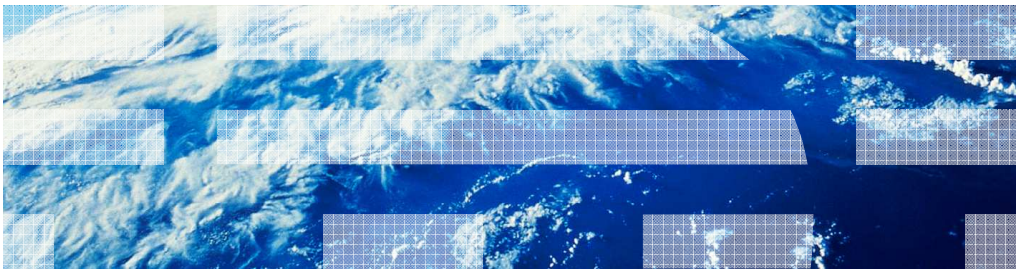
z/OS V1R12

Predictive failure analysis enhancements



Systems and servers

Predictive Failure Analysis provides customers with a way to detect soft failures, otherwise know as “sick, but not dead” incidents. It uses IBM Health Checker for z/OS® remote check support to provide this function.

Introduction

- This presentation discusses predictive failure analysis enhancements in V1R12
  - Layered approach to problem determination
  - SMF arrival rate check
  - Supervised learning support
  - Improved modeling for PFA
  - Migration information
- You should also read the chapters on PFA found in the z/OS Problem Management Guide for V1R12

This module covers V1R12 enhancements in Predictive Failure Analysis (PFA).

First discussed is PFA's layered approach to problem determination, which is continually evolving.

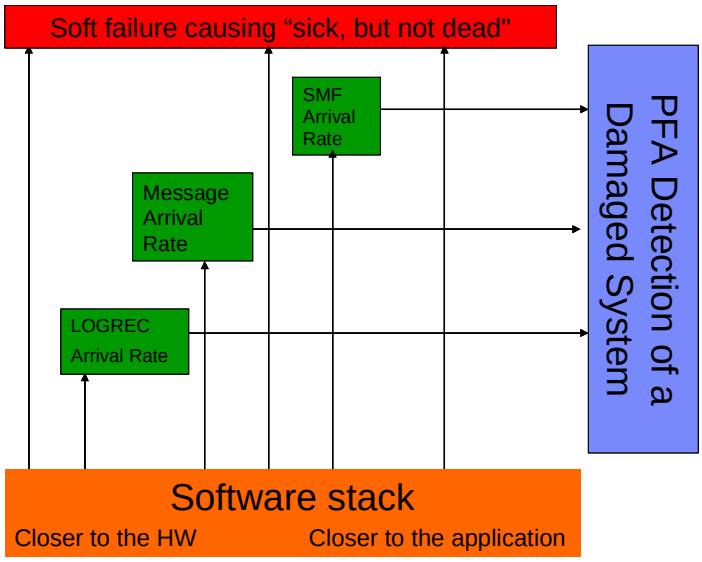
Second you will see the new check in V1R12 called the SMF Arrival Rate check. This check is an addition to the other checks that detect a damaged system. It has some similarities to the previously existing Message Arrival Rate check. You will want to get the most out of this check and understand the configuration parameters and the resulting reports.

Next you will see a new feature called supervised learning support, which is the direct result of stakeholder feedback. That is, it was a direct request by customers. Up until now, all of the checks have had "unsupervised" learning. However, it was discovered that there are cases where particular address spaces are erratic such as during development and testing and these address spaces should be excluded from the checks' results. This supervised learning support allows you to specify certain jobs to exclude. This presentation will describe the reasons to use supervised learning and how to use it. This support required a new "update" option on the modify PFA command and new display output so that you can get immediate feedback on which address spaces are being excluded.

You will then cover several items related to improving modeling for PFA. Some of these items improved performance while others improved functionality or allowed additional configuration.

Finally, installation and migration information is presented.

How PFA detects soft failures: Layered approach



Not to Scale

This chart depicts how PFA can intercept a failure in the software stack and issue a PFA exception before the failure causes a “sick, but not dead” condition indicative of a damaged system.

Some of the metrics that PFA uses in its analysis are closer to the hardware and some are closer to the application. It is easier for PFA to detect an error the closer the metric is to the hardware. The closer the metric is to the application, the harder it is to detect the error.

