



Stonebranch Solutions

Version 4.2.0

Universal Event Monitor for SOA Reference Guide

uemsoa-ref-4200

Universal Event Monitor for SOA

Reference Guide

Stonebranch Solutions 4.2.0

Document Name	Universal Event Monitor for SOA 4.2.0 Reference Guide				
Document ID	uemsoa-ref-4200				
Components	z/OS	UNIX	Windows	IBM i	HP NonStop
Universal Event Monitor for SOA		√	√		

Stonebranch Documentation Policy

This document contains proprietary information that is protected by copyright. All rights reserved. No part of this publication may be reproduced, transmitted or translated in any form or language or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission, in writing, from the publisher. Requests for permission to make copies of any part of this publication should be mailed to:

Stonebranch, Inc.
950 North Point Parkway, Suite 200
Alpharetta, GA 30005 USA
Tel: (678) 366-7887
Fax: (678) 366-7717

Stonebranch, Inc.[®] makes no warranty, express or implied, of any kind whatsoever, including any warranty of merchantability or fitness for a particular purpose or use.

The information in this documentation is subject to change without notice.

Stonebranch shall not be liable for any errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this document.

All products mentioned herein are or may be trademarks of their respective owners.

© 2008-2010 by Stonebranch, Inc.

All rights reserved.



Summary of Changes

Changes for Universal Event Monitor for SOA 4.2.0 Reference Guide
(uemsoa-ref-4200)
August 6, 2010

- Renamed the Universal Event Monitor for SOA 3.2.0 User Guide to the Universal Event Monitor for SOA 4.2.0 Reference Guide. Otherwise, no changes for required for this version of the document.

Changes for Universal Event Monitor for SOA 3.2.0 User Guide
(uemsoa-user-3203)
October 30, 2009

Universal Event Monitor for SOA 3.2.0.4

- Added support of MQ protocol:
 - Added Section [2.3 MQ Client Jar Files for Inbound](#).
 - Added Section [2.6 MQ Inbound Connection](#).
 - Added Section [3.4 MQ Connector – Inbound](#).
 - Added MQ Connection information in [Figure A.1 UAC.xml File - Main Elements](#).
 - Added MQ Connection information in [Figure A.2 UAC.xml File - Sample](#).

Changes for Universal Event Monitor for SOA 3.2.0 User Guide
(uemsoa-user-3202)
December 17, 2008

This is the first version of the Universal Event Monitor for SOA 3.2.0 User Guide.

Contents

Summary of Changes	5
Contents	6
List of Figures	9
List of Tables	10
Preface	11
Document Structure	11
Cross-Reference Links	11
Conventions	12
Vendor References	13
Document Organization	14
1 Universal Event Monitor for SOA Overview	15
1.1 Overview	15
1.1.1 Stonebranch Solutions for SOA	15
1.1.2 Universal Event Monitor for SOA Workflow	16
1.2 Supported Protocols	17
1.2.1 SOAP	17
1.2.2 JMS	17
1.3 Message Exchange Pattern	18
Listen MEP	18
1.4 Connectors	19

SOAP Connector	19
JMS Connector	19
1.5 Defined Ports	20
2 Configuration	21
2.1 Overview	21
2.1.1 Inbound Transactions	21
2.1.2 Required Order of Inbound Transactions	22
2.2 JMS Provider Client Jar Files for Inbound	23
2.2.1 Location of Client Jar Files	23
2.2.2 Queue or Topic Infrastructure	23
2.3 MQ Client Jar Files for Inbound	24
2.4 JMS Inbound Connection	25
2.4.1 JMS Connection Sample	26
2.5 SOAP Inbound Connection	27
2.5.1 SOAP Connection Sample	28
2.6 MQ Inbound Connection	29
2.6.1 MQ Connection Sample	30
3 Operation	31
3.1 Overview	31
3.2 SOAP Connector – Inbound	32
3.3 JMS Connector – Inbound	34
3.4 MQ Connector – Inbound	36
4 Troubleshooting	38
4.1 Overview	38
4.2 Logging Configuration	40
4.2.1 Server Component Logging Configuration	40
Server Component log4jConfiguration.xml Example	42
A UAC.xml File Description	43
A.1 UAC.xml File - Main Elements	44
A.2 UAC.xml File - Sample	47
B Web Services Description Language (WSDL) File	49

B.1 Overview	49
B.2 WSDL File	50
C Glossary	52
D Customer Support	56
Index	57

List of Figures

1	Universal Event Monitor for SOA Overview	15
	Figure 1.1 Universal Event Monitor for SOA Workflow	16
	Figure 1.2 Listen MEP - Logical View	18
2	Configuration	21
	Figure 2.1 JMS Connection Sample	26
	Figure 2.2 SOAP Connection Sample	28
	Figure 2.3 MQ Connection Sample	30
3	Operation	31
	Figure 3.1 SOAP Connector – Listen Inbound Operation	33
	Figure 3.2 JMS Connector – Inbound Listen Operation	35
	Figure 3.3 MQ Connector – Inbound Listen Operation	37
4	Troubleshooting	38
	Figure 4.1 Server Component log4jConfiguration.xml Example	42
A	UAC.xml File Description	43
	Figure A.1 UAC.xml File - Main Elements	46
	Figure A.2 UAC.xml File - Sample	48
B	Web Services Description Language (WSDL) File	49
	Figure B.1 WSDL file (page 1)	50
	Figure B.2 WSDL file (page 2)	51
	Figure B.3 WSDL file (page 3)	51

List of Tables

1	Universal Event Monitor for SOA Overview	15
	Table 1.1 Universal Event Monitor for SOA - Defined Ports	20
4	Troubleshooting	38
	Table 4.1 Product Directories and Files (/var/opt/universal) – Universal Event Monitor for SOA for AIX / Linux	39

Preface

Document Structure

This document is written using specific conventions for text formatting and according to a specific document structure in order to make it as useful as possible for the largest audience. The following sections describe the document formatting conventions and organization.

Cross-Reference Links

This document contains cross-reference links to and from other Stonebranch Solutions documentation.

In order for the links to work correctly:

- Place the documents in the same folder.
- In Adobe Reader / Adobe Acrobat, de-select **Open cross-document link in same window** in the **General** category of your **Preferences** dialog (selected from the **Edit** menu).

Conventions

Specific text formatting conventions are used within this document to represent different information. The following conventions are used.

Typeface and Fonts

This document provides tables that identify how information is used. These tables identify values and/or rules that are either pre-defined or user-defined:

- *Italics* denotes user-supplied information.
- **Boldface** indicates pre-defined information.

Elsewhere in this document, **This Font** identifies specific names of different types of information, such as file names or directories (for example, `\abc\123\he1p.txt`).

Operating System-Specific Text

Most of this document describes the product in the context of all supported operating systems. At times, it is necessary to refer to operating system-specific information. This information is introduced with a special header, which is followed by the operating system-specific text in a different font size from the normal text.

z/OS

This text pertains specifically to the z/OS line of operating systems.

This text resumes the information pertaining to all operating systems.

Tips from the Stoneman



Stoneman's Tip

Look to the Stoneman for suggestions
or for any other information
that requires special attention.

Vendor References

References are made throughout this document to a variety of vendor operating systems. We attempt to use the most current product names when referencing vendor software.

The following names are used within this document:

- **z/OS** is synonymous with IBM z/OS and IBM OS/390 line of operating systems.
- **Windows** is synonymous with Microsoft's Windows 2000 / 2003 / 2008, Windows XP, Windows Vista, and Windows 7 lines of operating systems. Any differences between the different systems will be noted.
- **UNIX** is synonymous with operating systems based on AT&T and BSD origins and the Linux operating system.
- **IBM i** is synonymous with IBM i/5, IBM OS/400, and OS/400 operating systems.
- **IBM System i** is synonymous with IBM i Power Systems, IBM iSeries, IBM AS/400, and AS/400 systems.

Note: These names do not imply software support in any manner. For a detailed list of supported operating systems, see the Stonebranch Solutions 4.2.0 Installation Guide.

Document Organization

The document is organized into the following chapters:

- [Universal Event Monitor for SOA Overview](#) (Chapter 1)
Overview of Universal Event Monitor for SOA.
- [Configuration](#) (Chapter 2)
Information on configuring Universal Event Monitor for SOA.
- [Operation](#) (Chapter 3)
Detailed information about Universal Event Monitor for SOA operations.
- [Troubleshooting](#) (Chapter 4)
Solutions to common problems that could be encountered using Universal Event Monitor for SOA.
- [UAC.xml File Description](#) (Appendix A)
Description of the `UAC.xml` file, which specifies listening attributes of UAC for inbound SOAP and JMS operations.
- [Web Services Description Language \(WSDL\) File](#) (Appendix B)
Example of a WSDL file, which specifies SOAP services and methods made available by a server.
- [Glossary](#) (Appendix C)
Glossary of terms specific to Internet and message-based concepts.
- [Customer Support](#) (Appendix D)
Customer support contact information for Universal Event Monitor for SOA.

