

UniEAP Platform ESB 组件用户手册

版权声明

《UniEAP Platform ESB 组件用户手册》的版权归东软集团（大连）有限公司所有。未经东软集团（大连）有限公司的书面准许，不得将本手册的任何部分以任何形式、采用任何手段（电子的或机械的，包括照相复制或录制）、或为任何目的，进行复制或扩散。

Copyright© 1997-2014 东软集团（大连）有限公司。版权所有，翻制必究。

Neusoft东软®是东软软件股份有限公司的注册。

更改履历

版本号	更改时间	更改的图表和 章节号	状态	更改简要描述	更改申 请编 号	更改人	批准 人
1.0	2014-10-9		N			王朝辉	
1.0.5	2014-11-17		M			于明光	
1.0.5	2014-11-17		M			陆国际	

注：状态可以为 N-新建、A-增加、M-更改、D-删除

目录

手册的结构	6
手册约定	6
第 1 章 ESB 组件简介	7
第 2 章 部署规划	7
2.1 部署架构	7
2.1.1 单实例部署架构	7
2.1.2 集群部署架构	8
2.2 推荐配置	8
2.3 环境部署及配置	8
2.3.1 单机环境的部署	8
2.3.2 集群环境的配置	9
第 3 章 ESB 管理控制台使用说明	11
3.1 服务管理	12
3.1.1 服务注册	12
3.1.2 服务维护	16
3.1.3 服务查看	17
3.2 集群管理	17
3.2.1 节点联通性状态查看	17
3.2.2 SOAP 代理运行状态管理	19
第 4 章 服务调用	22
4.1 服务调用工具配置说明	22
4.2 服务调用示例	23

第 5 章 服务监控.....	26
5.1 ESB 监控服务配置	27
5.2 查看服务总线监控散点分布图.....	32
5.3 查看散点分布图详细信息.....	34
5.4 多应用服务监控配置	37
5.5 服务提供端服务监控配置.....	38
5.6 服务调用端服务监控配置.....	45
5.7 监控数据分析	50
5.7.1 正常监控记录.....	50
5.7.2 服务异常监控记录	53
5.7.3 总线异常监控记录	57
第 6 章 日常维护.....	60
6.1 UniEAP ESB 启动.....	60
6.2 UniEAP ESB 停机.....	61
附录 A UniEAP Platform 服务发布实践	61
A1. 发布 SOAP 服务	61
A2. 发布 REST 服务.....	62
附录 B UniEAP Platform 服务调用实践	62
B1. 直接调用 SOAP 服务	63
B2. REST 服务直接调用	65

手册的结构

手册分主要内容包括五章：

第一章 ESB 组件简介。介绍了 ESB 组件相关基本概念。

第二章 部署规划。介绍了 ESB 组件的基本部署与配置方法。

第三章 ESB 管理控制台使用说明。介绍了 ESB 组件的基本操作方法。

第四章 服务调用。介绍了如何通过 ESB 组件进行服务调用。

第五章 服务监控。介绍了如何监控 ESB 组件上的服务调用。

第六章 日常维护。介绍了 ESB 组件的启动、停机等日常运维操作方法。

手册约定

阅读本手册时，您可能会看到如下标记，其含义如下：

【提示】：表示某些信息，读者可以参考。

【技巧】：表示读者可以参考其中介绍的技巧，提高效率。

【注意】：提醒读者关注某些事项，这些事项对您后续操作造成影响。

【警告】：请读者千万注意某些事项，否则可能会造成严重错误。

面向的受众

在阅读本手册前请务必先阅读 UniEAP V4.5 用户手册的第二章 UniEAP WorkShop 的安装与配置。其次，本手册面向有一定 UniEAP WorkShop 开发经验的开发人员。

第1章 ESB 组件简介

UniEAP Platform ESB 组件是面向系统集成领域，提供系统服务注册、管理及监控功能的应用组件。下文中如未作特殊说明，均以“ESB 组件”代称“UniEAP Platform ESB 组件”。

ESB 组件提供图形化管理控制台和 API 接口用以统一服务注册管理，聚合的、统一的管理视图，解决了服务的分散管理状况。ESB 组件向服务消费者提供统一的调用入口，解决了服务消费者与服务提供者的物理地址绑定和分散耦合关系，同时也更方便了服务的统一管控与治理。在庞大、复杂的 IT 系统环境中，ESB 组件将显著提升系统间交互与集成的工作成效。

第2章 部署规划

2.1 部署架构

ESB 组件的部署形态为 Java Web 应用。故其部署需求基本类同普通 Java Web 应用。部署形式基本分为两类：单实例部署和集群部署。

2.1.1 单实例部署架构

须有一台 Java Web 应用服务器运行 ESB 组件，要求网络环境保证服务相关提供方与调用方可与之通信。如图 2.1.1.1 所示：



图 2.1.1.1 单实例部署架构

2.1.2 集群部署架构

须有多台 Java Web 应用服务器运行 ESB 组件，要求网络环境保证服务相关提供方与调用方可与之通信，且部署 ESB 组件的各应用服务器间可相互通信。如图 2.1.2.1 所示：

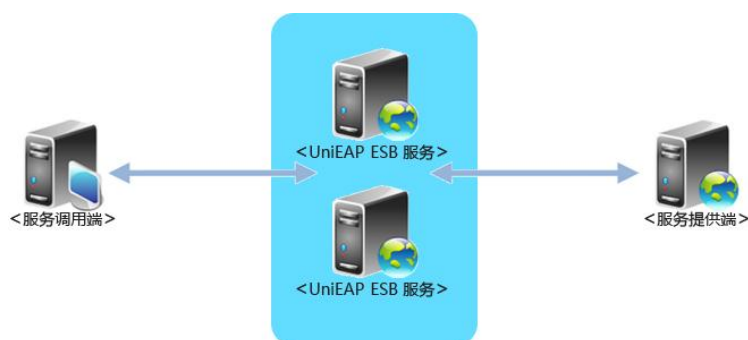


图 2.1.2.1 集群部署架构

2.2 推荐配置

项目名称	推荐配置
ESB 应用服务器	（免费开源）Tomcat （商业产品）Weblogic

2.3 环境部署及配置

2.3.1 单机环境的部署

前提准备：

- Java Web 应用服务器：用于部署运行 UniEAP ESB 组件。

步骤 1：通过服务支持人员获取资料如下：

应用部署包：“UniEAP ESB.zip”。

【备注】：以上所列资料文件名称有可能发生变动，以具体交付时为准，但资料数量与对应意义应该是一致的。

步骤 2：解压资料，解压后得到一个标准的 Java Web 应用目录。如图 2.3.1.1 所示

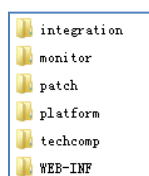


图 2.3.1.1 目录结构

步骤 3：将解压后的工程部署到“前提准备”中准备好的 WEB 服务器中。

步骤 4：部署完毕后，访问应用服务器提供的地址即可。

2.3.2 集群环境的配置

前提准备：

- 多个 Java Web 应用服务器（数量 ≥ 2 ）：每个应用服务器都将用于部署运行一个 UniEAP ESB 组件实例。如此，多个实例共同构成工作集群。该处具体情形应照实际选用的应用服务器的使用说明进行。

步骤 1：通过服务支持人员获取资料如下：

应用部署包：“UniEAP ESB.zip”。

【备注】：以上所列资料文件名称有可能发生变动，以具体交付时为准，但资料数量与对应意义应该是一致的。

步骤 2：解压资料，解压后得到一个标准的 Java Web 应用目录。如图 2.3.2.1 所示。

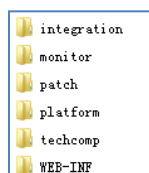


图 2.3.2.1 目录结构

步骤 3: 在解压后的应用目录中，找到 applicationContext-cluster.xml 配置文件。路径如下图所示：

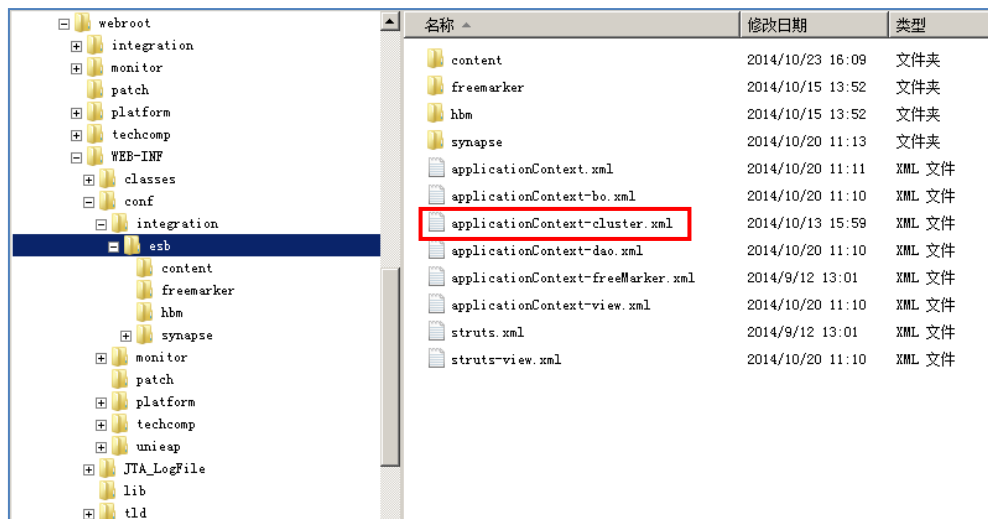


图 2.3.2.2 applicationContext-cluster 配置文件

打开并修改此配置文件，如图 2.3.2.3 所示。根据该文件中的说明，首先去掉 id=“clusterActivator”的 bean 定义注释。之后，针对 id=“clusterManager”的 bean 定义中属性名为“multicastAddress”的属性值可按需设定。此组播 IP 地址应设置为 224.5.6.7~239.5.6.7 的范围内任意 IP。

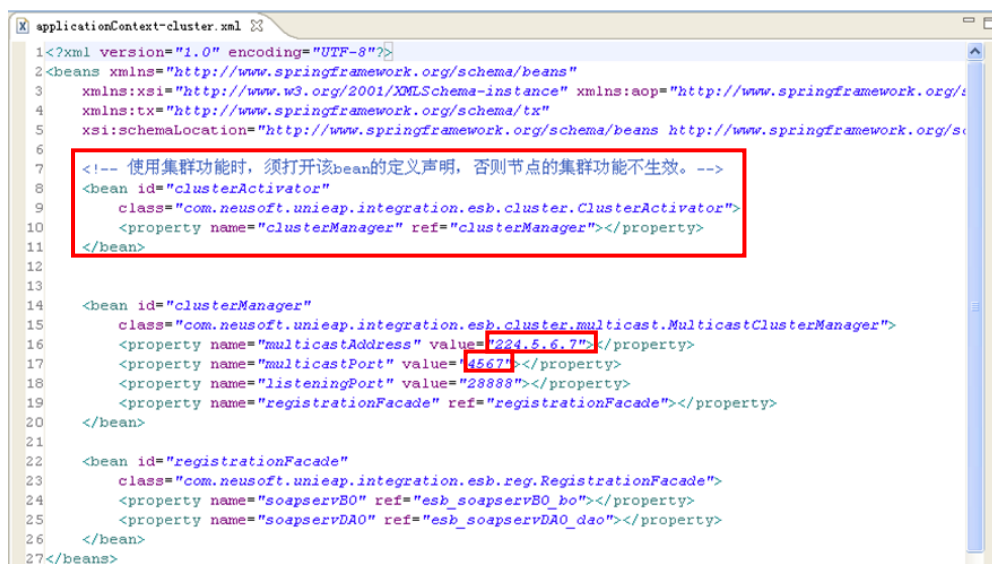


图 2.3.2.3 修改配置

步骤 4：将配置完毕后的工程部署到“前提准备”中准备好的 WEB 服务器集群中。该处具体情形应按照实际选用的应用服务器的使用说明进行。

步骤 5：部署完毕后，访问应用服务器提供的地址即可。

第3章 ESB 管理控制台使用说明

ESB 的服务管理页面提供了针对服务提供方的服务注册、服务维护和服务查看功能。

- 服务注册：将基于 SOAP 的 Web Service 注册到 ESB 上。
- 服务查看：查看已注册服务的信息和状态。
- 服务维护：对已注册服务进行变更维护。

集群管理页面提供了针对 ESB 每个节点的信息和同步功能：

- 节点连通性查看：可以查看当前运行环境中的 ESB 实例节点，以及其间的连通

性状态。

- SOAP 代理运行态管理：可以查看当前运行环境中的 ESB 实例节点，以及各节点上的服务运行状态。

3.1 服务管理

3.1.1 服务注册

ESB 服务器启动后，打开浏览器进行访问，成功访问后如图 3.1.1 所示。点击左侧服务注册，在显示的表单中填入注册信息。

服务标识：唯一显示该服务的标识，在服务维护中来区别其他服务。

服务名称：服务的名称，在服务维护中可以查看到。

服务描述：描述该服务。

认证账号：ESB 访问服务端时的账号。

认证密码：ESB 访问服务端时的密码。

目标 WSDL：服务端的 WSDL 地址。

关键点就是 WSDL 的地址，将第三章中复制到的 WSDL 地址，写入此文本框中。全部填写完毕后，如图 3.1.2 所示。

【注意】：WSDL 地址中一定要注意服务器的 IP 地址，不要输入“localhost”这样的地址。

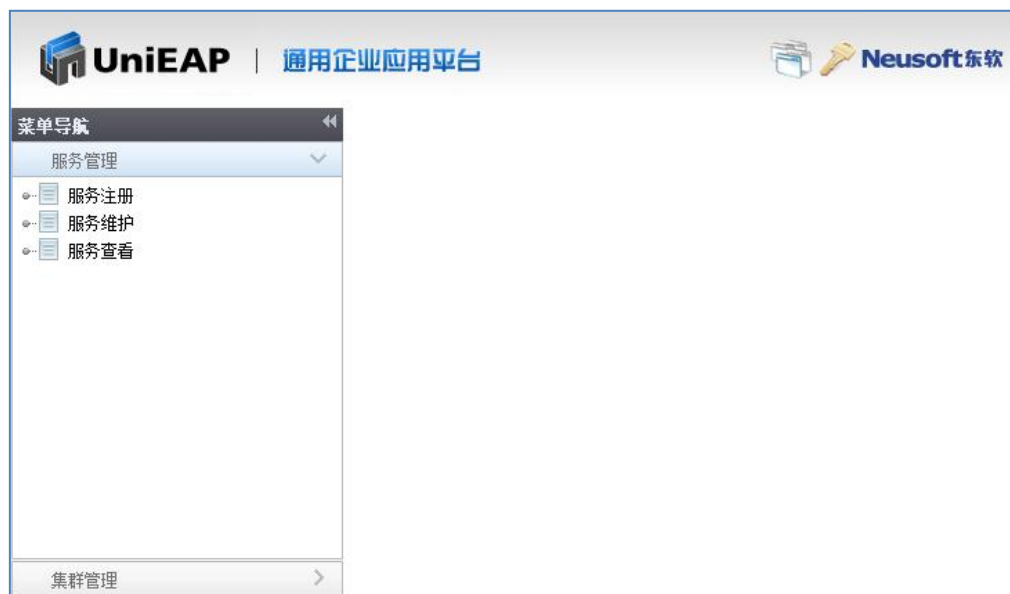


图 3.1.1 服务管理页面

The screenshot shows the '服务注册' (Service Registration) window, specifically 'Step1: 基本信息填写' (Step 1: Basic Information Filling). The window is divided into three main sections: '基本信息' (Basic Information), '目标服务信息' (Target Service Information), and '安全认证信息' (Security Authentication Information).
1. '基本信息' (Basic Information):

- '服务标识' (Service Identifier): Text input field containing 'providerService'.
- '服务名称' (Service Name): Text input field containing 'providerService'.
- '服务描述' (Service Description): Text area containing '服务测试' (Service Test).

