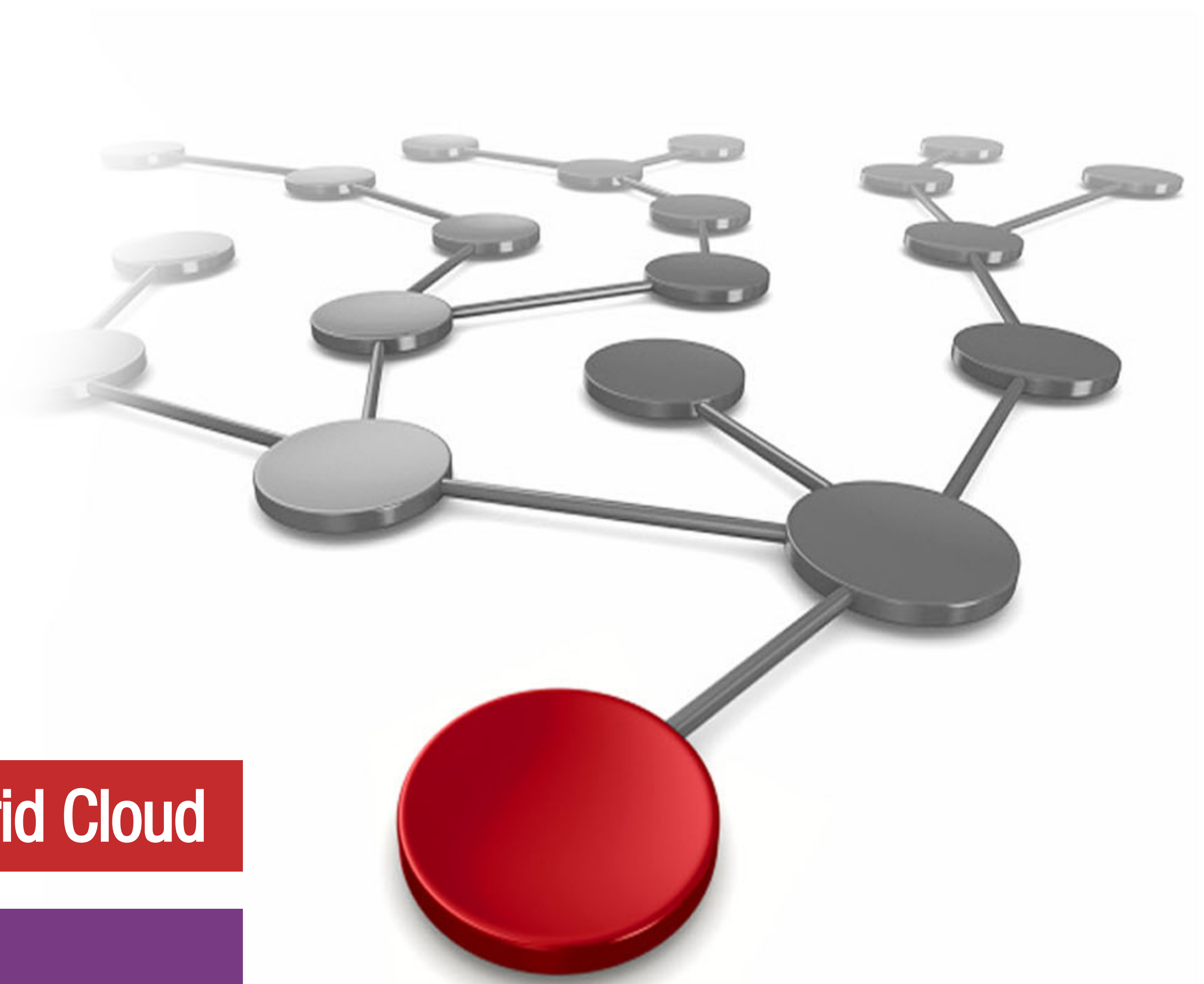
ibm.com/redbooks

ISV IBM zPDT Updates

Bill Ogden



Hybrid Cloud

IBM Z

Redpaper

ISV IBM System z Personal Development Tool GA12 coverage

Important: This IBM® Redbooks® publication generally covers zPDT GA11 and earlier zPDT releases. This specific IBM Redpaper is new and covers the important elements of zPDT GA12 including the last fix pack for GA11.

