Java Update

2024 TPF Users Group Conference
May 5-8, New Orleans, LA

Applications Subcommittee

—

Dan Gritter

IBM Z

Disclaimer

Agenda

IBM Semeru Runtime

z/TPF Release Calendar

JAM Operations

JAM Recycle Service

JAM Startup Enhancements

Java Dump Enhancements for z/TPF

ZFILE sudo Support

Build and Development

Maven Tooling Enhancements

IBM Semeru Runtime

z/TPF Release Calendar

IBM Semeru Runtime on z/TPF Release Calendar

LTS Java Version	Refresh Version *	Release Date	EOS
8	8.0.8.5 8.0.8.15 (Last Java 8 Refresh, 4Q 2024) PJ47239 (Remove Java 8, 2Q 2025)	Oct 2023 Jan 2024	Dec 31 2023
11	11.0.19.0 11.0.21.0 11.0.22.0 11.0.23.0 11.0.24.0 …	July 2023 Dec 2023 1Q2024 2Q2024 3Q2024	2H 2026
21	21.0.…	1Q 2025	TBD

* Vulnerability deadlines permitting, the lab may skip a Refresh to lower maintenance burden for community

JAM Operations

JAM Recycle Service

JAM Startup Enhancements

Java Dump Enhancements for z/TPF

ZFILE sudo Support

JAM Recycle Service

Problem Statement:

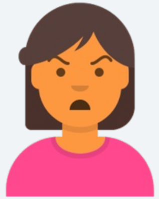
z/TPF JAM support recycles on any loadset activation or deactivation, which leads to unnecessary consumption of system resources when none of the programs or files in the loadset are used by a JAM.

Pain Points

Calvin
Capacity planner

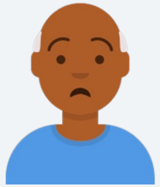
Undesirable for production systems during peak traffic hours

Sophie
System programmer



Disruptive for shared test systems with frequent E-type loader activity or limited resources

Derrick
Operator



Manual JAM recycling is required to avoid unnecessary disruptions

Background

Historically, a handful of long-running applications used the E-type loader recycle interface to determine a program changed in a loadset so that the application needed to recycle to pick up the changes.

If programs of interest did not change in the loadset, then the long running process continued to run and the system incremented the ECB activation number to the current system activation number.

Background

With the introduction of the version control file system and common deployment there has been an increase in the use of files in general by applications (DFDL, REST, and Java). The increase in file usage drove the need to enhance existing E-type loader recycle support to include files in addition to programs when checking to see if a recycle is required.

APAR PJ46750 (Nov 2022)

The E-type loader recycle interface was enhanced with the capability to monitor file changes in loadsets.

tpf_etype_loader_recycle_interface()

JAM Recycle Service (PJ47021 July 2023)

Value Statement:

JAM recycle service extends existing the JAM automatic recycle support to trigger recycles for only relevant programs, files, and directories. The support reduces unnecessary consumption of system resources when none of the programs or files in the loadset are used by this JAM.

Technical Details

Recycle coordination for JAMs is managed by the JAM monitor

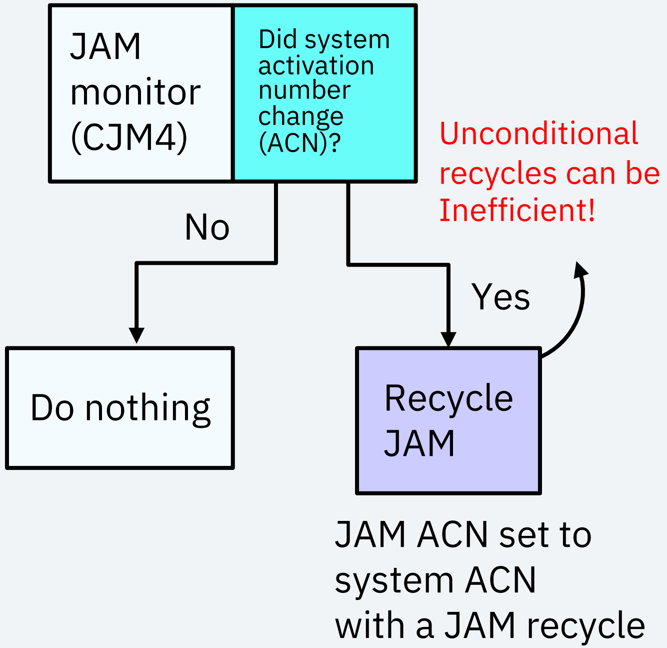
- The monitor maintains a JAM control table with the expected state of every JAM cluster.
- The monitor is constantly verifying the expected states of all the JAM clusters and is checking the system activation number.
- Signals are used to check and change states.

With PJ47021, the JAM monitor was updated to use existing JAM REST infrastructure (based on JAX-RS) to coordinate recycling.