2. '目标服务信息' (Target Service Information):

- '目标WSDL' (Target WSDL): Text input field containing 'http://10.4.46.58:8080/framework/ws/providersrc/providerdc/provider?wsdl'.

3. '安全认证信息' (Security Authentication Information):

- '认证账号' (Authentication Account): Text input field containing 'admin'.
- '认证密码' (Authentication Password): Password input field (masked with dots).

At the bottom left, there is a '下一步' (Next Step) button.

图 3.1.2 服务注册页面

注册信息确认无误后，点击图 3.1.2 中的下一步按钮，如果服务注册成功此处会显示服务的端口列表。如图 3.1.3 所示。因为只创建了一个服务，所以这里只能看到 **ProviderBOPort** 端口。然后继续点击下一步进入下一画面，画面中会显示出服务中定义的所有方法。如图 3.1.4 所示。每个方法的传入参数都可以写备注进行说明，方法的功能及使用方法也可以在操作说明中描述。

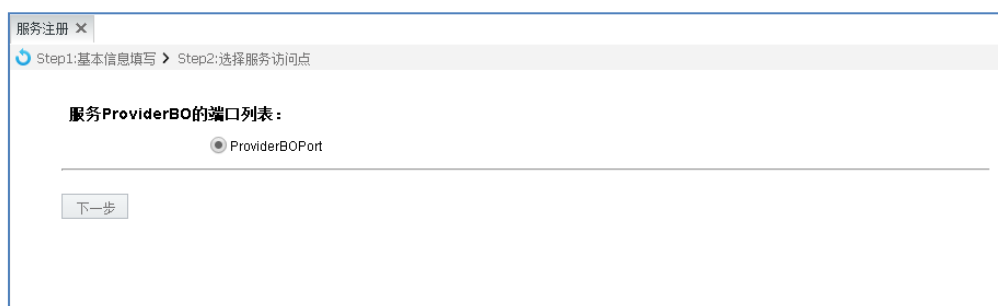


图 3.1.3 服务端口

服务注册

Step1:基本信息填写

Step2:选择服务访问点

Step3:补充操作说明

补充服务操作详细说明

getName操作

操作名称: **getName**

参数信息: **arg0:string**

传入一个字符串

返回值信息: **return:string**

返回一个字符串

操作说明: 为了后面章节的测试, 所以这个方法在执行的时候会报空指针的异常。

getCount操作

操作名称: **getCount**

参数信息: **无**

返回值信息: **return:int**

返回用户的数量

获得用户的数量

操作说明:

getList操作

操作名称: **getList**

参数信息: **string:string**

泛型为String的List集合

返回值信息: **string:string**

返回这个List集合

传入List, 并原样返回这个List

操作说明:

图 3.1.4 服务中的方法

确认后, 点击“完成”按钮, 会弹出成功提示。如图 3.1.5 所示。点击“即刻启动该服务”后, 此服务会马上启动, 并重新打开服务注册页面。点击“注册新服务”后, 此服务不会立即启动, 并重新打开服务注册页面。



图 3.1.5 注册成功提示




3.1.2 服务维护


点击服务维护，即可打开维护页面，见图 3.2.1 所示。该画面会显示服务标识，服务名称，注册时间，运行状态，和操作。操作列中包含 5 个图标，将鼠标悬停到对应的图标上，光标右侧会提示出对应的操作提示。分别有“开始服务”，“删除服务”，“编辑认证信息”，“停止服务”，“查看 WSDL”。

服务维护					
服务维护					
	服务标识	服务名称	注册时间	运行状态	操作
i	zdwservice1	zdwservice	2014-09-26 10:58:23	已停止	▶ 🗑️ 🛡️
i	zdwService2	zdwService2	2014-09-28 15:24:12	已停止	▶ 🗑️ 🛡️
i	zdwservice3	adfasdf	2014-09-28 16:16:26	已停止	▶ 🗑️ 🛡️
i	xyTest2	xyTest2	2014-10-13 16:08:47	已停止	▶ 🗑️ 🛡️
i	xyTest3	xyTest3	2014-10-14 13:45:09	已停止	▶ 🗑️ 🛡️
i	cluster62	集群测试服务	2014-10-14 14:42:08	已停止	▶ 🗑️ 🛡️
i	ESBservice	ESBservice	2014-10-17 18:05:06	已停止	▶ 🗑️ 🛡️
i	fisMes	fisMes	2014-10-17 21:31:32	已停止	▶ 🗑️ 🛡️
i	providerBO	providerBO	2014-10-20 16:24:58	已停止	▶ 🗑️ 🛡️
i	providerService	providerService	2014-10-21 10:49:33	运行中	▶ 🗑️ 🛡️

图 3.3.1 服务维护页面

操作功能图标：

-  点击按钮可以启动对应的服务。
-  点击按钮可以删除对应的服务。
-  点击按钮可以修改认证用户名和密码。

 点击按钮可以停止该服务。

 点击按钮可以查看对应的 WSDL 信息。

3.1.3 服务查看

点击服务查看，即可打开服务查看页面，见图 3.3.1 所示。该画面会显示服务标识，服务名称，注册时间，运行状态，和操作。

服务查看 ×

服务查看

	服务标识	服务名称	注册时间	运行状态	操作
	zdwservice1	zdwservice	2014-09-26:09-10-23 am	 已停止	<无>
	zdwService2	zdwService2	2014-09-28:09-03-12 pm	 已停止	<无>
	zdwservice3	adfasdf	2014-09-28:09-04-26 pm	 已停止	<无>
	xyTest2	xyTest2	2014-10-13:10-04-47 pm	 已停止	<无>
	xyTest3	xyTest3	2014-10-14:10-01-09 pm	 已停止	<无>
	cluster62	集群测试服务	2014-10-14:10-02-08 pm	 已停止	<无>
	ESBservice	ESBservice	2014-10-17:10-06-06 pm	 已停止	<无>
	fisMes	fisMes	2014-10-17:10-09-32 pm	 已停止	<无>
	providerBO	providerBO	2014-10-20:10-04-58 pm	 已停止	<无>
	providerService	providerService	2014-10-21:10-10-33 am	 运行中	

图 3.3.1 服务查看页面

3.2 集群管理

3.2.1 节点联通性状态查看

相关操作界面如图 3.2.1.1 所示。该图例中显示了当前集群中有两个节点。IP 分别是 10.4.44.62 和 10.4.44.99。



图 3.2.1.1 节点连通性状态查看

可以通过“手动指定”来设置集群探查起始节点，如图 3.2.1.2 所示。



图 3.2.1.2 指定起始节点

指定后起始节点变更为 10.4.44.99。如图 3.2.1.3 所示。



图 3.2.1.3 起始节点

3.2.2 SOAP 代理运行状态管理

相关操作界面如图 3.2.2.1 所示。



图 3.2.2.1 起始节点

可手动指定集群探索起始节点，如图 3.2.2.2 所示。



图 3.2.2.2 起始节点

“同步状态到集群”功能将会把对应节点的服务运行状态同步到集群其它节点上。



图 3.2.2.3 更改节点

点击“确定”后，该节点的服务运行态将同步到集群其它节点上。图 3.2.2.4。



图 3.2.2.4 同步状态到集群

第4章 服务调用

在 ESB 上注册的服务作为实际服务的代理运行，较之于实际服务，对于服务调用端而言服务接口是一致的，仅是物理绑定地址发生了变化，由实际服务地址变为 ESB 服务地址。所以，在 ESB 上被注册的服务，仍然可以通过各种通用的方式进行调用。除此之外，ESB 组件还提供了客户端调用开发工具包，方便 UniEAP Platform 开发环境下的服务调用程序开发。

4.1 服务调用工具配置说明

ESB 组件提供的客户端调用开发工具包的形态为 UniEAP Platform 开发组件，由一个 SC 和一个 DC 组成，分别为 integration SC 和 client DC。

在开发工程中引入上述组件后，需以 patch 方式修改 client DC 中 content/conf 目录下的 applicationContext.xml 文件的发布结果。关于 patch 工程的详细说明请参考《UniEAP V4 用户手册》相关章节。如图 4.1.1 所示，修改配置文件中 name 为 esbBaseUrl 的 property 节点。将 value 中的 IP 修改为 UniEAP ESB 组件的访问地址。authName 和 authPwd 的值要修改为认证的用户和密码，修改后保存。

【注意】：这里只修改 value="http://10.4.44.130:8280/services/" 中的红色加粗 IP 部分。其它内容不用修改。

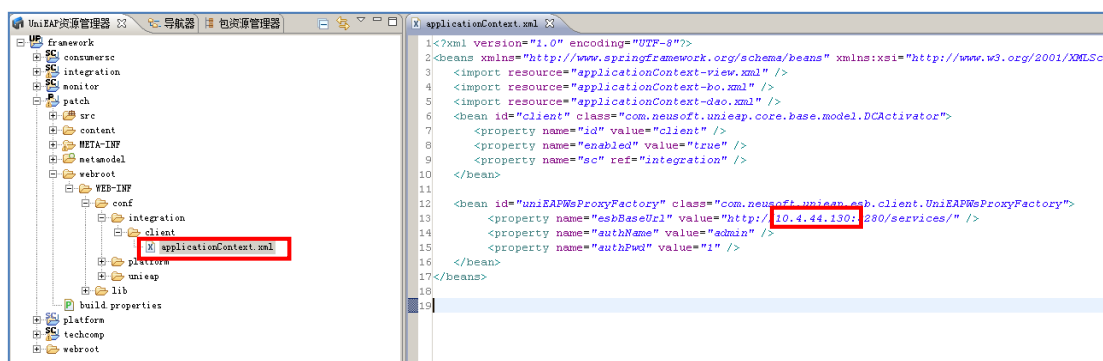


图 4.1.1 修改配置文件

4.2 服务调用示例

本节将基于 UniEAP V4 的模型驱动开发，创建一个开发示例，简单演示服务调用端开发过程。

首先创建一个名为“consumersc”的 SC 和一个名为“consumerdc”的 DC 工程。将服务提供端所发布服务对应的接口定义添加到 consumerdc 的编译类路径中。如图 4.2.1 所示。

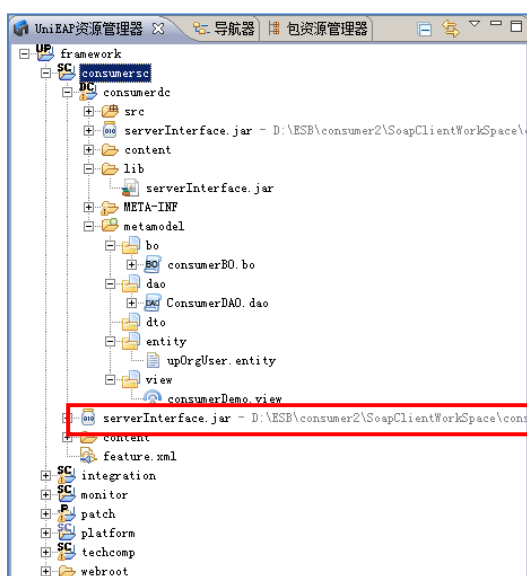


图 4.2.1 工程目录

配置服务接口实例。该接口实例的配置信息如图 4.2.2 所示。其中，

- `<constructor-arg value="providersc.providerdc.bo.ProviderBO"></constructor-arg>` 指定了接口名称。
- `<constructor-arg value="providerService"></constructor-arg>` 指定了 ESB 上的服务标识名称。