This document extends zPDT terminology so that IBM and Information Technology Company (ITC) documentation is more compatible, as shown in Table 1.

Table 1 Terminology

IBM terminology	ITC terminology
zPDT GA10.xx	Driver 55.16 ("16" denotes a very specific level)
zPDT GA11.2	Driver 57.16 ("57.16" is a shorter form)
zPDT GA12	Driver 59.xx (exact xx value unknown until after release)

The *driver* numbers are derived from the *rpm* name of the release or fix pack, for example z1090 1-11.57.16. The general availability (GA) numbers can denote a fix pack, such as GA11.2 (fix pack 2 for GA11). The last two digits of the driver number can change with fix pack releases. These slightly different names have no hidden meanings and are the results of different people making notes in different ways.

© Copyright IBM Corp. 2024.

zPDT GA12 (also known as Driver 59.xx, or 59.xx) contains the following key changes to zPDT:

- The internal operation of zPDT that is designed to optimize the performance of emulated IBM System z® instructions has been substantially changed. This internal design is not documented and was previously built, element by element, as zPDT progressed through many releases. With GA12, this internal performance function (sometimes known as the JIT[1]) is redesigned to provide a more stable base for future enhancements. There are no visible signs of this change, other than minor performance improvements. *This zPDT internal change is the most important component of GA12.* This seemingly minor change involved substantial (and ongoing) efforts.
- Fix pack GA11.2 (Driver 57.16) introduced support for a new series of IBM 1090 license tokens. zPDT use of the older 1090 tokens will be discontinued by 2025. This was the primary component of fix pack GA11.2. Details about the new tokens are widely distributed by ITC. Support for the new 1090 license tokens is extended to a fix pack for zPDT GA10, but no other additions were added to the GA10 release.
- New zPDT commands have been introduced in the mixture of GA12 and GA11.2:
  - The **`disasm`** command, as the name implies, produces a disassembly listing based on current System z memory (real or virtual) at a particular address.
  - The **`aws_coretype`** command deals with a new zPDT function that helps manage core image files that are produced by Linux. This *core image handling* function is complex and more details are covered later in this publication. This can be an odd topic because zPDT is an unsupported IBM product. However, many zPDT customers have made other support arrangements.
- A new *devmap* option (in GA11.2) can be used to block the automatic addition of leap seconds to the time-of-day obtained from the base Linux.
- The timing of the **`awsstop`** command is extended to reduce unimportant *timeout* messages that sometimes occur with this command.
- GA12 updates the Common Cryptographic Architecture (CCA) emulation to level 8.1. Additionally, it contains upgrades or fixes for some earlier obscure problems. The CCA 8.1 level provides SHA-3 support, adds X9-SWKB/TR-31 operational support, and makes minor changes to TR-31 blocks requested by customers.
- GA12 development included use with IBM z/OS® 3.1, although there are no IBM zPDT® changes related to this usage.
- As usual, both GA11.2 and GA12 (Drivers 57.16 and 59.xx) include fixes for minor problems that are found through normal usage.

Note: The following sections address zPDT *changes* that appear in GA11.2 and GA12. For more information, see other sections of this publication or general zPDT documentation. Some of the content in this publication is taken (with permission) from ITC materials.

[1] Do not associate this JIT with similar names associated with the Java language and other languages and software. In a sense, it has become a generic term.

