞书配置节点和飞书目标节点 |
| Postcondition | 当节点被部署时，当输入发送目标时，将目标添加到飞书目标节点中 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建飞书删除目标节点

该用例的详细描述如表20所示。

表20 创建飞书删除目标节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC017 | |
| Use Case Name | 创建飞书删除目标节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书删除目标节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了飞书配置节点，飞书目标节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户在编辑器中拖放中添加飞书删除目标节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置飞书目标节点,选择使用的飞书配置节点和飞书目标节点 |
| Postcondition | 当节点被部署时，当输入目标时，从目标节点中删除目标 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建飞书发送消息节点

该用例的详细描述如表21所示。

表21 创建飞书发送消息节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC018 | |
| Use Case Name | 创建飞书发送消息节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加飞书发送消息节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了飞书配置节点，添加了飞书目标节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中添加飞书发送消息节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置飞书目标节点,选择使用的飞书配置节点，飞书目标节点 |
| Postcondition | 当节点被部署时，将输入发送到目标节点维护的目标。 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建钉钉配置节点

该用例的详细描述如表22所示。

表22 创建钉钉配置节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC019 | |
| Use Case Name | 创建钉钉配置节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加钉钉配置节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中拖放钉钉配置节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置钉钉配置节点的 appKey 和 appSecret |
| Postcondition | 当节点被部署时，连接钉钉 API 并定时刷新 access\_token |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建钉钉获得上传图片节点

该用例的详细描述如表23所示。

表23 创建钉钉获得上传图片节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC020 | |
| Use Case Name | 创建钉钉获得上传图片节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加钉钉上传图片节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了钉钉配置节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中拖放钉钉上传图片节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置钉钉上传图片节点,选择使用的钉钉配置节点 |
| Postcondition | 当节点被部署时，根据输入的图片信息，上传图片并返回图片的media\_id，并输出 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建钉钉发送消息节点

该用例的详细描述如表24所示。

表24 创建钉钉发送消息节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC021 | |
| Use Case Name | 创建钉钉发送消息节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加钉钉发送消息节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了钉钉配置节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中添加钉钉发送消息节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置钉钉发送消息节点，选择使用的钉钉配置节点 |
| Postcondition | 当节点被部署时，将输入通过配置好的钉钉群机器人发送至钉钉群中。 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建钉钉获得接收消息节点

该用例的详细描述如表25所示。

表25 创建钉钉获得接收消息节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC022 | |
| Use Case Name | 创建钉钉获得接收消息节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加钉钉接收信息节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。用户添加了钉钉配置节点 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中拖放钉钉接收消息节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置接收消息节点,选择使用的钉钉配置节点 |
| Postcondition | 当节点被部署时，利用HTTPS协议从钉钉平台接收消息，输出到流中 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建并配置图表渲染节点

该用例的详细描述如表26所示。

表26 创建并配置图表渲染节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC023 | |
| Use Case Name | 创建并配置图表渲染节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加图表渲染节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中添加图表渲染节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置图表渲染节点的 Label 和 Type，根据不同的Type需要做不同的配置 |
| Step 5 | 如果用户选择 Type 值为 Line chart，则需要对 X-axis，Y-axis 以及 Blank label 进行配置。其中 X-axis 可选最近一段时间或是最近 n 个采样点，Y-axis可选图表展现的最小Y值以及最大Y值，Blank label 为占位标签。 |
| Replace Step 5 | 如果用户选择 Type 值为 Bar chart，则需要对 Y-axis 以及 Blank label 进行配置，Y-axis可选图表展现的最小Y值以及最大Y值，Blank label 为占位标签。 |
| Replace Step 5 | 如果用户选择 Type 值为 Bar chart (H)，则需要对 X-axis 以及 Blank label 进行配置，X-axis可选图表展现的最小X值以及最大X值，Blank label 为占位标签。 |
| Replace Step 5 | 如果用户选择 Type 值为 Pie chart，则需要对 Cutout 以及 Blank label 进行配置，Cutout 为内部切口的百分比，Blank label 为占位标签。 |
| Replace Step 5 | 如果用户选择 Type 值为 Polar area chart，则需要对 Y-axis 以及 Blank label 进行配置，Y-axis可选图表展现的最小Y值以及最大Y值，Blank label 为占位标签。 |
| Replace Step 5 | 如果用户选择 Type 值为 Radar chart，则需要对 Y-axis 以及 Blank label 进行配置，Y-axis可选图表展现的最小Y值以及最大Y值，Blank label 为占位标签。 |
| Postcondition | 当节点被部署时，节点将输出渲染得到图表经过base64编码后的字符串。 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

* + 1. 创建并配置富文本节点

该用例的详细描述如表27所示。

表27 创建并配置富文本节点用例描述

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC024 | |
| Use Case Name | 创建并配置富文本节点 | |
| Brief Description | 该用例描述了用户在编辑器中添加富文本节点并进行配置。 | |
| Precondition | Node-RED网页端、服务端处于运行状态。 | |
| Primary Actor | 用户 | |
| Secondary Actors | 无 | |
| Dependency | 无 | |
| Generalization | 无 | |
| Basic Flow | Step 1 | 用户打开编辑器 |
| Step 2 | 用户从编辑器中添加富文本节点 |
| Step 3 | 用户打开配置页面 |
| Step 4 | 用户配置富文本节点，以 Markdown 的格式配置富文本模板，模板中需要特定占位符预留文字以及图片位置，这些预留位置将会被富文本节点的输入填充。同时还需配置节点输出时间间隔，富文本节点将以定义的时间间隔输出填充好的富文本。 |
| Postcondition | 当节点被部署时，节点将按一定时间间隔输出填充好的富文本。 |
| Specific Alternative Flows | 无 | |
| Global Alternative Flows | 无 | |
| Bounded Alternative Flows | 无 | |