The LOGREC arrival rate check is closer to the hardware and detects LOGRECs grouped by program key across multiple time ranges.

The Message arrival rate check is detected when WTO and WTOR messages are issued at an abnormal rate which can indicate a damaged system.

V1R12 is moving even closer to the application layer. The SMF arrival rate check detects an abnormal generation rate of SMF records.

All of these metrics can detect a damaged system, but the operator is alerted by PFA before the soft failure occurs.

Layered approach overview

- Problem statement / need addressed:
  - Provide additional ways to detect soft failures to convert “sick, but not dead” situations into correctable incidents
  - Simplify customer use of this complex technology to help achieve higher availability by providing earlier warning of more unusual conditions
  - Improve accuracy, performance, and serviceability
- Solution:
  - SMF arrival rate check
  - Supervised learning – direct result of beta customer feedback
  - Many performance, modeling, serviceability, usability improvements
- Benefit:
  - Greater ability to correct soft failures before they cause a system outage

The PFA component was in its infancy in z/OS V1R10 where it was shipped as an SPE with two checks. The Common Storage Usage check was looking for resource exhaustion and the LOGREC arrival rate check was looking for a damaged system.

In z/OS V1R11 PFA became a preteen and two more checks were added. The Frames and Slots Usage check detects resource exhaustion and the Message Arrival Rate check detects a damaged system.

In z/OS V1R12, PFA is reaching young adulthood. One additional check was added to detect a damaged system so now there are two checks that detect resource exhaustion and three checks that detect a damaged system. These checks detect errors at different layers of the software stack and complement each other such that many soft failures that used to be undetected are now correctable incidents.

Focus then moved on to other areas that needed to be addressed, such as improving the accuracy of existing checks, reducing the footprint that PFA modeling requires, and improving serviceability so that when soft failures are detected, data is saved for future investigation.

The SMF Arrival Rate Check, supervised learning support, and the many improvements to performance, modeling, serviceability, and usability all lead to greater ability to detect soft failures and correct them before they cause a system outage.

PFA_SMF_ARRIVAL_RATE Check (1 of 2)

- Detects a damaged system based on an SMF arrival rate that is too high
- SMF Arrivals = All SMF records sent within the PFA collection interval
  - Not by SMF collection intervals
  - Not by SMF record type
- SMF Arrival Rate = Count of SMF Arrivals / CPU Utilization
- Collects, models, and compares four different categories across time ranges
  - Chatty, persistent address spaces
  - Non-chatty, persistent address spaces
  - Non-persistent address spaces
  - Total system rate

The SMF arrival rate check is similar in several ways to the Message Arrival Rate check. This check accumulates the SMF arrivals in a collection interval and normalizes it by the CPU seconds used in the collection. If the arrival rate found at the last collection is excessively high when compared to the prediction, an exception message is issued. The exception message can be issued by comparing the collected and predicted rates for the entire system, for each individual persistent job being tracked, for the other persistent jobs as a group, or for the non-persistent jobs as a group.

A job is considered persistent if it starts within one hour after IPL. All SMF arrival rates collected in the first hour after IPL are discarded to allow the system time to stabilize after the IPL.

The persistent jobs that are tracked individually are determined either by the jobs that were tracked before IPL or the jobs that had the highest arrivals after a six hour warm-up phase that beings an hour after IPL or when PFA starts, whichever is later. If PFA had not previously been running or data had not been collected before the IPL, PFA chooses the individual persistent jobs to track based on the arrival rates in the six hours from hour one to hour seven after IPL. The persistent jobs with the highest arrival rates are tracked individually. All other persistent jobs are put in the “other persistent jobs” category. If PFA had been running before IPL and had been collecting data, the jobs that were previously tracked are tracked again if that entire list of jobs is persistent again after the IPL. And, if PFA had collected data before IPL, the last hour’s worth of data collected that exists in the files when PFA starts again is discarded so that the arrivals collected during shutdown do not skew the predictions.

PFA_SMF_ARRIVAL_RATE Check (2 of 2)

- Report provides a list of jobs that caused the burst of SMF records
  - If the high rate is isolated to a job or small number of jobs, an address space is potentially damaged
  - Otherwise, the z/OS image is potentially damaged
- The STDDEV parameter and the EXCEPTIONMIN parameter can be used to adjust the sensitivity of the comparisons.
- If SMF is not running or stops
  - Previously collected data is discarded so that predictions aren't skewed
- If you change the SMF configuration
  - Delete the files in the PFA_SMF_ARRIVAL_RATE/data directory or your data will be skewed
- Unable to detect
  - Abnormal SMF record arrival patterns
  - Single SMF record arrivals

When an exception occurs, a report pertaining to the category is produced that lists the address spaces that had the highest rates in the last collection interval. This list should help in determining the address space causing the problem.

