Troubleshooting guide

This document describes how to troubleshoot migrator and gives some directions on how to resolve support tickets related to migration.

Overview

To resolve typical migration issues, it is strongly recommended to review the logs in the following order:

1) Check GUI messages and reports, or migrator output when running from CLI.

2) Check info.log.

3) Check debug.log.

4) Check other logs:

- PMM logs
- RPC agent logs
- Mail migration log
- Migration agent logs
- Plesk logs

Attention: never check debug.log before reviewing info.log and GUI messages. Very often that leads to additional time spent: you try to find the details on the issue in debug.log, while you have clear details and instructions how to resolve it in info.log.

Troubleshooting aspects

Failed connections

In case of failed connections, migrator usually writes all required information as GUI messages and to info.log.

Check that:

1. Corresponding service is started and running.
2. The server is listening on the port (use netstat utility).
3. Try to connect to the server by the port with telnet utility.
4. Check that there are no local firewall rules that could block connection.
5. Ask customer to check external firewall rules that could block connection.

Execution of command

Migrator executes different command during migration. For each executed command migrator writes the following information to debug.log:

1. The command that was executed.
2. Where it was executed.
3. Its exit code.
4. Its output (stdout and stderr).

Here is an example from debug.log. That command detects where Plesk is installed on the target server.

```
+|2016-07-07_04:23:34,213|D|MT|core.runners.base|||Execute command on the local server: /bin/grep -m1 -E
'^\s*PRODUCT_ROOT_D' /etc/psa/psa.conf
+|2016-07-07_04:23:34,216|D|MT|core.runners.base|||Command execution results:
=|2016-07-07_04:23:34,216|D|MT|core.runners.base|||stdout: PRODUCT_ROOT_D /usr/local/psa
=|2016-07-07_04:23:34,216|D|MT|core.runners.base|||
=|2016-07-07_04:23:34,216|D|MT|core.runners.base|||stderr:
=|2016-07-07_04:23:34,216|D|MT|core.runners.base|||exit code: 0
```

When some command failed:

1. Understand what command was going to do, what is the impact, and how we could work around the problem.
   For example, if you see that rsync utility fails, and as arguments you see directory with mail messages, obvious impact is that mail messages won't be copied. As a workaround, you could copy the messages manually.
2. Check its output. There could be an error which describes the issue.
   For example, in case of rsync there could be connection issue (then check firewall), or issue when source or target directory does not exist (try to understand why it does not exist, create it if necessary)
3. Try to execute command manually (if it makes sense).
   Note: not all commands could be executed as is, because migrator remove some temporary files due to security reasons (for example, SSH keys) or due to disk usage (for example, database dumps). In that case you could try to modify command so it does not required these files, or create these files yourself (check the previous commands in the log).

SQL queries

During migration, Plesk Migrator executes various SQL queries to get data from source and target panel. When executing query, migrator writes the following information to debug.log:

1. The query itself.
2. Where it was executed (server, database).
3. What was the result.

Here is an example from debug.log. As you may guess, in that query we try to get list of resellers with information about reseller subscription.

```
+|2016-07-07_07:53:57,038|D|MT|plesk.source.plesk.pmm_agent.unix||source|Execute SQL query on panel database:
=|2016-07-07_07:53:57,038|D|MT|plesk.source.plesk.pmm_agent.unix||source|          SELECT Subscriptions.id as
subscription_id, clients.id, login, pname
=|2016-07-07_07:53:57,038|D|MT|plesk.source.plesk.pmm_agent.unix||source|            FROM clients
=|2016-07-07_07:53:57,038|D|MT|plesk.source.plesk.pmm_agent.unix||source|            JOIN Subscriptions ON
=|2016-07-07_07:53:57,038|D|MT|plesk.source.plesk.pmm_agent.unix||source|              Subscriptions.object_id =
clients.id
=|2016-07-07_07:53:57,038|D|MT|plesk.source.plesk.pmm_agent.unix||source|              AND
Subscriptions.object_type = 'client'
=|2016-07-07_07:53:57,038|D|MT|plesk.source.plesk.pmm_agent.unix||source|              AND clients.type = 'reseller'
=|2016-07-07_07:53:57,038|D|MT|plesk.source.plesk.pmm_agent.unix||source|
+|2016-07-07_07:53:57,038|D|MT|core.runners.base||source|Execute command on the source server 'source'
(10.52.143.116): cd /tmp/pmm_agent; perl ExecutePleskSQL.pl
+|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|Command execution results:
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|stdout: {
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|   "status" : "ok",
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|   "rowset" : [
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|      {
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|         "pname" : "George Petrov",
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|         "id" : "2",
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|         "subscription_id" : "3",
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|         "login" : "gpetrov"
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|      }
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|   ]
=|2016-07-07_07:53:57,258|D|MT|core.runners.base||source|}
```

