Ivory Server for Java on Linux

Installation Guide

May 2017

E-mail:	**sales@gtsoftware.com or support@gtsoftware.com**
URL:	**www.gtsoftware.com**

CORPORATE HEADQUARTERS
GT Software, Inc.
6255 Barfield Road, Suite 200
Atlanta, GA 30328 USA
Phone: 404-253-1300 Fax: 404-253-1314

UK/EMEA OFFICE
1 Market Place Mews
Henley-on-Thames
Oxon, RG9 2AH
Phone: +44 (0)1183-842130

Contents

Introduction to Ivory Service Architect

Ivory Service Architect easily extends access to the mainframe through Web services—with no code written, generated, or changed, and with no additional MIPS usage. Mainframe integration is quick and easy, regardless of platform, programming language, or data format. With Ivory Service Architect, mainframe developers can seamlessly integrate new technologies like mobile, Web-based portals, business process management and packaged applications.

The main components of Ivory Service Architect are:

- Ivory Studio
- Ivory Server

Ivory Service Architect makes Web service development and deployment simple. Developers use Ivory Studio, an intuitive drag-and-drop tool, to assemble Web services from mainframe applications and data without additional coding. Developers define Web service inputs and outputs, and then graphically model the multi-step process to implement the service. Ivory® Server publishes information from Ivory Studio, accessing applications and data directly in each native environment. Ivory Server can be deployed on or off the mainframe.

What is Ivory Studio?

Ivory Studio is a PC-based IDE that provides common ground for mainframe programmers and PC developers to use existing business logic to drive new SOAP/REST services. The drag-and-drop graphical modeler provides "end-to-end" representations of business logic and data flow. The graphical models present complex business processing in an easy-to-follow diagram.

Ivory Studio provides the means to design a SOAP/REST Web service for existing business logic and deploy the process without having to write any program code. Ivory Studio also provides Callable services, which enable your CICS, IMS and batch applications to consume Web services.

The Ivory Studio modeler provides an easy-to-use method for diagramming CICS TS transactions and application program flow. You define the inputs and expected outputs, and then design the Web service using the component icons. The end result is a graphical design of the application from the Web service inputs to the final Web service response.

Ivory uses an open XML format to ensure that it is compatible with other systems.

Ivory Studio includes an FTP client to pull copybooks and BMS macros from mainframe file systems. They can then be imported by Ivory Studio and converted into an XML format.

Effective with version 4, Ivory Studio also offers a graphical user interface and IDE for creating and maintaining maps. Ivory BMS Studio takes advantage of SOA technologies, using built-in Web services to extend the capabilities of BMS/TS. Ivory BMS/TS (release 8.4 and later) provides SOA-based Web services for use with Ivory BMS Studio. Ivory BMS Studio is a separately licensed feature of Ivory Service Architect.

What is Ivory Server?

Ivory Server processes information published via Ivory Studio, accessing 3270, CICS, IMS, and data directly in each native environment. Ivory Server consists of a high-performance SOAP processor, a business service flow processor, and a central repository for WSDL discovery.

The server receives SOAP / REST (XML and JSON) requests, invokes the business service flow previously defined in Ivory Studio to satisfy the request, and formats the response. Ivory Server provides this support via HTTP and HTTPS protocols.

Ivory Server can execute custom code within the same address space—or process—when "delegates" are added to Ivory projects. Delegates are user written programs that provide Ivory users a method to locally access additional programming logic that's normally not available through service processing.

Ivory Server exploits CICS, IMS, and native data access capabilities, eliminating the need for middle-tier servers, and providing the flexibility to fully leverage mainframe processing power as appropriate within the SOA. Ivory Server optimizes the storage of data dynamically by only saving data as needed to serve the application or send output for the SOAP response.

For the zSeries environments, the Ivory Server supports zIIP, zAAP, and IFL specialty engines to help manage and control the z/OS General Processor use when processing services.

There are several Ivory Server products, including:

- Ivory Server for CICS/TS
- Ivory Server for z/OS
- Ivory Server for z/VSE
- Ivory Server (Java) for Windows
- Ivory Server (Java) for Linux
- Ivory Server (Java) for zLinux

Installing Ivory Server for Java on Linux

This document contains instructions on how to install or upgrade Ivory Server for Java on Linux.