GA12 Linux distributions and levels

zPDT is developed and tested using three different Linux distributions: Red Hat Enterprise Linux, Ubuntu, and SUSE Linux Enterprise Server. Three *builds* of zPDT are present in the zPDT installation package, one for each of these Linux distributions. zPDT is a complex application and can depend on specific Linux library levels and arrangements that can differ slightly among these three basic Linux distributions.

The Linux levels used to develop a zPDT release are important. Most of the IBM testing is done at these levels and customer usage is typically more stable when these levels are used.

Note: A zPDT release tends to be based on the Linux levels that are current when the development of that zPDT release is started. In general, the zPDT developers do not try to *adjust* to the latest levels.

For GA12 (59.xx) these levels are:

- Red Hat Enterprise Linux 8.6 and later 8.x distributions.
- SUSE Linux Enterprise Server 15 service pack 3 or later SUSE Linux Enterprise Server 15 levels.
- Ubuntu 20.04.2 and later 20.04 levels.

These are the base levels that are used to develop and test zPDT GA12 (59.xx). A few other distributions were informally used and these include the following:

- Several reasonably current Fedora levels.
- CentOS 9 Stream (with recent updates).
- openSUSE 15.4.

Note: The *informal usage* mention here does not constitute *testing* in a product sense. These Linux levels can be important for stable zPDT operation. Using different levels, including more recent ones, can introduce issues that developers or support organizations might not prioritize.

Small Linux notes

There are sections of this publication that describe elements that are involved in Linux installation. In general, future IBM® and ITC documentation tend to eliminate such material, partly because the details can change with various Linux distributions and maintenance releases.

A specific Linux detail is important. zPDT still requires a 32-bit version of the libstdc++ library. This library does not need to be installed at the time of the basic Linux installation, but it must be installed before the zPDT installation is started.

Note: The web search for this library might involve various names, such as libstdc++.i686, libstdc++6-32bit, lib32stdc++6, or something similar.

Slightly more obscure issues are known for several OpenSUSE 15 versions. The following Linux actions might assist the installation of current zPDT releases:

```
zypper install libncurses5 libcap-progs
mkdir -p /etc/rc.d
mkdir -p /usr/lib/systemd/system
```

These commands can be issued from Linux **`root`** or from a **`sudo`** arrangement. The need for these steps might be removed in future zPDT releases.

The Information Technology Company

ITC[2] is a vendor that is associated with IBM who manages most of the ISV zPDT licenses or systems now in use and some of the IBM ZD&T systems. ITC offers several options for more support levels and these options might be important. ITC also offers PC hardware systems especially systems that are especially suited for zPDT.

ITC is managing the distribution and license updates of the new zPDT token models.

The primary ITC office is located in Falls Church, Virginia with more offices in Germany and the United Kingdom. A general method of contacting ITC is by way of `mailto:sales@itconline.com`.

New license tokens

zPDT usage depends on zPDT licenses and, in GA11 and earlier releases, the licenses were held in IBM 1090 USB tokens.[3] These tokens had model numbers (L10, L02, or L03) that reflected the number of emulated System z CPUs that can be used with this license token. The token vendor software for these models is ending and IBM is moving to a new set of token models.

Some zPDT documentation (including this book) ues the term *Gen1* to describe these existing license tokens. The term *Gen2* refers to a more modern form of zPDT licenses.[4] This Gen1 and Gen2 terminology is informally used by the developers and has no relation to the many other uses of generation terms.

The new replacement tokens appear similar to the older 1090 tokens, although slightly smaller and a different color. The following token model numbers are important:

- Gen1: 1090-L01, -L02, -L03 (usage will be discontinued some time in 2025)
- Gen2: 1090-LT1, -LT2, -LT3, -LT6, -LT8 (usage starts in late 2023, with GA11.2 or GA12)

The new tokens retain the IBM 1090 name, but have different model numbers. The older Gen1 tokens can license use of up to three emulated System z CPUs. The newer Gen2 tokens can license use of 1, 2, 3, 6, or 8 emulated CPUs. The number of licensed CPUs is reflected in the cost of the licenses.

