IONA®

Artix ESB

Progress Actional Integration Guide

Version 5.1
March 2008

Making Software Work Together™

Progress Actional Integration Guide

IONA Technologies

Version 5.1

Published 29 Jul 2008
Copyright © 2001-2008 IONA Technologies PLC

Table of Contents

Preface **7**
- What is Covered in This Book 8
- Who Should Read This Book 9
- How to Use This Book 10
- The Artix ESB Documentation Library 11

Artix–Actional Integration **13**
- Artix–Actional Integration Architecture 14

Configuring Artix–Actional Integration **21**
- Prerequisites 22
- Configuring Artix Java Services for Actional Integration 23

Monitoring Artix Endpoints with Actional **27**
- Viewing Artix Endpoints in Actional 28

Index 33

List of Figures

1. Artix–Actional Integration Architecture 15
2. Artix Interception Points 17
3. Actional Server Administration Console 19
4. Actional Server Network Overview 29
5. Node details 30
6. Path Explorer 31
7. Path Explorer Details 32

Preface

What is Covered in This Book 8
Who Should Read This Book 9
How to Use This Book 10
The Artix ESB Documentation Library 11

What is Covered in This Book

Artix ESB supports integration with the following Progress Actional SOA management products:

- Actional for SOA Operations
- Actional Continuous Service Optimization (Actional CSO)

This guide explains how to enable Artix ESB Java solutions to be monitored by these Actional products. This guide applies to Artix ESB applications written in JAX-WS (Java APIs for XML-Based Web Services) and JavaScript.

Who Should Read This Book

This guide is aimed at system administrators using Actional to monitor SOA environments, system architects designing SOA environments, and developers writing SOA applications with Artix ESB. System administrators do not require detailed knowledge of the technology that is used to create distributed enterprise applications.

This book assumes that you already have a working knowledge of Actional SOA management products. For more information, see http://www.actional.com.

How to Use This Book

This book is organized into the following chapters:

Chapter 1 on page 13 describes the architecture of the Artix integration with Actional.

Chapter 2 on page 21 explains how to configure the Artix integration with Actional, showing examples from an Artix–Actional integration demo.

Chapter 3 on page 27 shows examples of viewing Artix endpoints in Actional.

The Artix ESB Documentation Library

For information on the organization of the Artix ESB library, the document conventions used, and where to find additional resources, see Using the Artix ESB Library [http://www.iona.com/support/docs/artix/5.1/library_intro/index.htm].

Artix–Actional Integration

Artix ESB provides support for integration with Progress Actional SOA management products. This explains how this integration works and describes the main components.

Artix–Actional Integration Architecture 14

Artix–Actional Integration Architecture

Overview

Integration between Artix and Actional enables Artix services to be monitored by Actional SOA management products. For example, you can use Actional SOA management tools to perform monitoring, auditing, and reporting on Artix services. You can also correlate and track messages through your network to perform dependency mapping and root cause analysis.

The Artix–Actional integration is deployed on Artix service endpoints to enable reporting of management data back to the Actional server. The data reported back to Actional includes system administration metrics such as response time, fault location, auditing, and alerts based on policies and rules.

The Artix–Actional integration can be used with Artix Web service applications implemented in JAX-WS and JavaScript.