Ivory Server for Java on Linux is packaged in a single .tar file named *ivoryserver4j.build.tar* where "*build*" is the current build version number.

Requirements

Ivory Server for Java on Linux executes as a Java servlet and requires a container Java application server or servlet engine that meets the following minimum requirements:

- Java Virtual Machine J2RE 1.5.0 or above
- Application server or servlet engine supporting the Java servlet API 2.3 specification or above
- (optional) IBM CICS Transaction Gateway 6.0 and above; required if Ivory projects access host CICS Commarea and/or Channels and Containers based resources

Ivory Server for Java on Linux has been tested with (and is supported for) the following containers:

- IBM WebSphere Application Server 5.1-7.0
- BEA WebLogic Application Server 9.0-10.3
- Apache Tomcat Servlet Engine 4.1.39-7.0.16
- Oracle Application Server Containers for J2EE 10g (10.1.2.0.0)
- JBOSS 5.0.1

Ivory Server for Java on Linux executes under any version of Intel 32-bit or 64 bit Linux that supports an appropriate application server or servlet engine specified above and meets these suggested minimum levels:

- Linux Kernel 2.6.9-67
- libstdc++.so.6libcrypt.so.1
- libgcc_s.so.1
- libc.so.6
- libm.so.6

Ivory Server for Java on Linux will execute in either 32 bit or 64 bit mode. This is determined by the Java Virtual Machine used to execute the servlet container. The web.xml parameter *ivoryNativeLibrary* may need to be modified based on addressing mode.

- Use libIvoryServer4J.so for 32 bit mode
- Use libIvoryServer4J64.so for 64 bit mode

In this document, the term servlet container refers to the application server or servlet engine chosen.

New Installation

Follow these instructions to install Ivory Server for Java on Linux for the first time.

1. Sign on to your Linux system with root level authority.
2. Place the installation tar file in the /opt directory and extract it. Issue the following commands replacing "*build*" with the appropriate build version:

   ```
   cd /opt
   tar -xvf ivoryserver4j.build.tar
   ```

   Note: You can install Ivory Server under a different directory structure, but you will then need to adjust the appropriate web.xml or ivory.properties settings later in the installation.
3. Copy the key.lic (license) file (obtained from GT Software) into the directory /opt/gtsoftware/ivoryserver4j/license, which was created during the .tar extract process.
4. Issue the commands listed below to create the project deploy directory. Ivory Studio users will deploy project files to the host Linux system via FTP or SFTP. The project files can reside in any directory structure; the default container directory is /var/ivory.

   ```
   cd /var
   mkdir ivory
   ```

   Note: You must give all Ivory Studio users read-write directory level authority to /var/ivory so that they can create subdirectories and files within the /var/ivory directory.
5. Configure the Linux FTP/SFTP server to allow Ivory Studio users to issue the FTP change directory command outside their home directory on the Linux system.

   Note: Optionally, you can create a single user ID on the Linux system for all Ivory Studio users, for example, "ivoryuser". All users then sign on to FTP/SFTP with the ivoryuser ID and store the project files in a directory such as /home/ivoryuser/ivory. All Ivory Studio users would need to configure their FTP profile Deploy Path property to /home/ivoryuser/ivory. In this case, it is also necessary to modify the Ivory Server web.xml; set the context parameter "ivoryDeployDir" to /home/ivoryuser/ivory.
6. Install the Java Web application into your servlet container. This procedure varies by servlet container. The file web.xml contains parameters necessary for Ivory Server to run properly. These parameters may be modified within web.xml or may be overridden using an external file named ivory.properties. If you will be installing the Java Web application using one of the supplied war files and you need to modify any of the parameters below, using ivory.properties is a good alternative. See the section "Using ivory.properties" for more details.