PFA ships with default parameters for each check that have been tuned for most installations. If you need to tune parameters due to false positive exceptions or missed exceptions, the SMF arrival rate check provides a STDDEV parameter and an EXCEPTIONMIN parameter. If exceptions are being produced for very low rates on your system and these exceptions are not needed, set the EXCEPTIONMIN parameter higher. If exceptions are produced that are not needed and the difference between the expected value predicted and the current usage is not drastic enough, set the STDDEV higher.

If SMF is not running or stops, PFA automatically discards previously collected data so that future predictions are not skewed by potential configuration changes and holes in the data.

If the SMF configuration changes, it is advisable to stop PFA, delete the /data directory, and restart PFA so that the previously collected data isn't used with the new configuration. If SMF is stopped and restarted across a collection interval, PFA will detect this change and will automatically delete the previously collected data and re-enter the warm-up phase to detect which jobs to track.

The SMF arrival rate check is not designed to detect abnormal SMF patterns or individual types of SMF records.

PFA_SMF_ARRIVAL_RATE Prediction Report

- Perform comparisons after every collection rather than on an INTERVAL schedule in IBM Health Checker for z/OS
- An appropriate report is printed for each type of exception. Example “no problem” report and “total system” exception report shown for SMF Arrival Rate.

```
                      SMF Arrival Rate Prediction Report
(heading lines intentionally omitted)
SMF arrival rate
   at last collection interval      :       83.52
Prediction based on 1 hour of data  :       98.27
Prediction based on 24 hours of data:       85.98
Prediction based on 7 days of data  :      100.22

Top persistent users:
                                              Predicted SMF
                          SMF                  Arrival Rate
  Job                  Arrival
  Name     ASID          Rate        1 Hour       24 Hour         7 Day
  _______ ____  ____________  ____________  ____________  ____________
  TRACKED1 001D         58.00         23.88         22.82         15.82
  TRACKED2 0028         11.00          0.34         11.11         12.11
  TRACKED3 0029         11.00         12.43          2.36          8.36
...
```


The SMF arrival rate report is very similar in format to the report for the Message Arrival Rate check. The SMF Arrival Rate check is also similar to the Message Arrival Rate check in the way the check is run to do the comparisons. Rather than having a set time INTERVAL as an IBM Health Checker for z/OS parameter, the checks are designed to run after every collection. By performing the check automatically upon successful completion of a collection, the check is able to compare the most recent arrivals with the predictions modeled at the last model interval. This enhances both the performance of the check itself and the responsiveness of the check to the current activity of the system.

The report in this example is shown when there is no problem identified on the system or if a “total system” exception occurs. The report lists the SMF arrival rate predictions based on 1 hour of data, 24 hours of data, and 7 days of data. If the amount of data required for any comparison is not available for the system as a whole, that line is not printed on the report. For the jobs listed in the report details, if not enough data is not available for a particular job for any given timeframe, UNKNOWN or INELIGIBLE are printed on the report (depending on the release).

Supervised learning support (1 of 2)

- To increase the accuracy of PFA and reduce the number of false positives, PFA now supports both supervised learning and unsupervised learning.
  - **Unsupervised learning** is the machine learning that PFA does automatically.
  - **Supervised learning** allows you to exclude jobs that are known to cause false positives. For example,
    - Exclude test programs that issue many LOGRECs and cause exceptions.
    - Exclude address spaces that issue many WTOs, but are inconsistent or spikey in their behavior and cause message arrival rate exceptions.
- Supported by all checks except Common Storage Usage

PFA provides both the ability to do supervised and unsupervised learning to determine what is abnormal behavior. In general, PFA is designed to require the minimum customer configuration and therefore, has always used unsupervised learning. However, there are some address spaces whose behavior is not predictable for certain checks and there are scenarios such as combined test and productions systems where PFA is unable to accurately determine abnormal behavior. In those cases, PFA can be configured to ignore certain jobs or sets of jobs that are making the predictions too noisy or are generating false positive exceptions..

