

Setting up the CI Pipeline Agent on IBM Z as a Started Task

Dennis Behm

dennis.behm@de.ibm.com

Mathieu Dalbin

mathieu.dalbin@fr.ibm.com



Abstract

Learn how to setup a CI pipeline agent as a Started Task. Sample RACF definitions provided to setup the necessary requirements to launch the agent.

Table of content

| | | |
|----------|---|-----------|
| 1 | INTRODUCTION | 3 |
| 2 | TRIGGERING BUILDS AS A CI PIPELINE USER..... | 4 |
| 3 | SETTING UP A TECHNICAL USER | 5 |
| 4 | RUNNING THE JENKINS AGENT AS A STARTED TASK..... | 6 |
| 4.1 | TWO METHODS TO LAUNCH THE JENKINS AGENT..... | 6 |
| 4.2 | CONFIGURING THE INBOUND AGENT METHOD FOR THE AGENT | 6 |
| 4.2.1 | <i>Defining the agent</i> | <i>6</i> |
| 4.2.2 | <i>Defining the TCP port.....</i> | <i>7</i> |
| 4.2.3 | <i>Retrieving the launch commands for the agent.....</i> | <i>7</i> |
| 4.2.4 | <i>Downloading the agent binary and transferring it to USS.....</i> | <i>7</i> |
| 4.2.5 | <i>Setting up RACF Security.....</i> | <i>8</i> |
| 4.2.6 | <i>Defining the STC procedure.....</i> | <i>8</i> |
| 4.2.7 | <i>Customizing USS and WLM parameters</i> | <i>10</i> |
| 4.2.8 | <i>Starting the agent.....</i> | <i>11</i> |
| 5 | APPENDIX..... | 12 |
| 5.1 | MANAGE SSL SECURITY BETWEEN TO THE JENKINS AGENT AND SERVER..... | 12 |
| 5.1.1 | <i>Use RACF keyrings to manage security certificates</i> | <i>12</i> |

1 Introduction

Several CI orchestrators, like Jenkins, or TeamCity, follow a server / agent architecture. The agent (also called the slave) runs on the target build machines. Some orchestrators, such as Jenkins, support z/OS as a target environment for the agent. In this document, we focus on z/OS and use Jenkins in the text as an example.

In a standard setup, the agent is launched from the server through SSH. Such a launch method is convenient as it is dynamic and does not require specific installation steps on the target environment. However, with this standard setup to launch the agent, the task is not easily identifiable within SDSF (System Display and Search Facility). Additionally, it appears as a Unix subtask of the user, which requires the TSO user ID and password stored on the Jenkins server to launch the agent.

Besides the standard setup, the agent can be launched as a Started Task (STC), which does not require SSH access. Therefore, credentials are not stored on Jenkins. In addition, the process can be embedded to the z/OS system automation.

This TechDoc outlines the required definitions to launch the agent as a Started Task (STC).

2 Triggering builds as a CI pipeline user

The pipeline is an automated process. The best practice is to create a CI pipeline user to drive the pipeline build on IBM Z as follows:

- It is expected that the CI pipeline jobs runs under a dedicated technical user ID.
- The agent has exclusive R+W permissions on the build workspaces in Unix System Services and the target build data sets. It is recommended that developers only have read permission on build workspaces and the build data sets.
- The agent performs all the necessary steps of the pipeline (building, code inspection, unit testing, packaging) under the authority of the technical user ID that launched it.
- It is recommended to use a dedicated HFS file system to manage the build workspaces to avoid impacting other activities on the system.

3 Setting up a technical user

This section provides a generic RACF definition for the technical user for the CI agent.

Environment prerequisites for the CI pipeline user are as follows:

- Minimum of 512 MB region size, 1 GB recommended.
- OMVS Segment
- To correctly execute some API of the DBB Toolkit to perform actions on JCLs, the CI pipeline user requires to have a TSO segment in his user definition.

```
ADDGROUP STCGROUP OMVS(GID(#group-id)) DATA('GROUP WITH OMVS SEGMENT FOR STARTED TASKS')
ADDUSER CIUSER DFLTGROUP(STCGROUP) NOPASSWORD NAME('CI AGENT') TSO(PROC(SYSUSER)
ACCTNUM(SYS0000) SIZE(240000) MAXSIZE(0)
UNIT(SYSALLDA) USERDATA(0000)) OMVS(UID(#user-id) HOME(#homeDirectory) PROGRAM(/bin/sh))
DATA('CI AGENT')

RDEFINE STARTED CIAGENT.* DATA('CI BUILD AGENT') STDATA(USER(CIUSER) GROUP(STCGROUP)
TRUSTED(NO))

SETROPTS RACLIST(STARTED) REFRESH
```

Comment: Ensure that the started task user ID is protected by specifying the NOPASSWORD keyword.

