Orbix Actional Integration Guide

V6.3.14

Table of Contents

Preface

What is covered in this book

Orbix supports integration with Aurea Actional® Application Performance Monitoring. This guide explains how to enable Orbix applications and services to be monitored by Actional SOA management tools. This guide applies to Orbix applications and services written in both Java and C++.

Who should read this book

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

Organization of this book

This book contains the following chapter:

- Orbix–Actional Integration describes the architecture of the Orbix integration with Actional.
- Configuring Orbix for Actional Integration explains how to configure integration between Orbix applications and services, and Actional.
- Configuring Actional for Orbix Integration provides some basic Actional configuration guidelines.
- Managing Orbix Applications in Actional shows examples of managing Orbix applications and services in Actional SOA management tools.

Related documentation

The Orbix documentation also includes the following related guides:

- *Orbix Administrator’s Guide*
- *Orbix Configuration Reference*
- *Orbix Deployment Guide*
- *Orbix Management User’s Guide*
- *Orbix Management Programmer’s Guide*

Document Conventions

This guide uses the following typographical conventions:

`Constant width`	Constant width font in normal text represents commands, portions of code and literal names of items (such as classes, functions, and variables). For example, constant width text might refer to the `itadmin orbname create` command. Constant width paragraphs represent information displayed on the screen or code examples. For example the following paragraph displays output from the `itadmin orbname list` command: `ifr` `naming` `production.test.testmgr` `production.server`
`*Italic*`	Italic words in normal text represent emphasis and new terms (for example, *location domains*).
`Code italic`	Italic words or characters in code and commands represent variable values you must supply; for example, process names in your `particular` system: `itadmin process create process-name`

`Code bold`	Code bold font is used to represent values that you must enter at the command line. This is often used in conjunction with constant width font to distinguish between command line input and output. For example: `itadmin process list` `ifr` `naming` `my_app`

The following keying conventions are observed:

No prompt	When a command’s format is the same for multiple platforms, a prompt is not used.
`%`	A percent sign represents the UNIX command shell prompt for a command that does not require root privileges.
`#`	A number sign represents the UNIX command shell prompt for a command that requires root privileges.
`>`	The notation > represents the DOS or Windows command prompt.
`...`	Horizontal ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.
`[]`	Italicized brackets enclose optional items in format and syntax descriptions.
`{}`	Braces enclose a list from which you must choose an item in format and syntax descriptions.
`|`	A vertical bar separates items in a list of choices. Individual items can be enclosed in {} (braces) in format and syntax descriptions.

Orbix–Actional Integration

Orbix provides support for integration with Actional SOA management products. This chapter explains the main components and concepts used in this integration.

Introduction

Aurea Actional® Application Performance Monitoring is a SOA management product that provides operational and business visibility, policy-based security, and control of services and business processes in a heterogeneous runtime environment. This section explains the main concepts and components used in the Orbix–Actional integration.

Note

Integration with Actional is not supported by Orbix for Microsoft Windows VC11 or later editions (neither 32-bit nor 64-bit). For more information on Orbix editions, see the Orbix *Installation Guide*.

Orbix and Actional

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

The Orbix–Actional integration is deployed on Orbix systems 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 Orbix–Actional integration can be used with Orbix applications written in both Java and C++.

Actional SOA management

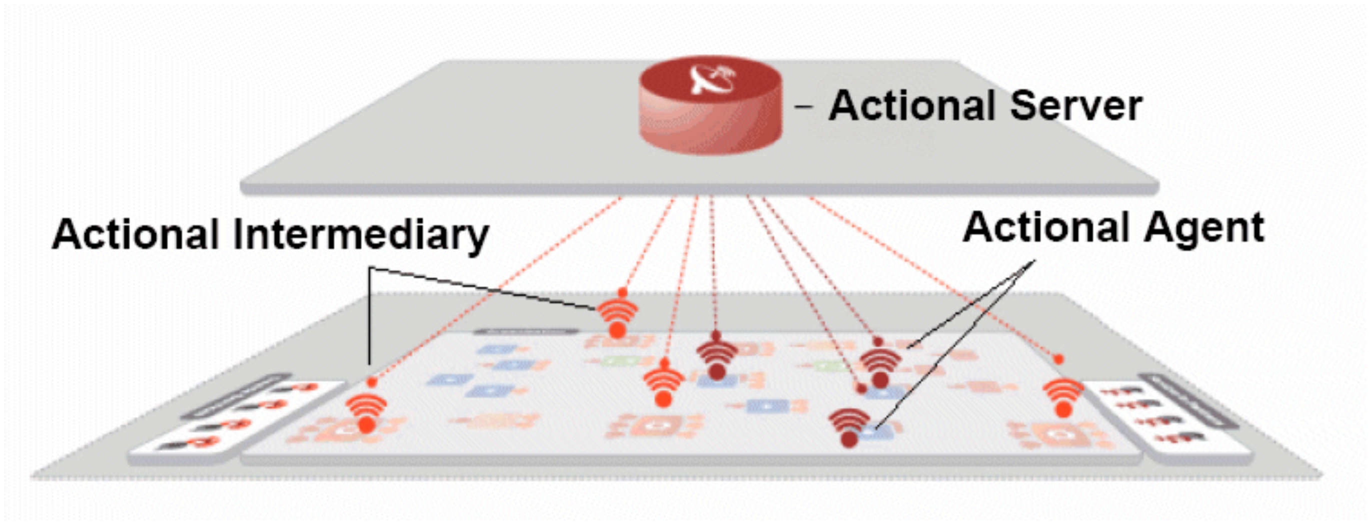
The main components in the Actional SOA management system are the Actional server, Actional agents, and Actional intermediaries.

The Actional server is the central engine that correlates data received from Actional agents and distributes policies. The Actional agent collects data about service traffic from an application server and applies policies. The Actional intermediary acts as a proxy that brokers interaction between Web service applications and systems built on them.

All Actional components are Java applications. The Actional server uses the Jetty application server by default, while its web console uses JSP and Adobe Flash.

Figure 1 shows a high-level overview of the main Actional components.

Figure 1 High-Level Actional Overview



Managed nodes

A node is defined as a system on the current network. A node with an Actional agent installed is referred to as an *instrumented node* or a *managed node*.

The managed node uses Actional's interceptor API to send monitoring data to the Actional agent. On any managed node, one Actional agent and one or more interceptors must be running.

Actional server

The Actional server is a central management server that manages nodes containing an Actional agent. The Actional server correlates the data it receives from each of its agents, and distributes policies to those agents. It enables an administrator to analyze service network data and create system-wide policies.

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 a console—for example, the **Actional Management Server Administration Console**. This is a Web application deployed on Apache Tomcat, which provides runtime management and agent configuration. In addition, any alerts triggered at the Actional agent are sent immediately to the Actional server.

The default Actional server database is Apache Derby. Other supported databases include:

- PostgreSQL
- OpenEdge
- MSDE
- SQL Server
- Oracle
- DB2

  By default, the Actional server uses port `4040` (for example, `http://HostName:4040/lgserver/`).

Actional agent

An Actional agent is run on each Orbix host that you wish to manage, and is used to provide instrumentation data back to the Actional server. The Actional agent includes two main components: an analyzer, and one or more interceptors. The analyser gathers and evaluates data such as records, statistics, and alerts. The interceptors collect data about service traffic from an application server, and apply policies to that traffic.

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 managed node. By default, the Actional agent uses port `4041` (for example, `http://HostName:4041/lgagent/`).

Actional intermediary

An Actional intermediary is an in-network service broker that includes an integrated Actional agent. It serves as a proxy for Web service applications, providing features such as security, bridging, and activity tracking. The Actional intermediary supports application servers such as WebLogic, WebSphere, JBoss, and Oracle.

Actional agent interceptor SDK

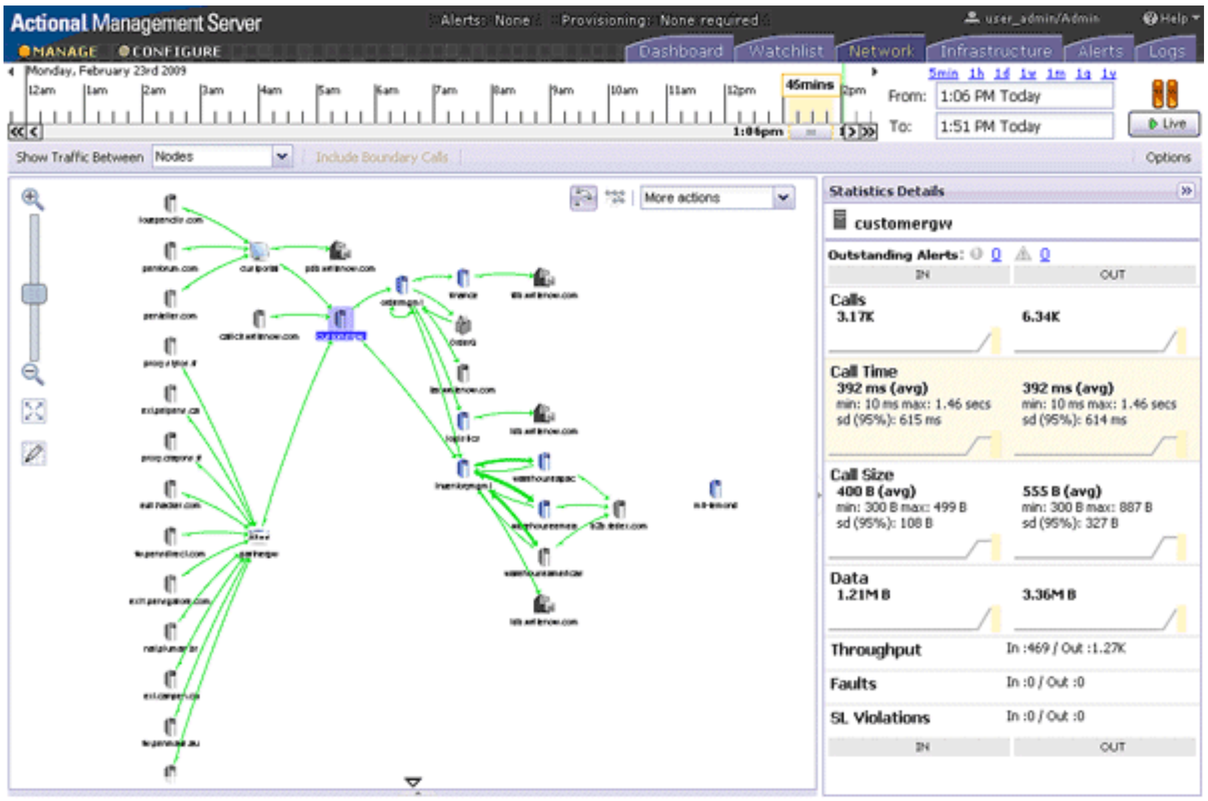
The Actional Agent Interceptor Software Development Kit (SDK) is an Actional-specific API used to create custom interceptors. These can be used to send management instrumentation data from an application to the Actional agent.

Actional SOA management tools

In this guide, 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 2 shows an example of the **Actional Management Server Administration Console**. Managed nodes are displayed as blue boxes, and unmanaged nodes are displayed as gray boxes. The green arrows indicate the message flow through various nodes. Clicking on each of the nodes shows more in-depth information regarding the response time, alerts and warnings, and so on.

Figure 2 Actional Management Server Administration Console



From: 1:06 PM Today
To: 1:51 PM Today
Show Traffic Between: Nodes

customergw

Outstanding Alerts: 0 0

	IN	OUT
Calls	3.17K	6.34K
Call Time	392 ms (avg) min: 10 ms max: 1.46 secs sd (95%): 615 ms	392 ms (avg) min: 10 ms max: 1.46 secs sd (95%): 614 ms
Call Size	400 B (avg) min: 300 B max: 499 B sd (95%): 108 B	555 B (avg) min: 300 B max: 887 B sd (95%): 327 B
Data	1.21M B	3.36M B
Throughput	In :469 / Out :1.27K	
Faults	In :0 / Out :0	
SL Violations	In :0 / Out :0	

NGSO mapping

When you click and drill down in the Actional **Path Explorer** view, the organization of the information displayed is *Node–Group–Service–Operation* (NGSO). In Orbix, this translates to *Host–Module–Interface–Operation*. Table 1 shows the mapping from Actional to Orbix.

Table 1: NGSO Mapping

Actional	Orbix
Node	Host
Group	Module
Service	Interface
Operation	Operation

NGSO mapping shown in the above table is the default mapping. You can change this default mapping in the application configuration scope in your Orbix configuration. For details on setting the configuration variables, see Monitoring plug-in configuration variables.

Further information

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

Orbix–Actional Integration Architecture

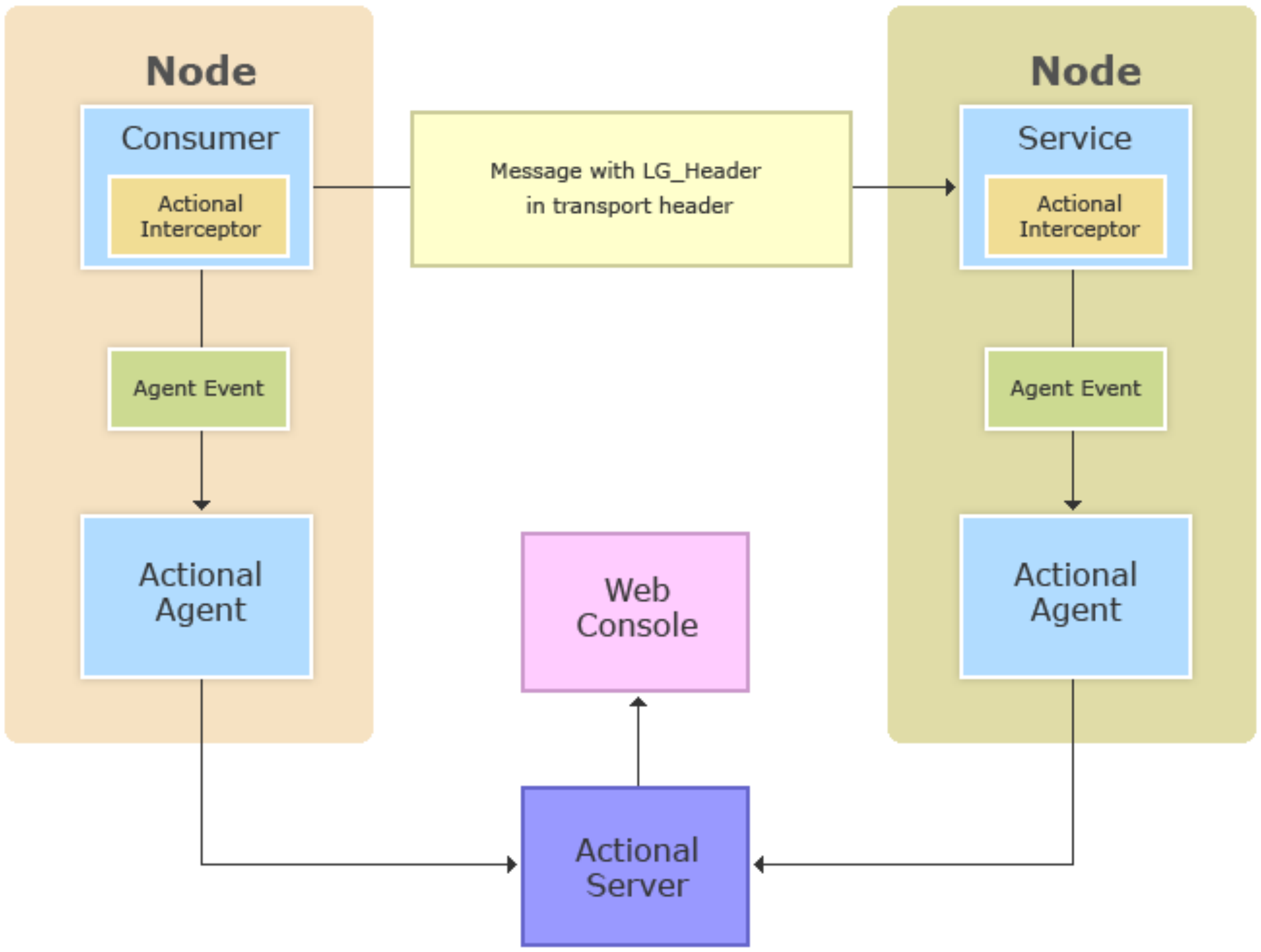
This section shows a basic Actional architecture, simplified for the purposes of this discussion. It explains how Actional interceptors provide data to the Actional agent, and how the Actional server manifest is used to correlate the origin and business flow of a request.

It then shows the Orbix–Actional integration architecture, and explains how Orbix plug-ins and Orbix interceptors are used to configure integration with Actional.

Basic Actional architecture

Figure 3 shows a high-level overview of a basic Actional architecture from the perspective of a consumer and service provider.

Figure 3 Basic Actional Architecture



In the interaction shown in Figure 3, the Actional interceptors sit in the flow between the application logic and the consumers and providers of other services. They intercept all inbound and outbound calls, and feed information about those calls to the Actional agent as asynchronous events.