Starting in z/OS V1R12, supervised learning is available for all checks except the PFA_COMMON_STORAGE_USAGE check.

Supervised learning support (2 of 2)

- Implementing supervised learning
  - Create EXCLUDED_JOBS file in the check's /config directory
    - /u/pfauser/PFA_LOGREC_ARRIVAL_RATE/config/EXCLUDED_JOBS
    - Comma-separated value file
      - Jobname,system,date_time,reason_added
      - Jobname and system name are required
    - Sample in /usr/lpp/bcp/samples/PFA
  - Start PFA or use f pfa,update,check(*check_name*) if PFA running
  - Supports wildcards in both job name and system name
    - * or ? Supported
    - For example,
      - KKA,*,03/15/2010 12:08,Exclude KKA jobs on all systems.
      - ABC*,LPAR1,03/03/2010,Exclude all ABC* jobs on LPAR1
  - The Message Arrival Rate check on z/OS V1R12 installs an EXCLUDED_JOBS file by default that excludes all JES* jobs on all systems.

Starting in z/OS V1R12 PFA now allows an EXCLUDED_JOBS file which must exist in the /config subdirectory of the check's directory. All address spaces listed in this file will be excluded from processing by the check. Wildcard names will be allowed for both the job name and the system name. A sample EXCLUDED_JOBS file is provided in /usr/lpp/bcp/samples/PFA.

The EXCLUDED_JOBS file is always read and processed when PFA starts. In order to support changes to this file without needing to stop the PFA address space, the PFA modify command has been enhanced to allow updates to this file. The modify command's display option has also been updated to display the contents of the file.

If PFA is running and updates have been made to an EXCLUDED_JOBS file and you want the changes to take affect immediately, you must issue the modify PFA,update command in order for the file to be reread.

The display option of the modify PFA command has also been enhanced to display the list of jobs being excluded for each check.

Improved modeling (1 of 3)

- **Common storage usage check increased granularity**
  - Six storage locations
  - Asterisk indicates storage locations of exception
  - Percentage of current to capacity
  - If expansion occurs,
    - Message printed on report
    - Comparisons on SQA (or ESQA) stop
    - Expanded usage accounted for in location expanded into for usage and predictions

```
                Common Storage Usage Prediction Report
(heading information intentionally omitted)
                                                 Capacity When   Percentage
Storage     Current Usage     Prediction         Predicted       of Current
Location     in Kilobytes   in Kilobytes       in Kilobytes     to Capacity
___________ _____________ _____________ _____________ ___________
*CSA               2796          3152          2956          95%
SQA                 455           455          2460          18%
CSA+SQA            3251          3771          5116          64%
ECSA             114922        637703        512700          22%
ESQA               8414          9319         13184          64%
ECSA+ESQA        123336        646007        525884          23%

Storage requested from SQA expanded into CSA and is being included in
CSA usage and predictions. Comparisons for SQA are not being
performed.

Address spaces with the highest increased usage:
  Job          Storage    Current Usage    Predicted Usage
  Name         Location   in Kilobytes     in Kilobytes
  ___________  _________  _______________  _____________
  JOB3         *CSA                 1235          1523
  JOB1         *CSA                  752           935
  JOB5         *CSA                  354           420
  JOB8         *CSA                  152           267
  JOB2         *CSA                   75            80
  JOB6         *CSA                   66            78
  JOB15        *CSA                   53            55
  JOB18        *CSA                   42            63
  JOB7         *CSA                   36            35
  JOB9         *CSA                   31            34

* = Storage locations that caused the exception.
```


The common storage usage check has been enhanced in the latest release to detect common storage exhaustion on a more granular basis. Whereas in z/OS V1R10 and z/OS V1R11 only the two storage locations (“above the line” and “below the line”) were tracked, starting in z/OS V1R12, the storage locations have been separated.

A partial Common Storage Usage check prediction report for z/OS V1R12 is shown on this chart. Notice that six separate storage locations are now reported: CSA, SQA, eCSA, eSQA, SQA + CSA (which represents total below the line), and eSQA + eCSA (which represents total above the line). In this example, there is an exception for the CSA storage location. Also notice that the SQA has expanded into CSA and is now included in the CSA usage and predictions. Comparisons for SQA have been suspended. The list of address spaces with the highest increased usage will list only the locations causing the exception. As in previous releases, this list is only created when an exception occurs or debug is on.