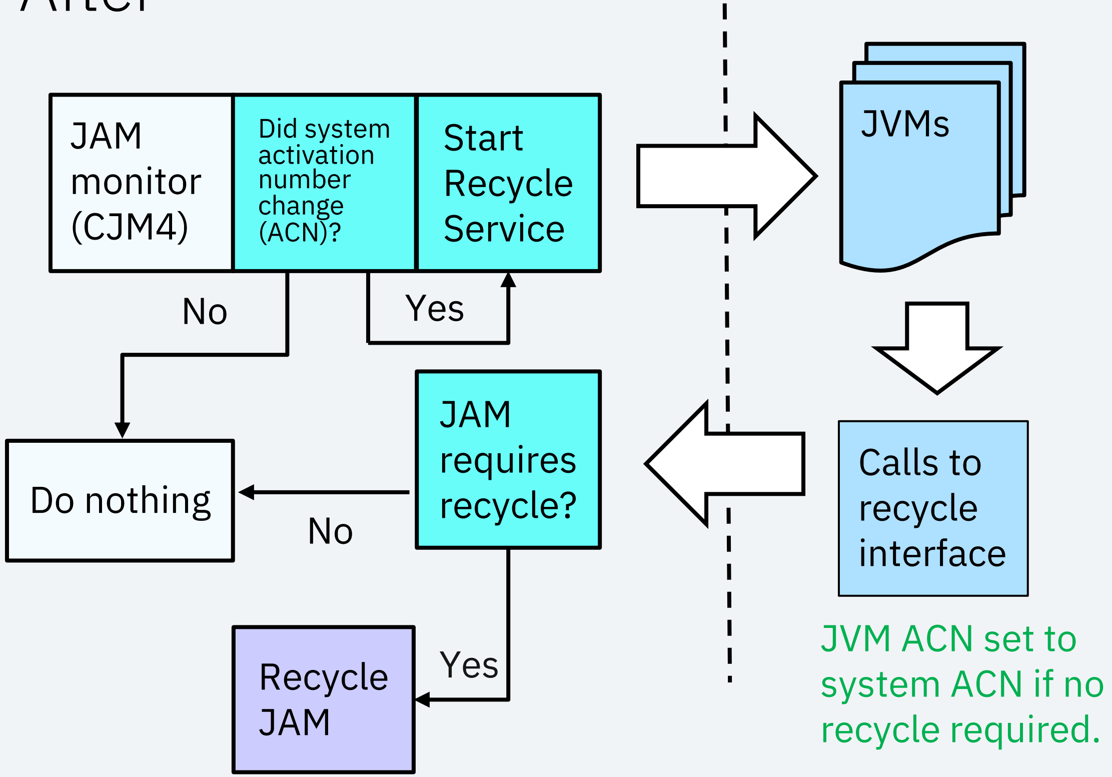
- The extended tpf_etype_loader_recycle_interface is called to monitor JAM artifacts.
- The support provides both JAM-scoped and subsystem-scoped monitoring for programs, files, or directories.

JAM Recycle Service (PJ47021)

Before


JAM monitor (CJM4)
Did system activation number change (ACN)?
No
Yes
Do nothing
Recycle JAM
Unconditional recycles can be Inefficient!
JAM ACN set to system ACN with a JAM recycle

After


JAM monitor (CJM4)
Did system activation number change (ACN)?
Start Recycle Service
JVMs
No
Yes
Calls to recycle interface
JAM requires recycle?
Do nothing
No
Yes
Recycle JAM
JVM ACN set to system ACN if no recycle required.

More Details - Relevant Programs and Files to Monitor for Recycles

Java refresh updates
JVM classpath (does not include dependency monitoring)
JAM infrastructure related programs & files
JAM descriptors
Application OpenAPI descriptors
Application service descriptors
Application service programs
Application property files
Other (well-known system configuration files, custom application files)

Automatic discovery

vs

User specification

More Details: User Customizations for Enhanced JAM Recycling

```
Example:  myJam.jam.xml

<tns:AutoStart>  YES_NORM  </tns:AutoStart>
<tns:AutoRecycle> YES  </tns:AutoRecycle>
<tns:AutoRestart> YES  </tns:AutoRestart>
<tns:AutoRecyclePrograms> QZZ6 QZZ7 </tns:AutoRecyclePrograms>
<tns:AutoRecycleFiles> /sys/tpf_pbfiles/tpf-fdes/myAPI.openapi.json /sys/tpf_pbfiles/tpf-
fdes/myService.json.srvc  </tns: AutoRecycleFiles>
<tns:AutoRecycleDirs> /usr/mykeys </tns:AutoRecycleDirs>
<tns:NumberJVMs>    2  </tns:NumberJVMs>
<tns:NumberThreadsPerJVM> 10 </tns:NumberThreadsPerJVM>
<tns:ApplicationList>
<tns:Application>
```

JAM Scoped!