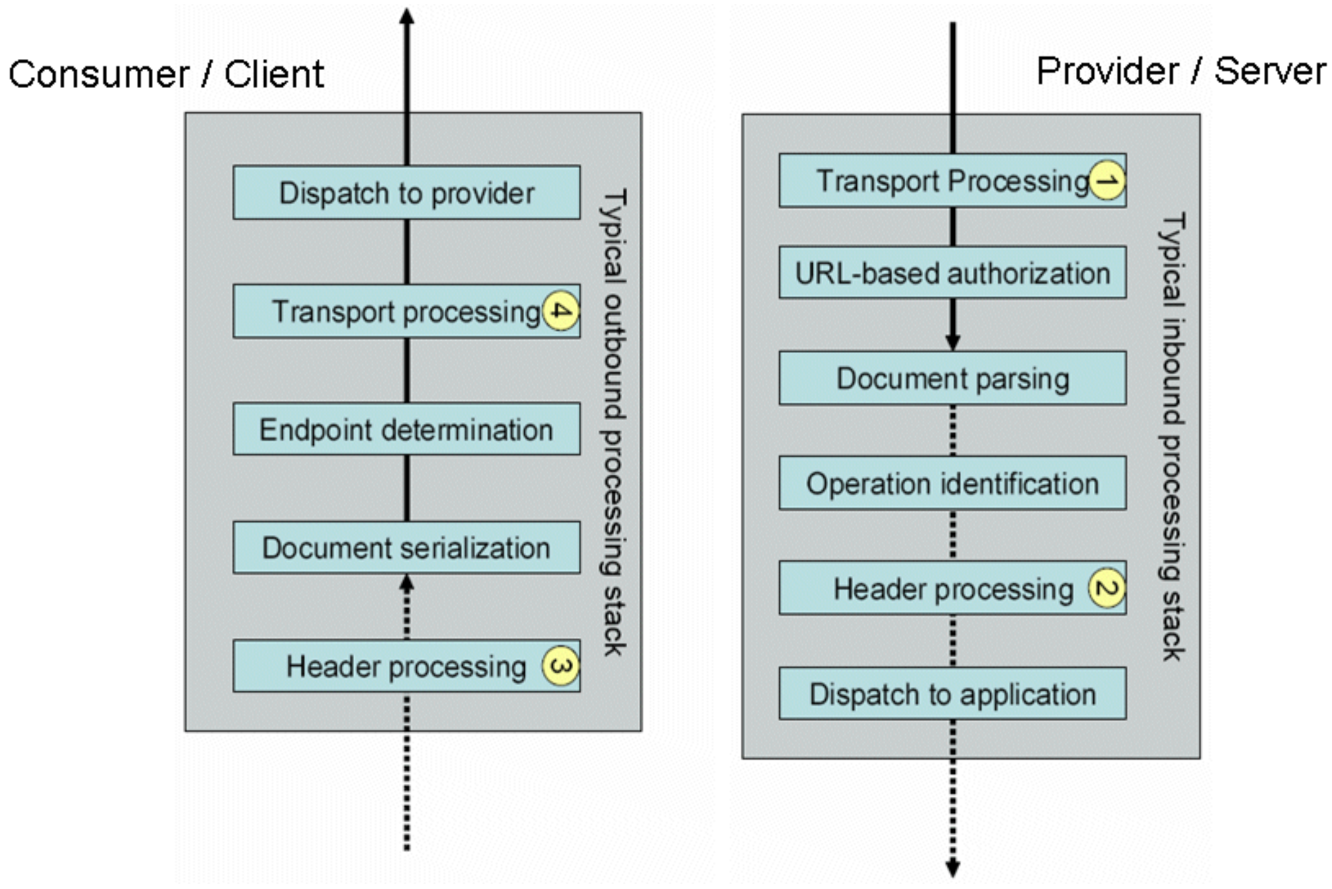
The Actional agent is responsible for processing the event stream from the interceptors, computing and storing aggregate statistics, executing policies, and communicating with the Actional server.

The Actional server manifest (`LG_Header`) is a token that is sent in the transport header of the message to each participant in a call. This token identifies the origin and business flow of a request. For more details, see Actional server manifest.

Actional interceptors

Actional interceptors sit in the flow at the edge of an application, intercepting all incoming and outgoing messages. An Actional interceptor is designed as a lightweight component that imposes minimal overhead on the application (typically less than 100 microseconds per call).

Figure 4 Actional Interceptors



The interceptor must perform the following tasks to gain the full functionality of the Actional server:

1. Extract an Actional server manifest (if any) from the incoming request document.
2. Insert an Actional server manifest into any outgoing request documents.
3. Transfer the interceptor context along the internal business flow, from the incoming interceptor, to any related outgoing interceptors.
4. Send the Actional agent an event for each incoming or outgoing document.

Actional server manifest

The Actional server sends an Actional server manifest (`LG_Header`) with a request document to provide information about the request’s origin and the business flow that the request belongs to.

The Actional server manifest is used by the Actional server to correlate information it receives, from multiple agents, about interactions between different services. For this reason, the server manifest is sometimes referred to as a correlation ID.

The consumer and provider of the service must have an agreed mechanism (transport or protocol) for transferring the manifest. The following is an example `LG_Header`:

```
Interaction=CgJkcB+YlN0ZyBABdysAAA==;
Locus=ApM1eYBGBAR4LFJ1VvHOdg==;
Flow=CgJkcB+YlN0ZyBABdSsAAA==;
UpstreamOpID=FtfEJXM1nqJ0C995IBMkEQ==;
Path=7Qg2aVWCdwmP8gGebyLWYA==;
name=E_10-2-100-112-e0c7c3-110c80b4df0--7fdd-INITIATED;
CPTime=1171591682345;
FlowFields=MF1:1254;MF2:1589;
```

The main components in the server manifest are the `Interaction`, `Locus`, `Flow`, and `UpstreamOpID`. The other components are optional.

Orbix–Actional integration architecture

The Orbix–Actional integration is built using the extensible Orbix plug-in architecture. This means that Orbix–Actional integration can be enabled by adding a monitoring plug-in to your Orbix configuration. No code changes are necessary for Orbix client and server applications.

Figure 5 shows an overview of the Orbix–Actional integration architecture from an Orbix client-server perspective. This builds on the architecture shown in Figure 3, with the addition of Orbix monitoring and GIOP plug-ins. In Figure 5, the CORBA GIOP message also includes the `LG_Header` in a GIOP service context. A GIOP service context is a general mechanism for including out-of-band data in a GIOP request or reply message. Service contexts in GIOP are analogous to headers in other protocols such as HTTP.

Orbix interceptors

In the Orbix–Actional integration, Orbix interceptors for Actional must also be added to your Orbix client and server binding lists. Orbix interceptors are objects that ORB services and transports implement to process operation invocations. Orbix interceptors are arranged in a chain, with each interceptor caching a reference to the next interceptor in the chain.

The Orbix monitoring plug-in is implemented as a *request-level interceptor*. This receives a request in the form of a request object from the preceding interceptor in the chain. This enables high-level request processing to be performed. In CORBA, a *binding* is a set of interceptors used to process requests.

Further information

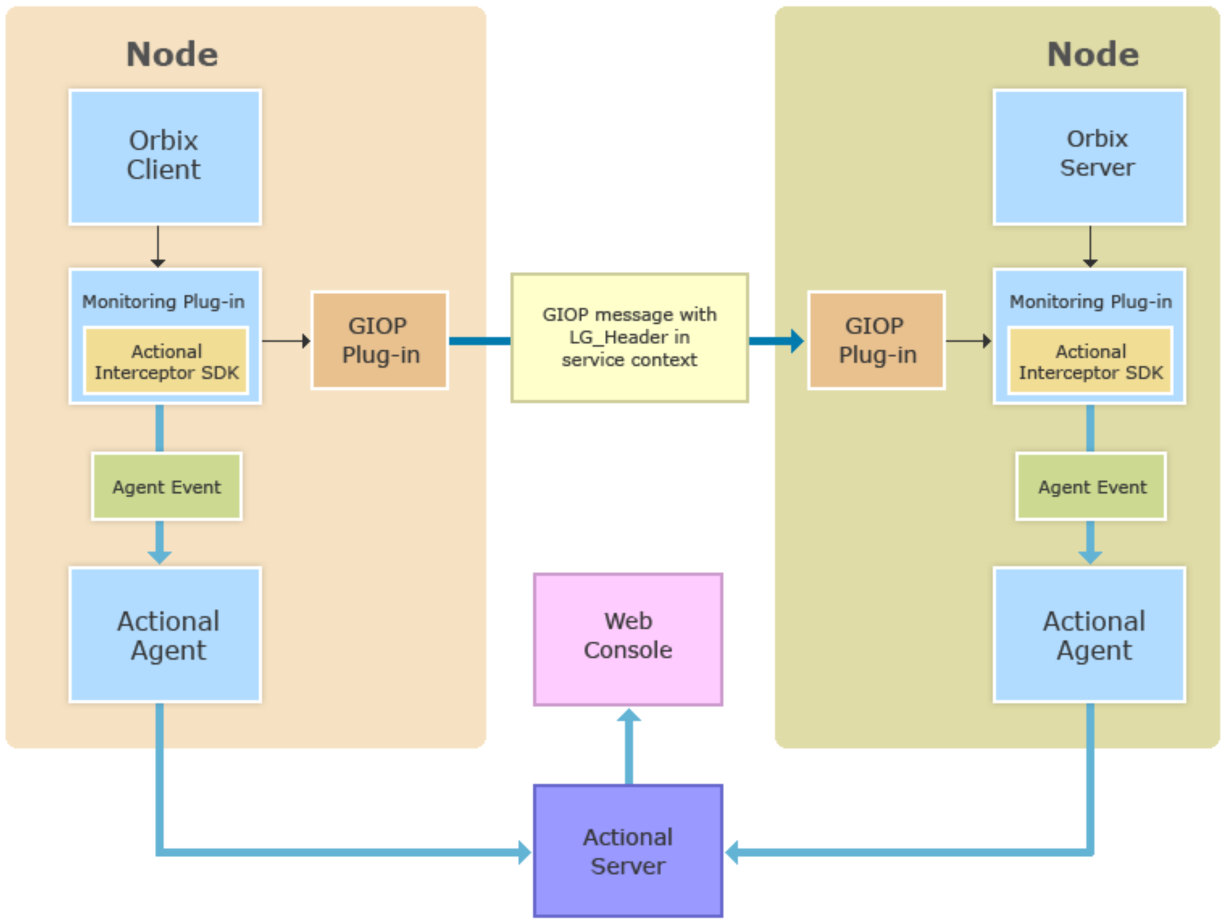
For detailed information on Actional architecture and components, see the Actional product documentation.

For details on how to configure the Orbix plug-in and interceptors for Orbix–Actional integration, see Configuring Actional for Orbix Integration.

For detailed information on Orbix interceptors, see:

- *Orbix Configuration Reference*
- *Orbix C++ Programmer’s Guide*
- *Orbix Java Programmer’s Guide*

Figure 5 Orbix–Actional Integration Architecture


Node
Orbix Client
Monitoring Plug-in
Actional Interceptor SDK
GIOP Plug-in
Agent Event
Actional Agent
GIOP message with LG_Header in service context
Web Console
Actional Server
Node
Orbix Server
GIOP Plug-in
Monitoring Plug-in
Actional Interceptor SDK
Agent Event
Actional Agent

Configuring Orbix for Actional Integration

This chapter explains the steps required to configure Orbix for integration with Actional SOA management products.

Configuring an Orbix Domain

This section explains how to use the **Orbix Configuration** tool to enable an Orbix configuration domain for Actional integration. It shows how to configure and deploy your Orbix domain services with the Orbix configuration settings required for monitoring by Actional. For example, Orbix domain services include the locator daemon, configuration repository, naming service and so on.

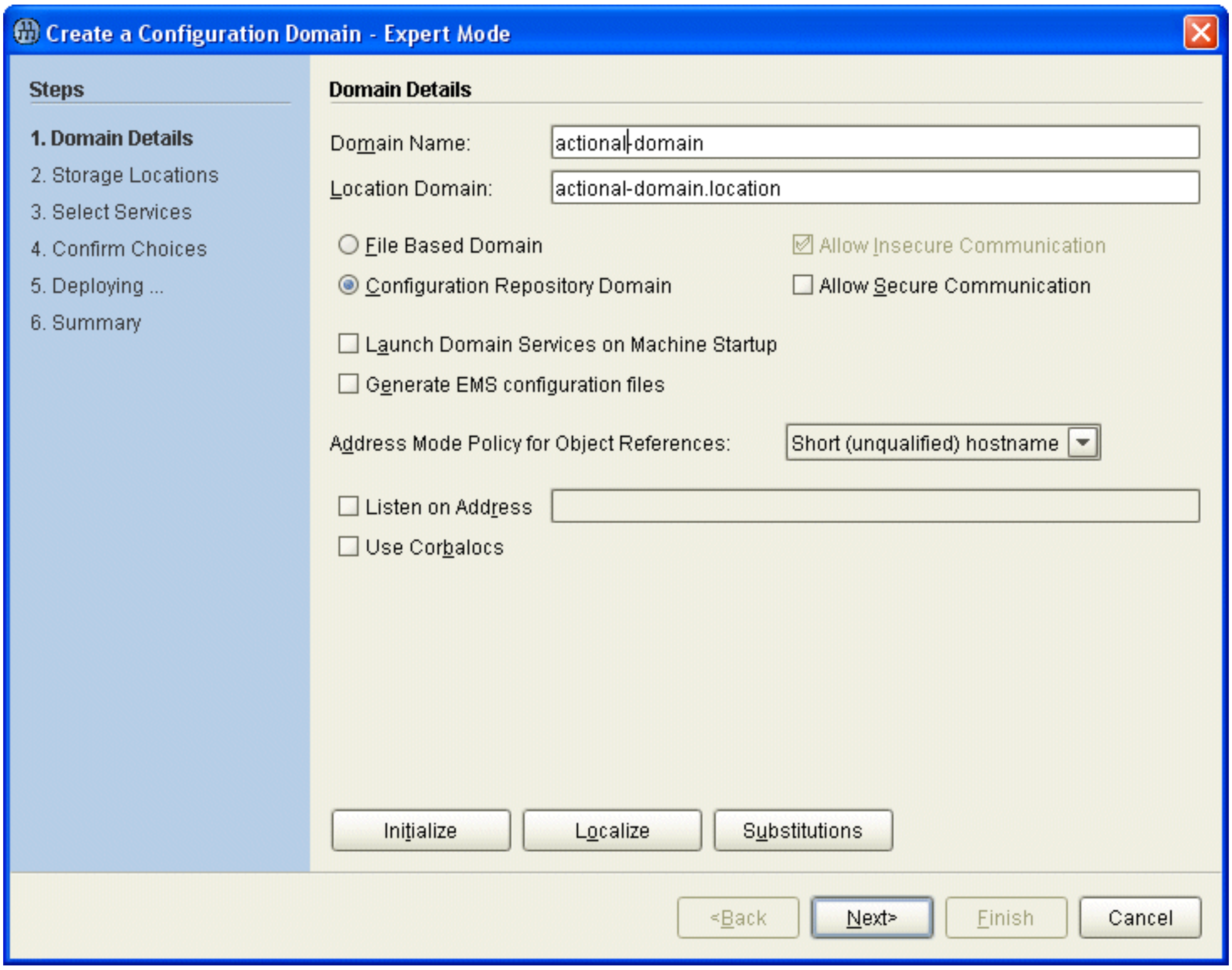
Configuring Orbix services for Actional integration

To configure Orbix domain services for Actional integration, perform the following steps:

1. Start the **Orbix Configuration** tool using the following command:

```
OrbixInstallDir\asp\6.3\bin\itconfigure
```

Figure 6 Creating a Domain in Expert Mode



Domain Name: actional-domain

Location Domain: actional-domain.location

( ) File Based Domain
(●) Configuration Repository Domain

[✓] Allow Insecure Communication
[ ] Allow Secure Communication

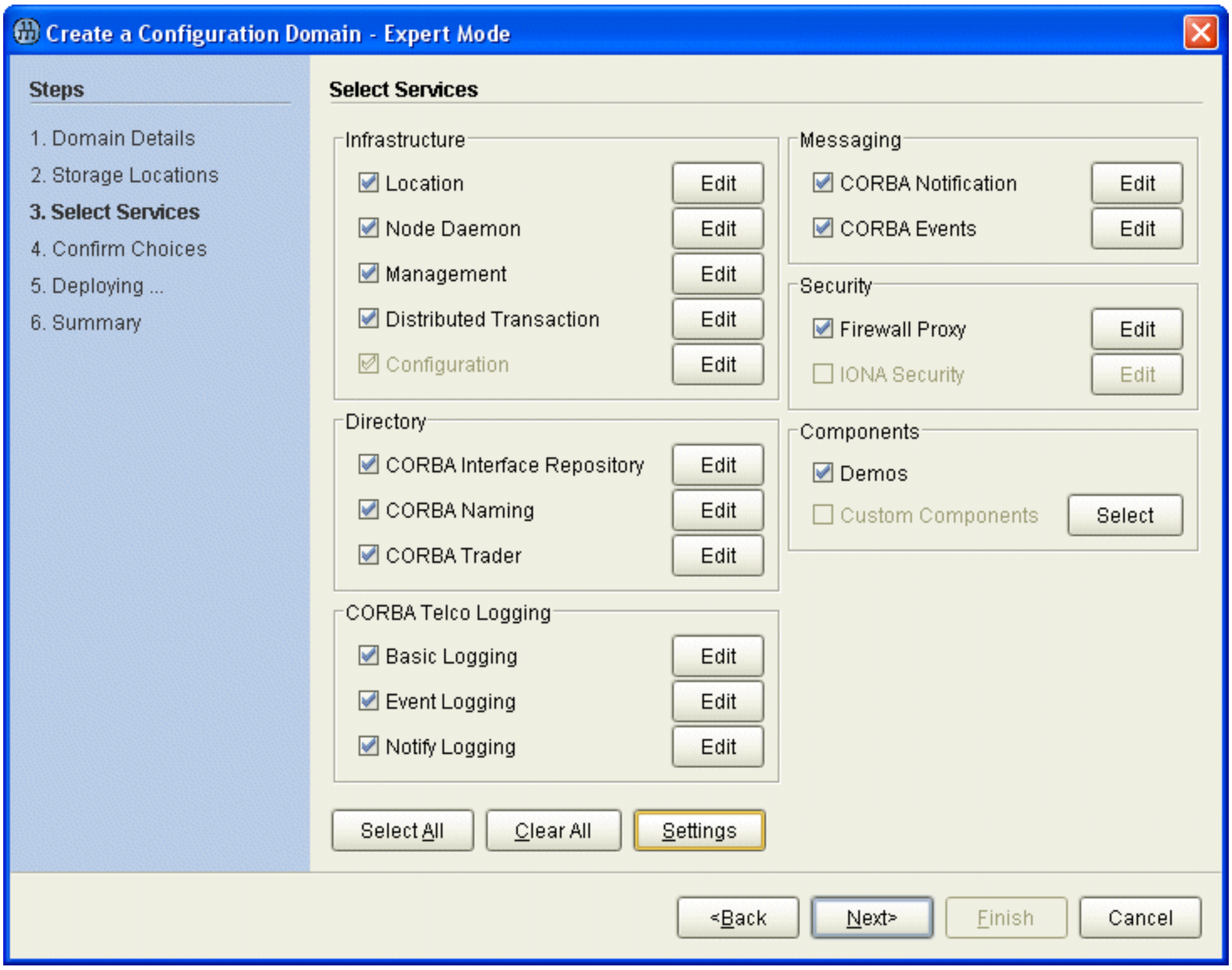
[ ] Launch Domain Services on Machine Startup
[ ] Generate EMS configuration files