The older Gen1 tokens are sometimes known as SHK types, and the newer Gen2 tokens are LDK types. This terminology is related to the token vendor's support and is sometimes used informally by the zPDT developers.

[2] `https://www.itconline.com/`
[3] ZD&T license details are different and our description of this area does not apply to ZD&T customers.
[4] Until these new tokens appeared, the Gen2 term reflected the ZD&T software only licenses.

The detailed meaning of *licensed CPUs* is slightly more complex, depending on the usage of Central Processors (CPs), IBM Z Integrated Information Processors (zIIPs), IBM Z Application Assist Processors (zAAPs) (outdated), and Integrated Facility for Linux (IFL). Also, the license tokens can be installed on the PC that is running zPDT or on a remote license server. These extended details are covered in other sections, in other IBM documentation, and in ITC documentation.

zPDT licenses are normally valid for 1 year and can usually be renewed for another year. At the time of writing, the plan is to stop renewing zPDT licenses for the older Gen1 tokens. When a zPDT license renewal is requested a new Gen2 token is automatically supplied, possibly with the requested new licenses already installed.[5] When one or more zPDT licenses in the old token expire, that token is useless. With this plan, all the old zPDT tokens are replaced over a period of a year, depending on when the licenses in the old token expire.

Attention: A minor charge for the new token can occur. A license renewal (and shipment of a new token) is normally handled at least 1 month before the old licenses expire. If the renewal is for more emulated CPUs than are present with the old licenses, this change is handled.

Also, the IBM process involved in allowing license renewals is changed and might take slightly longer than for previous renewals. Again, ITC usually handles these details.

As a minor detail, starting with zPDT GA11.2 (57.16) both old and new tokens can be used concurrently. This happens when the licenses in the old token are due to expire soon and a new replacement token has already been received. This results in a short period of free usage for other emulated CPUs.

In a practical sense, the switch to new tokens is apparent to most zPDT users. There are some minor changes for administration that is involved with the new tokens, but after installation, zPDT operation remains unchanged. Administration details include the following:

- Some zPDT commands are related only to the older Gen1 tokens:
  - **`z1090_token_update`** is used to generate a request file for a license update or to install the update.
  - Generally speaking, the zPDT administrative steps that can be used to avoid routine use of Linux root access apply only to Gen1 token usage. There are several zPDT commands that are involved, but few zPDT customers use them.
- Some zPDT commands are related only to the new Gen2 tokens:
  - **`request_license`** is used to generate a request license update file for Gen2 tokens or software-only licenses. This command automatically creates a file name for the request data. The user cannot specify a file name and it is placed in the Linux root directory by default.
  - **`update_license`** is used to install a license update file for Gen2 token licenses.
  - **`query_license`** displays details about Gen2 licenses that it can access. There is a different implementation of this command for ZD&T software-only licenses. If used with Gen1 licenses there is a relatively meaningless message about `No HASP-HL key available`.
  - Linux *root* access must be used for license updates and a few other functions. The zPDT processes to avoid using *root* that were present for Gen1 tokens are not present for *Gen2 tokens*.[6]

[5] If the new licenses are not already installed, the user goes through the normal process of generating a request file, sending it to ITC, and then installing the license update file when it is received.
[6] This was true at the time of writing this IBM Redpaper publication. It can change in future zPDT releases.

- Some license-related commands are only meaningful for ZD&T or are no longer used:
  - The **`serverconfig`**, **`serverconfig_cli`**, and **`display_gen2_acclog`** commands are for software-only licenses that are used by ZD&T and are covered in separate ZD&T documentation.[7]
  - The **`gen2_init`** command and the **`LDK_server_config`** commands are no longer used.