/sys/tpf_pbfiles/apps/tpfjax/jam.mon

Subsystem Scoped!

```
// Plain File - One per line

QZZ6
/usr/mykeys
/sys/tpf_pbfiles/tpf-fdes/myAPI.openapi.json
```

New User Exits

🟦 = Existing 🟩 = New with JAM recycle service

User Exit	Called	Potential Use Cases
UJAM (as Entry Point)	When each JVM of a JAM starts	Enabling debug options

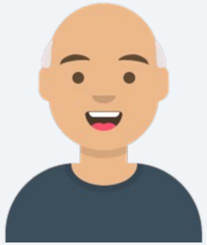
cjvm_jvm_startup_hook()
cjvm_jvm_shutdown_hook()
cjvm_thread_startup_hook()
cjvm_thread_shutdown_hook()
cjvm_dumpagent_preHook()
cjvm_dumpagent_postHook()
	Early in every JVM startup
	Before every JVM shutdown
	Before thread startup
	Before thread shutdown
	Before each dump type runs
	After each dump type runs
		Custom data collection or custom resource monitoring.
		Designate low priority JIT or monitoring threads.
		Diagnostic file renaming or transferring off z/TPF

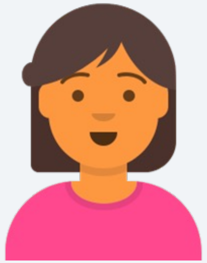
ujam_recycle_prehook()
ujam_recycle_postHook()
	Before JAM recycles
	After JAM recycles
		Save and restore JAM state information on the system.

ujam_stop_prehook()	Before JAM stop	Notify that JAM is stopping.

Conclusion

JAM recycle service built on existing JAM and loaders infrastructure and improves JAM operations experience. System resources are more intelligently managed when it comes to JAM recycling based only on relevant program and file changes. This can be especially helpful for test systems with limited capacity that have frequent OLDR loads.


Calvin
Capacity planner


Sophie
System programmer

Derrick
Operator

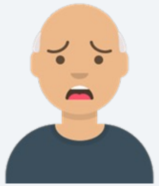
JAM Startup Enhancements

Problem Statement:

z/TPF JAM automatic start processing starts JAM clusters serially, in an arbitrary order, and with a Normal ECB priority. Approach ensures system doesn't get overwhelmed but doesn't take advantage of extra I-stream engines or support configuring JAM priority based on workload type.

Pain Points

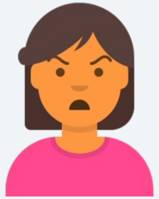
Calvin
Capacity planner



Minimizes JAM utilization of System Recovery Boost

https://www.ibm.com/docs/en/ztpf/2024?topic=functions-system-recovery-boost

Sophie
System programmer



Delays JAMs required for transactions to start

Anna
Application architect

Transactional work can be affected by JAMs not requiring normal ECB priority

Background

JAM Descriptor

```
Example:  myJam.jam.xml

…
<tns:JVMCommandLineOptions>
   …
</tns:JVMCommandLineOptions>
<tns:AutoStart>  YES_NORM  </tns:AutoStart>
<tns:AutoRecycle> YES  </tns:AutoRecycle>
<tns:AutoRestart> YES  </tns:AutoRestart>
<tns:NumberJVMs>    2  </tns:NumberJVMs>
<tns:NumberThreadsPerJVM> 10 </tns:NumberThreadsPerJVM>
…
```

Specifies whether to automatically start the JAM cluster on cycle up to NORM state.

Specifies whether to automatically recycle the JAM cluster when the system activation number changes.

Specifies whether to automatically restart the JVM when it exits programmatically or exits because of an abend.

ALL JVMs in a JAM run with Normal Execution Priority

JAM Startup Enhancements (PJ47243 April 2024)

Value Statement:

With z/TPF JAM startup enhancements, JAM clusters and JVMs within the JAM cluster can start in parallel. Additionally, ECB priority will be used to distinguish JAM workloads to avoid disrupting transactional workloads.

Details: Customizations for JAM Startup Enhancements

Subsystem Scoped!

/sys/tpf_pbfiles/apps/tpfjax/jam.env

To enable support

```
// Plain Environment Variable File - One per line

PATH=/sys/tpf_pbfiles/opt/ibm/ibm-semeru-certified-11-jdk/bin
TPF_AUTOSTART_TYPE=PARALLEL
```

Example JAM Descriptor: myJam.jam.xml

```
<tns:AutoStart>  YES_NORM  </tns:AutoStart>
<tns:AutoStartGroup> LOW </tns:AutoStartGroup>
<tns:AutoRecycle> YES  </tns:AutoRecycle>
<tns:AutoRestart> YES  </tns:AutoRestart>
…
<tns:NumberJVMs>    2  </tns:NumberJVMs>
<tns:NumberThreadsPerJVM> 10 </tns:NumberThreadsPerJVM>
<tns:JVMInitECBPriority> LOW </tns:JVMInitECBPriority>
<tns:JVMWorkECBPriority> NORMAL </tns:JVMWorkECBPriority>
…
<tns:ApplicationList>
<tns:Application>
```