The provided generic RACF definitions should be tailored to meet the installation's requirements, by defining specific value for attributes `#group-id`, `#user-id` and `#homeDirectory`.

To be able to override jobnames of forked USS address spaces, the CIUSER requires READ permission to the BPX.JOBNAME resource in class FACILITY. Please see the restrictions for `_BPX` environment variables, so that `_BPX_JOBNAME` can be applied. Before the job name can be changed, the invoker must have appropriate privileges. The privileges include either superuser authority or READ permission to the BPX.JOBNAME FACILITY class profile¹.

The following RACF commands provide this privilege to the CIUSER:

```
PERMIT BPX.JOBNAME CLASS(FACILITY) ID(CIUSER) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

¹ https://www.ibm.com/support/knowledgecenter/SSLTBW_2.4.0/com.ibm.zos.v2r4.bpxb200/bpxenv.htm

4 Running the Jenkins agent as a Started Task

This section describes how to run the Jenkins agent on z/OS (USS) as a Started Task.

4.1 Two methods to launch the Jenkins agent

Jenkins offers two methods to start the agent:

- The Jenkins server establishes a connection to the target z/OS environment through SSH. This requires that the user ID and password are stored on the Jenkins server. Binaries are transferred to the target environment and the agent is started. This is the standard setup.
- Alternatively, the agent can be installed on the target z/OS environment and then establishes a connection to the Jenkins server. It does not require SSH access and credentials are not stored on the Jenkins server. This option is known as **launching an inbound agent**. For more information, see <https://github.com/jenkinsci/remoting/blob/master/docs/inbound-agent.md>

To run the Jenkins agent as a Started Task, it must be launched with the inbound agent method.

4.2 Configuring the inbound agent method for the agent

The following procedure describes how to set up a Jenkins inbound agent and configure it in z/OS.

4.2.1 Defining the agent

Logon to the Jenkins server and define a new agent. Set the launch method to **“Launch agent by connecting it to the master”** and uncheck the **“Use WebSocket”**.

Disabling the use of WebSocket requires to define a TCP Port, on which Jenkins Server will listen for incoming connections, which we will do in the next step.

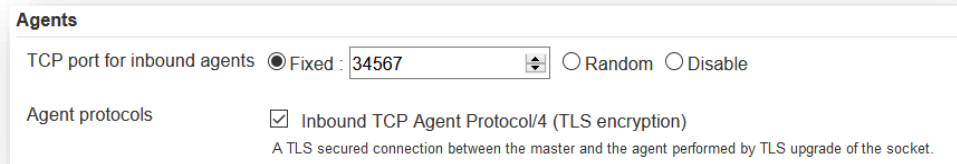


Warning: Please avoid the use of WebSockets. A loop has been detected in the implementation of the WebSocket on z/OS JVM, which causes high CPU consumption.

4.2.2 Defining the TCP port

To define the incoming TCP Port where Jenkins will listen, go to the Jenkins's Global Security Configuration (*Manage Jenkins --> Configure Global Security*).

In the *Agents* section, pick an available TCP port and check the use of TLS encryption:



The screenshot shows the 'Agents' configuration section in Jenkins. It features a form with the following elements:

- TCP port for inbound agents:** A radio button labeled 'Fixed' is selected, followed by a text input field containing '34567'. To the right are two unselected radio buttons labeled 'Random' and 'Disable'.
- Agent protocols:** A checked checkbox is followed by the text 'Inbound TCP Agent Protocol/4 (TLS encryption)'. Below this, a smaller line of text reads: 'A TLS secured connection between the master and the agent performed by TLS upgrade of the socket.'

4.2.3 Retrieving the launch commands for the agent

Return to the agent configuration page and retrieve the *launch command* provided by Jenkins to start the agent on the build machine. The commands include a secret and all necessary parameters.



The screenshot shows the 'Connect agent to Jenkins one of these ways:' section. It includes a 'Launch' button and a list of instructions:

- Launch agent from browser:** A button labeled 'Launch' with a small icon to its left.
- Run from agent command line:** A list of instructions including a Java command to run the agent jar with specific URLs and a secret, and a note about running from the command line with the secret stored in a file.

