*Collecting diagnostic data for **IBM Tivoli System Automation for Multiplatforms** (TSAMP) and **Reliable Scalable Cluster Technology** (RSCT) in a DB2 HADR or DB2 HA Share Disk environment.*

July 8th, 2013

Author:
Gareth Holl, IBM Tivoli Support (gholl@us.ibm.com)

Contents

General information for getsadata

The getsadata utility will gather detailed information about the Tivoli System Automation configuration , the RSCT domain, and the automation policy (resource model).

It can collect a variety of useful logs including the install logs and system logs. It can format and collect the IBM.RecoveryRM, IBM.GlbResRM, IBM.Test, and IBM.StorageRM traces.

RSCT's ctsnap utility can also be run and its output collected. The getsadata utility will tar & compress all output into a single tarball and provide the final filename details and location after it has completed executing.

The utility generally needs to be run on each server in the domain in order to have a complete diagnostic picture of the activity within the domain, though this requirement may not be necessary for all types of problems. Also, maximum data collection is only possible if the domain is online. If it is necessary to collect limited data whenthe domain is offline, you will be prompted to help getsadata decide how to best proceed ... read the on-screen prompts provided.

Important points to note:

1. Execute 'getsadata' with root authority

2. Run getsadata on all nodes in the domain where possible, but only after first running it on the node hosting the "master" automation engine (IBM.RecoveryRM), identified by using either of the following commands (executed from any node):

```
lssamctrl -V
lssrc -ls IBM.RecoveryRM | grep -i master
```

3. It is important to collect data as soon as possible after a problem is observed in order to collect all log and trace data before data is lost (First In, First Out, fixed size trace files). This doesn't necessarily apply if your environment has trace spooling enabled.

4. It is equally important to run the utility before any manual (user) recovery efforts are attempted. This will ensure an accurate snapshot of the current states which can then be correlated with the logs and traces collected.

The latest version of the TSAMP data collector (getsadata) can be obtained via the following technote :
http://www.ibm.com/support/docview.wss?&uid=swg21285496

Data collection strategy

TSAMP's data collection strategy consists of two options:

1. Complete Data Collection

If time permits, perform a **complete data collection** from all nodes immediately after observing a problem and before any recovery attempts.

Note: Execution time for *getsadata* using the maximum collection option (*'getsadata -all'*) is unpredictable as run times will vary from cluster to cluster and can be affected by other activities going on at the time. You may benefit from benchmarking a maximum collection in order to determine the typical run time in your specific environment.

OR

2. Quick Snapshot and Deferred Data Collection

Take a **quick snapshot** of the problem state immediately after observing a problem and before any recovery attempts, and then follow-up with a **complete data collection** immediately after recovery efforts have been completed.

- ⇨ This option assumes "Trace spooling" is enabled (see Appendix)
- ⇨ The amount of space made available in the trace spool directory will dictate how long you have before a **complete data collection** would be necessary to prevent eventual loss of trace data, so the goal should still be to collect data as soon as possible after an observed problem.
- ⇨ getsadata version 8.2.3 or greater is required (see download URL on previous page)

See the next page for actual usage instructions.

Sending the collected data to IBM Support

At any time, you can run the getsadata utility with just the –ftphelp flag and it will only display instructions on how to name the file and how to FTP to IBM :

```
./getsadata -ftphelp
```

Using Binary Transfer Mode, anonymously FTP the final tar file (ppppp.bbb.ccc.tar) to IBM:
Server: "ftp.emea.ibm.com"
Within directory: "/toibm/tivoli/"
An automatic update will be added to the PMR once the data arrives.

Please include some details about the data where appropriate, for example a timeline of events (when the problem occurred and when actions were performed, both before and after the observed problem).

Using the getsadata collector

__Option 1__ – "Complete data collection" immediately after observing a problem:

Step 1: Perform a complete data collection on the node hosting the "master" automation engine :

⇨ Determine node hosting master IBM.RecoveryRM process:

```
lssamctrl -V
```

⇨ Run getsadata on the master node as follows:

```
./getsadata -all
```

Step 2: A limited data collection should then be performed on each of the other nodes (this step can be performed in parallel for the remaining nodes):

```
./getsadata -ctsnap -env -logs -traces -noprompt
```

Note: If trace spooling is enabled, add the following cmdline option in both the above steps:

```
-spoolfrom YYYY-MM-DD.hh:mm
```

where YYYY-MM-DD.hh:mm should be the time just before the problem event

__Option 2__ – "Quick snapshot" and deferred "complete data collection":

When trace spooling is enabled for a particular environment, this allows a complete data collection to be executed at a later time, with less risk of trace data loss, so that recovery efforts can be the primary focus at the time a problem is discovered. For trace spooling setup details, refer to the Appendix.