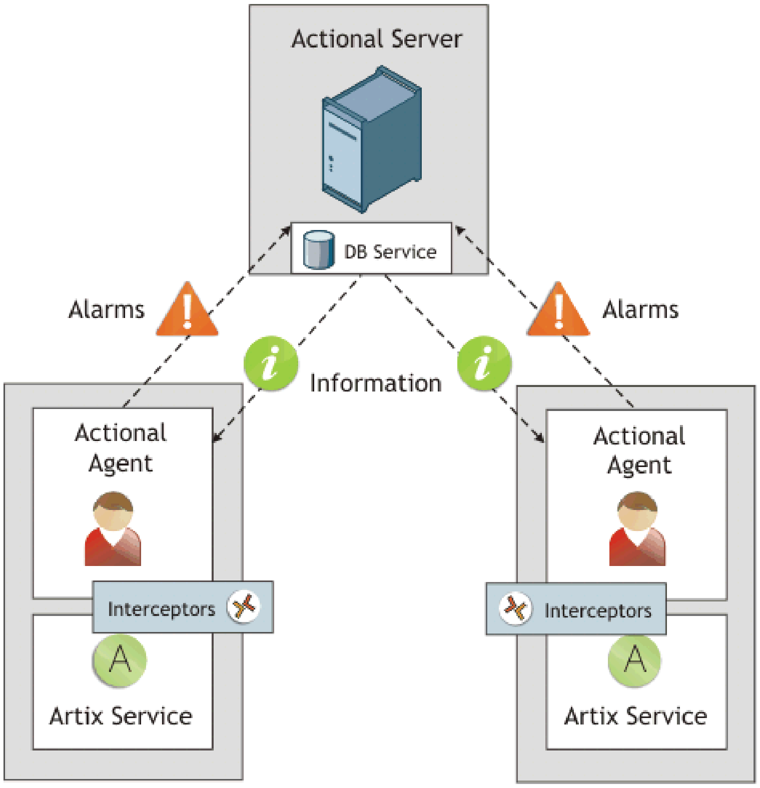
Integration architecture

The Actional SOA management system includes an Actional server and an Actional agent. An Actional agent is run on each Artix service endpoint node that you wish to manage.

The Artix service endpoint to be managed by Actional uses Actional's interceptor API to send monitoring data to the Actional agent. The Actional server pings the Actional agent periodically to retrieve the monitoring data. It analyzes this data and represents it in the Actional SOA management GUI tools. In addition, any alarms triggered at the Actional agent are sent immediately to the Actional server.

Figure 1 on page 15 shows how Artix Web service applications are integrated with Actional using this architecture.

Figure 1. Artix–Actional Integration Architecture



The main components in this architecture are:

- Actional server on page 16
- Actional agent on page 16
- Artix interceptors on page 16
- Actional agent interceptor API on page 18
- Artix service endpoint on page 18

- Service consumers on page 18

Actional server

The Actional server is a central management server that manages service endpoint nodes containing an Actional agent. The Actional server hosts a database and pings Actional agents to obtain management data at configured time intervals. It analyzes the management data and displays it in an Actional console; for example, the Actional Server Administration Console. This console is a Web application deployed on Apache Tomcat. It has a runtime management mode and agent configuration mode (for example, for setting up policies). By default, the Actional server uses port 4040. The default Actional server database is Apache Derby

Actional agent

An Actional agent is run on each Artix service endpoint node that you wish to manage. Actional agents provides instrumentation data to the Actional server. Actional agents are provisioned from the Actional server to establish initial contact and send configuration to the Actional agent. There is one Actional agent per service endpoint node. By default, the Actional agent uses port `4041`.

Artix interceptors

At the level of an endpoint node, the Artix interceptors send the instrumentation data to the Actional agent using an Actional-specific API. These interceptors essentially push events to the Actional agent. This data is analyzed and stored in the Actional agent for retrieval later by the Actional server. However, any alarms triggered at the Actional agent are sent immediately to the Actional server.

Figure 2 on page 17 shows the flow of information at the Artix interceptor points.

Figure 2. Artix Interception Points

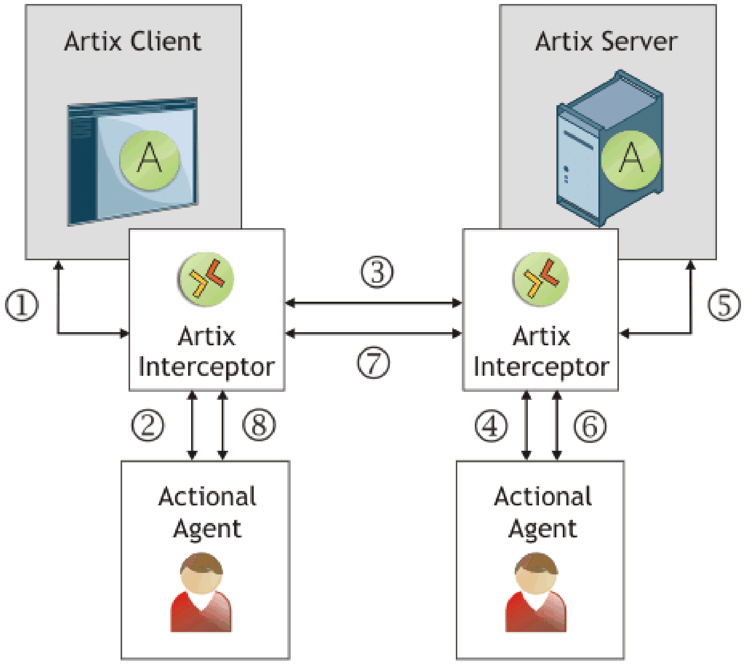


Figure 2 on page 17 shows the flow of information at the Artix interceptor points.