It makes sense to check migrator logs for SQL queries it executes when you don't have some object in the migration list or you don't have some object migrated.

Execute the query manually on the specified server and try to understand why it does not return required data: often that is caused by database inconsistency (broken references).

API requests

Migrator executes requests to Plesk API to get data and create some objects (for example, service plans). When migrator executes API request, it logs the following information to debug.log:

1. API endpoint, to which migrator connected to execute the request.
2. Request XML.
3. Response XML.

Here is an example of query, where, as you may guess, we get information about reseller:

```
+|2016-07-07_07:53:57,986|D|MT|core.utils.common.http_xml_client|||API request to
https://10.0.2.15:8443/enterprise/control/agent.php:
=|2016-07-07_07:53:57,986|D|MT|core.utils.common.http_xml_client|||<?xml version='1.0' encoding='utf-8'?>
=|2016-07-07_07:53:57,986|D|MT|core.utils.common.http_xml_client|||<packet
version="1.6.6.0"><reseller><get><filter><login>gpetrov</login></filter><dataset><gen-info
/></dataset></get></reseller></packet>
+|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||API response from
https://10.0.2.15:8443/enterprise/control/agent.php:
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||<packet version="1.6.6.0">
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||  <reseller>
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||    <get>
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||      <result>
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||        <status>ok</status>
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||        <filter-id>gpetrov</filter-id>
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||        <id>14</id>
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||        <data>
...
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||        </data>
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||      </result>
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||    </get>
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||  </reseller>
=|2016-07-07_07:53:58,054|D|MT|core.utils.common.http_xml_client|||</packet>
```

Sometimes migrator fails to execute API query or parse the results. There could be different reasons:

1. Issues with connection to Plesk API:
    a. Plesk does not listen on specified IP address or port.
    b. There is a firewall blocking connections to API.
2. Issues with the request. For example, migrator may create malformed request.
3. Issues with the response, for example
    a. Plesk completely failed to execute request and returned 500 error.
       In that case, troubleshoot and repair Plesk as usual: check panel log, search for the error in KB, and so on.
    b. Plesk executed request, but it contains an error.
       For example, this may happen if migrator requested information about the object that does not actually exist.
    c. Plesk executed request, but the response is malformed (for example, bad XML).
       In that case, troubleshoot and repair Plesk as usual.
    d. Migrator expected data that does not exist in the response.

How to execute API request manually

With curl

On Unix you could execute the query with the curl utility:

1. Create a file with request XML, for example:

```
# cat request.xml
<?xml version='1.0' encoding='utf-8'?>
<packet version="1.6.6.0"><reseller><get><filter><login>gpetrov</login></filter><dataset><gen-info
/></dataset></get></reseller></packet>
```

2. Obtain Plesk panel password. On Linux server:

```
# /usr/local/psa/bin/admin --show-password
setup
```

3. Execute the query, replacing %PASSWORD% and %SERVER_IP% with actual data:

```
curl -k -H 'Content-Type: text/xml' -H 'HTTP_AUTH_LOGIN: admin' -H 'HTTP_AUTH_PASSWD: %PASSWORD%'
https://%SERVER_IP%:8443/enterprise/control/agent.php -d @request.xml
```

With Plesk API Explorer

You could use the following tool when troubleshooting our local servers:

http://paex.pp.plesk.ru/

Migrator stack traces

When migrator fails to perform some action, usually an exception is thrown in migrator's code. Each exception contains stack trace which is the key point for developers to understand which code failed and why it failed.

