vms Software

CSWS_JAVA for OpenVMS (based on Apache Tomcat)

Installation Guide and Release Notes

December 2019

Software Version:
CSWS_JAVA (Apache Tomcat) for OpenVMS
Version 8.5-50A

Introduction

Apache Tomcat is an open-source Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSockets, and provides a pure Java HTTP web server environment in which Java code can run. For more information, refer to the Apache Tomcat project web site http://tomcat.apache.org/.

CSWS_JAVA Version 8.5-50A for OpenVMS is a full binary release of Apache Tomcat 8.5.50 for the OpenVMS operating system.

Significant changes

This release of Apache Tomcat for OpenVMS includes significant changes from previous releases of Apache Tomcat for OpenVMS that are intended to enhance performance and reliability and to simplify administration. These changes are as follows:

- **Apache Portable Runtime (APR) based Native library for Tomcat**

  The most significant change in this release of Apache Tomcat for OpenVMS is the inclusion of support for the APR-based native library. This facility allows Tomcat to use the APR to provide superior scalability, performance, and better integration with native server technologies. The APR is the same library that underpins the Apache HTTP 2.x web server. Use of the APR is enabled by default.

- **Not dependent on Apache HTTP**

  Previous releases of Apache Tomcat for OpenVMS installed by default into a directory hierarchy that assumed Tomcat would be installed and used in conjunction with the Apache HTTP web server. This has been changed with Tomcat now being installed into its own directory hierarchy

with the top-level directory of the installation being defined by the `tomcat$root` logical name. With the exception of start-up and shutdown files, all files reside under this root, including web server logs, which reside in the `tomcat$root:[logs]` directory.

- **Simplified OpenVMS administration interface**

  Previous releases of Apache Tomcat for OpenVMS included a command procedure that could be used to configure ownership of the Apache Tomcat directory tree and the username under which the web server would run. This command procedure is no longer provided and it is up to the system administrator to manually change file ownership and to specify at start-up (via a logical name or via a start-up parameter) the username under which the web server will run. Command procedures are provided to start up and shutdown the web server, and facilities are provided to allow the specification of site-specific details such as Java and DECC logical names, and site-specific JVM command line arguments.

What else is new in this release?

CSWS_JAVA Version 8.5-50A is available on OpenVMS Integrity only, and includes the following changes (see http://tomcat.apache.org/tomcat-8.5-doc/index.html for more details):

- **Dependency changes**

  Tomcat 8.5 is designed to run on Java SE 7 and later.

- **API stability**

  The public interfaces for the following classes are fixed and will not be changed at all during the remaining lifetime of the 8.x series:

  – All classes in the `javax` namespace

  The public interfaces for the following classes may be added to in order to resolve bugs and/or add new features. No existing interface methods will be removed or changed although it may be deprecated.

  – `org.apache.catalina.*` (excluding sub-packages)

- **Bundled API's**

  A standard installation of Tomcat 8.5 makes all of the following APIs available for use by web applications (by placing them in "lib"):

  – `annotations-api.jar` (Annotations package)

  – `catalina.jar` (Tomcat Catalina implementation)

  – `catalina-ant.jar` (Tomcat Catalina Ant tasks)

  – `catalina-ha.jar` (High availability package)

  – `catalina-storeconfig.jar` (Generation of XML configuration from current state)

  – `catalina-tribes.jar` (Group communication)

  – `ecj-4.6.3.jar` (Eclipse JDT Java compiler)

- `el-api.jar` (EL 3.0 API)
- `jasper.jar` (Jasper 2 Compiler and Runtime)
- `jasper-el.jar` (Jasper 2 EL implementation)
- `jsp-api.jar` (JSP 2.3 API)
- `servlet-api.jar` (Servlet 3.1 API)
- `tomcat-api.jar` (Interfaces shared by Catalina and Jasper)
- `tomcat-coyote.jar` (Tomcat connectors and utility classes)
- `tomcat-dbcp.jar` (package renamed database connection pool based on Commons DBCP)
- `tomcat-jdbc.jar` (Tomcat's database connection pooling solution)
- `tomcat-jni.jar` (Interface to the native component of the APR/native connector)
- `tomcat-util.jar` (Various utilities)
- `tomcat-websocket.jar` (WebSocket 1.1 implementation)
- `websocket-api.jar` (WebSocket 1.1 API)

You can make additional APIs available to all of your web applications by putting unpacked classes into a "`classes`" directory (not created by default), or by placing them in JAR files in the "`lib`" directory.

