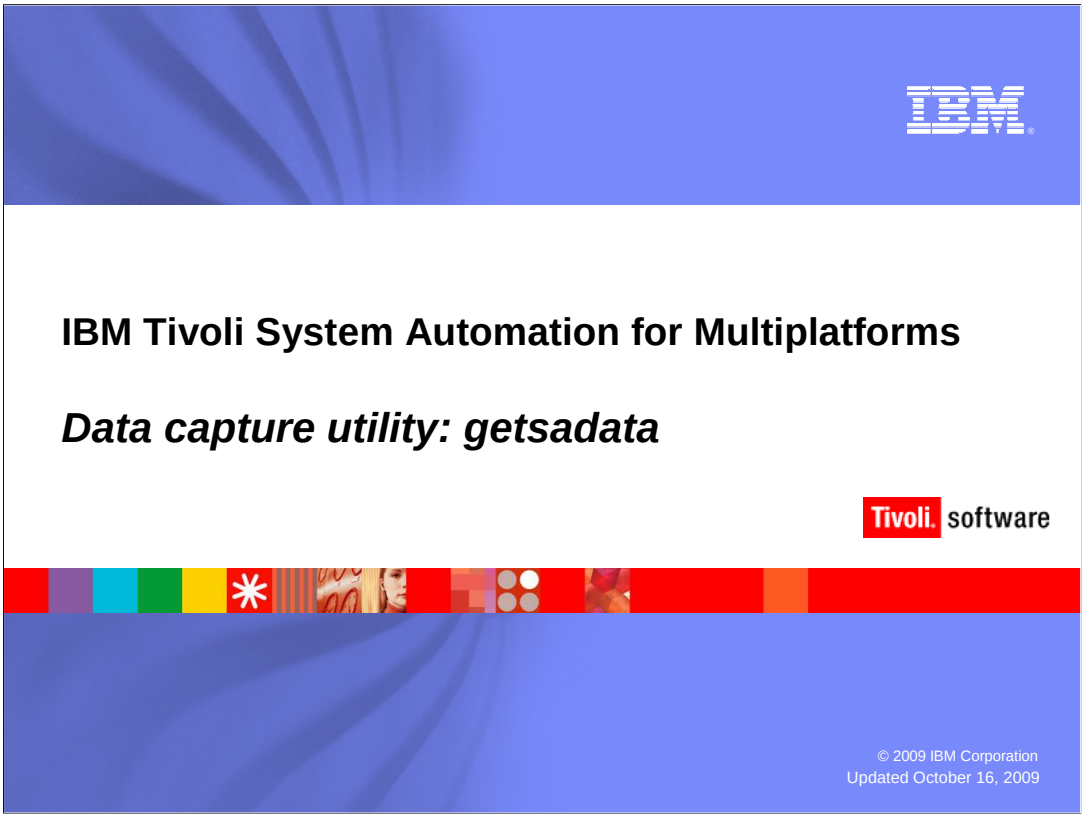


This presentation is focused on a data capture utility called getsadata, used to collect detailed information about a product called “Tivoli System Automation for Multiplatforms”. This product goes by the acronym TSAMP.