## **非功能性需求**

* + 1. 兼容性

Node-RED由Node.js构建，开发者在使用Node.js时，不需要考虑操作系统的差别，故应当支持在Linux，macOS等常用主流平台上进行部署运行。同时，Node-RED基于浏览器的流编辑器应当兼容Chrome，Firefox等主流浏览器的新版本。

* + 1. 高效性

（1）轻量级系统

Node-RED是一个基于Node.js的轻量级框架，充分利用了其事件驱动的非阻塞模型，这使得它非常适合在低成本Raspberry Pi等硬件上的网络边缘以及云中运行。

（2）支持并发运行

由于Node.js的特性，Node-RED可以在语言层面很自然地进行并行I/O的操作，每个调用无需等待之前的I/O调用结束，从而极大提升效率。

（3）低响应时间

Node-RED需要实时推送数据流，因而对数据实时性要求比较高。在网络状态良好的情况下，响应时间应当不超过30秒；网络状态不佳的情况下，响应时间应当不超过5分钟。

* + 1. 可靠性

系统应该在异常和危险情况下都能保证健壮的表现和稳定的性能，有健全的容错机制和方法，应该能处理系统运行过程中出现的各种异常情况，如人为操作错误、输入非法数据、硬件设备失败等，系统应该能正确的处理，恰当的回避。例如当前继节点的输出与后继节点的输入没有相匹配时，应当告知用户输入非法，而不是系统崩溃。

当程序运行出现不可避免的错误时，系统应告知发生了什么错误，而不是直接把这个错误的上下文丢掉，或者伴随着错误而退出程序。因软件系统的失效而造成不能完成业务的概率要小于5‰。

* + 1. 拓展性

Node-RED设计为模块式节点软件包，节点与节点之间没有耦合，开发者可以很方便地对节点进行拓展。目前软件包存储库中有225000个模块，可以轻松扩展面板节点，添加新功能。

各个节点的输入输出应当明确，结构相对简单明晰。同时应保证有良好的文档说明，保证以后其他节点能够处理这些节点的输入输出，保证可拓展性。

* + 1. 易用性

Node-RED应当具备快速部署能力，其提供了Docker镜像版本，只需要启动镜像就可以启动基于浏览器的流编辑器服务。

用户根据该基于浏览器的流编辑器，只需要点击按钮就可以将第三方节点软件包导入，通过拖拽的形式对流程进行编辑。节点的配置项支持各种下拉选项，增强了节点的易用性，用户友好程度Max。

## **软件故障需求**

* + 1. 需求故障

软件需求故障是指不合理不正确的需求或是需求临时变更，导致需求分析文档不完整或文档有误，功能性需求欠缺或规定有误。

当出现上述故障时，本项目组成员应该组织紧急会议，讨论需求变更的问题，重新对功能需求进行分析判断，更新需求文档，并及时记录问题，进行项目管理，以确保软件开发过程正常进行。

* + 1. 输入/输出故障

输入故障是指出现了不能接受的输入，参数有误或遗漏等。

输出故障是指输出格式有误，在错误的时间产生了正确的结果，输出了不合逻辑的结果等等。

本软件应该对输入、输出进行限制，当出现了错误的输入或是输出，开发人员应该能够正确分析该故障，并给出合理的处理方法，完善后台逻辑。

* + 1. 接口故障

接口故障是指不正确的中断处理，I/O时序出错，调用了错误的进程等。

应当严格按照运行需求进行软件部署，保证软件正常运行。当出现接口故障时，开发人员应该能根据错误类型正确调整运行环境，确保软件正常运行。

* + 1. 数据故障

数据故障是指不正确的初始化，不正确的存储访问，数据存储格式出错或是数据丢失等。

当出现数据故障时，软件应当产生对应的错误提示，并停止软件故障测试过程，开发人员需要分析故障原因，及时进行维护。

## **依从标准**

本项目将以下面这些标准作为参考：

1. GB/T 13702-1992 计算机软件分类与代码

2. GB/T 19003-2008 软件工程

3. GB/T 5538-1995 软件工程标准分类法

4. GB/T 9385-2008 计算机软件需求规格说明

5. GB/T 5532-2008 计算机软件测试规范

6. GB/T 18221-2000 信息技术程序设计语言

7. GB/T 8567-2006 计算机软件文档编制规范

# 运行要求

## 硬件要求

**amd64平台或aarch64平台。**

## **软件要求**

1. Node.js >= 12.16；
2. Node-RED >= 1.0.0；
3. 主流linux操作系统或MacOS。

# 参考资料

1. **Node-RED 官方网站：https://nodered.org/**
2. **Node-RED 官方文档：https://nodered.org/docs/**
3. **Node.js 官方文档：https://nodejs.org/en/**
4. **钉钉开发平台官方文档 https://ding-doc.dingtalk.com/doc#/serverapi3/rg1occ**
5. **飞书开放平台官方文档：https://open.feishu.cn/document/ukTMukTMukTM/uITNz4iM1MjLyUzM**
6. **Roger S.Pressman. 软件工程——实践者的研究方法[M]. 机械工业出版社, 2011.**
7. **Leszek A.Maciaszek. 需求分析与系统设计[M].机械工业出版社, 2009.**
8. **Nicholas C.Zakas. JavaScript高级程序设计[M].人民邮电出版社, 2012.**
9. **Mike Cantelon等. Node.js实战[M]. 人民邮电出版社, 2014.**