Some small details that are related to the transition to the new tokens or to Gen2 licenses include the following:

- The 1090 zPDT license installations and updates normally include the license information that is needed to decrypt and install the z/OS IPL volumes that are distributed with the Application Developer Controlled Distribution (ADCD) z/OS systems. References to zPDT licenses typically include this additional license.
- A basic zPDT license (as used with an IBM 1090 token) does not function with a ZD&T system, and a ZD&T license (as used with an IBM 1091 token or software-only license server) does not function with a basic zPDT system.
- The System z CPU serial numbers that are created by zPDT are changed when using the Gen2 tokens. The numbers that are created by the Gen1 licenses are partly based on the serial number of the Gen1 token. The new serial numbers are based on the new licenses/tokens and the customer cannot change them.

  Note: If your zPDT usage depends on consistent System z serial numbers, you probably need to find a way to handle this change.

- The **`clientconfig`** command presents a slightly different menu for the user, containing definitions for both Gen2 and Gen1 remote license servers. The **`clientconfig_cli`** command appears to be unchanged. A localhost parameter can still be used for Gen1 licenses and localhost is assumed for Gen2 licenses unless a specific license served is added.
- When starting, zPDT looks for Gen2 licenses first and then for *Gen1* licenses. Having both types present is unlikely except, possibly, when first using *Gen2 tokens*.
- It is also possible to generate a Gen2 license request file (**`request_license`** command) while zPDT is running, perhaps with multiple Gen2 tokens connected. It is possible to update a Gen2 license (**`update_license`** command) on a running zPDT system, perhaps with multiple Gen2 tokens connected. These actions, when zPDT is running or has multiple tokens that are connected, are not possible with Gen1 tokens.
- If a remote license server is used, it normally has the *uimserver* function running; this is unchanged.

  A remote license server must be updated to the same zPDT level (GA12, for example) that the clients are using. This is required for proper operation.
- The **`query_license`** command lists license types 300, for zPDT operation, and 331 for installing encrypted z/OS IPL volumes. These license type identifiers are not intended for any external use purposes.
- Some zPDT customers use a browser (`https://localhost:7002`, or something similar) to display license details. This is not an IBM-provided function. This is not functional for Gen2 licenses and the zPDT **`query_license`** command provides similar data for Gen2 licenses.
- The **`Z1090_ADCD_install`** command and usage is unchanged.

[7] IBM Z Development and Test Environment, found at `https://www.ibm.com/docs/en/zdt/12.0.5?topic=overview`.

- After you switch to the new tokens, direct usage of sntl-sud, zpdt-shk-server, and other such links are no longer relevant. There are not many zPDT users directly accessing these interfaces.
- The need for a particular 32-bit Linux library during zPDT installation (as mentioned in this book) still exists and will exist until all support for the old 1090 *tokens* is eventually dropped from zPDT. This library is often noted as *libstdc++.i686*, *libstdc++6-32bit,* or *libstdc++6,* or a similar name, depending on the Linux distribution being used.
- The new 1090 tokens use Linux port 1947 for remote zPDT license access. Older tokens used port 9450. Both the older and newer systems use port 9451 for UIM access.
- Some messages (by way of the base Linux) have terminology such as *HASP-HL* and *fingerprint*. These terms are used by the *token* vendor software and have no underlying meaning in zPDT. And this use of *HASP* does not refer to the older name for z/OS JES2.

There is one detail that is associated with the new license tokens that might surprise some users. If zPDT is stopped with the **awsstop** command, for example, there can be a delay of several minutes until the zPDT licenses are available again. That is, stopping zPDT, perhaps making minor *devmap* changes, and then promptly starting zPDT again might result in messages of no available licenses. There are two ways to bypass this problem:

1. Do not plan on stopping and almost immediately restarting zPDT.
2. While zPDT is stopped, unplug the token for a few seconds and replug it. This bypasses the problem.

Updating licenses with a new token

The Gen2 zPDT licenses are handled carefully and this care includes the use Linux *root* authority. There were methods to avoid this direct use of *root* for *Gen1* licenses; these methods are not extended to Gen2 licenses. A typical creation of a *Gen2* license request file is shown in Example 1.