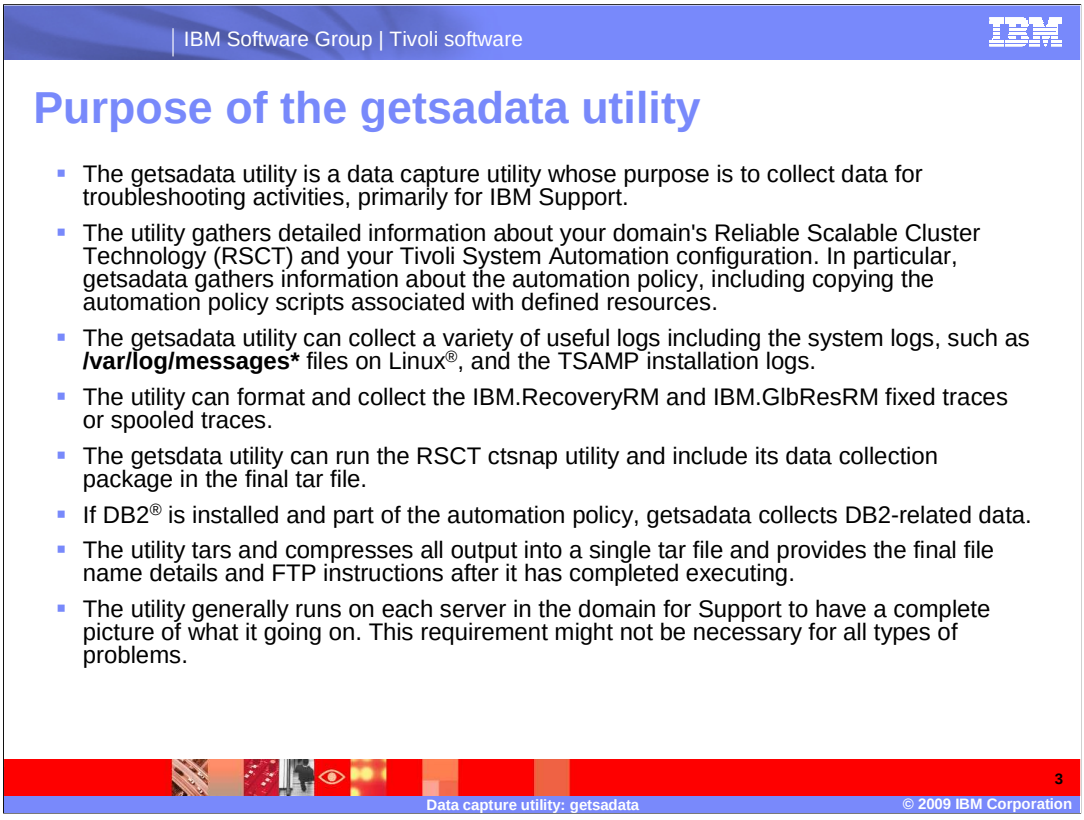
The getsadata utility was written by the TSAMP Support team and continues to be maintained and enhanced by the TSAMP Support team.

Agenda

- Purpose of the getsadata utility
- Initial setup for getsadata
- Usage and help for getsadata
- Options for getsadata explained
- Examples of using getsadata
- Important points to note
- Confirm you have run getsadata on master first
- Manually terminating the getsadata utility
- Options to skip certain collection types
- Results of running getsadata
- What is collected with the -all option

This presentation's agenda is as follows:

Purpose of the getsadata utility
Initial setup for getsadata
Usage and help for getsadata
Options for getsadata explained
Examples of using getsadata
Important points to note
Confirm you have run getsadata on master first
Manually terminating the getsadata utility
Options to skip certain collection types
Results of running getsadata
What is collected with the -all option



getsadata is a Data Capture Utility whose purpose is to collect data for troubleshooting Activities, primarily for IBM Support.

The getsadata utility gathers extensive details about the clustered environment. This can be divided into two main areas:

- the cluster software known as RSCT, and

- the automation software known as TSAMP

Specific to RSCT, getsadata can execute the 'ctsnap' utility and package its output. This is typically what RSCT Support engineers would need if diagnosing a problem thought to be at the cluster level.

As part of the collection specific to TSAMP, the automation policy is dumped into a variety of files and the automation policy scripts associated with defined resources are copied. Additionally getsadata can format and collect the trace files for the IBM.RecoveryRM and IBM.GlbResRM daemons (the two main TSAMP processes).

Generic system information is also captured, including system logs, for example “/var/log/messages*” files for the Linux platform and the AIX® error report.

If DB2 is installed and part of the automation policy, getsadata will attempt to collect DB2 related details as well.

This utility generally needs to be run on each server in the domain in order for Support to have a complete picture of what it going on, though this requirement might not be necessary for all types of problems.

Initial setup for getsadata

- Download the latest version of the getsadata utility using this URL:
  - http://www.ibm.com/support/docview.wss?&uid=swg21285496
- Ensure you always FTP the tar package using binary mode.
- Untar the package on the system where it is to be used, and preferably in its own subdirectory.
- You might need to set the permissions for the included scripts to execute them on your system if permissions of untarred files are automatically restricted in your environment.
  Example :

  chmod 755 getsadata

  chmod 755 ctsnap.level2
- If you cannot execute the scripts even with permissions set correctly, the scripts were probably changed to DOS file format. If so, run dos2unix against each script.
- You must be the root user to execute this data capture utility correctly.

The latest version of the getsadata utility can always be downloaded using the URL on this slide.

To avoid corruption of the packaged files, it should only be untarred on the destination server and preferably within its own directory.