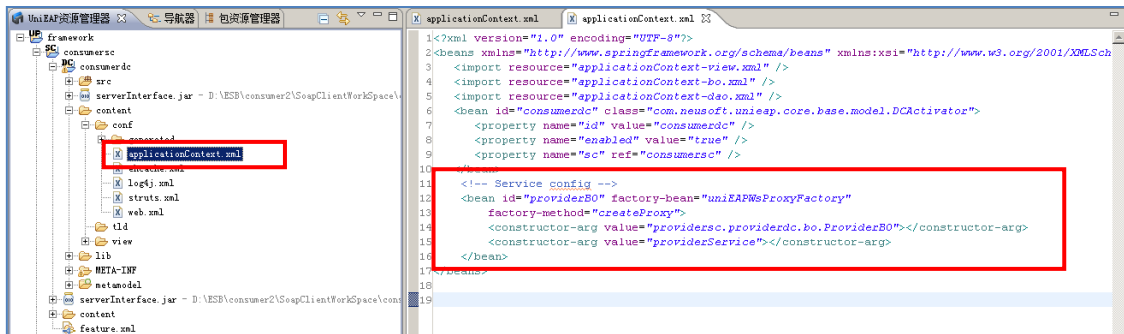


图 4.2.2 客户端配置文件

接下来简单定义 ConsumerBO 接口中的示例方法，图 4.2.3 所示。

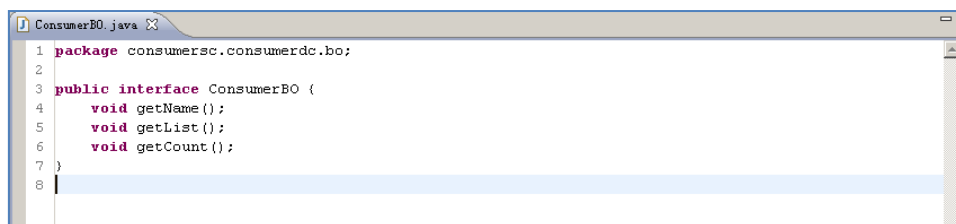


图 4.2.3 客户端方法

方法的实现如图 4.2.4 所示。该接口实现类中 ProviderBO 接口的成员属性是通过 Spring IoC 方式注入的，所注入的实例乃由图 4.2.2 中的配置而定义。关于该服务的调用，是以面向接口编程的方式进行。客户端调用工具向开发人员屏蔽了服务调用通信与对象拆解过程。

```

1 package consumersc.consumerdc.bo.impl;
2 import java.util.ArrayList;
3 import java.util.List;
4 import providersc.providerdc.bo.ProviderBO;
5 import com.neusoft.unieap.core.annotation.ModelFile;
6 import consumersc.consumerdc.bo.ConsumerBO;
7 import consumersc.consumerdc.dao.ConsumerDAO;
8 @ModelFile(value = "consumerBO.bo")
9 public class ConsumerBOImpl implements ConsumerBO {
10     private ProviderBO providerBO;
11     private ConsumerDAO consumerDAO;
12     public void setConsumerDAO(ConsumerDAO consumerDAO) {
13         this.consumerDAO = consumerDAO;
14     }
15     public void setProviderBO(ProviderBO providerBO) {
16         this.providerBO = providerBO;
17     }
18     @Override
19     public void getCount() {
20         int i = providerBO.getCount();
21         int j = consumerDAO.userCount();
22         System.out.println("用户数量: " + i);
23         System.out.println("Consumer端查询的用户数量: " + j);
24     }
25     @Override
26     public void getList() {
27         List<String> names = new ArrayList<String>();
28         names.add("demoUser1");
29         names.add("demoUser2");
30         names.add("demoUser3");
31         names.add("demoUser4");
32         System.out.println("名单: "
33             + providerBO.getList(names));
34     }
35     @Override
36     public void getName() {
37         String name = "张三";
38         System.out.println("姓名: " + providerBO.getName(name));
39     }
40 }

```

图 4.2.4 方法的具体实现

写好方法后，在 view 层的 consumerDemo.view 中添加三个按钮，分别给每个按钮添加调用后台方法的事件。获取用户数调用“getCount”方法，获取名单调用“getList”方法，获取姓名调用“getName”方法。并且在成功回调函数中弹出“SUCESS”提示窗。完成后，发布工程然后启动服务，此时页面如 4.2.5 所示。

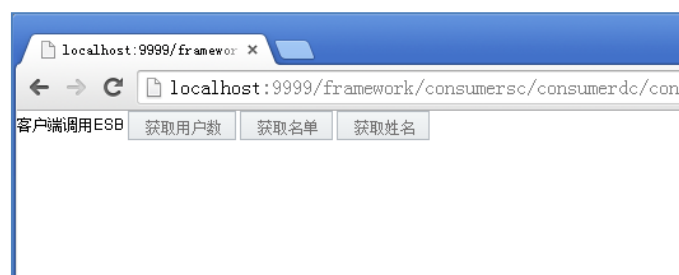


图 4.2.5 模拟客户端页面

点击“获取用户数”按钮，弹出 SUCESS 说明调用成功。如图 4.2.6 所示。



图 4.2.6 成功调用方法

此时观察控制台，调用的方法将内容全部返回并输出。图 4.2.7 此时说明通信成功。如果输入内容是 `null`。则说明以上步骤没有做好。请仔细检查核对。

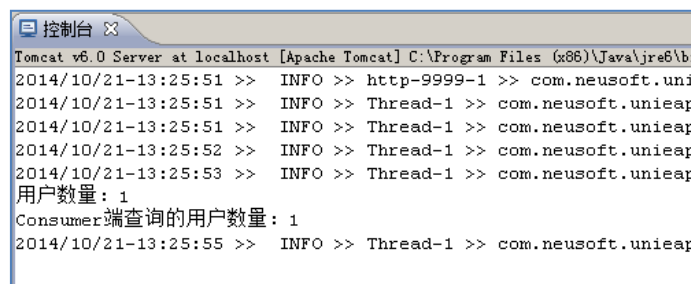


图 4.2.7 服务端成功返回数据

第5章 服务监控

在生产环境下实时追踪应用运行轨迹、监控人机交互过程，辅助快速定位与分析系统故障与性能瓶颈，通过在 ESB 端添加监控服务，最终达到保证应用服务质量、提升用户体验的效果目标。再添加监控服务之前，如果对 UniEAP V4 提供的应用监控功能不是很了解，或对以下操作过程和内容感到困惑，请先阅读《UniEAP 应用监控用户手册》的相关内容。

5.1 ESB 监控服务配置

打 UniEAP ESB 的 webroot 目录，将得到的 monitor 文件夹粘贴到 webroot 的根目录下。如图 5.1.1。

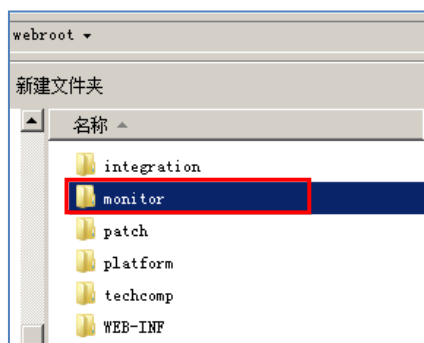


图 5.1.1 ESB 导入监控服务工程

打开 WEB-INF 中 classes 下的 MonitorConfig.properties 配置文件。内容如下所示。注册服务地址 `signIn.target.url` 属性需要修改为监控服务所在的服务器地址，只需要修改 IP 地址及端口。应用标识 `agent.appId` 属性是监控应用中的标识。宿主应用 IP `agent.ip` 属性是本服务所在的服务器地址。`#宿主应用端口 agent .port` 属性为本服务器所在服务器的端口。Agent 追踪数据接收服务地址与认证信息配置以及 Agent 注册相关配置同样都设置为监控服务器所在的 IP 地址和端口。如果配置参数中有不明确的请参照 UniEAP 应用监控用户手册。

开关配置中，扩展开关必须设置为 `true`。

【注意】:

(1) 请不要使用已经存在的应用标识。

(2) `agent.ip` 不建议填写，尤其是在集群环境下。

(3) 集群环境下每个节点的 `port` 要设置为统一值（如 8080），`agent.port` 即设置为该统一的端口值。

```
#-----
### 监控项开关配置
#-----
#
#缺省配置为：
#SpringBean方法监视
enableBeanMethodMonitoring = false
#Web请求监视
enableWebRequestMonitoring = false
#用户活动监视（登入、登出）
enableUserActivitiesMonitoring = false
#WebSession监视（创建、销毁）
enableWebSessionMonitoring = false
#SQL语句执行监视
enableSqlStatementMonitoring = false
#方法参数及返回值（注意：打开此开关后，可能会产生特别大的数据量，所以非特殊情况，不建议打开）
enableBoParameterMonitoring = false
#SQL ResultSet返回值阈值，如果不想进行阈值检测，不要设置该值
#sqlResultSetThreshold = 50000
#Log日志监视
enableLogMonitoring = false
#服务提供监视，注意打开此开关的话要同时打开enableBeanMethodMonitoring开关，否则此开关无效
enableServiceProviderMonitoring = false
#服务调用监视，注意打开此开关的话要同时打开enableBeanMethodMonitoring开关，否则此开关无效
enableServiceConsumerMonitoring = false
#扩展开关，此项必须设置为true
enableExtendMonitoring= true

#
#用户可通过覆盖自定义配置覆盖缺省选项。
```

```
#-----  
### Agent自描述信息配置  
#-----  
#  
#该组配置均为“必须设置选项”！  
#  
#应用标识  
agent.appId = esb  
  
#应用服务期信息  
agent.deployment.server = tomcat  
  
#对于weblogic服务器，在相关探测器类包含在类路径的情况下，ip、端口、应用路径不必配置，  
#支持自动获取。其他无对应探测器的应用服务器类型须配置此三项信息。  
#宿主应用IP  
agent.ip=10.4.44.130  
  
#宿主应用端口  
agent.port=8080  
  
#宿主应用路径  
agent.path=/framework  
  
#agent组件控制服务地址（相对应用路径的地址）  
agent.control.url=/ws/rest/  
  
#agent组件控制服务认证信息  
agent.control.auth.name=admin  
  
agent.control.auth.pwd=1  
  
#-----  
### Agent追踪数据接收服务地址与认证信息配置  
#-----  
#
```

```
#该组配置均为“必须设置选项”！
#
delivery.target.url=http://10.4.44.130:8888/framework/monitorPubEntry

delivery.target.auth.name=admin

delivery.target.auth.pwd=1

#-----
### Agent追踪数据发送配置。
#-----
#
#发送编码配置，缺省为plain，可自定义覆盖为gzip。
#delivery.content.encoding=plain

#发送批量数据包大小设置，缺省为20000。
#delivery.content.batchSize=20000

#-----
### Agent注册相关配置
#-----
#
#注册服务地址，为“必须设置选项”！
signIn.target.url=http://10.4.44.130:8888/framework/monitorRegister

#注册请求拒绝后的重试延迟，缺省为60分钟
#signIn.rejectDelay = 5

#注册请求被搁置后的重试延迟，缺省为30分钟
#signIn.waitDelay = 10

#注册请求发生网络失败后的充实延迟，缺省为5分钟
#signIn.timeoutDelay = 5

#数据传输连续失败次数阈值。数据传输连续失败次数超过该值后，将重新发起新的注册。
```

```
#缺省为3次。
#signIn.failureCeil = 3

#-----
### monitor4jdbc代理驱动设置
#-----

#被代理的真实驱动类，缺省为oracle.jdbc.driver.OracleDriver，为"必须设置项"。
monitor4jdbc.realDirverClass = oracle.jdbc.driver.OracleDriver
#日志过滤相关设置，logger.include代表该包下的日志需要采集，
#logger.exclude代表该包下的日志不需要被采集，如果有多个包，用分号分割
logger.include = com.auxapp.*
logger.exclude = com.neusoft.*
```

配置文件修改好后，发布服务并启动服务器。然后打开应用监控服务页面，如图 5.1.2 所示。

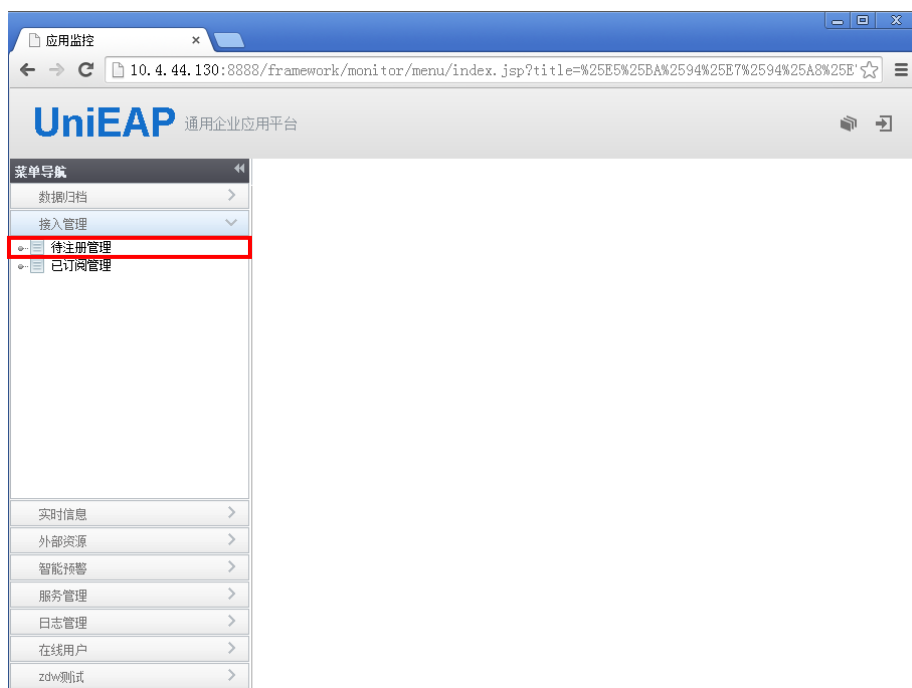


图 5.1.2 监控服务注册及订阅

在此页面的操作均可参照《UniEAP V4 应用监控用户手册》中第三章的监控总线与监控控制台使用说明。当监控服务注册并订阅成功后，可以在实时信息中的服务总线监控中查看到对应的应用名称。如图 5.1.3。