- **Security manager URLs**

  In order to grant security permissions to JARs located inside the web application repository, use URLs of the following format in your policy file:

  ```
  file:${catalina.base}/webapps/examples/WEB-INF/lib/driver.jar
  ```

- **Additional jar files**

  This release includes the following additional jar files:

  - `jstl-1.2.jar`

    The Apache Standard Taglib (https://tomcat.apache.org/taglibs/standard/), which required by Apache AXIS2 for various functions.

  - `commons-el.jar`

    See http://apache.claz.org/commons/el/binaries/). Functionality provided by this jar file is required by code generated by WSIT.

  - `catalina-jmx-remote.jar`

    JMX Remote Lifecycle Listener; see http://tomcat.apache.org/tomcat-8.5-doc/extras.html for additional information.

  - `catalina-ws.jar`

    Web Services support (JSR 109).; see http://tomcat.apache.org/tomcat-8.5-doc/extras.html for additional information.

Software prerequisites

CSWS_JAVA for OpenVMS 8.5-50A requires the following software:

- OpenVMS Integrity servers Version 8.4-1H1 or higher
- Java™ Development Kit (JDK) 8 for the OpenVMS Integrity servers (CSWS_JAVA V8.5-50A will not run with Java 6)
- It is required that you install CSWS_JAVA on an ODS-5 enabled disk

Documentation

For information about Tomcat, see the Jakarta Apache Project and Tomcat 8 documentation, which can be found at http://tomcat.apache.org/tomcat-8.5-doc/index.html.

Before beginning the installation

Before installing CSWS_JAVA 8.5-50A, it is necessary to manually remove any existing version of CSWS_JAVA if it is installed on your system. Shutdown the web server (if it is running) and remove the existing kit by entering the following command. Be sure to backup any site-specific files before performing this operating.

```
$ PRODUCT REMOVE CSWS_JAVA
```

Installing CSWS_JAVA

The kit is provided as an OpenVMS PCSI kit (`VSI-I64VMS-CSWS_JAVA-V0805-50A-1.PCSI`) that can be installed by a suitably privileged user using the following command:

```
$ PRODUCT INSTALL CSWS_JAVA
```

The installation will then proceed as follows (output may differ slightly from that shown):

```
$ PRODUCT INSTALL CSWS_JAVA

Performing product kit validation of signed kits ...

The following product has been selected:
    VSI I64VMS CSWS_JAVA V8.5-50A          Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and
for any products that may be installed to satisfy software dependency
requirements.

Configuring VSI I64VMS CSWS_JAVA V8.5-50A

    VMS Software Inc. & The Apache Software Foundation.

* This product does not have any configuration options.
```

```
Execution phase starting ...

The following product will be installed to destination:
    VSI I64VMS CSWS_JAVA V8.5-50A          DISK$I64SYS:[VMS$COMMON.]

Portion done: 0%...10%...20%...40%...50%...60%...70%...80%...90%...100%

The following product has been installed:
    VSI I64VMS CSWS_JAVA V8.5-50A          Layered Product

VSI I64VMS CSWS_JAVA V8.5-50A

    Post-installation tasks are required.

    To start the Tomcat web server at system boot time, add the following
    lines to SYS$MANAGER:SYSTARTUP_VMS.COM:

        $ file := SYS$STARTUP:TOMCAT$STARTUP.COM
        $ if f$search("'file'") .nes. "" then @'file'

    To shutdown Tomcat at system shutdown time, add the following lines to
    SYS$MANAGER:SYSHUTDWN.COM:

        $ file := SYS$STARTUP:TOMCAT$SHUTDOWN.COM
        $ if f$search("'file'") .nes. "" then @'file'

    Note that default installation uses the SYSTEM account to run the the
    Web server process. It is recommended that you run the web server as
    using a less privileged account. This may be done by supplying the
    account name as a parameter to tomcat$startup.com or by defining the
    logical name tomcat$user as the desired account name. It is also
    recommended that you  change the tomcat$root:[000000...] directory tree
    ownership to this account.
```