The getsadata package includes the data capture utility for RSCT, called 'ctsnap.level2'. Both scripts are executables, so the additional steps to set permission and change format to be UNIX executable are optional.

Note that this data capture utility needs to be run with root authority.

Usage and help for getsadata

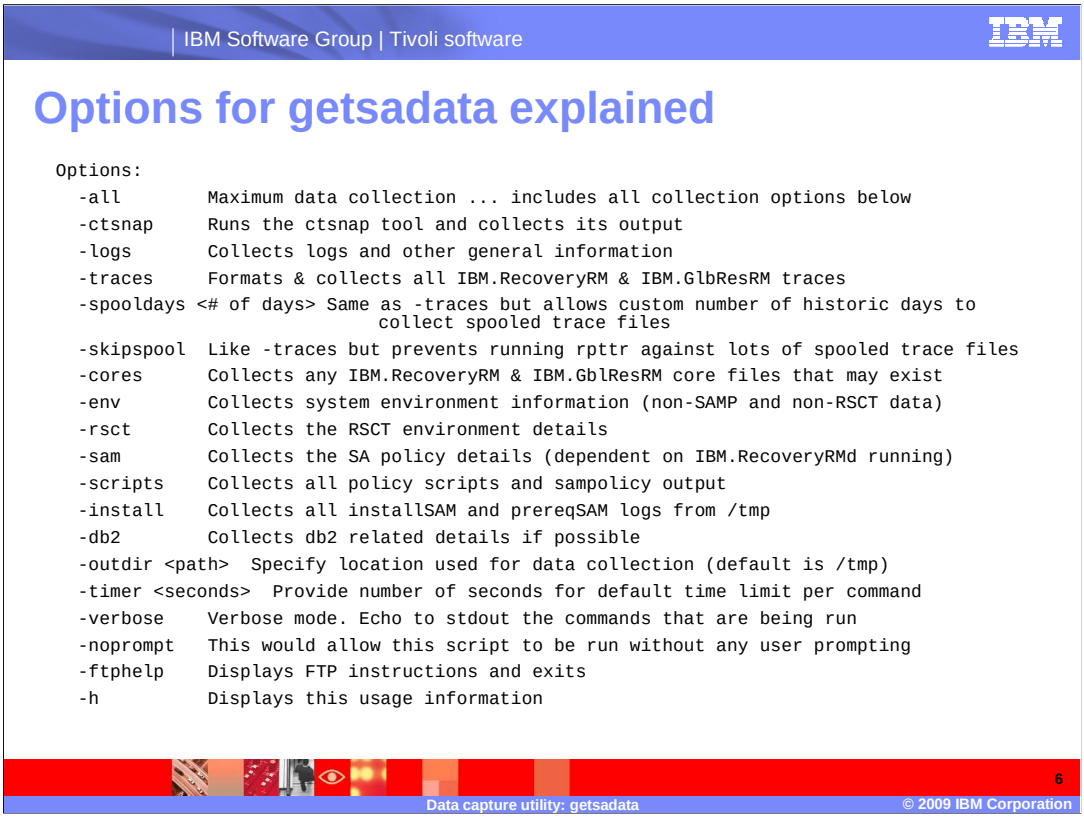
```
# ./getsadata -h

Purpose:
  getsadata - Collects cluster (RSCT) and automation (TSAMP) related
  information that can be used by Support to facilitate troubleshooting.

Usage:
  getsadata [-all] [-ctsnap | -skipctsnap] [-logs | -skiplogs]
    [-traces | -spooldays <days> | -skipspool | -skiptraces]
    [-cores [-skipdebugger] | -skipcores] [-env | -skipenv]
    [-rsct | -skiprsct] [-sam | -skipsam] [-scripts | -skipscripts]
    [-install | -skipinstall] [-db2 | -skipdb2] [-outdir <path>]
    [-timer <seconds>] [-verbose] [-noprompt] [-ftphelp] [-h]

Options:
    ...

Examples:
  1. To capture all data :
       getsadata -all
  2. To capture all data and see detailed progress information :
       getsadata -all -verbose
  3. To capture all but ctsnap :
       getsadata -all -skipctsnap
```


This slide shows the usage information for the getsadata utility, similar to man page format.

The descriptions of each option would typically be included in the usage information but have been left off this slide. They are described in the next slide instead.