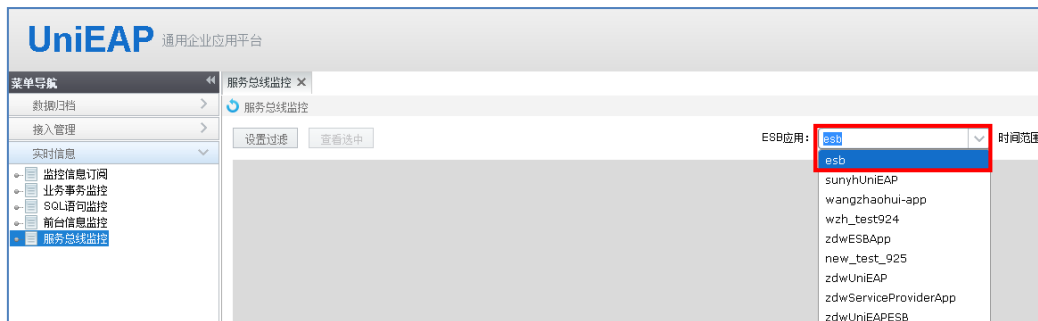


图 5.1.3 监控服务订阅成功

5.2 查看服务总线监控散点分布图

ESB 添加监控服务后，这时我们就可以查看调用过程和信息，打开应用监控页面，找到实时信息选项下的服务总线监控。如图 5.2.1 所示

【注意】：在使用实时信息监控时，建议使用火狐浏览器。

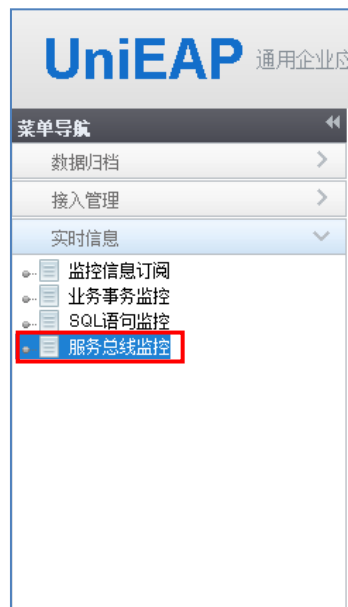


图 5.2.1 服务总线监控

点击服务总线监控后，会弹出监控页面。图 5.2.2。页面中包含如下内容：

设置过滤按钮：包含按“服务标识”，“执行结果”，“耗时范围”内容及范围条件，进行散点图的过滤。

选中查看按钮：当鼠标在散点图中选中 1 个或多个点后，点击此按钮会进入详细信息界面。

ESB 应用：该下拉列表中包含已经注册的监控服务，选择需要监控的服务即可。

时间范围：选择一个时间范围内的数据。

刷新按钮：刷新散点图。

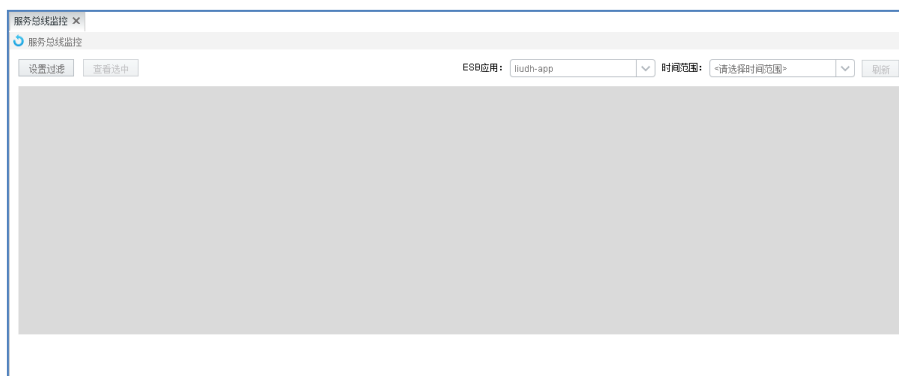


图 5.2.2 总线监控页面

在 ESB 应用中选择已经注册号的监控服务，选择时间范围后，散点图会马上刷新出该时间范围内的数据。如下图 5.2.3 所示。

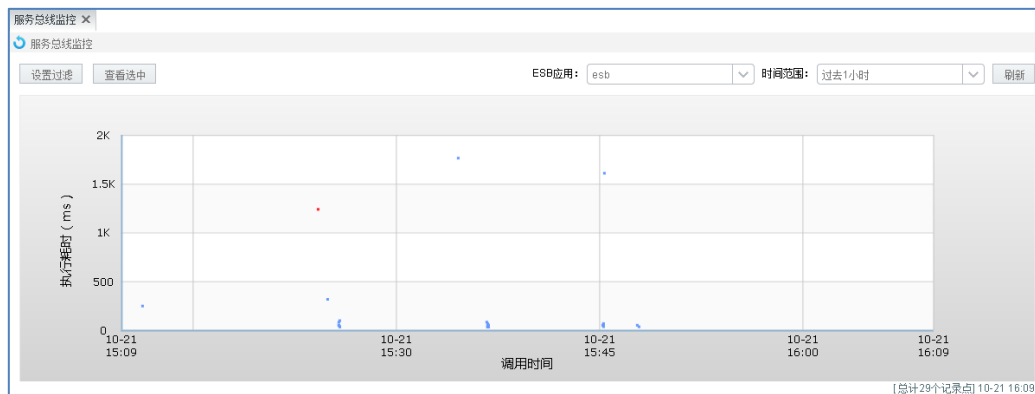


图 5.2.3 总线监控散点分布图

5.3 查看散点分布图详细信息

在第五章中已经创建了一个客户端，所以接下来通过这个客户端来进行具体的操作展示。首先打开客户端并点击客户端的按钮，此时客户端会通过 ESB 来找到服务端，调用已经注册号的 providerService 服务，并且这时散点图中将会产生蓝色的小点。如图 5.3.1 所示。

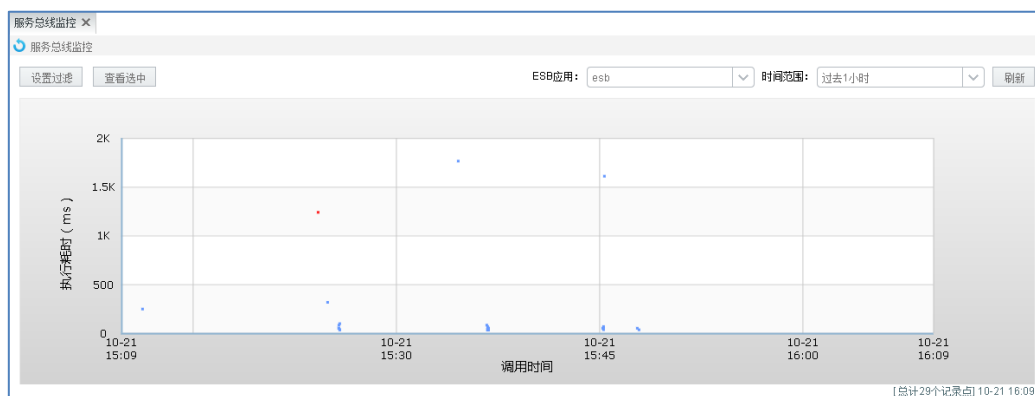


图 5.3.1 总线散点分布图

用鼠标在散点分布图中按住左键，选择最近的数据也就是刚刚点击客户端按钮所产生的记录。选中后松开鼠标左键。如图 5.3.2。然后点击查看选中按钮。

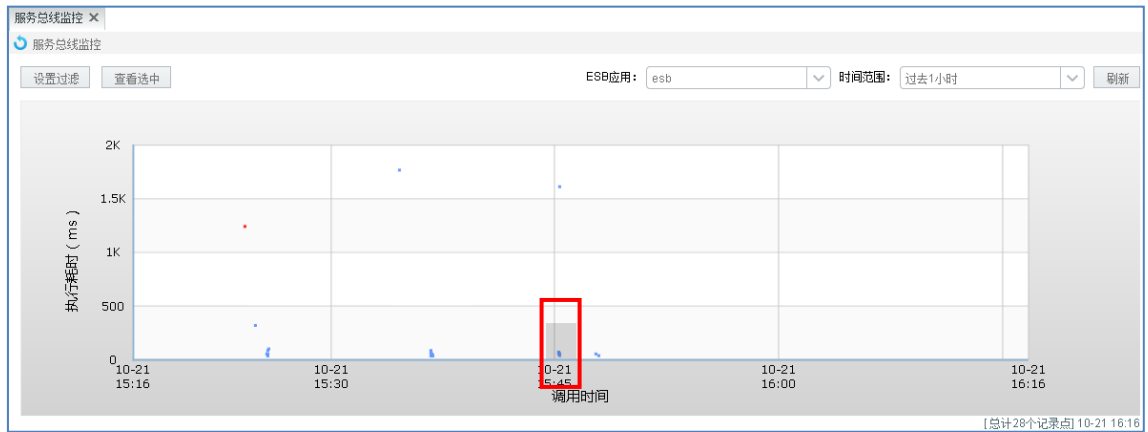


图 5.3.2 选择数据点

点击查看选中按钮后，进入下一画面，如图 5.3.3 所示。图中已经显示出在第 4 章注册的“providerService”服务，图中的“数量”为该注册服务中被调用的次数。点击服务标识“providerService”即可查看具体调用信息。

服务总线监控	
散点分布图 > 查看选中	
- 所有选中记录	
服务标识	数量
providerService	4
提示：共选中[4]条记录，涉及到[1]条SQL语句	
- 系统异常记录	
服务标识	数量
提示：所选范围内包含[0]条系统异常记录，涉及到[0]种业务方法	
- 业务异常记录	
服务标识	数量
提示：所选范围内包含[0]条业务异常记录，涉及到[0]种业务方法	

图 5.3.3 详细信息

画面中显示了平均耗时，最大耗时等数据。如图 5.3.4。在选中记录中，点击左侧的加号即可展开，如图 5.3.5 所示。展开后可以显示出服务请求方，服务访问地址，运行结果，其中完整调用可以查看整个调用过程。如图 5.3.6 所示。

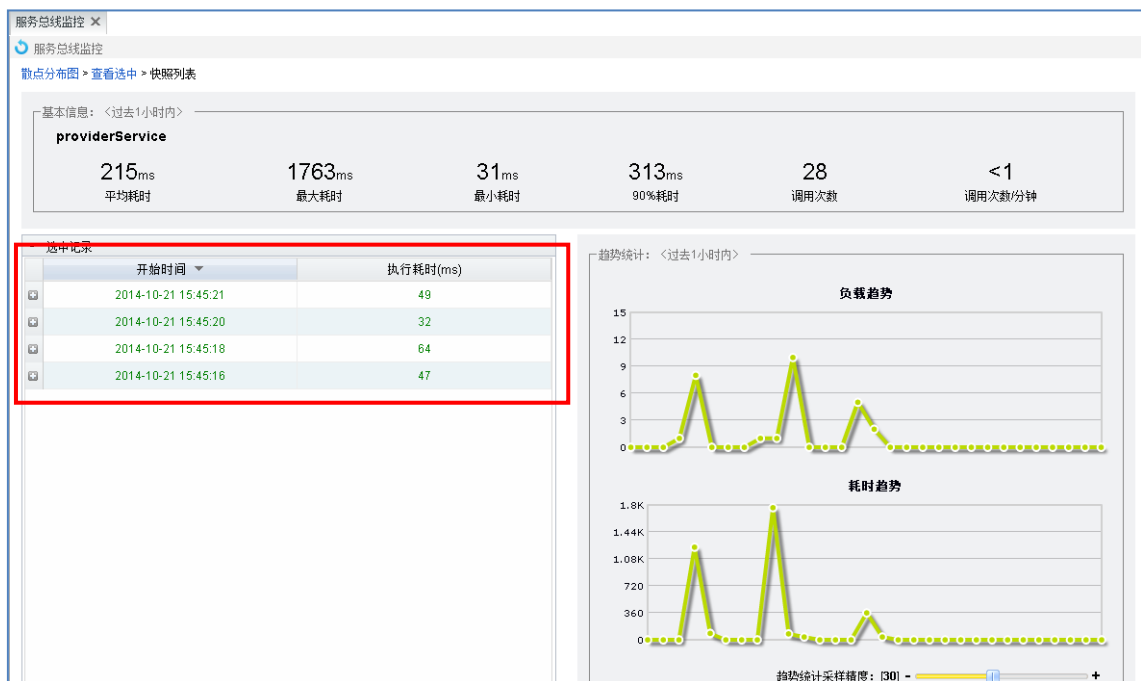


图 5.3.4 快照列表



图 5.3.5 详细服务信息

图 5.3.6 中，展示了服务调用方的 IP 地址，调用的 ESB 服务，和服务提供方信息。点击“查看详情”可查看具体信息。

名称	执行时间	查看详情
服务调用方信息: IP地址-10.4.46.58	-	查看详情
ESB服务标识: providerService	49	查看详情
服务提供方信息: 服务地址-http://10.4.46.58:8080/framework/ws/providers/c/providerdc/provider	-	查看详情

图 5.3.6 服务调用过程

当服务器中产生异常时，散点图中会显示出橘黄色的点。如图 5.3.7 所示。选中该点即可查看详细的异常信息。由于上面操作一直，所以这里不再重复。

通过图 5.3.6 可见，因为只有 ESB 端添加了监控功能，所以在以上监控信息中只能观察到 ESB 端的一个执行情况，对于调用端和服务端是无法进行监控的。

5.4 多应用服务监控配置

在第 5.1 中介绍了总线监控服务。因为只有 ESB 端添加了监控，所以只能监控到 ESB 端的信息。这种情况下存在如下缺点：

1. 只能获取客户端和服务端的 IP 地址。无法获得其他信息。
2. ESB 端调用服务端时，只能获取服务端注册的 WSDL 地址，监控不到有价值的信息。
3. ESB 或服务提供端发生异常时，无法获取具体的错误细节信息。

通过以三点可得知，只在 ESB 端增加监控后，无法获得具体的详细信息。此时的监控数据无论是在系统正常运行或出现异常时，只能确定是哪台服务器。如果服务调用端、服务提供端是基于 UniEAP V4 的模型驱动开发实现的，则可以为这两端也添加监控功能，从而进一步加以整合，让监控数据变得更有价值。

【注意】：组建监控模式时，要确保“服务提供端”，“服务调用端”都是使用的 UniEAP 产品，其他产品无法实现该模式。而且服务调用端、服务提供端、ESB 的监控数据要保存到同一个 MongoDB Server 中，所以推荐所采用的架构模式如下图 5.4.1 所示：