```
java -jar agent.jar -jnlpUrl http://10.3.20.156:8080/computer/EOLE53STC/slave-agent.jnlp -secret 9107e7f943f01cc0b88d636342bfd9cd273decf1641cc5cce69d76a75d2e1af6 -workDir "/var/jenkins"
```

```
Run from agent command line, with the secret stored in a file:  
echo 9107e7f943f01cc0b88d636342bfd9cd273decf1641cc5cce69d76a75d2e1af6 > secret-file  
java -jar agent.jar -jnlpUrl http://10.3.20.156:8080/computer/EOLE53STC/slave-agent.jnlp -secret @secret-file -workDir "/var/jenkins"
```

4.2.4 Downloading the agent binary and transferring it to USS

From the agent configuration page, download the .jar file, which might be shown as agent.jar or remoting.jar, and transfer it to the work directory in USS.

The agent JAR file must be present prior to starting the STC.

4.2.5 Setting up RACF Security

To comply with standard setup of z/OS Security Server depending on your installation and security requirements, define a RACF user to run the Started Task.

The following RACF commands define a specific group and a user 'JENKINS', with NOPASSWORD (to prevent this user to log on TSO), and associate this user to a resource of the STARTED class:

```
ADDGROUP JENKINSG OMVS(GID(14001)) DATA('Group for Jenkins agents')

ADDUSER JENKINS DFLTGRP(JENKINSG) NOPASSWORD NAME('Jenkins agent') OMVS(UID(14001)
HOME(/u/jenkins) PROGRAM(/bin/sh)) DATA('Jenkins Agent')

RDEFINE STARTED JENKINS.* DATA('Jenkins agent STC') STDATA(USER(JENKINS) GROUP(JENKINSG)
TRUSTED(NO))

SETROPTS RACLIST(STARTED) REFRESH
```

Please keep in mind this user doesn't have a TSO segment as it is not planned to run JCLs (with DBB's JCLExec API). Consider adding a TSO segment if you plan to use JCLExec API in your DBB implementation.

This user has its home directory in /u/jenkins, but all the files retrieved through the Jenkins agent will be located in the work directory on USS. This work directory should be defined before the agent accesses it. Also, ownership of the home directory and the work directory should be given to the 'jenkins' user defined through the Security Server commands shown above. Other users might need to be given read access to this work directory to check build results and logs of the CI pipeline.

4.2.6 Defining the STC procedure

Define a STC PROC in your preferred library of the PROCLIB concatenation. The EXEC statement is a call to BPXBATCH, which will in turn call your tailored shell script.

```
//JENKINS PROC
//*****
//RUN      EXEC PGM=BPXBATCH,DYNAMNBR=30,REGION=0M,TIME=1440,
//        PARM='SH /u/jenkins/start.sh'
//STDOUT   DD PATH='/u/jenkins/stdout',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//STDERR   DD PATH='/u/jenkins/stderr',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//STDENV   DD *
_BPX_JOBNAME=JENAGNT
_BPX_SHAREAS=YES
/*
```


Next, create the launch shell script `/u/jenkins/start.sh`. The following sample script `start.sh` contains the necessary commands to set up environment variables, including the launch command provided by Jenkins server to start the agent:

```
#!/bin/sh

. /usr/lpp/dbb/v1r0/conf/gitenv.sh
. /usr/lpp/dbb/v1r0/conf/dbbenv.sh
export JAVA_HOME=/usr/lpp/java/J8.0_64/
export IBM_JAVA_ENABLE_ASCII_FILETAG=ON
cd /var/jenkins

curl --output remoting.jar http://10.3.20.156:8080/jnlpJars/agent.jar
/usr/lpp/java/J8.0_64/bin/java -Dfile.encoding=ISO-8859-1 -Xnoargsconversion -jar
/var/jenkins/remoting.jar -jnlpUrl http://10.3.20.156:8080/computer/EOLE53STC/slave-agent.jnlp
-secret 9107e7f943f01cc0b88d636342bfd9cd273decf1641cc5cce69d76a75d2e1af6 -workDir
"/var/jenkins" -text
```

Tailor this shell script to add specific configurations, for instance invocations of additional scripts or defining new environment variables.

To download the latest version of the Jenkins agent from the Jenkins server, a `curl` command can be used, as shown in the above `start.sh` script. `Curl` is available as part of Rocket Open Source Languages and Tools for z/OS offering and should be available on the target system. Using this `curl` command in the startup script ensures compatibility between the Jenkins server and the Jenkins agent, which is downloaded every time the STC is recycled.

When using the latest versions of Jenkins, SSL handshake errors may occur when the Jenkins Agent tries to connect to the Jenkins server, when SSL is enabled. Typically, the agent on z/OS will try to use TLSv1 protocol, which is now superseded by TLSv2 and TLSv3. The following error message or similar is then displayed in the Jenkins server log:

```
javax.net.ssl.SSLHandshakeException: Client requested protocol TLSv1 is not enabled or
supported in server context
```

To manage SSL handshake errors that can occur when the Jenkins Agent on z/OS connects to the Jenkins Server, it is necessary to allow the use of newer TLS protocol at the IBM JVM on z/OS level. This can be performed by providing the `-Dcom.ibm.jsse2.overrideDefaultTLS=true` parameter in the `java` command used to start the Jenkins Agent. The startup script will now contain this additional parameter

```
/usr/lpp/java/J8.0_64/bin/java -Dfile.encoding=ISO-8859-1 -
Dcom.ibm.jsse2.overrideDefaultTLS=true
-Xnoargsconversion -jar /var/jenkins/remoting.jar -jnlpUrl
http://10.3.20.156:8080/computer/EOLE53STC/slave-agent.jnlp -secret
9107e7f943f01cc0b88d636342bfd9cd273decf1641cc5cce69d76a75d2e1af6 -workDir "/var/jenkins" -text
```

4.2.7 Customizing USS and WLM parameters

The build agent is expected to run with a high priority, as the triggering of builds is a critical part of a CI pipeline on mainframe. Consider setting the appropriate WLM settings for the STC.

Using BPXBATCH to launch the start script of the Jenkins Agent in USS will create a new address space, which uses the WLM OMVS classification rules. To facilitate the classification of jobs in WLM, two environment variables can be used: `_BPX_JOBNAME` to specify the name of the process, and `_BPX_ACCT_DATA` to specify the accounting information of the process. In the sample JENKINS PROC listed above, the `_BPX_JOBNAME` parameter for the new forked process is specified in the inline STDENV DD card and has the value JENAGNT.

In this configuration, jobname of all subsequent forks created by the Jenkins agent will start with JENAGNT prefix, followed by a number from 1 to 9.

To identify the forked processes of the Jenkins agent and to relate them to Jenkins jobs, the `_BPX_JOBNAME` environment variable can be used. This can be done in the Jenkins pipeline definition by providing a value to the `env._BPX_JOBNAME` variable:

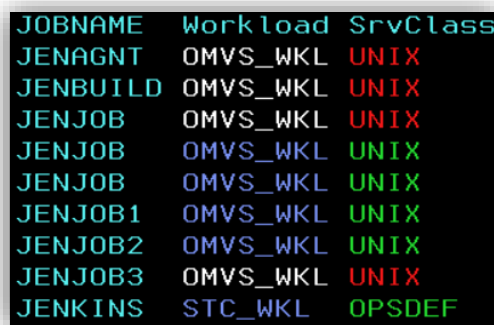
```
env._BPX_JOBNAME = "JENJOB"
```

Additionally, consider setting the `_BPX_JOBNAME` environment variable in groovy script to highlight the Dependency-Based Build operations in the Jenkins pipeline:

```
export _BPX_JOBNAME=JENBUILD
```

Optionally, set the environment variable `_BPX_SHAREAS` to YES, which allows the sharing of address space for forked USS processes. Please be aware that the execution of groovy will set `_BPX_SHAREAS` for this address space to NO, to support Multi-threaded MVSJob operations in subsequent forked processes.

See below the results of this customization as displayed in SDSF, during the execution of a Jenkins pipeline building a z/OS application with Dependency-Based Build:



| JOBNAME | Workload | SrvClass |
|----------|----------|----------|
| JENAGNT | OMVS_WKL | UNIX |
| JENBUILD | OMVS_WKL | UNIX |
| JENJOB | OMVS_WKL | UNIX |
| JENJOB | OMVS_WKL | UNIX |
| JENJOB | OMVS_WKL | UNIX |
| JENJOB1 | OMVS_WKL | UNIX |
| JENJOB2 | OMVS_WKL | UNIX |
| JENJOB3 | OMVS_WKL | UNIX |
| JENKINS | STC_WKL | OPSDEF |

In this configuration, JENKINS is the initial Started Task, JENAGNT is the address space executing the Jenkins agent, JENJOB address spaces are executing tasks of the Jenkins pipeline and JENBUILD is executing the groovy script corresponding to the Dependency-Based Build operation.

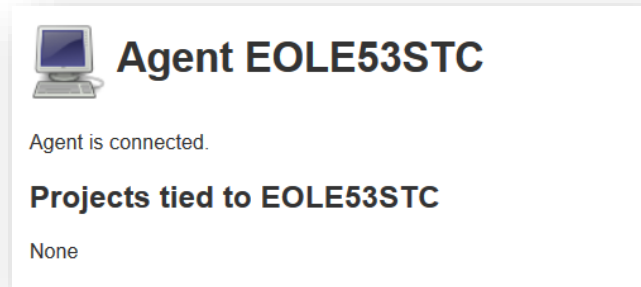
To meet with your WLM specifications, `_BPX_JOBNAME` and `_BPX_ACCT_DATA` can be used to filter and define WLM policies and goals.

4.2.8 Starting the agent

Start the STC by issuing the MVS START command: **/S JENKINS** command:

```
JENKINS  JENKINS  RUN          STC07054 STC          LO  FF  315   0.00  0.00
JENKINS1 *OMVSEX     STC07055 STC          LO  FF  366   0.00  0.00
JENKINS2 STEP1     STC07056 STC          LO  FF 2537   0.00  0.00
JENKINS3 *OMVSEX     STC07057 STC          IN  F0  21T   0.00  0.00
```

Within the Jenkins user interface, the agent is now listed as connected in the Jenkins server.



The agent can now be used in subsequent Jenkins pipelines, to perform actions on z/OS USS, such as Git operations, DBB builds, and shell script invocations.

To stop the agent, just cancel the Started Task on z/OS.

5 Appendix

5.1 Manage SSL security between to the Jenkins agent and server

In the described setup above, the Jenkins agent establishes a connection to the Jenkins server instance over secure HTTP. For the secure connection, it requires to trust the certificate which is presented by the server.

It works basically like surfing on the internet, when the server presents its certificate, and the browser validates it against the truststore of certificates which got signed from the certificate authority (CA). The Java JVM comes with a `cacerts` file which contains default certificates which got signed by the certificate authority.²

Please work with your Java security administrator to add in-house certificates to the `cacerts` file using the `keytool` command.

Some companies might consider managing the certificates within RACF, which provides this capability through RACF keyrings. The next section explains how to configure the Jenkins start script to direct the verification step to RACF instead of the internal Java security facility.

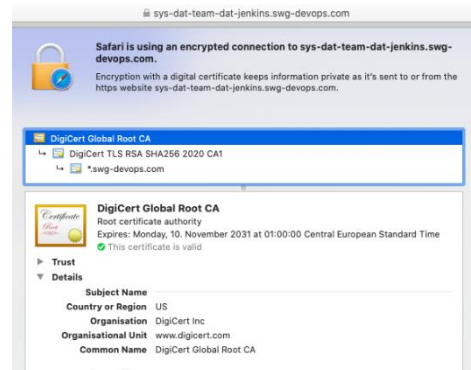
5.1.1 Use RACF keyrings to manage security certificates

To use RACF keyrings, it requires some configurations in RACF as well as modifying the Jenkins agent's `start.sh` script to use those.

As mentioned above, the process validates the certificate presented by the server against a known truststore. So, before setting up the RACF keyring, it first requires making the trusted root CA certificate available within RACF.

To identify which root CA is in use, the information collected through a web browser can be leveraged. In most of web browsers, the certificates information is available by clicking on the lock symbol next to the URL bar.

The next paragraph explains how to obtain the root CA certificate from the existing `cacerts` file from your Java installation. Please note that obtaining the root CA certificate will vary depending on the situation.



Prior to exporting the root CA certificate from the existing `cacerts` file, it is necessary to identify the alias in use in your `cacerts` file, with the help of the `list` subcommand of the `keytool` utility³. The `keytool` utility is available on several platforms. The below samples were executed on Unix System Services.