Process quotas

Apache Tomcat can require considerable resources in order to operate efficiently, depending on workload requirements. The following quotas should be adequate for most purposes; however resource usage should be carefully monitored, and quotas adjusted as necessary.

```
Maxjobs:          0  Fillm:      32767  Bytlm:       3000000
Maxacctjobs:      0  Shrfillm:        0  Pbytlm:            0
Maxdetach:        0  BIOlm:        1024  JTquota:       40000
Prclm:          100  DIOlm:        1024  WSdef:        100000
Prio:             4  ASTlm:         300  WSquo:        200000
Queprio:          4  TQElm:         400  WSextent:     800000
CPU:         (none)  Enqlm:      32767  Pgflquo:    10000000
```

If the web server is expected to support large numbers of connections then it may also be necessary to increase the `CHANNELCNT` system parameter (this parameter can usually be safely set to its maximum value of 65535).

Installing in an alternative location

By default CSWS_JAVA will be installed in `SYS$SYSDEVICE:[VMS$COMMON]`. If you wish to install the software in an alternative location this can be achieved using the `/DESTINATION` qualifier with the `PRODUCT INSTALL` command to specify the desired location; however it is important to note that an additional manual step will then be required to complete the installation. Specifically, when an alternative destination is specified, the start-up and shutdown procedures and associated files will be

placed into a subdirectory `[.SYS$STARTUP]` residing under the specified destination directory. If you wish to run these files from your standard `SYS$STARTUP` directory they will need to be copied from the destination subdirectory into your systems `SYS$STARTUP` directory.

Configuring CSWS_JAVA

After the installation is complete, you may wish to perform the following steps:

- Use the "`set file/owner`" command to change the `tomcat$root` directory tree's ownership to a less privileged user than `SYSTEM`. Ensure that the specified user exists and has the necessary process quotas required to run Tomcat.
- Configure any site-specific Apache Tomcat and JVM options by adding any site-specific logical names and JVM command line options to the files `sys$startup:tomcat$defs_local.com` and `sys$startup:tomcat$args_local.dat`, respectively.
- Start the web server and ensure that it is up and running:

  ```
  $ @sys$startup:tomcat @startup.com
  $ pipe show system | search sys$input APACHE$TOMCAT
  0000806D APACHE$TOMCAT   CUR   6  6    32178   0 00:00:29.54     39658
  42146 M
  ```

  Note that you may want to give the process a few seconds to start before checking that the `APACHE$TOMCAT` process exists. If Apache Tomcat web server does not start, check the log files in `tomcat$root:[logs]` for additional information.
- By default the web server will run under the username of the user that started the process. To run the process as a different user either define the logical name `tomcat$user` before running the start-up procedure or specify the desired username as a parameter to the start-up procedure.

  Note that the `tomcat$user` logical name can be defined at the process level; it is only required by the Tomcat start-up procedure, `sys$startup:tomcat$startup.com`.
- Access the included JSP and servlet examples via http://hostname:8080 after you have successfully configured and started Apache Tomcat to verify that the web server is operating correctly (replacing "hostname" with the name or IP address of the OpenVMS server on which Tomcat has been installed).

Common problems

This section contains notes about some common problems that may be encountered when using Apache Tomcat on OpenVMS.

- **Access to Tomcat server is slow when it is invoked for the first time**

  When you invoke the Tomcat server for the first time, there might be a delay in accessing http://hostname:8080. The reason for the delay is that Tomcat deploys all of the applications (mostly examples) in the webapps directory. This deployment is done only when the server is invoked for the first time.

- **Redeploying `.WAR` file fails**

Previously, to redeploy `.WAR` files you had to delete all the existing `.WAR` files and the directory tree where the webapp was executed. This was due to Tomcat's inability to delete multiple files and directories. You can overcome this problem by defining the following logical names in the site-specific configuration file `sys$startup:tomcat$defs_local.com`:

```
$ define java$delete_all_versions 1
$ define java$create_dir_with_owner_delete 1
```

- **Initially deploying `.WAR` files**

  It has been observed that `.WAR` files will sometimes not expand upon deployment. The reasons for this and the exact circumstances in which this may happen are being investigated; however evidence suggests that the problem is related to file ownership and permissions. A simple solution to the problem is to issue the command "`$ set file/protection=(w:re) filename.war`", whereupon after a short delay Tomcat will begin to expand the file.

