

# **SAS<sup>®</sup> 9.3 Web Applications Clustering**



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® 9.3 Web Applications: Clustering*. Cary, NC: SAS Institute Inc.

**SAS® 9.3 Web Applications: Clustering**

Copyright © 2011, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America.

**For a hardcopy book:** No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**For a Web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

**U.S. Government Restricted Rights Notice:** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227–19, Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, December 2011

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at [support.sas.com/publishing](http://support.sas.com/publishing) or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

---

# Contents

<i>What's New in SAS Web Applications Clustering</i> . . . . .	v
<b>Chapter 1 • Overview</b> . . . . .	<b>1</b>
Web Application Server Clustering Overview . . . . .	1
SAS Middle-Tier Limitations . . . . .	2
<b>Chapter 2 • Common Pre-Configuration Tasks</b> . . . . .	<b>5</b>
Purpose . . . . .	5
Event Generation Framework . . . . .	5
Reconfigure the SAS Content Server Repository . . . . .	7
Reconfigure SAS Workflow . . . . .	9
Access to Configuration Files for SAS Web Applications on Multiple Machines . . . . .	14
Configure Logging . . . . .	14
<b>Chapter 3 • JBoss Clustering</b> . . . . .	<b>17</b>
JBoss Clustering Overview . . . . .	17
Create Additional JBoss Server Instances . . . . .	19
Configure Additional JBoss Server Instances . . . . .	20
Configure Start-Up Scripts . . . . .	24
Configure a Load Balancing HTTP Server . . . . .	25
Configure JBoss to Use mod_jk . . . . .	27
Perform Common Configuration Tasks . . . . .	27
<b>Chapter 4 • Oracle WebLogic Server 11g Clustering</b> . . . . .	<b>29</b>
WebLogic Server Clustering Overview . . . . .	29
Pack the Domain and Unpack on Additional Machines . . . . .	32
Add Additional Machines . . . . .	32
Clone the Managed Server . . . . .	33
Create a Cluster and Assign the Servers . . . . .	34
Target the Mail Session to the Cluster . . . . .	34
Reconfigure the JMS Resources . . . . .	34
Retarget the JDBC Data Source . . . . .	37
Deploy the SAS Web Applications to the Cluster . . . . .	37
Configure a Load Balancing HTTP Server . . . . .	39
Perform Common Configuration Tasks . . . . .	41
<b>Chapter 5 • IBM WebSphere Application Server 7.0 Clustering</b> . . . . .	<b>43</b>
WebSphere Application Server Clustering Overview . . . . .	43
Add Additional Machines . . . . .	46
Create a Server Template from SASServer1 . . . . .	46
Create the Cluster and Add Servers . . . . .	46
Configure JVM Options . . . . .	47
Configure the Cookie Name . . . . .	47
Target SAS Web Applications to the Cluster . . . . .	48
Add SASCluster as a SAS Messaging Bus Member . . . . .	49
Configure a Load Balancing HTTP Server . . . . .	51
Perform Common Configuration Tasks . . . . .	51

<b>Chapter 6 • Common Tasks</b> .....	<b>53</b>
Confirm the SAS Web Report Studio Clustering Setting .....	53
Configure Web Report Studio General Purpose Log Location .....	53
Deploy SAS Themes and SAS Themes for Flex Application to an HTTP Server .....	54
Change the Connections for the SAS Web Applications .....	54
Change the Connection for SAS Content Server .....	55
Change the WebDAV Repository URL .....	55
Disable the Redirection Filter for the SAS Web Applications .....	56
Start Software in the Correct Sequence .....	57

## What's New

# What's New in SAS Web Applications Clustering

---

## Overview

This document applies to the first maintenance release for SAS 9.3.

---

## SAS Middle Tier Supported on 64-Bit Systems Only

The SAS 9.3 middle-tier software is supported on 64-bit operating systems only.

---

## SAS Workflow Modifications

SAS Workflow requires manual reconfiguration in order to be deployed in a cluster. If your site does not require high-availability and scalability for SAS Workflow, you can choose to deploy it to a stand-alone server instance to avoid the reconfiguration task.

---

## Cookie Management for IBM WebSphere Application Server

In previous releases of this document, one of the configuration tasks for IBM WebSphere Application Server was to override the default cookie name for the servers in the cluster. In this release, the strategy is reversed. The cookie name for the stand-alone server instance is changed, and the default cookie name is used for the servers in the cluster. This change is in response to sites that lost session affinity with SAS Information Delivery Portal, SAS BI Portlets, and SAS JSR 168 Remote Portlets Web applications.

---

## **SAS Content Server Modifications**

When SAS Content Server is clustered, one instance must be identified in the `-Dsas.scs.host` and `-Dsas.scs.port` JVM options. This modification is made to the JVM options for all Web application server instances. They are all modified to use a single host name and port number.

---

## **SAS BI Dashboard Modifications**

When SAS BI Dashboard is clustered on Oracle WebLogic Server, the `weblogic.xml` file for the `sas.bidashboard.war` file must be modified. The file is modified to include a cookie name that is used for session affinity.

## 1

## Overview

<i>Web Application Server Clustering Overview</i> .....	1
<i>SAS Middle-Tier Limitations</i> .....	2
SAS Workflow .....	2
Java Messaging Services .....	2
Strategies for Overcoming Limitations .....	3

---

## Web Application Server Clustering Overview

The SAS Web applications can be clustered to improve performance and to provide high availability. Performance is improved due to additional server instances to handle requests and distribution of memory needs to additional processes. Performance can also be improved by deploying servers on to additional hardware. Clustering can also provide high availability. If server instances are clustered on a single machine, then there is protection against software failure. If multiple machines are used for clustering servers, then the additional machines in the cluster provide protection from software failure and protection from hardware failure. This document provides application-specific considerations for clustering and Web application server-specific details for clustering the SAS Web applications that are deployed with SAS Enterprise Business Intelligence Server.

Here are the high-level steps to follow:

- 1 Plan for Deployment:** Using functional and non-functional requirements as well as a sizing and capacity plan, determine how many servers you need and the server capacity that you need.
- 2 Choose Deployment Plan:** Based on the input from the previous step, choose a deployment plan that uses single or multiple servers.
- 3 Install SAS Software with the SAS Deployment Wizard:** Use the wizard to install SAS software either interactively or non-interactively. The interactive method provides a series of prompts to enable you to modify parameters based on your environment. Alternatively, you can configure a response file with parameters for your environment and run the wizard in noninteractive mode. On the first machine in the middle tier, use the SAS Deployment Wizard to install and configure software. You can choose to have the wizard automatically configure the Web application server, or configure the Web application server yourself. If you choose to configure the Web application server yourself, follow the steps in the Instructions.html file that is generated by the wizard. On the additional

machines in the middle tier, do not use the SAS Deployment Wizard. Instead, copy selected directories and configuration files from the first machine.

- 4 Prepare for Clustering: Manually configure Web application servers and the SAS Web applications to take advantage of clustering. You might need to configure resources such as JDBC data sources and Java Messaging Services. The work to perform during this step depends on the number of clusters, the number of servers within each cluster, and whether SAS Web applications are relocated to new Web application servers.
- 5 Cluster: Create the Web application server cluster. WebLogic Server and WebSphere Application Server provide administrative interfaces that assist with creating a cluster. For JBoss Application Server, the cluster is based on multiple stand-alone servers.
- 6 Perform Post Configuration for the Cluster: After creating the cluster, configure a load-balancing HTTP server so that user requests are load balanced and routed to the clustered servers. The SAS Web applications rely on session affinity, also called “sticky sessions.” This feature ensures that subsequent HTTP requests for the SAS Web applications are always routed to the same Web application server.
- 7 Validate the Cluster: Perform a series of logon attempts to confirm that loads are distributed across the servers in the cluster. Afterward, perform more intensive tasks to confirm that the cluster meets your performance and high availability requirements.

---

## SAS Middle-Tier Limitations

### SAS Workflow

The SAS Workflow Web application must be modified with manual configuration steps to be used in a cluster. For sites that do not need performance and scalability for SAS Workflow, the Web application can be deployed to a stand-alone server without modification.

### Java Messaging Services

A Java Messaging Service (JMS) server and the JMS resources are used by the Event Generation Framework Web application, SAS Workflow, and SAS Web Infrastructure Platform Scheduling Services. Each Web application server vendor offers different strategies for providing JMS resources in a clustered deployment.

For JBoss, this document describes using a dedicated, single JMS server that is not to be shared with other application Java virtual machines (JVM). The clustered server instances are configured to use external Java naming and directory interface (JNDI) contexts to the JMS resources on the stand-alone server.

For WebLogic Server, this document describes leaving the JMS server on the stand-alone server. The JMS resources are made available to the servers in the cluster by configuring a JMS module and destinations that treat the JMS server on the stand-alone server as a foreign JMS server.

IBM WebSphere Application Server Network Deployment can provide JMS resources in a cluster and make the resources available to the cell with few

configuration tasks. As a result of the cell-level scope provided by WebSphere Application Server, each of the JMS resources are deployed to the cell.

## Strategies for Overcoming Limitations

### Deploy a Stand-alone Web Application Server

The Event Generation Framework cannot be clustered. The packaging of Event Generation Framework within the same EAR file that is used for SAS BI Dashboard presents a challenge for sites that intend to cluster SAS BI Dashboard. The strategy that is described in the detailed steps in the sections that follow is to deploy a stand-alone Web application server and to deploy the Event Generation Framework Web application to it.

### Hot and Cold Standby Strategy

This strategy can be used to meet high availability requirements. The strategy is to configure two machines, each with a stand-alone Web application server and the SAS BI Dashboard Event Generator Web application. One Web application server is active on the hot machine, and the other Web application server is inactive on the cold standby machine.

### High Availability Software Strategy

Another alternative for ensuring high availability is to use high availability software such as IBM PowerHA SystemMirror, Oracle Solaris Cluster, HP Serviceguard, and Veritas Cluster Server by Symantec. Most of these products provide virtual networking and replicated data storage. In this case, a virtual IP address is shared by two machines and the high availability software detects a failure and manages the transition of activity from the failed machine to the standby machine. The following list identifies the high-level steps that the high availability software takes:

- 1 Cleans up the process on the failed machine.
- 2 Releases disk resources.
- 3 Shuts down the network interface to which the virtual IP address is bound.
- 4 Mounts the disks on the standby machine.
- 5 Checks the disk.
- 6 Starts the network interface to which the virtual IP address is bound.
- 7 Starts the Web application server and the SAS Web applications that are deployed on it.



# 2

## Common Pre-Configuration Tasks

<i>Purpose</i> .....	5
<i>Event Generation Framework</i> .....	5
<i>Reconfigure the SAS Content Server Repository</i> .....	7
<i>Reconfigure SAS Workflow</i> .....	9
<i>Access to Configuration Files for SAS Web Applications on Multiple Machines</i> .....	14
<i>Configure Logging</i> .....	14

---

### Purpose

This section describes steps that are common to all Web application servers. These steps are required to perform the Web application server-specific configuration steps that are described later in this document.

---

### Event Generation Framework

The Event Generation Framework Web application is packaged in the SAS BI Dashboard EAR file. The SAS BI Dashboard Web application can be clustered, but the Event Generation Framework cannot be clustered. The events generated by the framework must be generated by a single instance of the framework so that the events are generated for a single target. You can keep the applications in the same EAR file and provide high availability with a hot and cold standby strategy, or use high availability software to run one active instance of both SAS BI Dashboard and the Event Generation Framework.

However, if SAS BI Dashboard receives a lot of use and multiple instances are needed, then there is another option. You can separate the event generation framework WAR file from the SAS BI Dashboard EAR file and then deploy a single instance of the framework on a stand-alone server, but deploy multiple instances of SAS BI Dashboard to the cluster. This strategy is used throughout the rest of this document.

The following steps describe the option of extracting the framework WAR file and then modifying the SAS BI Dashboard EAR file. When these steps are complete, the framework can be deployed to a stand-alone server, and the SAS BI Dashboard EAR file can be deployed to the cluster. To ensure high availability, consider using a

hot and cold standby strategy or high availability software for the stand-alone server. Follow these steps:

- 1 Make a backup of the `SAS-config-dir/Levn/Web/Staging/sas.bidashboard4.3.ear` file.

- 2 In a temporary directory, extract the contents of the `sas.bidashboard4.3.ear` file:

```
jar xf sas.bidashboard4.3.ear
```

- 3 Move the `sas.eventsgenerationframework.war` directory from the temporary directory to another temporary directory. (Modifications to this WAR file are done after the next steps that describe changes to the EAR file.)