Specifies whether this JAM will be started in the first group (normal priority) or in the second group (low priority)

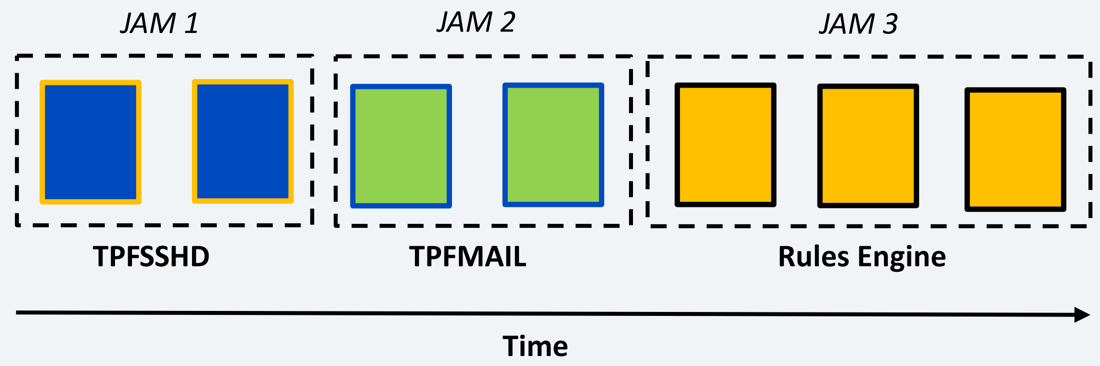
Specifies whether the JVM ECBs run as normal or low ECB priority when the JVM is starting

Specifies whether the JVM ECBs run as normal or low ECB priority when processing work

Startup Order Comparison

As-Is

JAM Serialized AutoStart

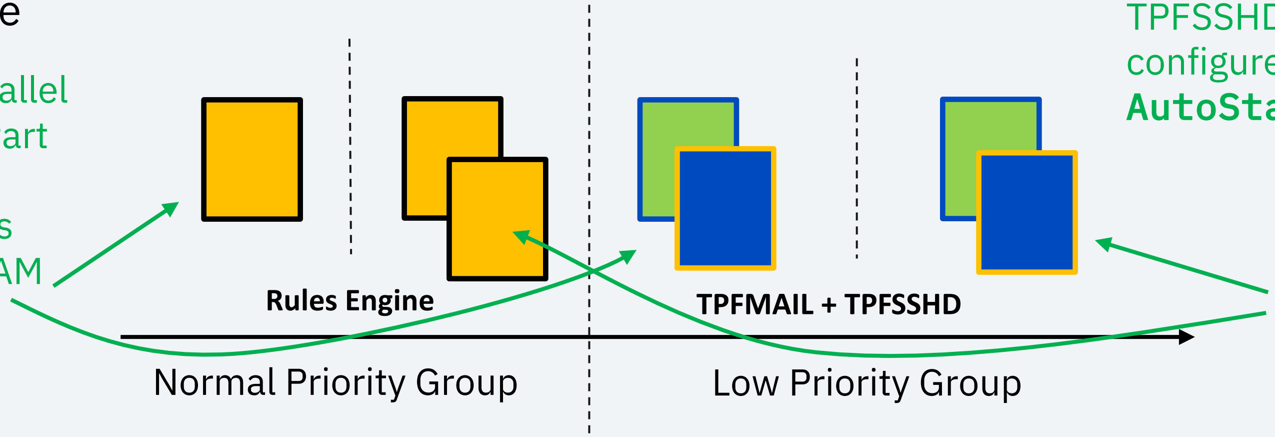


Order is arbitrary!

What if rules engine needs to be available first?

To-Be

JAM Parallel AutoStart



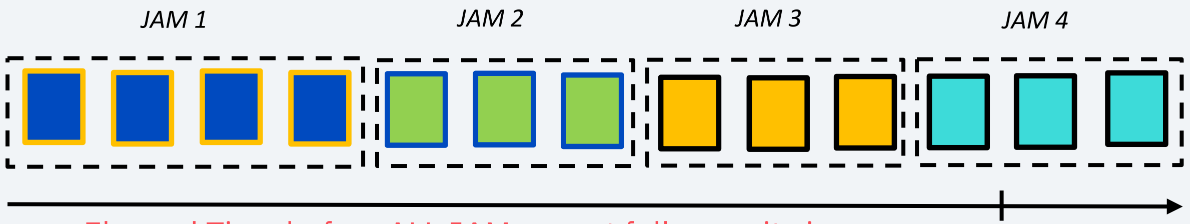
TPFSSHD and mail are configured with **`AutoStartGroup=LOW`**