Universal Event Monitor for SOA Overview

1.1 Overview

Universal Event Monitor for SOA – together with Universal Event Monitor – comprise Stonebranch Inc.'s Universal Event Monitor (UEM) family of products.

Universal Event Monitor (UEM) monitors one or more local or remote system events. It also can execute a system command or script based on the outcome of the events that it is monitoring.

Universal Event Monitor for SOA – the SOA "Listener" – lets you create file-based events from inbound Internet and message-based messages, and write the events to file. As such it integrates Internet and message-based applications with systems management functions such as:

- Alerting and notification
- Incident and problem management
- Job scheduling
- Data movement

1.1.1 Stonebranch Solutions for SOA

Universal Event Monitor for SOA is packaged with Universal Command Agent for SOA and is distributed as part of Stonebranch Solutions for SOA.

- Universal Command Agent for SOA comprises the workload execution functionality.
- Universal Event Monitor for SOA comprises the file-based event monitoring functionality.

1.1.2 Universal Event Monitor for SOA Workflow

Figure 1.1, below, illustrates the workflow of Universal Event Monitor for SOA.

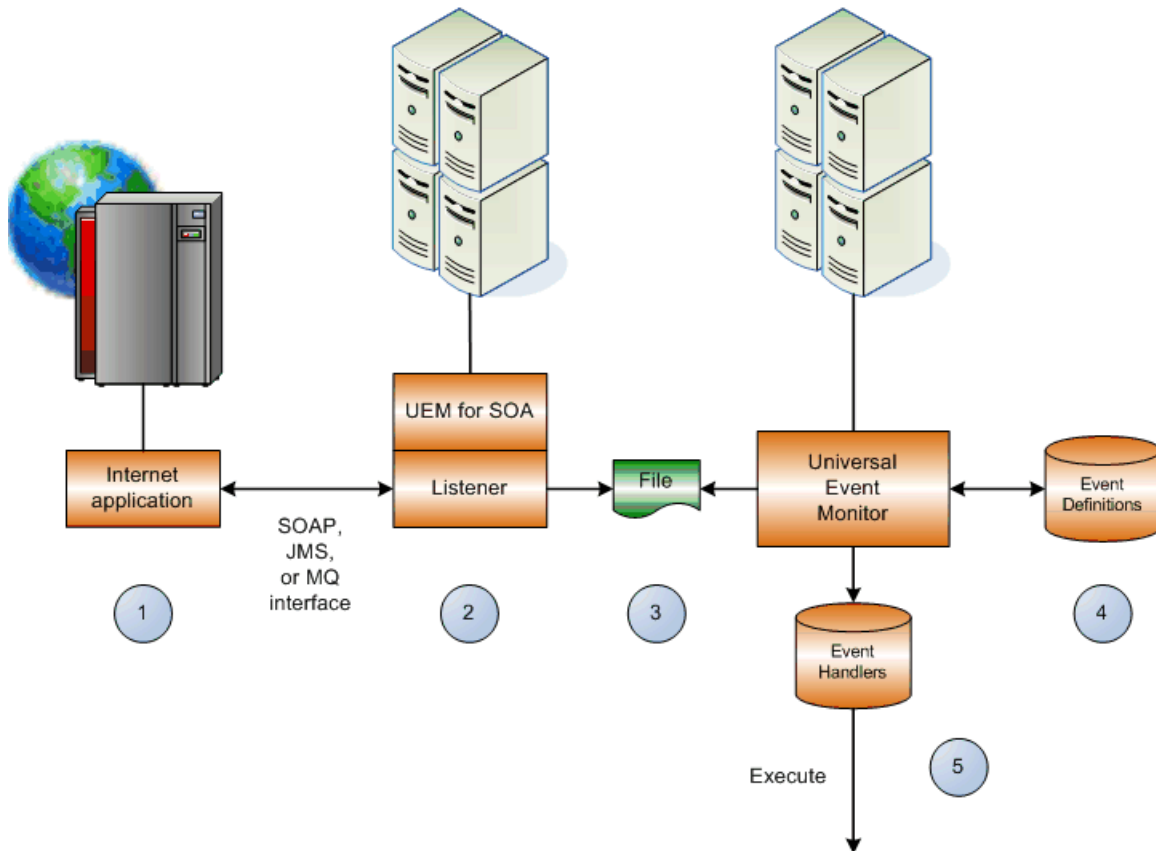


Figure 1.1 Universal Event Monitor for SOA Workflow

1. Internet or message-based application sends a message to Universal Event Monitor for SOA via a SOAP or JMS interface.
2. SOAP Listener processes the incoming message.
3. Universal Event Monitor for SOA writes the message to a file (payload or header / payload).
4. UEM reads the file and matches the message against an event definition.
5. UEM executes an event handler based on the message match up to the event definitions. UEM can:
 - Execute when it reads a file.
 - Execute depending on what's in the file.
 - Pass along the file data.
 The handler can execute either:
 - Script
 - Universal Command
 - Universal Data Mover

1.2 Supported Protocols

Universal Event Monitor for SOA supports synchronous and asynchronous communication for file-based event detection via two protocols: SOAP and JMS.

Synchronous communication requires that the calling party wait for a response from the target application before beginning the next task.

Asynchronous communication allows the calling party to move to the next task without waiting for a response (if there is one) from the target application.

1.2.1 SOAP

SOAP (Simple Object Access Protocol) is a synchronous protocol for exchanging XML-based messages over computer networks. It is the foundation layer of the web services stack.

For Universal Event Monitor for SOA, the SOAP protocol is used to allow an inbound SOAP message to be sent by any application that supports the SOAP protocol. This inbound message can represent an event and can contain data relevant to the event or the target application.

1.2.2 JMS

JMS (Java Message Service) defines the standard for reliable Enterprise Messaging and uses both point-to-point (queue-based) and publish / subscribe (topic-based) messaging patterns.

For Universal Event Monitor for SOA, the JMS protocol is used to allow an inbound JMS message to be sent by any application that supports the JMS protocol. This inbound message can represent an event and can contain data relevant to the event or the target application. Unlike a SOAP message, the JMS message is not required to contain data.

1.3 Message Exchange Pattern

A Message Exchange Pattern (MEP) describes the pattern of messages required by a communications protocol to establish or use a communication channel.

There are two major types of MEPs:

- Request-response pattern (synchronous)
- One-way pattern (asynchronous)

Universal Event Monitor for SOA supports the one-way (asynchronous) MEP, more commonly referred to as the Listen MEP.

Listen MEP

The Listen MEP represents an asynchronous inbound event initiated by an external application. This operation is a publish only from the external application, meaning that Universal Event Monitor for SOA will not reply to the inbound request. External applications can make use of both the JMS and SOAP protocols for inbound operations.

Currently, Universal Event Monitor for SOA will write the inbound message to a file for processing by Universal Event Monitor.

Figure 1.2, below, illustrates a logical view of the Listen MEP.

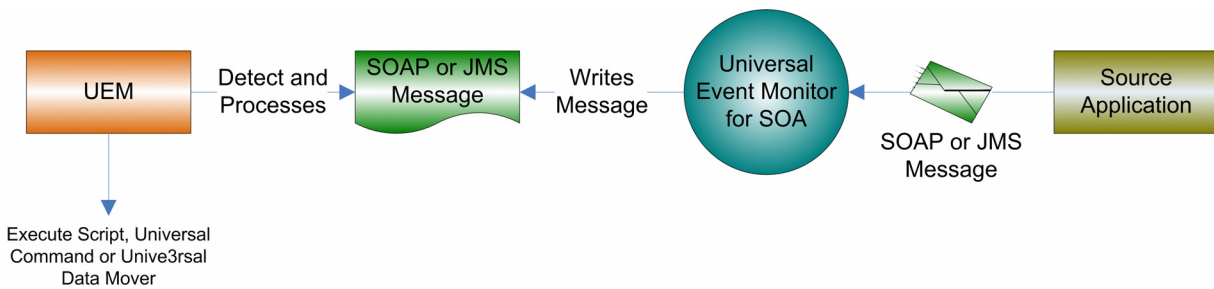


Figure 1.2 Listen MEP - Logical View

1.4 Connectors

The work of transforming the command line and STDIN input to the appropriate protocol message is the responsibility of the connectors that are deployed in the Universal Event Monitor for SOA environment.