Address Mode Policy for Object References: Short (unqualified) hostname

[ ] Listen on Address
[ ] Use Corbalocs

2. Click **Cancel** or press the Esc key to close the **Orbix Configuration Welcome** dialog box.
3. Select **File**>**New**>**Expert** to create a domain in **Expert Mode** (shown in Figure 6).
4. Specify the **Domain Details** (for example, whether it is configuration file-based or configuration repository-based).
5. Click **Next** to specify any custom storage locations.
6. Click **Next** to specify the required Orbix domain services.
7. Select the services you require and click the **Settings** button at the bottom of the screen (shown in Figure 7)

   Figure 7 Selecting Services



Create a Configuration Domain - Expert Mode

Steps

1. Domain Details
2. Storage Locations
3. **Select Services**
4. Confirm Choices
5. Deploying ...
6. Summary

Select Services

Infrastructure
- ✓ Location
- ✓ Node Daemon
- ✓ Management
- ✓ Distributed Transaction
- ✓ Configuration

Directory
- ✓ CORBA Interface Repository
- ✓ CORBA Naming
- ✓ CORBA Trader

CORBA Telco Logging
- ✓ Basic Logging
- ✓ Event Logging
- ✓ Notify Logging

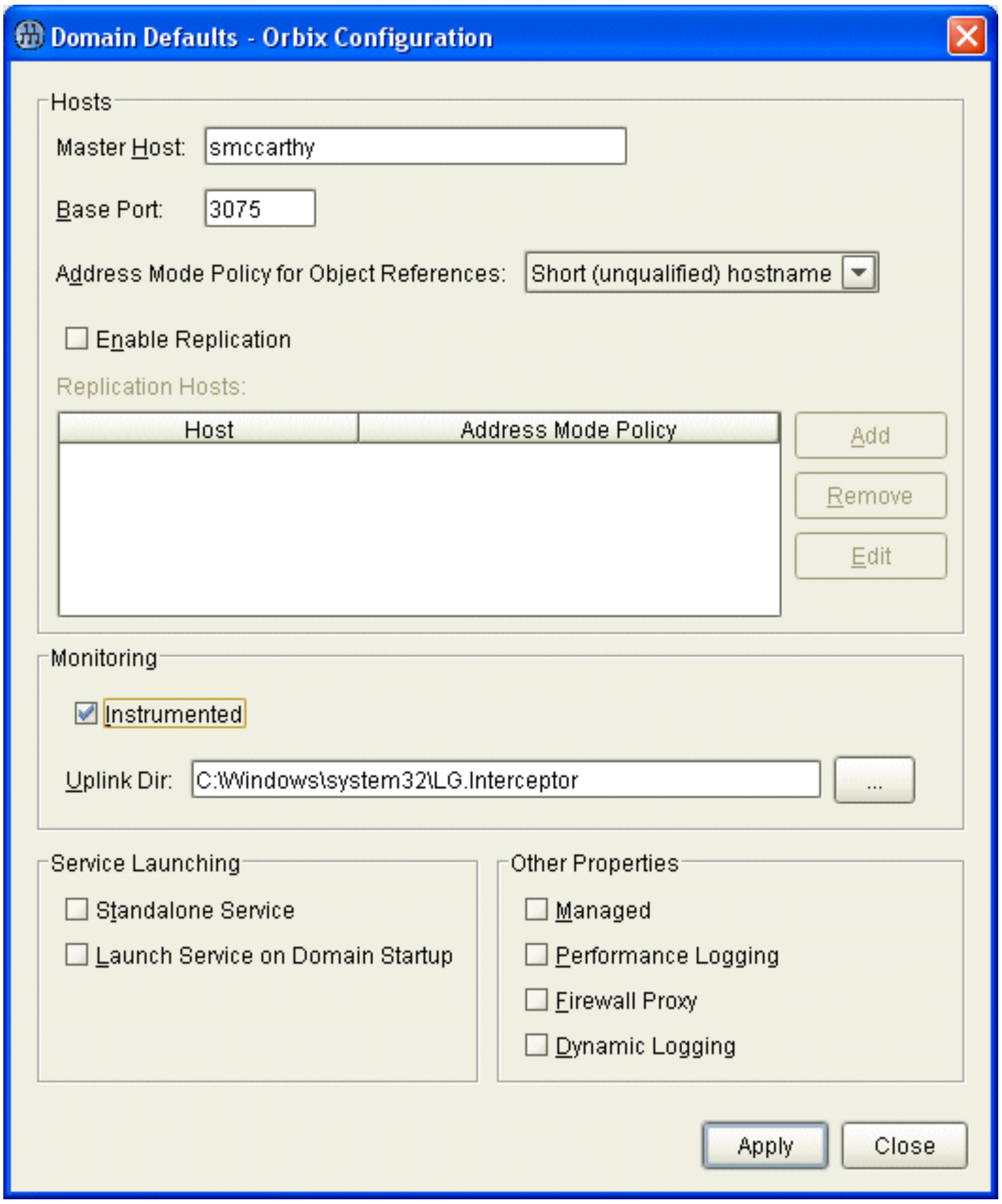
Messaging
- ✓ CORBA Notification
- ✓ CORBA Events

Security
- ✓ Firewall Proxy
- ☐ IONA Security

Components
- ✓ Demos
- ☐ Custom Components

8. In the **Domain Defaults** screen, in the **Monitoring** panel, select the **Instrumented** check box (shown in Figure 8). This will add the required Orbix configuration settings to the Orbix services that you selected.

Figure 8 Specifying Actional Monitoring


Domain Defaults - Orbix Configuration

Hosts

Master Host: smccarthy

Base Port: 3075

Address Mode Policy for Object References: Short (unqualified) hostname

- [ ] Enable Replication

Replication Hosts:

Host	Address Mode Policy

Monitoring

- [x] Instrumented

Uplink Dir: C:\Windows\system32\LG.Interceptor

Service Launching

- [ ] Standalone Service
- [ ] Launch Service on Domain Startup

Other Properties

- [ ] Managed
- [ ] Performance Logging
- [ ] Firewall Proxy
- [ ] Dynamic Logging

9. If your Actional `Uplink.cfg` configuration file is not located in its default path, specify its directory path in the **Uplink Dir** text box. The path specified must match that specified for your Actional agent. The default values are:

UNIX	**`/var/opt/actional/LG.Interceptor`**
Windows	`%systemroot%\system32\LG.Interceptor`

10. Click **Apply**.
11. Click **Close**.
12. Click **Next** to view your selections.
13. Click **Next** to deploy your domain.

14. Click **Finish**.

Using the command line

You can also use the `enable_actional.tcl` script to automatically add the configuration necessary for Actional integration to the configuration scope of any Orbix service. For more details, see Running the enable_actional Script.

Further information

For more detailed information on using the **Orbix Configuration** tool, see the Orbix Deployment Guide.

Configuring Orbix Java Applications

This section explains how to configure Orbix Java applications for integration with Actional. It shows some examples from the Orbix Actional integration demo:

```
OrbixInstallDir/asp/6.3/demos/corba/orb/actional_demo
```

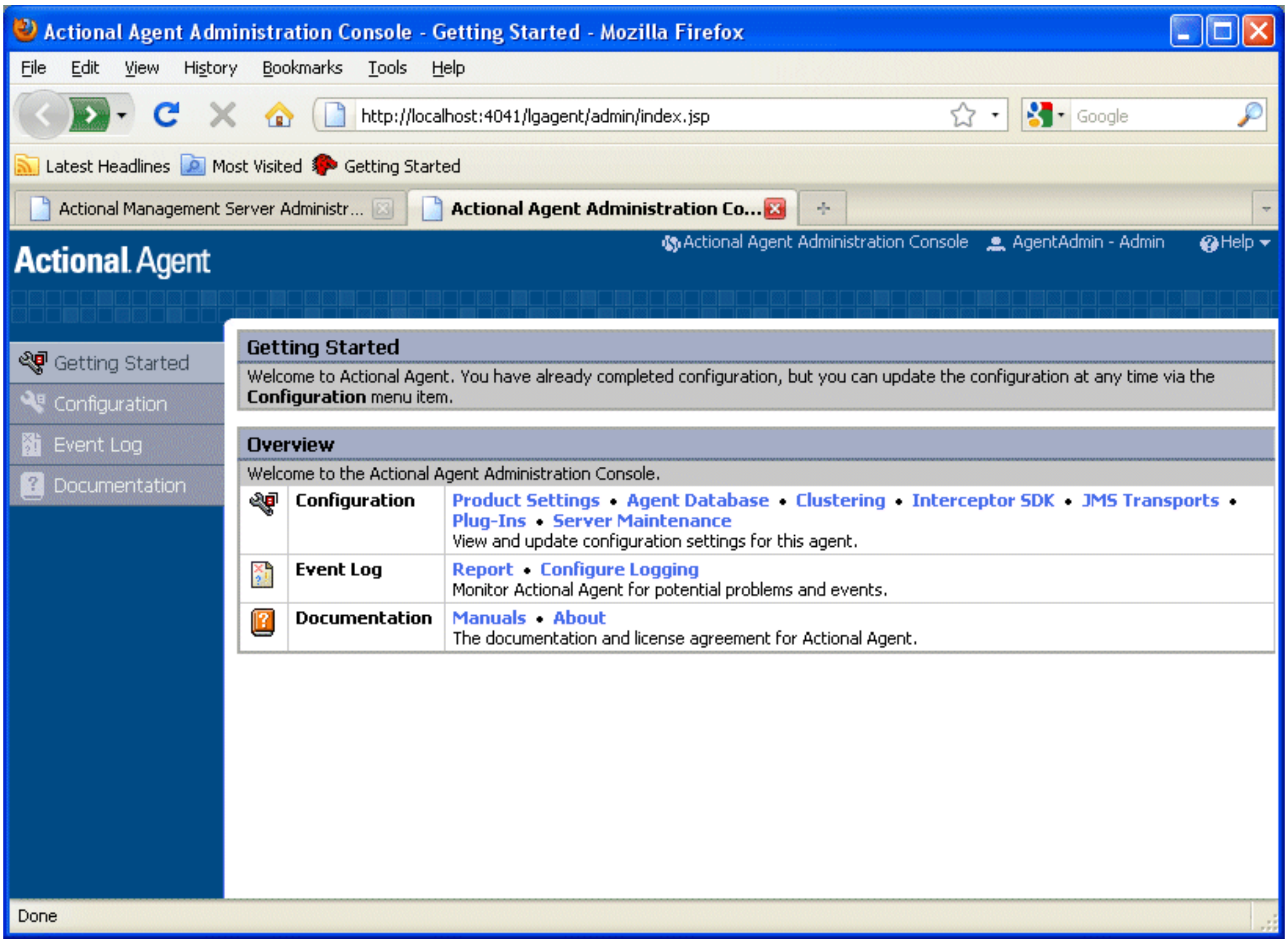
Update your Actional SDK

You must first update your Actional SDK JAR file as follows:

1. In the **Actional Agent Administration Console**, select **Getting Started>Interceptor SDK** (see Figure 9), and download the Windows `(.zip`) or UNIX (`.tar)` file. This includes the `actional-sdk.jar`, documentation, and samples.
2. Replace the existing `actional-sdk.jar` in the following location with the version that you downloaded:

```
OrbixInstallDir/lib/platform/orbmon/1.3
```

Figure 9 Actional Agent Administration Console


Actional Agent Administration Console - Getting Started - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:4041/lgagent/admin/index.jsp
Latest Headlines Most Visited Getting Started
Actional Management Server Administr...
Actional Agent Administration Co...
Actional Agent
Actional Agent Administration Console
AgentAdmin - Admin
Help
Getting Started
Configuration
Event Log
Documentation
Done

Getting Started

Welcome to Actional Agent. You have already completed configuration, but you can update the configuration at any time via the **Configuration** menu item.

Overview

Welcome to the Actional Agent Administration Console.

	Configuration	Product Settings • Agent Database • Clustering • Interceptor SDK • JMS Transports • Plug-Ins • Server Maintenance View and update configuration settings for this agent.
	Event Log	Report • Configure Logging Monitor Actional Agent for potential problems and events.
	Documentation	Manuals • About The documentation and license agreement for Actional Agent.

Configuring the Orbix monitoring plug-in

You can configure the monitoring plug-in by editing the settings in your application configuration scope in your Orbix configuration file. This includes the following steps:

- Specify the monitoring plug-in
- Add monitoring handlers to the interceptor chain
- Specify the monitoring log filter

Note

Alternatively, you can use the `enable_actional.tcl` script to add all the configuration necessary for Actional integration to an Orbix configuration scope (see Running the enable_actional Script).

Specifying the plug-in name

To set the monitoring plug-in name, add the following settings:

```
# Specify the monitoring class name.
plugins:orbmon:ClassName = "com.iona.corba.plugin.monitoring.MRIPlugIn";
# Load the monitoring plug-in:
orb_plugins = ["local_log_stream", "orbmon", "iiop_profile", "giop", "iiop"];
```

Adding handlers to the interceptor chain

You must also specify monitoring handlers to the Orbix interceptor binding lists, on both the client side and server side. For example:

```
# Add the client-side handlers to the interceptors chain.
binding:client_binding_list = ["POA_Coloc", "ORBMON+GIOP+IIOP", "GIOP+IIOP"];
# Add the server-side handlers to the interceptors chain.
binding:server_binding_list = ["ORBMON", ""];
```

For more details on configuring Orbix binding lists and interceptors, see the .

Specifying the monitoring filter

You can specify the monitoring log filter as follows:

```
event_log:filters = ["IT_MONITORING=*"];
```

For more details, see Troubleshooting Orbix.

Note

When you run the **Orbix Configuration** GUI tool (`itconfigure` command), all the configuration necessary for the **actional_demo** is added to your configuration file by default. If you select the **Expert** option, you must select the **Demos** component.

Running client and server applications

No changes are necessary when running your Orbix Java client and server applications if the Actional `Uplink.cfg` configuration file is located in its default path:

UNIX	**`/var/opt/actional/LG.Interceptor`**
Windows	`%systemroot%\system32\LG.Interceptor`

The `Uplink.cfg` file is responsible for communication between the Actional interceptors and the analyzer in the Actional agent.

If the `Uplink.cfg` is not located in its default path, the `-Dcom.actional.lg.interceptor.config` system property must be to be added the Java commands for both the client and the server. For example:

```
java -Dcom.actional.lg.interceptor.config=Path ...
```

Specifying endorsed directories

If you are using JDK 1.4.x, you must also specify `-Djava.endorsed.dirs` system property on the Java command line as follows:

Windows	**`-Djava.endorsed.dirs="IT_PRODUCT_DIR\\lib\\art\\omg\\5"`**
UNIX	`-Djava.endorsed.dirs=IT_PRODUCT_DIR/lib/art/omg/5`

Sample Orbix configuration

The following sample configuration shows the settings required for Java integration with Actional in an example application configuration scope:

```
my_app
{
plugins:orbmon:ClassName = "com.iona.corba.plugin.monitoring.MRIPlugIn";
orb_plugins = ["local_log_stream", "orbmon", "iiop_profile", "giop", "iiop"];
binding:client_binding_list = ["POA_Coloc", "ORBMON+GIOP+IIOP", "GIOP+IIOP"];
binding:server_binding_list = ["ORBMON", ""];
event_log:filters = ["IT_MONITORING=*"];
};
```

Configuring Orbix C++ applications

This section explains how to configure Orbix C++ application for integration with Actional. It shows some examples from the Orbix Actional integration demo:

```
OrbixInstallDir/asp/6.3/demos/corba/orb/actional_demo
```

Setting your environment

No changes are necessary if the Actional `Uplink.cfg` configuration file is located in its default path:

UNIX	**`/var/opt/actional/LG.Interceptor`**
Windows	`%systemroot%\system32\LG.Interceptor`

The `Uplink.cfg` file is responsible for communication between the Actional interceptors and the analyzer in the Actional agent.