Note that the numbers in this example are not from a real exception.

Improved modeling (2 of 3)

- Common storage usage check and LOGREC arrival rate check performance improvements
- LOGREC arrival rate check added EXCEPTIONMIN as new configuration parameter.
- Checks enhanced to improve dynamic modeling and comparison algorithms
  - Improve performance by reducing the number of model requests when the system is stable
    - Every 720 minutes (12 hours) by default instead of every 6 hours in previous releases
    - PFA dynamically determines when to model more frequently based on system behavior
  - LOGREC arrival rate – allow data to be used across an IPL and not require a 24 hour warm-up period

The common storage usage check was also enhanced to improve performance. Users reported that in z/OS V1R10 and z/OS V1R11 , the CSA check could have high CPU utilization while the modeling phase was occurring and sometimes, when data collection was occurring. The check has been re-factored so that these periods of high CPU utilization do not occur during collection or modeling. The only time that the extra processing is done that caused the high CPU utilization is when an exception occurs or at “run check” time when the debug parameter is on and the check has recently modeled.

Some of the enhancements done for the checks to improve the modeling and comparison algorithms are as follows:

To allow additional tuning in your environments to reduce false positives in the LOGREC arrival rate check, the EXCEPTIONMIN parameter has been added. The predicted arrival rate and the current arrival rate must be greater than this value in order for an exception to occur. The default value is 25. Therefore, installations who have very low arrival rates with small spikes will no longer see exceptions until both the prediction and the current rate are greater than this value.

All checks have been enhanced to have the default model interval set to 12 hours to reduce modeling on stable environments. The checks will model more frequently in less stable environments when a check is close to issuing an exception.

Checks that perform comparisons using data for time ranges have been enhanced to require enough data even when there are gaps in the data caused by PFA being stopped or by an IPL. These checks are also using an enhanced comparison algorithm when data across the time ranges is deemed to be inconsistent. This enhanced comparison will reduce false positives for spikes caused by processing done in certain time ranges only.

The LOGREC arrival rate has been enhanced so that it can use data before the IPL in the same manner that the Message Arrival Rate and the SMF Arrival Rate checks do and so that it no longer needs to wait 24 hours before making predictions and issuing exceptions. The last hour before shutdown and the first hour after IPL are excluded from modeling so as not to skew the predictions for LOGRECs generated during shutdown and IPL.

Improved modeling (3 of 3)

- Serviceability improvements
  - Before z/OS V1R12, several log files available in the /data directory, but were overwritten for different types of processing.
  - Starting in z/OS V1R12,
    - Each check will have log files for each step in processing
      - CONFIG.LOG, COLLECT.LOG, MODEL.LOG, and RUN.LOG
      - Other log files specific to the check are also created.
    - If PFA issues an exception for a potential problem, log files and data files needed to investigate the exception are copied to a new directory named "EXC_*timestamp*" which is created in the check's directory.
      - An additional REPORT.LOG is created which contains the information written to the health checker report.
      - The files for the last 30 exceptions are stored. If more than 30 exceptions are issued, the oldest EXC_ directories are deleted.

Before z/OS V1R12, there were several log files for each check, but much of the processing wrote to one log file and overlaid the logging of the previous step.

Starting in z/OS V1R12, each check will have several log files that are consistently named. There will be a log file for configuration changes, a log file for collection processing, a log file for modeling, and a log file for when the check is run. There will be other log files for each check depending on what other types of processing the check does.

In addition, if PFA issues an exception for a potential problem, the log files and the data files needed to investigate the exception are copied to a new directory that is created in the check's directory. For example, if an exception was issued for the PFA_COMMON_STORAGE_USAGE check, the new directory with the data and log files would be created in the /u/pfauser/PFA_COMMON_STORAGE_USAGE directory. The name of the new directory will start with EXC_ and the timestamp of the exception will be concatenated to it.

Once 30 directories have been created, the oldest directory and its contents will be deleted before creating a new directory.

Dependencies

- Software Dependencies in V1R12
  - IBM Health Checker for z/OS
  - z/OS UNIX® file system
  - Java™ 5.0 or later (31-bit only)

IBM Health Checker for z/OS, the z/OS UNIX file system, and Java are dependencies that must be installed and configured before installing and configuring PFA. Java 1.4 is no longer supported in z/OS V1R12. Java 5.0 and Java 6.0 (31-bit versions only) are supported in z/OS V1R12.