A summary of the current connectors follows.

SOAP Connector

The SOAP connector supports inbound operations via the SOAP protocol.

It is a synchronous listener component that supports the following features:

- Supports the SOAP 1.1 specification
- Supports the Listen MEP

JMS Connector

The JMS connector supports inbound operations via the JMS protocol using asynchronous communication.

It supports the following features:

- Supports the JMS 1.1 specification.
- Supports the Listen MEP.
- Supports Queue- and Topic-based operations.

Note: The Universal Event Monitor for SOA platform allows for the addition of connectors to support future business requirements.

1.5 Defined Ports

Universal Event Monitor for SOA uses a specific set of ports (see [Table 1.1](#), below).

Port Number	Description
7880	HTTP port for inbound SOAP operations.
7843	HTTPS port for secure inbound SOAP operations.

Table 1.1 Universal Event Monitor for SOA - Defined Ports

Configuration

2.1 Overview

Depending on which message exchange pattern (MEP) you are using – and, in the case of JMS, what JMS Provider you are using – there are several operations that may need to be configured before you can use Universal Event Monitor for SOA.

2.1.1 Inbound Transactions

To transact an inbound operation, you first must configure Universal Event Monitor for SOA for inbound transactions via the `UAC.xml` configuration file. This file specifies the listening attributes of UAC for SOAP, JMS, and MQ inbound operations, such as connection and destination information, as well as information regarding the name and location of the output file.

AIX and Linux

The `UAC.xml` file is located in `/etc/universal/UAC.xml`.

Windows

The `UAC.xml` file is located in `%ALLUSERSPROFILE%\Application Data\Universal\conf\UAC.xml`.

Insert new text between the line beginning with `<sb:UAC` and the last line, which must terminate with `</sb:UAC>`. An error will occur if `</sb:UAC>` appears in the file anywhere other than the last line.

The XML is just text; no special tools are required to create or modify an XML document.

Also, if you are using JMS or MQ for inbound operations, you must have a JMS or MQ Provider with a queue or topic configured on which UAC can listen.

2.1.2 Required Order of Inbound Transactions

The `UAC.xml1` file can contain multiple definitions for inbound transactions for multiple providers. When multiple different types of inbound transactions are defined, it is important to note that there is a required order in which these definitions occur.

The required order when defining multiple types of providers is:

1. JMS providers
2. SOAP transactions
3. MQ Providers

It is entirely acceptable to define a subset of these providers (including just a single provider). However, the order is still imperative.

For example:

1. JMS providers
2. MQ providers

or

1. SOAP transactions
2. MQ providers

The following scenario(s) are not allowed:

1. MQ providers
2. JMS providers
3. SOAP transactions

or

1. SOAP Transactions
2. JMS providers

2.2 JMS Provider Client Jar Files for Inbound

As each JMS provider implementation is vendor-specific, you must acquire the client jar files that allow third-party applications to connect and communicate with your JMS provider.

For example, if you are using the JMS functions in IBM's WebSphere Application Server on Linux or Windows, you need the following files:

- **sibc.jms.jar**
- **sibc.jndi.jar**
- **sibc.orb.jar**

2.2.1 Location of Client Jar Files

You must place the JMS provider client jar files in the following location:

Linux

`/opt/universal/uac/container/webapps/axis2/WEB-INF/lib`

Windows

`\Program Files\Universal\uac\container\webapps\axis2\WEB-INF\lib.`

The names of the jar files differ depending on which JMS provider you are using.

2.2.2 Queue or Topic Infrastructure

The Universal Event Monitor for SOA JMS or MQ Series Connector does not provide the queue or topic infrastructure.

You must have a JMS provider or MQ Series Broker with queues or topics configured to use the JMS or MQ Series inbound operations.

2.3 MQ Client Jar Files for Inbound

As is the case for outbound, you must have the IBM MQ client jar files for inbound or request / reply operations.

You would place the MQ client jar files in the following location:

Linux

```
/opt/universal/uac/container/webapps/axis2/WEB-INF/lib
```

Windows

```
\Program Files\Universal\uac\container\webapps\axis2\WEB-INF\lib
```

You will need the following jar files:

- `com.ibm.mq.commonservices.jar`
- `com.ibm.mq.jar`
- `com.ibm.mq.pcf.jar`
- `com.ibm.mq.headers.jar`
- `com.ibm.mq.jmqi.jar`
- `connector.jar`

The Universal Command Agent for SOA: MQ Connector does not provide the queue or topic infrastructure. You must have a WebSphere MQ Message Broker with queues configured to use the MQ outbound or request / reply operations.

2.4 JMS Inbound Connection

You can have multiple JMS Connections defined in the **UAC.xml** file. The only requirement is that you have the corresponding queues or topics configured in your JMS Provider.

To set up an inbound JMS Listener, you must:

1. Supply values for the following items in the **UAC.xml** file:
 - **Name**
Name of the connection which is user-defined. If you are going to define more than one JMS listener, the name for each JMS Connection must be unique.
 - **Initial Context Properties**
Name/value pairs identified as properties that specify JMS Provider-specific information. Two items that must be specified are the Initial Context Factory and the Provider URL.
 - **Connection Factory**
Name of the JMS Provider Connection Factory class to use.
2. Specify the attributes for the JMS Listener, which include Destination and Actions. Currently, the only Action that is supported is writing the inbound transaction to a file.
 - **Destination**
Queue or topic on which to listen for inbound messages.
 - **Directory**
Output directory where UAC will write the file. You must specify an absolute path to a directory associated with the relevant business process. This path should be valid and reachable either locally or via a network. The only requirement is that the directory must exist prior to commencing inbound operations.
 - **File Name Pattern**
Name of the file and the sequence pattern (if there will be more than one file written to the same directory). The format is `<filenamehere%Seq%.txt>`, where `filenamehere` is the name of the file you have chosen.
File name restrictions are enforced by the file system to which the file is being written. For example, using double quotation marks (") around a file name in Windows will result in an error.
 - **Start Sequence Number**
Starting number to use for the pattern sequence (`%Seq%`).
 - **Write Properties**
Specification for whether or not the JMS properties should be written to the file. The default value is **false**. However, if it is useful to see what was specified in the properties of the JMS message, set the value to **true**.

2.4.1 JMS Connection Sample

Figure 2.1, below, illustrates a sample JMS Connection section in the UAC.xml file. The values in dark red are the values discussed in Section 2.4 JMS Inbound Connection.

(This sample illustrates a configuration using IBM's WebSphere Application Server as the JMS Provider.)

See Appendix A UAC.xml File Description for details on the elements and structure of the UAC.xml file.

```

<sb:JMSConnection>
  <sb:Name>websphere</sb:Name>
  <sb:InitialContextProperties>
    <sb:Property>
      <sb:Name>java.naming.factory.initial</sb:Name>
      <sb:Value>com.ibm.websphere.naming.wsnInitialContextFactory</sb:Value>
    </sb:Property>
    <sb:Property>
      <sb:Name>java.naming.provider.url</sb:Name>
      <sb:Value>iiop://myServer:2809</sb:Value>
    </sb:Property>
    <sb:Property>
      <sb:Name>com.ibm.CORBA.ORBInit</sb:Name>
      <sb:Value>com.ibm.ws.sib.client.ORB</sb:Value>
    </sb:Property>
  </sb:InitialContextProperties>
  <sb:ConnectionFactory>jms/ConnectionFactory</sb:ConnectionFactory>
  <sb:Listeners>
    <sb:JMSListener>
      <sb:Destination>jms/IntegrationTestQueue1</sb:Destination>
      <sb:Actions>
        <sb:JMSFileWriter>
          <sb:Directory>/home/userdirectory/outputDirectory</sb:Directory>
          <sb:FilenamePattern>WASjmsMessageQueue%Seq%.txt</sb:FilenamePattern>
          <sb:StartSequenceNumber>1</sb:StartSequenceNumber>
          <sb:WriteProperties>false</sb:WriteProperties>
        </sb:JMSFileWriter>
      </sb:Actions>
    </sb:JMSListener>
  </sb:Listeners>
</sb:JMSConnection>

```

Figure 2.1 JMS Connection Sample

2.5 SOAP Inbound Connection

You can have only one SOAP Connection defined in the **UAC.xml** file.