First JVMs of each JAM start first

Remaining JVMs for each JAM start second

Elapsed Time Comparison for IBM z15 with 10 shared I-Streams

JAM Serialized AutoStart

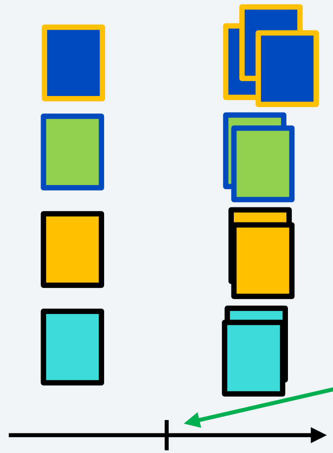


Elapsed Time before ALL JAMs are at full capacity is **90 seconds**

Elapsed Time for at least one JVM started for all JAMs is 54 seconds

JAM Parallel AutoStart

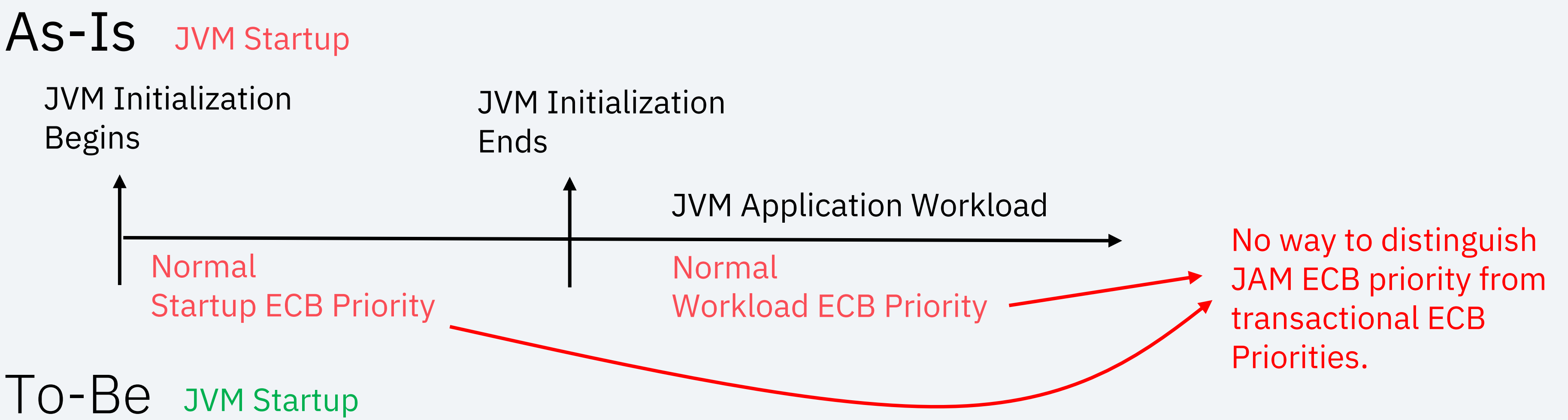
All 4 JAMs are configured with AutoStartGroup=NORMAL (default)