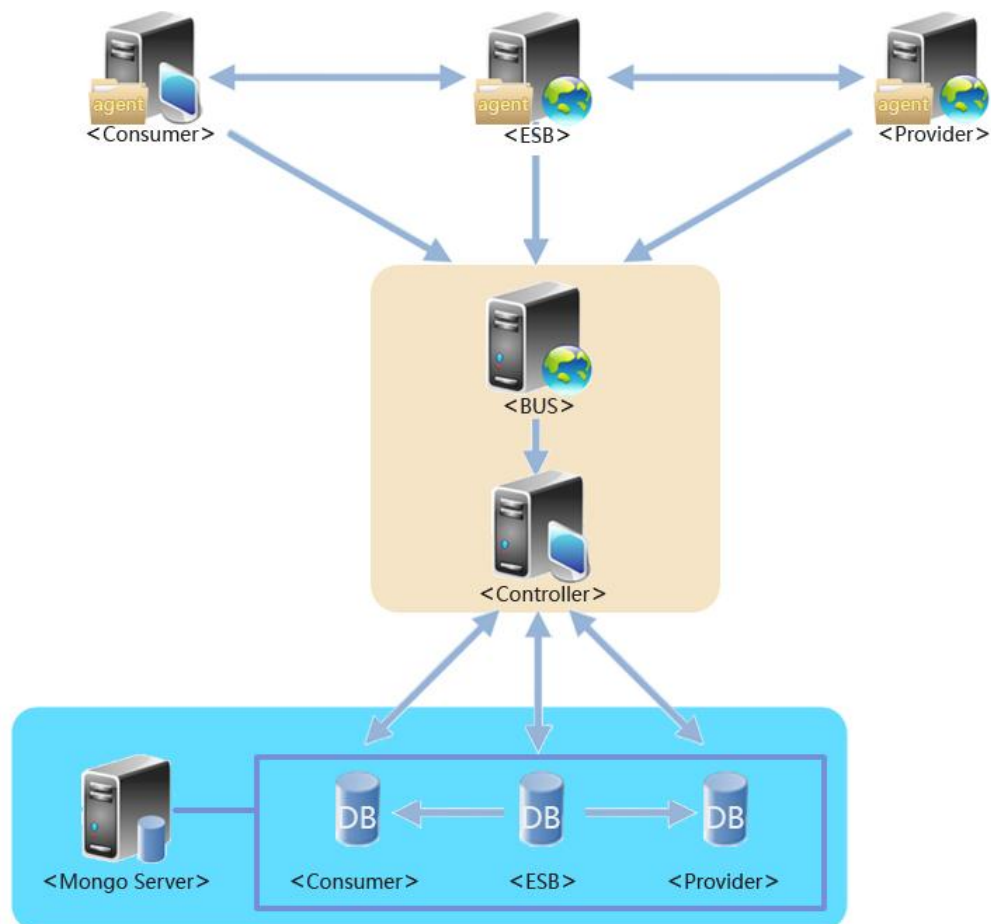


图 5.4.1 架构模式图

5.5 服务提供端服务监控配置

在服务提供端工程中导入图 5.5.1 中的 monitor 和 agent 两个项目。

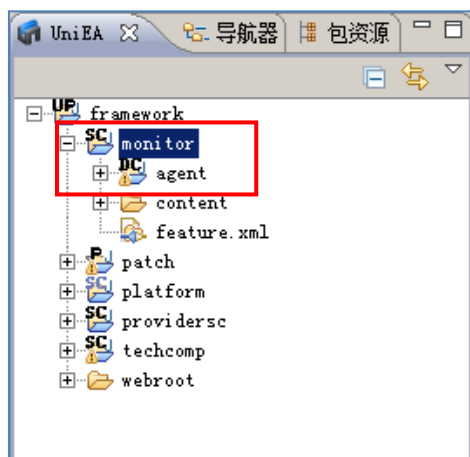


图 5.5.1 导入工程

导入成功后打开 agent 目录，找到 src 下面的 MonitorConfig.properties 配置文件，见图 5.5.2。

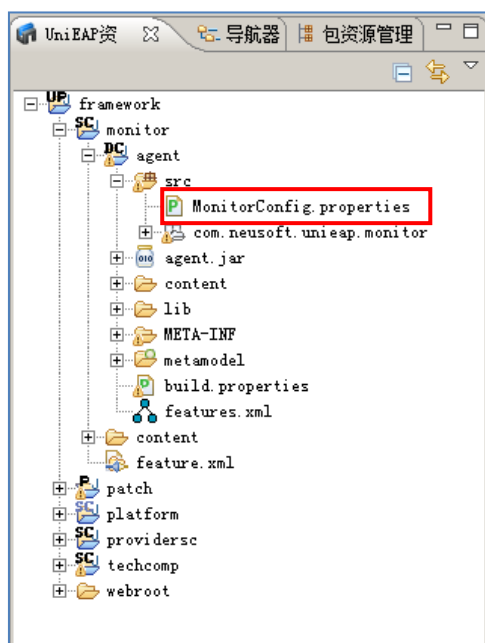


图 5.5.2 配置文件

该例中各个配置属性的值如下所示：

```
#-----
### 监控项开关配置
#-----
#
#缺省配置为：
#SpringBean方法监视
enableBeanMethodMonitoring = true
#Web请求监视
enableWebRequestMonitoring = true
#用户活动监视（登入、登出）
enableUserActivitiesMonitoring = true
#WebSession监视（创建、销毁）
enableWebSessionMonitoring = true
#SQL语句执行监视
enableSqlStatementMonitoring = true
#方法参数及返回值（注意：打开此开关后，可能会产生特别大的数据量，所以非特殊情况，不建议打开）
enableBoParameterMonitoring = true
#SQL ResultSet返回值阈值，如果不想进行阈值检测，不要设置该值
sqlResultSetThreshold = 50000
#Log日志监视
enableLogMonitoring = true
#服务提供监视，注意打开此开关的话要同时打开enableBeanMethodMonitoring开关，否则此开关无效
enableServiceProviderMonitoring = true
#服务调用监视，注意打开此开关的话要同时打开enableBeanMethodMonitoring开关，否则此开关无效
enableServiceConsumerMonitoring = false
#扩展开关
enableExtendMonitoring= false
```

```
#
#用户可通过覆盖自定义配置覆盖缺省选项。

#-----
### Agent自描述信息配置
#-----
#
#该组配置均为“必须设置选项”！
#
#应用标识
agent.appId = provider

#应用服务期信息
agent.deployment.server = tomcat

#对于weblogic服务器，在相关探测器类包含在类路径的情况下，ip、端口、应用路径不必配置，
支持自动获取。其他无对应探测器的应用服务器类型须配置此三项信息。
#宿主应用IP
agent.ip=10.4.46.58

#宿主应用端口
agent.port=8080

#宿主应用路径
agent.path=/framework

#agent组件控制服务地址（相对应用路径的地址）
agent.control.url=/ws/rest/

#agent组件控制服务认证信息
agent.control.auth.name=admin

agent.control.auth.pwd=1
```

```
#-----  
### Agent追踪数据接收服务地址与认证信息配置  
#-----  
#  
#该组配置均为“必须设置选项”！  
#  
delivery.target.url=http://10.4.44.130:8888/framework/monitorPubEntry  
  
delivery.target.auth.name=admin  
  
delivery.target.auth.pwd=1  
  
#-----  
### Agent追踪数据发送配置。  
#-----  
#  
#发送编码配置，缺省为plain，可自定义覆盖为gzip。  
#delivery.content.encoding=plain  
  
#发送批量数据包大小设置，缺省为20000。  
#delivery.content.batchSize=20000  
  
#-----  
### Agent注册相关配置  
#-----  
#  
#注册服务地址，为“必须设置选项”！  
signIn.target.url=http://10.4.44.130:8888/framework/monitorRegister  
  
#注册请求拒绝后的重试延迟，缺省为60分钟  
#signIn.rejectDelay = 5  
  
#注册请求被搁置后的重试延迟，缺省为30分钟  
#signIn.waitDelay = 10
```

```
#注册请求发生网络失败后的充实延迟，缺省为5分钟
#signIn.timeoutDelay = 5

#数据传输连续失败次数阈值。数据传输连续失败次数超过该值后，将重新发起新的注册。
#缺省为3次。
#signIn.failureCeil = 3

#-----
### monitor4jdbc代理驱动设置
#-----

#被代理的真实驱动类，缺省为oracle.jdbc.driver.OracleDriver，为"必须设置项"。
monitor4jdbc.realDirverClass = oracle.jdbc.driver.OracleDriver
#日志过滤相关设置，logger.include代表该包下的日志需要采集，
#logger.exclude代表该包下的日志不需要被采集，如果有多个包，用分号分割
logger.include = com.auxapp.*
logger.exclude = com.neusoft.*
```

配置文件修改好后，发布服务并启动服务器。然后打开监控总线对应的控制台页面，对刚刚发起注册请求的服务端应用进行注册和订阅管理，如图 5.5.2 所示。

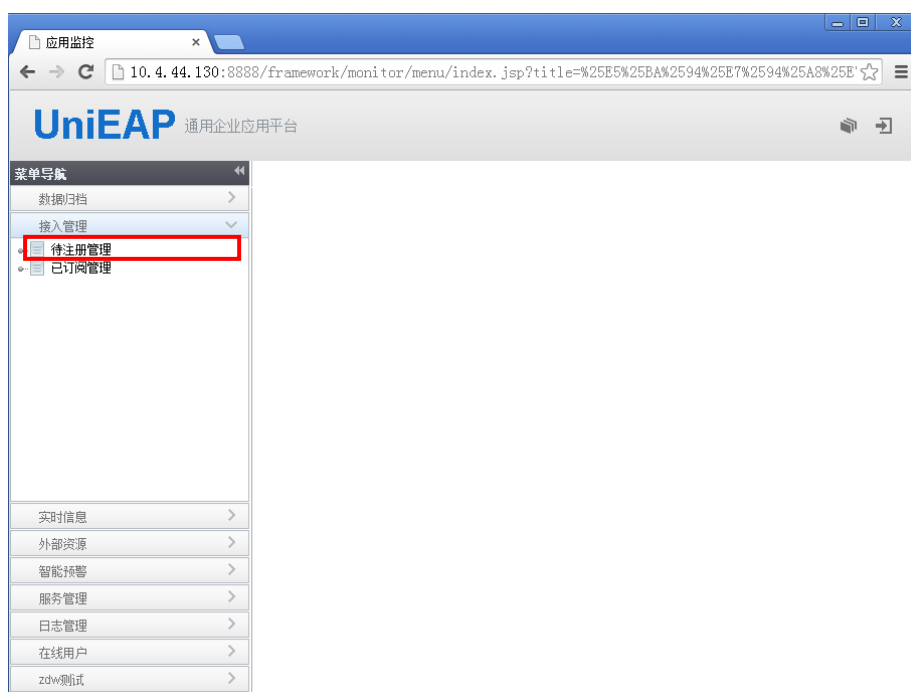


图 5.5.2 服务注册和订阅

该部分的具体操作请参照《UniEAP V4 应用监控用户手册》中第 3 章的 ESB 管理控制台使用说明。

当监控服务注册并订阅成功后，可以在实时信息中的服务总线监控中查看到对应的应用名称。如图 5.5.3。

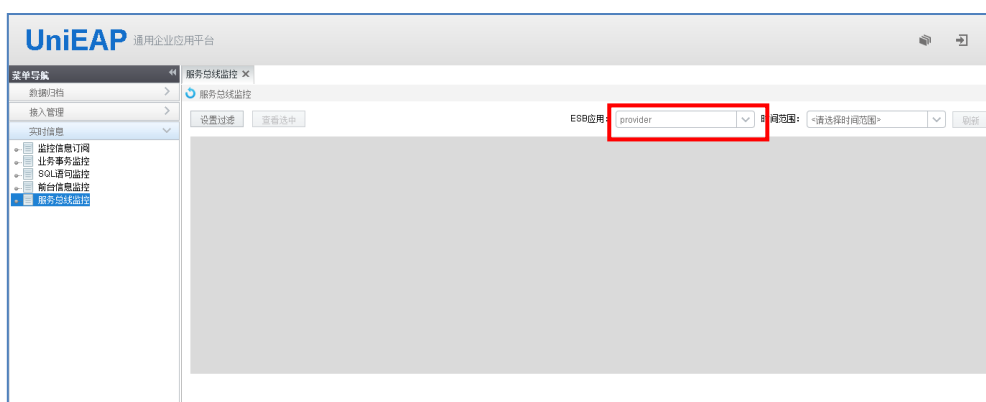


图 5.5.3 服务订阅成功

5.6 服务调用端服务监控配置

与 5.5 服务提供端添加监控服务基本一致，首先导入 monitor 和 agent 项目。如图 5.5.1。

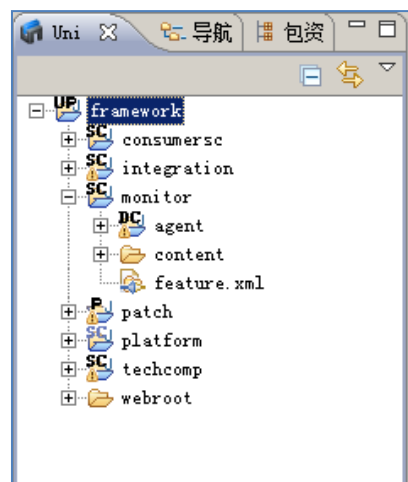


图 5.5.1 导入工程

打开 agent 中 src 下的 MonitorConfig.properties 配置文件。该示例配置的内容如下所示：

【注意】：应用标识请不要使用相同的名字。

```
#-----
### 监控项开关配置
#-----
#
#缺省配置为：
#SpringBean方法监视
```

```

enableBeanMethodMonitoring = true
#Web请求监视
enableWebRequestMonitoring = true
#用户活动监视（登入、登出）
enableUserActivitiesMonitoring = true
#WebSession监视（创建、销毁）
enableWebSessionMonitoring = true
#SQL语句执行监视
enableSqlStatementMonitoring = true
#方法参数及返回值（注意：打开此开关后，可能会产生特别大的数据量，所以非特殊情况，不建议打开）
enableBoParameterMonitoring = true
#SQL ResultSet返回值阈值，如果不想进行阈值检测，不要设置该值
sqlResultSetThreshold = 50000
#Log日志监视
enableLogMonitoring = true
#服务提供监视，注意打开此开关的话要同时打开enableBeanMethodMonitoring开关，否则此开关无效
enableServiceProviderMonitoring = false
#服务调用监视，注意打开此开关的话要同时打开enableBeanMethodMonitoring开关，否则此开关无效
enableServiceConsumerMonitoring = true
#扩展开关
enableExtendMonitoring= false

#
#用户可通过覆盖自定义配置覆盖缺省选项。

#-----
### Agent自描述信息配置
#-----
#
#该组配置均为“必须设置选项”！
#