To set up an inbound SOAP Listener, you must:

1. Supply values for the following items in the **UAC.xml** file:
 - **Operation**
Name of the SOAP operation from the inbound source application's web service call.
Note: If the operation of the calling web service does not match the operation specified here, the inbound web service call will fail.
 - **URI**
Path to the inbound service. This value should be set to:
`/axis2/services/UACInbound`.
2. Specify the attributes for the SOAP Listener, which include Destination and Actions. Currently, the only Action that is supported is writing the inbound transaction to a file.
 - **Directory**
Output directory where UAC will write the file. If only a name is given, it is considered relative to the UAC directory. A fully qualified path can be given, as long as it is reachable locally or via a network. The only requirement is that the directory must exist prior to commencing inbound operations.
 - **File Name Pattern**
Name of the file and the sequence pattern, if there will be more than one file written to the same directory. The format is `<filenamehere%Seq%.txt>`, where `filenamehere` is the name of the file you have chosen.
File name restrictions are enforced by the file system to which the file is being written. For example, using double quotation marks (") around a file name in Windows will result in an error.
 - **Start Sequence Number**
Starting number to use for the pattern sequence (`%Seq%`).
 - **Write Properties**
The default value is **false**. However, if it is useful to see what was specified in the properties of the SOAP message, especially if more than one node is involved, set the value to **true**.

2.5.1 SOAP Connection Sample

Figure 2.2, below, illustrates a sample SOAP Connection section. The values in dark red are the values discussed in Section 2.5 SOAP Inbound Connection.

See Appendix A UAC.xml File Description for details on the elements and structure of the UAC.xml file.

```
<sb:SOAPConnection>
  <sb:URI>/axis2/services/UACInbound</sb:URI>
  <sb:Listeners>
    <sb:SOAPListener>
      <sb:Operation>Operation</sb:Operation>
      <sb:Actions>
        <sb:SOAPFilewriter>
          <sb:Directory>outputDirectory</sb:Directory>
          <sb:FilenamePattern>soapMessage%Seq%.xml</sb:FilenamePattern>
          <sb:StartSequenceNumber>1</sb:StartSequenceNumber>
          <sb:WriteEnvelope>false</sb:WriteEnvelope>
        </sb:SOAPFilewriter>
      </sb:Actions>
    </sb:SOAPListener>
  </sb:Listeners>
</sb:SOAPConnection>
```

Figure 2.2 SOAP Connection Sample

2.6 MQ Inbound Connection

You can have multiple MQ Connections defined in the **UAC.xml** file. The only requirement is that you have the corresponding queues configured in your MQ Broker.

To set up an inbound MQ Listener, you must:

1. Supply values for the following items in the **UAC.xml** file:
 - **Name**
Name of the connection which is user-defined. If you are going to define more than one MQ listener, the name for each MQ Connection must be unique.
 - **Host**
Name or address of the server hosting the MQ Broker
 - **QueueManagerName**
Name of the MQ QueueManager.
 - **Channel**
Name of the MQ Channel.
 - **Port**
Port number where the MQ Series Broker is listening (normally 1414)
2. Specify the attributes for the MQ Listener, which include QueueName and Actions. Currently, the only Action that is supported is writing the inbound transaction to a file.
 - **QueueName**
Queue on which to listen for inbound messages.
 - **Directory**
Output directory where UAC will write the file.
You must specify an absolute path to a directory associated with the relevant business process. This path should be valid and reachable either locally or via a network. The only requirement is that the directory must exist prior to commencing inbound operations.
 - **File Name Pattern**
Name of the file and the sequence pattern (if there will be more than one file written to the same directory).
The format is `<filenamehere%Seq%.txt>`, where `filenamehere` is the name of the file you have chosen.
File name restrictions are enforced by the file system to which the file is being written. For example, using double quotation marks (") around a file name in Windows will result in an error.
 - **Start Sequence Number**
Starting number to use for the pattern sequence (`%Seq%`).
 - **Write Properties**
Specification for whether or not the MQ message properties should be written to the file.
The default value is false. However, if it is useful to see what was specified in the properties of the MQ message, set the value to true.

2.6.1 MQ Connection Sample

Figure 2.3, below, illustrates a sample MQ Connection section. The values highlighted in dark red are the values discussed in Section 2.6 MQ Inbound Connection.

See Appendix A UAC.xml File Description for details on the elements and structure of the UAC.xml file.

```
<sb:MQConnection>
  <sb:Name>MQ Series Listener</sb:Name>
  <sb:Host>Server Host or IP</sb:Host>
  <sb:QueueManagerName>MQ QueueManager</sb:QueueManagerName>
  <sb:Channel>MQ Channel</sb:Channel>
  <sb:Port>MQ Listening Port</sb:Port>
  <sb:Listeners>
    <sb:MQListener>
      <sb:QueueName>MQ Queue to Listen to</sb:QueueName>
      <sb:Actions>
        <sb:MQFilewriter>
          <sb:Directory>Directory location to write file
            to</sb:Directory>
          <sb:FilenamePattern>Filename pattern to write with
            %Seq% variable</sb:FilenamePattern>
          <sb:StartSequenceNumber>0</sb:StartSequenceNumber>
          <sb:WriteProperties>false</sb:WriteProperties>
        </sb:MQFilewriter>
      </sb:Actions>
    </sb:MQListener>
  </sb:Listeners>
</sb:MQConnection>
```

Figure 2.3 MQ Connection Sample

3.1 Overview

Universal Event Monitor for SOA allows you to execute Internet and message-based workload using a variety of transaction scenarios.

Each connector supports a set of transaction scenarios that are comprised of both standard and combination MEPs.

In this chapter, operations are grouped by connector.

They detail the supported business scenarios and the usage required for each transaction scenario.

The following list summarizes the transaction scenarios:

- SOAP Connector uses the Listen MEP.
- JMS Connector uses the Listen MEP.

3.2 SOAP Connector – Inbound

The SOAP Connector – Inbound operation is a synchronous operation that uses the Listen MEP.

It consists of an application workload making a web services call using the SOAP protocol to the Universal Event Monitor for SOA SOAP Listener on (default) port 7880.

The body of the message, which is usually associated with the application data, is written to a file where either:

- Universal Event Monitor (UEM) can use the event as a trigger.
- External application, such as a scheduler, can access the files directly.

Note: A SOAP operation that matches the SOAP operation of the remote application workload must exist in the **UAC.xml** configuration file.

See Section [2.5 SOAP Inbound Connection](#) for examples and Appendix [A UAC.xml File Description](#) for a description of the elements and structure of the **UAC.xml** file.

Figure 3.1, below, illustrates the system flow for a SOAP Connector – Listen inbound operation using the Universal Event Monitor for SOA SOAP Connector.

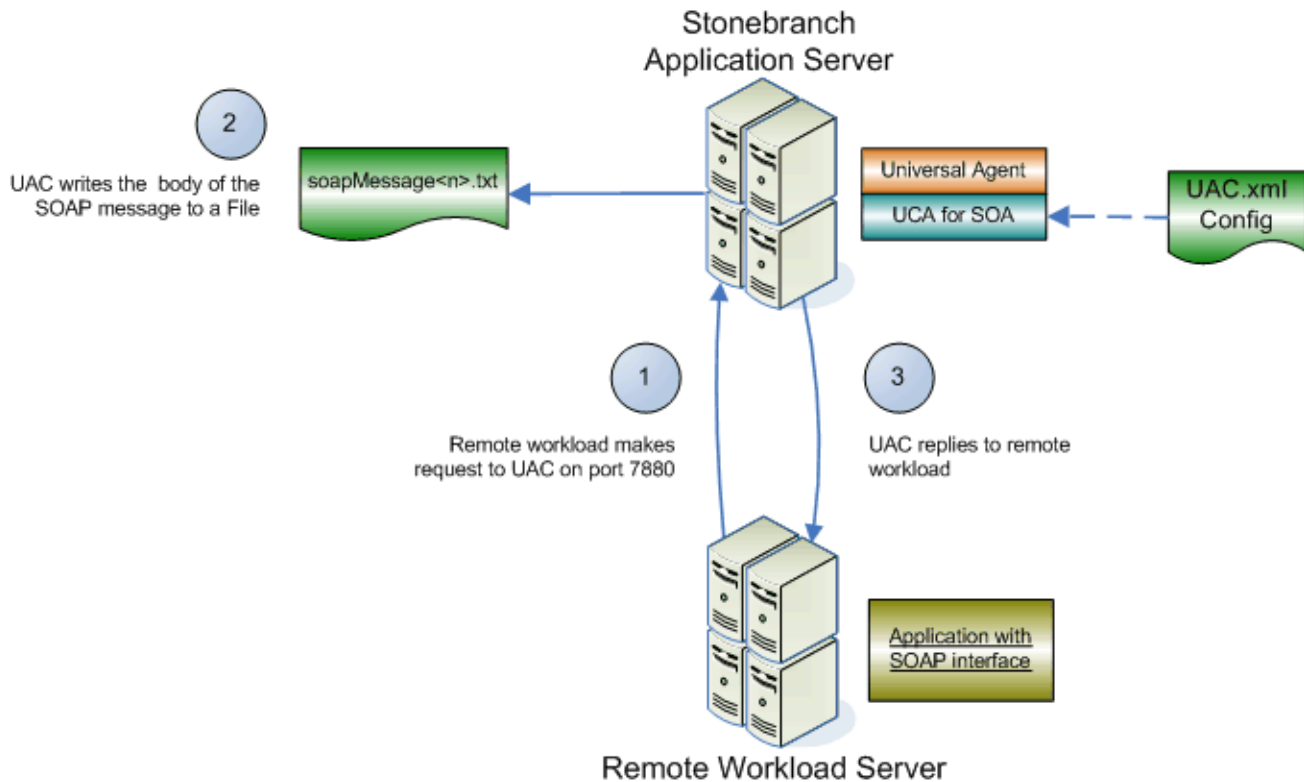


Figure 3.1 SOAP Connector – Listen Inbound Operation

The following list describes the steps (1 - 3) identified in Figure 3.1:

1. The application workload makes a SOAP request to Universal Event Monitor for SOA using the SOAP protocol on port 7880. Note that Universal Event Monitor for SOA consumes the message locally, meaning on the server that hosts Universal Event Monitor for SOA.
2. The SOAP connector consumes (that is, processes) the SOAP request and writes the content of the SOAP message to a file whose name and location are specified in the `UAC.xml` configuration file. If you set the `writeEnvelope` attribute to `true`, Universal Event Monitor for SOA will write the entire SOAP message, including the envelope, to the specified file.
3. The SOAP connector will then reply to the initial request using a message with a single node, meaning its an acknowledgement only. The purpose of the reply is to satisfy the request / reply operation and release the block of the initial request.

3.3 JMS Connector – Inbound

The JMS Connector – Inbound operation is an asynchronous operation that uses the Listen MEP.

It consists of a remote application workload publishing a JMS message using the JMS protocol to a specified destination queue hosted by the JMS provider. The JMS provider, which hosts the queues that represent the JMS destination, can be located on a remote application server or on the local application server that hosts Universal Event Monitor for SOA.

The JMS inbound connector listens on the specified destination queue for messages to arrive. When a message arrives, it reads the message off of the queue. The body of the message, which is usually associated with the application data, is written to a file where either:

- Universal Event Monitor (UEM) can use the event as a trigger.
- External application, such as a scheduler, can access the files directly.

Note: The destination queue must be hosted by the JMS provider and be associated to the application workload via the `UAC.xml` configuration file.

See Section [2.4 JMS Inbound Connection](#) for examples and Appendix [A UAC.xml File Description](#) for a description of the elements and structure of the `UAC.xml` file.

Figure 3.2, below, illustrates the system flow for a JMS Listen inbound operation using the Universal Event Monitor for SOA JMS Connector.

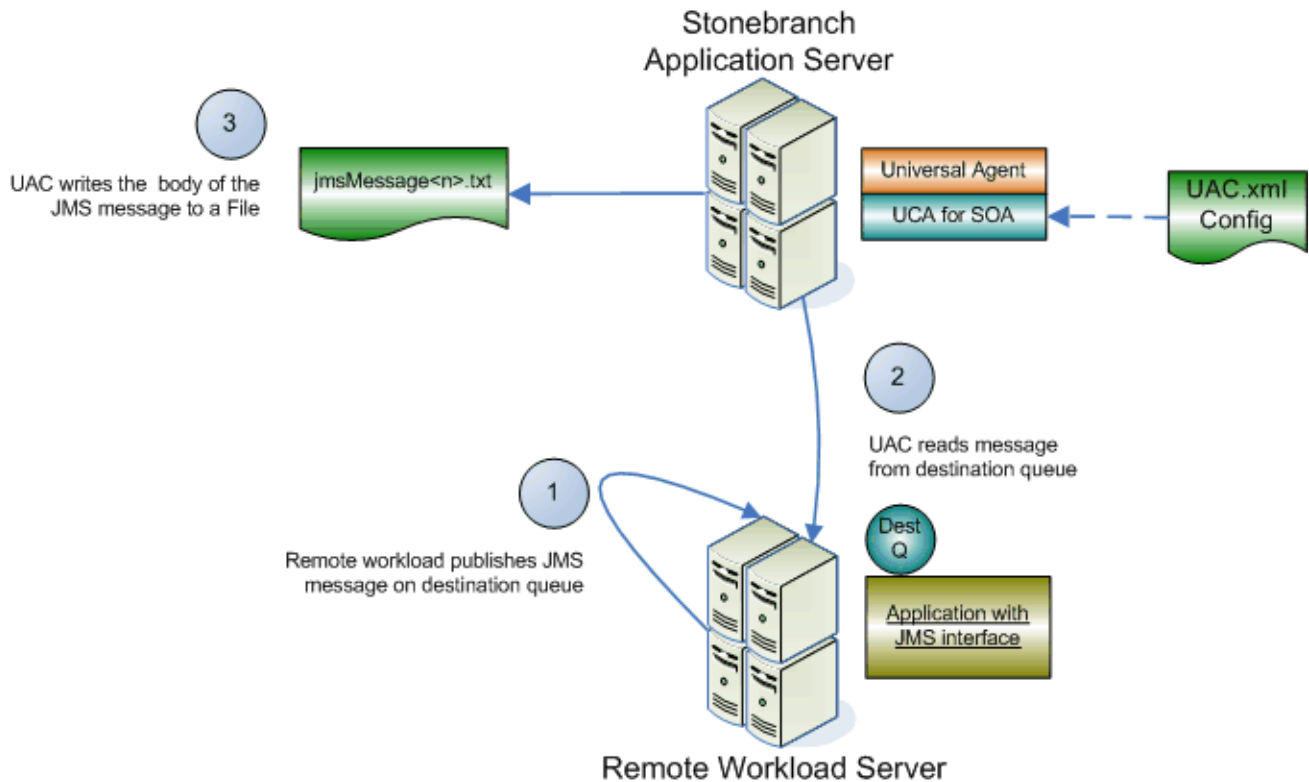


Figure 3.2 JMS Connector – Inbound Listen Operation

The following list describes the steps (1 - 3) identified in Figure 3.2.

1. The application workload publishes a JMS message to a specified destination queue hosted by a JMS provider. The JMS Inbound Connector listens on the provider queue for the incoming message.
2. The JMS Inbound Connector reads the JMS message from the inbound queue. Because the read is destructive, once the message is read from the queue, it is no longer available to other applications. The destination queue must be configured in the **UAC.xml** file in order for the JMS Inbound Connector to know what to read from.
3. The JMS Inbound Connector writes the content of the message to the file whose name and location you specify in the UAC.xml configuration file. If you set the **writeProperties** attribute to **true**, Universal Event Monitor for SOA will write both the properties and content of the JMS message to the file.

3.4 MQ Connector – Inbound

The MQ Connector – Inbound operation is an asynchronous operation that uses the Listen MEP.

It consists of a remote application workload publishing an MQ message using the MQ protocol to a specified destination queue hosted by the MQ provider. The MQ provider, which hosts the queues that represent the MQ destination, can be located on a remote application server or on the local application server that hosts Universal Event Monitor for SOA.

The MQ inbound connector listens on the specified destination queue for messages to arrive. When a message arrives, it reads the message off of the queue. The body of the message, which is usually associated with the application data, is written to a file where either:

- Universal Event Monitor (UEM) can use the event as a trigger.
- External application, such as a scheduler, can access the files directly.

Note: The destination queue must be hosted by the MQ provider and be associated to the application workload via the `UAC.xml` configuration file.

See Section [2.6 MQ Inbound Connection](#) for examples and Appendix [A UAC.xml File Description](#) for a description of the elements and structure of the `UAC.xml` file.

Figure 3.3, below, illustrates the system flow for an MQ Listen inbound operation using the Universal Event Monitor for SOA MQ Connector.