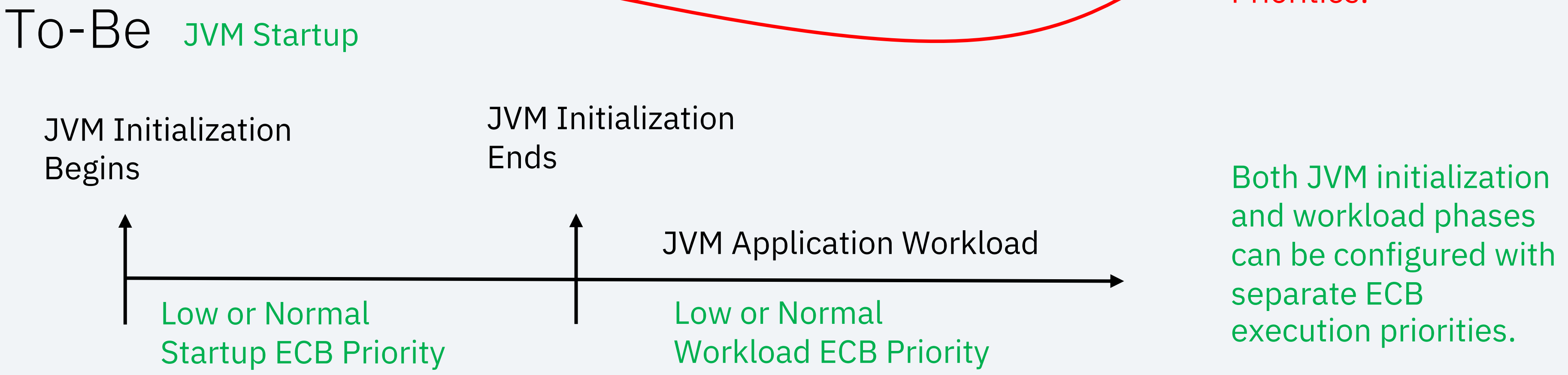


Elapsed Time for at least one JVM started for all JAMs is 16 seconds

Elapsed Time before for ALL JAMs are at full capacity is **30 seconds**

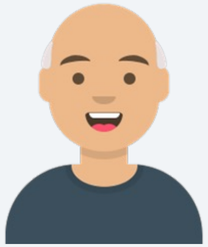
ECB Priority Updates

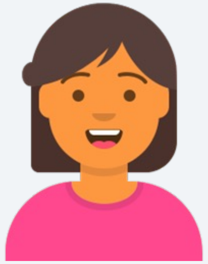

As-Is
JVM Startup
JVM Initialization Begins
JVM Initialization Ends
JVM Application Workload
Normal Startup ECB Priority
Normal Workload ECB Priority
No way to distinguish JAM ECB priority from transactional ECB Priorities.


To-Be
JVM Startup
JVM Initialization Begins
JVM Initialization Ends
JVM Application Workload
Low or Normal Startup ECB Priority
Low or Normal Workload ECB Priority
Both JVM initialization and workload phases can be configured with separate ECB execution priorities.

Conclusion

JAM startup enhancements improves use of I-stream capacity during JAM startup and provides the capability to distinguish between transactional workloads and lower value JAM workloads.


Calvin
Capacity planner


Sophie
System programmer

Anna
Application architect

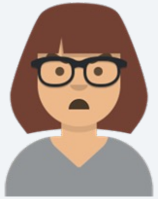
Java Dump Enhancements for z/TPF

Problem Statement:

If a JVM in a JAM has an unexpected failure it can take 10s of seconds if not minutes to write out diagnostics before starting recovery.

Pain Points

Anna
Application architect



Undesirable delay in restarting failing JVM

Background

Java Diagnostics Created by Different Dump Agents

Dump Agent (default order)	Type of Diagnostic	Post Processing Tools	Size (Time to Write)
System Dump	Process image (includes all heap, stacks, program static storage)	jdmpview	Process image size + size of shared cache, < 2 min
Java Dump	JVM summary (includes Java callstacks, summaries of heap use, lock use, class use)	Any text editor	< 3 MBs, < 30 sec
Heap Dump	Java object map, useful for finding Java object leaks	GMVC, Eclipse	< 1 sec
Snap Dump	Similar to z/TPF function trace	monitoring-api.jar	< 1 sec
Jit Dump	JIT compilation details	Any text editor	< 1 sec

Java Dump Enhancements for z/TPF (PJ47169 Dec 2023)

Value Statement :

With Java Dump Enhancements for z/TPF, a JVM can recover within 1 or 2 seconds instead of 10’s of seconds or minutes.

Technical Details

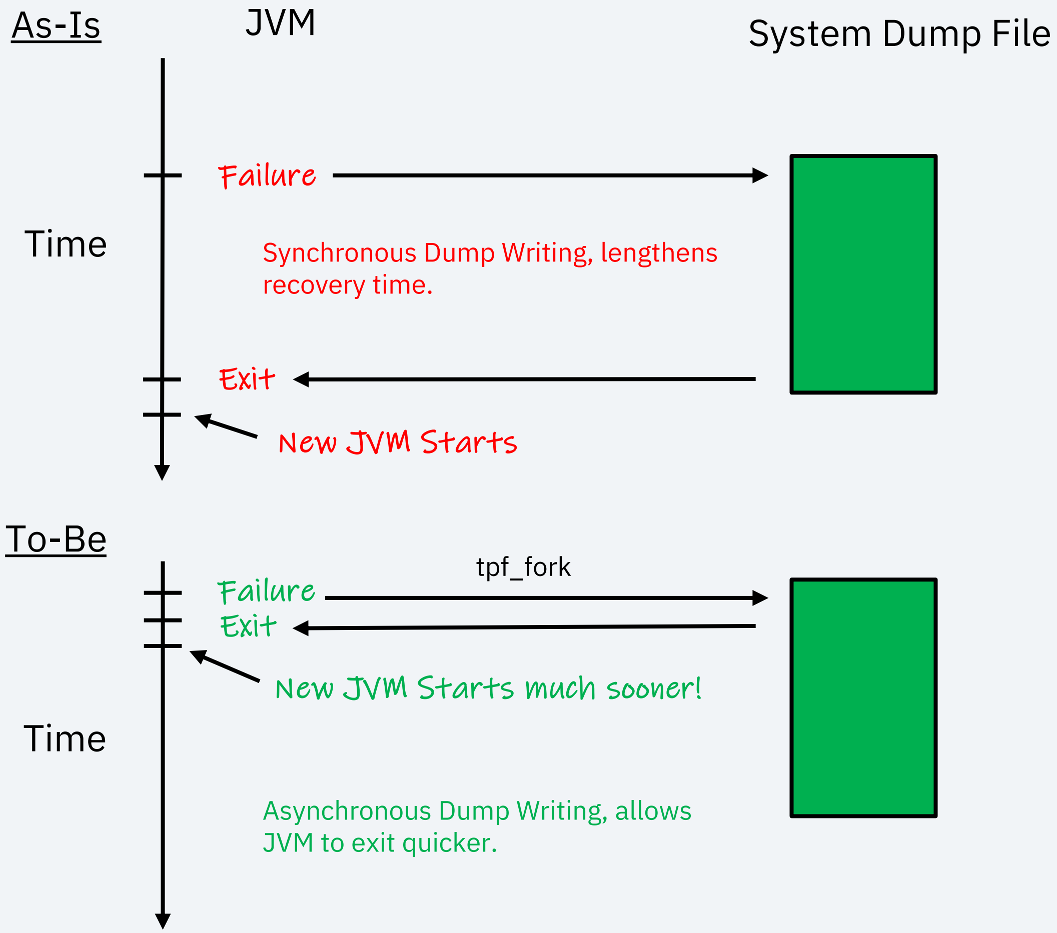
Default behavior is changed to enable asynchronous system dumps. Set property below to false to revert behavior:

```
-Dcom.ibm.tpf.jvm.asynchSysDumps=false
```