1. The outbound client interceptor is invoked, which starts a client interaction, and records the outgoing message. All other management data, such as the service, port, and operation names are stored. The correlation ID used to track the message is assigned in the transport, and the client interaction is marked as analyzed.
2. All the management data is sent to the local client-side Actional agent.
3. The outbound client interceptor sends the management data to the server interceptor.
4. The inbound server interceptor receives the request, starts a server interaction, and the correlation ID is fetched from the transport. All

management data is set on the server side, and the interaction is marked as analyzed. For one-way calls, the interaction is marked as ended..

5. This shows the interaction with the Artix server.
6. All the management data is transmitted to the local server-side Actional agent.
7. The outbound server interceptor point mimics the outbound client interceptor, and sends the management data to the client interceptor.
8. The inbound client interceptor mimics the inbound server interceptor.

Actional agent interceptor API

The Actional Agent Interceptor SDK is an Actional-specific API used to send the management instrumentation data from the endpoint to the Actional agent. The Artix service application to be managed by Actional must use the Actional Agent Interceptor SDK to send monitoring data to the Actional agent.

For detailed information on how to use this API, see the Actional product documentation.

Artix service endpoint

An Artix service endpoint is a service built using Artix, and described using WSDL. The endpoint can be implemented using JAX-WS, or a scripting language such as JavaScript. However, the main characteristic of an Artix service endpoint is that it can be described in WSDL, and classified as a service, which can be consumed.

Service consumers

Service consumers are clients that consume service endpoints by exchanging messages based on the service interface. Consumers can be built using Artix, or any product that supports the technology used by the endpoint. For example, a pure CORBA client could be a consumer for a CORBA endpoint. A .NET client could be a consumer for an Artix SOAP endpoint.