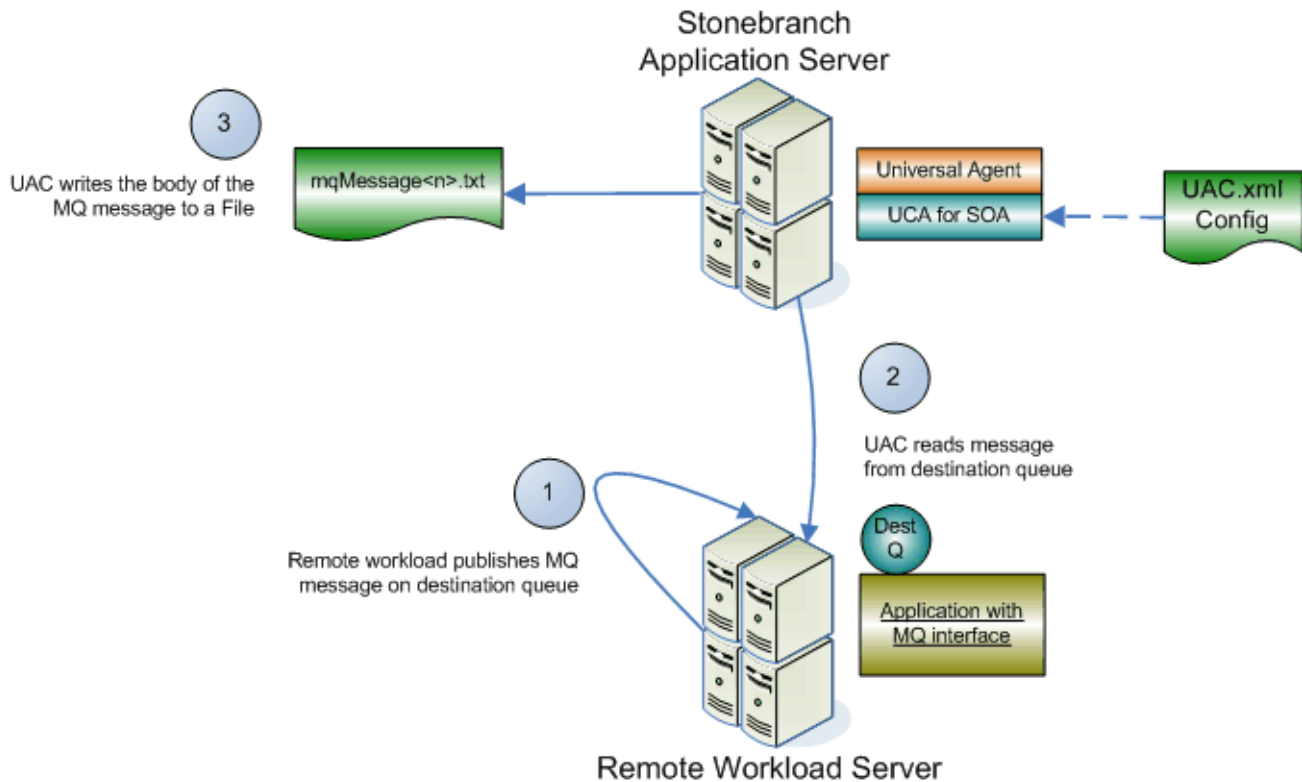


Figure 3.3 MQ Connector – Inbound Listen Operation

The following list describes the steps (1 - 3) identified in Figure 3.3.

1. The application workload publishes an MQ message to a specified destination queue hosted by an MQ provider. The MQ Inbound Connector listens on the provider queue for the incoming message.
2. The JMS Inbound Connector reads the MQ message from the inbound queue. Because the read is destructive, once the message is read from the queue, it is no longer available to other applications. The destination queue must be configured in the **UAC.xml** file in order for the MQ Inbound Connector to know what to read from.
3. The JMS Inbound Connector writes the content of the message to the file whose name and location you specify in the **UAC.xml** configuration file. If you set the **writeProperties** attribute to **true**, Universal Event Monitor for SOA will write both the properties and content of the JMS message to the file.

Troubleshooting

4.1 Overview

This chapter provides information pertaining to logging operations that may assist you in troubleshooting, should that be necessary. This information includes different types of log files, their locations, methods of directing output, and examples.

[Table 4.1](#), below, identifies the Universal Event Monitor for SOA for AIX / Linux product directories and files located under the `/var/opt/universal` parent directory.

Windows

Logging output is directed to the Windows Event Viewer.

Directory / File	Description
<code>log/uac</code>	Directory containing log and work artifacts for the UAC component. It includes the following sub-directories and files.
<code>work</code>	Subdirectory used by the web services framework for temporary operations. It requires no user interaction.
<code>catalina.out</code>	Container engine log file. This file contains messages and errors pertaining to the operation of the container and its infrastructure. On Windows, there is no <code>catalina.out</code> file.
<code>container.log</code>	Web services framework log file. This file contains messages and errors pertaining to the operation of the web services framework (axis2) and its infrastructure. On Windows, this file is located under <code>\program files\universal\uac\container\logs</code> .
<code>derby.log</code>	Database log file. This file contains messages and errors pertaining to the operation of the database and its infrastructure. On Windows, this file is located in <code>\program files\universal\uac</code> .

Directory / File	Description
uac.log	uac log file. This file contains the messages and errors pertaining to the operation of the Universal Event Monitor for SOA. This includes application messages and errors related to the business process being implemented. On AIX / Linux, the file is written to the file system. On Windows, the file is written to the Event Viewer.

Table 4.1 Product Directories and Files (/var/opt/universal) – Universal Event Monitor for SOA for AIX / Linux

4.2 Logging Configuration

At some point, you may want to check the logs for information regarding the operation of Universal Event Monitor for SOA.

Configuration of the logging operations is done via the `log4jConfiguration.xml` file for the server component.

AIX and Linux

This file is located in the `uac` product directory at `/opt/universal/uac`.

Windows

This file is located under `\program files\universal\uac`.

The logging levels supported by the logging implementation are:

- DEBUG
- INFO
- WARN
- ERROR (default)

Note: The logging level should be changed only at the request of Stonebranch, Inc. [Customer Support](#), as it can have a huge impact on performance.

4.2.1 Server Component Logging Configuration

By default, the logs are configured to write to a file on Linux and to write to the Event Viewer on Windows with the logging level set to `error`.

The following appenders, or sinks, are available to the server component.

Rolling File Appender

This is the default appender on Linux and logs to a file.

The following attributes can be specified:

- **Name** - name of the appender.
- **File** - path and name of the log file.
- **Max File Size** - maximum size of the log file before rolling over.
- **Max Backup Index** - number of times the log file can be rolled before starting over.
- **Class** - java class that implements the logger.
- **Conversion Pattern** - output format of the log text.

LF5 Appender

Logs to the LF5 program with a user interface that displays the log in row/column format and enables searches within the log file. Use this for debug only.

The following attributes can be specified:

- **Name** - name of the appender.
- **Class** - java class that implements the logger.
- **Max Number Of Records** - maximum number of records displayed.

NT Event Log Appender

This is the default appender on Windows and logs to the Windows Event Viewer.

The following attributes can be specified:

- **Name** - name of the appender.
- **Class** - java class that implements the logger.
- **Source** - source component that is outputting the log.
- **Conversion Pattern** - output format of the log text.

Console Appender

Logs to the console on STDOUT or STDERR.

The following attributes can be specified:

- **Name** - name of the appender.
- **Class** - java class that implements the logger.
- **Target** - specification to log to STDOUT or STDERR (STDERR is the default).
- **Conversion Pattern** - the output format of the log text.