Although trace spooling allows a delayed period before complete data collection, snapshot data is still very important, especially when trying to interpret the description of the reported problem.

Step 1: Run getsadata with the -quicksnap option, preferably on the current master node, to take a quick snapshot of the problem, prior to recovery steps :

⇨ Determine node hosting master IBM.RecoveryRM process:

```
lssamctrl -V
```

⇨ Run the following on the master node :

```
./getsadata -quicksnap
```

Step 2: Carry out whatever recovery steps are necessary.

Step 3: As soon as possible, a complete data collection should be performed on the "master" node :

```
./getsadata -all -spoolfrom YYYY-MM-DD.hh:mm -spoolto YYYY-MM-DD.hh:mm
```

Step 4: Finally a limited data collection should be performed on all other nodes:

```
./getsadata -ctsnap -env -logs -spoolfrom YYYY-MM-DD.hh:mm -spoolto
YYYY-MM-DD.hh:mm -noprompt
```

where YYYY-MM-DD.hh:mm should be the from/to times to limit data collection

Appendix

Configuring trace spooling for TSAMP and RSCT

Detailed trace files are extremely dynamic and would be voluminous and ever (rapidly!) growing if written as a normal file. Instead, the trace data is written to a circular file of fixed size. When the EOF is reached, the file pointer is moved back to the top of the file, and older data at the head of the trace file is overwritten, and so on ad infinitum. Because of this fact, in order to capture meaningful, detailed trace data that corresponds to the exact moment the incident of interest occurred, getsadata must be run within minutes (or possibly even seconds!) of the incident that we need to analyze. This is, of course, is impractical, in most cases. Trace spooling is a feature of RSCT that enables trace data to be "spooled" to a permanent directory before the circular file has wrapped and the detailed trace data is forever lost.

To prevent loss of trace data due to wrapping of standard trace files, use the configuration below to create and continually save round-robin trace files for IBM.RecoveryRM and IBM.GblResRM (the two core TSAMP daemons), to a specified directory. Check if a trace.conf file already exists in /var/ct/cfg. If it does not exist, then create a new one and paste the following text shown in green:

```
recrm_spool_summary:
pat=/var/ct/.*/log/mc/IBM.RecoveryRM/trace_summary
spooling = ON
pages = 4
dest = /trcspool/
size = 1024000

recrm_spool_pub:
pat=/var/ct/.*/log/mc/IBM.RecoveryRM/trace_pub
spooling = ON
pages = 3
dest = /trcspool/
size = 1024000

recrm_spool:
pat=/var/ct/.*/log/mc/IBM.RecoveryRM/trace
spooling = ON
pages = 8
dest = /trcspool/
size = 4096000

gblresrm_spool_summary:
pat=/var/ct/.*/log/mc/IBM.GblResRM/trace_summary
spooling = ON
pages = 4
dest = /trcspool/
size = 2048000

gblresrm_spool:
pat=/var/ct/.*/log/mc/IBM.GblResRM/trace
spooling = ON
pages = 8
dest = /trcspool/
size = 4096000
```

The stanzas shown above enable trace spooling for IBM.RecoveryRM and IBM.GblResRM for the trace and trace_summary files. The “size” attribute is for the individual trace files and does not need to be changed … the values above are chosen to help improve the data collection process via getsadata.

Note: The "dest" line specifies the location to spool the trace files to. If you choose to spool the files to a sub-directory within /tmp, ensure your system is not configured to remove the contents of /tmp on a reboot. Ideally, you should create a dedicated filesystem to hold the trace data so that filesystems such as /var and

/tmp are not filled, inadvertently causing larger issues. The “dest” attribute is likely the only value you would need to adjust for each stanza.
For clusters that have shared filesystems managed by IBM.StorageRM, add the following stanza to the trace.conf file:

```
IBM.StorageRM Trace:
pat=/var/ct/.*/log/mc/IBM.StorageRM/trace
spooling = ON
pages = 9
dest = /trcspool/
size = 4096000
```

For clusters that have DB2 with HADR, add the following stanza to the trace.conf file:

```
IBM.TestRM Trace:
pat=/var/ct/.*/log/mc/IBM.TestRM/trace
spooling = ON
pages = 4
dest = /trcspool/
size = 4096000
```

