Java Family of Products
Installation Guide

Table of Contents

1. Introduction

The purpose of this document is to enable customers and their IT teams install Adserversolutions.Java products. This guide covers:

1. Technologies and versions supported
2. Standard deployment on shared and dedicated hosting
3. Enterprise deployment on virtual dedicated and dedicated hosting

The guide also covers how to install Geo database. Customers who do not wish to purchase Geo database may opt for our Geo web service where their systems can connect to our central Geo database for Geo targeting.

2. Technologies & Versions Supported

Operating Systems	Any platform
Scripting	JSP, server side includes, Servlets, Beans
Container	Any container with J2EE support, most popular being Tomcat
Databases	MySQL
Hosting	Shared, virtual dedicated, dedicated
System Architecture	Multiple server support.
SMTP Server	Will work with any SMTP server.
Third party components	Graphical components, pre-packaged
Minimum space requirements	500 mb MSSQL space, 1GB if using Geo database.
Other requirements	SMTP server, outbound webservices calls (if using geo webservice),

3. Packages Supplied

adserversolutions_java_dep_x.y.z.zip	This package contains files and components for both standard and enterprise versions. This is a deployment package for live installations. X.y.x represents version number.
adserversolutions_java_dev_x.y.z.zip	This package contains source associated with the deployment package. This package is not for installation and is only for customers who have purchased source and who wish to extend the product. Source code includes code for core manager programs and APIs for business components. X.y.z represents version number.

4. Standard Deployment

Standard version of Adserversolutions product can be deployed on shared and dedicated servers. Given below are step by step instructions on how to deploy the application. Steps mentioned apply to both shared and dedicated servers, unless otherwise stated.

Step 1	Install Adserver Package	Unzip the supplied package. Upload using FTP or copy the contents into your project directory. Use binary mode for FTP. Ensure that project directory has write permission. Also ensure that after you right click on directory and click on properties and then security tab, Users on the machine must have full control to the Adserver project directory. For linux users, use chmod 777. In tomcat, you could put your adserversolutions directory under webapps or in an external directory that can be pointed to from tomcat.
Step 2	Web Server	You have an option to use standard web servers (apache or IIS) or you can use tomcat's inbuilt web server. If using standard web servers, connectors will have to be installed. Please contact your system administrator for web server and container installation (tomcat).
Step 3	Verify web server set up	We recommend that you test your IIS or apache/tomcat installation by running sample html, aspx, jsp and servlets.
Step 4	Database Setup	Create a database on your SQL server installation. E.g. adserversolutions. Following scripts have been provided in Database directory of the package: 1. mysql_create_table.sql 2. mysql_create_sp.sql 3. mysql_create_data.sql Run each script in the order given above on the database you created. We recommend that you use MySQL UI tool or command line. If you use any other tool to connect to MySQL, these scripts may not work. For command line on Linux, use SOURCE command after selecting the database to work with (use <yourdb>). As per your organizations security policy, create a SQL userid. Ensure that this sql id has all permissions to the database. Our supplied scripts use root as default user. To install Geo database, please see instructions in Section named "Geo Database Installation".
Step 4	Adserver Configuration	Following files will need to be changed manually: 1. adserversolutions\WEB-INF\web.xml (see Appendix A) 2. adserversolutions\META-INF\context.xml (see Appendix B)

		Then copy the following files into appropriate locations: 1. copy naming-factory-dbcp.jar from adserversolutions/web-inf/lib to $CATALINA_HOME/common/lib 2. copy mysql-connector-java-3.1.14-bin.jar from adserversolutions\WEB-INF\lib to $CATALINA_HOME/common/lib
Step 5	Activate Product	Open link http://www.myadserver.com/activate.jsp. This will open a page that will be used to activate the product. Enter the license key provided. Upon activation, you will be taken to the login page. If you do not see anything in module dropdown, it means activation was not successful. Please open Logs/log.txt to analyze the issue and report it to our product support team.
Step 6	Sign On	Open link http://www.myadserver.com/index.jsp. This will open the sign on page. Following ids will be used based on product purchased: • Userid: netadmin, password: temp This id will be used if you have purchased an adnetwork product. • Userid: pubadmin, password: temp for default publisher id for admanagement module. • Userid: mercadmin, password: temp for default merchant • Userid: affadmin, password temp for default affiliate admin. Note that modules you will be able to access will depend on the license key so these ids will be applicable only to specific modules.

