文章
Louis Lu · 一月 15, 2021 阅读大约需1 分钟

什是核心文件，它们什时候有用

什是核心文件？它们什时候有用？

本文档中的信息以 2019 年 6 月 30 日发布的 InterSystems 产品最新版本为准。此更新涵盖了截至 2020 年 4 月 14 日发现的错误，但不包括 InterSystems 产品新版本中的更改。 不过，现有产品的细节不会经常变化。 本文的 PDF 版本可以从 WRC 获取

??		
核心文件基础知识	SuSE Linux	Windows
AIX	Ubuntu Linux	测试
Docker	macOS (Darwin)	健全测试
HP – UX	OpenVMS	传输
RedHat Linux	Solaris	索引

????????

Caché、Ensemble、HealthShare 和 InterSystems IRIS 数据平台非常可靠。
我们的绝大多数客户从未经历过任何种类的故障。
但是，在极少数情况下，进程发生过故障，并因此生成核心文件（在 Windows 和 OpenVMS 上称为进程转储文件process dump file）。
核心文件记录了进程发生故障时的进程状态，包括进程寄存器和内存（是否包括共享内存的信息取决于配置）。
核心文件实质上是发生故障的进程在试图执行错误操作时的瞬时画面。
我们可以根据该画面向回推断，以找出导致故障的最初错误。
随着我们回溯过去的时间越久远，进程的信息逐渐变得模糊。
核心越详细，我们可以回溯的时间就越远，直到画面变得过于模糊。
借助正确收集的核心文件和相关信息，我们通常可以解决问题，或者以其他方式提取故障进程的有价值信息。对于人为生成的核心文件，通常我们只能说（在经过数小时分析后）："我们到这个进程发生了什么，有人手动生成进程核心。"手动生成核心文件可以包含更多正在运行进程的信息，它可用作辅助信息源，以补充无法从系统默认核心文件提供的信息细节。InterSystems 产品可以配置为在发生任何进程故障时都记录完整核心。
这对日常操作的性能没有任何影响。
您只需确保有足够的磁盘可用空间，以应对任何潜在的、小概率发生的故障。InterSystems 只要能获得完整核心，在解决问题方面都有着良好的记录。
有时我们会发现由难以解决的硬件引发的故障，并保证这种故障不会再次发生。InterSystems 产品还可以配置为很少记录或不记录进程故障的信息。虽然禁用核心没有性能优势，但您可能会获得运行优势。
核心文件可能包含敏感信息。
如果您不想制定操作核心文件的策略，可以只在反复出现故障后再启用生成核心文件。InterSystems 产品默认采用中间方案。也就是生成限大小的核心文件。利用这些小核心文件，InterSystems 通常可以识别是否为以前解决过的问题，并解决这些问题。
当然我们无法保证使用默认的有限的核心文件可以解决所有问题。
用于确定所获得的核心文件的大小和类型的主要控制参数是 DumpStyle。这是 cache.cpf 或 iris.cpf 文件中的参数。
还有其他几个特定于操作系统的控制参数。 DumpStyle 的说明在这里：。 DumpStyle 取一个介于 0 到 8 之间的整数值，适用于 Caché、Ensemble、HealthShare 或 InterSystems IRIS 数据平台实例中的每个进程，并定义当进程遇到严重错误时，应保存那种核心（或进程转储）文件。定义的值为：

??	??	??	??
0	NORMAL	Unix OpenVMS Windows	生成整核心（取决于其他设置）。 生成

			CACCVI O-pid.LOG（大小受限）。 生成 pid.dmp （大小受限）。
1	FULL	Unix OpenVMS Windows	生成整核心（取决于其他 设置）。 生成 CACHE.DMP （可能非常大）。 生成 cachefpid.dm p（可能非常大）。
2	DEBUG	Unix OpenVMS Windows	在 Caché2014.1 之前，生成 省略共享内存的核心，现已 弃用。最好使用操系统特 定的方法省略共享内存。 未实现。 为 InterSystems 预留。
3	INTERMEDIATE	Unix OpenVMS Windows	未实现。 未实现。 2014.1 版本后有 效，生成 cacheipid.dmp
4	MINIMAL	Unix OpenVMS Windows	未实现。 未实现。 2014.1 版本后有 效，生成 cacheipid.dmp
5	NOHANDLER	Unix OpenVMS Windows	不 注 册为信号处理程序。将 有关核心创建的所有决定都 留给操系统。 未实现。 未实现。
6	NOCORE	Unix OpenVMS Windows	不生核心文件。 未实现。 未实现。
7	NOFORK	Unix OpenVMS Windows	创建核心dump（包含共享内 存），但是从原始故障进程 而不是从故障进程的分支副 本进行创建。 未实现。 未实现。
8	NOFORKNOSHARE	Unix OpenVMS Windows	创建不包含共享内存的核心 转储，但是从原始故障进程 而不是从故障进程的分支副 本进行创建。 未实现。 未实现。

默认 DumpStyle 为 0 = NORMAL，除了在 Windows 中从 Caché 2014.1 开始，为 3 = INTERMEDIATE。

更改 DumpStyle 值的方法有三种。分别是：

①将以下部分放在 cache.cpf 或 iris.cpf 文件中，为此，需使用操系统的文本编辑器：

[Debug]

dumpstyle=1

等号后面的数字是新的默认 DumpStyle值。重新启动 Caché、Ensemble、HealthShare 或 InterSystems IRIS 数据平台后，如果不使用下面的方法 ② 或 ③，它会将这个新的值设定为所有进程的默认值。

②使用命令：

SET old=$SYSTEM.Config.ModifyDumpStyle(1)

括号中的数字是新的 DumpStyle 值。旧值将返回。此命令对运行后创建的所有新进程都有效。
现有进程继续以先前的 DumpStyle 运行。此命令从 Caché 2014.1 开始有效。