Server Component log4jConfiguration.xml Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/"
debug="false" threshold="all">
  <appender name="RollingFileAppender"
class="org.apache.log4j.RollingFileAppender">
    <param name="File" value="/var/opt/universal/log/uac/uac.log"/>
    <param name="MaxFileSize" value="1000KB"/>
    <param name="MaxBackupIndex" value="4"/>
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%d %-5p %-17c{2} (%30F:%L) %3x -
%m%n"/>
    </layout>
  </appender>
  <appender name="LF5Appender" class="org.apache.log4j.lf5.LF5Appender">
    <param name="MaxNumberOfRecords" value="1000"/>
  </appender>
  <appender name="NTEventLogAppender"
class="org.apache.log4j.nt.NTEventLogAppender">
    <param name="Source" value="UAC"/>
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%c{1} %M - %m%n"/>
    </layout>
  </appender>
  <appender name="ConsoleAppender" class="org.apache.log4j.ConsoleAppender">
    <param name="Target" value="System.err"/>
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%d %-5p %-17c{2} (%30F:%L) %3x -
%m%n"/>
    </layout>
  </appender>
  <logger name="com.stonebranch" additivity="true">
    <level value="error"/>
  </logger>
  <root>
    <priority value="error"/>
    <!--<appender-ref ref="LF5Appender"/>-->
    <appender-ref ref="RollingFileAppender"/>
    <!--<appender-ref ref="ConsoleAppender"/>-->
    <!--appender-ref ref="NTEventLogAppender"/>-->
  </root>
</log4j:configuration>
```

Figure 4.1 Server Component log4jConfiguration.xml Example

UAC.xml File Description

The **UAC.xml** file specifies listening attributes of UAC for inbound SOAP, JMS, and MQ operations.

As with the outbound operations, inbound listeners require connection and destination information as well as information regarding the name and location of the output file.

This appendix provides the following **UAC.xml** file information:

- [UAC.xml File - Main Elements](#)
- [UAC.xml File - Sample](#)

A.1 UAC.xml File - Main Elements

Figure A.1, below, identifies the main elements of the UAC.xml file and describes the information specified by each element.

Element <UAC>

Root element (node) of the UAC.xml document of which there can be only one.

Element <JMSConnection>

Properties for the connection to the JMS provider and for the listeners associated with the queues or topics for the specified inbound operation. Multiple <JMSConnection> elements are allowed within the <UAC> element.

Element <Name>

Name of the JMS Connection. You could name your connection myOperation, ActiveMQ, WebSphere, or any other reasonably descriptive name.

Element <InitialContextProperties>

Any properties of the JMS connection. They are listed in name-value pairs using the <property> element. Only one <InitialContextProperties> element is allowed in a <JMSConnection> element.

Element <Property>

Any properties associated with the <JMSConnection> element that may be needed to initiate the connection such as the initial context factory and the service URL. It consists of a <Name> and <Value> element. Multiple <Property> elements are allowed within the <InitialContextProperties> element.

- Element <Name> - Name of the property.
- Element <Value> - value of the property.

Element <ConnectionFactory>

Connection factory name that is associated with the JMS Provider being accessed. Only one <ConnectionFactory> element is allowed per <JMSConnection> element.

Element <Listeners>

Listeners that will be created for associated <JMSConnection> element. The <Listeners> element includes the information for the destination and action associated with the inbound operation.

Element <Destination>

Destination in the JMS Provider, queue or topic, of the inbound operation.

Element <Actions>

Action to take when a message is received. Currently only writing the message to a file is supported via the <JMSFilewriter> element.

Element <JMSFilewriter>

Information necessary to write the body of the JMS message to a file and includes the following:

- **Element <Directory>**
Directory location of the file. This can be any available local or network based location.
- **Element <FileNamePattern>**
Name of the file and the sequence pattern if there will be more than one file written to the same directory.
- **Element <StartSequenceNumber>**
Starting number to use for the pattern sequence.
- **Element <writeProperties>**
Specification for whether or not the JMS properties should be written to the file. Default is false.

Element <SOAPConnection>

Properties for the SOAP connection that the target workload can connect to for the specified inbound operation. Multiple <SOAPConnection> elements are not allowed within the <UAC> element.

Element <URI>

Unique Resource Identifier, the name of the object or resource requested by the user. This identifier contains a "path" to a method. This element must contain /axis2/services/UACInbound.

Element <Listeners>

Listeners that will be created for associated <SOAPConnection> element. The <Listeners> element includes the information for the operation and action associated with the inbound operation.

Element <Operation>

SOAP operation associated with the inbound operation.

Element <Actions>

Action to take when a message is received. Currently only writing the message to a file is supported via the <SOAPFilewriter> element.

Element <SOAPFilewriter>

Information necessary to write the body of the JMS message to a file and includes the following:

- **Element <Directory>**
Directory location of the file. This can be any available local or network based location.
- **Element <FileNamePattern>**
Name of the file and the sequence pattern if there will be more than one file written to the same directory. File name restrictions are enforced by the file system to which the file is being written. For example, using double quotation marks (") around a file name in windows will result in an error.
- **Element <StartSequenceNumber>**
Starting number to use for the pattern sequence.
- **Element <writeEnvelope>**
Specification for whether or not the SOAP envelope should be written to the file. Default is false.

Element <MQConnection>

Properties for the connection to the MQ provider and for the MQ listeners associated with the queues for the specified inbound operation. Multiple <MQConnection> elements are allowed within the <UAC> element.

Element <Name>

Name of the connection which is user-defined. If you are going to define more than one MQ listener, the name for each MQ Connection must be unique

Element <Host>

Name or address of the server hosting the MQ Broker

Element <QueueManagerName>

Name of the MQ QueueManager.

It is the queue manager that provides the queuing services to the application program.

Element <Channel>

Name of the MQ Channel

Channels are named links between platforms across which messages are transmitted.

Element <Port>

Port number where the MQ Series Broker is listening (normally 1414)

Element <Listeners>

Listeners that will be created for associated <MQConnection> element.

The <Listeners> element includes the information for the operation and action associated with the inbound operation.

Element <QueueName>

Name of the MQ Queue to listen to.

Queues are named message repositories upon which messages accumulate until they are retrieved by programs that service those queues.

Element <MQFilewriter>

Information necessary to write the body of the MQ message to a file and includes the following:

•Element <Directory>

Directory location of the file. This can be any available local or network based location.

•Element <FileNamePattern>

Name of the file and the sequence pattern if there will be more than one file written to the same directory.

•Element <StartSequenceNumber>

Starting number to use for the pattern sequence.

•Element <WriteProperties>

Specification for whether or not the MQ properties should be written to the file. Default is false.

Figure A.1 UAC.xml File - Main Elements

A.2 UAC.xml File - Sample

Figure A.2, below, illustrates a sample UAC.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<sb:UAC xmlns:sb="http://com.stonebranch/uac/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://com.stonebranch/uac/ UAC.xsd ">
  <sb:JMSConnection>
    <sb:Name>ActiveMQ</sb:Name>
    <sb:InitialContextProperties>
      <sb:Property>
        <sb:Name>java.naming.factory.initial</sb:Name>
        <sb:Value>org.apache.activemq.jndi.ActiveMQInitialContextFactory
        </sb:Value>
      </sb:Property>
      <sb:Property>
        <sb:Name>java.naming.provider.url</sb:Name>
        <sb:Value>tcp://172.16.30.101:61616</sb:Value>
      </sb:Property>
    </sb:InitialContextProperties>
    <sb:ConnectionFactory>ConnectionFactory</sb:ConnectionFactory>
    <sb:Listeners>
      <sb:JMSListener>
        <sb:Destination>dynamicQueues/MyQueue</sb:Destination>
        <sb:Actions>
          <sb:JMSFilewriter>
            <sb:Directory>outputDirectory</sb:Directory>
            <sb:FilenamePattern>jmsMessageQueue%Seq%.txt
            </sb:FilenamePattern>
            <sb:StartSequenceNumber>1</sb:StartSequenceNumber>
            <sb:WriteProperties>>false</sb:WriteProperties>
          </sb:JMSFilewriter>
        </sb:Actions>
      </sb:JMSListener>
      <sb:JMSListener>
        <sb:Destination>dynamicTopics/MyTopic</sb:Destination>
        <sb:Actions>
          <sb:JMSFilewriter>
            <sb:Directory>outputDirectory</sb:Directory>
            <sb:FilenamePattern>jmsMessageTopic%Seq%.txt
            </sb:FilenamePattern>
            <sb:StartSequenceNumber>1</sb:StartSequenceNumber>
            <sb:WriteProperties>>false</sb:WriteProperties>
          </sb:JMSFilewriter>
        </sb:Actions>
      </sb:JMSListener>
    </sb:Listeners>
  </sb:JMSConnection>
</sb:UAC>
```

```

        </sb:JMSFilewriter>
    </sb:Actions>
</sb:JMSListener>
</sb:Listeners>
</sb:JMSConnection>
<sb:SOAPConnection>
    <sb:URI>/axis2/services/UACInbound</sb:URI>
    <sb:Listeners>
        <sb:SOAPListener>
            <sb:Operation>myOperation</sb:Operation>
            <sb:Actions>
                <sb:SOAPFilewriter>
                    <sb:Directory>outputDirectory</sb:Directory>
                    <sb:FilenamePattern>soapMessage%Seq%.xml</sb:FilenamePattern>
                    <sb:StartSequenceNumber>1</sb:StartSequenceNumber>
                    <sb:writeEnvelope>false</sb:writeEnvelope>
                </sb:SOAPFilewriter>
            </sb:Actions>
        </sb:SOAPListener>
    </sb:Listeners>
</sb:SOAPConnection>
<sb:MQConnection>
    <sb:Name>MQ Series Listener</sb:Name>
    <sb:Host>Server Host or IP</sb:Host>
    <sb:QueueManagerName>MQ QueueManager</sb:QueueManagerName>
    <sb:Channel>MQ Channel</sb:Channel>
    <sb:Port>MQ Listening Port</sb:Port>
    <sb:Listeners>
        <sb:MQListener>
            <sb:QueueName>MQ Queue to Listen to</sb:QueueName>
            <sb:Actions>
                <sb:MQFilewriter>
                    <sb:Directory>Directory location to write file
                        to</sb:Directory>
                    <sb:FilenamePattern>Filename pattern to write with
                        %Seq% variable</sb:FilenamePattern>
                    <sb:StartSequenceNumber>0</sb:StartSequenceNumber>
                    <sb:writeProperties>false</sb:writeProperties>
                </sb:MQFilewriter>
            </sb:Actions>
        </sb:MQListener>
    </sb:Listeners>
</sb:MQConnection>
</sb:UAC>

```

Figure A.2 UAC.xml File - Sample

Web Services Description Language (WSDL) File

B.1 Overview

Web Services Description Language (WSDL) is a way for an external client to obtain knowledge of the SOAP services and methods made available by a server.