Be sure to get the latest PTFs for any release before running PFA.

Installation and migration (1 of 2)

- Follow installation and configuration instructions in the **z/OS Problem Management Guide**
  - Ensure dependencies are configured and working
  - Create PFA user ID to own the PFA started task and directories
  - Run install script from the pfauser's home directory
    - Use "**new**" if you are installing for the first time or want to delete data from previous releases
    - Use "**migrate**" if you want to retain data from previous releases (recommended)
    - For z/OS V1R12, run AIRSHREP.sh directly or use the JCL file provided in SYS1.SAMPLIB(AIRINJCL).
      - If using AIRINJCL, you must update it to specify your pfauser's home directory.
  - Update the Java configuration in each check's ini file if needed
  - If you install the PFA code in a place other than the default (/usr/lpp/bcp), update the PROC to have the correct path.

It is imperative that you follow the installation instructions in the z/OS Problem Management guide. The instructions on this slide are abbreviated.

PFA allows you to either create a totally new installation or to use data from a previous release by specifying either the "new" or "migrate" options when running the installation script. If you have used PFA in previous releases, choose the "migrate" option.

PFA is also providing sample JCL in z/OS V1R12 so that your PFA installation can run in batch.

If you are running AIRSHREP.sh directly, you must execute it from the pfauser's home directory so that files are created in proper locations and so that permissions are correct.

If you are using AIRINJCL to run AIRSHREP.sh, you must update the AIRINJCL file to specify your pfauser's home directory.

In either case, you must specify the required parameter to tell the script whether to create a new installation or to migrate previous data.

Installation and migration (2 of 2)

- z/OS V1R12 - PFA now supports one or multiple ini files
  - Uses ini file in /etc/PFA unless check-specific ini exists
    - /etc/PFA directory automatically created when z/OS V1R12 installed
      - If an installation does steps that causes the /etc/PFA directory to no longer exist, the new ini file processing cannot be used.
      - The /etc/PFA directory can be created manually as described in the z/OS Problem Management guide.
    - If “new” was chosen,
      - /etc/PFA/ini copied from /usr/lpp/bcp/samples/PFA
      - Update Java configuration in /etc/PFA/ini if necessary
    - If “migrate” was chosen,
      - the ini file copied to /etc/PFA/ini was from an existing check so that the Java configuration should be accurate already.
      - and multiple ini files are required, copy the file from /etc/PFA/ini to the checks’ directories and update them if needed.

In previous releases of PFA in order to specify the Java configuration information, an ini file needed to be changed for every check on every system. Customers requested that PFA provide the option of using just one default ini file so that users did not have so many files to update when they needed to make a change. IBM has responded to this request. Beginning in V1R12 PFA supports either one or multiple ini files.

If an ini file exists in the checks’ directory, then PFA will use that check-specific ini file. However, if a check-specific ini file does not exist, the PFA will use the common ini file which resides in the /etc/PFA directory.

If an installation saves the /etc directory and then restores it after release upgrade, they must create the /etc/PFA directory manually after release upgrade since it has now been overlaid. If they omit this step and the /etc/PFA directory doesn’t exist on their system, they cannot take advantage of this new feature.

If this is a new install request, any previous directories are deleted, all directory paths are created, the sample ini file in /usr/lpp/bcp/samples/PFA is copied to /etc/PFA, and the EXCLUDED_JOBS file is created for the message arrival rate check.

If this is a request to migrate, it is assumed that you want to use a default ini file in /etc/PFA instead of maintaining an ini file for each check. When migrate is requested, all directory paths that don’t exist are created, the ini file from an existing check starting with the common usage check is copied to the /etc/PFA directory, all ini files in check’s directories are deleted, and the EXCLUDED_JOBS file is created for the message arrival rate check. The ini file is copied from an existing check so that you shouldn’t need to update the new default file in /etc/PFA if your Java configuration hasn’t changed. The script will attempt to copy an existing ini file. If it cannot locate one, it will copy the one from /usr/lpp/bcp/samples/PFA.

Once the installation is complete, update the Java configuration in /etc/PFA/ini if it is not correct for your installation.

If you still want multiple ini files, create ini files in the checks’ directories and update those as well.