对于较新版本，可以使用以下命令：

VIEW $ZUTIL(40,2,165):-2:4:1

其中新的 DumpStyle 值是最后一位数字。

③执行以下命令或将其放在应用程序中：

VIEW $ZUTIL(40,1,48):-1:4:1

其中新的 DumpStyle 值是最后一位数字。这只对执行命令的进程有效，并覆盖方法 ① 和 ②的设定值。

大多数操作系统都可以控制将生成核心文件重定向到一个公用目录，并控制核心文件中包含的信息数量。
您也应该手动检查这些设置，并且应考虑相应的影响，尤其是从数据隐私的角度。
以下节介绍了各个操作系统的详细信息。

将核心文件的输出移动到一个公用目录对于容量规划非常有用，但也可能让窥探从您的站点窃取数据的人更容易访问到核心文件。

如果核心文件不包含共享内存，很多类型的问题无法从根本进行解决。
包含共享内存的核心文件往往比不包含共享内存的核心文件大得多。大部分差异是global和routine缓冲区的大小。

如果你正在处理的是敏感信息，不含共享内存的核心文件将只包含发生故障的进程正在处理的敏感信息。
含有共享内存的核心文件还将包含每个进程最近访问的所有全局变量。
这里用到的“最近”可能表示几分钟，或者相当长的时间。

AIX

生成整的核心文件应使用 smit 启用：

```
System Environments

> Change / Show Characteristics of Operating System

> > Enable full CORE dump                                  true
> > Use pre-430 style CORE dump                            false
```

也可以从命令行使用以下命令查看：

```
#
lsattr -E -l sys0 | egrep 'fullcore|pre430core'?
fullcore          true             Enable full CORE dump              True
```

```
pre430core          false             Use pre-430 style CORE dump         True
```

使用以下命令设置：

```
#
chdev -l sys0 -a fullcore=true -a pre430core=false -P?
```

`-P` 使更改永久有效。

◆

默认情况下，当进程发生故障时在进程的默认目录写入核心文件。该目录通常是主 CACHE.DAT 或 IRIS.DAT 文件所在的目录。这可以通过 smit 更改：

```
Problem Determination
> Change/Show/Reset Core File Copying Directory
```

或从命令行运行：

```
#
chcore -p on -l /cores -n on -d?
```

◆

确保件 /etc/security/limits 的某个部分包含以下行：

```
default:
 core = -1
```

最后，无论通过什么方式为用户进程设置环境变量，都确保每个用户都已根据需要定义或未定义 CORE_NOSHM。如果定义了 CORE_NOSHM=1，核心文件不包含共享内存的内容。如果定义 CORE_NOSHM=0 或完全未定义，核心文件将包含共享内存的内容。为所有用户设置此参数的简单方法是编辑 /etc/environment 并包含以下行：

```
CORE_NOSHM=1
```

要分别指定每个用户是否在核心文件中包含共享内存的内容，请根据用户和他们使用的 shell 编辑以下任一文件：

```
CORE_NOSHM=1; export CORE_NOSHM     # sh in /etc/profile or $HOME/.profile
export CORE_NOSHM=1                 # ksh in /etc/.kshrc or $HOME/.kshrc
export CORE_NOSHM=1                 # bash in /etc/bashrc or ~/.bashrc
setenv CORE_NOSHM 1                 # csh in ~/.cshrc
```

Docker

InterSystems IRIS 数据平台 docker 容器的核心文件创建由主机 Linux 系统控制。
您必须规划将核心文件直接发送至操作系统文件。该文件可以保存在 docker 容器内，也可以发送至映射的主机 Linux 系统上的目录中。将核心文件发送至映射的主机 Linux 系统上的目录的优点是，它将在容器完全失效的情况下仍然保存。

由于核心文件必须发送到操作系统文件，因此必须在主机平台上禁用所有高级核心捕获软件。
您将需为主机和容器系统的 /proc/sys/kernel/core_pattern 设置适当的值。

您应该选择一个相对简单并且在主机和容器上都存在的目录（ /tmp 或 /cores 显然是最佳选择）。
您可能还需包含变量，以确保每个 docker 容器的核心不会相互覆盖。因此
/cores/core.%p.%e 是一个好选择。

??	??	??
RedHat Linux	必须禁用错误自动报告工具 (ABRT)。	
SuSE Linux	截至 SuSE Linux Enterprise Server 11，SuSE 没有任何高级核心捕获软件。因此您只需照说明设置 /proc/sys/kernel/core_pattern。但是，到目前为止，我们还没有提供在 SuSE Linux Enterprise Server 12 中禁用高级核心捕获软件的说明，因此 SuSE 12 及更高版本目前不适合作为 docker 容器的主机。	
Ubuntu Linux	必须禁用 apport。	

启动容器时，您可能需包含将用于核心文件的目录映射到主机操作系统的选项。因此：

```
#

docker run ? -v /cores:/cores ? &#x21a9;
```

如果不包含 -v /cores:/cores，则 docker 容器内的进程故障所创建的任何核心文件都只会在 docker 容器运行时存活。如果 -v
选项给出的映射不是对称的，即冒号左侧和右侧的值不一样，则可能无法捕获某些核心文件内容。
将核心文件大小设置为无限制。由于这是运行时决定，因此将以下内容添加到 docker run 命令中：

```
#

docker run ? --ulimit core=-1 ? &#x21a9;
```

HP-UX

使用以下命令使核心文件保存在公用目录中并带有扩展名称：

```
# coreadm -e global -g /cores/core.%p.%f?
```

%p 表示将 pid 放在路径名中，%f 表示将可执行文件名（如 cache 或 iris）放在路径名中。请参见：

```
%

man 1m coreadm&#x21a9;
```

了解更多选项。

◆

使用以下命令查看是否已在核心文件中启用共享内存：

```
#
/usr/sbin/kctune core_addshmem_read?
#
/usr/sbin/kctune core_addshmem_write?
```