- 4 Edit the `META-INF/application.xml` file and remove the following information:

```
<module>
  <web>
    <web-uri>sas.eventsgenerationframework.war</web-uri>
    <context-root>SASBIDashboardEventGen</context-root>
  </web>
</module>
```

- 5 For deployments that use WebLogic Server, perform the following steps:

- a In a temporary directory, extract the contents of the `sas.bidashboard.war` file:

```
jar xf sas.bidashboard.war
```

- b Edit the `WEB-INF/weblogic.xml` file and add the following information just before the closing `</weblogic-web-app>` tag:

```
<session-descriptor>
  <cookie-name>SASBIDashboard.sessionID</cookie-name>
  <url-rewriting-enabled>>false</url-rewriting-enabled>
</session-descriptor>
```

- c Repackage `sas.bidashboard.war` file from the temporary directory:

```
jar cf ../sas.bidashboard.war .
```

- 6 Repackage `sas.bidashboard4.3.ear` file from the temporary directory:

```
jar cf ../sas.bidashboard4.3.ear .
```

- 7 In the temporary directory that contains the `sas.eventsgenerationframework.war` file, extract the contents of the WAR file to a temporary directory. Then make the edit that applies to your Web application server:

- For WebLogic Server, edit the `WEB-INF/weblogic.xml` file, and add a `context-root` element as shown in the following example:

```
<!DOCTYPE weblogic-web-app PUBLIC "-//BEA Systems, Inc.//DTD Web
Application 8.1//EN"
"http://www.bea.com/servers/wls810/dtd/weblogic810-webjar.dtd">
<weblogic-web-app>
  <context-root>/SASBIDashboardEventGen</context-root>
  <jsp-descriptor>
    <jsp-param>
...

```

- For JBoss, edit the `WEB-INF/jboss-web.xml` file, and change the context-root element as shown in the following example:

```
<jboss-web>
  <resource-ref>
    <res-ref-name>jms/AlertQueue</res-ref-name>
    <res-type>javax.jms.Queue</res-type>
    <jndi-name>sas/jms/AlertQueue</jndi-name>
  </resource-ref>
  <resource-ref>
    <res-ref-name>jms/QueueConnectionFactory</res-ref-name>
    <res-type>javax.jms.QueueConnectionFactory</res-type>
    <jndi-name>sas/jms/QueueConnectionFactory</jndi-name>
  </resource-ref>
  <context-root>/SASBIDashboardEventGen</context-root>
</jboss-web>
```

- For WebSphere Application Server, the context root is set when the WAR file is deployed.

- 8 Repackage the `sas.eventsgenerationframework.war` file from the temporary directory:

```
jar cf ../sas.eventsgenerationframework.war .
```

- 9 Stop, undeploy, and remove the existing `sas.bidashboard4.3.ear` file on the Web application server.
- 10 Deploy the repackaged `sas.bidashboard4.3.ear` and `sas.eventsgenerationframework.war` applications.

For deployments that use WebSphere Application Server, make sure that when the SAS Web applications are reinstalled, that the class loader order is set to **Classes loaded with the application class loader first** for each Web module. If the class loading order is incorrect, then SAS BI Dashboard can generate HTTP 500 errors when trying to access the `com.sas.collection.StaticDictionaryInterface` class.

---

## Reconfigure the SAS Content Server Repository

The SAS Content Server Web application can be configured by the SAS Deployment Wizard to use a file system or a database for storing content. The index for the repository is stored on a file system even if content is stored in a database. To deploy multiple instances of SAS Content Server, it must be configured to use a database for storing content, and it must also be configured so that indexes are stored in unique directory names.

The following steps describe how to determine whether SAS Content Server is using file system for storage and how to reconfigure it to use unique directory names for indexes.

- 1 Determine whether the file system or a database is used for storage. Open the `SAS-config-dir/Levn/AppData/SASContentServer/Repository/repository.xml` file and check the class attribute for the `FileSystem` element.

**Example Code 2.1** *Repository.xml with DbFileSystem Class*

```
<FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
  <param name="driver" value="javax.naming.InitialContext"/>
  ...

```

**Example Code 2.2** *Repository.xml with LocalFileSystem Class*

```
<FileSystem class="org.apache.jackrabbit.core.fs.local.LocalFileSystem">
  <param name="path" value="{rep.home}/repository"/>
</FileSystem>

```

- 2 If the repository.xml file uses the LocalFileSystem class, then follow the steps in “Reconfiguring the SAS Content Server to Use a Database for Storage” in Chapter 9 of *SAS Intelligence Platform: Middle-Tier Administration Guide*. When performing those steps, instead of recreating a directory named Repository, create a directory named **SASServer2**.

Return here after reconfiguring SAS Content Server to use a database for storage.

- 3 For each Web application server instance in the cluster, create a directory in the SASContentServer directory that is named for the Web application server instance name, such as **SASServer2**. (Deployments that just reconfigured SAS Content Server to use a database already have a SASServer2 directory instance.)

The name of the directory must match the value of the **-Dsas.appserver.instance.id** JVM option:

```
mkdir SAS-config-dir/Levn/AppData/SASContentServer/SASServer2
```

**Note:** If the Web application server instances are deployed on separate machines, then create the directory and parent directories on each machine. Use the Web application server instance name that runs on the applicable machine as part of the path.

- 4 Copy the **SASServer2/repository.xml** file into each of the directories created in step 3. If SAS Content Server was configured by the SAS Deployment Wizard to store content in a database, then copy the **Repository/repository.xml** file instead.
- 5 Edit each of the copied repository.xml files and set the node ID by uncommenting the Cluster element and setting the value of the **id** attribute to the Web application server instance name. As with step 3, the value must match the value for the **-Dsas.appserver.instance.id** JVM option:

```
<!-- Cluster configuration -->
<Cluster id="SASServer2">
  <Journal class="org.apache.jackrabbit.core.journal.DatabaseJournal">
    <param name="revision" value="{rep.home}/revision.log"/>
    <param name="driver" value="javax.naming.InitialContext"/>
    <param name="url" value="sas/jdbc/SharedServices"/>
    <param name="schemaObjectPrefix" value="SAS_SCS_"/>
    <param name="schema" value="mysql"/>
  </Journal>
</Cluster>

```

- 6 Select one host and port to use for setting the **-Dsas.scs.host** and **-Dsas.scs.port** JVM options on all the other Web application server instances. These settings are used by SAS Web Infrastructure Platform for accessing SAS Content Server.

## Reconfigure SAS Workflow

Follow the steps in this section to configure the SAS Workflow Web application in a cluster. If performance and scalability are not needed for SAS Workflow, then deploy the Web application to a stand-alone server instance instead of following the steps in this section.

- 1 Create a JBoss Cache 1.4.1 configuration file named `jboss-cache.xml` that uses multicast synchronous replication. Modify the UDP IP address, port, and time to live (TTL) settings, and use the same values for all servers in the cluster. When selecting the port, be aware that SAS performs multicast communication on UDP ports 8561, 7570, 7571, and 7572.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ===== -->
<!--                                     -->
<!-- Sample TreeCache Service Configuration -->
<!-- Recommended for use as Hibernate's 2nd Level Cache -->
<!-- For use with JBossCache >= 1.3.0 ONLY!!! -->
<!--                                     -->
<!-- ===== -->

<server>
  <!-- ===== -->
  <!-- Defines TreeCache configuration -->
  <!-- ===== -->

  <mbean code="org.hibernate.cache.OptimisticTreeCache"
        name="jboss.cache:service=WorkflowCache">

    <depends>jboss:service=Naming</depends>
    <depends>jboss:service=TransactionManager</depends>

    <!--
      Configure the TransactionManager
    -->
    <!-- attribute name="TransactionManagerLookupClass">
      org.jboss.cache.GenericTransactionManagerLookup
    </attribute -->

    <!--
      Node locking scheme:
      OPTIMISTIC
      PESSIMISTIC (default)
    -->
    <attribute name="NodeLockingScheme">OPTIMISTIC</attribute>

    <!--
      Note that this attribute is IGNORED if your NodeLockingScheme
      above is OPTIMISTIC.
      Isolation level : SERIALIZABLE
    -->
```

```

        REPEATABLE_READ (default)
        READ_COMMITTED
        READ_UNCOMMITTED
        NONE
-->
<attribute name="IsolationLevel">REPEATABLE_READ</attribute>

<!--
    Valid modes are LOCAL
        REPL_ASYNC
        REPL_SYNC
        INVALIDATION_ASYNC
        INVALIDATION_SYNC
-->

<!-- This should ideally be set to INVALIDATION_ASYNC but due to
    JBCACHE-806 this has to be REPL_ASYNC for now -->
<attribute name="CacheMode">REPL_SYNC</attribute>

<!--
    Just used for async repl: use a replication queue
-->
<attribute name="UseReplQueue">false</attribute>

<!--
    Replication interval for replication queue (in ms)
-->
<attribute name="ReplQueueInterval">0</attribute>

<!--
    Max number of elements which trigger replication
-->
<attribute name="ReplQueueMaxElements">0</attribute>

<!-- Name of cluster. Needs to be the same for all clusters, in order
    to find each other
-->
<attribute name="ClusterName">WorkflowCache</attribute>

<!-- JGroups protocol stack properties. Can also be a URL,
    e.g. file:/home/bela/default.xml
    <attribute name="ClusterProperties"></attribute>
-->

<attribute name="ClusterConfig">
  <config>
    <!-- UDP: if you have a multihomed machine,
        set the bind_addr attribute to the appropriate NIC IP address -->
    <!-- UDP: On Windows machines, because of the media sense feature
        being broken with multicast (even after disabling media sense)
        set the loopback attribute to true -->
    <UDP mcast_addr="235.1.2.153" mcast_port="4699"
        ip_ttl="32" ip_mcast="true"
        mcast_send_buf_size="150000" mcast_rcv_buf_size="80000"
        ucast_send_buf_size="150000" ucast_rcv_buf_size="80000"
        loopback="true"/>
  </config>
</attribute>

```

```

<PING timeout="2000" num_initial_members="3"
    up_thread="false" down_thread="false"/>
<MERGE2 min_interval="10000" max_interval="20000"/>
<!--    <FD shun="true" up_thread="true" down_thread="true" /> -->
<FD_SOCKET/>
<VERIFY_SUSPECT timeout="1500"
    up_thread="false" down_thread="false"/>
<pbcast.NAKACK gc_lag="50" retransmit_timeout="600,1200,2400,4800"
    max_xmit_size="8192" up_thread="false" down_thread="false"/>
<UNICAST timeout="600,1200,2400" window_size="100" min_threshold="10"
    down_thread="false"/>
<pbcast.STABLE desired_avg_gossip="20000"
    up_thread="false" down_thread="false"/>
<FRAG frag_size="8192"
    down_thread="false" up_thread="false"/>
<pbcast.GMS join_timeout="5000" join_retry_timeout="2000"
    shun="true" print_local_addr="true"/>
<pbcast.STATE_TRANSFER up_thread="true" down_thread="true"/>

    <!-- UDP mcast_addr="235.1.2.152" mcast_port="4699"/>
    <PING/>
    <MERGE2/>
    <FD_SOCKET/>
    <VERIFY_SUSPECT/>
    <pbcast.NAKACK/>
    <UNICAST/>
    <pbcast.STABLE/>
    <FRAG/>
    <pbcast.GMS/-->
</config>
</attribute>

<!--
    Whether or not to fetch state on joining a cluster
    NOTE this used to be called FetchStateOnStartup and
    has been renamed to be more descriptive.
-->
<attribute name="FetchInMemoryState">false</attribute>

<!--
    Number of milliseconds to wait until all responses for a
    synchronous call have been received.
-->
<attribute name="SyncReplTimeout">20000</attribute>

<!-- Max number of milliseconds to wait for a lock acquisition -->
<attribute name="LockAcquisitionTimeout">15000</attribute>

<!--
    The max amount of time (in milliseconds) we wait until the
    initial state (ie. the contents of the cache) are retrieved from
    existing members in a clustered environment
-->
<attribute name="InitialStateRetrievalTimeout">20000</attribute>

<!-- Name of the eviction policy class. -->

```

```

        <attribute name="EvictionPolicyClass">
            org.jboss.cache.eviction.LRUPolicy
        </attribute>
        <attribute name="EvictionPolicyConfig">
            <config>
                <attribute name="wakeUpIntervalSeconds">5</attribute>
                <!-- Cache wide default -->
                <region name="/_default_">
                    <attribute name="maxNodes">5000</attribute>
                    <attribute name="timeToLiveSeconds">1000</attribute>
                </region>
            </config>
        </attribute>