Fall back considerations by release

- From z/OS V1R11 to z/OS V1R10
  - Falling back → No action
  - Moving forward to z/OS V1R11 after rollback → rerun an installation script if
    - you deleted any directory structures for z/OS V1R11 checks (use AIRSHREP.sh to replace data or use AIRSHKP.sh to keep data)
    - you want to delete z/OS V1R10 data (use AIRSHREP.sh)
    - If using AIRSHREP.sh, update ini files for each check
- From z/OS V1R12 to z/OS V1R11 or z/OS V1R10 (manual steps until co-existence APAR available)
  - Falling back
    - If using /etc/PFA/ini, copy that file before fall back for simplicity.
    - After fall back, run install script for release to replace directory structures (AIRSHREP.sh)
    - Update ini files for Java paths (use data in copied /etc/PFA/ini if the same)
  - Moving forward to z/OS V1R12 after rollback
    - Run install script (AIRSHREP.sh -- either *new* or *migrate*)
    - If using /etc/PFA/ini, restore it if needed. Otherwise, re-create individuals and update them.

When falling back to z/OS V1R10 from z/OS V1R11, there is no action to take. When moving forward again to z/OS V1R11 after a fallback, there is no action to take if you want to keep the old data and you've not modified any of the PFA checks' directory structures. However, if you want to delete the z/OS V1R10 data or you have modified structures, you must rerun the installation script.

When falling back to z/OS V1R10 or z/OS V1R11 from z/OS V1R12, there are currently manual steps that must be done until a co-existence APAR is produced. In order to ensure that the file formats are compatible with z/OS V1R10 or z/OS V1R11 code, you must delete the data for each check and re-create the directory structures. Also, the prior releases didn't use the one /etc/PFA/ini file and will revert to using individual ini files. The easiest way to handle the ini file is to copy the /etc/PFA/ini file before falling back or make sure it is not deleted upon fall back. Then, after falling back, run the AIRSHREP.sh script which will replace the directory structures and delete the data. It will also create an ini file for each check. Those files must be updated if the path to either the PFA Java code or the system Java code is not correct. If your z/OS V1R12 installation's Java paths were the same as z/OS V1R10 and z/OS V1R11, overlay each individual ini file with the one you copied before the fall back.

Once you are ready to move back to z/OS V1R12, you must re-create the directory structures for the new checks. Run AIRSHREP.sh with either the "new" or the "migrate" option depending on your preference for keeping the old data. Then, if you are planning to use /etc/PFA/ini, restore your prior version or update the one that exists in /etc/PFA. If you want to use individual ini files, create them and update them to have the correct paths.

To avoid the manual steps, a co-existence APAR is being planned. When moving back to a previous release from z/OS V1R12, you will need to copy the /etc/PFA/ini file if you are using it and restore it after the fall back. The co-existence APAR will allow z/OS V1R10 and z/OS V1R11 code to recognize the /etc/PFA/ini file. If your Java paths are different in the previous release, update /etc/PFA/ini and any individual ini files that you are also using.

If you have left all directories intact (even those for z/OS V1R12 checks) and you want to keep previous release data, there is no action moving forward. However, if you either want to delete or data or you have modified the directory structures, you must rerun the AIRSHREP install script. If you have modified directories, run “new” again.

If you have not been using /etc/PFA/ini, there is no action to take falling back with the PTF. Once you move forward again, the individual ini files will be consolidated when the installation script is run. Therefore, if you want to keep the individual ini files and you want to keep your data, do not run the installation script again. However, if you want to consolidate your ini files or you want to delete your data, you must rerun the installation script. Either option consolidates your ini files. Therefore, if you want separate ini files after the install, you must re-create them and update them as necessary.

If necessary, update your /etc/PFA/ini file after moving forward.

Session summary

- PFA provides a layered approach to problem determination
- SMF Arrival Rate check provides additional detection of a damaged system
- Supervised learning support allows you to exclude address spaces from PFA processing
- Only one ini file is now required for all checks
- Many enhancements made to improve performance, modeling, usability, and serviceability

PFA has made significant enhancements in V1R12.

First, the layered approach to problem determination has been extended. Second, the SMF arrival rate check has been added. This check provides PFA with another metric for detecting damaged systems. Next, the addition of supervised learning support allows you to exclude specific address spaces from PFA processing. In response to customer requests, only one ini file is now required for all checks. Finally, many other enhancements have been made to improve performance, modeling accuracy, usability, and serviceability.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_V1R12_Availability_PFA_Enhancements.ppt

This module is also available in PDF format at: ../V1R12_Availability_PFA_Enhancements.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information