更改为：

```
#

/usr/sbin/kctune core_addshmem_read=1&#x21a9;

#

/usr/sbin/kctune core_addshmem_write=1&#x21a9;
```

1 表示启用， 0 表示禁用。HP – UX 将共享内存分为两种类型。通常，InterSystems 仅使用写共享内存，但我们建议将两种类型设置为相同值。

◆

在 HP – UX 上，生成核心文件大小受 maxdsiz_64bit 内核参数限制。确保该参数设置得足够高，以便完整的核心文件。可

使用以下命令查看：

```
#

/usr/sbin/kctune maxdsiz_64bit&#x21a9;
```

使用以下命令设置：

```
#

/usr/sbin/kctune maxdsiz_64bit=4294967296&#x21a9;
```

用户可以使用 ulimit -c 命令进一步限制核心文件大小。应该从 /etc/profile、$HOME/.profile 以及其他 shell 的类似文件中删除此设置，除非您有意限制核心文件。

RedHat Linux

如果您正在运行 Rhel 6.0 或更高版本（ CentOS 同样），RedHat 已增加了错误自动报告工具 (ABRT)。此工具安装后与 Caché、Ensemble、HealthShare 或 InterSystems IRIS 数据平台不兼容。您需要确定您是将 ABRT 配置为支持 Caché、Ensemble、HealthShare、InterSystems IRIS 数据平台，还是禁用 ABRT。

以标记了 ABRT 的部分适用于使用 ABRT 的情况，
而标记了 AB/RT 的部分适用于传统的不使用 ABRT 的情况。

◆

ABRT 要使 InterSystems 产品与 ABRT 兼容，请确定正在运行的 ABRT 版本：

```
#
abrt-cli --version?
```

编辑 ABRT 配置文件。名称会因 ABRT 版本的不同而有所差异：
ABRT 1.x：/etc/abrt/abrt.conf
ABRT 2.x：/etc/abrt/abrt-action-save-package-data.conf
如果您使用 cinstall 命令安装了 Caché、Ensemble 或 HealthShare（最常见），或使用 irisinstall 命令安装了 InterSystems IRIS 数据平台，请找到 ProcessUnpackaged= 行，将值更改为 yes。

```
ProcessUnpackaged = yes
```

否则，如果是从 RPM 模块安装的 Caché、Ensemble、HealthShare 或 InterSystems IRIS 数据平台，则找到 OpenGPGCheck= 行，将值更改为 no。

```
OpenGPGCheck = no
```

不管是如何安装的 Caché、Ensemble、HealthShare 或 InterSystems IRIS 数据平台，都找到 BlackListedPaths= 行，并添加对 installation/bin 目录中的 cstat 或 irisstat 的引用。如果 BlackListedPaths= 行不存在，则在末尾添加此行以及 cstat 或 irisstat 引用。

```
BlackListedPaths=[retain_existing_list,]installation_directory/bin/cstat
```

保存做编辑，然后重新启动 abrtd：

```
#
```

```
service abrtd restart&#x21a9;
```

这样配置后，ABRT 会为每次进程故障创建一个新目录（在 /var/spool/abrt 或 /var/tmp/abrt 中），并在该目录中放置核心文件以及相关信息。

当进程发生错误时，执行命令：

```
#
```

```
abrt-cli --list&#x21a9;
```

```
# for ABRT 1.x  #
```

```
abrt-cli list&#x21a9;
```

```
# for ABRT 2.x
```

这将显示最近进程故障的列表，并为每个故障提供一个目录规范。每个目录中都将有一个 coredump 文件，以及许多其他小文件，这些文件对于确定进程故障的原因非常有用。

```
%
  tar -cvzf <var>wrcnumber</var>-core.tar.gz /var/spool/abrt/<var>directory</var>/*?
```

其中 wrcnumber 是 InterSystems 分配用来调查案例的编号。您可以将压缩的 wrcnumber-core.tar.gz 文件发送给我们。

◆

AB/RT 或者，可以使用以下命令禁用 ABRT：

```
#
 service abrtd stop?
#
 service abrt-ccpp stop?
     # ABRT 2.x only.
```

要永久禁用 ABRT：

```
#
 chkconfig abrtd off?
#
 chkconfig abrt-ccpp off?
     # ABRT 2.x only.
```

最后，您需要更新 /proc/sys/kernel/core_pattern，请参见下一节。

◆

AB/RT 您可以控制核心文件的存储位置（除非您正在使用 ABRT）。

①：如果正在使用 ABRT，则必须跳过此步骤。

②：如果已禁用 ABRT，则必须执行此步骤。

③：如果从未安装 ABRT，则此步骤是可选的。

编辑文件 `/proc/sys/kernel/core_pattern` 在简单的示例中，只需使用：core 添加生成核心的程序的 pid 和名称通常很有用：core.%p.%e 还可以将核心放在公用目录中：/cores/core.%p.%e
确认所有用户对所选目录都具有写访问权限。请参见 `man core` 了解更多选项。您应该在目录 /etc/sysctl.d 中创建一个名称以 `.conf` 为结尾的文件，并包含以下内容，从而使此更改永久有效：
kernel.core_pattern=/cores/core.%p.%e

◆

ABRT AB/RT 您应该设置 /proc/self/coredump_filter 以控制转储到核心文件中的内存大小。这可以在 /etc/profile.d/*some-thing*.sh 文件中设置。命令为：

```
#

echo 0x33 &gt;/proc/self/coredump_filter&#x21a9;
```