Here is an example of real stack trace, caused by inconsistency in database and inability of migrator to handle that inconsistency:

```
+|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||Exception:
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||Traceback (most recent call last):
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||  File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/core/workflow/runner/by_subscription.py
", line 119,
in _run_common_action_plain
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||    run()
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||  File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/core/workflow/runner/by_subscription.py
", line 110,
in run
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||    action.run(self._context)
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||  File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/plesk/source/plesk/actions/fetch/fetch_b
ackup.py", l
ine 20, in run
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||    self._fetch_dump(global_context,
local_runner, source_id)
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||  File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/plesk/source/plesk/actions/fetch/fetch_b
ackup.py", l
ine 50, in _fetch_dump
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||
cls._create_dump(global_context.dump_agent, dump_filename, selection)
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||  File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/plesk/source/plesk/actions/fetch/fetch_c
apability_in
fo.py", line 23, in _create_dump
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||
agent.create_capability_dump(dump_filename, selection=selection)
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||  File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/core/utils/pmm/agent.py", line 165, in
create_capabi
```

lity_dump
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription||| self._run_capability(filename,
self.capability_dump_log, selection)
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription||| File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/plesk/source/plesk/pmm_agent/unix.py",
line 124, in
_run_capability
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||
CapabilityXMLConverter(capability_model).write_xml(local_data_filename)
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription||| File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/plesk/source/plesk/capability_dump/xml
_converter.py"
, line 18, in write_xml
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription||| capability_dump_contents =
xml_to_string_pretty(self.create_xml())
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription||| File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/plesk/source/plesk/capability_dump/xml
_converter.py"
, line 35, in create_xml
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||] + [
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription||| File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/plesk/source/plesk/capability_dump/xml
_converter.py"
, line 69, in _create_client_node
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||] if domains else [])
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription||| File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/plesk/source/plesk/capability_dump/xml
_converter.py"
, line 83, in _create_domain_node
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||] + [
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription||| File
"/usr/local/psa/admin/plib/modules/panel-migrator/backend/lib/python/parallels/plesk/source/plesk/capability_dump/mod
el/plesk.py",

```
line 160, in get_domain_ips
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||    for plesk_ip_pool_item in plesk_ip_pool:
=|2016-07-06_10:48:54,164|D|MT|core.workflow.runner.by_subscription|||TypeError: 'NoneType' object is not iterable
```

If you want developers to investigate the issue, it would be good to collect stack traces for failed operations. But consider the following "features" of migrator:

1. Often, the last stack trace is not the useful one. If you are not sure which stack trace to provide, provide several last stack traces.
2. Often, the stack trace in the log is not caused by an error, but is a normal situation. So, when you see stack trace, don't panic immediately: check that there is a message at ERROR (|E| in debug.log) or WARNING (|W| in debug.log) level nearby.

Migration dumps

It could be useful to understand the data flow of migrator and how migrator stores data in migration dumps when you have one the following issues:

1. Some information was not migrated.
   For example, migrator have not created some addon domain.
2. Some information was restored incorrectly.
   For example, migrator has restored incorrect IP address of a DNS record.
3. Migrator fails to restore information, because it is invalid.
   For example, migrator has failed to restore settings of system user because its login is not correct.

Migrator divides all migrated data into 2 type:

1. Configuration data.
   By configuration data we mean information about business and hosting objects, for example, list of subscriptions, domains, mailboxes with their settings, resellers, and so on.
2. Content.
   By content we mean customer's data: application files and other files, database content (tables, rows, triggers, etc), mail messages.

Here we are going to speak about configuration data: it is stored in migration dumps and takes a long way from the source panel to the target panel. Here is a schema briefly describing data flow:

```
source panel database => shallow dump (*.shallow.xml)
source panel database => raw dump => converted dump => imported dump => per-domain info.xml => target panel database
```

Here is description of each kind of data:

Type of data	Description	Location
Source panel database	Source panel database	For Plesk for Linux, run "plesk db" (12.5 and above) or "mysql -u admin -p`cat /etc/psa/.psa.shadow` psa" (below 12.5) to get into the database.
Shallow dump	Contains information about top-level object: resellers, customers, subscriptions and plans, and is used to create migration list. In some cases, for example when migrating from Confixx or cPanel, there is no shallow dump, and migrator uses raw dump to create migration list.	In session directory of migrator (*.converted.xml)
Raw dump	Raw dumps are create by migration agent or migrator itself and contain all data about objects in the source panel as-is	In session directory of migrator (*.raw.tar, *.raw.zip)
Converted dump	Converted dumps are created by migrator from raw dumps and contain information as we are going to restore it on the target panel. For example, DNS records have replaced source IP addresses with target IP addresses.	In session directory of migrator (*.converted.tar, *.converted.zip)

Imported dump	Imported dump is created by PMM from converted dump. It is a set of XML files, per-domain, which contains data as PMM is going to restore it.	/var/lib/psa/dumps
Per-domain info.xml	XML file describing single subscription, that is used to create or update objects by deployer (part of PMM).	/usr/local/psa/PMM/rsessions/%SESSION_ID%/dump/info.xml

So key point to troubleshoot data issues, is to trace how it goes from the source panel to the target one:

1. Check if data is correct in the source panel database.
   If not - most probably there is inconsistency in the source panel, or such data is valid for the source panel but is not valid for the target panel.
   Example: invalid symbols in system user login.
2. Check if data is correct in raw dump.
   Raw dump is created by special dump agents (in most cases) or migrator itself. If data is correct in the source panel database, but is not correct in raw dump - it's time to troubleshoot dump agent.
   Example: absent addon domain, bug in agent.
3. Check if data is correct in converted dump.
   If data gets incorrect here, it's time to troubleshoot migrator itself.
   Example: IP address in DNS record was not replaced, bug in migrator.
4. Check if data is correct in imported dump.
   Very rare, but if data gets incorrect here, it's time to troubleshoot PMM.
5. Check if data is correct in per-domain info.xml file.
   If data gets incorrect here, it's time to troubleshoot PMM.

PMM restoration

When resolving issues with restoration of configuration data of subscriptions, it could be useful to check PMM logs and data files. You could find the in the following way with the help of debug.log:

```
+|2016-07-07_07:56:07,247|D|ST2|core.utils.restore_hosting_utils|myapps.tld||Use these files when investigating issues happened while restoring hosting settings of domain 'myapps.tld':
=|2016-07-07_07:56:07,247|D|ST2|core.utils.restore_hosting_utils|myapps.tld||- Session directory of restoration: /usr/local/psa/PMM/rsessions/20160707075537978
=|2016-07-07_07:56:07,247|D|ST2|core.utils.restore_hosting_utils|myapps.tld||- Restore hosting settings log: /usr/local/psa/PMM/rsessions/20160707075537978/migration.log
=|2016-07-07_07:56:07,247|D|ST2|core.utils.restore_hosting_utils|myapps.tld||- Restore hosting settings XML: /usr/local/psa/PMM/rsessions/20160707075537978/dump/info.xml
```

As you may guess from the log messages, the most useful files are:

1. migration.log, also known as PMM deployer log. It contains commands that were executed by PMM deployer to restore settings of subscription. Here is example where deployer calls Plesk utilities to create database:

```
[2016-07-07 07:55:45.138|10736] INFO: Deployer action: create
[2016-07-07 07:55:45.138|10736] INFO: => mywordpress1 Restore database
[2016-07-07 07:55:45.138|10736] INFO: Request is ready for:
PMM_PROFILE_LOG=/usr/local/psa/PMM/rsessions/20160707075537978/profile.log PLESK_RESTORE_MODE=1
PLESK_DISABLE_PROVISIONING= LANG=en_US.UTF-8 ALLOW_WEAK_PASSWORDS=
/usr/local/psa/admin/plib/api-cli/database.php '--create' 'mywordpress1' '-domain' 'myapps.tld' '-type' 'mysql' '-print-id'
'-skip-webspace-checking' '-charset' 'utf8' '-collation' 'utf8_general_ci' '-server' 'localhost:3306'
'-ignore-nonexistent-options'
[2016-07-07 07:55:45.599|10736] INFO: HTTP status code is: 200
[2016-07-07 07:55:45.599|10736] INFO: Request is ready for: echo '<?xml version="1.0"?>
<database-info><database name="mywordpress1" owner-guid="89d1320e-9548-46e7-be84-68c92f7161a1"
charset="utf8" version="5.5.41" collation="utf8_general_ci" guid="273f868a-f12c-476d-bf97-5e99ad93290e_db_1"
type="mysql" id="1" deployer-action="create">
      <db-server type="mysql">
        <host>localhost</host>
        <port>3306</port>
      </db-server>
      <dbuser default="true" id="1" name="mywordpress1" deployer-action="create">
        <password
type="sym">$AES-128-CBC$MS4NS4uKYTguUPvXakT4Aw==$qRr8DuHT0dOUx/4/JgAx2A==</password>
        <acl>
          <allowed-host>%</allowed-host>
        </acl>
      </dbuser>
    </database></database-info>
' | PMM_PROFILE_LOG=/usr/local/psa/PMM/rsessions/20160707075537978/profile.log PLESK_RESTORE_MODE=1
PLESK_DISABLE_PROVISIONING= LANG=en_US.UTF-8 ALLOW_WEAK_PASSWORDS=
/usr/local/psa/admin/bin/backup_restore_helper '--restore-database' 'myapps.tld' '-ignore-nonexistent-options'
[2016-07-07 07:55:45.599|10736] INFO: HTTP status code is: 100
[2016-07-07 07:55:45.794|10736] INFO: HTTP status code is: 200
```

2. info.xml. It contains the data that PMM deployer tried to restore. For example, DNS records are stored in the following way in that file:

```
<dnsrec opt="" src="www.myown.tld." dst="myown.tld." type="CNAME"/>
<dnsrec opt="" src="ftp.myown.tld." dst="myown.tld." status="syn" type="CNAME"/>
<dnsrec opt="10" src="myown.tld." dst="mail.myown.tld." status="syn" type="MX"/>
<dnsrec opt="" src="myown.tld." dst="ns.myown.tld." status="syn" type="NS"/>
<dnsrec opt="24" src="10.50.2.80" dst="myown.tld." type="PTR"/>
<dnsrec opt="" src="myown.tld." dst="v=spf1 +a +mx -all +a:a10-52-143-116.qa.plesk.ru" type="TXT"/>
<dnsrec opt="" src="ipv4.myown.tld." dst="10.50.2.80" type="A"/>
<dnsrec opt="" src="mail.myown.tld." dst="10.50.2.80" type="A"/>
<dnsrec opt="" src="webmail.myown.tld." dst="10.50.2.80" type="A"/>
<dnsrec opt="" src="myown.tld." dst="10.50.2.80" type="A"/>
<dnsrec opt="" src="ns.myown.tld." dst="10.50.2.80" type="A"/>
```

RPC agent

RPC agent is a program that runs on source servers and executes commands from Plesk Migrator which runs on target. It is used on Windows only, on Linux we use SSH to execute remote commands.

RPC agent logs are available on source server by the following paths:

```
C:\panel_migrator\rpc-agent\info.log
C:\panel_migrator\rpc-agent\debug.log
```

In the logs you can find which commands were executed on the source server, and some other information. Example of the debug log:

```
[2016-07-05 18:21:21,263] [DEBUG] parallels | Execute command: "C:\Program Files
(x86)\Parallels\Plesk\admin\bin\websrvmng" --get-mime-types --vhost-name=mailtest.tld --vdir-name=
test
[2016-07-05 18:21:21,513] [DEBUG] parallels | Command stdout:
[2016-07-05 18:21:21,513] [DEBUG] parallels | Command stderr:
[2016-07-05 18:21:21,513] [DEBUG] parallels | Command exit code: 0
[2016-07-05 18:21:22,887] [DEBUG] parallels | Put contents to file 'C:\panel_migrator\libmariadb.dll'
[2016-07-05 18:21:22,980] [DEBUG] parallels | Put contents to file 'C:\panel_migrator\MailMigrator.exe'
```

Here we see that migrator tried to get MIME types of a domain, and then deploy MailMigrator.exe with necessary libraries.

Check the log for messages at ERROR level and for stack traces.

Dump agent

When dump agent fails, creates incorrect dump or puts wrong data in the dump, it makes sense to check it logs.