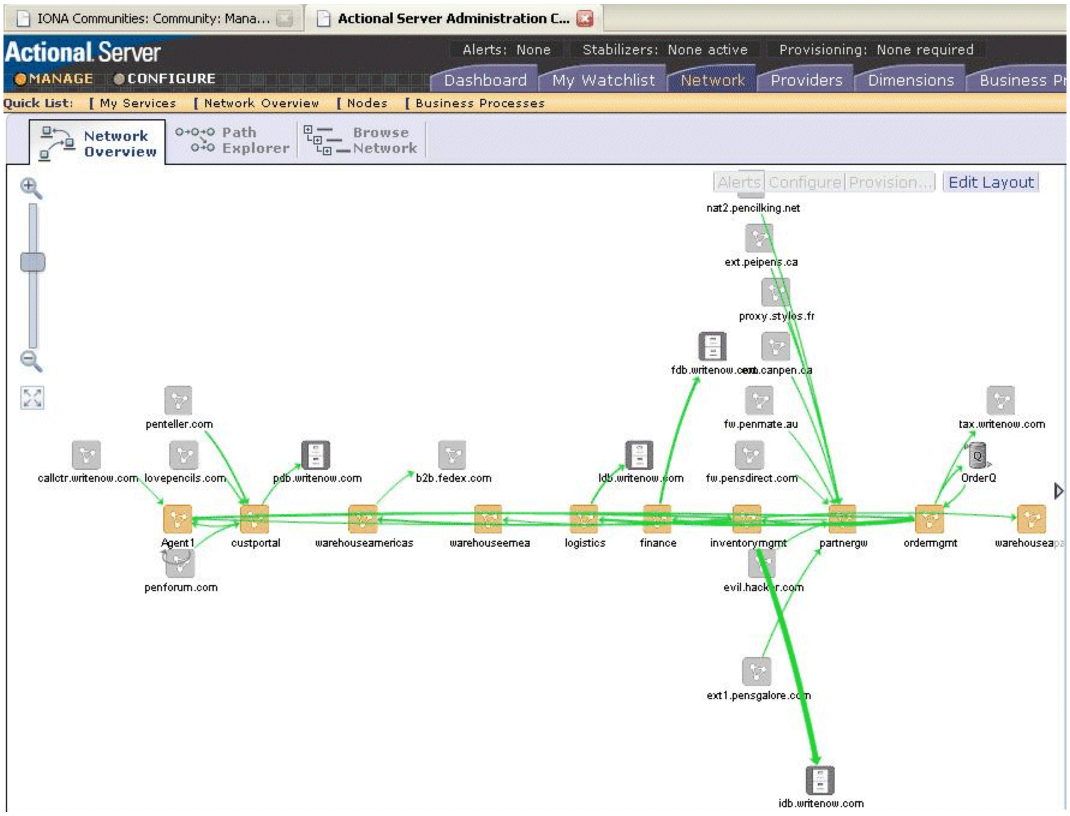
Actional SOA management system

In this document, Actional is the general term used to describe the Actional SOA management system in which all data is stored and viewed. This simplifies the architecture of Actional for the sake of this discussion.

Figure 3 on page 19 shows an example of the Actional Server Administration Console. Managed endpoint nodes are displayed as orange boxes, and unmanaged nodes are displayed as grey boxes. The green arrow indicates the message flow through various nodes.

Clicking on each of the nodes shows more in-depth information regarding the response time, alarms and warnings, and so on. The organization of the information in this web console is in the form of *Node–Group–Service–Operation*. In Artix, this translates to *Node–Service–Port–Operation*.

Figure 3. Actional Server Administration Console



Further information

For detailed information on using Actional features, see the Actional product documentation.

Configuring Artix–Actional Integration

This chapter explains how to configure integration between Artix and Actional SOA management products. It shows examples from an Artix–Actional integration demo.

Prerequisites 22
Configuring Artix Java Services for Actional Integration 23

Prerequisites

Overview

This section describes prerequisites for integration between Artix and Actional SOA management products.

Supported product versions

You must have the following version of Artix installed:

- Artix ESB 5.1—patch level 20080326

This version of Artix supports integration with the following Actional product versions:

- Actional for SOA Operations—version 7.0.1 or 7.1
- Actional Continuous Service Optimization (Actional CSO)—version 7.1

Supported transports and protocols

The following protocols and transports are supported:

- SOAP over HTTP
- SOAP over JMS
- XML over HTTP
- XML over JMS
- CORBA

Actional agents

You must ensure that Actional agents have been set up on each Artix service endpoint node that you wish to manage. The provisioning of Actional agents is performed using the Actional server.

For information on how to set up Actional agents on endpoint nodes, see the Actional product documentation.

Further information

In addition, for information on the full range of platform versions and database versions supported by Actional, see the Actional product documentation.

Configuring Artix Java Services for Actional Integration

Overview

This section explains how to configure Artix services written using JAX-WS for integration with Actional. It shows examples of Artix XML configuration from the Artix–Actional integration demo. For information on how to run the demo, see the readme.txt file in the following directory:

```
ArtixInstallDir/Version/java/samples/management/action
al/jms_queues
```

This demo is based on `.../java/samples/transports/jms_queue/`, with some modifications to illustrate Artix and Actional integration, and with two JMS queues instead of one.

Service endpoint configuration

The Artix Java configuration mechanism is based on the XML-based Spring Framework. The following example from the `server1.xml` file in the Artix `jms_queues` demo shows the XML configuration used by the Artix service endpoint:

```
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:jaxws="http://cxf.apache.org/jaxws"
 xmlns:soap="http://cxf.apache.org/bindings/soap"
 xsi:schemaLocation="http://www.springframe
work.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-
2.0.xsd
 http://cxf.apache.org/bindings/soap
 http://cxf.apache.org/schema/bindings/soap.xsd
 http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jax
ws.xsd"
 xmlns:mgmt="http://www.iona.com/management/actional/">

<bean name="{http://apache.org/hello_world_soap_http}SOAPSer
vice" abstract="true">
  <property name="properties">
    <map>
      <entry key="schema-validation-enabled" value="true"/>
    </map>
  </property>
</bean>

<jaxws:endpoint name="{ht
tp://cxf.apache.org/jms_greeter}GreeterPort1" created
FromAPI="true">
```

```
  <jaxws:properties>
    <entry key="schema-validation-enabled" value="true"/>
  </jaxws:properties>
  <jaxws:features>
     <mgmt:management capturePayload="true"/>
   </jaxws:features>
</jaxws:endpoint>
</beans>
```

This example shows how an Artix service endpoint named `GreeterPort1` is configured for Artix and Actional integration using the `jaxws:features` element. The `mgmt:management capturePayload` attribute must be set to `true` to enable Artix and Actional integration.