所使用的具体映射取决于您要收集的数据级别。bits的含义可以在 man core 中找到，对于 InterSystems 产品有意义的示例为：

Bit	??	InterSystems ????
0x01	匿名私有映射。	始终需
0x02	匿名共享映射。	复杂问题需
0x04	文件支持的私有映射。	$ZF() 相关问题可能需
0x08	文件支持的共享映射。	$ZF() 相关问题可能需
0x10	转储 ELF 标头。	始终需
0x20	转储私有大页。	InterSystems 当前未使用。
0x40	转储共享大页。	InterSystems 当前未使用。
0x80	转储私有 DAX 页 (Rhel 8)。	InterSystems 当前未使用。
0x100	转储共享 DAX 页 (Rhel 8)。	InterSystems 当前未使用。

作为将此命令放在 shell 特定的脚本中的替代方案，您可以在启动期间进行修 只有使用 grub2 引导时，这些指令才适用。您可以通过以下命令测试：

```
#
 grub2-install --version?
grub2-install (GRUB) 2.02~beta2
```

编辑 /etc/default/grub。更改以 GRUB_CMDLINE_LINUX_DEFAULT= 开头的行。
如果文件中不存在该行，则在末尾添加。它应该包含：

```
GRUB_CMDLINE_LINUX_DEFAULT="oldcmd
 coredump_filter=newval
"
```

注 意：oldcmd 是 GRUB_CMDLINE_LINUX_DEFAULT 的旧值（如果该行先前不存在，则省略）。 newval` 是 coredump_filter 的新值，以十六进制表示，带有前导“`0x`”。

运行：

```
#
 grub2-mkconfig -o /boot/grub2/grub.cfg?
```

◆

ABRT AB/RT 您应该针对所有进程设置 ulimit -c 为无限制。这可以在文件 /etc/security/limits.conf 中全局设置。
添加以下两行：

```
 *           soft    core    unlimited
 *           hard    core    unlimited
```

SuSE Linux

如果运行 SuSE Linux Enterprise Server 12 或更高版本，SuSE 现在将所有生成的核心文件存储在 systemd 日志中。存储在 systemd 日志中的核心文件是临时的。它们在系统重启后即消失。
如果需 必须在系统重启前从 systemd 日志中提取核心。要列出 systemd 日志中当前含有的核心文件：

```
# [
  systemd-
]
 coredumpctl list?
```

要提取按照创建了核心的 pid 选择的核心文件：

```
# [
 systemd-
]
 coredumpctl -o core.morename dump pid
```

注 意：从 SuSE 12-SP2 开始， systemd- 前缀已从命令名称中删除。建议留此 systemd 行为，不要试图修改。

如果正在运行旧版本的 SuSE Linux Enterprise（11 或更早版本），可以通过编辑文件 /proc/sys/kernel/core_pattern来控制存放核心文件的位置。

在简单的示例中，只需用：

```
core
```

通常也需添加生成核心文件程序的 pid 和名称，使用：

```
core.%p.%e
```

还可以将核心文件放在公用目录中：

```
/cores/core.%p.%e
```

确认所有用户对所选目录都具有写访问权限。请参见 man core 了解更多选项。

可以将以几行追加到文件 etc/sysctl.conf 中来使此更改永久有效：

```
# Make this core pattern permanent (SuSE 12 breaks this, don't use):

kernel.core_pattern=/cores/core.%p.%e
```

◆

您应该设置 /proc/self/coredump_filter以控制转储到核心文件的内存数量。这可以在适当的 /etc/profile.d/*something*.sh文件中运行。命令为：

```
# echo 0x33 >/proc/self/coredump_filter?
```

所使用的具体bits映射取决于您想收集的数据级别。bit 的含义可以在 man core 中找到，对于 InterSystems 产品有意义的示例为：

Bit	??	InterSystems ????
0x01	匿名私有映射。	始终需
0x02	匿名共享映射。	复杂问题需
0x04	文件支持的私有映射。	$ZF() 相关问题可能需
0x08	文件支持的共享映射。	$ZF() 相关问题可能需
0x10	转储 ELF 标头。	始终需
0x20	转储私有大页。	InterSystems 当前未使用。
0x40	转储共享大页。	InterSystems 当前未使用。
0x80	转储私有 DAX 页 (SuSE 15)。	InterSystems 当前未使用。
0x100	转储共享 DAX 页 (SuSE 15)。	InterSystems 当前未使用。

作为将此命令放在 shell 特定的脚本中的替代方案，您可以在启动期间进行修改要执行此操作，请使用 yast2。根据您连接的是终端界面（将使用 curses 界面）还是 GUI 界面，yast2 的用户界面会有所不同。

以说明尽量做到与界面无关。

①：启动 yast2 后，从菜单中选择 System → Boot Loader。

②：选择 Kernel Parameters 选项卡。

③：查找 Optional Kernel Command Line Parameter字段。

④：如果该字段尚未包含 coredump_filter=0xvalue，则使用空格分隔符将其追加到该字段中。
如果已经包含该赋值，则只编辑 value。

⑤：退出菜单系统，然后重新启动。

◆

您应该针对所有进程设置 ulimit -c 为无限制。这可以在文件 /etc/security/limits.conf 中进行全局设置。
添加以下两行：

```
*          soft    core       unlimited
*          hard    core       unlimited
```

◆

注意：可能需要禁用 AppArmor，它会阻止它认为不寻常的应用程序行为，而写入核心文件可能被视为不寻常。

```
# rcapparmor stop?
```

Ubuntu Linux

Ubuntu 使用 apport 捕获所有进程故障，对于使用安装包添加的软件包，会创建 apport
报告，其中包含带有附加信息的经过编码和压缩的核心内容。当然也可以要求 apport 处理未使用 Ubuntu
的软件包管理器(Ubuntu's package manager)安装的应用程序的代码。不幸的是，如果这样做，Canonical
会将针对未打包代码创建的 apport 报告视为为改进 Ubuntu而收集的信息，进而进行内容检查。

由于可以从 apport 报告中提取数据，您几乎肯定不会启用对未打包代码的 apport 处理。您的唯一选择是禁用 ap-
port。为此，请编辑 etc/default/apport，然后编辑 enabled= 行：

```
enabled=0
```