On Unix they are stored at:

```
%SESSION_DIR%/pmm-agent.%DATE%/*.log
```

The most interesting is configuration-dump.log, that is log of raw dump creation. For example there you could see which SQL queries are used by the agent to retrieve information about databases of a specific domain:

```
[18600]: 2016-07-07 13:54:29.175 DEBUG Dumping domain databases
[18600]: 2016-07-07 13:54:29.175 TRACE SQL: SELECT id, type, db_server_id, name FROM data_bases WHERE
dom_id=?: Params: 18
[18600]: 2016-07-07 13:54:29.175 DEBUG Dumping database users belonging to any database
[18600]: 2016-07-07 13:54:29.175 TRACE SQL: SELECT dbu.id, dbu.login, a.type, a.password, dbs.host, dbs.port,
dbs.type AS dbservertype  FROM db_users dbu, accounts a, DatabaseServers dbs WHERE dbu.account_id = a.id AND
dbu.db_id = 0 AND dbu.dom_id = ? AND dbu.db_server_id = dbs.id: Params: 18
```

Less interesting is capability-dump.log - that is log of creation special dump necessary for capability checks.

Mail migration

On Linux, mail messages are copied with rsync, which is executed just as usual command. So most probably you need refer to "Execution of command" section to troubleshoot it.

On Windows, everything is bit more complex: there is a special tool called MailMigrator.exe used to copy mail messages.

By default, logging is disabled for that utility due to performance reasons. To enable logging, you need to edit configuration file of migrator (config.ini) and add the following options to the section which corresponds to the source server.

To enable logging of the utility on the source server:

```
mail-source-log-enabled: true
```

Once you enabled the option, You will get log of messages back up at:

```
%SESSION_DIR%/mail/<domain-name>/<mail-name>/mail-backup.log
```

To enable logging of the utility on the target server:

```
mail-target-log-enabled: true
```

You will get log of restoration at:

```
%SESSION_DIR%/mail/<domain-name>/<mail-name>/mail-restore.log
```

and log of list mail messages at:

```
%SESSION_DIR%/mail/<domain-name>/<mail-name>/mail-list-messages.log
```

For more details, refer to Mail migration article.

MSSQL databases

One of the issues with MSSQL databases is when migrator fails to connect to MSSQL server. This may have different reasons:

1. Firewall blocks connection.
2. MSSQL server is not listening on IP address or port to which migrator tries to connect to.
3. MSSQL browsing service is stopped.

In most cases, you need to obtain MSSQL Management Studio (or any other client) and make it able to connect to the MSSQL server before running migrator again. Search how to troubleshoot MSSQL connection issues in Google - it is quite complex topic with many pitfalls.

Hanged migrator

Important notice before debugging hanged migration is that migrator is working in multi-threading mode. Multi-threading mode is needed to improve performance of migration: for example, when one thread is creating addon domains of one subscription, the other one copies mail messages of the other subscription. But because of multi-threading mode, log messages from different threads are mixed, and the last line of the log is not always what you are looking for.

1. When it seems that migrator hangs, the first thing you should do is to understand what it is doing (what was the last thing it was doing). There could be 2 situations:

- Migrator is executed in a single thread: the last line of the log contains "|MT|" string. In that case you just need to review the last lines of the log.
- Migrator is executed in multi-threading mode: the last line of the log contains "|STx|" string, where "x" is some number, which means number of the thread.
  In that case you should review the last lines of the log for each thread (from "ST1" to "ST5").

Also, to quickly understand what migrator is doing right now check "progress" file in session directory of migrator:

```
Subscription migration status:
+--------------+-------------+-------------------------------------------------------------+
| Subscription |    Status   |                          Action                             |
+--------------+-------------+-------------------------------------------------------------+
|   sub1.tld   | In Progress |           Synchronize subscription with plan                |
+--------------+-------------+-------------------------------------------------------------+
|  myapps.tld  | In Progress | Adjust common migrated applications for new environment |
+--------------+-------------+-------------------------------------------------------------+
|  myown.tld   | In Progress |         Restore hosting settings of subscription            |
+--------------+-------------+-------------------------------------------------------------+
```

When you have migrator hanged for the only subscription, that subscription will be the only with status "In Progress". Then you may check debug log for that subscription.