   The following are the web.xml parameters (the web.xml parameters are described here):

   ```
     <context-param>
     <param-name>ivoryNativeLibrary</param-name>
     <param-
   value>/opt/gtsoftware/ivoryserver4j/bin/libIvoryServer4J.so</pa
   ram-value>
       </context-param>
       <context-param>
     <param-name>ivoryKeyLicFile</param-name>
   ```

```
  <param-
value>/opt/gtsoftware/ivoryserver4j/license/key.lic</param-
value>
    </context-param>
    <context-param>
  <param-name>ivoryDeployDir</param-name>
  <param-value>/var/ivory/</param-value>
    </context-param>
    <context-param>
  <param-name>ivoryHostCodepage</param-name>
  <param-value>37</param-value>
    </context-param>
    <context-param>
  <param-name>ivoryCTGUrl</param-name>
  <param-value></param-value>
    </context-param>
    <context-param>
  <param-name>ivoryForceIVS</param-name>
  <param-value>false</param-value>
    </context-param>
    <context-param>
  <param-name>ivoryTraceOverride</param-name>
  <param-value>0</param-value>
    </context-param>
    <context-param>
  <param-name>ivoryTraceMaxCDataLen</param-name>
  <param-value>0</param-value>
    </context-param>
    <context-param>
  <param-name>ivoryDataAccessConfigDir</param-name>
  <param-value></param-value>
    </context-param>
    <context-param>
  <param-name>ivoryInternalWebServicesSourceDir</param-name>
  <param-value>/opt/gtsoftware/ivoryserver4j/ivoryiws/</param-
value>
    </context-param>
    <context-param>
  <param-name> ivoryWebAppIVW</param-name>
  <param-value></param-value>
    </context-param>
```

7. Identify the context root. Each servlet container has its own way of specifying the context root of an installed Web application. The context root is a variable and is important for both Ivory Studio users and consumers of Ivory Web services. You can choose any context root for the Ivory Server application, but you must provide this value to Ivory Studio developers. The context root will be used as part of each Web service URI. Traditionally, other Ivory Server versions have shipped with the context root of "soap". You can use "soap" or any other value.

8. If your Ivory projects will not access CICS resources, skip this step.

Ivory Server projects can be designed to access CICS resources. If so, Ivory Server relies on IBM CICS Transaction Gateway being installed on the Linux system and at least the client installed within the servlet container.

The following .jar files need to be available for class loading within the servlet container:

ctgclient.jar
ctgserver.jar (optional, see web.xml parameter ivoryCTGUrl for details)

IBM CICS Transaction Gateway also relies on access to its native library (libctgjni.so). Make sure that IBM CICS Transaction Gateway is properly installed before accessing it via Ivory Server. See IBM documentation for further details.

9. Run the supplied installation verification program (ivp.srv) to verify that Ivory Server is ready for use. After your servlet container is updated and ready for use, start a Web browser that has access to the servlet container. Enter a URL with the following pattern:

   http://linuxhost:port/context-root/ivp.srv

   where:

   - *linuxhost* is the domain name or IP address of the Linux Server
   - *port* is the http port the Servlet Container is listening to for requests
   - *context-root* is the value identified in step 6
   - *ivp.srv* is specified in web.xml as the default mapping for the installation verification program

   A sample URL would be:

   http://www.myhost.com:9080/soap/ivp.srv

   The installation verification program will return detailed information to the Web browser about the Ivory Server installation. Fix any problems it reports and rerun ivp.srv until the summary status reports the installation succeeded.

   If you will be using projects containing DataAccess Ivory Supplied Delegate Nodes, do the following.

   - Click on the "Manage internal Web services" link.
   - Deploy the internal web service used by the DataAccess wizard to the deploy directory.

Upgrade Installation

Follow these instructions to upgrade an existing Ivory Server for Java on Linux installation.

1. Sign on to your Linux system with root level authority.
2. Ivory Server for Java on Linux is packaged in a single .tar file. Place the installation .tar file in the /opt directory and extract it, overlaying your original installation. Issue the following commands replacing "*build*" with the appropriate build version:

   ```
   cd /opt
   tar -xvf ivoryserver4j.build.tar
   ```

   Note: If you originally installed Ivory Server for Java on Linux under a different directory structure, you should place and extract the .tar in the directory you used for the original installation.