◆

创建文件 /etc/sysctl.d/30-core-pattern.conf（或该目录中的任意相似名称）。在该文件中添加：

```
kernel.core_pattern=/cores/core.%p.%e
```

确保指定用于保存核心文件的目录可公开并写入，并且有足够的磁盘空间。请参见 man core 了解更多选项。

◆

您应该设置 /proc/self/coredump_filter 以控制转储到核心文件中的内存数量。
这可以在 /etc/profile.d/*something*.sh 文件中进行设置。命令为：

#

```
echo 0x33 &gt;/proc/self/coredump_filter&#x21a9;
```

所使用的具体bit 映射取决于您要收集的数据级别。bit 的含义可以在 man core 中找到，对于 Caché 有意义的示例为：

Bit	??	InterSystems ????
0x01	匿名私有映射。	始终需
0x02	匿名共享映射。	复杂问题需
0x04	文件支持的私有映射。	$ZF() 相关问题可能需
0x08	文件支持的共享映射。	$ZF() 相关问题可能需
0x10	转储 ELF 标头。	始终需
0x20	转储私有大页。	InterSystems 当前未使用。
0x40	转储共享大页。	InterSystems 当前未使用。
0x80	转储私有 DAX 页 (16.04LTS)。	InterSystems 当前未使用。
0x100	转储共享 DAX 页 (16.04LTS)。	InterSystems 当前未使用。

作为将此命令放在 shell 特定的脚本中的替代方案，您可以在启动期间进行修 只有使用 grub2 引导时，这些指令才适用。您可以通过以下命令测试：

#

```
grub-install --version&#x21a9;

grub-install (GRUB) 2.02-2ubuntu8.12
```

编辑 /etc/default/grub。更改以 GRUB_CMDLINE_LINUX_DEFAULT= 开头的行。
如果文件中不存在该行，则在末尾添加。它应该包含：

```
GRUB_CMDLINE_LINUX_DEFAULT="oldcmd


coredump_filter=<em>newval
</em>

"
```

注 意：oldcmd 是 GRUB_CMDLINE_LINUX_DEFAULT 的旧值（如果该行先前不存在，则省略）。 newval 是 coredump_filter 的新值，以十六进制表示，带有前导“ 0x”。运行：

#

```
grub-mkconfig -o /boot/grub2/grub.cfg&#x21a9;
```

◆

您应该针对所有进程设置 ulimit -c 为无限制。这可以在文件 /etc/security/limits.conf 中进行全局设置。
添加以下两行：

```
*         soft    core    unlimited
*         hard    core    unlimited
```

macOS?OS X?Darwin?

Mac OS X 曾重命名为 OS X，后来又重命名为 macOS。所有这些操系统都是 Apple 在 Darwin 上堆叠的专有用户界面，Darwin 是 Apple 从 BSD Unix 衍生出来的操系统，理论上已发布到公共领域。然而，以 Apple 发布 Darwin 的方式，实际上没有人会只运行 Darwin。

InterSystems 产品只需Darwin，但由于 Darwin 实际上不可用，因此所有说明均基完整的 Apple Mac OS X、OS X 或 macOS。

macOS 包括 CrashReporter。该工具可以自动拦截进程故障，将故障详细信息打包为文本日志，然后将数据发送给 Apple 进行分析。CrashReporter 将捕获第三方软件（如 Caché、Ensemble、HealthShare 和 InterSystems IRIS 数据平台）的进程故障详细信息。理论上，Apple 可将这些信息转发给 InterSystems。

InterSystems 没有从 Apple 收到 CrashReporter 日志，我们也没有开发分析这些日志的功能。InterSystems 产品严格地生核心文件。幸运的是，CrashReporter 独立于核心文件创建。也就是说，处理进程故障可以通过 CrashReporter 和核心文件创建这两种方式中的任意一种或两种，或者都不使用。

CrashReporter 设置可以在“系统偏好设置”→“安全和隐私”、“隐私”选项卡中设置。面板名称和框的选择因版本而异。
在 Mac OS X 10.4 中，该面板的名称仅为“安全性”没有相关的复选框。
在较旧的版本中，发生任的进程故障时都会向用户显示一个对话框，询问他们是否要将数据发送到 Apple 进行分析。根据您处理的数据的敏感性您可能想要取消选中与 CrashReprter 相关的所有选项。

◆

在 macOS 中，启用生核心文件的方法在不同版本之间有很大变化。请参见下表，并使用适合您的版本的方法。

??	??	InterSystems ??	??
公测版	Kodiak	不支持	方法 1：编辑 /hostconfig
Mac OS X 10.0	Cheetah	不支持	
Mac OS X 10.1	Puma	不支持	
Mac OS X 10.2	Jaguar	不支持	
Mac OS X 10.3	Panther	Caché (PowerPC) 5.0, 5.1	
Mac OS X 10.4	Tiger	Caché（标记 PowerPC 或 x86）5.0PowerPC, 5.1PowerPC, 5.2*, 2007.1*, 2008.1x86, 2008.2x86, 2009.1x86	方法 2：编辑 /etc/launchd.conf
Mac OS X 10.5	Leopard	Caché (x86) 2008.1, 2008.2, 2009.1, 2010.1	
Mac OS X 10.6	Snow Leopard	Caché (x86 – 64) 2010.1, 2010.2, 2011.1, 2012.1, 2012.2	
Mac OS X 10.7	Lion	Caché (x86 – 64) 2011.1, 2012.1, 2012.2, 2013.1, 2014.1	
OS X 10.8	Mountain Lion	Caché (x86 – 64) 2012.2, 2013.1, 2014.1, 2015.1	
OS X 10.9	Mavericks	Caché (x86 – 64) 2013.1, 2014.1, 2015.1, 2015.2, 2016.1, 2016.2	
OS X 10.10	Yosemite	Caché (x86 – 64) 2014.1, 2015.1, 2015.2, 2016.1, 2016.2	方法 3：非自动。
OS X 10.11	El Capitan	Caché (x86 – 64) 2016.1, 2016.2, 2017.1DEV,2017.2DEV, 2018.1DEV	
macOS 10.12	Sierra	Caché (x86 – 64) 2017.1, 2017.2, 2018.1	
macOS 10.13	High Sierra	Caché (x86 – 64) 2018.1,	