The three examples shown at the bottom of the slide are also part of the usage information.

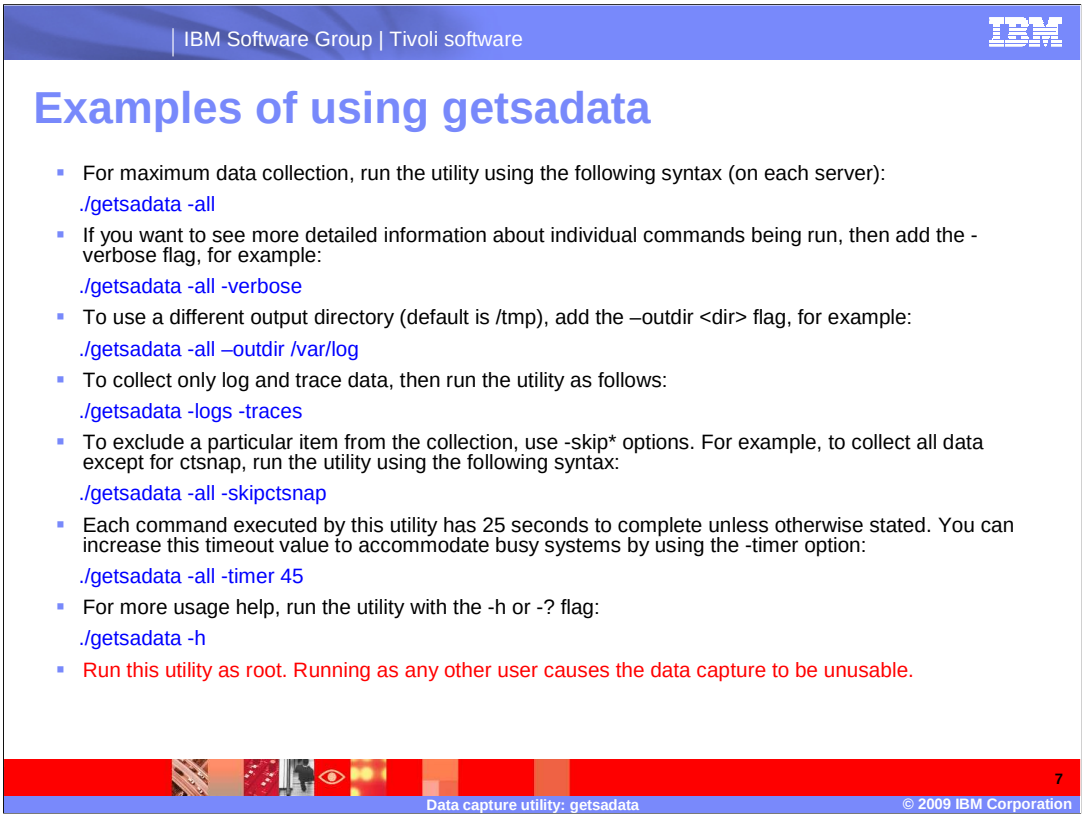
This usage information can be displayed with either the -h or -? flags.



Here's the list of options available for the getsadata utility.

The -all option is all inclusive. It will result in the maximum amount of data captured. The -all option includes :

- ctsnap,
- all logs,
- TSAMP traces,
- TSAMP core files,
- generic system environment information,
- RSCT command output
- TSAMP command output, mainly the automation policy
- the automation scripts
- TSAMP install logs
- DB2 HADR related details, if DB2 is part of the automation policy.



In the majority of cases, the -all option would be the recommended usage.

In order to see the individual commands that are executed, you can add the -verbose option. This option can be useful for environments that contain many resources and thus give the appearance that the getsadata utility has hung on a certain section.

By default, all data collected is stored in a sub-directory within the /tmp directory. The final tar file also ends up in the /tmp directory by default. However the -outdir option can be used to specify an alternative location.