```
keytool -list -keystore /usr/lpp/java/J8.0_64/lib/security/cacerts -storepass changeit -v
```

² <https://www.ibm.com/docs/en/sdk-java-technology/8?topic=certificate-cacerts-certificates-file>

³ <https://www.ibm.com/docs/en/sdk-java-technology/8?topic=keytool-key-certificate-management-tool>

When the alias has been identified, the `export` subcommand of the `keytool` utility must be executed to export the certificate:

```
keytool -exportcert -keystore /usr/lpp/java/J8.0_64/lib/security/cacerts -file /u/ibmuser/cacert-export -storepass changeit -alias digicertglobalrootca
```

In the next step, the exported certificate must be copied from USS to a sequential dataset:

```
cp cacert-export "//ibmuser.cacert"
```

Adding the certificate in RACF can be performed from the ISPF Option 6 menu, by running the `RACDCERT`⁴ command:

```
racdcert add('ibmuser.cacert') certauth trust withlabel('cicdca')
```

As the certificate is now available in RACF, a keyring can be created, and the certificate can be connected to this keyring. These actions can be performed through the RACF ISPF interface.

To create the keyring, the option `7 DIGITAL CERTIFICATES, KEY RINGS, AND TOKENS` on the RACF primary screen is used:

```

                                     RACF - SERVICES OPTION MENU

SELECT ONE OF THE FOLLOWING:

  1  DATA SET PROFILES
  2  GENERAL RESOURCE PROFILES
  3  GROUP PROFILES AND USER-TO-GROUP CONNECTIONS
  4  USER PROFILES AND YOUR OWN PASSWORD
  5  SYSTEM OPTIONS
  6  REMOTE SHARING FACILITY
  7  DIGITAL CERTIFICATES, KEY RINGS, AND TOKENS
 99  EXIT

                                     Licensed Materials - Property of IBM
                                     5650-ZOS Copyright IBM Corp. 1983, 2017
                                     All Rights Reserved - U.S. Government Users

OPTION ==> 7_
```

⁴ <https://www.ibm.com/docs/en/zos/2.5.0?topic=certificates-racdcert-add-add-certificate>

The option **2. Key Ring Functions** is then used to access keyring-specific actions:

```
RACF - Digital Certificates and Related Functions
OPTION ==> 2_

Select one of the following:

1. Digital Certificate Functions
2. Key Ring Functions
3. Certificate Name Filtering Functions
4. Token Functions
```

A new keyring is created for the user assigned to the Started Task which runs the Jenkins agent:

```
RACF - Digital Certificate Key Ring Services
OPTION ==> 1_

For user: STCUSER_

Enter one of the following at the OPTION line:

1 Create a new key ring
2 Delete an existing key ring
3 List existing key ring(s)
4 Connect a digital certificate to a key ring
5 Remove a digital certificate from a key ring
```

The keyring name (case sensitive) is specified on the following screen:

```
          RACF - Digital Certificate Key Ring Name
COMMAND ===>

Enter a ring name:

CICDRING_
_____
_____
_____

A ring name may not be a single asterisk * and
blanks are not allowed.
```

To add the certificate to the keyring, the option *4 Connect a digital certificate to a key ring* of the keyring submenu is used. Again, the user ID which runs the Started Task is supplied.

```
          RACF - Digital Certificate Key Ring Services
OPTION ===> 4_
For user: STCUSER_

Enter one of the following at the OPTION line:

1 Create a new key ring
2 Delete an existing key ring
3 List existing key ring(s)
4 Connect a digital certificate to a key ring
5 Remove a digital certificate from a key ring
```

The keyring name (case sensitive) that was just created is specified, with additional parameters related to the usage of the certificate:

```
RACF - Connect a Digital Certificate to a Key Ring
COMMAND ==>

Ring Owner: STCUSER

Ring Name: CICDRING _____
          _____
          _____

Certificate Type => Personal
                  (user ID)      or Site      or Certificate Authority
                  _____      => _         => x
Label name: 'cicdca' _____ (in quotes)

Usage          => Personal      or Site      or Certificate Authority
              => _             => _         => x _
Default       => _ (blank defaults to NO)
```

Finally, the startup script `start.sh` for the Jenkins Agent is tailored to instruct the JVM to use the RACF keyring. The following settings, once adapted to your local configuration, can be used as options to the JVM when launching the Jenkins agent.

```
-Djavax.net.ssl.keyStore=safkeyring://STCUSER/CICDRING \  
-Djavax.net.ssl.keyStoreType=JCERACFKS \  
-Djavax.net.ssl.trustStore=safkeyring://STCUSER/CICDRING \  
-Djavax.net.ssl.trustStoreType=JCERACFKS \  
-Dcom.ibm.ssl.keyStoreFileBased=false \  
-Dcom.ibm.ssl.trustStoreFileBased=false \  
-Djava.protocol.handler.pkgs=com.ibm.crypto.provider \  
-Djavax.net.ssl.keyStorePassword=password \  

```

The server certificate will now be validated against the certificate stored in the RACF keyring.