Service consumer configuration

The following example from the `client.xml` file in the Artix `jms_queues` demo shows the client-side configuration:

```
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:http="http://cxf.apache.org/transports/http/configura
tion"
 xmlns:jaxws="http://cxf.apache.org/jaxws"
 xmlns:soap="http://cxf.apache.org/bindings/soap"
 xsi:schemaLocation="http://cxf.apache.org/transports/http/con
figuration
 http://cxf.apache.org/schemas/configuration/http-conf.xsd
 http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-
2.0.xsd
 http://cxf.apache.org/bindings/soap
 http://cxf.apache.org/schema/bindings/soap.xsd
 http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jax
ws.xsd"
 xmlns:mgmt="http://www.iona.com/management/actional/">

<http:conduit name="{http://apache.org/hello_world_soap_ht
tp}SoapPort9001.http-conduit">
  <http:client DecoupledEndpoint="http://localhost:9990/de
coupled_endpoint"/>
</http:conduit>

<bean name="{http://apache.org/hello_world_soap_http}SOAPSer
vice" abstract="true">
  <property name="properties">
   <map>
      <entry key="schema-validation-enabled" value="true"/>
   </map>
  </property>
</bean>

<jaxws:client name="{http://cxf.apache.org/jms_greeter}Greeter
Port1" createdFromAPI="true">
  <jaxws:features>
    <mgmt:management capturePayload="true"/>
   </jaxws:features>
 </jaxws:client>

<jaxws:client name="{http://cxf.apache.org/jms_greeter}Greeter
Port2" createdFromAPI="true">
  <jaxws:features>
     <mgmt:management capturePayload="true"/>
   </jaxws:features>
</jaxws:client>
</beans>
```

Like on the server-side, the `mgmt:management capturePayload` attribute must be set to true to enable Artix and Actional integration. This server and client side configuration enables the appropriate interceptors to be loaded into the Artix Java runtime to transmit the monitoring information to the Actional agent.

Accessing Artix Java configuration

You can make your Artix Java configuration available to the Artix Java runtime in one of the following ways:

- Specify the XML configuration file on your `CLASSPATH`.
- Programmatically, by creating a bus and passing the configuration file location as either a URL or string, as follows:

  ```
  (new SpringBusFactory()).createBus(URL myCfgURL)
  ```

  ```
  (new SpringBusFactory()).createBus(String myCfgResource)
  ```

- Use one of the following command-line arguments to point to your XML configuration file:

  ```
  -Dcxf.config.file.url=myCfgURL
  ```

  ```
  -Dcxf.config.file=myCfgResource
  ```

This enables you to save your XML configuration file anywhere on your system and avoid adding it to your `CLASSPATH`.

Example commands

The following example command is used to start a server:

```
start java -Djava.util.logging.config.file=%CXF_HOME%\etc\log
ging.properties -Dcxf.config.file=server.xml demo.hw.serv
er.Server
```

The following example command is used to start a client:

```
start java -Djava.util.logging.config.file=%CXF_HOME%\etc\log
ging.properties -Dcxf.config.file=client.xml demo.hw.client.Cli
ent .\wsdl\hello_world.wsdl
```

Further information

For information on how to set up and run the Actional server, Actional agent, and Actional Server Administration Console, see the Actional product documentation.

For information on Artix Java configuration, see the following:

- *Configuring and Deploying Artix Solutions, Java Runtime* (http://www.iona.com/support/docs/artix/5.1/deploy/java/index.htm)
- *Artix Configuration Reference, Java Runtime* (http://www.iona.com/support/docs/artix/5.1/config_ref/java/index.html)

For information on the Spring Framework, see www.springframework.org.

Monitoring Artix Endpoints with Actional

This chapter shows examples of monitoring Artix service endpoints and consumers in Actional SOA management tools

Viewing Artix Endpoints in Actional 28

Viewing Artix Endpoints in Actional

Overview

When your Artix service endpoints and consumers have been configured for integration with Actional, they can be viewed in Actional SOA management tools.

For example, when you run the Artix–Actional `jms_queues` demo, the Actional Server Administration Console displays the server queues and agent nodes. Monitoring information such as response times is displayed as green arrows, while alarms are displayed as red arrows, flowing to and from the queues. The implementation for `Server2` includes a delayed response, which can also be viewed in the console.