		2019.1, 2019.2
macOS 10.14	Mojave	IRIS 2019.1, 2019.2
macOS 10.15	Catalina	未发布

方法 1：对于 OS X 10.3 (Cheetah) 以及先前不受支持的版本：编辑文件 /hostconfig。查找行 COREDUMPS=，然后将值更改为 -YES-。

```
COREDUMPS=-YES-
```

◆

方法 2：对于 OS X 10.4 (Tiger) 到 OS X 10.9 (Mavericks) 版本，编辑文件 /etc/launchd.conf，然后添加以行：

```
limit core unlimited
```

重新启动。

◆

方法 3：对于 OS X 10.10 (Yosemite) 及更高版本， /etc/launchd.conf 已去除。核心文件生成在是半禁用状态。用户必须为每个进程启用核心：

```
%
```

```
ulimit -c unlimited&#x21a9;
```

在允许他们的应用程序之前，特权用户必须运行：

```
#
```

```
launchctl limit core unlimited&#x21a9;
```

然后 注 销，并在启动 Caché 前再次登录。Apple 特地没有提供一个好方法来自动执行这些操作，因为他们认为默认生成核心文件是一个潜在的安全漏洞。

Apple 提供了完全禁用核心文件生成的方法。这通过编辑文件 /etc/sysctl.conf 并添加以行来完成

```
kern.coredump=0
```

可以通过删除该行或将值更改为 `1` 来重新启用此功能。

OpenVMS

默认情况下，Caché 和 Ensemble 只针对故障进程生成 CACCVIO-*pid*.LOG 文件。使用这些文件只能解决相对简单的问题。这些 CACCVIO-pid.LOG 文件将始终放在进程默认目录中（通常是 CACHE.DAT 文件的目录），只有通过更改进程默认目录才能重定向。

Caché 和 Ensemble 也可能生成 CERRSAVE-*pid*.LOG文件。这些文件与 CACCVIO-pid.LOG 文件类似。通常，您无需关心它们的差异。在某些情况下，Caché 和 Ensemble 将同时生成两种文件以响应故障。

在迄今为止的所有情况，CACCVIO-pid.LOG 文件就生成了其中包含错误的完整上下文，而 CERRSAVE-pid.LOG 文件则在进程的最后断开过程中生成，包含的有价值的信息相对较少。

如果启用了扩展进程 dumps（完整转储），它们也会放置在进程默认目录中。
但是，可以将它们重定向，方法是定义逻辑名称 SYS$PROCDMP，将其指向要存储进程dump的目录。
该逻辑名称可以在 /SYSTEM 级别定义。文件名将是 CACHE.DMP 或 CSESSION.DMP。

OpenVMS 还提供了逻辑名称 SYS$PROTECTED_PROCDMP。您也应该使用 /EXECUTIVE_MODE 和 /SYSTEM 定义该逻辑名称。这适用于特权映像（privileged images）的进程故障，Caché 的某些部分也拥有特权。
OpenVMS 文档中建议将这两个逻辑名称定义到不同的目录，并为与 SYS$PROTECTED_PROCDMP 对应的目录设置更高的安全性
这基于假设：特权映像处理的数据比非特权映像处理的数据更敏感。
如果两种数据都很敏感，将两个逻辑名称指向同一目录也是可以的。

◆

影响 CACCVIO-pid.LOG 和 CERRSAVE-pid.LOG 文件创建以及完整进程dump的缺陷是有历史记录的。
以下是最重要的变更。

??	?????	??
JLC1809	Caché 2015.2	在此变更之前，大多数 CERRSAVE-pid.LOG 文件是无用的。
JO2422	Cache 2012.1	在此变更之前，生成 CERRSAVE-pid.LOG 文件的条件在创建信息有限的文件时总是会忽略 DumpStyle。
JLC1326	Caché 2011.1	在此变更之前，Itanium 平台上的 CACCVIO-pid.LOG 和 CERRSAVE-pid.LOG 文件不包含寄存器。这严重阻碍了我们使用这些文件解决问题的能力，而只能处理简单问题。我们仍然可以与已经解决的问题相匹配。
JLC931 和 JLC959	Cache 2007.2	在此变更之前，Itanium 平台上的 CACCVIO-pid.LOG 和 CERRSAVE-pid.LOG 文件不记录任何有用信息。
JO1968	Caché 5.2	在此变更之前，生成 CACCVIO-pid 文件的条件在创建信息有限的文件时总是会忽略 DumpStyle。

Solaris

可以使用以下命令使核心文件放在公用目录中并带有扩展名称：

```
#
coreadm -e global -g /cores/core.%p.%f -G all?
```

① %p 表示将 pid 放在路径名中。

② %f 表示将可执行文件名（如 cache）放在路径名中。

③ -G all 表示包括所有类型的内存，即完整核心文件内容。
省略此选项将生成默认核心文件内容，其中仍包括大部分共享内存。核心文件中可以存储以下内容：

??	InterSystems ??	????
stack	需要	是
heap	需要	是
shm	不使用	是
ism	不使用	是
dism	Caché 共享内存	是

text	用于 $ZF() 相关故障	是
data	需	是
rodata	不使用	是
anon	需	是
shanon	通常较小	是
ctf	需	是
symntab	用于 $ZF() 相关故障	否
shfile	不使用	否

all 表示包括所有类型的内存，默认不包括最后两种。
如果您需大幅减小的核心文件大小以节省空间，但代价是可解决的问题变少），移除 dism 共享内存可节省出大部分空间。使用以下命令执行此操作：

```
#

coareadm -e Global -g /cores/core.%p.%f -G (default-dism)&#x21a9;
```

