SMB Witness Service

In Samba CTDB Clusters

Stefan Metzmacher <metze@samba.org>

Samba Team / SerNet

2024-04-18

https://samba.org/~metze/presentations/2024/SambaXP/
Topics

- What is the Service Witness Protocol [MS-SWN]
- Examples how it works
- rpcd_witness design
- Some strange things a Windows client is doing.
- How to configure rpcd_witness
- net witness commands
- Questions? Feedback!
What is the Service Witness Protocol [MS-SWN]

The Service Witness Protocol [MS-SWN]:
- Provides a way to notify SMB3 clients about cluster failures
- Either network interface or node failures
- Or planned downtimes or loadbalancing by administrators

The protocol itself is independent of SMB3:
- It is based on DCERPC over TCP (ncacn_ip_tcp)
- It uses kerberos or NTLMSSP integrity protection
The Service Witness Protocol [MS-SWN]:
- Provides a way to notify SMB3 clients about cluster failures
- Either network interface or node failures
- Or planned downtimes or loadbalancing by administrators

The protocol itself is independent of SMB3:
- It is based on DCERPC over TCP (ncacn_ip_tcp)
- It uses kerberos or NTLMSSP integrity protection
Basic flow of a client connecting with witness

<table>
<thead>
<tr>
<th>Time</th>
<th>IP Address 1</th>
<th>IP Address 2</th>
<th>Service</th>
<th>Request Description</th>
<th>Client</th>
<th>Node0</th>
<th>Node1</th>
<th>Node2</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:27:47</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>SMB</td>
<td>Session Setup Request</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:27:47</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>SMB</td>
<td>Session Setup Response</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:27:47</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>SMB</td>
<td>Tree Connect Request Tree: \ubcluster.w2022-l7.base\IPC$</td>
<td>Client:</td>
<td>Node0:</td>
<td>Node1:</td>
<td>Node2:</td>
</tr>
<tr>
<td>12:27:47</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>SMB</td>
<td>Tree Connect Request Tree: \ubcluster.w2022-l7.base\shm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:27:51</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>SMB</td>
<td>Tree Connect Response</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:27:51</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>GetInterfaceList request</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:27:51</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>RegisterEx response</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:27:51</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify request</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:29:51</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify request</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:35:51</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify response, Error: WERR_TIMEOUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:35:51</td>
<td>172.31.9.118</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify response, Error: WERR_TIMEOUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<= continuous availability, scaleout, cluster
### Resource-Unavailable Flow

<table>
<thead>
<tr>
<th>Time</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Type</th>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>18:08:33</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>SMB2 Session Setup Request</td>
<td></td>
</tr>
<tr>
<td>18:08:33</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>SMB2 Session Setup Response</td>
<td></td>
</tr>
<tr>
<td>18:08:33</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>SMB2 Tree Connect Request Tree: \ubcluster.w2022-l7.base\shm</td>
<td></td>
</tr>
<tr>
<td>18:08:33</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>SMB2 Tree Connect Response</td>
<td></td>
</tr>
<tr>
<td>18:08:34</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>SMB2 Tree Connect Request Tree: \ubcluster.w2022-l7.base\IPC</td>
<td></td>
</tr>
<tr>
<td>18:08:34</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>SMB2 Tree Connect Response</td>
<td></td>
</tr>
<tr>
<td>18:08:36</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>WITNESS GetInterfaceList request</td>
<td></td>
</tr>
<tr>
<td>18:08:36</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>WITNESS RegisterEx response</td>
<td></td>
</tr>
<tr>
<td>18:08:38</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>WITNESS AsyncNotify request</td>
<td></td>
</tr>
<tr>
<td>18:10:49</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>WITNESS AsyncNotify request</td>
<td></td>
</tr>
<tr>
<td>18:10:49</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>WITNESS Session Setup Request</td>
<td></td>
</tr>
<tr>
<td>18:10:49</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>WITNESS Session Setup Response</td>
<td></td>
</tr>
<tr>
<td>18:10:49</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>WITNESS Tree Connect Request Tree: \ubcluster.w2022-l7.base\shm</td>
<td></td>
</tr>
<tr>
<td>18:10:49</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>WITNESS Tree Connect Response</td>
<td></td>
</tr>
<tr>
<td>18:10:51</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>UnRegister request</td>
<td></td>
</tr>
<tr>
<td>18:10:51</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>UnRegister response, Error: WERR_NOT_FOUND</td>
<td></td>
</tr>
<tr>
<td>18:10:51</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>GetInterfaceList request</td>
<td></td>
</tr>
<tr>
<td>18:10:51</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>RegisterEx response</td>
<td></td>
</tr>
<tr>
<td>18:10:51</td>
<td>172.31.9.118</td>
<td>172.31.99.167</td>
<td>WITNESS AsyncNotify request</td>
<td></td>
</tr>
</tbody>
</table>