Special Notes:

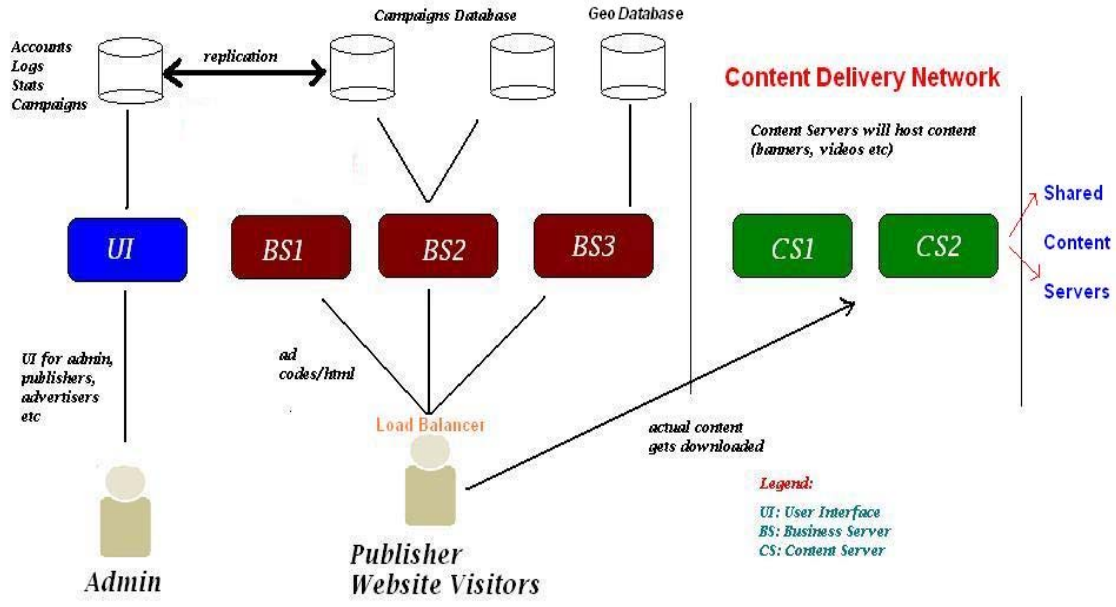
If reinstalling the package, new changes may not take effect due to caching within tomcat. Following steps may be taken:

1. From \Tomcat 5.5\work\Catalina\localhost directory, delete adserversolutions directory.
2. Restart tomcat

5. Enterprise Deployment

5.1. Architecture Planning

Enterprise deployment requires planning the network architecture based on capacity requirements. Typical enterprise architecture for a high volume operation is presented below.



Definitions of proposed logical servers

UI Server (UI):
Reserve a server for the main portal that will serve as a point of access for all users. This server will use its own database for persisting information like ads, campaigns, orders and statistics. You may use multiple UI servers that may be load balanced.

Business Servers (BS1, BS2 and so on):
Business servers will be used for ad selection. These servers will be the busiest servers that will select ads from database based on supplied criteria. Business servers will have to be load balanced for high throughput and failover. Each business server will work with its own database.

Content Servers (CS1, CS2 and so on):
You may reserve separate servers for serving content. While business servers select ads to stream, it will be content servers that will actual stream content down to client browser and devices. Linux servers with high bandwidth availability are most ideally suited for content servers.

Companies that do not wish to segregate roles of business and content servers may stream content from business servers themselves.

If you wish to use our proprietary hosted CDN, all you will need is business servers and one entry into CDNServers table to point to our CDN server. This model can enable customers with virtual dedicated or even shared servers to run high bandwidth application. Contact us for details.

Database Servers (DB):
For this high throughput and scalable architecture to work, all databases must work in merge replication mode.