Example 1   Creation of a Gen2 license request

```
su -                        (change to Linux root; a password is usually
required)
cd /usr/z1090/bin           (this is where the request_license program is
found)
./request_license           (the primary command to generate a request file)
   zPDT/LKD license request started.
   The following is a list of available keys     (“keys” = tokens that were found
   connected)
   HASP-HL key_id=1611917245  feature(s): 0 330 331
   updateinfo status 0   fingerprint status 0
   Host information: localhost 127.0.0.1 Linux
   The requested file created is: ~/localhost_1697203866.zip     (the number is a
   “timestamp”)
   Success

cd /root
ls
   (you should find the generated file here)
exit                        (leave Linux root status)
```

Example 1 on page 7 contains numbers and timestamps relating to a single new token that was connected to the Linux zPDT system. This particular token already contained licenses that soon expire, making the creation of a license request file relevant. While your experience might differ, the general flow is similar. Some of the information displayed, such as *updateinfo* and *fingerprint*, has no specific documentation and can be ignored.

Note: It is possible to have multiple Gen2 tokens connected. In this situation, the **`request_license`** command prompts you to select which token should be used by the command.

Although not obvious in Example 1, the command can be used while zPDT is operational. This was not possible with Gen1 tokens.

In Example 1 on page 7, the `localhost_1697203866.zip` file is sent to whoever handles your license updates (this is usually ITC). There might be additional information (send by way of email or post office mail) required for IBM approval of the license updates, but this is not further described here. The result is a returned compressed file containing the updated licenses.

When a new license file is received the installation process can be like the following:

```
(we placed the new license file in our home directory because this was convenient)
cd ~                                        (move to directory containing the
file)
su -                                        (switch to root, or use sudo)
update_license xxxxxxx.zip                  (xxxxx is the name of the file you
received)
   The license update was successful.
   There were no additional results to display.
query_license                               (Optional. Display new license
information.)
   The following key is available:
   HASP-HL  Feature Name         Expiration             Logins  MaxLogins
   330-                     Wed Nov 15,2023 18:55:00      0         3
   331-                     Wed Nov 15,2023 18:55:00      0         1
   Host information: localhost  localhost
exit                                        (Leave Linux root status)
```

The *Logins* column in the query_license display reflects the number of current uses of the license. The *MaxLogins* column reflects the number of usage licenses in the token. This *Logins* and *MaxLogins* terminology within the license structure is not used elsewhere. Most zPDT users say that they have three zPDT licenses.

Both the **`request_license`** and the **`update_license`** commands (for Gen2 licenses/tokens) can be used while zPDT is actively running. Both can deal with multiple Gen2 tokens being installed.

Automatic leap second control

The **`cpuopt`** statement (in a *devmap*) has several possible parameters that are seldom used by customers with basic zPDT offerings. The new parameter appears as follows:

```
cpuopt ADD_LEAPSEC=ON                  (default usage)
cpuopt ADD_LEAPSEC=OFF                 (this is the new option)
```

With this statement, the possible operands do not have internal blank characters.

zPDT (including GA12) automatically adds the current leap second value (about 27 seconds) to the time-of-day (TOD) value present in Linux and uses this adjusted value as the initial zPDT time-of-day.[8] This corresponds to the default **`ADD_LEAPSEC=ON`** action.[9] Placing *cpuopt* **`ADD_LEAPSEC=OFF`** in the *devmap* prevents this addition of leap seconds to the zPDT TOD, comparing the zPDT TOD (as seen by z/OS, for example) and the Linux TOD (as seen by the base Linux) more reliable.

Note: It is possible that future zPDT releases might change the default **`ADD_LEAPSEC`** value to **`OFF`**.