---

**Client:** 172.31.9.118  
**Node0:** 172.31.99.166  
**Node1:** 172.31.99.167  
**Node2:** 172.31.99.168
## Client-Move flow

<table>
<thead>
<tr>
<th>Client: 172.31.9.118</th>
<th>172.31.9.118</th>
<th>SMB2</th>
<th>Negotiate Protocol Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:44:36,717268</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Negotiate Protocol Response</td>
</tr>
<tr>
<td>15:44:36,723718</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Session Setup Request</td>
</tr>
<tr>
<td>15:44:36,724414</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Session Setup Response</td>
</tr>
<tr>
<td>15:44:36,731287</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Tree Connect Request Tree: \ubcluster.w2022-l7.base\shm</td>
</tr>
<tr>
<td>15:44:36,731763</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Tree Connect Response</td>
</tr>
<tr>
<td>15:44:36,732881</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Tree Connect Request Tree: \ubcluster.w2022-l7.base\IPC$</td>
</tr>
<tr>
<td>15:44:37,739894</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Tree Connect Response</td>
</tr>
<tr>
<td>15:44:37,741150</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Tree Connect Response</td>
</tr>
<tr>
<td>15:44:41,745394</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>GetInterfaceList request</td>
</tr>
<tr>
<td>15:44:41,853592</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>RegisterEx request NetName[ubcluster.w2022-l7.base] IpAddress[172.31.99.167]</td>
</tr>
<tr>
<td>15:44:41,855292</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>RegisterEx response</td>
</tr>
<tr>
<td>15:44:41,863502</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify request</td>
</tr>
<tr>
<td>15:46:41,868076</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify response, Error: WERR_TIMEOUT</td>
</tr>
<tr>
<td>15:46:41,869075</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify request</td>
</tr>
<tr>
<td>15:48:41,971270</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify request</td>
</tr>
<tr>
<td>15:50:26,175499</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify request</td>
</tr>
<tr>
<td>15:50:26,176791</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Negotiate Protocol Request</td>
</tr>
<tr>
<td>15:50:26,176807</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Negotiate Protocol Response</td>
</tr>
<tr>
<td>15:50:26,186724</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Session Setup Request</td>
</tr>
<tr>
<td>15:50:26,194004</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Session Setup Response</td>
</tr>
<tr>
<td>15:50:26,194490</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Tree Connect Request Tree: \ubcluster.w2022-l7.base\shm</td>
</tr>
<tr>
<td>15:50:26,196587</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Tree Connect Response</td>
</tr>
<tr>
<td>15:50:29,196623</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Tree Connect Request Tree: \ubcluster.w2022-l7.base\IPC$</td>
</tr>
<tr>
<td>15:50:29,198861</td>
<td>172.31.9.118</td>
<td>SMB2</td>
<td>Tree Connect Response</td>
</tr>
<tr>
<td>15:50:33,203320</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify response, Error: WERR_NOT_FOUND</td>
</tr>
<tr>
<td>15:50:33,204027</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>UnRegister request</td>
</tr>
<tr>
<td>15:50:33,204604</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>UnRegister response, Error: WERR_NOT_FOUND</td>
</tr>
<tr>
<td>15:50:33,306338</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>GetInterfaceList request</td>
</tr>
<tr>
<td>15:50:33,319486</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>RegisterEx request NetName[ubcluster.w2022-l7.base] IpAddress[172.31.99.167]</td>
</tr>
<tr>
<td>15:50:33,319983</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>RegisterEx response</td>
</tr>
<tr>
<td>15:50:33,325602</td>
<td>172.31.9.118</td>
<td>WITNESS</td>
<td>AsyncNotify request</td>
</tr>
</tbody>
</table>

**Client:** 172.31.9.118

**Node0:** 172.31.99.166

**Node1:** 172.31.99.167

**Node2:** 172.31.99.168

*Hack to trigger a re-registration*
We had some source3/rpc_server rewrites in the last years
  - The merge to dcesrv_core.c by Samuel Cabrero
  - The samba-dcerpcd infrastructure by Volker Lendecke