5.2. Deployment Scenarios

1. **One physical server acting as UI, BS and CS on single database** – This is typically a standard product and clients will not derive real benefits of an enterprise set up. Only some benefits of enterprise product like memory based processing will be available.

2. **One server for UI and BS on single database and separate CS** – This enterprise set is useful for clients who wish to segregate their content streaming from core business logic. Clients with low throughput but high bandwidth requirements (video ads) can benefit from this scenario.

3. **Separate servers for UI and BS on single database and separate CS** – Good for high throughput and high bandwidth applications. Easy set up but no failover on database side and reliance on regular database backups.

4. **Separate servers for UI and BS on multiple databases and separate CS** – This is the most robust and scalable scenario. Merge replication required to keep all databases in sync with each other.

5.3. Deployment Details per Logical Server

User Interface Server

Product Set Up
Deploy the product as explained in “standard deployment” section.

CDN Servers Set Up

Additionally, a table called CDNServers is present in the database. To prevent accidental updates, this table cannot be edited from an online screen.

```
    CREATE TABLE `cdnservers` (
  `ServerID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `Domain` varchar(100) DEFAULT NULL,
  `FTPId` varchar(30) DEFAULT NULL,
  `FTPPwd` varchar(30) DEFAULT NULL,
  `ServerStatus` char(1) DEFAULT NULL,
  PRIMARY KEY (`ServerID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Domain is the domain path where content resides. FTP id/pwd are required so that content broadcast program can broadcast content to all content servers.

Operational Notes:

- If a server is not active, set ServerStatus = N.
- If new entries are added to this table, website must be restarted in IIS.
- If the server that runs adserver product also needs to be used as a CDN server, then an entry must be added as follows:
    - Use the AppRoot value in web.config and append “Ads/” and store this in domain column.

Process Set Up

- ContentBroadcast.class <mins> – this process will to be scheduled (cron) every N minutes, as defined in the argument. This process will take all ads uploaded by users in the last N minutes and will broadcast (FTP) these ads to the designated content servers, as defined in CDNServers table.

  Note: This program is not required to be run if [a] all servers have capability to map to a common drive [b] external CDN system like our proprietary CDN is being used.

Business Servers

Each business server will have a copy of adserver product installed and configured the same way as described in Section 4 above.

For higher performance, we recommend that <AppRoles>, <CommandTypes> and <ViewID> tags be removed from web.config of business servers that will run only the rotation program. Server that will run the user interface programs must have these tags.

Process Set Up

Following processes will have to be set up for the enterprise version. These processes reside in the Processes directory. It is recommended that these exes are run on the business servers and instead run on another server (database server may be used). This will ensure that web servers are not overloaded with this processing.

- LogsArchive.class <mins>, <path> – this program will archive all ad logs from the database into the file system. This program may be scheduled (windows or third party) and it takes two arguments.
    - <mins> - logs older than specified minutes to be deleted. So, if this parameter is 300, then all logs older than now minus 300 minutes will be archived and purged.
    - <path> - directory location where logs will be archived in txt files.

Database Servers

Depending on deployment scenario, ad server database will be set up on each designated database server and web.config will accordingly be configured with the connection string.

In the deployment scenario where multiple replicated databases are used, all databases configured with UI and BS servers will need to be set up for merge replication. All tables except logs table will be replicated.

You may choose to install one Geo database per database server. Geo database is required only for business servers. Geo database set up instructions are provided in the following section.

6. Geo Database Deployment

Customers who have purchased Geo database will need to follow the steps below. All files are supplied along with license key information.

- Create new database. E.g. adserversolutionsgeo.
- Unzip GeoIP-MySQL.zip. Run the scripts on the adserversolutionsgeo database to create tables.
- Unzip GeoIP.zip. This will extract two txt files that can be imported into the two tables. GeoLiteCityBlocks.txt will be imported into GeoLiteCityBlocks table. GeoLiteCityLocation.txt will be imported into GeoLiteCityLocation table. Given below are mysql commands that can be run to import the files:

```
load data local infile
'D:\\inetpub\\wwwroot\\AdserverGeo\\database\\GeoIP\\GeoLiteCityLocation.txt'
into
table geolitecitylocation
fields terminated by ',' enclosed by '"' lines terminated by '\n'
(LOC_ID,COUNTRY,REGION,CITY,POSTALCODE,LATITUDE,LONGITUDE,METROCODE,AREA
CODE,CONTINENT)

load data local infile 'D:\\inetpub\\wwwroot\\AdserverGeo\\database\\GeoIP\\GeoLiteCityBlocks.txt'
into table geolitecityblocks
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
(IP_FROM, IP_TO, LOC_ID)
```

- NOTES:
  All Geo files are delimited by character comma.
  On windows, use \\ when specify location of file (as shown above).
  Ignore any warnings during upload process.

- Alternatively, customers may use our Geo web service in which case they must check with their hosting provider if they permit outbound web service connections. See Appendix B for details on configuring Geo as a web service.

7. Appendix A

Given below is web.xml file present under WEB-INF folder. Strings in yellow need to be updated. Below each string is a comment on how to configure that parameter.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <!--Application Parameters-->
    <context-param>
            <param-name>AppName</param-name>
            <param-value>Adserversolutions</param-value>
    </context-param>
    <context-param>
            <param-name>AppRegion</param-name>
            <param-value>PROD</param-value>
    </context-param>
    <context-param>
        <param-name>appRoot</param-name>
        <param-value>http://localhost:8118/adserversolutions</param-value>

        // This will be your application root for adserver product. No / must be
        entered in the end. Application name is case sensitive and must match the
        definition in web server.

    </context-param>
     <context-param>
        <param-name>appPath</param-name>
        <param-value>D:\adserversolutions</param-value>

        // This will be physical path to your application root. It is case sensitive.
        Must not end in a \. On linux/unix, use / instead of \.

    </context-param>
     <context-param>
        <param-name>serverIp</param-name>
        <param-value>localhost</param-value>
    </context-param>
     <context-param>
        <param-name>ENC_PASSWORD</param-name>
        <param-value>Y</param-value>
    </context-param>
     <context-param>
        <param-name>ENC_PASSWORD_TYPE</param-name>
        <param-value>MD5</param-value>
    </context-param>
     <context-param>
        <param-name>QS_ENC_TYPE</param-name>
        <param-value>Rijndael:#</param-value>
    </context-param>
    <context-param>
        <param-name>WEB_SERVICE_AUTH</param-name>
        <param-value>Y</param-value>
    </context-param><context-param>
        <param-name>WEB_SERVICE_ENC_REQUEST</param-name>
        <param-value>Y</param-value>
    </context-param><context-param>
        <param-name>WEB_SERVICE_ENC_TYPE</param-name>
        <param-value>base64:!#$a54?3</param-value>
    </context-param><context-param>
        <param-name>MOBILE_ENC_REQUEST</param-name>
        <param-value>Y</param-value>
    </context-param><context-param>
        <param-name>MOBILE_ENC_TYPE</param-name>
        <param-value>base64:!#$a54?3</param-value>
    </context-param><context-param>
        <param-name>WEBF_VERSION</param-name>
        <param-value>4.6.3</param-value>
    </context-param><context-param>
        <param-name>APP_VERSION</param-name>
```

```
        <param-value>3.0.0</param-value>
    </context-param>

    <!--Logging-->
    <context-param>
      <!--Following keys are used for clsLogger class.-->
        <param-name>messageLevel</param-name>

         <param-value>E***</param-value>

        // This parameter needs to be changed only if system has to be run in debug
        mode for troubleshooting issues. To set up debug mode, change this value to
        EWID.

    </context-param>
        <context-param>
        <param-name>LogFilePath</param-name>

         <param-value>D:\adserversolutions_Java1\web\Logs\log.txt</param-value>

        // This is the physical location where log file will be created (full path). On
        linux/unix, use / instead of \.


  </context-param>

  <!--Database Connection Strings-->
    <context-param>
                <param-name>DataSource</param-name>
                 <param-value>mysql</param-value>
    </context-param>
    <context-param>
                <param-name>ConnectionDriver</param-name>
                 <param-value>com.mysql.jdbc.Driver</param-value>
    </context-param>

    <context-param>
            <param-name>DEFAULT</param-name>
            <param-value>
    </context-param>