The disasm command

This new zPDT command appears as follows:

```
disasm <hex-address>.<hex-length>          (this addresses real memory)
disasm -vp 7020.20                         (primary virtual memory)
disasm -vp -t 72C80.30                     (primary virtual, with EBCDIC text
displayed)
disasm -vs 14122.10                        (secondary virtual memory)
disasm -vh A280.10                         (home virtual memory)
```

This command can provide a display (on the Linux console) containing an equivalent assembly language of the IBM Z instructions at a specified address in emulated IBM Z memory. Memory is displayed in 32-byte units, starting at a 32-byte boundary at (or just below) the address that the user provides. The letter `t` can be added to any of these prefixes to include a character display. The display output first lists a simple hexadecimal dump of the area specified, followed by a listing of assembly instructions that match this data. There are some considerations that are involved in using this command:

- The memory that is specified is in the default CP. This is CP 0 unless the zPDT **`cpu`** command was used to assign a different default CP.
- The virtual addresses are in the address space that is currently set on that CP.
- The maximum length is 1 MB.
- If the data at the address that is specified or defaulted in the adjustment to a 32-byte boundary does not match a System z instruction, it is displayed as a constant. Starting at an incorrect address might produce a meaningless disassembly display.

The CPU does not need to be stopped to use the **`disasm`** command, but the results might not be meaningful in that situation because, if the CPU is running, the instructions that **`disasm`** shows might not accurately represent the program's state. Possibly the best way to use the command is to:

1. Run with a single active CP.
2. Use the zPDT **`ztrace`** command to stop the CP at a specific address of interest.
3. Do not try to use **`disasm`** command when there is a wait state PSW.

This is a powerful command, but effective usage requires some practice. It is not intended for routine use during routine development of normal application programs.

[8] After zPDT starts, and has the initial TOD from Linux, it manages the TOD itself. This can result in small drifts between the base Linux TOD and the TOD being used by software running under zPDT.
[9] Lowercase letters can be used for this option. We show uppercase letters here to match other documentation.

Handling core image events

Although rare, it is possible that zPDT users might experience Linux core image dumps.[10] ISV zPDT is an unsupported IBM product and there is no official way to send such dumps to IBM or to request help with them. The zPDT GA12 release (Driver 59.xx) provides a method to package such dumps with the specific Linux library files that are associated with the Linux process that produced the core image dump. Many zPDT customers have various support agreements with ITC or other organizations and providing a complete package of dump material might be helpful in rare cases.

There is no specific zPDT title for this function and it is described as *core image handling*. The key element is the inclusion of many (or all) of the Linux library files that are relevant to the failing program. Anyone trying to analyze a core image dump might need to reference the same level of Linux library modules that were in use by the failing program. Packaging these library modules with the core image file can help resolve potential issues.

The general design involves the **`kernel.core_pattern`** statement in the Linux `/etc/sysctl.conf` file. In older zPDT releases, this statement defined a name pattern for core image files. With this new function active in zPDT GA12, this statement is changed to indicate a Linux command script that is to be started as part of core image handling. This Linux script (named **`/usr/z1090/bin/core.sh`**) manages the enhanced core image handling functions.

Normal zPDT installation involves running the **`/usr/z1090/bin/aws_sysctl`** command script. This script asks the user whether various changes should be included in `/etc/sysctl.conf`, and one of the changes involves setting the **`kernel.core_pattern`** statement to indicate this enhanced core image handling. Almost all zPDT customers automatically type `y` to all the prompts displayed by the **`aws_sysctl`** command script, and this results in the enhanced core image handling being active.