We can now have isolated service binaries
  - /usr/libexec/samba.rpcd_
  - With ’rpc start on demand helpers = no’ we support ncacn_ip_tcp

Simple async responses are possible
  - If we do not care about user impersonation
rpcd_witness design (Part 1)

- We had some source3/rpc_server rewrites in the last years
  - The merge to dcesrv_core.c by Samuel Cabrero
  - The samba-dcerpcd infrastructure by Volker Lendecke

- We can now have isolated service binaries
  - /usr/libexec/samba/rpcd_
  - With ’rpc start on demand helpers = no’ we support ncacn_ip_tcp

- Simple async responses are possible
  - If we do not care about user impersonation
We had some source3/rpc_server rewrites in the last years
  ▶ The merge to dcesrv_core.c by Samuel Cabrero
  ▶ The samba-dcerpcd infrastructure by Volker Lendecke

We can now have isolated service binaries
  ▶ /usr/libexec/samba/rpcd_
  ▶ With ’rpc start on demand helpers = no’ we support ncacn_ip_tcp

Simple async responses are possible
  ▶ If we do not care about user impersonation
We had some witness service prototypes implemented in the past
- By Gregor Beck/Stefan Metzmacher
- By Günther Deschner/Jose A. Rivera
- By David Disseldorp/Samuel Cabrero

The interaction with ctdbd is important
- But it was missing in 2 prototypes
- And 1 prototype tried to implement too much in ctdbd itself

Finally I came up with a very simple ctdbd change
- It was trivial to add CTDB_SRVID_IPREALLOCATED notifications to ctdbd

Each rrpcd_witness instance just needs this:
- Load all addresses of the whole cluster at start
- Wait for CTDB_SRVID_IPREALLOCATED to be posted
- Reload all addresses of the whole cluster
- Compare the changes in the list in order to notice changes
We had some witness service prototypes implemented in the past
- By Gregor Beck/Stefan Metzmacher
- By Günther Deschner/Jose A. Rivera
- By David Disseldorp/Samuel Cabrero

The interaction with ctdbd is important
- But it was missing in 2 prototypes
- And 1 prototype tried to implement too much in ctdbd itself

Finally I came up with a very simple ctdbd change
- It was trivial to add CTDB_SRVID_IPREALLOCATED notifications to ctdbd

Each rpd_witness instance just needs this:
- Load all addresses of the whole cluster at start
- Wait for CTDB_SRVID_IPREALLOCATED to be posted
- Reload all addresses of the whole cluster
- Compare the changes in the list in order to notice changes
We had some witness service prototypes implemented in the past
- By Gregor Beck/Stefan Metzmacher
- By Günther Deschner/Jose A. Rivera
- By David Disseldorp/Samuel Cabrero

The interaction with ctdbd is important
- But it was missing in 2 prototypes
- And 1 prototype tried to implement too much in ctdbd itself

Finally I came up with a very simple ctdbd change
- It was trivial to add CTDB_SRVID_IPREALLOCATED notifications to ctdbd

Each rpcd_witness instance just needs this:
- Load all addresses of the whole cluster at start
- Wait for CTDB_SRVID_IPREALLOCATED to be posted
- Reload all addresses of the whole cluster
- Compare the changes in the list in order to notice changes
We had some witness service prototypes implemented in the past
- By Gregor Beck/Stefan Metzmacher
- By Günther Deschner/Jose A. Rivera
- By David Disseldorp/Samuel Cabrero

The interaction with ctdbd is important
- But it was missing in 2 prototypes
- And 1 prototype tried to implement too much in ctdbd itself

Finally I came up with a very simple ctdbd change
- It was trivial to add CTDB_SRVID_IPREALLOCATED notifications to ctdbd

Each rpcd_witness instance just needs this:
- Load all addresses of the whole cluster at start
- Wait for CTDB_SRVID_IPREALLOCATED to be posted
- Reload all addresses of the whole cluster
- Compare the changes in the list in order to notice changes
rpdc_witness design (Part 3)

- rpdc_witness needs support for ncacn_ip_tcp
  - So it requires 'rpc start on demand helpers = no'
  - We also register each connection with ctdbd to get tickle-acks

- Each Register[Ex]() results in a global registration
  - They are stored in rpdc_witness_registration.tdb
  - With the registration context/policy handle as key
  - And the server_id (node+pid) also in the content

- This allows 'net witness’ commands to work
  - List registrations
  - Send specific administrative actions to the individual registrations
  - See later slides for more details and examples
rpcd_witness design (Part 3)