Java dump ECB priority still controlled with Java property

```
-Dcom.ibm.tpf.jvm.lowPriorityDumps
```

Support doesn't require configuration change to existing Java dump buffer, but size on disk will be about 100 MB larger.


As-Is
JVM
System Dump File
Time
Failure
Synchronous Dump Writing, lengthens recovery time.
Exit
New JVM Starts
To-Be
tpf_fork
Failure
Exit
New JVM Starts much sooner!
Time
Asynchronous Dump Writing, allows JVM to exit quicker.

Additions to JavaCore

- EMPS Setting (PJ47169)
- MAXXMMES Setting (PJ47169)
- MTHD Setting (PJ48008)

Easy confirmation on Java application system resource requirements.

Additional keypoint values to assist with problem determination.

```
========================================
Native configuration information for z/TPF
========================================
Highwater mark 64-bit heap 1MB Frames (limited by MAXXMMES): 49
MAXXMMES keypoint A setting:  100
Highwater mark 64-bit MMAP heap 1MB Frames (limited by MAXMMAP):    245
maxmmap process setting (either keypoint A MAXMMAP or com.ibm.tpf.maxmmap property):    800
Total highwater mark 64-bit heap 1MB Frames (limited by MAXXMMES+MAXMMAP):    294
Total highwater mark 31-bit heap 1MB Frames (limited by EMPS):    19
EMPS keypoint A setting:  70
Total highwater mark GC heap 1MB Frames (limited by -Xmx):        119
...
MTHD Keypoint A setting:  200
```

Conclusion

JAM dump enhancements for z/TPF improves both diagnostic and operations capability. Moving to an asynchronous system dump model provides support for the new JVM to recover much faster.

Anna
Application architect

ZFILE sudo Support

Problem Statement:

If you are using a prime CRAS, or an alternate CRAS with file system security disabled, you cannot run a ZFILE command as a nonroot user.

Background

On prime CRAS consoles, all `ZFILE` commands run as the root user.

For alternate CRAS consoles, `ZFILE` commands run as root by default. If file system security is enabled, you can run as a different user by using the ZPVFS command.

Background

Carol
Coverage programmer

Carol wants to use the Health Center to diagnose performance issues on a JAM in production.

Carol's enterprise does not use file system security. They manage operator command security by using other means.

As-is

Carol
Coverage programmer

All JVMs for JAM clusters run as the `tpfdfltu` user. The Health Center application must run as the same user that is running the monitored JVM.

Because Carol's system does not have file system security enabled, she cannot log in as `tpfdfltu`.

Problem Statement

Carol cannot run the Health Center application as the same user that is running the application she's trying to investigate.

As a result, Carol cannot use the Health Center to diagnose performance issues.

ZFILE sudo

PJ47113 introduces support for the `ZFILE sudo` command.

If you are logged in as root, you can use the `ZFILE sudo` command with the `-u` parameter to run any other `ZFILE` command as a specified user.

A nonroot user cannot use `ZFILE sudo` to elevate their own permissions.

To-be

Carol

Coverage programmer

Carol can attach the Health Center to her Java application, and can now diagnose the application's performance issues.

```
ZFILE sudo –u tpfdfltu java
       -jar healthcenter.jar ID=pid
```

Build and Development

Maven Tooling Enhancements

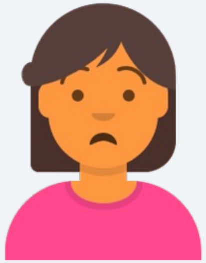
Maven Tooling Enhancements

Problem Statement:

The process for developing and maintaining Java applications that are built by using Maven for z/TPF is often inefficient and prone to inconsistencies.