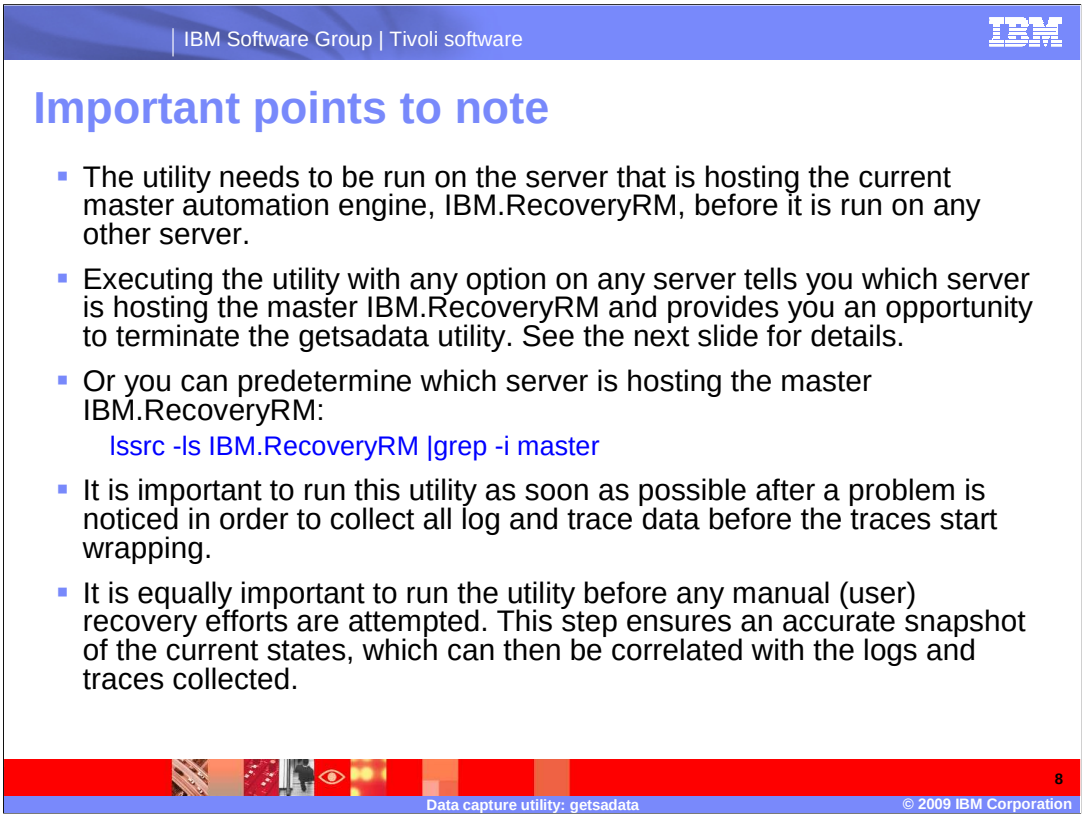
In clustered environments that contain many servers/nodes, for example 10, it might be more efficient to execute getsadata with the -all option on the server currently hosting the “master” IBM.RecoveryRM process, and then limit the data collection to just -logs and -traces on each of the other 9 servers. This will reduce repetitive collection since many of the commands executed are cluster wide, and therefore it would reduce time needed to collect and FTP each of the tar files to IBM Support.

It is possible to exclude individual collection items using the skip options that are listed in a future slide.

By default, most commands executed within getsadata have a 25 second timeout to protect the getsadata utility from hanging. This timer can be adjusted using the -timer option followed by an integer value which approximately represents the number of seconds. It is not recommended that a value lower than 25 be used.

As previously pointed out, getsadata needs to be executed as root user.

Use the -h or -? option for usage information.



In a clustered environment, the automation software (namely IBM.RecoveryRM) runs concurrently on each cluster node. However, only one of the IBM.RecoveryRM daemons is considered the “master” automation engine. The significance of this is that tracing needs to be collected from the server that is currently hosting the "master" automation engine. This means executing getsadata on the current master and doing so before running it on any other server, otherwise you risk losing trace data because of wrapping.