If the zPDT customer types `n` to the `aws_sysctl` prompt for setting the **`kernel.core_pattern`**, then the new core image handling function is not enabled and any Linux core image taken is stored in the original Linux format. Alternatively, the user can use the new **`aws_coretype`** command to enable or disable the enhanced core image handling function. The syntax for this command is:

```
aws_coretype [query | basic | enhanced] (must have Linux root authority)
```

This command can be used regardless of what options were chosen when running the **`aws_sysctl`** command script. The **`query`** option simply displays the status of core image handling. The **`basic`** option causes normal Linux handing of any core image, and the **`enhanced`** option causes the new enhanced core image handling to be active. As mentioned earlier, producing core image dumps is fairly rare for most zPDT customers and they probably do not care whether this enhanced core image handling is active. However, this enhanced core image handling, if active, applies to any core image events in the Linux system and is not restricted to zPDT operation.

[10] We recognize that devoted Linux experts tend to use the core image term without adding the word dump to it. However, this book assumes that most readers are more oriented to z/OS and the term dump is more standard in that context.

Operation

The basic operation includes the following notes:

- Assuming that the function is enabled, if a Linux core image event occurs then a compressed file is produced and placed (by default) in the `~/z1090/logs` directory. The tar file contains the actual core image file plus various other files that represent Linux library files and some other information data.
  - There are typically about 20 files that are included inside the compressed file, but this number can vary considerably.
  - This process is associated with any core image produced, and is not limited to zPDT operation.
  - Depending on the associated files that are included in the tar file, there can be a noticeable system delay while the files are being copied.
- The size of the compressed file can vary greatly, mostly depending on the size of the core image file itself. Sometimes the user might want to control the placement of the resulting compressed file. This is done by editing the `/usr/z1090/bin/aws_core_config` file, which is new with zPDT GA12.

  This file contains two lines:

  ```
  compressed_core_path= (A blank operand causes default usage of ~/z1090/logs)
  logfile_path= (A blank operand defaults to ~/z1090/logs)
  ```

  Most customers do not change these statements.
- The resulting compressed file can be expanded with a normal **`tar -xvf`** command.
- Copy the compressed file to a new Linux directory before you expand it. Some of the file names within the compressed file match the basic Linux library name. This can result in overlays if multiple compressed files are expanded within the same Linux directory.
- The compressed file can be forwarded to anyone helping resolve zPDT problems and can be expanded by the recipient. There are no unusual security controls and no zPDT licenses are involved.
- If the situation is producing multiple core-image events, then the **`aws_coretype`** command is used to disable the collection of more compressed files.
- If some of the Linux environment includes libraries being used by way of NFS, these libraries will probably not be included in the resulting compressed file.

Comments

The zPDT memory size from the **`memory`** statement in the *devmap* can be important. Some zPDT users have rather large values (such as 100 GB) and, depending on what is happening when the core image event occurs, this can produce a large core image file. It might not do so depending on exactly which process encountered the failure and many other factors.

The default location for the compressed file(s) is in the `~/z1090/logs` directory. This might not be appropriate if the compressed files are large or if there are many of them.

Many zPDT customers are not interested in core image handling and might include a **`aws_coretype`** command (to disable this function) in their routine zPDT startup procedure, or disable it when running the **`aws_sysctl`** script. In such cases, the customer can later use the **`aws_coretype`** command to enable it when needed.

We again stress the point that ISV zPDT is an unsupported IBM product. The file discussed here, a compressed file that contains a core image file plus other related files, is not intended for routine transmission to IBM.

Authors

This paper has been developed by:

Bill Ogden is a retired IBM Senior Technical Staff Member who continues to work part-time with projects, such as ones for new mainframe users and entry-level systems.

Now you can become a published author, too!

Here’s an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks® residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, IBM Redbooks
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on LinkedIn:

  https://www.linkedin.com/groups/2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/subscribe

- Stay current on recent Redbooks publications with RSS Feeds:

  https://www.redbooks.ibm.com/rss.html

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

IBM®	Redbooks (logo) ®	z/OS®
Redbooks®	System z®	zPDT®

The following terms are trademarks of other companies:

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Red Hat, Fedora are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

REDP-5724-00

ISBN 0738427888

Printed in U.S.A.

Get connected

ibm.com/redbooks