2. After you reviewed the log, you know which commands are executed right now.

- If it is some content copy command, you may check if data is actually transferred right now: for example for mail copy command check if number of mail messages increases, for copy web content command, check if amount of used disk space on the target server increases, and so on.
- If data is not actually copied:
  - For advanced users: try to debug the utility, for example on Unix you may use strace.
  - Kill the process.
    Note: In many cases migrator will try to restart the command once more (to avoid reliability issues).

Tips & tricks

Run migrator from CLI

When you or customer started migrator from GUI, and you want to continue execution from CLI, check log for lines containing "MIGRATOR START" or "MIGRATOR END". For example:

```
+|2016-07-07_07:58:57,998|D|MT|core.workflow.runner.base|||MIGRATOR END:
/usr/local/psa/admin/sbin/modules//panel-migrator/plesk-migrator transfer-accounts
/usr/local/psa/var/modules/panel-migrator/sessions/20160707075349/config.ini --skip-services-checks
--skip-infrastructure-checks --skip-main-node-disk-space-checks --skip-license-checks --skip-capability-checks
--ignore-migration-list-errors --migration-list-format=json
--migration-list-file=/usr/local/psa/var/modules/panel-migrator/sessions/20160707075349/migration-list.json
```

You could just copy-paste the command, and it will be executed practically in the same way if you run it from GUI.

Note: migrator will ask to set passwords which are encrypted in config.ini file for security reasons. Just follow the instruction.

Run copy content commands from CLI

There are 3 commands that migrator has which could be useful when debugging copy content issues:

- copy-web-content
- copy-mail-content
- copy-db-content

For example, to run mail content transfer when investigating some issues with mail messages, run (replace %SESSION_ID% with actual session ID):

```
/usr/local/psa/admin/sbin/modules/panel-migrator/plesk-migrator copy-mail-content
/usr/local/psa/var/modules/panel-migrator/sessions/%SESSION_ID%/config.ini
```

View debug log

Use the following command to filter all records related to specific subscription:

```
grep '|subscription.tld|' panel-migrator.debug.log
```

Also filter by specific thread:

```
grep '|ST3|' panel-migrator.debug.log
```

Or by log level: only messages at info level

```
grep '|I|' panel-migrator.debug.log
```

Also, use cut Linux command to get rid of unnecessary fields.

Refer to Logging article for more details. Additionally we have HTML migration log, which is an alternative to using grep, especially on Windows where you don't have grep by default.

Unpack and pack dumps

There are 2 commands that could simplify view or modify migrator dumps.

If you don't want to use archive tool, or don't want to remember its options, use unpack-backups command:

```
# /usr/local/psa/admin/sbin/modules/panel-migrator/plesk-migrator unpack-backups
/usr/local/psa/var/modules/panel-migrator/sessions/20160707075349/config.ini
...
[2016-07-07 11:50:30][INFO] Extracting files from backup
'/usr/local/psa/var/modules/panel-migrator/sessions/20160707075349/plesk.backup.source.converted.tar' to
'/usr/local/psa/var/modules/panel-migrator/sessions/20160707075349/unpacked/plesk.backup.source.converted.tar'
[2016-07-07 11:50:30][INFO] Extracting files from backup
'/usr/local/psa/var/modules/panel-migrator/sessions/20160707075349/plesk.backup.source.raw.tar' to
'/usr/local/psa/var/modules/panel-migrator/sessions/20160707075349/unpacked/plesk.backup.source.raw.tar'
```

Use pack-backups command when you modified unpacked raw dump files and want migrator to use data from them:

```
# /usr/local/psa/admin/sbin/modules/panel-migrator/plesk-migrator pack-backups
/usr/local/psa/var/modules/panel-migrator/sessions/20160707075349/config.ini
```

Post-migration checks

Once you resolved the issue, it makes sense to run post-migration checks to make sure that everything works well. For example, if web files or databases were not transferred during migration, post-migration checks will detect that.