Hannah

Build manager



Sophie

System programmer

APAR PJ47157 Oct 2023

Download script can consume 30 minutes or more per run and is prone to download failures.

As-Is	To-Be
The download_deps script preemptively downloads all z/TPF product dependencies.	Only download what is needed during subsequent reruns of this step, dramatically eliminates download failures.
Creating or updating Java applications that are built by using Maven potentially requires changing a dependency version used by other applications.	Instead of having to change this version for multiple applications, this now happens in one location for all applications.
If a Java application fails to compile, it can take time to diagnose the problem by using multiple commands and searching through the TPF_ROOT layered build environment.	New configuration properties are included in the INFO level build output to better help identify issues. Additionally, the checkenv target was enhanced to aggregate Maven related build configurations reducing time to gather diagnostic information.

Technical Details

- The download_deps.* sample scripts no longer trys to re-download files already downloaded.

```
PS C:\Users\MyUserName\Desktop> .\download_deps.ps1 –h

Usage: C:\Users\MyUserName\Desktop\download_deps.ps1 [-h] [-f] [-r <repo_url>] [-l C:\dep\list.txt] [-o C:\out\dir] [-a <attempts>]

  Options:
    -h                  Display this help message
    -f                  Force remove -o C:\out\dir if it previously exists
    -r <repo_url>       URL where central repository resides
                        Default: https://repo.maven.apache.org/maven2
    -l C:\dep\list.txt  Text file listing the dependencies required
                        Default: C:\Users\MyUserName\Desktop\dependencies.txt
    -o C:\out\dir       Directory where dependencies should be downloaded to
                        Default: C:\Users\MyUserName\Desktop\repository
    -a <attempts>       Number of times to attempt to download each dependency
                        Default: 2

PS C:\Users\MyUserName\Desktop> .\download_deps.ps1
In the C:\Users\MyUserName\Desktop\repository directory:
2398/2402 items were previously downloaded
4/4 missing items were newly downloaded
```

Technical Details

- MakeTPF automatically enables use of global properties files for use within a POM configuration.

  Example tpftools/include_ztpf/maketpf.rules_maven_tpf_properties:

  ```
  tpf.log4j.version=2.17.1
  ```

  Example base/tpfjax/pom.xml:

  ```
  <dependency>
     <groupId>org.apache.logging.log4j</groupId>
     <artifactId>log4j-api</artifactId>
     <version>${tpf.log4j.version}</version>
  </dependency>
  ```

  Similar maketpf.rules_maven_user_properties file for application only properties

Technical Details

- MakeTPF adds new diagnostic output at the INFO level.

  Example maketpf.cfg:

  ```
  MAKETPF_MSG_LEVEL := INFO
  ```

  Example output from `maketpf tpfjax link`:

  ```
  ...
  [INFO] [echoproperties] java.version=1.8.0_371
  [INFO] [echoproperties] maven.home=/opt/maven
  [INFO] [echoproperties] project.artifactId=tpfjax
  [INFO] [echoproperties] project.build.directory=/home/userid/ztpf/base/gen/tpfjax
  [INFO] [echoproperties] settings.localRepository=/home/jtplotzk/.m2/repository
  [INFO] [echoproperties] tpf.log4j.version=2.17.1
  ...
  ```

  Existing and new [INFO] messages are now suppressed at the WARN or ERROR levels.

Technical Details

- MakeTPF checkenv target added for Maven applications.

  Example output from `maketpf tpfjax checkenv`:

```
...
# Config variables:
TPF_ROOT="/home/jtplotzk/ztpf /ztpf/cur"
APPL_ROOT="/ztpf/curdrv"
TPFJAVA_VERSION="8"
TPF_JAVA_HOME="/opt/ibm/java-s390x-80"
MAKETPF_MSG_LEVEL="INFO"
MAKETPF_MVN_SETTINGS=""

# Rules variables:
MVN_IS_ONLINE="yes"
MVN_LOCAL_REPO="/home/userid/.m2/repository"
MVN_OCO_GROUPIDS=""
MVN_TPF_PROPERTIES="/ztpf/cur/tpftools/include_ztpf/maketpf.rules_maven_tpf_properties"
MVN_USER_PROPERTIES="/ztpf/cur/tpftools/include_ztpf_user/maketpf.rules_maven_user_properties"
PARENT_VERSION="1.1"
...
```

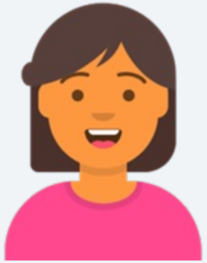
  All diagnostic information is gathered in one command to expedite build failure investigation.

Conclusion

Maven tooling enhancements (APAR PJ47157 – Oct 2023) provides updates to integrated Apache Maven support for the MakeTPF build solution.

These enhancements improve both the development experience and ability to debug and diagnose Maven configuration issues.

Hannah
Build manager



Sophie
System programmer

Thank you