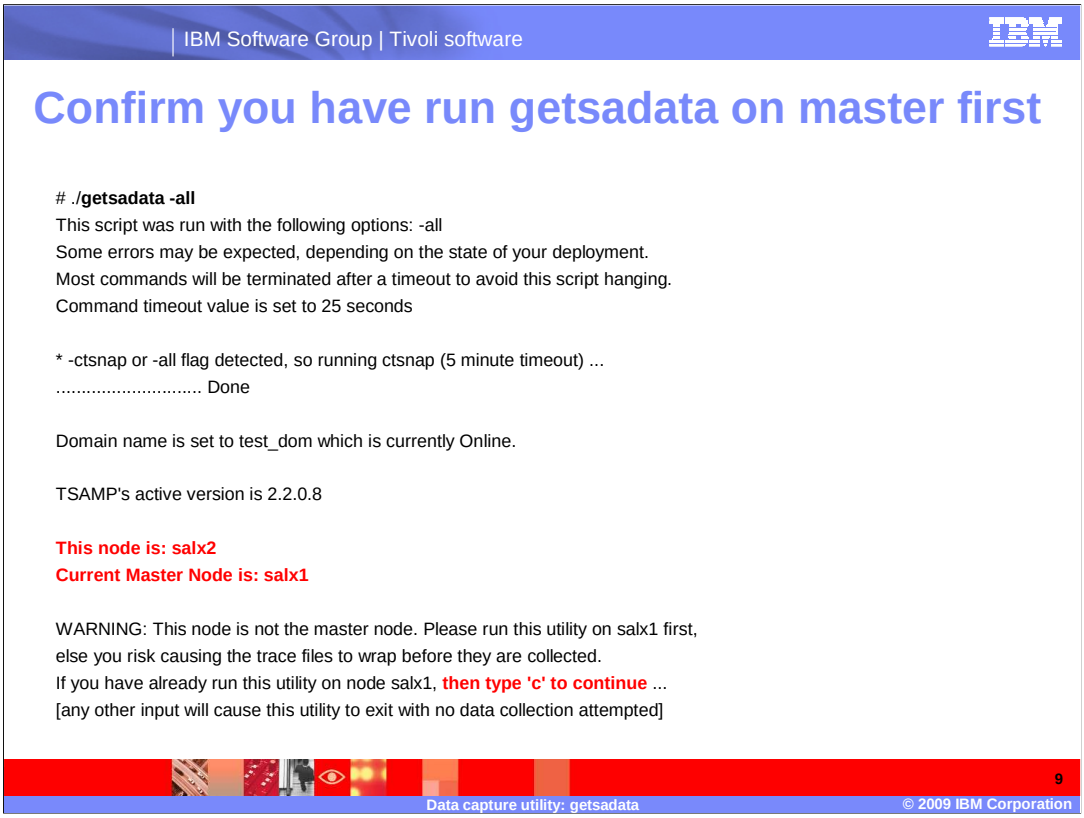
Executing the utility with any option on any server will tell you which server is currently hosting the master IBM.RecoveryRM and will provide you an opportunity to terminate the getsadata utility. An example of this is shown in the next slide.

Alternatively you can predetermine which server is hosting the master IBM.RecoveryRM by using the command:

lssrc -ls IBM.RecoveryRM |grep -i master

It is important to run this utility as soon as possible after a problem is noticed in order to collect all log and trace data before the traces start wrapping.

It is equally important to run the utility before any manual 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. Obviously this is only necessary it you need help from IBM Support.



This slide shows an example of running the getsadata utility on a server that is not currently hosting the “master”.

The utility pauses and waits for the user to enter the letter “c” if the utility should be allowed to continue with data collection. The idea is that you only allow the utility to continue data collection if you have already executed it on the server that is currently hosting the “master” IBM.RecoveryRM process.

Note that the server name that is currently hosting the “master” is listed in the above output.

Manually terminating the getsadata utility

Caught interrupt signal, calling cleanup ... please wait!

****
****
**** Caught interrupt signal, calling cleanup ... please wait ****
****
****

Tarring all output into a file called 081009_165344-salx2-sa_data.tar ...

Compressing the 081009_165344-salx2-sa_data.tar file ...
Filename /tmp/081009_165344-salx2-sa_data.tar.gz has been created.

Checking size of final tar file ...

The final file is smaller than 10 MB, so can be emailed to IBM Support. However FTP is still preferable.
For FTP instructions, run the following 'getsadata -ftphelp'

Let's say the getsadata utility is taking longer than expected to execute or longer than you can wait. So you terminate the execution before the utility completes itself, say with a control-C. In this scenario, the interrupt is caught and a cleanup routine is performed, which will end up tarring whatever data has already been collected.

So you will still end up with a tar file that could potentially be something you would want to provide IBM Support.