```

```
#应用标识
agent.appId = consumer

#应用服务期信息
agent.deployment.server = tomcat

#对于weblogic服务器，在相关探测器类包含在类路径的情况下，ip、端口、应用路径不必配置，
支持自动获取。其他无对应探测器的应用服务器类型须配置此三项信息。
#宿主应用IP
agent.ip=10.4.46.58

#宿主应用端口
agent.port=9999

#宿主应用路径
agent.path=/framework

#agent组件控制服务地址（相对应用路径的地址）
agent.control.url=/ws/rest/

#agent组件控制服务认证信息
agent.control.auth.name=admin

agent.control.auth.pwd=1

#-----
### Agent追踪数据接收服务地址与认证信息配置
#-----
#
#该组配置均为“必须设置选项”！
#
delivery.target.url=http://10.4.44.130:8888/framework/monitorPubEntry

delivery.target.auth.name=admin
```



```
delivery.target.auth.pwd=1

#-----
### Agent追踪数据发送配置。
#-----
#
#发送编码配置，缺省为plain，可自定义覆盖为gzip。
#delivery.content.encoding=plain

#发送批量数据包大小设置，缺省为20000。
#delivery.content.batchSize=20000

#-----
### Agent注册相关配置
#-----
#
#注册服务地址，为“必须设置选项”！
signIn.target.url=http://10.4.44.130:8888/framework/monitorRegister

#注册请求拒绝后的重试延迟，缺省为60分钟
#signIn.rejectDelay = 5

#注册请求被搁置后的重试延迟，缺省为30分钟
#signIn.waitDelay = 10

#注册请求发生网络失败后的充实延迟，缺省为5分钟
#signIn.timeoutDelay = 5

#数据传输连续失败次数阈值。数据传输连续失败次数超过该值后，将重新发起新的注册。
#缺省为3次。
#signIn.failureCeil = 3

#-----
### monitor4jdbc代理驱动设置
#-----
```

```
#被代理的真实驱动类，缺省为oracle.jdbc.driver.OracleDriver，为"必须设置项"。
monitor4jdbc.realDirverClass = oracle.jdbc.driver.OracleDriver
#日志过滤相关设置，logger.include代表该包下的日志需要采集，
#logger.exclude代表该包下的日志不需要被采集，如果有多个包，用分号分割
logger.include = com.auxapp.*
logger.exclude = com.neusoft.*
```

配置文件修改好后，发布服务并启动服务器。然后打开应用监控服务页面，如图 5.5.2 所示。

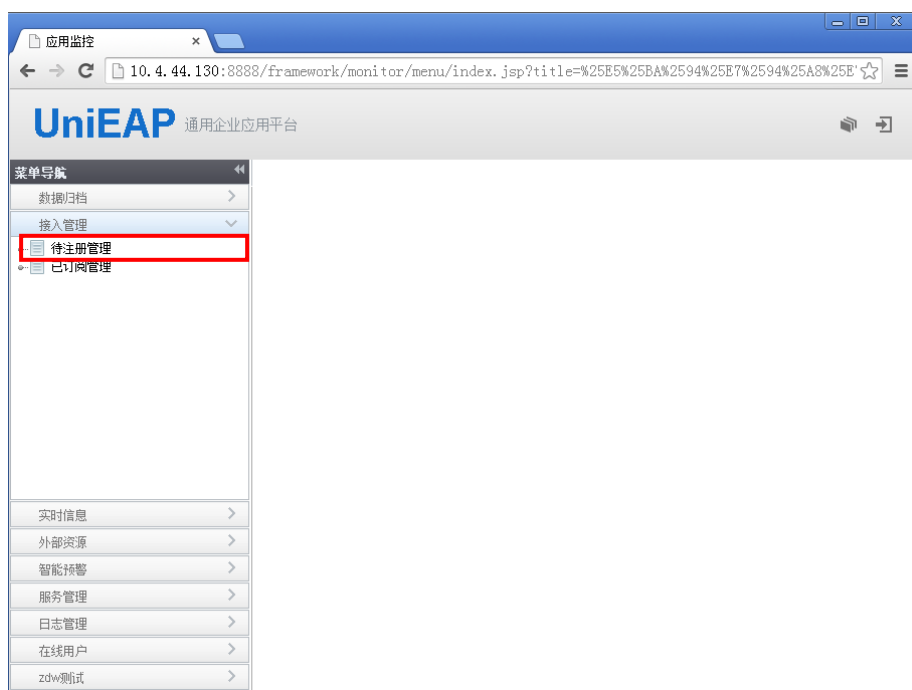


图 5.5.2 注册和订阅

在此页面的操作均可参照《UniEAP V4 应用监控用户手册》中第三章的监控总线与监控控制台使用说明。当客户端应用注册并订阅成功后，可以在实时信息中的服务总线监控中查看到对应的应用名称。如图 5.5.3。

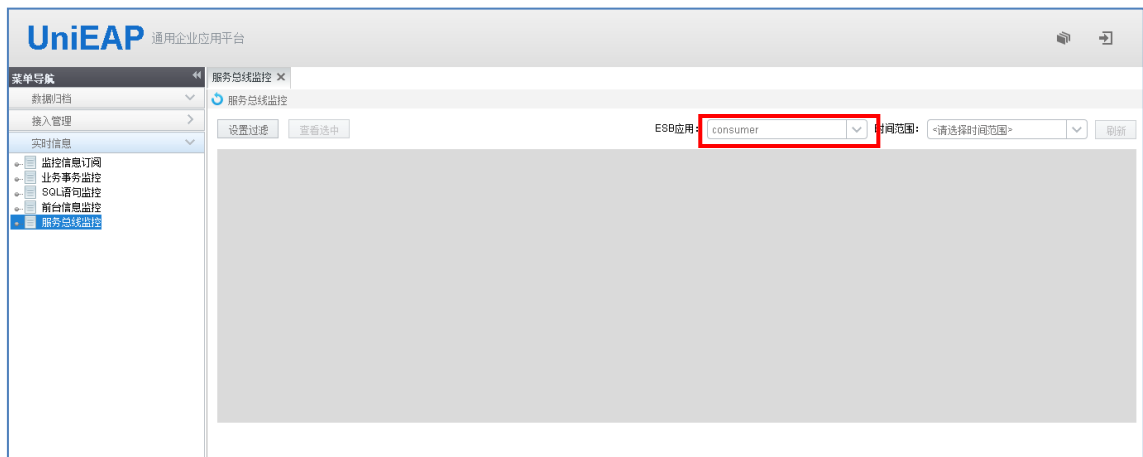


图 5.5.3 注册订阅成功

5.7 监控数据分析

分别在服务提供端，ESB 端和服务调用端中添加监控服务，这时所产生的监控数据会更详细。从调用一直到被调用服务结束，整个流程无任何遗漏的被记录在监控数据中。无论是系统正常运行，或出现异常等情况。用户通过监控记录都可以找到有价值的信息。

下面介绍监控时会发生的三种情况，分别是正常情况，服务异常和总线异常。

5.7.1 正常监控记录

在散点分布图中，蓝色的点代表正常的的数据。首先用客户端页面上的“获取用户数”按钮进行访问，该按钮会在客户端及服务端分别进行数据查询操作。此时观察散点图如下图 5.7.1。1 所示。

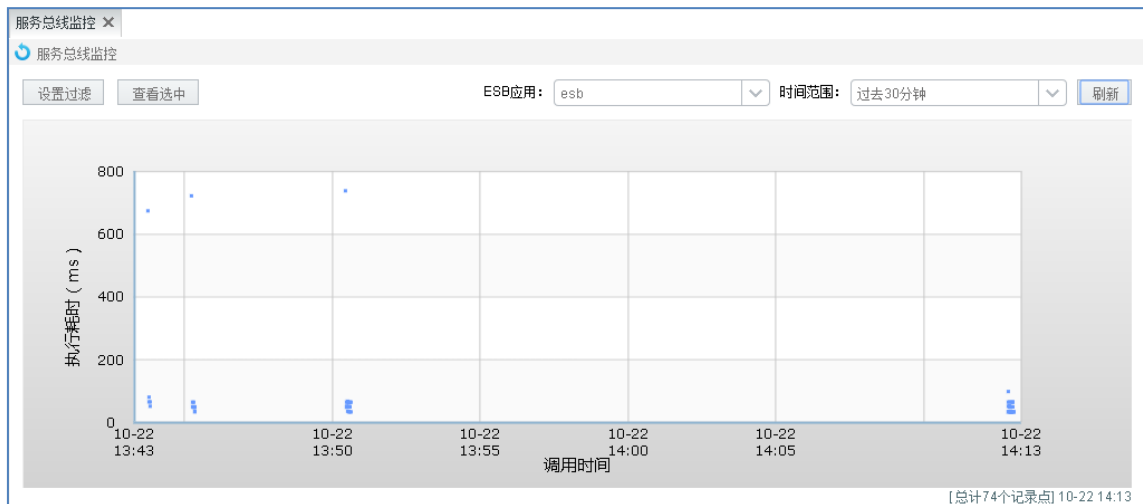


图 5.7.1.1 散点图

选中刚刚访问的这些点后，得到下图 5.7.1.2 信息。可以看到“总线异常记录”和“服务异常记录”中数量为 0，说明这些点中没有异常的记录。

服务总线监控	
散点分布图 > 查看选中	
所有选中记录	
服务标识	数量
providerService	30
提示：共选中[30]条记录，涉及到[1]条SQL语句	
总线异常记录	
服务标识	数量
提示：所选范围内包含[0]条总线异常记录，涉及到[0]种业务方法	
服务异常记录	
服务标识	数量
提示：所选范围内包含[0]条服务异常记录，涉及到[0]种业务方法	

图 5.7.1.2 快照列表

点击“点击查看”链接后，弹出详细信息页面。图 5.7.1.3。图中可以看到完整的调用过程，从 consumer 端的 getCount 方法一直到最后被调用的 provider 端的 userCount 方

法。整个过程清晰明了。完整的监控到了每一个步骤。这时的监控数据才具有实际的参考价值。

名称	执行时间	查看详情
服务调用方信息: IP地址-10.4.46.58, 应用标识-consumer	48	查看详情
consumersc.consumerdc.bo.impl.ConsumerBOImpl.getCount	48	查看详情
consumersc.consumerdc.dao.impl.ConsumerDAOImpl.userCount	2	查看详情
[SQL statement]	2	查看详情
ESB信息: IP地址-172.16.0.86, 服务标识: providerService	31	查看详情
服务提供方信息: 服务地址-http://10.4.46.58:8080/framework/ws/providersc/providerdc/provider, 应用标...	2	查看详情
providersc.providerdc.bo.impl.ProviderBOImpl.getCount	2	查看详情
providersc.providerdc.dao.impl.ProviderDAOImpl.userCount	2	查看详情
[SQL statement]	2	查看详情

图 5.7.1.3 详细信息

点击 7.3.3 图中的[SQL statement] 后面的“查看详情”，会弹出 SQL 的详细操作。图 5.7.1.4，如果是普通的方法，这里会显示出方法传入的参数和返回值（如果已经开启了相应的监控开关）。让用户对调用的方法一目了然。

名称	执行时间	查看详情
服务调用方信息: IP地址-10.4.46.58, 应用标识-consumer	48	查看详情
consumersc.consumerdc.bo.impl.ConsumerBOImpl.getCount	48	查看详情
consumersc.consumerdc.dao.impl.ConsumerDAOImpl.userCount	2	查看详情
[SQL statement]	2	查看详情
ESB信息: IP地址-172.16.0.86, 服务标识-providerService	31	查看详情
服务提供方信息: 服务地址-http://10.4.46.58:8080/framework/ws/providersc/providerdc/provider, 应用标...	2	查看详情
providersc.providerdc.bo.impl.ProviderBOImpl.getCount	2	查看详情
providersc.providerdc.dao.impl.ProviderDAOImpl.userCount	2	查看详情
[SQL statement]	2	查看详情

详细信息

执行时间: 2ms

影响行数: 1

SQL: select count(*) as col_0_0_ from up_org_user uporguser0_

图 5.7.1.4 查看详情

5.7.2 服务异常监控记录

单击客户端页面的“获取姓名”按钮后，会有异常产生，此时异常情况的信息如下图 5.7.2.1 所示。此时的散点中既有橙色的点还有蓝色的点。

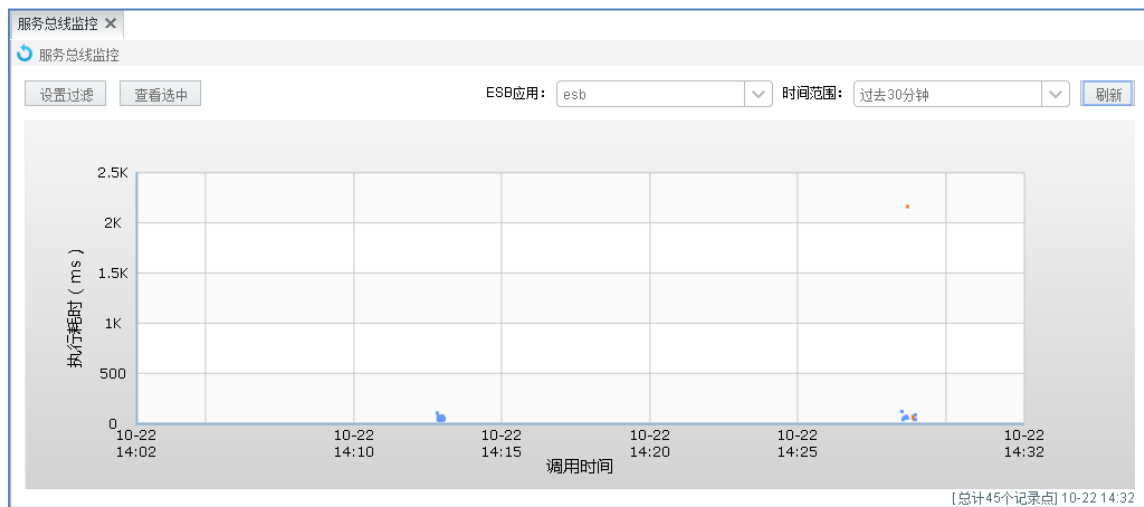


图 5.7.2.1 散点图

选中这些点后，可以观察图 5.7.2.2。其中包含了 4 个服务异常记录。

Figure 5.7.2.2 is a screenshot of the '查看选中' (View Selected) window. It displays a list of selected records. The first section, '所有选中记录' (All selected records), shows a table with columns '服务标识' (Service Identifier) and '数量' (Quantity). The 'providerService' is listed with a quantity of 13. Below this, a note states: '提示: 共选中[13]条记录, 涉及到[1]条SQL语句' (Note: A total of [13] records are selected, involving [1] SQL statements). The second section, '总线异常记录' (Bus abnormal records), shows a similar table with 'providerService' and a quantity of 0. A note states: '提示: 所选范围内包含[0]条总线异常记录, 涉及到[0]种业务方法' (Note: The selected range contains [0] bus abnormal records, involving [0] business methods). The third section, '服务异常记录' (Service abnormal records), shows a table with 'providerService' and a quantity of 4. A note states: '提示: 所选范围内包含[4]条服务异常记录, 涉及到[1]种业务方法' (Note: The selected range contains [4] service abnormal records, involving [1] business methods).