        <!--
            Indicate whether to use region based marshalling or not.
            Set this to true if you are running under a scoped
            class loader, e.g., inside an application server. Default is "false".
        -->
        <attribute name="UseRegionBasedMarshalling">false</attribute>

    </mbean>
</server>

```

- 2 Make a backup of the `SAS-config-dir/Levn/Web/Staging/sas.workflow9.3.ear` file.
- 3 In a temporary directory, extract the contents of the `sas.workflow9.3.ear` file.
- 4 In another temporary directory, extract the contents of the `sas.workflow.war` file.
- 5 In a third temporary directory, extract the contents of the `sas.workflow.war/WEB-INF/lib/sas.workflow.engine.server.jar` file.
- 6 Add the `jbosscache.xml` file to the root of the extracted JAR file. Put it in the same directory as the existing `ehcache.xml` file.
- 7 Edit each hibernate mapping file (\*.hbm.xml) in the hibernate directory of the extracted JAR file. Replace all instances of `read-write` with `transactional`.
- 8 Repackage `sas.workflow.engine.server.jar` from the temporary directory and copy it to the first temporary directory. (Overwrite the original JAR file.)
- 9 Edit the `sas.workflow.war/WEB-INF/spring-config/data-config.xml` file and make the following changes:
  - a Change the `hibernate.cache.provider` class from `org.hibernate.cache.EhCacheProvider` to `org.hibernate.cache.OptimisticTreeCacheProvider`.
  - b Change the `hibernate.cache.provider_configuration_file_resource_path` from `/ehcache.xml` to `jbosscache.xml`. Do not add a leading slash.
  - c Add two new `<prop>` elements under the existing hibernate properties. See the following code example for the changes:

```

<property name="hibernateProperties">
    <props>
        <prop key="hibernate.dialect">

```

```

        ${workflow.hibernate.dialect}
    </prop>
    <prop key="show_sql">true</prop>
    <prop key="use_outer_join">true</prop>
    <prop key="hibernate.cache.provider_class">
        org.hibernate.cache.OptimisticTreeCacheProvider
    </prop>
    <prop key="hibernate.cache.provider_configuration_file_resource_path">
        jboss-cache.xml
    </prop>
    <prop key="hibernate.cache.use_query_cache">false</prop>
    <prop key="hibernate.connection.isolation">2</prop>
    <prop key="hibernate.cache.use_second_level_cache">true</prop>
    <prop key="hibernate.transaction.manager_lookup_class">
        org.hibernate.transaction.JBossTransactionManagerLookup
    </prop>
</props>

```

**Note:** For WebSphere Application Server, use `org.hibernate.transaction.WebSphereTransactionManagerLookup` for the `manager_lookup_class`. For WebLogic Server, use `org.hibernate.transaction.WeblogicTransactionManagerLookup` for the `manager_lookup_class`.

- 10 Download JBoss Cache 1.4.1SP13. This is listed as the JBoss Cache Core Edition at <http://community.jboss.org/wiki/JBossCacheDownloadsPage>.
- 11 Extract `jboss-cache-dist-1.4.1.SP13.zip`.
  - For JBoss, copy `lib/jboss-cache.jar` to `sas.workflow.war/WEB-INF/lib`.
  - For WebSphere Application Server, copy the following JAR files to `sas.workflow.war/WEB-INF/lib`:
    - `jboss-cache.jar`
    - `jboss-common.jar`
    - `jboss-jmx.jar`
      - 1 Before copying the `jboss-jmx.jar` file, extract it to a temporary directory.
      - 2 Delete the entire `javax` package from the extracted contents and repackage the JAR file.
    - `jboss-minimal.jar`
    - `jboss-serialization.jar`
    - `jboss-system.jar`
    - `trove.jar`
- 12 Repackage `sas.workflow.war` and copy it to the temporary directory for the `sas.workflow9.3.ear` file.
- 13 Repackage `sas.workflow9.3.ear` and copy it to `SAS-config-dir/Levn/Web/Staging/sas.workflow9.3.ear`.
- 14 Deploy the modified `sas.workflow9.3.ear` file.

---

## Access to Configuration Files for SAS Web Applications on Multiple Machines

If you plan to configure a cluster on multiple machines, then copy the following directories (files and subdirectories) to each additional machine in the cluster. Use the same directory paths on all the machines:

- *SAS-config-dir/Levl/Web/Applications*
- *SAS-config-dir/Levl/Web/Common*
- *SAS-config-dir/Levl/Web/Temp*
- *SAS-config-dir/Levl/Web/Utilities*
- The directories needed by SAS Content Server should already be copied to the additional machines. These are named similar to *SAS-config-dir/Levl/AppData/SASContentServer/SASServer2*.

---

## Configure Logging

Most of the SAS Web applications use log4j for logging. The logging location for each application is controlled with log4j configuration files in the *SAS-config-dir/Levn/Web/Common/LogConfig*. The default configuration for each application includes some type of file appender. The file appender identifies a single pathname for each SAS Web application. As a result, when multiple instances of each application open a log file, each instance attempts to open an identical pathname. This causes resource contention for vertical clusters. The following task describes how to modify the log4j configuration file to use the unique server identity. This resolves the resource contention for vertical clusters, but is also useful for horizontal clusters if the log files are written to shared storage, or if they are aggregated to a central location. Later steps describe adding the unique server identity as a JVM option for each Web application server instance.

Follow these steps:

- 1 Navigate to the *SAS-config-dir/Levn/Web/Common/LogConfig* directory.
- 2 Edit each of the log4j configuration files, and add the `#{sas.appserver.instance.id}` parameter as shown in the following example:

```
<appender
  class="org.apache.log4j.DailyRollingFileAppender"
  name="SAS_FILE">
  <param
    name="datePattern"
    value="'.'yyyy-MM-dd"/>
  <param
    name="append"
    value="true"/>
  <param
    name="file"
```

```
value="/opt/SAS/Config/Lev1/Web/Logs/  
${sas.appserver.instance.id}/SASPortal4.3.log"/>  
...
```

**Note:** Keep the log file value on a single line. The log file location is split across two lines in the example due to space limitations.

Logging for SAS Web Report Studio is controlled through metadata for the general purpose log and with a JVM option for the key actions log. Steps that describe those changes are provided later in this document.



# 3

## JBoss Clustering

<i>JBoss Clustering Overview</i> .....	17
<i>Create Additional JBoss Server Instances</i> .....	19
<i>Configure Additional JBoss Server Instances</i> .....	20
Configure Port Numbers .....	20
Disable Local JMS Resources .....	22
Use JMS Resources on the Stand-Alone Server .....	23
<i>Configure Start-Up Scripts</i> .....	24
<i>Configure a Load Balancing HTTP Server</i> .....	25
<i>Configure JBoss to Use mod_jk</i> .....	27
<i>Perform Common Configuration Tasks</i> .....	27

### JBoss Clustering Overview

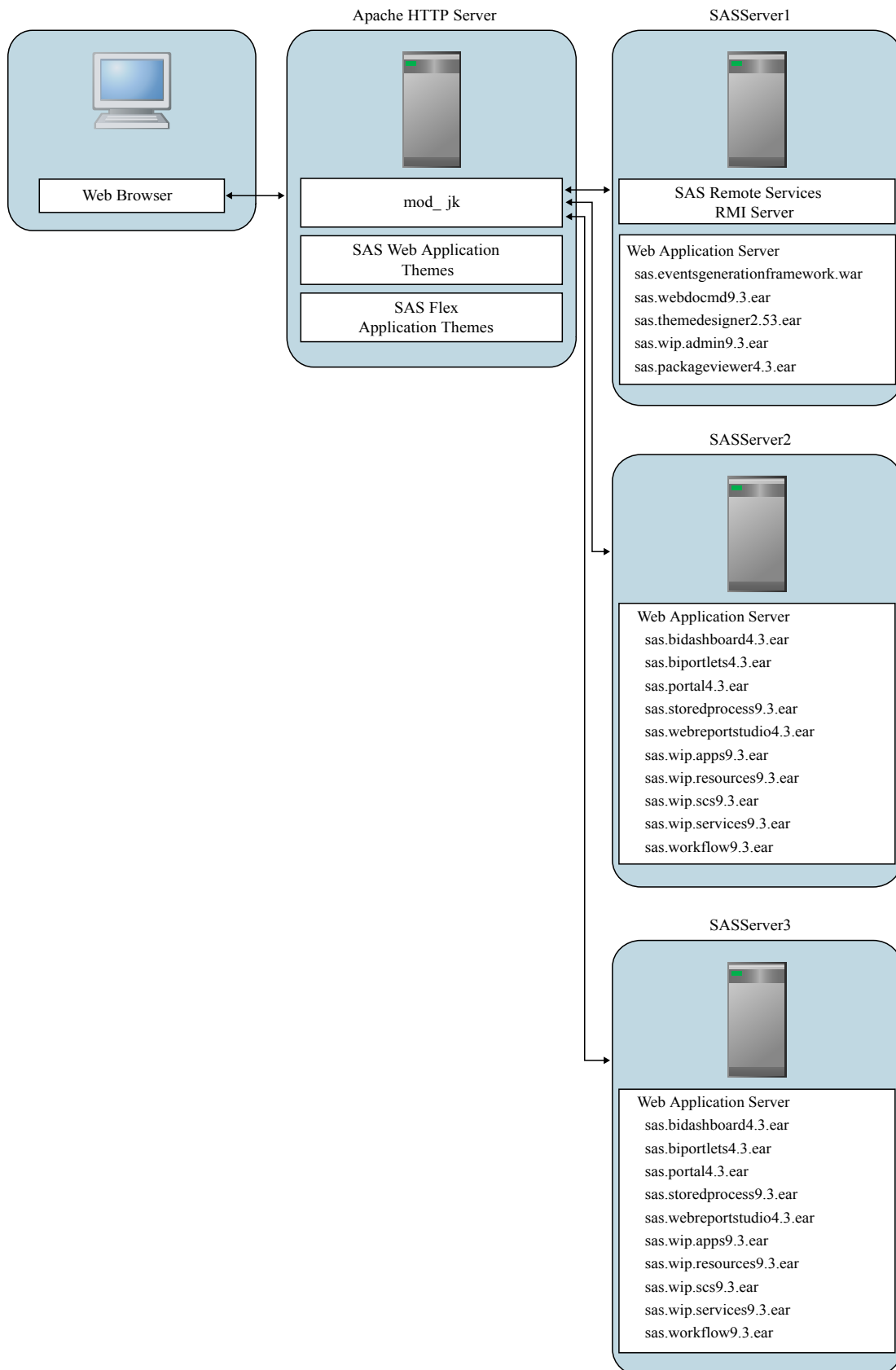
The concepts for clustering JBoss versions JBoss AS 4.2.3.GA, JBoss EAP 4.3, and JBoss AS 5.1.0.GA are the same. When the detailed steps differ, the version is identified in the instructions. The following list identifies the necessary steps to perform clustering with JBoss:

**TIP** Use the SAS Deployment Wizard to configure JBoss Application Server and deploy the SAS Web applications. This step is not described in this document.

- 1 Review the limitations in “[SAS Middle-Tier Limitations](#)” on page 2 and perform the pre-configuration tasks in “[Common Pre-Configuration Tasks](#)” on page 5.
- 2 Use the server created by the SAS Deployment Wizard as a template for creating the additional server for the cluster and the stand-alone server. Modify the server instances to use unique ports.
- 3 Configure an HTTP server to act as a load-balancing HTTP server. This document describes using Apache HTTP Server with mod\_jk.
- 4 Configure JBoss to work with the load-balancing HTTP server.
- 5 Perform the common configuration tasks.

The tasks in the following sections describe how to create a topology that is similar to the example in the following figure.

Figure 3.1 Sample JBoss Cluster Topology



## Create Additional JBoss Server Instances

Based on requirements and capacity planning, copy the SAServer1 directory for each additional server and rename the directory SAServer2, SAServer3, and so on. Reconfigure each additional server to use unique ports. The following example shows the steps for modifying SAServer2:

- 1 Copy the contents of SAServer1 to a new directory such as SAServer2. As shown in the previous figure, SAServer1 is used as a stand-alone server, and the additional servers are used in the cluster.
- 2 Decide which applications you want to cluster. Leave those EAR files in the `SAServer2/deploy_sas` directory and remove the other EAR files. For information about which applications to deploy on each server, see the following table:

Application	Target
<code>sas.bidashboard4.3.ear</code>	Target this EAR file to each server in the cluster if the <code>sas.eventsgenerationframework.war</code> file was removed from the EAR file. Otherwise, target this EAR file to the stand-alone server.
<code>sas.eventgenerationframework.war</code>	If this WAR file is extracted from <code>sas.bidashboard4.3.ear</code> , then deploy this WAR file to the stand-alone server.
<code>sas.biportlets4.3.ear</code>	Target all servers in the cluster.
<code>sas.flexthemes3.2.ear</code>	Do not target this application to any servers. Later tasks in this document describe how to deploy the Web application content to an HTTP server.
<code>sas.packageviewer4.3.ear</code>	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server. It might not receive as much use by business users as other Web applications.
<code>sas.portal4.3.ear</code>	Target all servers in the cluster.
<code>sas.storedprocess9.3.ear</code>	Target all servers in the cluster.
<code>sas.themedesigner3.2.ear</code>	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server because it is not used by business users.
<code>sas.themes.ear</code>	Do not target this application to any servers. Later tasks in this document describe how to deploy the Web application content to an HTTP server.
<code>sas.webdocmd9.3.ear</code>	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server because it is not used by business users.
<code>sas.webreportstudio4.3.ear</code>	Target all servers in the cluster.

Application	Target
sas.wip.admin9.3.ear	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server because it is not used by business users.
sas.wip.apps9.3.ear	Target all servers in the cluster.
sas.wip.resources9.3.ear	Target all servers in the cluster.
sas.wip.scs9.3.ear	Target all servers in the cluster.
sas.wip.services9.3.ear	Target all servers in the cluster.
sas.workflow9.3.ear	Target all servers in the cluster if you have performed the steps in <a href="#">"Reconfigure SAS Workflow" on page 9</a> . Otherwise, target the stand-alone server.

## Configure Additional JBoss Server Instances

### Configure Port Numbers

#### JBoss AS 4.2.3.GA and JBoss EAP 4.3

- 1 Edit the `SASServer2\conf\jboss-service.xml` file. Change all instances of `SASServer1` to `SASServer2` as shown in this example:

```
<!-- ===== -->
<!-- Service Binding -->
<mbean code="org.jboss.services.binding.ServiceBindingManager"
  name="jboss.system:service=ServiceBindingManager">
  <attribute name="ServerName">SASServer2</attribute>
  <!-- NOTE: Put the StoreURL attribute all on one line, it is -->
  <!-- on more than one line here due to space limitations. -->
  <attribute name="StoreURL">
    C:\jboss\server\SASServer2\conf\port-bindings.xml
  </attribute>
  <attribute name="StoreFactoryClassName">
    org.jboss.services.binding.XMLServicesStoreFactory
  </attribute>
</mbean>
```

- 2 For vertical cluster deployments, with multiple instances of JBoss on the same machine, edit the `SASServer2\conf\port-bindings.xml` file to avoid port conflicts. Change the instances of `ports-01` to `SASServer2`. This change sets `SASServer2` to use a unique set of ports. All the ports are incremented by 100 over the ports used by `SASServer1`. This typically results in `SASServer2` using port 8180 for HTTP. See this example:

```

<!-- ***** -->
<!-- * ports-01 Change to SAServer2 * -->
<!-- ***** -->
<server name="SAServer2"> <!-- was "ports-01" -->
  <!-- EJB3 Remoting Connector ... -->
  <service-config name="jboss...

```

When you add SAServer3 and SAServer4, use ports-02 and ports-03. If you need more than four servers, then you need to modify port-bindings.xml to add a new and unique port stanza.

## JBoss AS 5.1.0.GA

Edit the `SAServer2\conf\bindingservice.beans\META-INF\bindings-jboss-beans.xml` file and make the following changes:

- 1 Change the binding set from SAServer1 to SAServer2, and change the Ports01Bindings to SAServer2Bindings as shown in this example:

```

<bean class="org.jboss.services.binding.managed.ServiceBindingManagementObject"
  name="ServiceBindingManagementObject">

  <constructor>
    <!-- The name of the set of bindings to use for this server -->
    <parameter>${jboss.service.binding.set:SAServer2}</parameter>

    <!-- The binding sets -->
    <parameter>
      <set>
        <inject bean="SAServer1Bindings"/>
        <inject bean="PortsDefaultBindings"/>
        <inject bean="SAServer2Bindings"/>
        <inject bean="Ports02Bindings"/>
        <inject bean="Ports03Bindings"/>
      </set>
    </parameter>

```

- 2 Further down in the file, change the name of the original Ports01Bindings set to SAServer2Bindings:

```

<!-- The ports-01 bindings are obtained by taking the base
  bindings and adding 100 to each port value -->
<bean class="org.jboss.services.binding.impl.ServiceBindingSet"
  name="SAServer2Bindings">
  <constructor>
    <!-- The name of the set -->
    <parameter>SAServer2</parameter>
    <!-- Default host name -->
    <parameter>${jboss.bind.address}</parameter>
    <!-- The port offset -->
    <parameter>100</parameter>
    <!-- Set of bindings to which the "offset by X" approach
      can't be applied -->
    <parameter>
      <null/>
    </parameter>
  </constructor>
</bean>

```

**Note:** Following these instructions, SAServer2 will listen on port 8180 and use a set of unique ports. SAS provides two more sets of unique ports with the Ports02Bindings and the Ports03Bindings.

## Disable Local JMS Resources

### JBoss AS 4.2.3.GA

For all servers except the stand-alone server, prevent the clustered JBoss instances from creating JMS destinations by editing the `SAServer2\deploy\jms\sas-jms-service.xml` file. Comment out all the mbean definitions:

```
<!--
<mbean code="org.jboss.naming.LinkRefPairService"
      name="jboss.jms:alias=SASTopicConnectionFactory">
  <attribute name="JndiName">sas/jms/TopicConnectionFactory</attribute>
  <attribute name="RemoteJndiName">ConnectionFactory</attribute>
  <attribute name="LocalJndiName">java:/JmsXA</attribute>
  <depends>jboss:service=Naming</depends>
</mbean>
... -->
```

### JBoss EAP 4.3

- 1 Edit the `SAServer2\deploy\jboss-messaging.sar\destinations-service.xml` file. Remove or comment out all mbean definitions that include `sas/jms` in the JNDI name. For a SAS 9.3 Enterprise Business Intelligence Server installation, there are 11 mbeans.
- 2 For all servers except the stand-alone server, prevent the clustered JBoss instances from creating JMS connection factories by editing the `SAServer2\deploy\jboss-messaging.sar\connection-factories-service.xml` file. Comment out the two bindings that are related to SAS JMS resources:

```
<attribute name="JNDIBindings">
  <bindings>

    <!-- disable for clustered servers
    <binding>sas/jms/QueueConnectionFactory</binding>
    <binding>sas/jms/TopicConnectionFactory</binding>
    -->
    <binding>/ConnectionFactory</binding>
    <binding>/XAConnectionFactory</binding>
    <binding>java:/ConnectionFactory</binding>
    <binding>java:/XAConnectionFactory</binding>
  </bindings>
</attribute>
```

### JBoss AS 5.1.0.GA

- 1 Edit the `SAServer2\deploy\messaging\destinations-service.xml` file. Remove all mbean definitions that include `sas/jms` in the JNDI name. For a SAS 9.3 Enterprise Business Intelligence Server installation, there are 11 mbeans.

- 2 For all servers except the stand-alone server, prevent the clustered JBoss instances from creating JMS connection factories by editing the `SASServer2\deploy\messaging\connection-factories-service.xml` file. Comment out the two bindings that are related to SAS JMS resources:

```
<attribute name="JNDIBindings">
  <bindings>

    <!-- disable for clustered servers
    <binding>sas/jms/QueueConnectionFactory</binding>
    <binding>sas/jms/TopicConnectionFactory</binding>
    -->

    <binding>/ConnectionFactory</binding>
    <binding>/XAConnectionFactory</binding>
    <binding>java:/ConnectionFactory</binding>
    <binding>java:/XAConnectionFactory</binding>
  </bindings>
</attribute>
```

## Use JMS Resources on the Stand-Alone Server

This section applies to all versions of JBoss Application Server. Reconfigure the clustered Web application servers to use the JMS resources on SASServer1:

- 1 Edit the `SASServer2\conf\jboss-service.xml` file and add the following mbean:

```
<mbean code="org.jboss.naming.ExternalContext"
  name="jboss.jndi:service=ExternalContext,jndiName=SASServer1JNDI">
  <attribute name="JndiName">SASServer1JNDI</attribute>
  <attribute name="Properties">
    java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
    java.naming.provider.url=jnp://SASServer1-ip-address:1099
  </attribute>
  <attribute name="RemoteAccess">true</attribute>
  <depends>jboss:service=Naming</depends>
</mbean>
```

**Note:** SASServer1JNDI does not need to be the JNDI name. Just make sure that you use the same value in the next step.

- 2 For the following Web applications, change the `WEB-INF\jboss-web.xml` file. Replace all occurrences of `sas/jms` with `SASServer1JNDI/sas/jms` for all the `<jndi-name>` elements.

- `deploy_sas\sas.wip.services9.3.ear\sas.wip.services.war`
- `deploy_sas\sas.wip.services9.3.ear\sas.wipsoapservices.war`
- `deploy_sas\sas.workflow9.3.ear\sas.workflow.war`

**Note:** Modify `sas.workflow.war` only if you are using the Web application in the cluster.

The strategy described in this section leaves the stand-alone server as a single point of failure for JMS resources. If the deployment requires high availability, then consider the options described in [“Hot and Cold Standby Strategy” on page](#)

3. Another alternative is to deploy high-availability JMS resources. See JBoss documentation for information about JMS and high availability.

---

## Configure Start-Up Scripts

- 1 In the `JBOSS_HOME\bin` directory, copy `SASServer1.bat` to `SASServer2.bat`.
- 2 Edit `SASServer2.bat` and make the following changes:

- a Change all instances of `SASServer1` to `SASServer2`.

- b Change the value for the following JVM options:

```
-Dsas.auto.publish.port=8180
-Dsas.scs.port=
-Dsas.scs.host=
```

Set the `-Dsas.scs.port` and `-Dsas.scs.host` JVM options to use the SAS Content Server instance that you selected in [“Reconfigure the SAS Content Server Repository”](#) on page 7.

- c Add the following JVM options:

```
-Dsas.appserver.instance.id=SASServer2
-Dsas.wrs.keyUserActionLog.path=
c:\SAS\Config\Levl\Web\Logs\SASServer2
```

Also add this parameter to each server instance, using the appropriate server instance ID value. This value is used by SAS Content Server and in logging.

- d For UNIX deployments, change the port number that is used in the stop target. For `SASServer2`, the typical value is 1199:

```
stop)
# add args for username/password as in "SASServer2.sh stop -u username...
JAVA_OPTS="-d64 -Xms512M -Xmx512M"
"JBOSS_HOME/bin/shutdown.sh" -s localhost:1199 $* -S
```

- e Set the values for any other JVM parameters that are specific to your SAS middle-tier topology.

On Windows operating systems, the default configuration for JBoss is to run as a service. The service configuration is set in `JBOSS_HOME\server\SASServer2\wrapper.conf`. If you plan to run JBoss as a Windows service, then edit the `wrapper.conf` file and make all the changes that you set in the previous step. For example, change all instances of `SASServer1` to `SASServer2`, and change the value for the `-Dsas.auto.publish.port` parameter to the new HTTP port. On all the machines except the machine that is running SAS Remote Services, remove the `wrapper.ntservice.dependency` entries. After the edits are complete, create the service entry as shown in the following example:

```
SASServer2.bat -install
```

The outcome at this point is a series of JBoss servers that are configured to run on unique ports. At least one JBoss server instance is configured as a stand-alone server with the SAS BI Dashboard Event Generator and any applications that you prefer not to cluster. Some of the JBoss servers have identical deployments of the

SAS Web applications are configured to use the JMS resources on the stand-alone server.

---

## Configure a Load Balancing HTTP Server

HTTP requests must be distributed to the JBoss cluster members. This can be done with hardware or software. This document describes the steps that are used to configure Apache HTTP Server to load balance HTTP requests with mod\_jk. Mod\_jk also works with Microsoft Internet Information Systems (IIS). Follow these steps:

- 1 Download the mod\_jk binaries from <http://tomcat.apache.org/download-connectors.cgi>.
- 2 Rename the downloaded file to mod\_jk.so and move the file to **APACHE\_HOME\modules**.
- 3 Edit the **APACHE\_HOME\conf\httpd.conf** file, and add the mod\_jk configuration information as shown in the following example:

```
LoadModule jk_module modules/mod_jk.so

JkWorkersFile conf/workers.properties
JkMountFile conf/uriworkermap.properties
JkShmFile logs/jk.shm
JkLogFile logs/mod_jk.log
JkLogLevel error
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
```

```
<Location /jkstatus>
  JkMount status
  Order deny,allow
  Deny from all
  Allow from 127.0.0.1
</Location>
```

- 4 Create an **APACHE\_HOME\conf\workers.properties** file. The following example identifies SAServer1 as the stand-alone server and two servers for a cluster:

```
worker.list=loadbalancer,status
worker.list=SAServer1

# the stand-alone server has no load balancing settings
worker.SAServer1.host=hostname1.example.com
worker.SAServer1.port=8009
worker.SAServer1.type=ajp13

worker.SAServer2.host=hostname2.example.com
worker.SAServer2.port=8109
worker.SAServer2.type=ajp13
worker.SAServer2.lbfactor=1

worker.SAServer3.host=hostname2.example.com
worker.SAServer3.port=8209
```

```

worker.SASServer3.type=ajp13
worker.SASServer3.lbfactor=1

worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=SASServer2,SASServer3
worker.loadbalancer.sticky_session=True

worker.status.type=status

```

- 5** Create an `APACHE_HOME\conf\uriworkermap.properties` file. In conjunction with the `workers.properties` file, this file maps the location of the SAS Web applications to the JBoss cluster and the stand-alone server. Add the following information to the file:

```

# uncomment the following line if the event generator WAR
# was separated from the EAR and deployed on the stand-alone server
/SASBIDashboardEventGen*=SASServer1

# stand-alone server applications. These applications can be
# clustered, but they do not have a business user audience
/SASWebDoc*=SASServer1
/SASAdmin*=SASServer1
/SASPackageViewer*=SASServer1
/SASThemeDesignerForFlex*=SASServer1

# load-balanced applications that are clustered
/SASBIDashboard*=loadbalancer
/SASBIPortlets*=loadbalancer
/SASBIWS*=loadbalancer
/SASContentServer*=loadbalancer
/SASJSR168RemotePortlet*=loadbalancer
/SASLogon*=loadbalancer
/SASPortal*=loadbalancer
/SASPreferences*=loadbalancer
/SASStoredProcess*=loadbalancer
/SASSharedApps*=loadbalancer
/SASWIPClientAccess*=loadbalancer
/SASWIPServices*=loadbalancer
/SASWIPSoapServices*=loadbalancer
/sasweb*=loadbalancer
/SASWebReportStudio*=loadbalancer

# load-balance SAS Workflow if it is deployed to the cluster
# otherwise,
/SASWorkflowWebServices*=loadbalancer
/SASWorkflowServices*=loadbalancer

# if SAS Themes and SAS Flex Themes are not deployed
# statically, as described at the end of this document,
# then uncomment the following four entries
#/SASTheme_default/*=loadbalancer
#/SASFlexThemes/*=loadbalancer

```

---

## Configure JBoss to Use mod\_jk

At this stage of the reconfiguration, Apache HTTP Server is prepared to load balance HTTP requests. There is a stand-alone JBoss server that does not participate in clustering and at least two JBoss servers that are prepared to run in a cluster. This stage describes the steps that you must perform to configure each managed server to interface with the load-balancing HTTP server. Follow these steps for each JBoss server:

**1** Edit the server.xml file. Select the correct location:

- JBoss AS 4.2.3.GA and JBoss EAP 4.3

```
JBOSS_HOME\server\SASServern\deploy\jboss-web.deployer
```

- JBoss AS 5.1.0.GA

```
JBOSS_HOME\server\SASServern\deploy\jbossweb.sar
```

Make the following changes:

**a** Confirm that the AJP Connector is enabled:

```
<Connector address="{jboss.bind.address}"
  emptySessionPath="true"
  enableLookups="false" port="8009" protocol="AJP/1.3"
  redirectPort="8443"/>
```

**b** Add a jvmRoute attribute to the existing Engine element:

```
<Engine defaultHost="localhost" name="jboss.web"
  jvmRoute="SASServer1">
```

Use "SASServer2" when you edit the file for SASServer2. These jvmRoute values match the names and case that is used in the worker.properties and uriworkermap.properties files. If the case does not match, then session affinity does not work.

**2** (Optional) Enable logging of access requests by enabling the AccessLogValve in the server.xml file. You can disable this after confirming that your configuration is stable:

```
<!-- Access logger -->
<Valve className="org.apache.catalina.valves.AccessLogValve"
  prefix="localhost_access_log." suffix=".log"
  pattern="common" directory="{jboss.server.home.dir}/log"
  resolveHosts="false" />
```

---

## Perform Common Configuration Tasks

To complete JBoss clustering, perform the tasks in [“Common Tasks” on page 53](#).



## 4

# Oracle WebLogic Server 11g Clustering

<i>WebLogic Server Clustering Overview</i> .....	29
<i>Pack the Domain and Unpack on Additional Machines</i> .....	32
<i>Add Additional Machines</i> .....	32
<i>Clone the Managed Server</i> .....	33
<i>Create a Cluster and Assign the Servers</i> .....	34
<i>Target the Mail Session to the Cluster</i> .....	34
<i>Reconfigure the JMS Resources</i> .....	34
Strategy .....	34
Create a JMS Module .....	35
<i>Retarget the JDBC Data Source</i> .....	37
<i>Deploy the SAS Web Applications to the Cluster</i> .....	37
<i>Configure a Load Balancing HTTP Server</i> .....	39
<i>Perform Common Configuration Tasks</i> .....	41

## WebLogic Server Clustering Overview

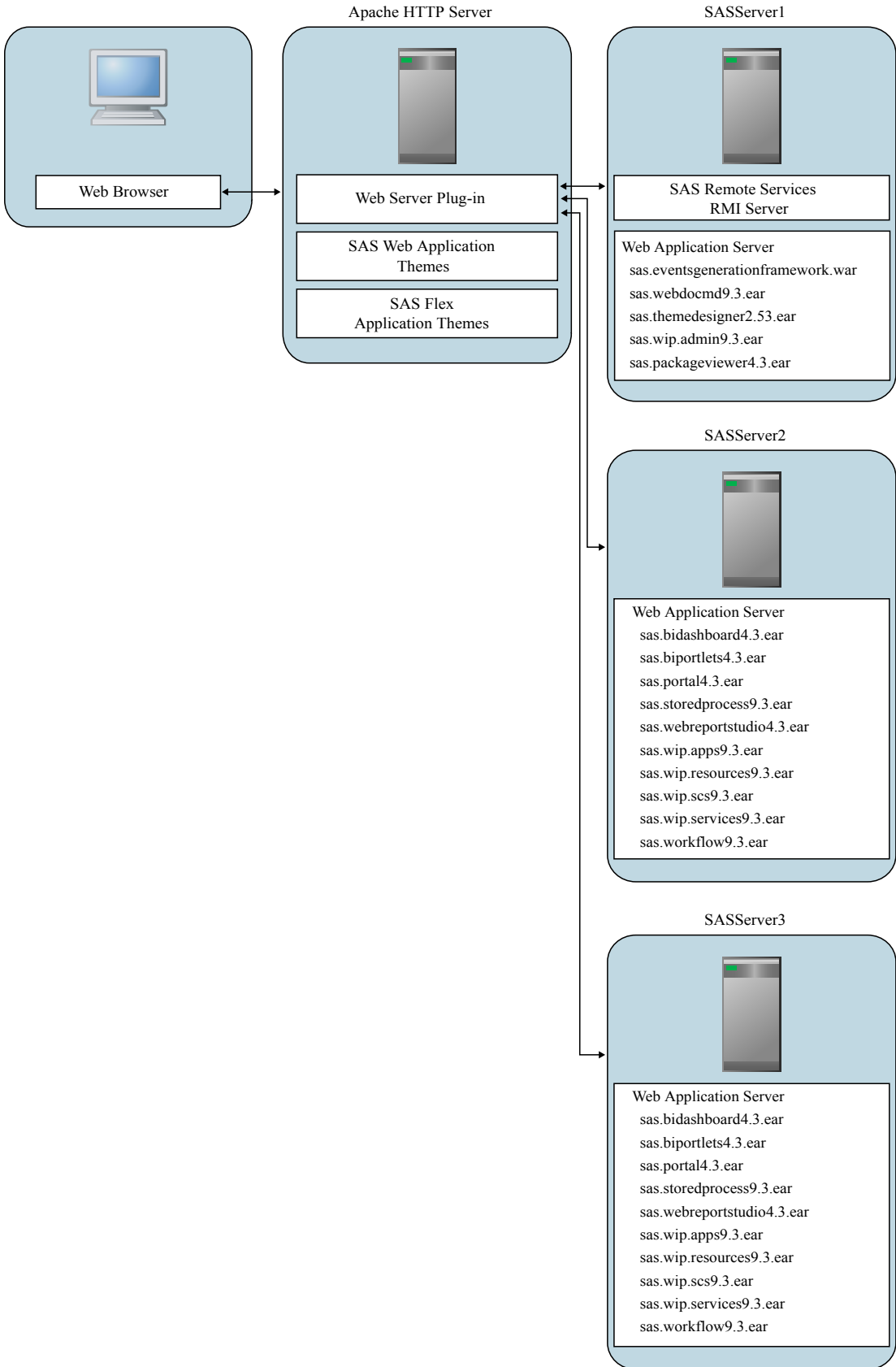
The starting point for this configuration is an Oracle WebLogic Server 10.3 domain, SASDomain, that was created by the SAS Deployment Wizard and is not in a clustered configuration. This section describes how to clone the managed server that was configured by the wizard, configure the resources needed by the SAS Web applications, and then create a cluster. The tasks in this section include steps that use the WebLogic Administration Console, but the same tasks can be completed with the WebLogic Scripting Tool (WLST). The following list identifies the high-level configuration steps:

- 1 Review the limitations in [“SAS Middle-Tier Limitations” on page 2](#) and perform the pre-configuration tasks in [“Common Pre-Configuration Tasks” on page 5](#).
- 2 Clone the existing server or configure a new server, and leave one server as a stand-alone server for the SAS Web applications that are not clustered.
- 3 Create the cluster and assign servers.
- 4 Reconfigure resources such as mail, JDBC, and JMS.

- 5** Deploy the SAS Web applications to the cluster and some applications to the stand-alone server.
- 6** Configure a load-balancing HTTP server.
- 7** Perform the common configuration tasks.

The tasks in the following sections describe how to create a topology that is similar to the example in the following figure.

Figure 4.1 Sample WebLogic Server Cluster Topology



---

## Pack the Domain and Unpack on Additional Machines

For deployments that use multiple machines in the cluster, the domain used for the SAS Web applications must be copied to each additional machine in the cluster. This topology is often called a horizontal cluster. The information in this section does not apply to deployments that use a vertical cluster topology—where multiple server instances are deployed on a single machine.

To pack the domain on the original machine and unpack the domain on an additional machine, follow these steps:

- 1 Navigate to the WL\_HOME bin directory, and use the pack command to pack SASDomain. The following example uses UNIX syntax:

```
cd /opt/Oracle/Middleware/wlserver_10.3/common/bin

./pack.sh -managed=true -domain=/opt/SAS/Config/Lev1/Web/SASDomain
-template=/export/home/sas/midtier_template.jar -template_name="Managed
Server SASDomain"
```

- 2 For each additional machine in the cluster, copy the midtier\_template.jar file to the additional machine, and unpack the domain template. The following example uses UNIX syntax:

```
cd /opt/WLS/wlserver_10.3/common/bin

./unpack.sh -domain=/opt/SAS/Config/Lev1/Web/SASDomain
-template=temporary-location/midtier_template.jar
```

- 3 Copy the WL\_HOME/common/nodemanager/nodemanager.domains file to the SASDomain/nodemanager directory:

```
cp /opt/WLS/wlserver_10.3/common/nodemanager/nodemanager.domains
/opt/SAS/Config/Lev1/Web/SASDomain/nodemanager
```

- 4 Start the node manager on each additional machine from the SASDomain/bin directory with the startNodeManager.sh command.

For information about the pack and unpack commands, see the Oracle WebLogic Server documentation.

---

## Add Additional Machines

If the SAS Web applications will be deployed to application server instances on additional machines, follow these steps to add a new machine in the administrative console:

- 1 Select **SASDomain** ► **Environment** ► **Machines** and then click **New**.
- 2 Enter the fully qualified domain name for the **Name** field, use the Machine OS menu to select the operating system type, and click **OK**.

- 3 Click the link for the newly added machine and then select **Node Manager**.
- 4 Enter the host name in the **Listen Address** field. Do not use the fully qualified domain name. Click **Save**.
- 5 Select the machine name and then click **Monitoring**. Confirm that the **Status** field indicates **Reachable**. Do not proceed unless the additional machines are reachable.

---

## Clone the Managed Server

The SAS Deployment Wizard can configure WebLogic Server and deploy the SAS Web applications to WebLogic Server. The wizard creates a server named SASServer1. The following steps describe how to clone SASServer1 to create managed servers. Later steps describe how to create a cluster from the managed servers. To clone SASServer1, follow these steps:

- 1 Use the Administration Console to shut down the managed server, SASServer1.
- 2 In the Domain Structure panel, select **Environment** ► **Servers**.
- 3 In the **Servers** table, select the check box for **SASServer1** and click **Clone**.
- 4 On the Clone a Server page, provide a unique name (**SASServer2**) and port number (**7101**) for the new server. One way to avoid port conflicts is to increment the port number by 100 for each additional server. However, do not use port 7501 because that port is used by the administrative server. Any port management strategy is acceptable so long as it avoids port conflicts. If the server will be deployed on another machine, enter the host name in the **Server Listen Address** field. Click **OK**.
- 5 For each additional server, select the server name (such as **SASServer2**). If the server will be deployed on another machine, use the **Machine** menu to select the machine.
- 6 Click **Server Start**.
- 7 In the **Arguments** text field, perform the following steps:

- a Change the values for the following JVM options:

```
-Dsas.server.name=SASServer2
-Dsas.auto.publish.port=7101
-Dsas.scs.port=
-Dsas.scs.host=
```

**Note:** Do not add these parameters. You must change the existing values for these parameters to the values that you specified in step 4.

Set the `-Dsas.scs.port` and `-Dsas.scs.host` JVM options to use the SAS Content Server instance that you selected in [“Reconfigure the SAS Content Server Repository”](#) on page 7.

- b Add the following JVM options:

```
-Dsas.appserver.instance.id=SASServer2  
-Dsas.wrs.keyUserActionLog.path=  
/opt/SAS/Config/Lev1/Web/Logs/SASServer2
```

Also add this parameter to each server instance, using the appropriate server instance ID value. This value is used in logging.

Click **Save**.

---

## Create a Cluster and Assign the Servers

Create a cluster from the newly cloned servers. Leave SASServer1 as the stand-alone server. Follow these steps:

- 1 In the Domain Structure panel, select **Environment** ► **Clusters** and then click **New**.
- 2 On the Create a New Cluster page, provide a name for the cluster such as **SASCluster1**. The other values can remain unchanged. Click **OK**.
- 3 Select **SASCluster1**.
- 4 Click **Servers** and then click **Add**.
- 5 On the Add a Server to Cluster page, select a cloned server from the list and then click **Next**.
- 6 Add additional servers to the cluster in the same way.

---

## Target the Mail Session to the Cluster

The SAS Mail Session must be targeted to all servers in the cluster and the stand-alone server. To target the SAS Mail Session, follow these steps:

- 1 In the Domain Structure panel, select **Services** ► **Mail Sessions**.
- 2 Select **SASMailSession**.
- 3 Click **Targets**. Select the check box for each server in the cluster and the stand-alone server.  
Click **Save**.

---

## Reconfigure the JMS Resources

### Strategy

The Events Generation Framework and other SAS Web applications use JMS resources. The SAS Deployment Wizard configured the JMS resources for the first

managed server, SAServer1. Several strategies are possible for providing scalable and highly available JMS resources. Consult the WebLogic Server documentation for alternatives to the strategy presented in this document.

The strategy described in this document is to leave the JMS Server associated with SAServer1 and to reference the JMS resources from the cluster. A JMS module and foreign JMS server are added to the cluster. The SAS Web applications can look up the JMS resources by JNDI name on the clustered servers and actually use the JMS resources on the stand-alone server.

## Create a JMS Module

### Create the JMS Module

- 1 In the Domain Structure panel, select **Services** ► **Messaging** ► **JMS Modules** and click **New**.
- 2 Enter a name such as `foreignJmsServerModule`.  
Click **Next**.
- 3 On the Targets page, select the **All servers in the cluster** check box.  
Click **Next**.
- 4 Select the **Would you like to add resources to this JMS system module?** check box.  
Click **Finish**.

### Configure the JMS Module

- 5 In the Summary of Resources table, click **New**.
- 6 Select the **Foreign Server** radio button.  
Click **Next**.
- 7 Enter a name like `SAServer1JMSServer`.  
Click **Finish**.

### Configure the Foreign Server

- 8 In the Summary of Resources table, click **SAServer1JMSServer**.
- 9 In the **JNDI Connection URL** field, enter `t3://fully-qualified-hostname.example.com:7001`. Use the host name and port number of the stand-alone server.  
Click **Save**.

### Create the Connection Factories

- 10 Select **Connection Factories**.  
Click **New**.
- 11 Set the following properties:

Name	Local JNDI Name	Remote JNDI Name
QueueConnectionFactory	sas/jms/ QueueConnectionFactory	sas/jms/ QueueConnectionFactory

Click **OK**.

**12** Click **New** and create another foreign connection factory:

Name	Local JNDI Name	Remote JNDI Name
TopicConnectionFactory	sas/jms/ TopicConnectionFactory	sas/jms/ TopicConnectionFactory

Click **OK**.

### *Create Destinations for Queues and Topics*

**13** Select **Destinations**.

Click **New** and create a foreign JMS destination:

Name	Local JNDI Name	Remote JNDI Name
AlertQueue	sas/jms/AlertQueue	sas/jms/AlertQueue

Click **OK**.

**14** Click **New** and create a foreign JMS destination for each of the destinations in the following table:

Name	Local JNDI Name	Remote JNDI Name
AuditQueue	sas/jms/AuditQueue	sas/jms/AuditQueue
WorkflowCommandQueue	sas/jms/ WorkflowCommandQueue	sas/jms/ WorkflowCommandQueue
WorkflowEventsQueue	sas/jms/ WorkflowEventsQueue	sas/jms/ WorkflowEventsQueue
WorkflowQueue	sas/jms/WorkflowQueue	sas/jms/WorkflowQueue
scheduler.pip.jobQueue	sas/jms/ scheduler.pip.jobQueue	sas/jms/ scheduler.pip.jobQueue
scheduler.pip.outputQueue	sas/jms/ scheduler.pip.outputQueue	sas/jms/ scheduler.pip.outputQueue
scheduler.pip.resultsQueue	sas/jms/ scheduler.pip.resultsQueue	sas/jms/ scheduler.pip.resultsQueue

Name	Local JNDI Name	Remote JNDI Name
<code>NotificationEventTopic</code>	<code>sas/jms/ NotificationEventTopic</code>	<code>sas/jms/ NotificationEventTopic</code>
<code>SASPublishedEventTopic</code>	<code>sas/jms/ SASPublishedEventTopic</code>	<code>sas/jms/ SASPublishedEventTopic</code>
<code>scheduler.pip.broadcastTopic</code>	<code>sas/jms/ scheduler.pip.broadcas tTopic</code>	<code>sas/jms/ scheduler.pip.broadcas tTopic</code>

## Retarget the JDBC Data Source

The SAS Web Infrastructure Platform uses JDBC to provide the SharedServices data source. The SharedServices data source must be configured for all the servers in the cluster and the stand-alone server. Follow these steps:

- 1 In the Domain Structure panel, select **Services** ► **JDBC** ► **Data Sources**.
- 2 Select **SharedServices**.
- 3 Click **Targets** and then select the check box for the stand-alone server and each server in the cluster.  
Click **Save**.

## Deploy the SAS Web Applications to the Cluster

All the SAS Web applications were deployed to SASServer1. To have the applications available from the cluster and some applications available from the stand-alone server, the applications must be targeted to the correct servers. Follow these steps:

- 1 In the Domain Structure panel, click **Deployments**.
- 2 For each SAS Web application, perform the following steps:
  - a Click **Targets**.
  - b Select the check box for the top-level EAR file and click **Change Targets**.
  - c Select the check box for the target server. For applications that can be clustered, select the check box for each server in the cluster.
Click **Save**.

Repeat the previous steps for each EAR file. For information about which applications to deploy on each server, see the following table:

Application	Target
sas.bidashboard4.3.ear	<p>Target this EAR file to each server in the cluster if the sas.eventsgenerationframework.war file was removed from the EAR file. Otherwise, target this EAR file to the stand-alone server.</p> <p><i>Note:</i> If this application is targeted to the cluster, then make sure that the weblogic.xml file is updated with a cookie name as described in <a href="#">“Event Generation Framework” on page 5</a>.</p>
sas.eventgenerationframework.war	If this WAR file is extracted from sas.bidashboard4.3.ear, then deploy this WAR file to the stand-alone server.
sas.biportlets4.3.ear	Target all servers in the cluster.
sas.flexthemes3.2.ear	Do not target this application to any servers. Later tasks in this document describe how to deploy the Web application content to an HTTP server.
sas.packageviewer4.3.ear	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server. It might not receive as much use by business users as other Web applications.
sas.portal4.3.ear	Target all servers in the cluster.
sas.storedprocess9.3.ear	Target all servers in the cluster.
sas.themedesigner3.2.ear	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server because it is not used by business users.
sas.themes.ear	Do not target this application to any servers. Later tasks in this document describe how to deploy the Web application content to an HTTP server.
sas.webdocmd9.3.ear	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server because it is not used by business users.
sas.webreportstudio4.3.ear	Target all servers in the cluster.
sas.wip.admin9.3.ear	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server because it is not used by business users.
sas.wip.apps9.3.ear	Target all servers in the cluster.
sas.wip.resources9.3.ear	Target all servers in the cluster.
sas.wip.scs9.3.ear	Target all servers in the cluster.
sas.wip.services9.3.ear	Target all servers in the cluster.
sas.workflow9.3.ear	Target all servers in the cluster if you have performed the steps in <a href="#">“Reconfigure SAS Workflow” on page 9</a> . Otherwise, target the stand-alone server.

## Configure a Load Balancing HTTP Server

There are several ways to use hardware or software for distributing HTTP requests to the servers in the cluster. This section describes how to use Apache HTTP Server. The information in this section supplements the WebLogic Server documentation that is provided in [Using Web Server Plug-Ins with WebLogic Server](#). To configure Apache HTTP Server with the WebLogic Server plug-in, follow these steps:

- 1 Locate the correct plug-in for your operating system and machine architecture from the directories in `WL_HOME/server/plugin`. For Apache HTTP Server 2.2, the `mod_wl_22.so` file provides regular encryption. The `mod_wl128_22.so` file provides 128-bit encryption.
- 2 Copy the plug-in to the `APACHE_HOME/modules` directory.
- 3 Edit the `APACHE_HOME/conf/httpd.conf` file so that Apache HTTP Server:
  - loads the WebLogic Server plug-in
  - configures the plug-in with information about the WebLogic Server
  - uses the plug-in for the SAS Web applications

Add the `LoadModule`, `IfModule`, and `Include` directives to the `httpd.conf` file as shown in the following example:

```
LoadModule weblogic_module modules/mod_wl_22.so
<IfModule mod_weblogic.c>
    Include conf/weblogic.conf
</IfModule>
```

- 4 Create the `APACHE_HOME/conf/weblogic.conf` file as shown in the following example:

```
WLogFile "/opt/apache2/logs/wlproxy.log"
KeepAliveEnabled ON
KeepAliveSecs 120
Idempotent OFF
WLIOTimeoutSecs 600
#Debug ALL
#DebugConfigInfo ON

# sasserver1 is a stand-alone server
# sasserver2 and sasserver3 are cluster members

# stand-alone server applications
MatchExpression /SASBIDashboardEventGen
WebLogicHost=sasserver1.example.com|WebLogicPort=7001|
SetHandler=weblogic-handler

# these applications can be clustered, but do not have
# a business user audience

MatchExpression /SASAdmin
WebLogicHost=sasserver1.example.com|WebLogicPort=7001
```