Options to skip certain collection types

- Each collection type has an equivalent option that instructs getsadata to skip collecting that data.
- Here are the skip options:

-skipctsnap	=> do not collect ctsnap
-skiptraces	=> do not format/collect traces
-skipcores	=> do not collect core files
-skipdebugger	=> do not run debugger against core files
-skiplogs	=> do not collect logs
-skipenv	=> do not collect environment details
-skiprsct	=> do not collect RSCT details
-skipsam	=> do not collect TSAMP details
-skipscripts	=> do not collect automation scripts
-skipinstall	=> do not collect install logs
-skipdb2	=> do not collect DB2 information

- It would be typical to use one or more of these skip options along with the -all option. For example, to collect all but ctsnap data, run:
  ./getsadata -all -skipctsnap

Here are the options to **exclude** individual collection categories.

You would typically use one or more of these skip options with the -all option. In the example included, a complete collection minus ctsnap is requested.

Results of running getsadata

- After execution, the result will be a single tar file located and named like one of the following:

  /tmp/<MMDDYY_HHMMSS>-<node_name>-sa_data.tar.gz (Linux, Windows®)

  /tmp/<MMDDYY_HHMMSS>-<node_name>-sa_data.tar.Z (AIX, Solaris)

  Example: /tmp/071609_002544-salx2-sa_data.tar.gz

- Run the utility on each server in your domain. At the very least you should run this utility on the server currently hosting the master IBM.RecoveryRM (first) and the problematic servers (if different and known). If you are collecting logs and traces for Support, then you will usually need to run the utility on all servers. Run the script on the master first.
- You should end up with a tar file named similarly to those shown above for each of your servers where you run the getsadata utility. Collect the tar file from each server and create a single tar file named using the PMR number so that it gets routed correctly when you FTP it to IBM :

  cd /tmp

  tar cvf ppppp.bbb.ccc.tar *-sa_data.tar.gz

  where **ppppp.bbb.ccc** is your PMR number.

- Using **Binary Transfer Mode**, anonymously FTP 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 after the data arrives.

Each server where getsadata is executed will end up with a compressed tar file named as shown in this slide.

After running getsadata on each server in the domain, or on the servers requested by IBM Support, move all compressed tar files to one of the servers and tar them together into a single file whose name would be your PMR number. The tar syntax on this slide provides an example of how you would do that after all the getsadata output tar files are located in the same directory.

The FTP instructions on this slide are for sending the final tar file to IBM Support. The getsadata utility can be executed with the -ftphelp option to show the FTP instructions at any time.

What is collected with -all option

- Unpacking the contents of final tar file results in a directory named using the format MMDDYY_HHMMSS-hostname, for example “081009_164520-salx2” .
- The contents of the directory would be similar to the following list of files and directories (directories start with a dot) :

.cores	ctrmc.acls	lsrsrc_TieBreaker.out
.env	ctsnap.salx2.08101645.log	lsrsrc_charm.out
.install_logs	ctsnap.salx2.08101645.tar.gz	lsrsrc_vg-lv-fs.out
.logs	domain_node.out	lssam.out
.policy_scripts	gblresrm_test.out	lssam_trace.out
.resgroups	ls_al.usr-sbin-rsct-bin.out	lssamctrl.out
.samdiags	lscomg.out	lssrc_rsct.out
.traces	lsequ.out	lssrc_samp.out
081009_164520-salx2.log	lsrel.out	publisher_cfg.tar
_7.4.0	lsrg_m.out	recrm_test.out
_id.out	lsrgreq.out	samadapter_cfg.tar
_no-db2info-collected	lsrsrc_Application.out	samlicm.out
_no-netmon.cf-found	lsrsrc_NetworkInterface.out	
_no-samtb_net_blockreserve-files-found	lsrsrc_ServiceIP.out	

- Included in the list of files is a log of getsadata's execution. In the above example, the log is called “081009_164520-salx2.log”
- The file “_7.4.0” indicates the version of the getsadata utility that was used.

This slides provides an example of what each tar file would contain if the -all option was used. Much of the collected data is sorted into individual directories, which are shown starting with a dot.

The messages displayed on standard-out during the execution of getsadata is also logged to a file named with the format MMDDYY_HHMMSS-hostname.log.

There can be a set of underscore files which IBM Support can use to quickly see what version of the getsadata utility was used and what data categories were not collected for some reason. Although not shown in this example, one of the underscore files would let IBM Support know that the data set was collected from the **current** “master” automation engine. This file would be called “_thisNodeIsMaster”. This is important because it is usually the first data set that needs to be examined. The data set contains the trace data for automation activity of the entire domain.

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 e-mail feedback:

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

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

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

Trademarks, copyrights, and disclaimers