请参见：

```
%

main 1m coreadm&#x21a9;
```

◆

默认情况下，用户已经

```
%

ulimit -c unlimited&#x21a9;
```

您可以使用 ulimit（或在 csh 中使用 limit 命令）禁用核心，但 coreadm 通常更灵活。因此您应该确保 ulimit 命令不会出现在 /etc/profile 或 $HOME/.profile 中，或者其他 shell 的相应文件中。

Windows

Windows 的转储文件中包含的信息完全由 cache.cpf 文件中的 DumpStyle 参数（或上文定义的其他用于更改 DumpStyle 的接口）控制。

??

除了特定问题外，本地安全组也可能阻止核心文件的写入。
在真实条件下测试是否可以成功创建核心文件非常有用。为此，请输入命令：

```
USER>DO $ZUTIL(150,"DebugException")?
```

可以肯定的是，您应该在 JOB 内以交互方式测试该语句（假定您的应用程序使用 JOB 命令），甚至在应用程序内隐藏一个不会被用户意外选择到的选项。

验证是否会获得核心文件，并按照下一节的健全性查来验证它是否是好的核心文件。

?????

核心文件（和进程转储）可能非常大，并且可能包含 敏感信息 。在将核心文件传输到 InterSystems 进行分析之前，最好在生成核心文件的系统或非常相似的系统上对核心文件进行健全测试。
根据您的操作系统，请执行以下健全测试：

????	?????
AIX	# dbx cache core (dbx) set $stack_details (dbx) where (dbx) quit 向 WRC 开启问题时，将上述命令的输出结果也同时发送给我们。 如果您的系统未安装 dbx，则开启一个新问题。
HP – UX	# gdb cache core (gdb) frame 0 (gdb) while 1 > info frame > up > end (gdb) quit 根据您拥有的调试器，将上述两个命令集的其中之一的输出结果发送给我们。 如果两个调试器都有，则首选 gdb（实际是 Wildebeest）。 # adb core adb> $c adb> $q
RedHat Linux SuSE Linux	对于所有版本的 Linux 都使用此通用健全测试。 # gdb cache core (gdb) frame 0 (gdb) while 1 > info frame > up > end (gdb) quit 向 WRC 开启问题时，将上述命令的输出结果发送给我们。 如果您的系统未安装 gdb，则开启一个新问题。
Ubuntu Linux	
macOS (Darwin)	# lldb (lldb) target create -c core (lldb) thread backtrace all (lldb) quit 将 lldb 的输出结果（如果是 OS X 10.8 (Mountain Lion) 或更高版本）或 gdb 的输出结果（如果是 Mac OS X 10.7 (Lion) 或更早版本）发送给我们。 # gdb cache core (gdb) frame 0 (gdb) while 1 > info frame > up > end (gdb) quit
OpenVMS	$ ANALYZE/CRASH dumpfile.DMP SDA> SHOW CALL_FRAME/ALL 如果仍在运行 OpenVMS v7.x 或更早版本，先前的命令不适用，请改用： SDA> SHOW CALL_FRAME SDA> SHOW CALL_FRAME/NEXT 重复先前的命令，直到获得错误。 SDA> QUIT 将 SDA 或 debugger调试器 的输出结果发送给我们，但首选 SDA 的输出。 如果您仅有 CACCVIO-pid.LOG 文件，请检查其是否为空或几乎为空。 $ ANALYZE/PROCESS dumpfile.DMP DBG> SHOW CALL/IMAGE DBG> QUIT
Solaris	# mdb cache core > ::stackregs > ::quit 在 Solaris 上，InterSystems 对于几乎所有应用程序都首选 dbx 调试器，但对于健全测试， mdb 更好。当您向 WRC # dbx cache core (dbx) where (dbx) quit

开启问题报告时，请将 mdb 或 dbx 值选 mdb
）生成堆栈跟踪信息发送给我们。
Windows 对于 Windows 进程转储，当前没有推荐的健全性查。
请将健全测试的详细信息附加到 WRC 案例，或通过电子邮件发送到 support@intersystems.com 。
@intersystems.com>

??

请准备好向我们发送完整核心文件以及您的特定操作系统可能需要支持文件。我们需要知道生成核心文件的 Caché、Ensemble、HealthShare 或 InterSystems IRIS 数据平台的准确版本。
如果您重定向了软件并包含自定义的 ` $ZF()` 函数，请发送可执行文件。
（实际上，如果您总是发送可执行文件，会更加方便 在大多数 Unix 系统上，最好还发送可执行文件使用的库。
我们需要库的可能性给定的平台而异。请查阅下表：

????	??	???	????
AIX	PowerPC	不确定	A
HP – UX	PA – RISC	非常可能	C
HP – UX	Itanium	有可能	A
Linux（所有版本）	x86	有可能	A
Linux（所有版本）	x86_64	有可能	A
Linux（所有版本）	Itanium	有可能	D
macOS	PowerPC	不确定	D
macOS	x86	不确定	C
macOS	x86_64	不确定	A
OpenVMS	VAX	不适用	D
OpenVMS	ALPHA αxp	不适用	B
OpenVMS	Itanium	不适用	B
Solaris	x86_64	非常可能	B
Solaris	Sparc	不确定	B
Tru64 UNIX	ALPHA αxp	非常可能	D
Windows	x86	不适用	A
Windows	x86_64	不适用	A
Windows	Itanium	不适用	D