3. Use your servlet container's upgrade process to install the upgrade. This procedure varies by servlet container. The file web.xml contains parameters necessary for Ivory Server to run properly. These parameters may be modified within web.xml or may be overridden using an external file named ivory.properties.

   If you will be upgrading the Java Web application using one of the supplied war files and you need to modify any of the parameters below, using ivory.properties is a good alternative. See the section "Using ivory.properties" for more details.

   Note: The web.xml files are modified for each release level (such as 3.5.0 to 3.6.0). When upgrading, you must use the new web.xml file. This is especially true if you are using the expanded directory structures instead of a .war file to upgrade Ivory Server.

   The following are the web.xml parameters (the web.xml parameters are described here):

   ```
     <context-param>
     <param-name>ivoryNativeLibrary</param-name>
     <param-
   value>/opt/gtsoftware/ivoryserver4j/bin/libIvoryServer4J.so</pa
   ram-value>
       </context-param>
       <context-param>
     <param-name>ivoryKeyLicFile</param-name>
     <param-
   value>/opt/gtsoftware/ivoryserver4j/license/key.lic</param-
   value>
       </context-param>
       <context-param>
     <param-name>ivoryDeployDir</param-name>
     <param-value>/var/ivory/</param-value>
       </context-param>
       <context-param>
     <param-name>ivoryHostCodepage</param-name>
     <param-value>37</param-value>
       </context-param>
       <context-param>
     <param-name>ivoryCTGUrl</param-name>
     <param-value></param-value>
   ```

```
    </context-param>
    <context-param>
  <param-name>ivoryForceIVS</param-name>
  <param-value>false</param-value>
    </context-param>
    <context-param>
  <param-name>ivoryTraceOverride</param-name>
  <param-value>0</param-value>
    </context-param>
    <context-param>
  <param-name>ivoryTraceMaxCDataLen</param-name>
  <param-value>0</param-value>
    </context-param>
    <context-param>
  <param-name>ivoryDataAccessConfigDir</param-name>
  <param-value></param-value>
    </context-param>
    <context-param>
  <param-name>ivoryInternalWebServicesSourceDir</param-name>
  <param-value>/opt/gtsoftware/ivoryserver4j/ivoryiws/</param-
value>
    </context-param>
    <context-param>
  <param-name> ivoryWebAppIVW</param-name>
  <param-value></param-value>
    </context-param>
```

4. Because Ivory Server relies on native library code, you may need to restart your servlet container for all changes to become effective. See your servlet container's documentation for more details.
5. Run the supplied installation verification program (ivp.srv) to verify that Ivory Server is ready for use. After your servlet container is updated and ready for use, start a Web browser that has access to the servlet container. Enter a URL with the following pattern:

   http://linuxhost:port/context-root/ivp.srv

   where:

   - *linuxhost* is the domain name or IP address of the Linux Server
   - *port* is the http port the Servlet Container is listening to for requests
   - *context-root* is the value identified during the original installation
   - *ivp.srv* is specified in web.xml as the default mapping for the installation verification program

   A sample URL would be:

   http://www.myhost.com:9080/soap/ivp.srv

   The installation verification program will return detailed information to the Web browser about the Ivory Server installation. Fix any problems it reports and rerun ivp.srv until the summary status reports the installation succeeded.
6. If you will be using projects containing DataAccess Ivory Supplied Delegate Nodes, do the following.
   - Click on the **Manage internal Web services** link.

- Check the deploy date against the current installer date.
- If the installer date is newer and you have not deployed the internal web service project from Ivory Studio, you should click the **Deploy** button to update.

Appendix A: Using ivory.properties

If you would like to install Ivory Server via a war file but need to change the parameters within web.xml you may use *ivory.properties* instead of having to modify web.xml.

ivory.properties is an external file outside the war file and its values override the values within web.xml. This is accomplished by placing a copy of ivory.properties within a directory which is part of Ivory Server's web application classpath. Ivory Server will attempt to access ivory.properties via the web applications's class loader. If the file is found, Ivory Server will use the values in ivory.properties, overriding the parameters in web.xml.