Network overview

Figure 4 on page 29 shows a running `jms_queues` demo displayed in the Network Overview screen of the Actional Server Administration Console.

Figure 4. Actional Server Network Overview

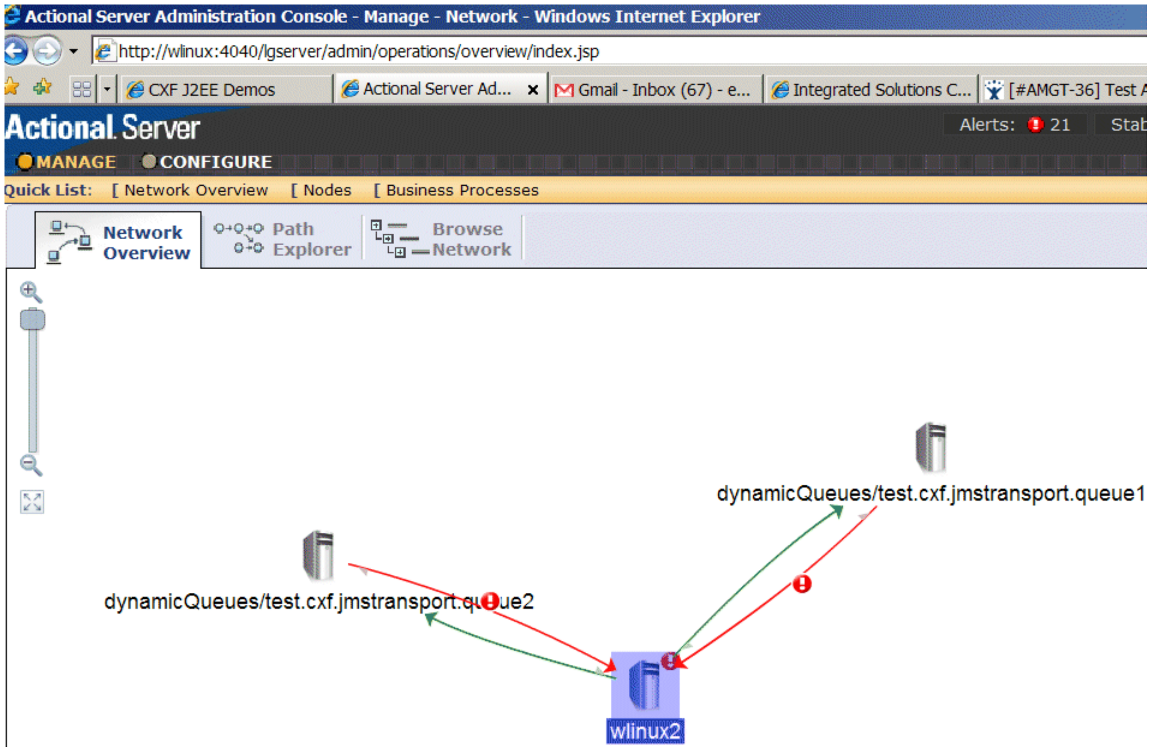
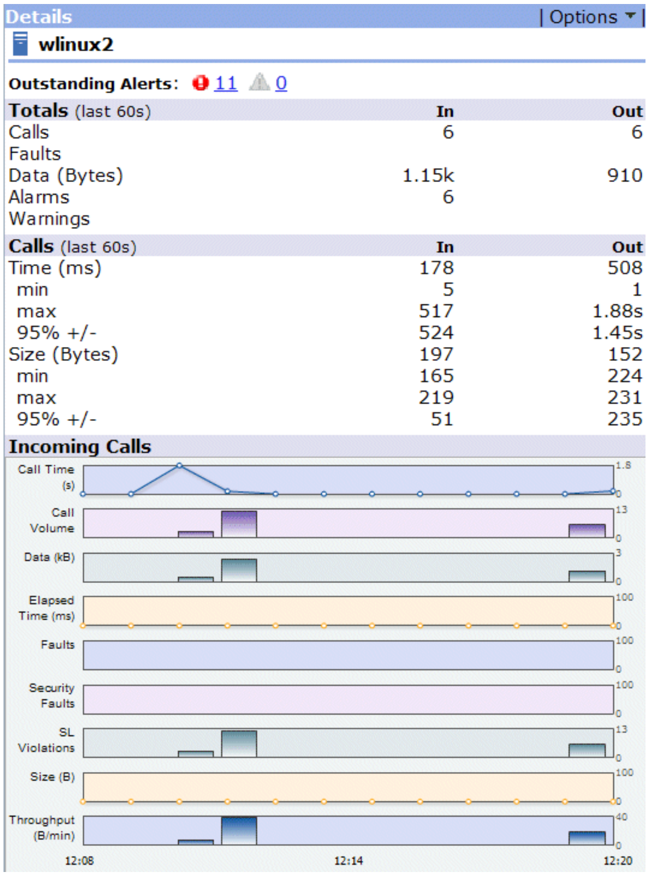