The location of the WSDL file is:

`http://[hostname or Address]:[Port]/axis2/services/UACInbound?wsdl`

In this format:

- **[hostname or Address]** = Hostname or IP Address of the server hosting the Universal Event Monitor for SOA.
- **[Port]** = Listening port of the SOAP Connector (default 7880).

Section [B.2 WSDL File](#) illustrates the information provided by a WSDL request. The information has been formatted for readability.

B.2 WSDL File

WSDL file: "UACInbound?wsdl"

Target namespace: <http://inbound.uac.stonebranch.com>

Overview:

Services	Bindings	Port types	Messages
UACInbound	UACInboundSOAP11Binding UACInboundSOAP12Binding UACInboundHttpBinding	UACInboundPortType	updateRequest , processRequest , processResponse , UACException ,

WSDL Definition	
Name	Documentation
N/A	UACInbound

Services	
Name	Documentation
UACInbound	N/A

Service : UACInbound			
Port Name	Binding	Address Extensibility	Documentation
UACInboundSOAP11port_http	UACInboundSOAP11Binding	<soap:address location="http://[Host]:7888 /axis2/services /UACInbound"/>	N/A
UACInboundSOAP12port_http	UACInboundSOAP12Binding	<soap12:address location="http://[Host]:7888 /axis2/services /UACInbound"/>	N/A
UACInboundHttpport	UACInboundHttpBinding	<http:address location="http://[Host]:7888 /axis2/services /UACInbound"/>	N/A

Binding : UACInboundSOAP11Binding	
Port Type	UACInboundPortType
Extensibility	<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
Operations	update, process,

Binding : UACInboundSOAP12Binding	
Port Type	UACInboundPortType
Extensibility	<soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
Operations	update, process,

Figure B.1 WSDL file (page 1)

Binding : UACInboundHttpBinding			
Port Type	UACInboundPortType		
Extensibility	<http:binding verb="POST"/>		
Operations	update, process,		

Port Type : UACInboundPortType			
Operation Name	Input message	Output message	Documentation
process	processRequest	processResponse	N/A

Messages:

updateRequest

Documentation :

N/A

Part Name	Element	Type	Documentation
parameters	ns0:update	N/A	N/A

processRequest

Documentation :

N/A

Part Name	Element	Type	Documentation
parameters	ns0:process	N/A	N/A

processResponse

Documentation :

N/A

Part Name	Element	Type	Documentation
parameters	ns0:processResponse	N/A	N/A

Figure B.2 WSDL file (page 2)

UACException

Documentation :

N/A

Part Name	Element	Type	Documentation
parameters	ns0:UACException	N/A	N/A

Figure B.3 WSDL file (page 3)

Glossary

API

API (Application Programming Interface) is a set of functions, procedures, methods, classes, or protocols that an operating system, library, or service provides to support requests made by computer programs.

Asynchronous Communication

Transmission of data or sending of messages without the need to wait for a reply from the destination before continuing with the next operation.

Channel

Medium used to convey information from sender to receiver.

Communications Protocol

Set of standard rules for data representation, signaling, authentication, and error detection required to send information over a communications channel.

Connector

Component used to allow one system or application to communicate with another system or application. A connector can be embedded within an application or operate as a stand-alone component.

Container

An application environment that provides a runtime environment that offers services such as security, authentication, transaction management, and deployment to an application developer, thus enabling a faster implementation and rollout.

HTTP

HTTP (HyperText Transfer Protocol) is a synchronous request / reply protocol; the underlying protocol of the World Wide Web.

Internet Application

A web application (webapp) that is accessed via web browser over the Internet.

Internet Workload

Internet workload is any application, service, or function that operates in an Internet environment, such as web applications or container applications, and supports an Internet-based communication protocol such as HTTP or SOAP.

JMS

JMS (Java Message Service) is an [API](#) for the asynchronous exchange of messages between two or more clients. JMS uses both point-to-point (queue-based) and publish / subscribe (topic-based) messaging patterns.

JMS Connector

Component that allows the sending and receiving of JMS messages between applications.

Listen MEP

Listen Message Exchange Pattern, or MEP, refers to a component that listens for a message from an application or service.

Message

Abstract format, or container, for sending data between applications or services. No implementation is implied.

Message-Based Application

A message-based application accesses a target application by sending a message to a queue that is controlled by the target application. This queue must be known and accessible to the application sending the message.

Message-Based Workload

Any application, service, or function that supports a message-based communication protocol such as JMS or MQ.

Message Exchange Pattern

A Message Exchange Pattern (MEP) describes the pattern of messages required by a communications protocol in order to establish or use a communication channel.

Publish MEP

Publish Message Exchange Pattern, or MEP, refers to a component that sends a message from an application or service to a target destination. This is an asynchronous, or non blocking, operation.

Remote Procedure Call

Remote Procedure Call (RPC) is the most common messaging pattern in [SOAP](#). In RPC, one network node (the client) sends a request message to another node (the server). The server immediately sends a response message to the client. This type of transaction also is known as "request / reply."

Request / Reply MEP

Request / Reply MEP represents an outbound request to a target workload followed by an inbound reply from a target workload. This is a synchronous operation, as the calling party waits, or blocks, for the reply to come back before releasing its resources and moving on to the next task.

SOA

SOA (Service-Oriented Architecture) provides methods for systems development and integration where systems group functionality around business processes and package these as interoperable services.

SOA also describes IT infrastructure, which allows different applications to exchange data with one another as they participate in business processes.

SOAP

SOAP (Simple Object Access Protocol) is a synchronous, XML-based messaging protocol. It defines a set of rules for exchanging XML-based messages over computer networks, normally using HTTP / HTTPS, although you can send SOAP messages over JMS, as well.

SOAP is used predominantly to provide an interface to web service-based workload or legacy workload with a web service interface.

SOAP Connector

Component that allows the sending and receiving of SOAP messages.

Web Services Description Language (WSDL)

WSDL is an XML-based language that provides a model for describing Web services. WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet.

Workload

Jobs, processes, applications, and services that require execution, usually in a scheduler or automation-based environment.

Customer Support

Stonebranch, Inc. provides customer support, via telephone and e-mail, for Universal Event Monitor for SOA and all Stonebranch Solutions components.

E-MAIL

All Locations

support@stonebranch.com

Customer support contact via e-mail also can be made via the Stonebranch website:

www.stonebranch.com

TELEPHONE

Customer support via telephone is available 24 hours per day, 7 days per week.

North America

(+1) 678 366-7887, extension 6

(+1) 877 366-7887, extension 6 [toll-free]

Europe

+49 (0) 700 5566 7887

Index

A

asynchronous communication *17*

C

checking logs *40*

configuration

inbound *21*

configuration file (UAC.xml) *21*

configuring

logging operations *40*

operations *21*

connectors *19*

D

defined ports *20*

F

files

configuration (UAC.xml) *21*

jar

JMS inbound *23*

UAC.xml file

main elements *44*

sample *47*

I

inbound configuration *21*

inbound connection

JMS *25*

MQ *29*

SOAP *27*

J

jar files

JMS inbound *23*

JMS *17*

inbound connection *25*

JMS Connector *19*

Inbound *34*

L

Listen MEP *18*

listening attributes of UAC *43*

logging

configuring for server component *40*

configuring for UAC *42*

configuring operations *40*

levels *40*

logs *40*

M

MEP

Listen *18*

Message Exchange Pattern (MEP) *18*

MQ

inbound connection *29*

MQ Connector

Inbound *36*

O

operations

configuration *21*

P

ports *20*
protocols *17*
 JMS *17*
 SOAP *17*

S

server component
 logging configuration *40*
SOAP
 inbound connection *27*
SOAP Connector *19*
 Inbound *32*
SOAP protocol *17*
synchronous communication *17*

U

UAC.xml file
 main elements *44*
 sample *47*
Universal Command Agent for SOA
 ports *20*



**950 North Point Parkway, Suite 200
Alpharetta, Georgia 30005
U.S.A.**