If the `Uplink.cfg` is not located in its default path, you must specify the path to this file as follows:

UNIX	**`export LG_INTERCEPTORCONFIG=PathToFile`**
Windows	`set LG_INTERCEPTORCONFIG=PathToFile`

Configuring the Orbix monitoring plug-in

You can configure the monitoring plug-in by editing the settings in your application configuration scope in your Orbix configuration file. This includes the following steps:

- Specify the monitoring plug-in
- Add the monitoring handlers to the interceptor chain
- Specify the monitoring log filter

Note

Alternatively, you can use the `enable_actional.tcl` script to add all the configuration necessary for Actional integration to an Orbix configuration scope (see Running the enable_actional Script).

Specifying the plug-in name

To set the monitoring plug-in name, add the following settings:

```
# Specify the monitoring library.
plugins:orbmon:shlib_name = "it_orb_monitoring";
# Load the monitoring plug-in.
orb_plugins = ["local_log_stream", "orbmon", "iiop_profile", "giop", "iiop"];
```

Adding handlers to the interceptor chain

You must also specify monitoring handlers to the Orbix interceptor binding lists, on both the client side and server side. For example:

```
# Add the client-side handlers to the interceptors chain.
binding:client_binding_list = ["POA_Coloc", "ORBMON+GIOP+IIOP", "GIOP+IIOP"];
# Add the server-side handlers to the interceptors chain.
binding:server_binding_list = ["ORBMON", ""];
```

For more details on configuring Orbix binding lists and interceptors, see the *Orbix Configuration Reference*.

Specifying the monitoring filter

You can specify the monitoring log filter as follows:

```
event_log:filters = ["IT_MONITORING=*"];
```

For more details, see Troubleshooting Orbix.

> **Note**
>
> When you run the **Orbix Configuration** GUI tool (`itconfigure` command), all the configuration necessary for the **actional_demo** is added to your configuration file by default. If you select the **Expert** option, you must select the **Demos** component.

Sample Orbix configuration

The following sample configuration shows some example settings in a `my_app` configuration scope:

```
my_app {
plugins:orbmon:shlib_name = "it_orb_monitoring";
orb_plugins = ["local_log_stream", "orbmon", "iiop_profile", "giop", "iiop"];
binding:client_binding_list = ["POA_Coloc", "ORBMON+GIOP+IIOP", "GIOP+IIOP"];
binding:server_binding_list = ["ORBMON", ""];
event_log:filters = ["IT_MONITORING=*"];
};
```

Monitoring plug-in configuration variables

plugins:orbmon

The plugins:orbmon namespace contains the following variables that you can set for C++ and Java applications:

- use_msg_fields
- group
- service

- operation

`use_msg_fields` accepts boolean value and `group`, `service`, and `operation` accept string value.

use_msg_fields

`use_msg_fields` specifies whether message field names with their corresponding values are reported to the Actional agent.

You can view these message fields in the Actional Management Server when they are added or selected at "Audit Message Fields in a Request or Reply" in a policy rule.The message fields along with their values should appear in the audit log details.

In the Orbix configuration, enabling message fields for reporting ,by default, is set to false:

```
plugins:orbmon:use_msg_fields = "true"
```

When this variable is set to true, the following message fields and their corresponding values are reported:

Table 2: MSG Fields reporting

MSG Field	Server side	Client side	Java	C++
ORBID	ü	ü	ü	ü
ORBNAME	ü	ü	ü	ü
SERVERPORT	ü	ü	ü	ü
CLIENTPORT	ü		ü	ü
SERVERTID	ü		ü	ü
SERVERPID	ü			ü
CLIENTPID		ü		ü

MSG Field	Server side	Client side	Java	C++
CLIENTTID		ü	ü	ü

Configuring NGSO

You can change the default NGSO mapping by setting variables for group, service, and operation individually to override the default NGSO mappings. See NGSO mapping.

The following are the configuration variables that you need to set to change the default NGSO mappings for the particular field of the GSO:

- plugins:orbmon:group
- plugins:orbmon:service
- plugins:orbmon:operation

Note

If no value is set for the configuration variable, the NGSO mapping defaults back to the original mapping.

The following is the list of substitutes that you can use within the variable's string and these substitutes are replaced with their corresponding values on the fly during an interceptor invocation:

Table 3: Substitutes used in the variable strings

Substitute	Definition
%MODULE%	The module defined in IDL
%INTERFACE%	The interface defined in IDL
%OPERATION%	The operation name defined in IDL
%ORBNAME%	The unique name that identifies the ORB.
%ORBID%	The unique ID of the ORB
%SERVERPORT%	The IP port on which the server is connected to the client.

Substitute	Definition
%CLIENTPORT%	The IP port on which the client is connected to the server.

The resulting string is used for group, service or operation. The variable's string can contain any characters except "%" as the character is used as delimiter.

group

`group` specifies the value displayed for Group in the NGSO mapping. For example,

```
plugins:orbmon:group = "%MODULE% - %ORBNAME%";
```

service

`service` specifies the value displayed for Services in the NGSO mapping. For example,

```
plugins:orbmon:service = "%INTERFACE% - %ORBID%";
```

operation

`operation` specifies the value displayed for Operation in the NGSO mapping. For example,

```
plugins:orbmon:operation = "%SERVERPORT%, %CLIENTPORT% ->IDL:%MODULE%/
%INTERFACE%:1.0";
```

Running the enable_actional Script

This section explains how to use the `enable_actional.tcl` script to automatically add the configuration for Actional integration to an Orbix configuration scope. This script can be used to instrument an Orbix C++ or Java application, or an Orbix domain service (for example, locator daemon, naming service, and so on).

Script usage

The `enable_actional.tcl` script is located in the following directory:

```
OrbixInstallDir\asp\6.3\bin\enable_actional.tcl
```

This script has the following syntax:

```
itadmin enable_actional.tcl *ScopeToBeInstumented*
```

You must supply the Orbix configuration scope to be instrumented. This script does not apply to nested configuration scopes.

Script output

When you run the `enable_actional.tcl` script, it adds the monitoring plug-in (`orbmon`) to the following configuration variables in the specified scope:

- `orb_plugins`
- `binding:server_binding_list`
- `binding:client_binding_list`

It also adds the necessary C++ and Java libraries to the global scope, if not present:

- `plugins:orbmon:shlib_name`
- `plugins:orbmon:ClassName`

Examples

The following are some example commands

- `itadmin enable_actional.tcl my_c++_app`
- `itadmin enable_actional.tcl my_java_app`
- `itadmin enable_actional.tcl iona_services.locator.MyHost`
- `itadmin enable_actional.tcl iona_services.node_daemon`.`MyHost`

The following is an example of the configuration settings that are added when the script is run:

```
...
plugins:orbmon:shlib_name = "it_orb_monitoring";
plugins:orbmon:ClassName = "com.iona.corba.plugin.monitoring.MRIPlugIn";
...
my_app {
orb_plugins = ["orbmon", "local_log_stream", "iiop_profile", "giop", "iiop"];
binding:server_binding_list = ["ORBMON", "OTS", ""];
binding:client_binding_list = ["ORBMON+GIOP+IIOP", "POA_Coloc", "GIOP+IIOP"];
} ;
```

Troubleshooting Orbix

This section provides some tips to help troubleshoot your Orbix integration with Actional.

Ensure the monitoring plug-in is loaded

To verify that the Orbix monitoring plug-in is loaded and participating in the Orbix interceptor chain, you can enable logging by adding `IT_MONITORING` filter to the event log. For example:

```
event_log:filters = ["IT_MONITORING=*"];
```

When logging has been enabled for the monitoring plug-in, logging statements for `IT_MONITORING` should appear in your log files or on screen. This verifies that the monitoring plug-in is correctly loaded, and that and calls are going through the Orbix interceptors.

Java example

The following are some example logging statements for Orbix Java client and server applications:

```
13:30:43 11/05/2009 [_it_orb_id_1@zajonzd690/10.2.4.13] (IT_MONITORING:203) I
- Client Interaction begin
13:30:43 11/05/2009 [_it_orb_id_1@zajonzd690/10.2.4.13] (IT_MONITORING:203) I
- Server Interaction begin
```

In addition, when the `actional-sdk.jar` is used, it prints the following logging statement to `stderr`:

```
2009-11-05 13:30:43.070+0000 Actional logging to System.err
```

C++ example

The following are some example logging statements for Orbix C++ client and server applications:

```
Thu, 05 Nov 2009 13:38:32.0000000 [ZAJONZD690:4584] (IT_MONITORING:4) I -
ServerInteraction url: Simple/SimpleObject opname: call_me self: 10.2.4.13
peer: 10.2.4.13
Thu, 05 Nov 2009 13:38:32.0000000 [ZAJONZD690:5688] (IT_MONITORING:4) I -
ClientInteraction url: Simple/SimpleObject opname: call_me peer: 10.2.4.13
```

Configuring Actional for Orbix Integration

This chapter gives some basic guidelines on setting up Actional to run the Orbix Actional integration demo.

Prerequisites

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

Actional products

The following Actional products should be installed:

- Actional Management Server 8.0 (Actional server)
- Actional Flex Point 8.0 (Actional agent/intermediary)

  Alternatively, the following Actional products can be installed separately:
- Actional Point of Operational Visibility 8.0 (Actional agent)
- Actional Client Security Enforcement 8.0 (Actional intermediary)

Actional agents

You must ensure that Actional agents have been set up on each Orbix host node that you wish to manage. The provisioning of Actional agents is performed using the Actional server. For some basic details, see Configuring Actional for Orbix Integration.

For full details on how to set up Actional agents on managed nodes, see the Actional product documentation.

Further information

For information on installing Actional products, and the full range of platform and database versions supported by Actional, see the Actional product documentation.

This Orbix integration with Actional supports the full range of operating systems and compilers supported by Orbix. For more details, see the Orbix Installation Guide.

Configuring Actional

This section provides some basic configuration guidelines on Actional agent and server configuration. For full details, see the Actional product documentation.

This basic configuration helps to set up the Orbix actional_demo. For information on how to run this demo, see the `README` text files in the following directory:

```
OrbixInstallDir/asp/6.3/demos/corba/orb/actional_demo
```

Actional agent configuration

No specific Actional agent configuration settings are required for integration with Orbix. For example, for the purposes of the Orbix–Actional integration demos, the Actional agent can be started with the default configuration settings.

Actional server configuration

The following sample configuration steps describe how to set up the Actional server to run an simple Orbix–Actional demo:

1. Install the Actional server with typical installation options, and select the Apache Derby database.

> **Note**
>
> The Apache Derby database is provided for demo purposes only, and is not recommended for a production environment.

2. Specify the following URL in your browser:

   `http://localhost:4040/lgserver`

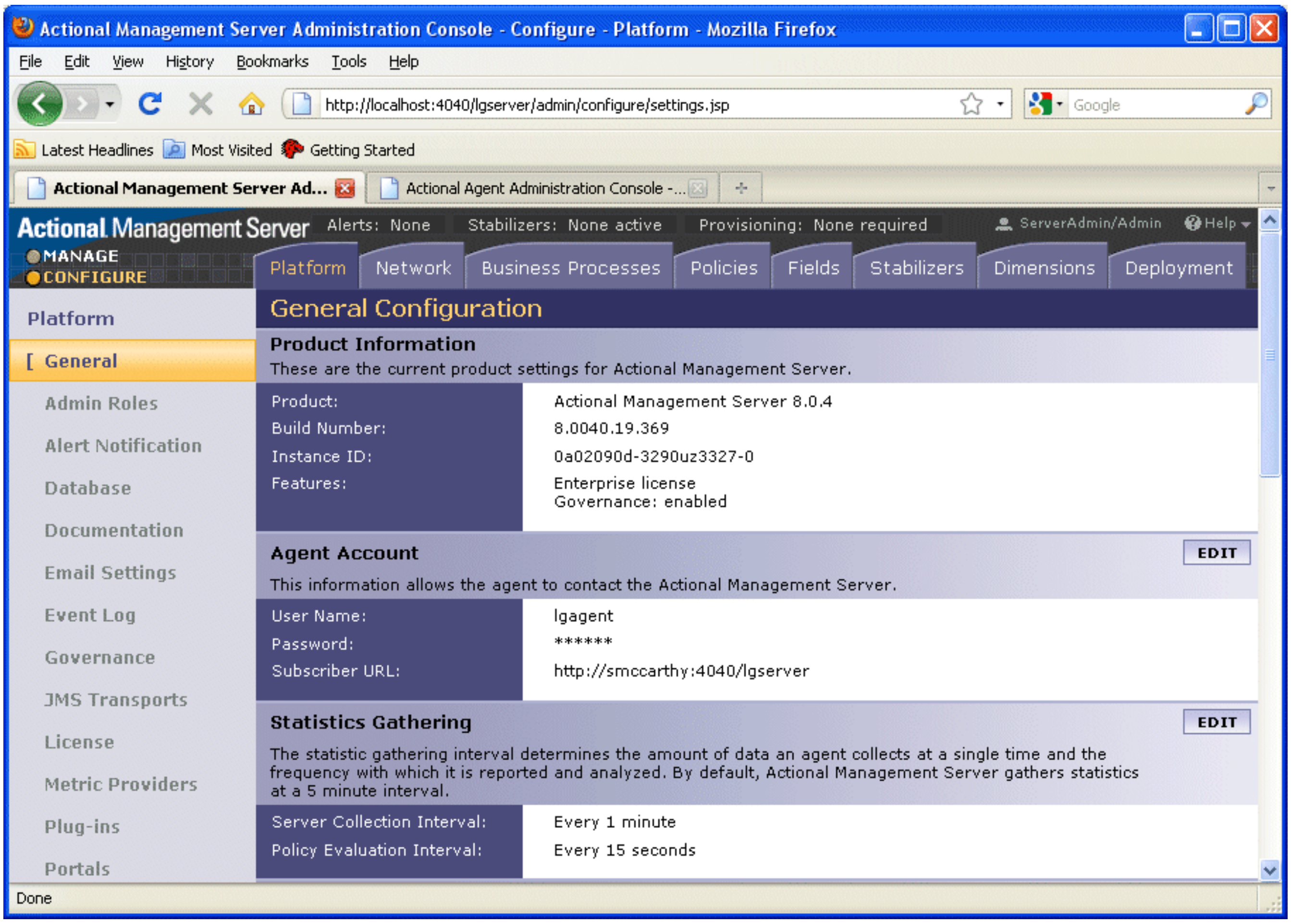
3. If this is a new installation click **Start**, and follow the new Actional server setup steps.

   Otherwise, if the Actional server is already installed, perform the following steps:

   i. In the Actional console Web interface, select the **Configure** radio button in the top left of the screen.

   ii. Select the **Platform** tab. This displays the general configuration settings, as shown in Figure 10.

   Figure 10 Actional Server Configuration Settings


Actional Management Server Administration Console - Configure - Platform - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:4040/lgserver/admin/configure/settings.jsp
Google
Latest Headlines Most Visited Getting Started
Actional Management Server Ad...
Actional Agent Administration Console -...
Actional Management Server
Alerts: None Stabilizers: None active Provisioning: None required
ServerAdmin/Admin Help
MANAGE
CONFIGURE
Platform Network Business Processes Policies Fields Stabilizers Dimensions Deployment
Platform
[General
Admin Roles
Alert Notification
Database
Documentation
Email Settings
Event Log
Governance
JMS Transports
License
Metric Providers
Plug-ins
Portals
EDIT
EDIT
Done

General Configuration

Product Information

These are the current product settings for Actional Management Server.

Product:	Actional Management Server 8.0.4
Build Number:	8.0040.19.369
Instance ID:	0a02090d-3290uz3327-0
Features:	Enterprise license Governance: enabled

Agent Account

This information allows the agent to contact the Actional Management Server.

User Name:	lgagent
Password:	******
Subscriber URL:	http://smccarthy:4040/lgserver

Statistics Gathering

The statistic gathering interval determines the amount of data an agent collects at a single time and the frequency with which it is reported and analyzed. By default, Actional Management Server gathers statistics at a 5 minute interval.

Server Collection Interval:	Every 1 minute
Policy Evaluation Interval:	Every 15 seconds

Creating a managed node

To create a managed node for a simple Orbix demo, perform the following steps:

1. In the Actional **Configure** view menu bar, open the **Network** tab. This displays the **Network Nodes**.
2. Select **Add**. This displays **Node Creation / Managing Agents**.
3. Click **Managed Node**.

Configuring a new node

To configure a managed node for the demo, perform the following steps in the wizard:

Step 1: New Node - Identification

1. Specify the **Name** as `agent1`.
2. Specify the **Display icon** as `Auto Discover`.
3. Click **Next**.

Step 2: New Node - Management

1. Specify the **Transport** as `HTTP/S`.
2. Supply your Actional agent user name and password.
3. Ensure that **Override Agent Database** is checked.
4. Click **Next**.

Step 3: New Node - Agents

1. Specify the following URL:

   `http://HostName:4041/lgagent`

   You can specify a host name or an IP address in this URL.
2. Click **Add**. The agent URL is added.
3. Click **Next**.