    <context-param>
            <param-name>AdserversolutionsGeo</param-name>
            <param-value></param-value>
    </context-param>

    <!--Misc App Parameters-->
    <context-param>
            <param-name>roleName</param-name>
            <param-value>NETADMIN</param-value>
    </context-param>
    <context-param>
            <param-name>sessionTimeoutMins</param-name>
            <param-value>1200</param-value>
    </context-param>

<!--SMTP Details-->
     <context-param>
           <param-name>smtpserver</param-name>
           <param-value>202.63.164.4</param-value>
    </context-param>
     <context-param>
                <param-name>smtpserverport</param-name>
                 <param-value>25</param-value>
    </context-param>
     <context-param>
                <param-name>sendusing</param-name>
                 <param-value>2</param-value>
    </context-param>
    <context-param>
                <param-name>smtpconnectiontimeout</param-name>
                 <param-value>120</param-value>
    </context-param>
    <context-param>
                <param-name>smtpusessl</param-name>
                 <param-value>False</param-value>
    </context-param><context-param>
```

```
        <param-name>smtpauthenticate</param-name>
        <param-value>1</param-value>
    </context-param>
    <context-param>
            <param-name>sendusername</param-name>
            <param-value/>
    </context-param>
    <context-param>
            <param-name>sendpassword</param-name>
            <param-value/>
    </context-param>

    // Given above are all parameters related with sending mails from the application.

<!--Objects-->
     <context-param>
        <param-name>com.sun.faces.verifyObjects</param-name>
        <param-value>false</param-value>
    </context-param>
    <context-param>
        <param-name>com.sun.faces.validateXml</param-name>
        <param-value>true</param-value>
    </context-param>
    <context-param>
        <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
        <param-value>client</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
        </servlet>
    <servlet>
        <servlet-name>Logger1</servlet-name>
        <servlet-class>WebF_Common_Logger.Logger1</servlet-class>
    </servlet>

    <!--Servlet Definition-->
    <servlet>
        <servlet-name>AdManager</servlet-name>
        <servlet-class>WebF_AdManager.AdManager</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>AdLinkManager</servlet-name>
        <servlet-class>WebF_AdManager.AdLinkManager</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>Logger1</servlet-name>
        <url-pattern>/Logger1</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
              <servlet-name>AdManager</servlet-name>
              <url-pattern>/servlet/WebF_AdManager.AdManager</url-pattern>
       </servlet-mapping>
        <servlet-mapping>
              <servlet-name>AdLinkManager</servlet-name>
              <url-pattern>/servlet/WebF_AdManager.AdLinkManager</url-pattern>
       </servlet-mapping>

    <!--System Parameters-->
    <session-config>
            <session-timeout>
            30
            </session-timeout>
    </session-config>

    <welcome-file-list>
            <welcome-file>
            index.jsp
            </welcome-file>
    </welcome-file-list>
   </web-app>
```

8. Appendix B

Given below is context.xml file present under adserversolutions/META_INF folder. Strings in yellow need to be updated. Below each string is a comment on how to configure that parameter.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/adserversolutions" reloadable="no">

  <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="100"
maxIdle="30" maxWait="10000" name="jdbc/DEFAULT" type="javax.sql.DataSource"
url="jdbc:mysql://localhost:3306/adserversolutions?autoReconnect=true" username="adserver"
password="adserver"/>

  <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="100"
maxIdle="30" maxWait="10000" name="jdbc/AdserversolutionsGeo" type="javax.sql.DataSource"
url="jdbc:mysql://localhost:3306/adserversolutionsgeo?autoReconnect=true" username="adserver"
password="adserver"/>

</Context>
```

Notes:

1. Change the context path to match the project directory.
2. Change mysql connection url, username and password for both resources.
3. Connection pooling parameters may be adjusted to suit your environment. Details may be found in the following location: http://commons.apache.org/dbcp/configuration.html
4. On some installations, it may be required to copy these entries into context.xml that is located in your web container root ($CATALINA_HOME/conf for tomcat).

End of Document