```

MatchExpression /SASThemeDesignerForFlex
WebLogicHost=sasserver1.example.com|WebLogicPort=7001

MatchExpression /SASWebDoc
WebLogicHost=sasserver1.example.com|WebLogicPort=7001

MatchExpression /SASPackageViewer
WebLogicHost=sasserver1.example.com|WebLogicPort=7001

# load-balanced applications that are clustered
MatchExpression /SASContentServer
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASContentServer.sessionID

# the following two use SASPortal.sessionID
MatchExpression /SASBIPortlets
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASPortal.sessionID

MatchExpression /SASJSR168RemotePortlet
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASPortal.sessionID

MatchExpression /SASWorkflowServices
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASWorkflowServices.sessionID

MatchExpression /SASWorkflowWebServices
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASWorkflowWebServices.sessionID

MatchExpression /SASWIPServices
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASWIPServices.sessionID

MatchExpression /SASWIPSoapServices
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASWIPSoapServices.sessionID

MatchExpression /SASWIPClientAccess
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASWIPClientAccess.sessionID

MatchExpression /SASBIWS
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASBIWS.sessionID

MatchExpression /SASWebReportStudio
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASWebReportStudio.sessionID

MatchExpression /SASStoredProcess
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASStoredProcess.sessionID

```

```

MatchExpression /SASBIDashboard
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASBIDashboard.sessionID

MatchExpression /SASLogon
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASLogon.sessionID

MatchExpression /SASPreferences
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASPreferences.sessionID

MatchExpression /SASSharedApps
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASSharedApps.sessionID

MatchExpression /SASPortal
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201|WLCookieName=SASPortal.sessionID

MatchExpression /sasweb
WebLogicCluster=sasserver2.example.com:7101,
sasserver3.example.com:7201

# if SAS Themes and SAS Flex Themes are not deployed
# statically, as described in later steps, then
# uncomment the following two entries
#MatchExpression /SASTheme_default
#WebLogicCluster=sasserver2.example.com:7101,
#sasserver3.example.com:7201

#MatchExpression /SASFlexThemes
#WebLogicCluster=sasserver2.example.com:7101,
#sasserver3.example.com:7201

```

---

## Perform Common Configuration Tasks

To complete WebLogic Server clustering, perform the tasks in [“Common Tasks” on page 53](#).



# 5

## IBM WebSphere Application Server 7.0 Clustering

<i>WebSphere Application Server Clustering Overview</i> .....	43
<i>Add Additional Machines</i> .....	46
<i>Create a Server Template from SASServer1</i> .....	46
<i>Create the Cluster and Add Servers</i> .....	46
<i>Configure JVM Options</i> .....	47
<i>Configure the Cookie Name</i> .....	47
<i>Target SAS Web Applications to the Cluster</i> .....	48
<i>Add SASCluster as a SAS Messaging Bus Member</i> .....	49
<i>Configure a Load Balancing HTTP Server</i> .....	51
<i>Perform Common Configuration Tasks</i> .....	51

---

### WebSphere Application Server Clustering Overview

IBM WebSphere Application Server can be clustered by using the administrative console or by scripting with the wsadmin tool. The starting point for this configuration is a WebSphere Application Server that has been configured by the SAS Deployment Wizard. The wizard configures WebSphere Application Server to provide the resources needed by the SAS Web applications such as mail, JDBC, and JMS. The wizard can also deploy the SAS Web applications to a server instance. The following list identifies the high-level steps for deploying the SAS Web applications on a WebSphere Application Server cluster:

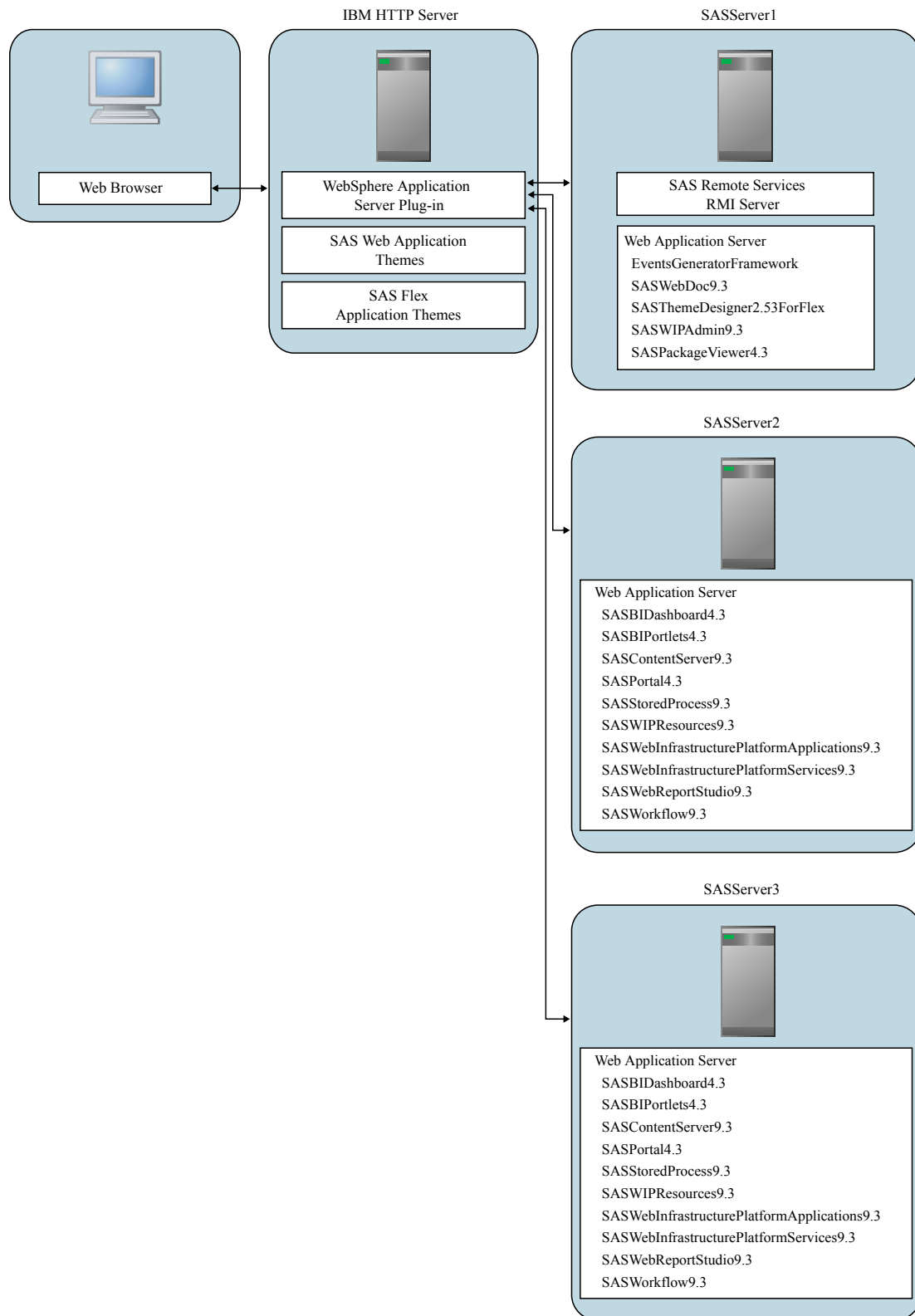
- 1 Review the limitations in [“SAS Middle-Tier Limitations” on page 2](#) and perform the pre-configuration tasks in [“Common Pre-Configuration Tasks” on page 5](#).
- 2 Create a server template from the application server that was configured by the wizard.
- 3 Create a cluster and populate it with servers created with the server template.
- 4 Configure JVM options for the clustered servers and the stand-alone server.
- 5 Deploy the SAS Web applications to the cluster.

- 6** Configure a load-balancing HTTP server.
- 7** Perform the common configuration tasks.

When configuring a cluster of WebSphere Application Servers, do not enable HTTP memory-to-memory replication. The SAS Web applications can access large data structures such as data sets. Memory-to-memory replication would reduce performance.

The tasks in the following sections describe how to create a topology that is similar to the example in the following figure.

Figure 5.1 Sample WebSphere Application Server Cluster Topology



---

## Add Additional Machines

If the SAS Web applications will be deployed to application server instances on additional machines, then see WebSphere Application Server documentation for information about adding nodes to the cell.

---

## Create a Server Template from SASServer1

When the SAS Deployment Wizard is used to configure WebSphere Application Server, an application server, SASServer1, is configured with all the resources that are needed by the SAS Web applications. This task describes how to use SASServer1 as a server template to reduce the effort that is required to configure additional servers for the SAS Web applications and the cluster. Follow these steps:

- 1 Select **Servers** ► **Server Types** ► **WebSphere application servers**. Click **Templates** and then click **New**.
- 2 Select the radio button for SASServer1 and click **OK**.
- 3 Provide a name, such as **SASServerTemplate**, and a description. Click **OK**.

---

## Create the Cluster and Add Servers

To create a cluster and add servers to it, follow these steps:

- 1 Select **Servers** ► **Clusters** ► **WebSphere application server clusters** and click **New**.
- 2 Provide a name, such as **SASCluster**, and make sure that the **Prefer local** and the **HTTP session memory-to-memory replication** check boxes are cleared. Click **Next**.

**Note:** SAS Web applications do not use Enterprise Java Beans, and do not support session-level failover.

- 3 On the **Create first cluster member** page, select **Create the member using an application server template**, and then select SASServerTemplate. Click **Next**.
- 4 You can add server instances to the cluster. To add a server, provide the member name, and make sure that **Generic Unique HTTP Ports** is selected. Click **Add Member**.
- 5 Click **Finish** and then save the changes to the master configuration.

---

## Configure JVM Options

Each server instance inherits the JVM configuration from the SASSTemplate. Some JVM options must be modified for each server instance. The `sas.auto.publish.port` JVM option must be modified to match the HTTP port for the server instance. Follow these steps:

- 1 Select **WebSphere application servers**, and then click the link for the server instance.
- 2 Click **Ports** and make a note of the value for `WC_defaulthost`.
- 3 Select **Java and Process Management** ▶ **Process Definition**.
- 4 Select **Java Virtual Machine** from the right panel of the Web page.
- 5 In the **Generic JVM arguments** field, change the following values to the value for `WC_defaulthost`:
  - `-Dsas.auto.publish.port`
  - `-Dsas.scs.port`
  - `-Dsas.scs.host`

Set the `-Dsas.scs.port` and `-Dsas.scs.host` JVM options to use the SAS Content Server instance that you selected in [“Reconfigure the SAS Content Server Repository”](#) on page 7.

- 6 Add the following JVM options:

```
-Dsas.appserver.instance.id=SASServer2
-Dsas.wrs.keyUserActionLog.path=
/opt/SAS/Config/Lev1/Web/Logs/SASServer2
```

Also add this parameter to each server instance, using the appropriate server instance ID value. This value is used by SAS Content Server and in logging.

- 7 Click **OK** and then save the changes to the master configuration.

---

## Configure the Cookie Name

The SAS Information Delivery Portal, SAS BI Portlets, and SAS JSR168 Remote Portlets Web applications use the cookie name `JSESSIONIDSASServer1`. This is the default cookie name that the SAS Deployment Wizard configures for `SASServer1`. In order to provide session affinity for the SAS Web applications, this cookie name must not be changed for the Web application server instances that serve those three Web applications. However, for the stand-alone server, you must change the cookie name to a different value to preserve session affinity on the stand-alone server. SAS recommends changing the value to `SASStandAlone`. Follow these steps for the stand-alone server instance:

- 1 Select **WebSphere application servers**, and then click `SASServer1`.

- 2 Select **Web Container Settings** ► **Web container**.
- 3 Select **Session management** from the right panel of the Web page.
- 4 Make sure the check box for **Enable cookies** is selected, and then click the **Enable cookies** link.
- 5 Change the value for Cookie name to **SASstandAlone**.
- 6 Click **OK** and then save the changes to the master configuration.

## Target SAS Web Applications to the Cluster

Because the starting configuration for WebSphere Application Server was a single server, SASServer1, all the SAS Web applications are targeted to that server. Now that the cluster is configured, some of the SAS Web applications need to be reconfigured to use the stand-alone server and the others to use the cluster. Target SAS BI Dashboard Event Generator to the stand-alone server. To target each SAS Web application, follow these steps:

- 1 Select **Applications** ► **Application Types** ► **WebSphere enterprise applications**, and then click the respective SAS Web application. For information about which applications to deploy on each server, see the following table:

Application	Target
SASBIDashboard4.3	Target this EAR file to each server in the cluster if the <code>sas.eventsgenerationframework.war</code> file was removed from the EAR file. Otherwise, target this EAR file to the stand-alone server.
<code>sas.eventsgenerationframework.war</code>	If this WAR file is extracted from <code>sas.bidashboard4.3.ear</code> , then deploy this WAR file to the stand-alone server.
SASBIPortlets4.3	Target all servers in the cluster.
SASFlexThemes3.2	Do not target this application to any servers. Later tasks in this document describe how to deploy the Web application content to an HTTP server.
SASPackageViewer4.3	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server. It might not receive as much use by business users as other Web applications.
SASPortal4.3	Target all servers in the cluster.
SASStoredProcess9.3	Target all servers in the cluster.
SASThemeDesigner3.2ForFlex	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server because it is not used by business users.

Application	Target
SASThemes9.3	Do not target this application to any servers. Later tasks in this document describe how to deploy the Web application content to an HTTP server.
SASWebDoc9.3	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server because it is not used by business users.
SASWebReportStudio4.3	Target all servers in the cluster.
SASWIPAdmin9.3	This Web application can be clustered, but you might prefer to deploy it to the stand-alone server because it is not used by business users.
SASWebInfrastructurePlatformApplications 9.3	Target all servers in the cluster.
SASWIPResources9.3	Target all servers in the cluster.
SASContentServer9.3	Target all servers in the cluster.
SASWebInfrastructurePlatformServices9.3	Target all servers in the cluster.
SASWorkflow9.3	Target all servers in the cluster if you have performed the steps in <a href="#">"Reconfigure SAS Workflow" on page 9</a> . Otherwise, target the stand-alone server.

- 2 Select **Manage Modules**, and then select each check box for all the modules.
- 3 From the **Clusters and servers** field, select the cluster or server instance, and then click **Apply**. The Server field for each module is updated with your choices.
- 4 Click **OK** to target the application to the cluster or server.
- 5 Repeat these steps for each SAS Web application to target the cluster or stand-alone server.
- 6 Click **Save** to save the changes to the master configuration.

---

## Add SASCluster as a SAS Messaging Bus Member

The Events Generation Framework and SAS Web Infrastructure Platform need access to JMS resources. Adding the cluster as a member of the SAS Messaging Bus can provide high availability. The following steps describe how to add the cluster as a bus member, with a single messaging engine.

One of the requirements for high availability is that the message store that is used by the messaging engine must be accessible by all the servers in the cluster. A JDBC data source can be used as a message store, or a file system location on network storage can be used as the message store. Before you begin this task, make sure

that either a JDBC data source has been configured in the administration console, or know the network storage location that will be used.

To reconfigure the SAS Messaging Bus and add SASCluster as a member, follow these steps:

#### *Add SASCluster as a Bus Member*

- 1** Select **Service integration** ► **Buses** ► **SAS Messaging Bus**, and then select **Bus members**.
- 2** Click **Add**, select **Cluster**, and select the cluster name from the menu. Click **Next**.
- 3** Clear the check box for **Enable messaging engine policy assistance?** and click **Next**.
- 4** Choose either **File store** or **Data store** and click **Next**.
- 5** For deployments that use a network storage location for a file store, provide the path to the directory for log files and the directory for the message store. For deployments that use a JDBC data source as the data store, provide the JNDI name of the data source. Click **Next**. For more information about configuring a message store, see the WebSphere Application Server help.

#### *Create and Use Destinations on the Bus*

These steps describe creating new destinations on the service integration bus for the queues and topics that are used by the SAS Web applications. The new destinations are assigned to use the cluster as the bus member.

- 6** Select **Resources** ► **JMS** ► **Queues**. For each queue that is listed, follow these steps to reconnect the queue to the SAS Messaging Bus:
  - a** Click the queue name.
  - b** In the Administration area, copy the queue name, such as `AlertQueue`, to your clipboard.
  - c** Select **Create Service Integration Bus destination** from the **Queue name** menu.
  - d** Paste the queue name in the **Identifier** text field, add "SIB" to the end of the name, and click **Next**.
  - e** Select the cluster name from the **Bus member** menu and click **Next**. Click **Finish**.
- 6** Select **Service integration** ► **Buses** ► **SAS Messaging Bus**, and then select **Destinations**.
- 7** Delete the queue destinations that were originally used by the SAS Web applications. These destinations do not have names with the "QueueSIB" suffix.

---

## Configure a Load Balancing HTTP Server

Load balancing the HTTP requests to the application server instances can be accomplished in many ways, using hardware or software. One option for load balancing HTTP requests is to configure IBM HTTP Server for load balancing. For more information about performing this configuration, see *Configuring IBM HTTP Server as a Reverse Proxy Server for SAS 9.3 Web Applications Deployed on IBM WebSphere Application Server* or IBM documentation.

WebSphere Application Server settings must be changed so that SAS Content Server is available from IBM HTTP Server. These changes enable IBM HTTP Server to accept content, such as attachments, with HTTP requests. Follow these steps in the WebSphere administrative console:

- 1 Change the virtual host for the SAS Content Server application to the default host.
- 2 Select **Servers** ▶ **Server Types** ▶ **Web servers** ▶ **webserver1** ▶ **Plug-in properties**.
- 3 Select the **Accept content for all requests** check box
- 4 Click **OK** and then save the changes to the master configuration.

After you configure the HTTP server, regenerate the Web server plug-in configuration file and propagate it.

---

## Perform Common Configuration Tasks

To complete WebSphere Application Server clustering, perform the tasks in [“Common Tasks” on page 53](#).



## 6

## Common Tasks

<i>Confirm the SAS Web Report Studio Clustering Setting</i> .....	53
<i>Configure Web Report Studio General Purpose Log Location</i> .....	53
<i>Deploy SAS Themes and SAS Themes for Flex Application to an HTTP Server</i> .....	54
<i>Change the Connections for the SAS Web Applications</i> .....	54
<i>Change the Connection for SAS Content Server</i> .....	55
<i>Change the WebDAV Repository URL</i> .....	55
<i>Disable the Redirection Filter for the SAS Web Applications</i> .....	56
<i>Start Software in the Correct Sequence</i> .....	57

---

### Confirm the SAS Web Report Studio Clustering Setting

Perform the following steps to confirm that SAS Web Report Studio is configured to support in-process scheduling in a clustered Web application server environment. This setting is typically performed during installation with the SAS Deployment Wizard, but if it was not set, correct the setting:

- 1 Use SAS Management Console and select **Application Management** ► **Configuration Manager**.
- 2 Right-click **Web Report Studio 4.3** and select **Properties**.
- 3 Check the value for **Web Container is Clustered**. If it is not set to Yes, then select **Yes** from the menu and click **OK**.

---

### Configure Web Report Studio General Purpose Log Location

- 1 In SAS Management Console, select **Environment Management** ► **Foundation Services Manager**.

- 2 Select **SASWebReportStudio4.3 Local Services**, and then select **Core ▶ Logging Service**.
- 3 Right-click **Logging Service** and select **Properties**.
- 4 Select the **Service Configuration** tab and then click **Configuration**.
- 5 Select the **Outputs** tab.
- 6 Select **SAS\_LS\_FILE**, and then click **Edit**.
- 7 Change the File parameter to use the `sas.appserver.instance.id` identifier. See the following example:

```
/opt/SAS/Config/Lev1/Web/Logs/${sas.appserver.instance.id}/SASWebReportStudio4.3.log
```

- 8 Click **OK** on each of the dialog boxes to return to SAS Management Console.

---

## Deploy SAS Themes and SAS Themes for Flex Application to an HTTP Server

In all the topologies described in this document, SAS recommends deploying the SAS Themes content and Flex Application Themes content to an HTTP server. By serving the content from an HTTP server, you shift the processing load of serving static files from the Web application server to the HTTP server. This task describes how to perform this deployment on Apache HTTP Server. Performing this reconfiguration on other HTTP servers is similar. To configure Apache HTTP Server to serve the content for SAS Themes and Flex Application Themes, follow these steps:

- 1 Create a new directory that is named `APACHE_HOME/htdocs/SASTheme_default`.
- 2 Copy the `SAS-config-dir/Lev1/Web/Staging/exploded/sas.themes.ear/sas.theme.default.war` directory into the `APACHE_HOME/htdocs/SASTheme_default` directory.
- 3 Create a new directory that is named `APACHE_HOME/htdocs/SASFlexThemes`.
- 4 Copy the `SAS-config-dir/Lev1/Web/Staging/exploded/sas.flexthemes3.2.ear/sas.flexthemes3.2.war` directory into the `APACHE_HOME/htdocs/SASFlexThemes` directory.

---

## Change the Connections for the SAS Web Applications

After SAS Themes and Flex Application Themes are deployed to the HTTP server and the SAS Web applications are distributed to different servers, information about how to access the applications such as host and port must be updated in SAS

metadata. Change the connection information to a URL that includes the load-balancing HTTP server host name and port.

To change the connection access point, follow these steps in SAS Management Console.

- 1 Select **Application Management** ► **Configuration Manager**.
- 2 Right-click on the SAS Web application that you want to reconfigure, and select **Properties**.
- 3 Click the **Connection** tab, set **Host Name** and **Port Number** to the host name and port number of the load-balancing HTTP server, and then click **OK**.

---

## Change the Connection for SAS Content Server

Reconfigure SAS metadata for the SAS Content Server connection to the load-balancing HTTP server host name and port. This change is similar to the change required for each of the SAS Web applications.

To reconfigure the host name and port of the SAS Content Server in SAS metadata, follow these steps in SAS Management Console.

- 1 Select **Server Manager** ► **SAS Content Server**.
- 2 Right-click the **Connection: SAS Content Server** icon in the right panel and select **Properties**.
- 3 Click the **Options** tab and set the **Host name** and **Port number** fields to the host name and port number of the load-balancing HTTP server. Click **OK**.
- 4 Select the **Folders** tab.
- 5 Right-click the **SAS Folders** icon at the root of the folder tree in the left pane and select **Properties**.
- 6 Click the **Content Mapping** tab and use the **Server** menu to select **SAS Content Server**. The URL field then shows the load-balancing HTTP server host name and port.
- 7 Click **OK**. Confirm the Change Content Mapping dialog box.

---

## Change the WebDAV Repository URL

Just as in the previous step, because SAS Content Server is accessed through the load-balancing HTTP server, you must reconfigure SAS metadata with the connection information for the WebDAV repository. The following applications use SAS metadata to identify the connection information for the WebDAV repository provided by SAS Content Server:

- Remote Services

- SASBIPortlets4.3 Local Services
- SASJSR168ReportPortlet4.3 Local Services
- SASPackageViewer4.3 Local Services
- SASPortal4.3 Local Services
- SASStoredProcess9.3 Local Services
- SASWebReportStudio4.3 Local Services

To reconfigure the WebDAV URL for the applications, follow these steps in SAS Management Console.

- 1 Select **Environment Management** ► **Foundation Services Manager**.
- 2 Select the application, and then select **Core** ► **Information Service**.
- 3 Right-click **Information Service** and select **Properties**.
- 4 On the Information Service Properties dialog box, click the **Service Configuration** tab, and then click **Configuration**.
- 5 On the Information Service Configuration dialog box, click the **Repositories** tab.
- 6 Select **WebDAV**, and then click **Edit**.
- 7 Change the **Host** and **Port** values to the host name and port of the load-balancing HTTP server.
- 8 Click **OK** to close the Information Service Configuration dialog box.
- 9 Click **OK** to close the Information Service Properties dialog box.
- 10 Restart SAS Remote Services and the Web application servers that are hosting the SAS Content Server application.

---

## Disable the Redirection Filter for the SAS Web Applications

By default, the SAS Web applications use a special redirection filter. When used with an HTTP server, this filter must be disabled. Start SAS Management Console, and then follow these steps.

- 1 Select **Application Management** ► **Configuration Manager**.
- 2 Right-click **SAS Application Infrastructure**, and select **Properties**.
- 3 Select **Advanced**, and then click **Add**.
- 4 Enter a property name of `App.RedirectionFilterDisabled` and a value of `true`.

---

## Start Software in the Correct Sequence

Follow these steps to start software in the correct sequence:

- 1** Start SAS Remote Services.
- 2** Start the load-balancing HTTP server.
- 3** Start the Web application servers.
  - For JBoss and WebSphere Application Server, start the stand-alone server instance and then the clustered servers.
  - For WebLogic Server, start the clustered servers and then the stand-alone server.