Step 4: New Node - Endpoints

1. For **Endpoints**, add the hostname, fully qualified hostname, or IP address.
2. Click **Next**.

Step 5: New Node - Filters

1. Do not specify any filters for the demo.
2. Click **Next**.

Step 6: New Node - Trust Zone

1. Do not specify a trust zone for the demo.
2. Click **Finish**

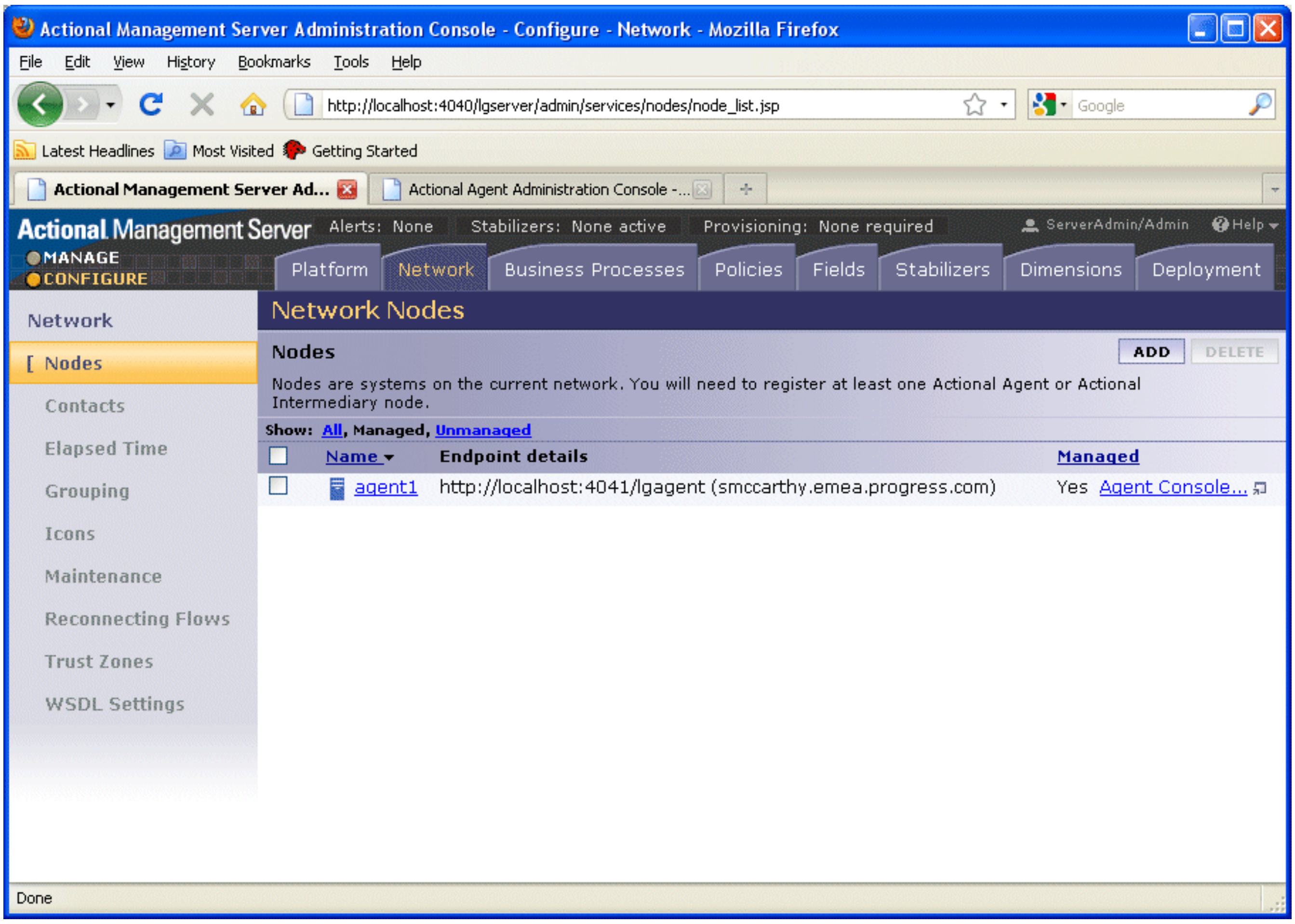
The newly created managed node now needs to be provisioned.

Provisioning a new node

To provision the new node to bring it under management, perform the following steps:

1. Select the **Configure** radio button at the top left of the screen.
2. Select the **Deployment** tab from the **Configure** menu bar.
3. The **Provisioning** page is displayed, and `agent1` is listed as not provisioned.
4. Select the `agent1` check box.
5. Click **Provision**. This displays a message when complete: `Successfully provisioned`.
6. Click the **Manage** radio button at the top left of the screen. You should see `agent1` added to the **Network** view as shown in Figure 11.

Figure 11 Actional Server Provisioned Node


Actional Management Server Administration Console - Configure - Network - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:4040/lgserver/admin/services/nodes/node_list.jsp
Google
Latest Headlines Most Visited Getting Started
Actional Management Server Ad...
Actional Agent Administration Console -...
Actional Management Server
Alerts: None
Stabilizers: None active
Provisioning: None required
ServerAdmin/Admin
Help
MANAGE
CONFIGURE
Platform
Network
Business Processes
Policies
Fields
Stabilizers
Dimensions
Deployment
Network
[Nodes
Contacts
Elapsed Time
Grouping
Icons
Maintenance
Reconnecting Flows
Trust Zones
WSDL Settings
Network Nodes
Nodes
ADD
DELETE
Done

Nodes are systems on the current network. You will need to register at least one Actional Agent or Actional Intermediary node.

Show: All, Managed, Unmanaged

	Name	Endpoint details	Managed
☐	agent1	http://localhost:4041/lgagent (smccarthy.emea.progress.com)	Yes Agent Console...

Further information

For more details on setting up and running Actional SOA management tools, see the Actional product documentation.

Troubleshooting Actional

This section provides some tips to help troubleshoot your Actional integration with Orbix.

Setting default polling

For demonstration purposes, to update the display in your Actional server console more frequently, you can set the default polling to a shorter time span as follows:

1. Select the **Configure** radio button at the top left of the screen.
2. Select the **Platform** tab from the **Configure** menu bar.
3. In **Statistics Gathering** on the right, select **EDIT**.
4. Set the **Server Collection Interval** to 1 minute by using the drop down list.
5. Set the **Policy Evaluation Interval** to 15 seconds.

Note

These settings are for demonstration purposes only, and may not be suitable for a production environment.

Ensuring events are reported to the Actional Agent

To ensure that Orbix monitoring events are being reported to your Actional agent, perform the following steps:

1. Ensure your Actional agent is running, and added as a managed node in your Actional server.
2. Verify that the agent generated the `Uplink.cfg` file in the directory specified during installation. If this file was not specified during the installation, it should be in the following default path (which should have write permission):

UNIX	`/var/opt/actional/LG.Interceptor`
Windows	`%systemroot%\system32\LG.Interceptor`

3. Open your Actional agent console and login:

   `http://AgentHostName:Port/lgagent/`

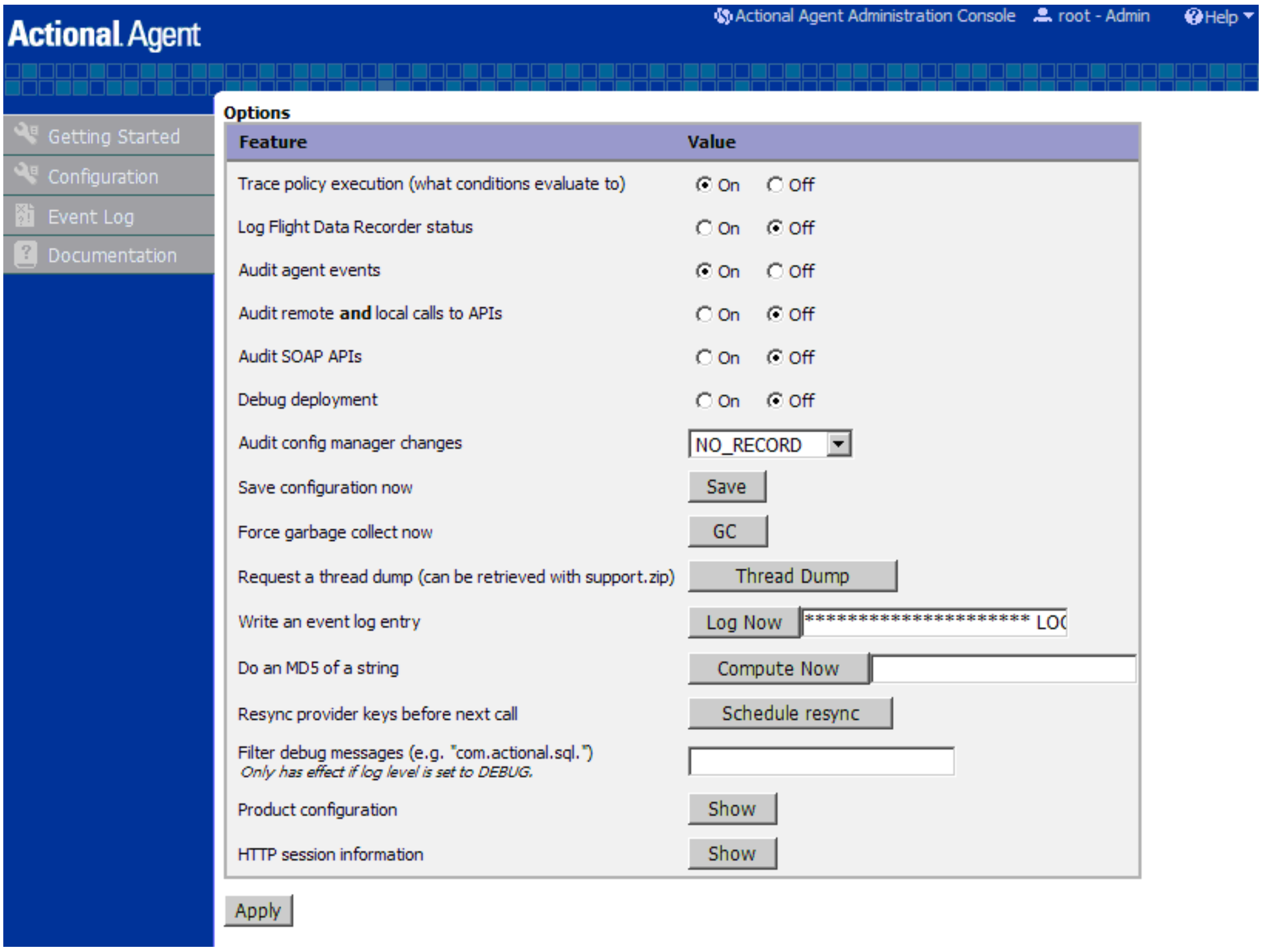
4. Specify the following URL to display the **Options** page shown in Figure 12:

   `http://AgentHostName:Port/lgagent/admin/options.js`

5. For **Audit agent events**, Click **On**.

6. Click **Apply**.

   Figure 12 Actional Agent Options



Feature	Value
Trace policy execution (what conditions evaluate to)	[✓ On] Off
Log Flight Data Recorder status	On [✓ Off]
Audit agent events	[✓ On] Off
Audit remote **and** local calls to APIs	On [✓ Off]
Audit SOAP APIs	On [✓ Off]
Debug deployment	On [✓ Off]
Audit config manager changes	NO_RECORD
Save configuration now	Save
Force garbage collect now	GC
Request a thread dump (can be retrieved with support.zip)	Thread Dump
Write an event log entry	Log Now ********************** LOG
Do an MD5 of a string	Compute Now
Resync provider keys before next call	Schedule resync
Filter debug messages (e.g. "com.actional.sql.") *Only has effect if log level is set to DEBUG.*	
Product configuration	Show
HTTP session information	Show

Apply

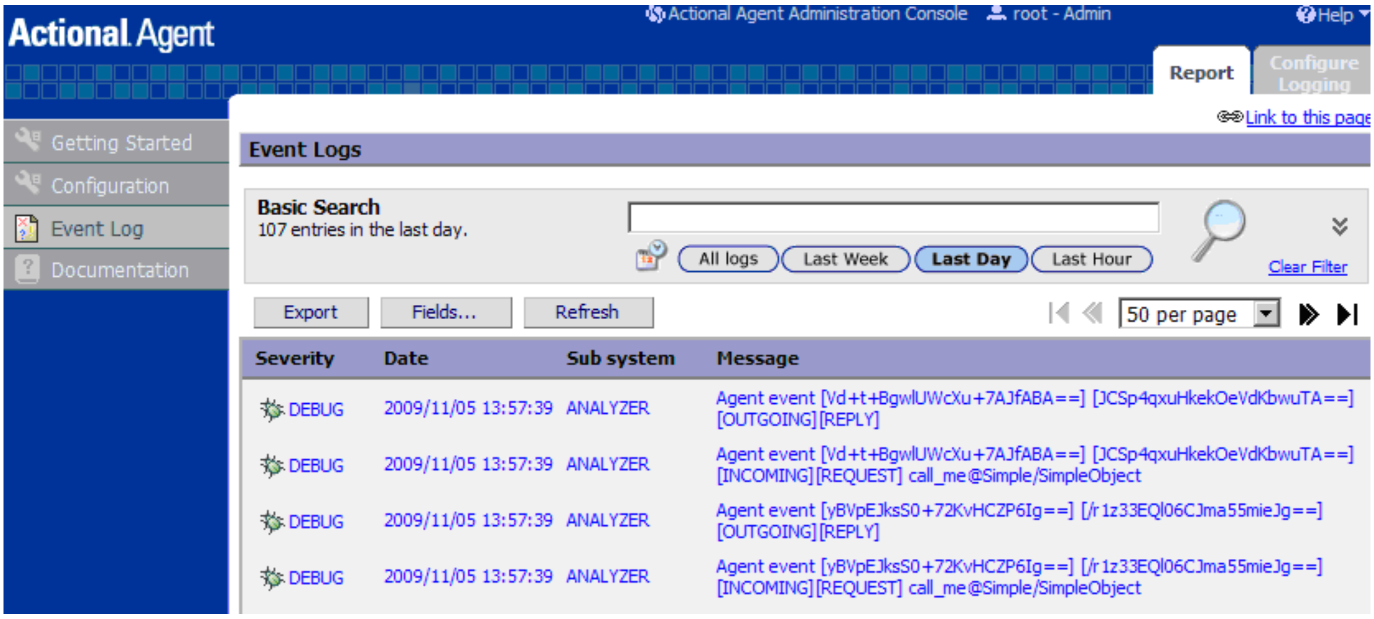
Note

These settings are not persistent, and are reset when the Actional agent is restarted.

Viewing agent events

When **Audit agent events** is turned on, all external events coming from the Orbix monitoring plug-in can be reviewed in the Actional agent **Event Logs**, shown in Figure 13.

Figure 13 Actional Agent Event Logs



Severity	Date	Sub system	Message
DEBUG	2009/11/05 13:57:39	ANALYZER	Agent event [Vd+t+BgwlUWcXu+7AJfABA==] [JCSp4qxuHkekOeVdKbwuTA==] [OUTGOING] [REPLY]
DEBUG	2009/11/05 13:57:39	ANALYZER	Agent event [Vd+t+BgwlUWcXu+7AJfABA==] [JCSp4qxuHkekOeVdKbwuTA==] [INCOMING] [REQUEST] call_me@Simple/SimpleObject
DEBUG	2009/11/05 13:57:39	ANALYZER	Agent event [yBVpEJksS0+72KvHCZP6Ig==] [/r1z33EQl06CJma55mieJg==] [OUTGOING] [REPLY]
DEBUG	2009/11/05 13:57:39	ANALYZER	Agent event [yBVpEJksS0+72KvHCZP6Ig==] [/r1z33EQl06CJma55mieJg==] [INCOMING] [REQUEST] call_me@Simple/SimpleObject

Figure 13 shows `INCOMING`, `OUTGOING`, `REQUEST`, and `REPLY` events reported from the monitoring plug-in. If these events are not reported, the path for the `uplink.cfg` may be incorrect, and the monitoring plug-in can not find the agent.

C++ applications

For C++ applications, verify that the `LG_INTERCEPTORCONFIG` environment variable is set correctly, and points to the directory where the agent has written the `uplink.cfg` file.

Java applications

For Java applications, verify that the `com.actional.lg.interceptor.config` property is passed on to the application correctly, and points to the directory where the agent has written the `uplink.cfg` file. For example:

```
java -Dcom.actional.lg.interceptor.config=%SystemRoot%
\system32\LG.Interceptor -classpath .\java\classes;"%CLASSPATH%"
actional_demo.Server -ORBname demos.actional_demo
```

When incoming monitoring events are arriving at the agent, and the agent is configured correctly, you should see the calls displayed in the Actional server console **Network** view, as shown in Managing Orbix Applications in Actional.

Further information

For any problems with Actional agent configuration, please refer to the Actional product documentation.

Managing Orbix Applications in Actional

This chapter shows examples of managing a simple Orbix application and Orbix domain services in Actional SOA management tools.

Monitoring Orbix Applications

When your Orbix applications have been configured for integration with Actional, they can be monitored using the Actional SOA management tools. No code changes are required for monitoring of Orbix applications.

For example, when you run the simple Orbix `actional_demo`, the **Actional Management Server Administration Console** displays the managed node that the demo is running on. Invocations are displayed as arrows flowing to and from managed components.