To allow for historical trace data for various other RSCT components, consider adding the following stanzas:

```
IBM.ConfigRM Trace.summary:
pat=/var/ct/.*/log/mc/IBM.ConfigRM/trace.summary
spooling = ON
pages = 3
dest = /trcspool/
size = 2048000

IBM.ConfigRM Trace.detail:
pat=/var/ct/.*/log/mc/IBM.ConfigRM/trace.detail
spooling = ON
pages = 3
dest = /trcspool/
size = 2048000

IBM.ConfigRM Trace:
pat=/var/ct/.*/log/mc/IBM.ConfigRM/trace
spooling = ON
pages = 5
dest = /trcspool/
size = 1048576

hags_spool_summary,summary.last:
pat=/var/ct/.*/log/cthags/trace.s*
spooling = ON
pages = 4
dest = /trcspool/
size = 262144

hags_spool_last:
pat=/var/ct/.*/log/cthags/trace.last
spooling = ON
pages = 4
dest = /trcspool/
size = 262144

hags_spool:
pat=/var/ct/.*/log/cthags/
spooling = ON
pages = 9
dest = /trcspool/
size = 1048576

rmcd_spool_nodestat:
pat=/var/ct/.*/log/mc/trace.nodestat
spooling = ON
pages = 4
dest = /trcspool/
size = 131072

rmcd_spool_cmdflow:
pat=/var/ct/.*/log/mc/trace.cmdflow
```

```
spooling = ON
pages = 3
dest = /trcspool/
size = 1048576

rmcd_spool:
pat=/var/ct/.*/log/mc/trace
spooling = ON
pages = 9
dest = /trcspool/
size = 1048576
```

Copy the trace.conf file to /var/ct/cfg/ on all nodes in the domain.

You will need to stop and restart the domain to enable trace spooling or to pick up changes to an existing trace.conf file. If trace spooling has been enabled for IBM.ConfigRM and the core RSCT daemons, then 'stopsrc -g rsct' and 'startsrc -g rsct' should be run after first stopping the domain (before restarting the domain).

Note: You should be cautious if planning to force stop the domain with resources online as this can result in unexpected node reboots. In this scenario, it is recommended to first disable critical resource protection so that RSCT cannot reboot the node.

To disable critical resource protection, (take no action upon loss of quorum) run the following as root:

```
        chrsrc -c IBM.PeerNode CritRsrcProtMethod=5
```

After the domain has been restarted, re-enable critical resource protection:

```
        chrsrc -c IBM.PeerNode CritRsrcProtMethod=3
```

The CritRsrcProtMethod=3 setting is recommended to prevent the potential for truncated log files upon reset. This will aid in post mortem problem determination by logging every possible message to permanent filesystem storage prior to reboot.

Once the cluster/domain is restarted you can verify that spooling is turned on with the following command:

```
        lssrc -ls IBM.RecoveryRM
```

... if spooling is enabled, you will see the filename and destination of the spooled trace files that are configured to be spooled. You can do this for other Resource Managers as well.

File system Space Considerations for Trace Spooling

Trace spooling can use up a lot of filesystem space. On larger clusters or very active clusters, this can happen very quickly, so caution must be exercised when enabling this functionality. The amount of data spooled to your designated filesystem is dependent on a number of variables and very much specific to an individual cluster. For this reason, it is not possible to provide accurate advice as to how much space to allocate to the file system that will store the spooled trace files. Therefore it would be necessary to initially monitor the space allocated verse the space being used for two reasons:

1. To know how much historical data can be retained before the oldest data starts being removed. This will dictate how long a **complete data capture** can be delayed from the point a problem/failure it observed.
2. To accurately setup a cronjob to remove the oldest spool files so as to ensure the filesystem being used for spooling doesn't reach 100% capacity.

Use chkspool to control the space usage within the spool directory with the following cron jobs. Note the --megabyte_limit needs to be set a little below the maximum size of the filesystem (/trcspool) :

```
# Run chkspool twice each hour--clean old files to keep below 800 MB threshold (for 1GB /trcspool)
25,55 * * * * /usr/bin/chkspool --spool_dir /trcspool/ --megabytes_limit 800
```

Result: Saved trace files are deleted (starting with the oldest ones) until no more than 800MB total space is used under /var/opt/perf.

```
# Run chkspool once daily--clean out trace files more than 3 weeks old
20 * * * * /usr/bin/chkspool --spool_dir /trcspool/ --days_limit 21
```

Result: Any trace files older than three weeks (21 days) will be deleted.