A note about using SSL/TLS

When using the Apache Portable Runtime based Native library for Tomcat (the default for this release) Apache Tomcat uses OpenSSL for SSL/TLS support as opposed to using the SSL/TLS facilities that are provided by Java, and consequently when using HTTPS it is necessary to use OpenSSL-generated certificate and key files, and to specify certificate and key file locations as illustrated in the example connector below, instead of using a Java keystore (replacing "device" and "path" as appropriate for your environment). VSI SSL111 is statically linked into the APR native library.

```
<Connector protocol="HTTP/1.1"
           connectionTimeout="20000"
           enableLookups="false"
           maxKeepAliveRequests="1000"
           port="8443"
           maxThreads="200"
           scheme="https"
           secure="true"
           SSLEnabled="true"
           SSLCertificateFile="/device/path/cert.pem"
           SSLCertificateKeyFile="/device/path/key.pem"
           SSLVerifyClient="none"
           SSLProtocol="TLSv1+TLSv1.1+TLSv1.2"
/>
```

You will also notice that in the above connector example that `SSLVerifyClient="none"` has been specified. It is recommended to use this setting if you do not expect clients to verify their identity.

Notes on using CGI with Tomcat on OpenVMS

The CGI (Common Gateway Interface) defines a way for a web server to interact with external content-generating programs, which are often referred to as CGI programs or CGI scripts.

By default CGI support is disabled in Tomcat. To enable CGI Support, one needs to perform the following Tasks:

1. Edit the `tomcat$root:[conf]web.xml` file. Locate the pre-existing (but commented out) section that defines the CGI environment, and uncomment the servlet configuration section. For example:

```
<!-- -->
<servlet>
  <servlet-name>cgi</servlet-name>
  <servlet-class>org.apache.catalina.servlets.CGIServlet</servlet-class>
    <init-param>
      <param-name>clientInputTimeout</param-name>
      <param-value>100</param-value>
    </init-param>
    <init-param>
      <param-name>debug</param-name>
      <param-value>6</param-value>
    </init-param>
    <init-param>
      <param-name>executable</param-name>
      <param-value></param-value>   <!--
Blank this value,  if using DCL or Executables for CGIs.-->
    </init-param>
    <init-param>
      <param-name>cgiPathPrefix</param-name>
      <param-value>WEB-INF/cgi</param-value>
    </init-param>
  <load-on-startup>5</load-on-startup>
</servlet>
```

Note that if the parameter-name "`Executable`" and the parameter-value resources are missing, you will want to add these in manually as shown above.

```
<init-param>
  <param-name>executable</param-name>
  <param-value></param-value>
</init-param>
```

If you are going to be using DCL or native executables as CGI programs, you will want to leave the "`param-value`" blank as seen above, otherwise you may receive an error message and the CGI scripts will fail to work.

2. Next you need to uncomment the "`cgi`" servlet-mapping definitions located just a little further down in the file. After making this change save your changes and exit the editor back to DCL.

```
<!-- The mapping for the CGI Gateway servlet -->
<!-- -->
    <servlet-mapping>
        <servlet-name>cgi</servlet-name>
        <url-pattern>/cgi-bin/*</url-pattern>
    </servlet-mapping>
```

3. Locate the file `tomcat$root:[conf]context.xml` and edit it. Look for the line that reads "`<Context>`" and modify it to include the following directives:

```
 <Context reloadable="true" privileged="true" >
```

4. Once you have these files modified to allow CGI scripting to function within Apache Tomcat on OpenVMS, you will want to create the following directory path, replacing "directory-owner" with the UIC or username of the owner of the `tomcat$root` directory tree.

```
$ set default tomcat$root:[webapps]
$ create/directory/owner=[directory-owner]  [.cgi.web-inf.cgi]
```

5. After making these changes and restarting the web server it should be possible to run CGI scripts from the CGI directory using URLs of the following form:

http://hostname:8080/CGI/cgi-bin/test.cgi