The Orbix `actional_demo` illustrates the simple use of the ORB monitoring plug-in to report calls made between Orbix clients and servers to Actional. This demo is similar to `demos/corba/orb/simple`, and shows how to configure visibility of your application in Actional. For details on how to run this demo, see the `README` text files in the following directory:

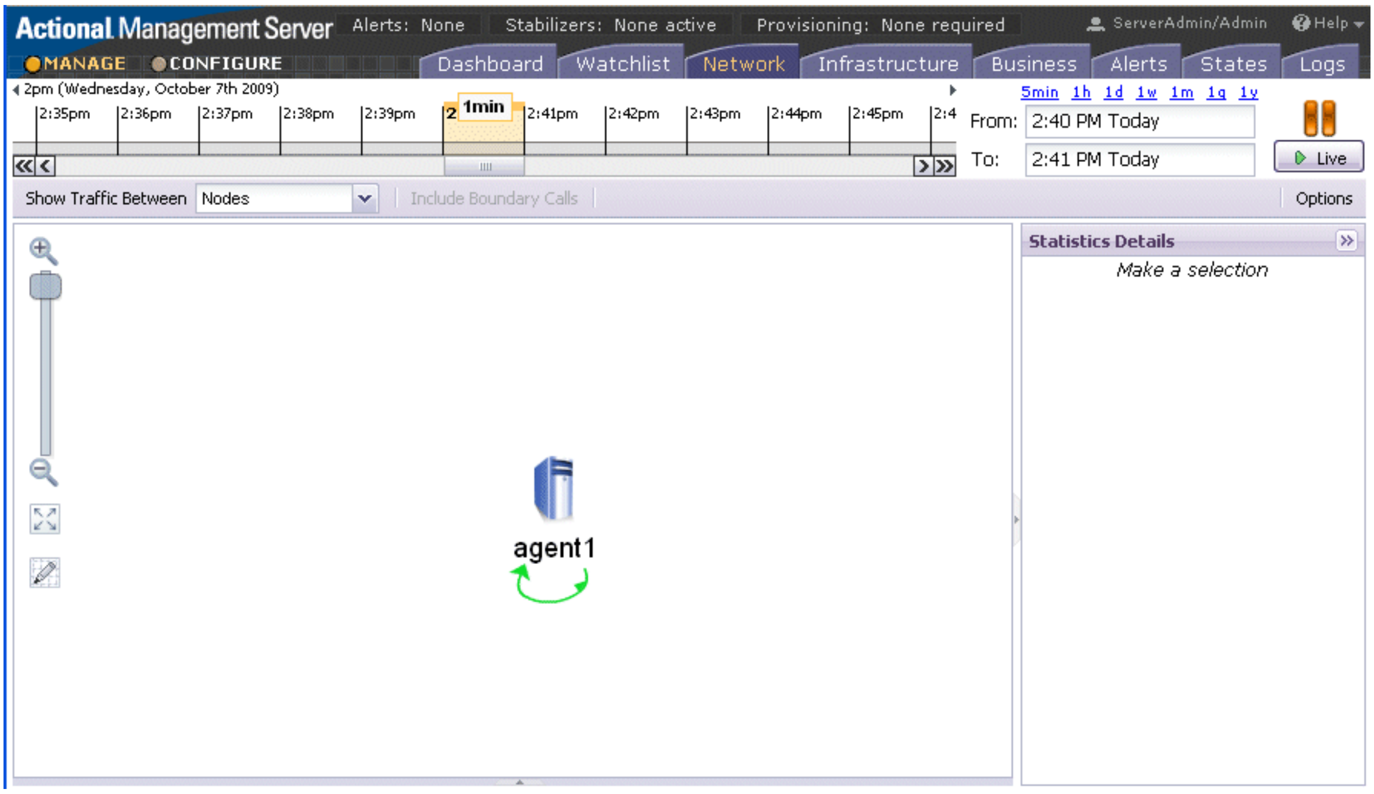
```
OrbixInstallDir/asp/6.3/demos/corba/orb/actional_demo
```

Network view

The Actional network view displays the traffic between various components in your network environment. These include nodes, packages, services and operations.

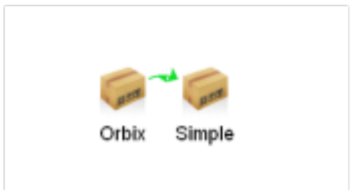
Figure 14 shows the running Orbix `actional_demo` displayed in the **Network** tab of the **Actional Management Server Administration Console**. In this simple demo, the **Network** tab displays the Actional agent on the Orbix managed node that the demo is running on. This agent reports the monitoring data back to the Actional server. The single invocation is displayed as a green arrow flowing from the node and back to itself. In more complex examples with multiple nodes, the arrows flow between nodes.

Figure 14 Actional Server Network View



By default, the **Network** view shows traffic between nodes. There is only one node in this case. You can also select to show traffic between packages in the top left of the screen. Figure 15 shows the traffic between the Orbix client and server packages.

Figure 15 Traffic Between Packages

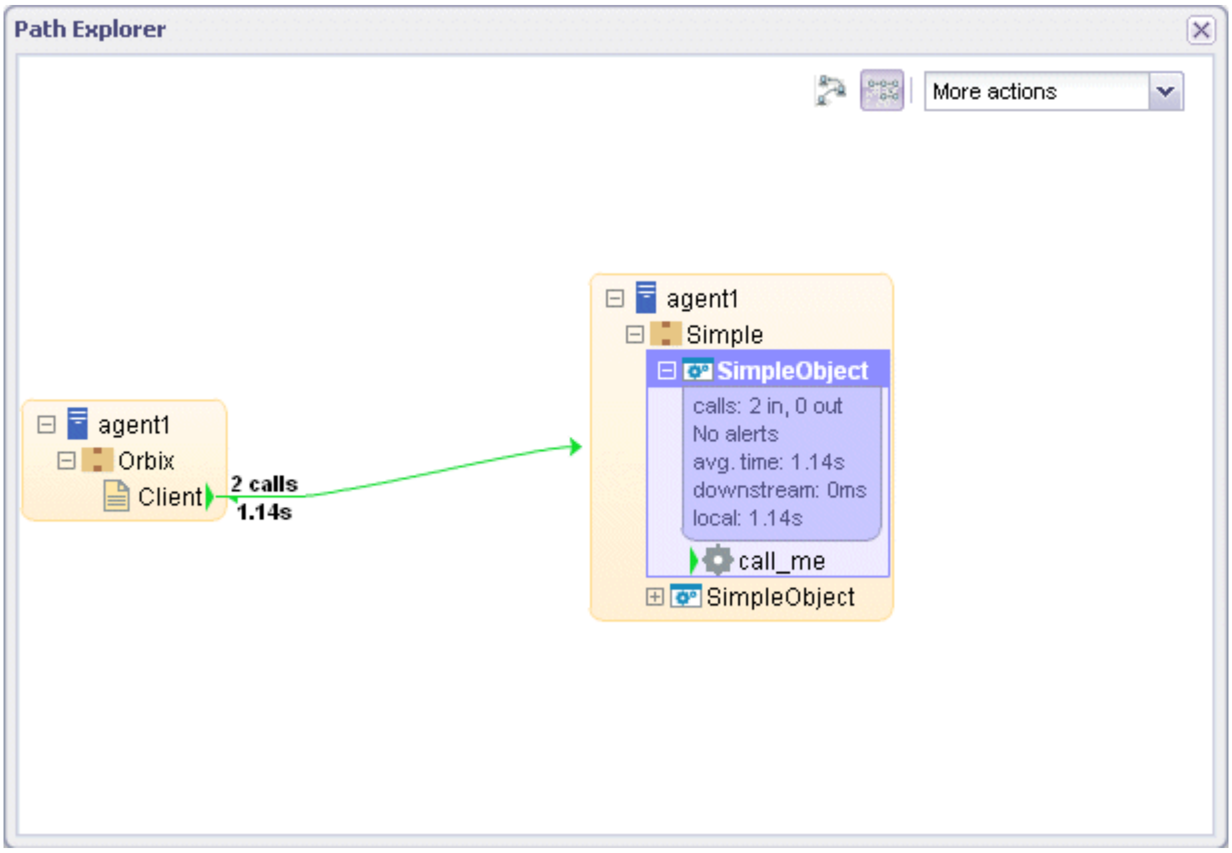


Path Explorer

Figure 16 shows the Orbix `actional_demo` displayed in the **Path Explorer** view of the **Actional Management Server Administration Console**.

To view this screen, double click on the managed node shown in Figure 14. Alternatively, click the **Display Path Explorer** button at the top right of the **Network** view.

Figure 16 Actional Server Path Explorer



calls: 2 in, 0 out
No alerts
avg. time: 1.14s
downstream: 0ms
local: 1.14s

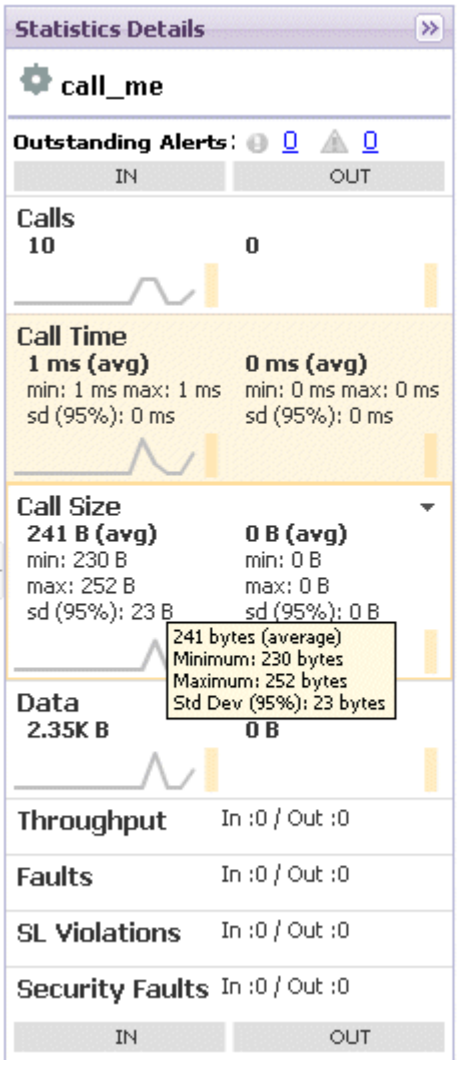
The **Path Explorer** view displays the relationships between different components in more detail. For example, you can view the call chain between services and consumers. Summary statistics are also displayed for the selected component.

Statistics details

The **Statistics Details** pane on the right displays statistics gathered by the selected component. These include the number of incoming and outgoing calls, call time, call size, and so on. Alerts, faults and violations are also displayed.

For example, Figure 17 shows the **Statistics Details** displayed on the right when the `call_me()` operation is selected in the **Path Explorer**.

Figure 17 Actional Server Statistics Details



Statistics Details

call_me

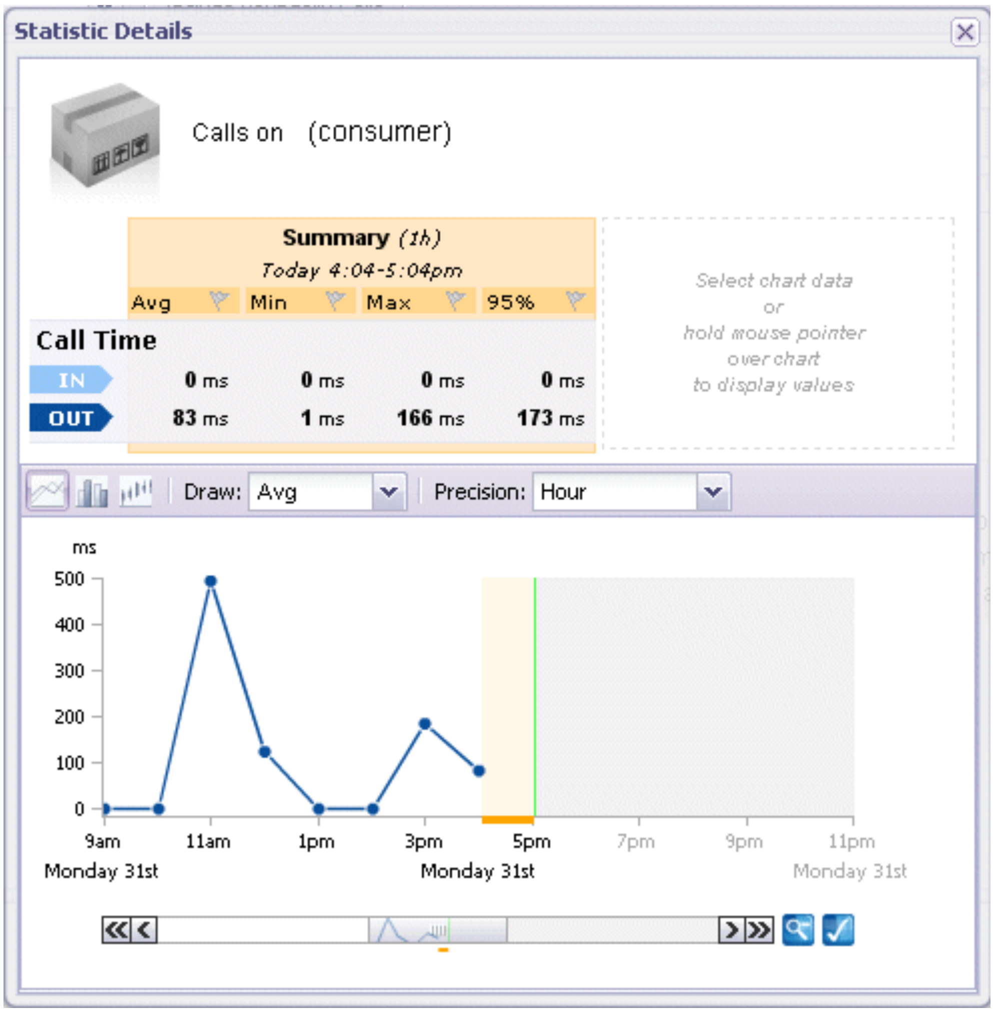
Outstanding Alerts: 0 0

	IN	OUT
Calls	10	0
Call Time	1 ms (avg) min: 1 ms max: 1 ms sd (95%): 0 ms	0 ms (avg) min: 0 ms max: 0 ms sd (95%): 0 ms
Call Size	241 B (avg) min: 230 B max: 252 B sd (95%): 23 B	0 B (avg) min: 0 B max: 0 B sd (95%): 0 B
Data	2.35K B	0 B
Throughput	In :0 / Out :0	
Faults	In :0 / Out :0	
SL Violations	In :0 / Out :0	
Security Faults	In :0 / Out :0	

241 bytes (average)
Minimum: 230 bytes
Maximum: 252 bytes
Std Dev (95%): 23 bytes

Double clicking on a particular statistic in this view (for example, **Call Size**) displays a summary chart. For example, Figure 18 shows a **Call Time** summary chart for the consumer.

Figure 18 Actional Server Statistics Chart



	Summary (1h) Today 4:04-5:04pm			
Call Time	Avg	Min	Max	95%
IN	0 ms	0 ms	0 ms	0 ms
OUT	83 ms	1 ms	166 ms	173 ms

Server manifest

The Actional server manifest (`LG_Header`) is a unique ID used by the Actional server to correlate information it receives from agents about interactions between different applications. For example, when you run theclient application in the Orbix `actional_demo,` the following `LG_Header` is output on the command line:

```
Interaction=CgIEAUD6LU2sLiQBBwAAAA==;
Locus=4/LcwgqvldfxotEoegsSGg==;
Flow=CgIEAUD6LU2sLiQBBgAAAA==;
UpstreamOpID=xPnAfuwlTEV7QGYoGRBgYA==;
CallerAddress=10.2.4.1;
```

For more details, see Actional server manifest.

Further information

For detailed information on using Actional SOA management tools, see the Actional product documentation.

Monitoring Orbix Domain Services

Orbix configuration domain services can be integrated with Actional automatically using the **Orbix Configuration** tool. These include services such as the Orbix configuration repository, locator daemon, node daemon, and so on. No manual configuration updates are required. For more details, see Configuring an Orbix Domain

This section shows examples of monitoring Orbix domain services in Actional SOA management tools.

Starting Orbix services

To start your Orbix configuration domain services, perform the following steps:

1. Set your Orbix domain environment, for example:

```
c:\orbix\etc\bin>actional-cfr-domain_env.bat
Setting environment for domain actional-cfr-domain
```

   You must have configured your domain to be monitored by Actional (see Configuring an Orbix Domain).

2. Start your domain services, for example:

```
c:\orbix\etc\bin>start_actional-cfr-domain_services.bat
Orbix services logging to: C:\orbix\var\actional-cfr-domain\logs
Starting iona_services.config_rep.Hostname
Starting iona_services.locator.Hostname
Starting iona_services.node_daemon.Hostname
Finished.
```

Monitoring Orbix services

Figure 19 shows the traffic between packages for the Orbix configuration domain services. The services displayed are the node daemon, configuration repository, and locator daemon.

Figure 19 Traffic Between Domain Services Packages

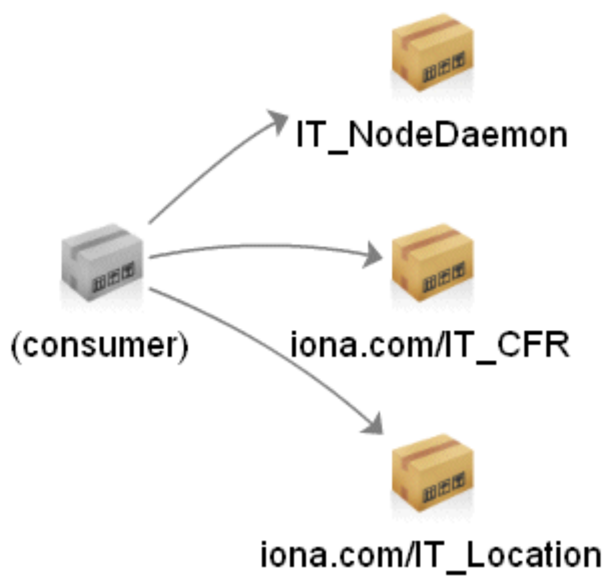


Figure 20 shows the running Orbix domain services displayed in the **Path Explorer** view.