Actional Server Administration Console - Manage - Network - Windows Internet Explorer
http://wlinux:4040/lgserver/admin/operations/overview/index.jsp
CXF J2EE Demos
Actional Server Ad...
Gmail - Inbox (67) - e...
Integrated Solutions C...
[#AMGT-36] Test A
Actional Server
Alerts: 21
Stat
MANAGE
CONFIGURE
Quick List: [Network Overview [Nodes [Business Processes
Network Overview
Path Explorer
Browse Network
dynamicQueues/test.cxf.jmstransport.queue1
dynamicQueues/test.cxf.jmstransport.queue2
wlinux2

Figure 5 on page 30 shows the Details displayed for the example `wlinux2` node selected in Figure 4 on page 29.

Figure 5. Node details



wlinux2

Outstanding Alerts: 11 0

Totals (last 60s)	**In**	**Out**
Calls	6	6
Faults		
Data (Bytes)	1.15k	910
Alarms	6	
Warnings		

Calls (last 60s)	**In**	**Out**
Time (ms)	178	508
min	5	1
max	517	1.88s
95% +/-	524	1.45s
Size (Bytes)	197	152
min	165	224
max	219	231
95% +/-	51	235

Path Explorer

Figure 6 on page 31 shows the path displayed for the example queue1 in the Path Explorer view.