支持级别说明

A

：截至本文档发布之时，InterSystems 拥有用于诊断该平台上的核心文件的资源。

B

：对该平台的完全支持在最近失效。不过，InterSystems 仍然有资源来诊断平台上的核心文件。
某些诊断出的问题可能无法使用专门版本更正。

C

：旧版支持。InterSystems 可能仍然拥有有限的资源来诊断该平台上的核心文件，但是，可能无法再提供专门版本来修复所发现的任何缺陷。

D

：远古支持。InterSystems 未保留任何诊断这些平台上的问题的资源。不过一些有限的功能仍保留下来。
这些平台上的核心可能会被分析。但不可
能修复所发现的任何缺陷。

发出 ldd 命令可列出所需库：

```
# ldd install_directory/bin/image
        linux-vdso.so.1 => (0x00007fffd1320000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007f23e5002000)
        librt.so.1 => /lib64/librt.so.1 (0x00007f23e4dfa000)
        libstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007f23e4af0000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f23e47ee000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f23e45d8000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f23e4216000)
```

```
        /lib64/ld-linux-x86-64.so.2 (0x00007f23e521a000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f23e3ffa000)
```

上面的内容包含 Rhel 7 的示例输出。所有 Unix 系统的输出都是类似的。 install_directory`指 Caché、Ensemble、HealthShare 或 InterSystems IRIS 数据平台的安装目录。 image 对于 2018 年以前的所有产品均为 cache，对于自 2019 年以后的产品为 irisdb。
如果您要发送多个文件，最好将它们放在一个压缩的容器文件中。一般来说， .ZIP是最好的容器。 .tar.gz 也是合理的。对于 OpenVMS，使用以下命令创建备份文件：

```
$ BACKUP *.* [-]<var>saveset</var>.BCK/SAVE/DATA=COMPRESS
```

包含一个说明所发送文件的manifest会有帮助。请准备纯文本文件形式的manifest。
如果以电子方式发送数据，请不要加密文件，而是使用加密的传输方式。

您可以使用以下任意方法向 InterSystems 发送核心文件：

??	???	????
直接上传至 WRC 应用 。上传文件前，必须开启一个 WRC 问题，上传文件后，可以选择将问题标记为更高安全性问题标记为更高安全性，所有对与调查相关的文件的访问都仅限于实际进行调查的人员。除了 300 M 的附件大小限制外，还有 60 秒的时间限制，因此如果您的有效带宽小于 42 Mbps，最大上传量会减少	安全或更高	300 M 和 60 秒
电子邮件 。一般来说，除了不要任客户数据就可以进行调查的简单问题，应避免使用电子邮件。示例：您刚刚在一台新计算机上安装了 Caché，但它在启动时出现故障，并生成一个小核心文件。通过电子邮件发送该文件是合理的。	不安全	40 M
我们的 kite-works 服务器。 对于任何给定问题，您都必须申请数据上传链接。这些链接在 30 天或更短时间内过期。这是上传安全数据的首选方法。绝对大小限制是我们服务器上的可用空间。但是，由于大多数客户都使用这种方法，如果您要上传大于 4G 的文件，请事先告知。	安全	> 4 G
我们的 sftp 服务器 。您必须申请特定于用于问题的目录。将为问题创建一个目录。对于更高安全性问题，我们创建了一个限制访问的机器（或虚拟机），并实现了一个自动流程将上传的任何文件都移动到该机器。绝对大小限制是指我们服务器上的可用空间，如果您要上传大于 100 G 的文件，请事先告知。	更高	> 100 G
您的 ftp/sftp 服务器 。您要求我们从中下载数据的服务器必须归您所有，并且您有完全控	取决于您	?

制权。InterSystems 不会从任何第三方服务器下载数据。第三方服务器被视为安全风险。

SecurLink。InterSystems 可以通过我们的 SecurLink 远程控制工具直接从您的网络上的任何已核准的计算机中下载文件。没有绝对大小限制。但是，如果您通过 V.90 调制解调器连接到互联网，我们下载一个 3 Gio 的核心文件需要一周的时间。	安全且更高	?
物理介质。您可以将物理介质邮寄给您当地的 InterSystems 办公室。InterSystems 可以读取介质，并将数据发送到我们的剑桥办公室，在那里进行大多数核心文件分析。大多数办公室都可以处理 U 盘以及 ISO 9660 光盘介质。我们的剑桥办公室可以处理多种磁带格式。在发送任何介质之前，您应该先与 InterSystems 确认。如果通过挂号（非认证）邮件发送介质，那么数据可以被认为是安全的（可能安全性更高）。	视情况	?

务必要记住，我们需要文件有些是二进制文件，有些是文本文件。对于某些文件传输方法（尤其是在不同的操作系统之间），必须指定文件是二进制还是文本，以防止文件被损坏。

??

a

ABRT ,
AppArmor
abrtd
abrt-cli
apport

b

BlackListedPaths

c

CACCVIO-pid.LOG ,
CACHE.DMP ,
CERRSAVE-pid.LOG
CentOS
CORE_NOSHM
CrashReporter
CSESSION.DMP
cachefpid.dmp
pid.dmp
cachempid.dmp
cache.cpf ,
chcore
chdev
chkconfig

r

RPM
rcapparmor

s

SYS$PROCDMP
SYS$PROTECTED_PROCDMP
$SYSTEM.Config.ModifyDumpStyle
敏感信息
smit
systemd

u

ulimit -c , , ,
/usr/sbin/kctune

y

yast2

z

$ZUTIL(40,1,48)
$ZUTIL(40,2,165)
$ZUTIL(150,"DebugException")

#提示和技 #Caché

源 URL: https://cn.community.intersystems.com/post/%E4%BB%80%E4%B9%88%E6%98%AF%E6%A0%B8%E5%BF%83%E6%96%87%E4%BB%B6%EF%BC%8C%E5%AE%83%E4%BB%AC%E4%BB%80%E4%B9%88%E6%97%B6%E5%80%99%E6%9C%89%E7%94%A8