Figure 20 Domain Services in Path Explorer

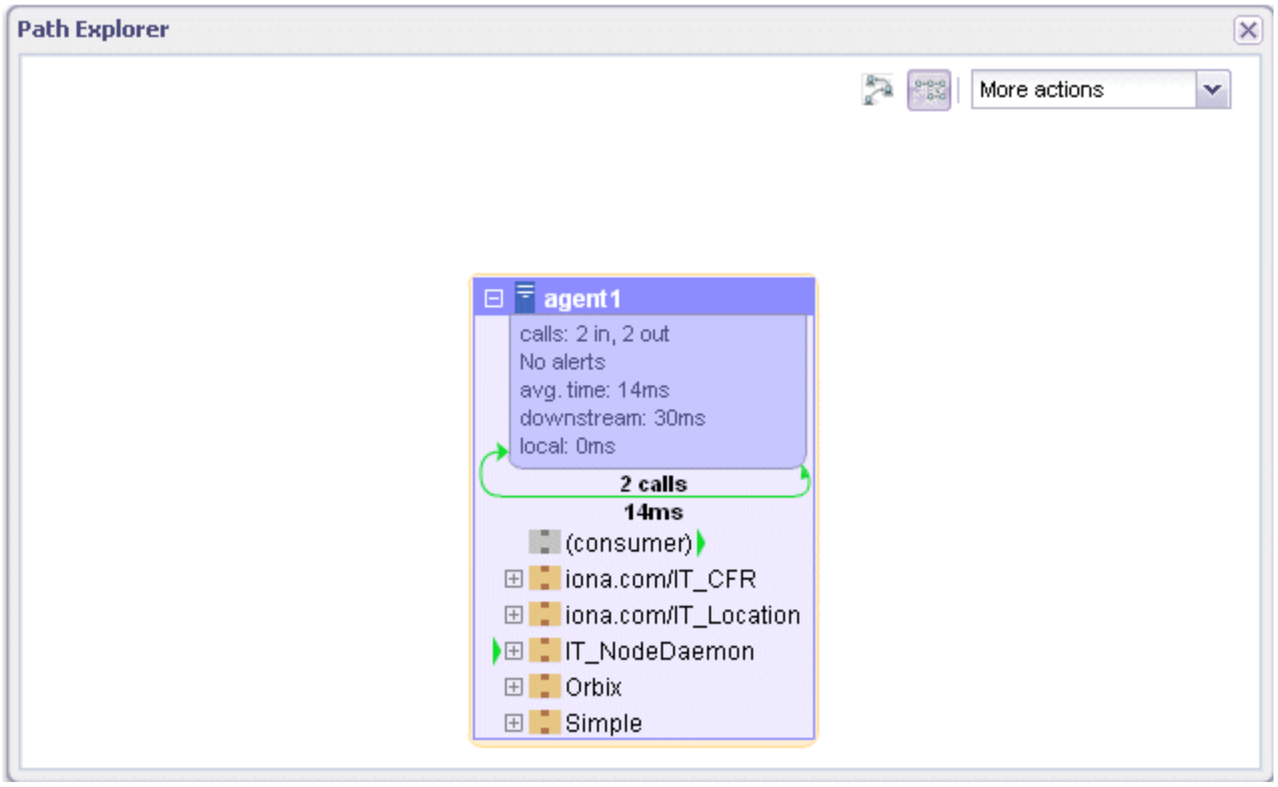
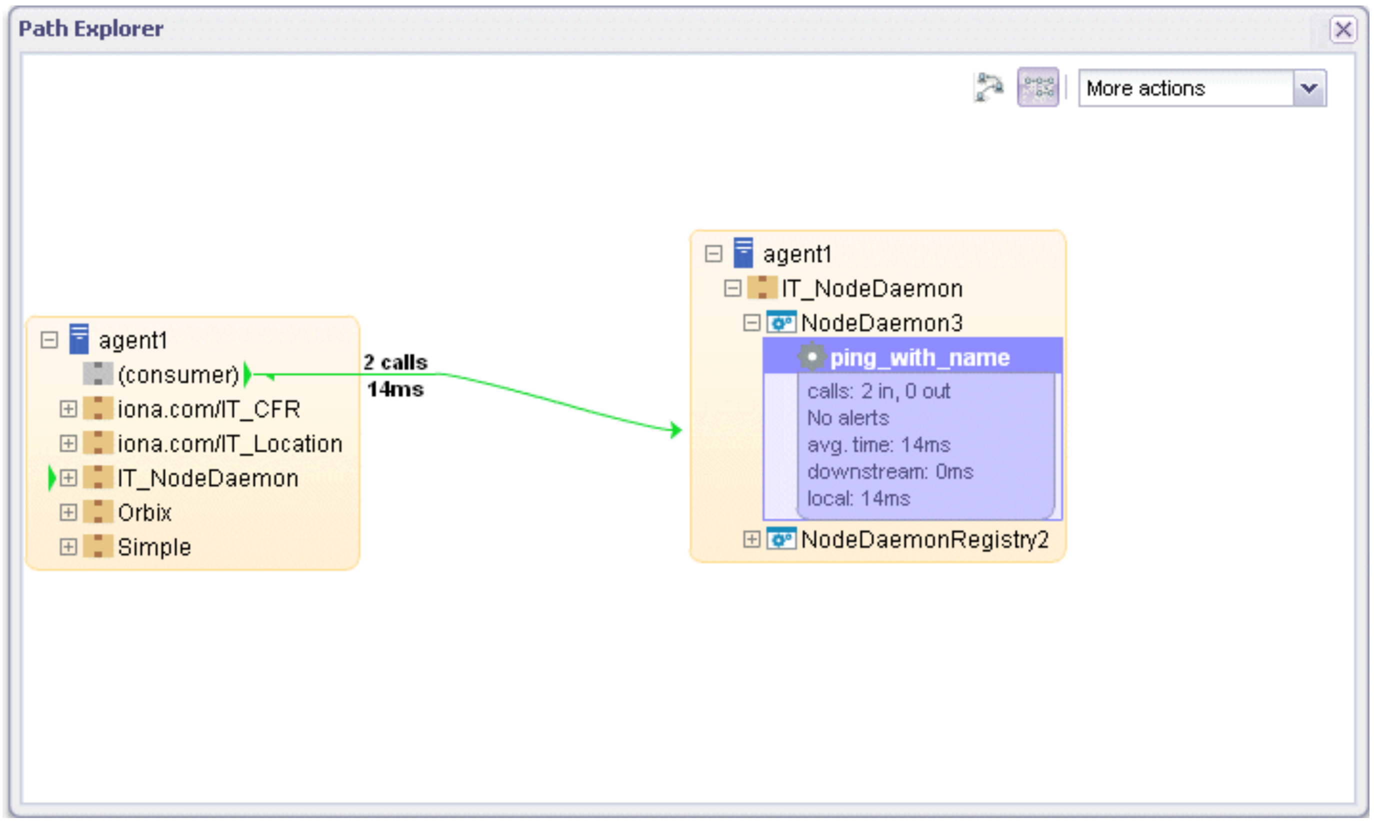


Figure 21 shows the call displayed for the node daemon `ping_with_name()` operation:

Figure 21 Node Daemon Operation



Further information

For detailed information on using Actional SOA management tools, see the Actional product documentation.

Auditing Orbix Applications

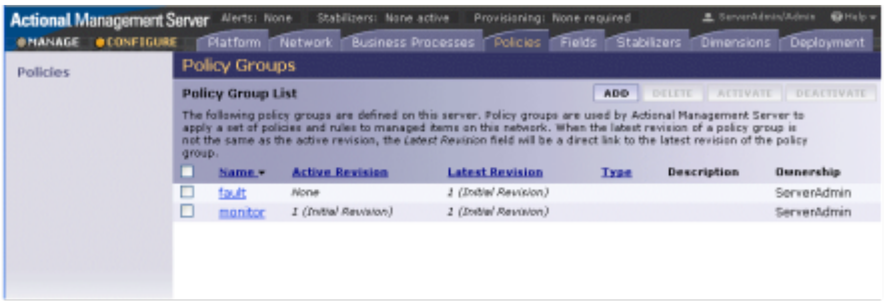
This section shows some simple examples of auditing the Orbix `actional_demo` and Orbix domain services.

Actional policy groups

Policy groups are used by Actional server to apply a set of policies and rules to managed items on your network. Policies and rules can be used to raise alerts on certain failure reasons. For example, when an Orbix operation takes too long to return, or when a specified IDL exception or fault is raised.

Figure 22 shows some example policy groups that have be defined in the **Policies** view.

Figure 22 Actional Policy Groups



Policy Group List

The following policy groups are defined on this server. Policy groups are used by Actional Management Server to apply a set of policies and rules to managed items on this network. When the latest revision of a policy group is not the same as the active revision, the *Latest Revision* field will be a direct link to the latest revision of the policy group.

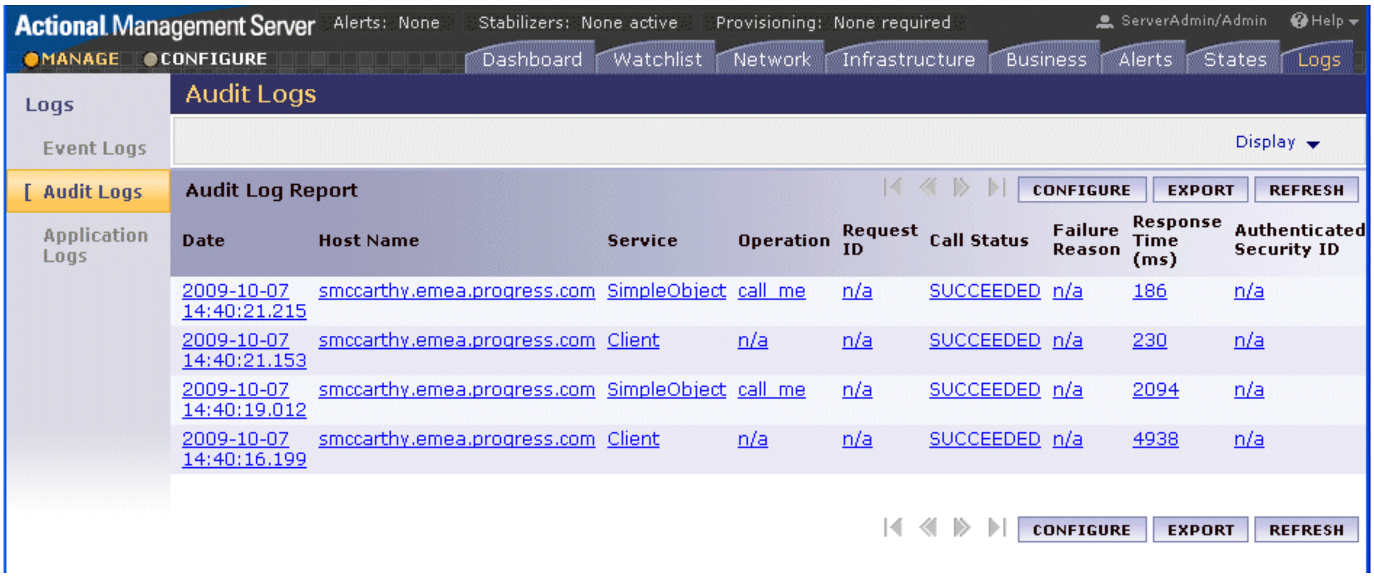
Name	Active Revision	Latest Revision	Type	Description	Ownership
fault	*None*	*1 (Initial Revision)*			ServerAdmin
monitor	*1 (Initial Revision)*	*1 (Initial Revision)*			ServerAdmin

Viewing audit logs

When you have defined policies for your network, you can use them to audit and monitor alerts on certain failure reasons (for example, when a specified IDL exception or fault is raised).

Figure 23 shows some example audit logs for the Orbix `actional_demo` in the **Logs** view.

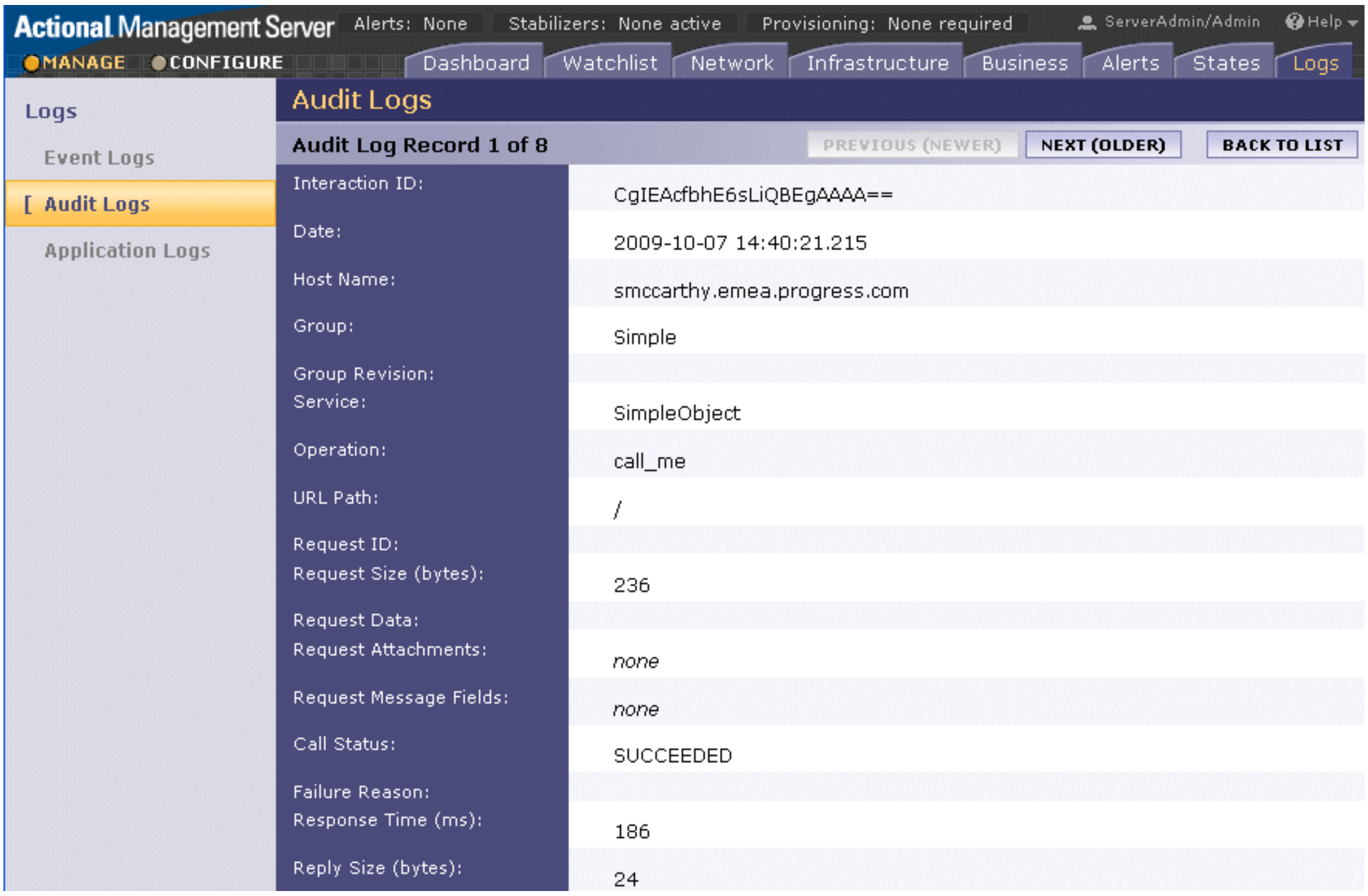
Figure 23 Actional Demo Audit Logs



Date	Host Name	Service	Operation	Request ID	Call Status	Failure Reason	Response Time (ms)	Authenticated Security ID
2009-10-07 14:40:21.215	smccarthy.emea.progress.com	SimpleObject	call_me	n/a	SUCCEEDED	n/a	186	n/a
2009-10-07 14:40:21.153	smccarthy.emea.progress.com	Client	n/a	n/a	SUCCEEDED	n/a	230	n/a
2009-10-07 14:40:19.012	smccarthy.emea.progress.com	SimpleObject	call_me	n/a	SUCCEEDED	n/a	2094	n/a
2009-10-07 14:40:16.199	smccarthy.emea.progress.com	Client	n/a	n/a	SUCCEEDED	n/a	4938	n/a

Figure 24 shows an example audit log record displayed when clicking an entry for the Orbix `actional_demo` in Figure 23.