图 5.7.2.2 查看选中

进入快照列表时，可以看到正常的的数据是绿色的，图 5.7.2.3。异常的数据是橙色的。

此时的数据是汇合到一起的，如果数据量特别大的时候，这种方查看方式非常的不直观。所以可以通过散点分布图中的“设置过滤”按钮，图 5.7.2.4。对图中的点进行过滤，去掉不需要的点。此时散点分布图中只剩下异常的点。



图 5.7.2.3 快照列表

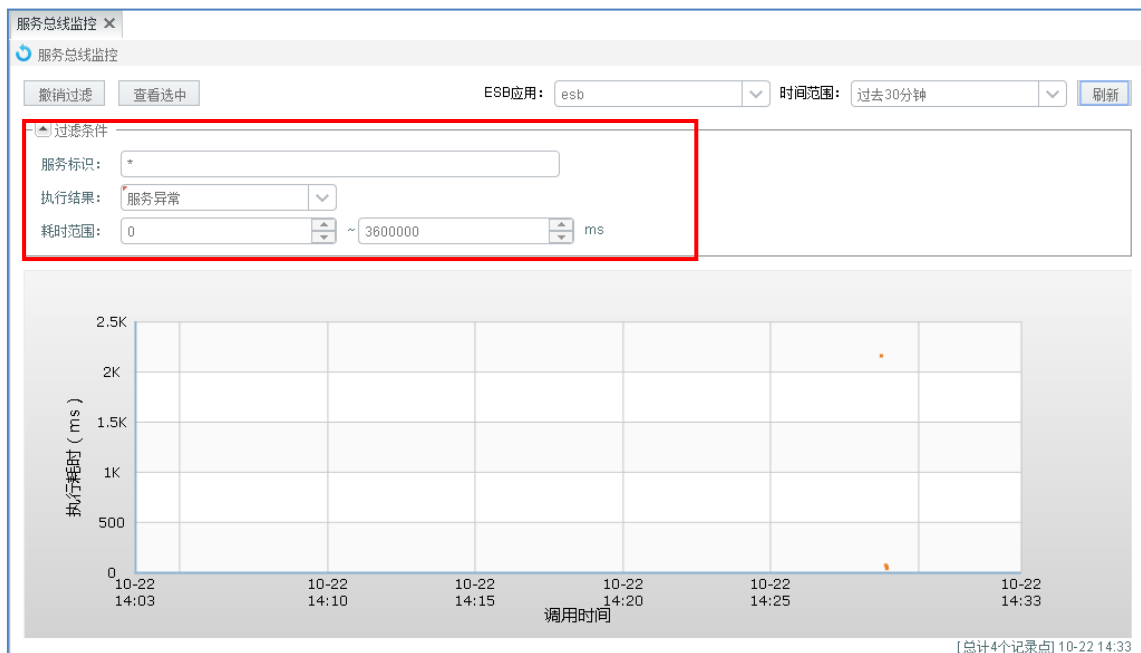


图 5.7.2.4 设置过滤

这时查看选中时，只有异常的点。图 5.7.2.5。查看快照列表时，也只有橙色的异常记录。图 5.7.2.6。

服务总线监控		
散点分布图 > 查看选中		
- 所有选中记录		
服务标识		数量
providerService		4
提示：共选中[4]条记录，涉及到[1]条SQL语句		
- 总线异常记录		
服务标识		数量
提示：所选范围内包含[0]条总线异常记录，涉及到[0]种业务方法		
- 服务异常记录		
服务标识		数量
providerService		4
提示：所选范围内包含[4]条服务异常记录，涉及到[1]种业务方法		

图 5.7.2.5 查看选中



图 5.7.2.6 异常记录

展开异常信息可以看到图 5.7.2.7 中，发生异常的方法是 `getName` 方法。点击“查看详情”可以看到方法的参数和返回值。因为发生了异常，所以返回值为空。



图 5.7.2.7 异常记录

点击 [详细错误信息]后可以查看具体详细的异常信息，图 5.7.2.8。此时出现的异常，问题在什么地方代码的行数等信息，完整的展示出来。这时如果想定位服务端的异常原

因，将会非常的准确。既节省人力物力，还节省时间。

【注意】：查看 [详细错误信息] 时，请务必使用火狐浏览器。否则可能会导致错误信息无法正确显示。

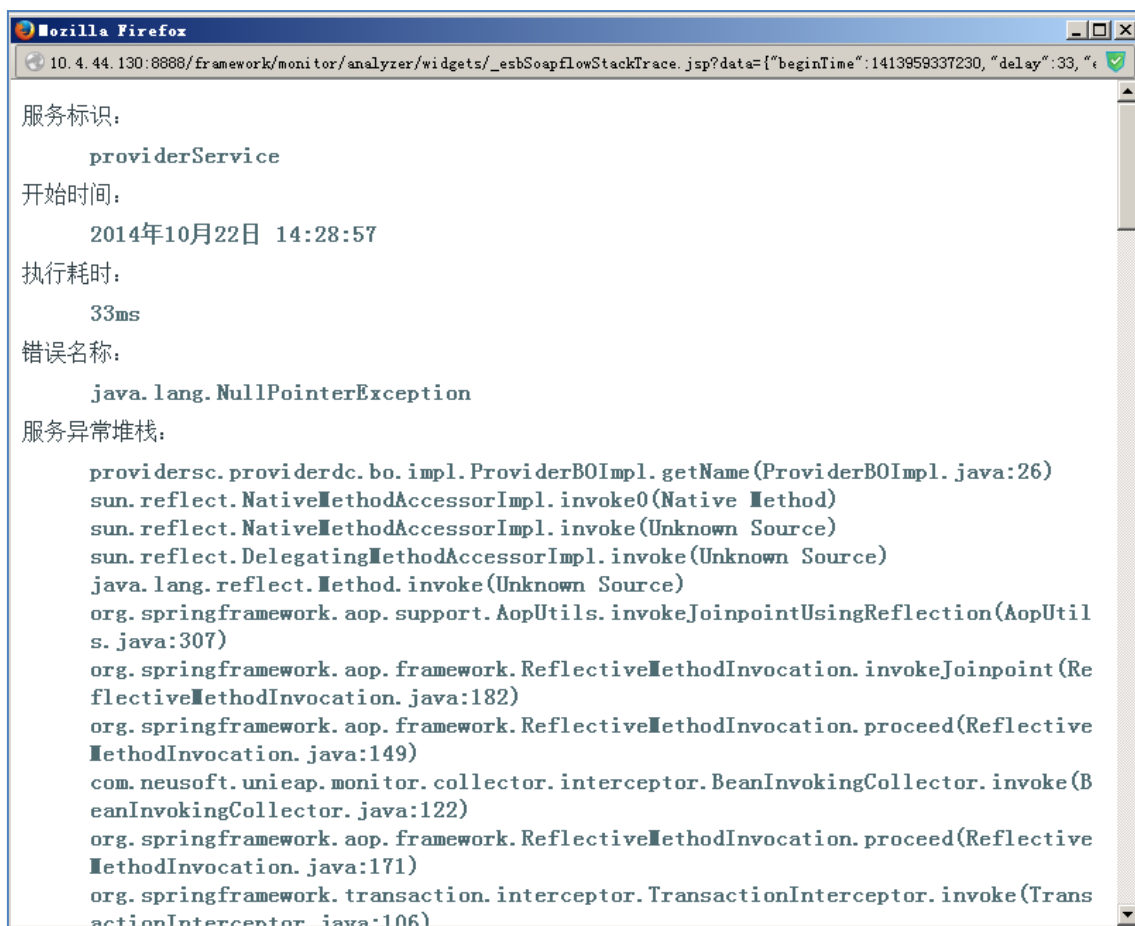


图 5.7.2.8 异常信息

5.7.3 总线异常监控记录

总线异常的记录产生，一般都是因为服务端没有启动或网络不联通所导致的。本例中将服务提供端停止，此时由于 ESB 端无法访问服务端，ESB 端会抛出总线异常。如图 5.7.3.1 。

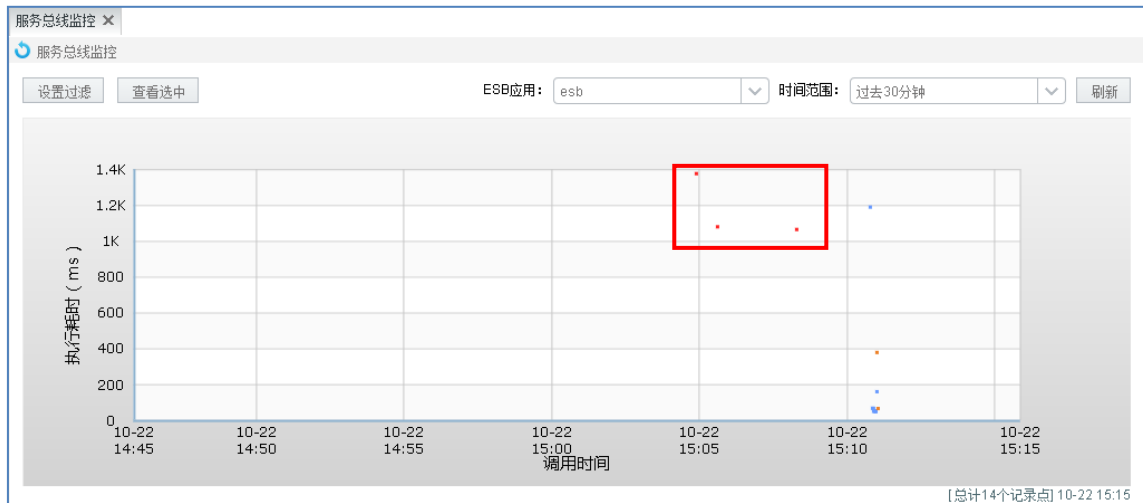


图 5.7.3.1 总线异常

选中这三个红点后, 可以看到在查看选中页面中出现了三条总线异常记录, 图 7.3.3.2。

Figure 5.7.3.2 is a detail view titled '服务总线监控' (Service Bus Monitoring). It shows three sections: '所有选中记录' (All selected records), '总线异常记录' (Bus anomaly records), and '服务异常记录' (Service anomaly records). Each section shows a table with '服务标识' (Service Identifier) and '数量' (Quantity). The '总线异常记录' section shows 3 records for providerService, with a hint: '提示: 共选中[3]条记录, 涉及到[1]条SQL语句'.

服务标识	数量
providerService	3
提示: 共选中[3]条记录, 涉及到[1]条SQL语句	
服务标识	数量
providerService	3
提示: 所选范围内包含[3]条总线异常记录, 涉及到[1]种业务方法	
服务标识	数量
	0
提示: 所选范围内包含[0]条服务异常记录, 涉及到[0]种业务方法	

图 5.7.3.2 总线异常

点击查看详情可以看到发生错误的访问地址, 和发生错误的运行结果。图 5.7.3.3。

	开始时间 ▼	执行耗时(ms)
+	2014-10-22 15:08:18	1063
+	2014-10-22 15:05:37	1078
-	2014-10-22 15:04:54	1375
服务请求方: 10.4.46.58 服务访问地址: http://10.4.46.58:8080/framework/ws/providersc/providerdc/provider?eap_username=admin&eap_password=1 完整调用树:  [点击查看] 运行结果:  发生总线异常!  [详细错误信息]		
总计[3]条记录。		

图 5.7.3.3 异常详细

点击详细错误信息，图 5.7.3.4。可以看到是因为连接 10.4.46.58:8080 被拒绝或失败。通过该信息，可以得知与 10.4.46.58:8080 连接不通畅。检查网络或检查 10.4.46.58:8080 服务端的服务是否仍在正常运行，即可解决问题。

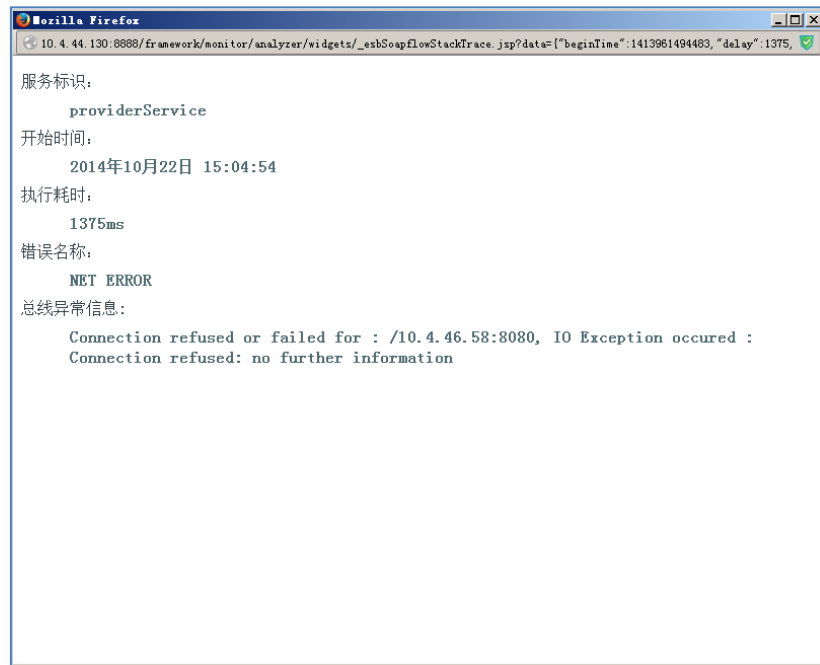


图 5.7.3.4 总线异常

第6章 日常维护

6.1 UniEAP ESB 启动

单机启动：在单机的情况下，因为 ESB 服务不依赖任何一个应用，所以不需要任何的修改及配置，只需将解压后的工程发布到一个可以正常启动的 Web 服务器中即可。发布后可直接启动。

集群启动：集群启动时，先参考单机服务的配置，然后再按集群环境的配置修改相应的配置文件，修改好后只需将工程发布到一个可以正常启动的 Web 服务器中即可。发布后可直接启动。