A sample ivory.properties is located at:

```
/opt/gtsoftware/ivoryserver4j/version44/samples/overrides/ivory
.properties
```

Copy this sample to a directory on the Ivory Server web application classpath and modify accordingly. The installation verification program, ivp.srv, has the following status information:

```
Override ivory.properties in use   Yes | No
```

Once this value is presented as "Yes" ivory.properties is being used. Since Ivory Server Servlet only reads ivory.properties once during servlet init() you must have your servlet container restart the Ivory Server Servlet for the modified overrides to be used.

Appendix B: web.xml Parameters

ivoryNativeLibrary

Identifies the location of the Ivory Server native library code.

Ivory Server attempts to directly load this library from the location specified.

If Ivory Server was installed in the default directory structure (/opt/gtsoftware/...), this parameter does not need to be changed.

ivoryKeyLicFile

Identifies the location of the Ivory license file.

If Ivory Server was installed in the default directory structure (/opt/gtsoftware/...), this parameter does not need to be changed.

ivoryDeployDir

Identifies the location of deployed Ivory project files.

If Ivory project files will be contained in the /var/ivory directory, this parameter does not need to be changed.

ivoryHostCodepage

Identifies the host system codepage.

If your host systems (CICS, IMS) use US-EBCDIC codepage 37, this parameter does not need to be changed.

ivoryCTGUrl

Identifies the URL to use when making IBM CICS Transaction Gateway requests.

This value is required when IBM CICS Transaction Gateway is running outside the servlet container. The default empty value indicates ctgserver.jar is accessible within the servlet container and all IBM CICS Transaction Gateway requests are to be made locally.

ivoryForceIVS

Indicates whether the old style textual xml server instructions should be used instead of the newer better performing binary formatted server instructions.

Old style textual xml server instructions are contained in files with the .ivs extension. Setting the parameter value to true will cause Ivory Server to use the textual xml server instructions.

Note: The default value is false.

ivoryTraceOverride

Overrides the project level Server Side Tracing property contained in the Start Node. Valid values are as follows:

0	No override, use project level setting
1	Override, turn tracing off
2	Override, turn Complete tracing on for all project executions

ivoryTraceMaxCDataLen

When server side tracing is set to complete, data areas such as CICS COMMAREA storage are written to the trace in CDATA sections. This parameter may be used to control the amount of storage to write to the trace.

The value of ivoryTraceMaxCDataLen is applied to each separate storage area. Valid values are as follows:

-1	Do not write any storage areas to the trace
0	Write out complete storage areas to the trace, regardless of size
val	Limit the size of the data written to the trace to "val" where val is an integer value

ivoryTraceMaxSize

When server side tracing is on, this parameter can be used to limit the size of the trace. Tracing will stop when it hits the specified value in kilobytes.

The value of **ivoryTraceMaxSize** is applied to each separate storage area. Valid values are as follows:

0	No limit to the trace size (default).
Val	Limit the size of the trace in kilobytes. Valid values 1 – 32767 kb.

ivoryDataAccessConfigDir

Identifies the directory containing configuration files used by Ivory Data Hub.

Usually this is the same directory which contains the Ivory Data Hub JDBC driver .jar file. This parameter is necessary if you will be executing Ivory projects that contain Ivory Data Hub Supplied Delegate Nodes.

ivoryInternalWebServicesSourceDir

Identifies the root directory containing the installer versions of the deployable internal Web service files.

If Ivory Server was installed in the default directory structure (/opt/gtsoftware/...), this parameter does not need to be changed.

ivoryWebAppIVW

Identifies the location of the web application level work variable initialization file. All Ivory projects executed under the current context root will have the work variable initialization values set in this file applied before project execution.

If none of the above web.xml parameters need to be modified and your servlet container supports the installation of .war files, install one of the .war files listed below.

- If the servlet container is not WebSphere, install:

```
/opt/gtsoftware/ivoryserver4j/version44/war-
default/ivoryserver4j.war
```

- If the servlet container is WebSphere, install:

```
/opt/gtsoftware/ivoryserver4j/version44/war-
websphere/ivoryserver4j.war
```

Note: Read the WebSphere Notes (below) before installing.