Here is an example:

```
# /usr/local/psa/admin/sbin/modules/panel-migrator/plesk-migrator test-all
/usr/local/psa/var/modules/panel-migrator/sessions/20160707075349/config.ini
...
Transferred Domains' Functional Issues
 |
 └ Source server 'source'
   |
   └ Subscription 'myapps.tld'
     |
     ├ Mail domain 'myapps.tld'
     |
     ├ Database service 'myapps.tld'
     |   |
     |   ├ error: MySQL database 'mywordpress1' on the source server and on the target server has a differing set of
tables. The databases that are present on the source server, but are missing at the target server: 'wp_posts',
'wp_term_taxonomy', 'wp_postmeta', 'wp_users', 'wp_terms', 'wp_links', 'wp_comments', 'wp_options', 'wp_usermeta',
'wp_commentmeta', 'wp_term_relationships', 'wp_termmeta'
     |   |  Make sure that the database content was migrated successfully.
     |   |  If it was not, run the migration of the whole subscription once more, or copy the database content manually.
...
```

Here you could see that database of domain was not transferred correctly and you need to continue investigations.

Logs and files locations

Reference table (replace words surrounded by % with corresponding data)

Path	Server	Linux	Windows
Per-session info.log	Target	/usr/local/psa/var/modules/panel-migrator/sessions/%SESSION_ID%/info.log	%PLESK_DIR%\var\modules\panel-migrator\sessions\%SESSION_ID%\info.log
Per-session debug.log	Target	/usr/local/psa/var/modules/panel-migrator/sessions/%SESSION_ID%/debug.log	%PLESK_DIR%\var\modules\panel-migrator\sessions\%SESSION_ID%\debug.log
Global info.log (used rarely)	Target	/usr/local/psa/var/modules/panel-migrator/logs/info.log	%PLESK_DIR%\var\modules\panel-migrator\logs\info.log
Global debug.log (used rarely)	Target	/usr/local/psa/var/modules/panel-migrator/logs/debug.log	%PLESK_DIR%\var\modules\panel-migrator\logs\debug.log
PMM restoration directory	Target	/usr/local/psa/PMM/rsessions/%PMM_SESSION_ID%	%PLESK_DIR%\PMM\rsessions\%PMM_SESSION_ID%
PMM deployer log	Target	/usr/local/psa/PMM/rsessions/%PMM_SESSION_ID%/migration.log	%PLESK_DIR%\PMM\rsessions\%PMM_SESSION_ID%\migration.log
PMM per-domain info.xml	Target	/usr/local/psa/PMM/rsessions/%PMM_SESSION_ID%/dump/info.xml	%PLESK_DIR%\PMM\rsessions\%PMM_SESSION_ID%\dump\info.xml
Windows RPC agent info log	Source	N/A	C:\panel_migrator\rpc-agent\info.log
Windows RPC agent debug log	Source	N/A	C:\panel_migrator\rpc-agent\debug.log

Windows copy mail logs - backup logs	Source	N/A	C:\panel_migrator\mail\%DOMAIN_NAME%\%MAIL_NAME%\mail-restore.log
Windows copy mail logs - restore logs	Target	N/A	%PLESK_DIR%\var\modules\panel-migrator\sessions\%SESSION_ID%\mail\%DOMAIN_NAME%\%MAIL_NAME%\mail-restore.log
Windows copy mail logs - get message IDS	Target	N/A	%PLESK_DIR%\var\modules\panel-migrator\sessions\%SESSION_ID%\mail\%DOMAIN_NAME%\%MAIL_NAME%\mail-list-messages.log
Unix migration agent logs	Target	/usr/local/psa/var/modules/panel-migrator/sessions/%SESSION_ID%/pmm-agent.%DATE%/*.log	N/A
Shallow dump	Target	/usr/local/psa/var/modules/panel-migrator/sessions/%SESSION_ID%/*.shallow.xml	%PLESK_DIR%\var\modules\panel-migrator\sessions\%SESSION_ID%*.shallow.xml
Raw dump	Target	/usr/local/psa/var/modules/panel-migrator/sessions/%SESSION_ID%/*.raw.tar	%PLESK_DIR%\var\modules\panel-migrator\sessions\%SESSION_ID%*.raw.zip
Converted dump	Target	/usr/local/psa/var/modules/panel-migrator/sessions/%SESSION_ID%/*.converted.tar	%PLESK_DIR%\var\modules\panel-migrator\sessions\%SESSION_ID%*.converted.zip