Figure 24 Actional Demo Audit Log Record

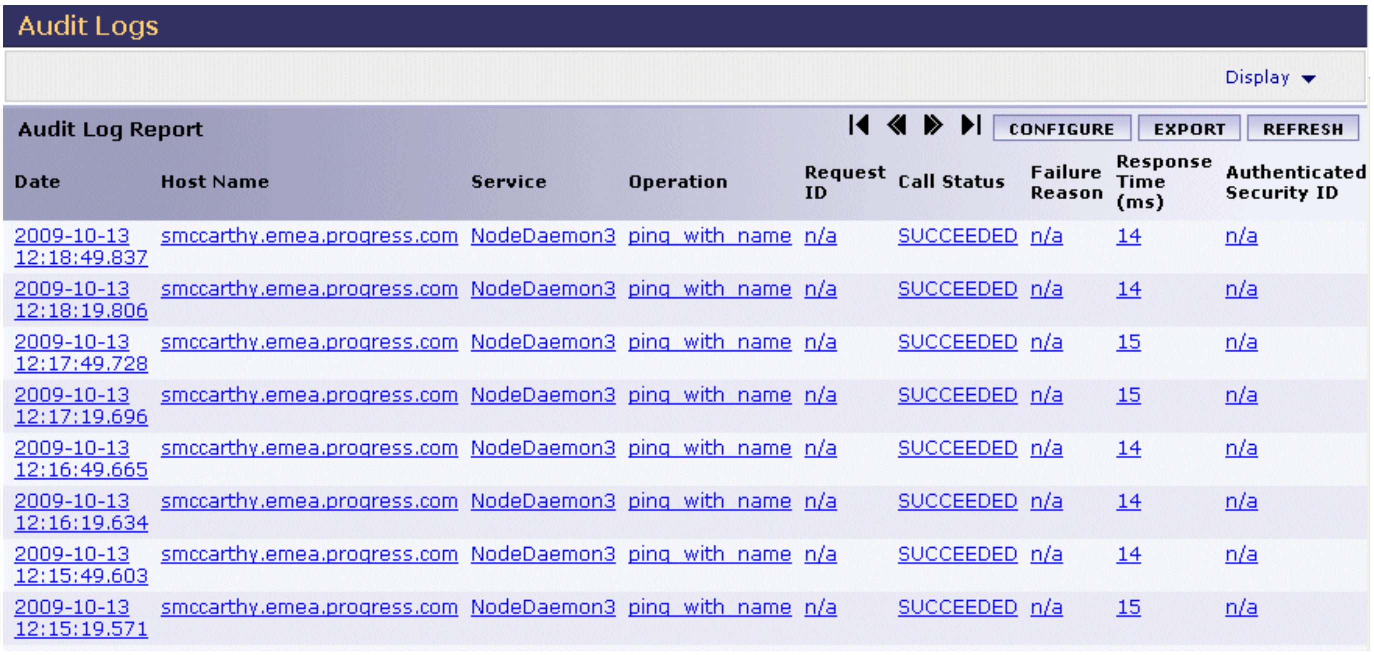


Interaction ID:	CgIEAcfbhE6sLiQBEgAAAA==
Date:	2009-10-07 14:40:21.215
Host Name:	smccarthy.emea.progress.com
Group:	Simple
Group Revision:	
Service:	SimpleObject
Operation:	call_me
URL Path:	/
Request ID:	
Request Size (bytes):	236
Request Data:	
Request Attachments:	*none*
Request Message Fields:	*none*
Call Status:	SUCCEEDED
Failure Reason:	
Response Time (ms):	186
Reply Size (bytes):	24

The **Interaction ID** displayed at the top of the screen is used by the Actional server to correlate information it receives, from multiple agents, about interactions between different services. For more details, see Actional server manifest.

Figure 25 shows some example audit logs for Orbix configuration domain services in the **Logs** view. The Orbix service displayed in this example is the Orbix node daemon.

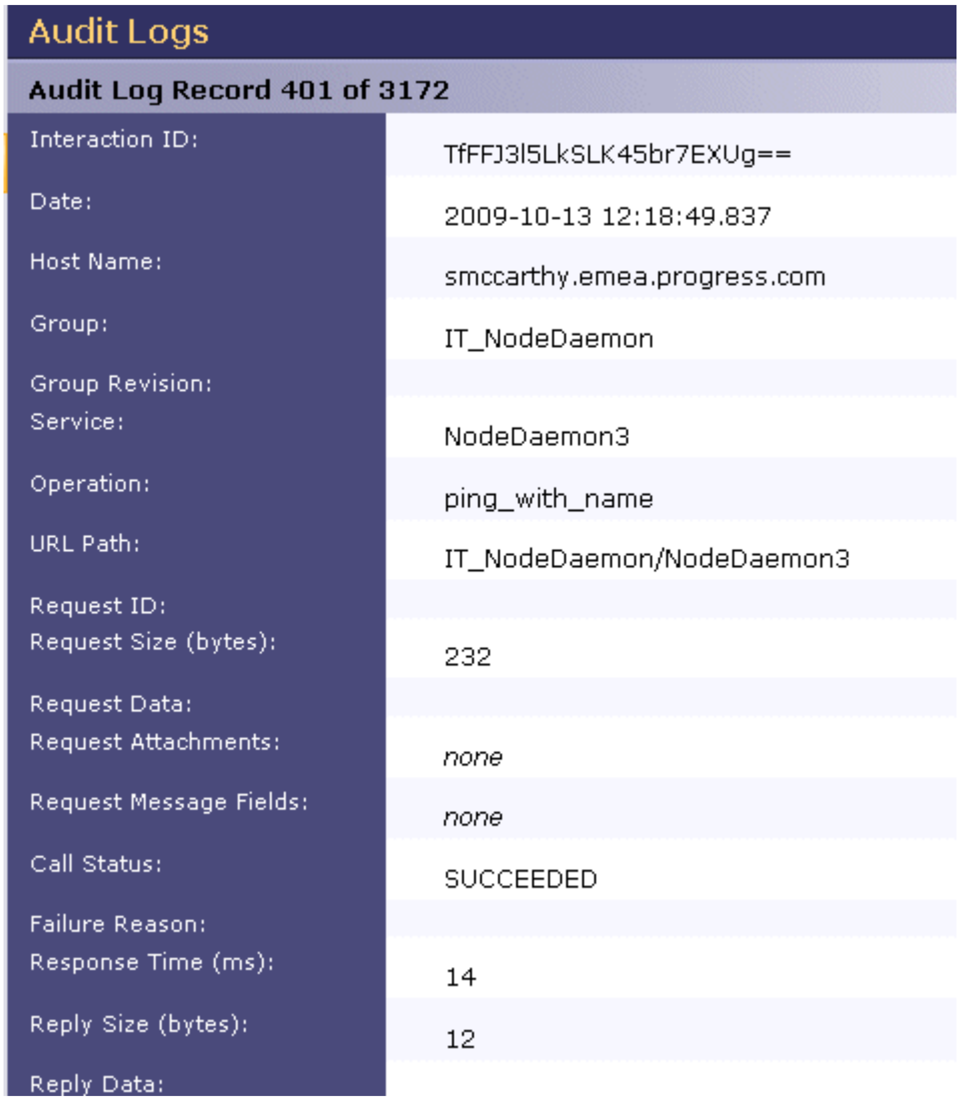
Figure 25 Domain Services Audit Logs



Date	Host Name	Service	Operation	Request ID	Call Status	Failure Reason	Response Time (ms)	Authenticated Security ID
2009-10-13 12:18:49.837	smccarthy.emea.progress.com	NodeDaemon3	ping_with_name	n/a	SUCCEEDED	n/a	14	n/a
2009-10-13 12:18:19.806	smccarthy.emea.progress.com	NodeDaemon3	ping_with_name	n/a	SUCCEEDED	n/a	14	n/a
2009-10-13 12:17:49.728	smccarthy.emea.progress.com	NodeDaemon3	ping_with_name	n/a	SUCCEEDED	n/a	15	n/a
2009-10-13 12:17:19.696	smccarthy.emea.progress.com	NodeDaemon3	ping_with_name	n/a	SUCCEEDED	n/a	15	n/a
2009-10-13 12:16:49.665	smccarthy.emea.progress.com	NodeDaemon3	ping_with_name	n/a	SUCCEEDED	n/a	14	n/a
2009-10-13 12:16:19.634	smccarthy.emea.progress.com	NodeDaemon3	ping_with_name	n/a	SUCCEEDED	n/a	14	n/a
2009-10-13 12:15:49.603	smccarthy.emea.progress.com	NodeDaemon3	ping_with_name	n/a	SUCCEEDED	n/a	14	n/a
2009-10-13 12:15:19.571	smccarthy.emea.progress.com	NodeDaemon3	ping_with_name	n/a	SUCCEEDED	n/a	15	n/a

Figure 26 shows an example audit log record displayed on clicking an entry for the Orbix node daemon in Figure 25.

Figure 26 Node Daemon Log Record



Audit Logs

Audit Log Record 401 of 3172

Interaction ID:	TfFFJ3l5LkSLK45br7EXUg==
Date:	2009-10-13 12:18:49.837
Host Name:	smccarthy.emea.progress.com
Group:	IT_NodeDaemon
Group Revision:	
Service:	NodeDaemon3
Operation:	ping_with_name
URL Path:	IT_NodeDaemon/NodeDaemon3
Request ID:	
Request Size (bytes):	232
Request Data:	
Request Attachments:	*none*
Request Message Fields:	*none*
Call Status:	SUCCEEDED
Failure Reason:	
Response Time (ms):	14
Reply Size (bytes):	12
Reply Data:	

Further information

For detailed information on using Actional SOA management tools, see the Actional product documentation.

Glossary

A

Actional agent

Run on each host that you wish to manage, and used to provide instrumentation data back to the Actional server. It includes two main components: an analyzer, and one or more interceptors. The analyser gathers and evaluates data such as records, statistics, and alerts. The interceptors collect data about service traffic from an application server, and apply policies to that traffic.

Actional server

A central management server that manages nodes containing an Actional agent. The Actional server correlates the data it receives from each of its agents, and distributes policies to those agents. It enables an administrator to analyze service network data and create system-wide policies.

Actional server manifest

A token sent by the Actional server sends with a request document to provide information about the request's origin and the business flow that the request belongs to. The Actional server manifest (LG_Header) is used by the Actional server to correlate information it receives, from multiple agents, about interactions between different services. For this reason, the server manifest is sometimes referred to as a correlation ID.

administration

All aspects of installing, configuring, deploying, monitoring, and managing a system.

ART

Adaptive Runtime Technology. A modular, distributed object architecture that supports dynamic deployment and configuration of services and application code. ART provides the foundation for Orbix and Artix software products.

C

CFR

See configuration repository.

client

An application (process) that typically runs on a desktop and requests services from other applications that often run on different machines (known as server processes). In CORBA, a client is a program that requests services from CORBA objects.

configuration

A specific arrangement of system elements and settings.

configuration domain

Contains all the configuration information that Orbix ORBs, services and applications use. Defines a set of common configuration settings that specify available services and control ORB behavior. This information consists of configuration variables and their values. Configuration domain data can be implemented and maintained in a centralized Orbix configuration repository or as a set of files distributed among domain hosts. Configuration domains enable you to organize ORBs into manageable groups, bringing scalability and ease-of-use to large environments. See also configuration file and configuration repository.

configuration file

A file that contains configuration information for Orbix components within a specific configuration domain. See also configuration domain.

configuration repository

A centralized store of configuration information for all Orbix components within a specific configuration domain. See also configuration domain.

configuration scope

Orbix configuration is divided into scopes. These are typically organized into a root scope and a hierarchy of nested scopes, the fully-qualified names of which map directly to ORB names. By organizing configuration properties into various scopes, different settings can be provided for individual ORBs, or common settings for groups of ORB. Orbix services, such as the naming service, have their own configuration scopes.

CORBA

Common Object Request Broker Architecture. An open standard that enables objects to communicate with one another regardless of what programming language they are written in, or what operating system they run on. The CORBA specification is produced and maintained by the OMG. See also OMG.

CORBA naming service

An implementation of the OMG Naming Service Specification. Describes how applications can map object references to names. Servers can register object references by name with a naming service repository, and can advertise those names to clients. Clients, in turn, can resolve the desired objects in the naming service by supplying the appropriate name. The Orbix naming service is an example.

CORBA objects

Self-contained software entities that consist of both data and the procedures to manipulate that data. Can be implemented in any programming language that CORBA supports, such as C++ and Java.

CORBA transaction service

An implementation of the OMG Transaction Service Specification. Provides interfaces to manage the demarcation of transactions and the propagation of transaction contexts. Orbix OTS is such as service.

correlation ID

See Actional server manifest.

D

deployment

The process of distributing a configuration or system element into an environment.

G

GIOP

General Inter-ORB Protocol. The general CORBA standard messaging protocol, defined by the OMG, for communications between ORBs and distributed applications. The implementation of GIOP for TCP/IP is IIOP. See IIOP.

H

HTTP

HyperText Transfer Protocol. The underlying protocol used by the World Wide Web. It defines how files (text, graphic images, video, and other multimedia files) are formatted and transmitted. Also defines what actions Web servers and browsers should take in response to various commands. HTTP runs on top of TCP/IP.

I

IDL

Interface Definition Language. The CORBA standard declarative language that allows a programmer to define interfaces to CORBA objects. An IDL file defines the public API that CORBA objects expose in a server application. Clients use these interfaces to access server objects across a network. IDL interfaces are independent of operating systems and programming languages.

IFR

See interface repository.

IIOP

Internet Inter-ORB Protocol. The CORBA standard messaging protocol, defined by the OMG, for communications between ORBs and distributed applications. IIOP is defined as a protocol layer above the transport layer, TCP/IP.

implementation repository

A database of available servers, it dynamically maps persistent objects to their server's actual address. Keeps track of the servers available in a system and the hosts they run on. Also provides a central forwarding point for client requests. See also location domain and locator daemon.

IMR

See implementation repository.

instrumentation

Code instructions that monitor specific components in a system (for example, instructions that output logging information on screen). When an application contains instrumentation code, it can be managed using a management tool such as Actional.

installation

The placement of software on a computer. Installation does not include configuration unless a default configuration is supplied.

Interface Definition Language

See IDL.

interceptor

An Actional interceptor collects data about service traffic from an application server, and applies policies to that traffic. It sits in the flow between the application logic and the consumers and providers of other services. It intercepts all inbound and outbound calls, and feeds information about those calls to an Actional agent.

An Orbix interceptor is an object that ORB services and transports implement to process operation invocations. Orbix interceptors are arranged in a chain, with each interceptor caching a reference to the next interceptor in the chain.

interface repository

Provides centralized persistent storage of IDL interfaces. An Orbix client can query this repository at runtime to determine information about an object's interface, and then use the Dynamic Invocation Interface (DII) to make calls to the object. Enables Orbix clients to call operations on IDL interfaces that are unknown at compile time.

invocation

A request issued on an already active software component.

IOR

Interoperable Object Reference. See object reference.

L

LG_Header

See Actional server manifest.

location domain

A collection of servers under the control of a single locator daemon. Can span any number of hosts across a network, and can be dynamically extended with new hosts. See also locator daemon and node daemon.

locator daemon

A server host facility that manages an implementation repository and acts as a control center for a location domain. Orbix clients use the locator daemon, often in conjunction with a naming service, to locate the objects they seek. Together with the implementation repository, it also stores server process data for activating servers and objects. When a client invokes on an object, the client ORB sends this invocation to the locator daemon, and the locator daemon searches the implementation repository for the address of the server object. In addition, enables servers to be moved from one host to another without disrupting client request processing. Redirects requests to the new location and transparently reconnects clients to the new server instance. See also location domain, node daemon, and implementation repository.

N

naming service

See CORBA naming service.

node

An Actional node is defined as a system on the current network. A node with an Actional agent installed is referred to as an instrumented node or a managed node.

node daemon

An Orbix node daemon starts, monitors, and manages Orbix servers on a host machine. Every machine that runs an Orbix server must run a node daemon.

O

object reference

Uniquely identifies a local or remote object instance. Can be stored in a CORBA naming service, in a file or in a URL. The contact details that a client application uses to communicate with a CORBA object. Also known as interoperable object reference (IOR) or proxy.

OMG

Object Management Group. An open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications, including CORBA. See www.omg.org.

ORB

Object Request Broker. Manages the interaction between clients and servers, using the Internet Inter-ORB Protocol (IIOP). Enables clients to make requests and receive replies from servers in a distributed computer environment. Key component in CORBA.

OTS

See CORBA transaction service.

P

POA

Portable Object Adapter. Maps object references to their concrete implementations in a server. Creates and manages object references to all objects used by an application, manages object state, and provides the infrastructure to support persistent objects and the portability of object implementations between different ORB products. Can be transient or persistent.

protocol

Format for the layout of messages sent over a network.

S

server

An application that provides services to clients. CORBA servers act as containers for CORBA objects, allowing clients to access those objects using IDL interfaces.

service context

A GIOP service context is a general mechanism for including out-of-band data in a GIOP request or reply message. Service contexts in GIOP are analogous to headers in other protocols such as HTTP.

SSL

Secure Sockets Layer protocol. Provides transport layer security—authenticity, integrity, and confidentiality—for authenticated and encrypted communications between clients and servers. Runs above TCP/IP and below application protocols such as HTTP and IIOP.

T

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic suite of protocols used to connect hosts to the Internet, intranets, and extranets.

TLS

Transport Layer Security. An IETF open standard that is based on, and is the successor to, SSL. Provides transport-layer security for secure communications. See also SSL.

Notices

Corporate information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Contacting Technical Support

The Rocket Community is the primary method of obtaining support. If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Community and report a problem, download an update, or read answers to FAQs. To log in to the Rocket Community or to request a Rocket Community account, go to www.rocketsoftware.com/support. In addition to using the Rocket Community to obtain support, you can use one of the telephone numbers that are listed above or send an email to support@rocketsoftware.com.

Rocket Global Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA

Country and Toll-free telephone number

To contact Rocket Software by telephone for any reason, including obtaining pre-sales information and technical support, use one of the following telephone numbers.

- United States: 1-855-577-4323
- Australia: 1-800-823-405
- Belgium: 0800-266-65
- Canada: 1-855-577-4323
- China: 400-120-9242
- France: 08-05-08-05-62
- Germany: 0800-180-0882
- Italy: 800-878-295
- Japan: 0800-170-5464
- Netherlands: 0-800-022-2961
- New Zealand: 0800-003210
- South Africa: 0-800-980-818
- United Kingdom: 0800-520-0439