Figure 6. Path Explorer

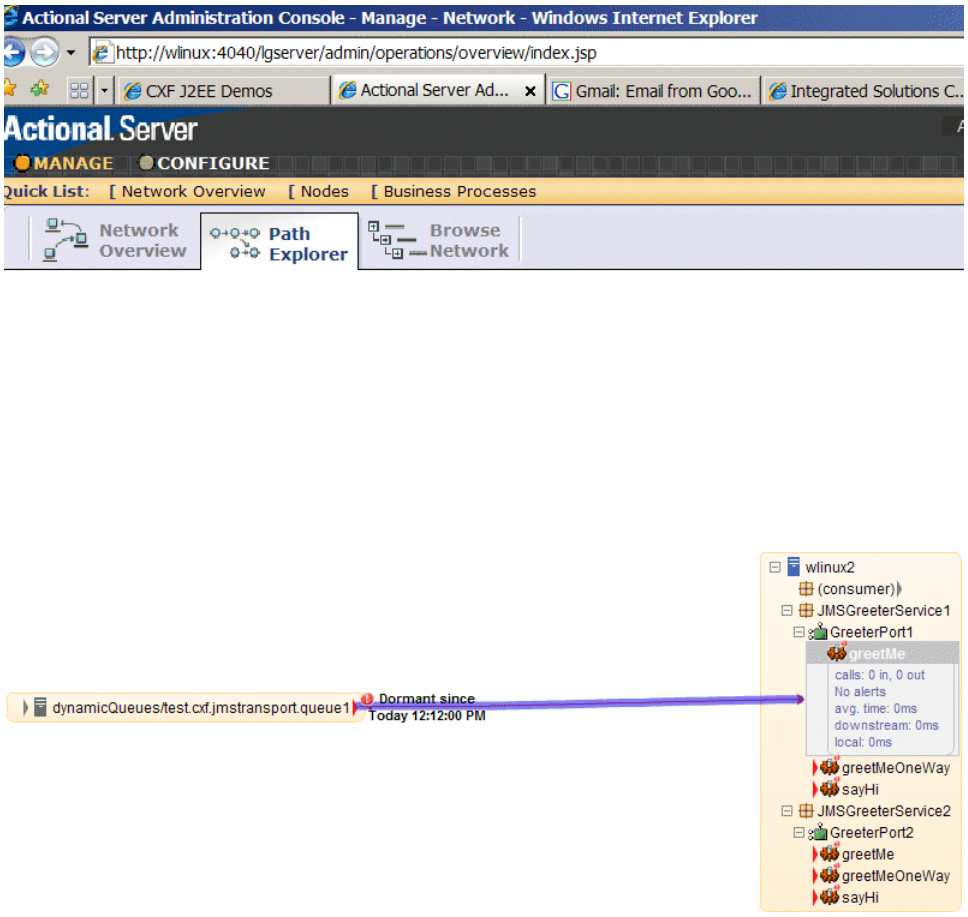
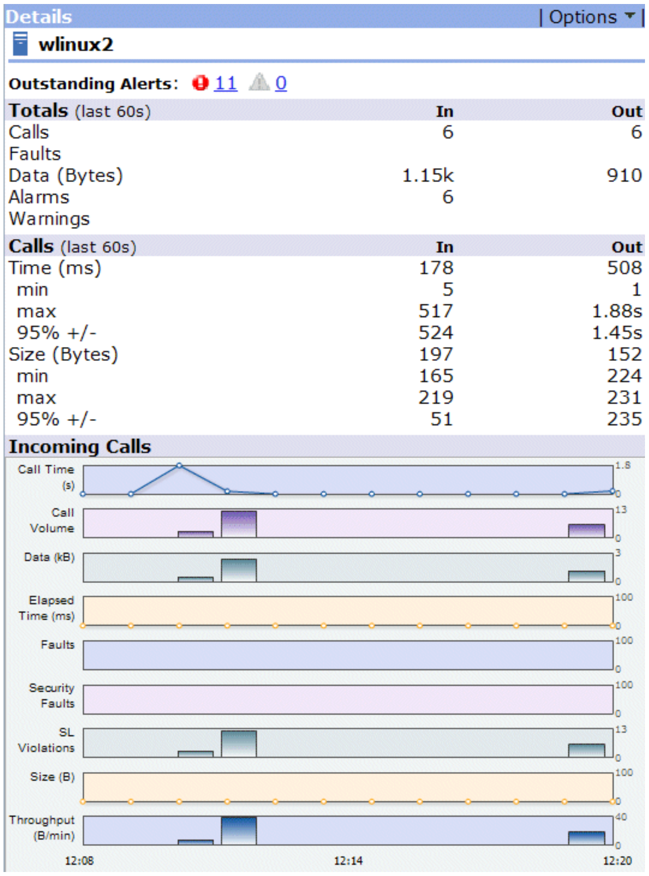


Figure 7 on page 32 shows the Details displayed for the example `wlinux2` node selected in Figure 4 on page 29.

Figure 7. Path Explorer Details



Details | Options ▾ |

wlinux2

Outstanding Alerts: ❗ 11 ⚠ 0

Totals (last 60s)	**In**	**Out**
Calls	6	6
Faults		
Data (Bytes)	1.15k	910
Alarms	6	
Warnings		

Calls (last 60s)	**In**	**Out**
Time (ms)	178	508
min	5	1
max	517	1.88s
95% +/-	524	1.45s
Size (Bytes)	197	152
min	165	224
max	219	231
95% +/-	51	235

Incoming Calls

Index

A
Actional agent, 16
Actional agent interceptor API, 18
Actional agents, 22
Actional Continuous Service Optimization, 8, 22
Actional for SOA Operations, 8, 22
Actional server, 16
Actional Server Administration Console, 18, 28
Actional SOA management system, 18
Apache Derby, 16
architecture, 14
Artix interceptors, 16
Artix service endpoint, 18

C
CLASSPATH, 25
client.xml, 24
configuration commands, 26
consumers, 18
CORBA, 22
correlation ID, 16
createBus(), 25

D
database, 16

E
endpoint, 18

I
interceptor API, 18
interceptors, 16

J
JavaScript, 14
JAX-WS, 14, 23

N
Network overview, 28
Node details, 28

P
Path Explorer, 30

S
server.xml, 23
service consumer, 24
service consumers, 18
service endpoint, 18, 23
SOAP over HTTP, 22
SOAP over JMS, 22
Spring Framework, 23
SpringBusFactory(), 25
supported versions, 22
system administrators, 9
system architects, 9

X
XML over HTTP, 22
XML over JMS, 22