If you need to modify one of the web.xml parameters listed above, or need to modify the .war file contents, or your servlet container does not support the installation of .war files, use one of the directory structures listed below.

- If the servlet container is not WebSphere, install:

```
/opt/gtsoftware/ivoryserver4j/version44/war-default/expanded
```

- If the servlet container is WebSphere, install:

```
/opt/gtsoftware/ivoryserver4j/version44/war-websphere/expanded
```

Note: Read the WebSphere Notes (below) before installing.

Appendix C: WebSphere Specific Notes

WebSphere currently includes an older version of the open source XML formatter named JDOM described at:

```
    http://www.jdom.org/
```

Ivory Server uses JDOM but relies on a newer version than the WebSphere included version. During the .war file installation, you must change the "Class loader mode" option from its default value to "PARENT LAST". This option can be found within the WebSphere Administrative Console at:

```
    Enterprise Applications> ivoryserver4j_war> Web module>
    ivoryserver4j.war
```

If you don't make this change, you will most likely see the "JDOM compatibility check" fail when running the installation verification program.

Appendix D: OpenAPI Web Application Installation

To provide testing for REST Web Services, GT Software provides a customized web application for deployment along with the Ivory Server.

openapi.war contains the web application and is located in the war-default directory listed below.

```
    /opt/gtsoftware/ivoryserver4j/version60/war-default/openapi.war
```

Install the customized OpenAPI web application into your application container. This installation procedure varies by application container.

Appendix E: Startup JVM Properties Required for SSL Client Certificates

The following JVM properties should be added to your application server or servlet engine:

Mandatory JVM System Properties

Property	Description
javax.net.ssl.keyStore	The name of the file containing the KeyStore object that the default KeyManager should use. This is the keystore client certificates are stored in, which will be referenced in Ivory projects.
javax.net.ssl.keyStorePassword	The password for the KeyStore object that the default KeyManager should use.

Optional JVM System Properties

Property	Description
javax.net.ssl.keyStoreType	The type of KeyStore object that the default KeyManager should use. The default value is the value returned by the KeyStore.getDefaultType method.
sun.security.ssl.allowUnsafeRenegotiation	Set this system property to True to permit full legacy renegotiation.

Appendix F: Using Ivory Service Architect with Ivory Data Hub

Complete the following steps to use Ivory Data Hub with Ivory Server for Java.

1. After installing Ivory Data Hub, install the Data Hub JDBC driver on the system where Ivory Server is running.
2. Update the ivoryDataAccessConfigDir parameter in the web.xml file.
3. Configure the JDBC properties of the Data Hub Point Node in Ivory Studio.

   Refer to the Data Hub Point Node Properties in the Ivory Studio documentation for additional information.

Getting Technical Support

Product Documentation

The online help system is the first place to look when you need information about the product.

GT Software Website

From the **Support** section of the GT website you can:

- search the Support Portal for knowledge base articles and software downloads
- submit an **Enhancement Request**

Technical Support Staff

If you cannot resolve the problem using the product documentation or online knowledge base, the GT Technical Support staff is available for telephone consultation Monday through Friday between the hours of 8:30 AM and 5:15 PM Eastern time. After business hours, you can leave a message. Your call will be returned the next business day. Before you call:

- Try to re-create the problem
- Have the following information available:
  - Product release number
  - Operating system type and version
  - The text of the system message (if any)
  - Complete description of the problem

Important: In order to resolve a problem it is important to tell the Technical Support representative exactly what was being done when the problem occurred—details are important. It may be helpful to make notes about what happened before you call.

Contacting GT Software

Visit the GT Software website at http://www.gtsoftware.com
Send Internet e-mail to sales@gtsoftware.com or support@gtsoftware.com
To send a facsimile (fax), dial 404-253-1314.

Call GT Software at **404-253-1300 or toll free: 800-756-43GT (4348)**. Customers outside of the U.S. can contact their authorized GT agent for technical support. If the agent is unavailable, contact GT Software directly.

Index