6.2 UniEAP ESB 停机

单机停机：关于 UniEAP ESB 服务停机，从系统健壮性角度来讲，由于之前所述原因（各元素间无强依赖），并无硬性规定。ESB 的 web 服务可以直接停止，但是停止服务之前，应确认调用 ESB 的相应服务调用端此时不需要调用的情况下进行。否则，会影响正常的调用。导致服务提供端出现异常。

集群停机：在集群的情况下，如果该节点不是唯一的节点，并且在没有服务调用端访问的情况下，关闭该节点，则不会对系统和服务调用端产生影响。

附录 A UniEAP Platform 服务发布实践

该附录内容涉及使用 UniEAP Platform 开发构建应用过程中，将其中的 BO 模型方法发布为远程调用服务的相关操作说明。目前，UniEAP Platform 提供了两种不同形式的 web 服务：SOAP 和 REST，可将业务逻辑方法（BO 方法）发布为 SOAP 服务或 REST 服务。以下分别作简单介绍，有关更详细的说明请参考《UniEAP Platform 用户手册》中相关章节。

A1. 发布 SOAP 服务

1. 右键点击 BO 模型，在菜单中选择“UniEAP 工具”->“一键发布 Webservice”。
2. 在该 DC 的 content/conf 文件夹下会自动生成名为 applicationContext-ws.xml 的文件。
3. 可以通过 CXFServlet 的映射地址，使用浏览器访问查看已发布的 SOAP 服务列表。地址如：[http://主机名 + 应用名 + “/ws”](http://主机名 + 应用名 +)。

附 applicationContext-ws 配置文件内容截图作简要说明。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx" xmlns:simple="http://cxf.apache.org/simple"
       xmlns:soap="http://cxf.apache.org/bindings/soap" xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://cxf.apache.org/bindings/soap http://cxf.apache.org/bindings/soap/schemas/soap.xsd" >
    <simple:server id="demoAF_chartBO_service"
        serviceClass="com.neusoft.test.demosc.demosc.bo.ChartBO" serviceBean="#demoAF_chartBO_bo"
        address="/demoSC/demoAF/chart">
        <simple:binding id="demoAF_chartBO_service"
            serviceClass="com.neusoft.test.demosc.demosc.bo.ChartBO" serviceBean="#demoAF_chartBO_bo"
            address="/demoSC/demoAF/chart">
            <simple:dataBinding>
                <simple:binding id="demoAF_chartBO_service"
                    serviceClass="com.neusoft.test.demosc.demosc.bo.ChartBO" serviceBean="#demoAF_chartBO_bo"
                    address="/demoSC/demoAF/chart">
                    <simple:binding id="demoAF_chartBO_service"
                        serviceClass="com.neusoft.test.demosc.demosc.bo.ChartBO" serviceBean="#demoAF_chartBO_bo"
                        address="/demoSC/demoAF/chart">
                    </simple:binding>
                </simple:binding>
            </simple:binding>
        </simple:binding>
    </simple:server>
    <simple:server id="demoAF_testBO_service"
        serviceClass="com.neusoft.test.demosc.demosc.bo.TestBO" serviceBean="#demoAF_testBO_bo"
        address="/demoSC/demoAF/test">
        <simple:binding id="demoAF_testBO_service"
            serviceClass="com.neusoft.test.demosc.demosc.bo.TestBO" serviceBean="#demoAF_testBO_bo"
            address="/demoSC/demoAF/test">
            <simple:dataBinding>
                <simple:binding id="demoAF_testBO_service"
                    serviceClass="com.neusoft.test.demosc.demosc.bo.TestBO" serviceBean="#demoAF_testBO_bo"
                    address="/demoSC/demoAF/test">
                    <simple:binding id="demoAF_testBO_service"
                        serviceClass="com.neusoft.test.demosc.demosc.bo.TestBO" serviceBean="#demoAF_testBO_bo"
                        address="/demoSC/demoAF/test">
                    </simple:binding>
                </simple:binding>
            </simple:binding>
        </simple:binding>
    </simple:server>
</beans>
```

Bo对应接口

与Bo生成Spring bean名字策略相同，将后缀变为_service

引用Bo生成Spring bean

Service调用地址，规则为：/sc/dc/bo模型的语义名字--去掉末尾"BO"

A2. 发布 REST 服务

只需要在 BO 模型实现类对应的方法上加入注释@RestService，该方法就可以通过 REST 远程调用了。如下图所示：

```
@RestService
public String getWorkFlowMethod(String arg1, String arg2) {
    // TODO Auto-generated method stub
    System.out.println("getWorkFlowMethod "+参数1: "+arg1+" "+参数2: "+arg2);
    return "1";
}
```

如果业务方法不加上此注释，通过 REST 访问时会抛出“EAPTECHRIA1002:请求方法不允许远程调用”的异常。

附录 B UniEAP Platform 服务调用实践

对于 UniEAP Platform 应用开发过程中面临的服务调用实践问题，应分情形对待。

如果使用了 ESB 组件，且被调用服务为 SOAP 服务。则应当如本文第 3 和第 4 章节所述，通过 ESB 组件进行服务注册与调用即可。（补充说明：目前的 ESB 组件版本暂不支持 REST 服务的注册管理、调用与监控。）

若非如此，则参照以下说明。

B1. 直接调用 SOAP 服务

举例说明，假设需要调用 TestBO 的 print 方法来打印用户列表，BO 代码如下：

```
public interface TestBO
{
    public void print(List<User> userList);
}

@ModelFile(value = "testBO.bo")
public class TestBOImpl implements TestBO
{
    @Override
    public void print(List<User> userList)
    {
        if (userList != null && userList.size() > 0)
        {
            for (User user : userList)
            {
                System.out.println("User name is :" + user.getName());
            }
        }
    }
}
```



```
public class User
{
    private String name;

    public String getName()
    {
        return name;
    }

    public void setName(String name)
    {
        this.name = name;
    }
}
```

调用该 BO 对应的 Webservice 的代码为如下：

```
public static void main(String[] args)
{
    ClientProxyFactoryBean factory = new ClientProxyFactoryBean();
    factory.setServiceClass(TestBO.class);
    factory.setAddress("http://localhost:8090/framework/
ws/demoSC/demoAF/test?eap_username=admin&eap_password=1");
}
```

```

/*注：需要注意的是，客户端调用的地址配置为：主机名 + 应用名 + CXFServlet
的映射地址 + 相对地址（可在applicationContext-ws中的对应address属性）。
*/

factory.getServiceFactory().setDataBinding(new
AegisDataBinding());// 指定aegis绑定

TestBO service = (TestBO) factory.create();

List<User> userList = new ArrayList<User>();

User user1 = new User();

user1.setName("aa");

User user2 = new User();

user2.setName("bb");

userList.add(user1);

userList.add(user2);

service.print(userList);

}

```

【注意】客户端调用地址后面必须拼接上用户名密码，调用请求才能通过安全认证。

B2. REST 服务直接调用

1、数据交换协议

对资源 “/ws/rest/” 发送的 HTTP Request 请求须涵盖一下几项参数：

参数名	含义
boId	要调用的 boId。型如： <i>demoDC_demoProjectBO_bo。</i>
methodName	BO 的方法名。型如： <i>testMethod</i>
parameters	参数信息。型如： <i>[{string:' aaa' }, {int:' 123' }, {string:{a:1,b:2}}]</i>
returnType	返回数据的格式。参看资源访问成功返回结果说明。

paraTable	可选参数。型如： <i>com.neusoft.example.entity</i> , <i>java.lang.String</i> 。该项参数内容对应了 BO 方法的参数表。当 parameters 中的参数信息不足以定位 BO 方法时（比如存在重载函数时），可使用该参数来确定 BO 方法。
-----------	---

若资源访问成功，HTTP 请求状态行将标识为 200 OK。Response 依据请求参数中的 returnType 值分为如下形式：

returnType 值	含义
<i>string</i>	返回 Plain text 结果；
<i>dataCenter</i>	返回 DataCenter 格式的数据；
<i>json</i>	返回 Json 格式的数据；

若资源访问失败，HTTP 请求状态行中将标识 40X 状态码，Response 中附带具体失败信息。

2、参考样例

样例场景描述：请求 ID 为“testDC_testBO_bo”的 BO 中的“simpleMethod”方法。该方法已被注解@RestService，可通过通用 REST 接口进行访问。它接受一个 String 类型参数，返回的 java 类型为 BOContext。

样例 1：GET 请求。

a) 使用 GET 方法，要求返回 Json 格式数据，请求与响应内容如下所示。

Request Raw:

```
GET
http://localhost:9090/framework/ws/rest/?boId=testDC_testBO_bo&methodName=simpleMethod&returnType=json&parameters=[{String:'test'}] HTTP/1.1
Host: localhost:9090
```

Request Raw:

```
HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Date: Fri, 09 Aug 2013 09:35:41 GMT

Content-Type: application/json;charset=UTF-8

Content-Length: 42

{"pass2":false,"pass1":true,"pass3":false}
```

- b) 使用 GET 方法，要求返回 string 格式数据，请求与响应内容如下所示。

Request Raw:

```
GET
http://localhost:9090/framework/ws/rest/?boId=testDC_testBO_bo&methodName=simpleMethod&returnType=string&parameters=[{String:'test'}] HTTP/1.1
Host: localhost:9090
```

Request Raw:

```
HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Content-Type: text/xml;charset=UTF-8

Content-Length: 38

Date: Fri, 09 Aug 2013 09:42:20 GMT

{pass2=false, pass1=true, pass3=false}
```

- c) 使用 GET 方法，要求返回 dataCenter 格式数据，请求与响应内容如下所示。

Request Raw:

```
GET
http://localhost:9090/framework/ws/rest/?boId=testDC_testBO_bo&methodName=sim
```

```
pleMethod&returnType=dataCenter&parameters=[{String:'test'}] HTTP/1.1
Host: localhost:9090
```

Response Raw:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=UTF-8
Content-Length: 132
Date: Fri, 09 Aug 2013 09:46:07 GMT
{header:{"code":0,"message":{"title":"","detail":""}},body :{parameters :{"pass2":false,"pass1":true,"pass3":false},dataStores :{}}}
```

样例 2: POST 请求。

使用 POST 方法，且通过 URL 传递参数的情形与 GET 方法相似。以返回 Json 格式数据为例，如下所示。

注意，使用 POST 方法，且欲在 Message Body 中传递参数时，请参照“样例 3”。

Request Raw:

```
POST
http://localhost:9090/framework/ws/rest/?boId=testDC_testBO_bo&methodName=simpleMethod&returnType=dataCenter&parameters=[{String:'test'}] HTTP/1.1
Host: localhost:9090
```

Response Raw:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Date: Fri, 09 Aug 2013 09:51:17 GMT
Content-Type: application/json;charset=UTF-8
```

```
Content-Length: 42  
  
{"pass2":false,"pass1":true,"pass3":false}
```

样例 3: POST 请求的 Message Body。

若使用 POST 方法，且欲在 Message Body 中传递参数，那么需要将参数内容写成 DataCenter 格式，请求与响应内容如下所示。

Request Raw:

```
POST http://localhost:9090/framework/ws /rest/ HTTP/1.1  
Host: localhost:9090  
Connection: keep-alive  
Content-Length: 201  
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.17 (KHTML, like Gecko)  
Chrome/24.0.1312.56 Safari/537.17  
Accept: */*  
Accept-Encoding: gzip,deflate,sdch  
Accept-Language: zh-CN,zh;q=0.8  
Accept-Charset: GBK,utf-8;q=0.7,*;q=0.3  
Cookie: JSESSIONID=9488F7F9B97E2E8C52B8C91D1E89AFCF  
  
{header:{"code":0,"message":{"title":"","detail":""}},body:{dataStores:{},parameters:{"parameters":[{"String:'test'}],"bold":"testDC_testBO_bo","methodName":"simpleMethod","returnType":"dataCenter"}}}
```

Response Raw:

```
HTTP/1.1 200 OK
```

Server: Apache-Coyote/1.1

Content-Type: text/xml;charset=UTF-8

Content-Length: 132

Date: Fri, 09 Aug 2013 09:04:59 GMT

```
{header:{"code":0,"message":{"title":"","detail":""}},body:{parameters :{"pass2":false,"pass1":true,"pass3":false},dataStores :{}}}
```

DataCenter 的结构为：

```
{
  header: {...},
  body:{
    parameters: {...},
    dataStroes: {...}
  }
}
```

在本例场景中，只需将请求参数以 json 格式写在 parameters 中即可。

3、参数格式

JAVA 基本类型

{<primitive-type-name>:<value>}

例如：{int:99}、{char:"c"}、{boolean:"true"}

JAVA 封装类型

{<object-type-name>:<value>}

例如：{java.lang.String:"str"}

Pojo 类型

```
{
  pojo:{
    <className>:{
      <attribute-name>:<attribute-value>,
      <attribute-name>:<attribute-value>,
      ... ..
      <attribute-name>:<attribute-value>
    }
  }
}
```

例如：

```
{
  pojo:{
    testsc.testdc.dto.Test:{
      attr1:"str1",
      attr2:"str2"
    }
  }
}
```

PojoList 类型

```
{
```



```

pojoList:{
  <className>:[
    {
      <attribute-name>:<attribute-value>,
      <attribute-name>:<attribute-value>,
      ... ..
      <attribute-name>:<attribute-value>
    },
    {
      <attribute-name>:<attribute-value>,
      <attribute-name>:<attribute-value>,
      ... ..
      <attribute-name>:<attribute-value>
    },
    ... ..
    {
      <attribute-name>:<attribute-value>,
      <attribute-name>:<attribute-value>,
      ... ..
      <attribute-name>:<attribute-value>
    }
  ]
}

```

例如：

```
{pojoList:{testsc.testdc.dto.Test:[{t1:"t1",t2:"t2"},{t1:"t21",t2:"t22"}]}}
```

Map 类型

```
{
  map:{
    <key-name>:<key-value>,
    <key-name>:<key-value>,
    ... ..
    <key-name>:<key-value>
  }
}
```

例如：

```
{map:{k1:"k1-value",k2:"k2-value"}}
```

4、附录：编码样例

使用 `unieap.Action.requestData` 在浏览器中发送 ajax 请求的 javascript 代码片段示例如下：

```
unieap.Action.requestData({
  url:" /framework/ws/rest/",
  parameters:{
    bold:"testDC_testBO_bo",
    methodName:"simpleMethod",
    returnType:"json",
    parameters:"[{String:'hello'}]"},
  load:function(dc){console.log(dc)}
```

}}