- rpcd_witness needs support for ncacn_ip_tcp
  - So it requires 'rpc start on demand helpers = no'
  - We also register each connection with cttdbd to get tickle-acks

- Each Register[Ex]() results in a global registration
  - They are stored in rpcd_witness_registration.tdb
  - With the registration context/policy handle as key
  - And the server_id (node+pid) also in the content

- This allows 'net witness' commands to work
  - List registrations
  - Send specific administrative actions to the individual registrations
  - See later slides for more details and examples
rpcd_witness design (Part 3)

- rpcd_witness needs support for ncacn_ip_tcp
  - So it requires ’rpc start on demand helpers = no’
  - We also register each connection with ctdbd to get tickle-acks

- Each Register[Ex]() results in a global registration
  - They are stored in rpcd_witness_registration.tdb
  - With the registration context/policy handle as key
  - And the server_id (node+pid) also in the content

- This allows ’net witness’ commands to work
  - List registrations
  - Send specific administrative actions to the individual registrations
  - See later slides for more details and examples
Windows clients behave in strange ways (Part 1)

- The SMB2 Tree Connect response has flags for cluster capabilities:
  - SMB2\SHARE\CAP\CONTINUOUS\AVAILABLE
  - SMB2\SHARE\CAP\SCALEOUT
  - SMB2\SHARE\CAP\CLUSTER
  - SMB2\SHARE\CAP\ASYMMETRIC

- SMB2\SHARE\CAP\CLUSTER:
  - This is the indication the [MS-SWN] service runs on the server
  - And the client should make use of it when using the connected share
  - Sadly only effective together with
    SMB2\SHARE\CAP\CONTINUOUS\AVAILABLE

- SMB2\SHARE\CAP\SCALEOUT:
  - Means the cluster can have more than one active node at a time
The SMB2 Tree Connect response has flags for cluster capabilities:

- SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY
- SMB2_SHARE_CAP_SCALEOUT
- SMB2_SHARE_CAP_CLUSTER
- SMB2_SHARE_CAPASYMMETRIC

SMB2_SHARE_CAP_CLUSTER:
- This is the indication the [MS-SWN] service runs on the server
- And the client should make use of it when using the connected share
- Sadly only effective together with SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY

SMB2_SHARE_CAP_SCALEOUT:
- Means the cluster can have more than one active node at a time
Windows clients behave in strange ways (Part 1)

- The SMB2 Tree Connect response has flags for cluster capabilities:
  - SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY
  - SMB2_SHARE_CAP_SCALEOUT
  - SMB2_SHARE_CAP_CLUSTER
  - SMB2_SHARE_CAPASYMMETRIC

- SMB2_SHARE_CAP_CLUSTER:
  - This is the indication the [MS-SWN] service runs on the server
  - And the client should make use of it when using the connected share
  - Sadly only effective together with
    SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY

- SMB2_SHARE_CAP_SCALEOUT:
  - Means the cluster can have more than one active node at a time
Windows clients behave in strange ways (Part 2)

- **SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY:**
  - This indicates that the share is always available.
  - The client should try to reconnect (maybe to other nodes) fast.
  - Windows clients also use this as trigger to request persistent handles.
  - Even if the server does not provide `SMB2_CAP_PERSISTENT_HANDLES`.
  - Each open generates a warning in the client event log.

- **SMB2_SHARE_CAPASYMMETRIC:**
  - This is used to indicate that a share is attached to a disk owner.
  - Other nodes act as proxy.
  - It means the client uses separate set of connections for the share.
  - The client might connect to a different cluster node.
  - And provides a share name for `RegisterEx()`.
Windows clients behave in strange ways (Part 2)

- **SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY:**
  - This indicates that the share is always available
  - The client should try to reconnect (maybe to other nodes) fast
  - Windows clients also use this as trigger to request persistent handles
  - Even if the server does not provide SMB2_CAP_PERSISTENT_HANDLES
  - Each open generates a warning in the client event log

- **SMB2_SHARE_CAP_SYMMETRIC:**
  - This is used to indicate that a share is attached to a disk owner
  - Other nodes act as proxy.
  - It means the client uses separate set of connections for the share
  - The client might connect to a different cluster node
  - And provides a share name for RegisterEx()
Windows clients behave in strange ways (Part 3)

▶ After a AsyncNotify response there is no re-registration
  ▶ A Windows client reacts on a RESOURCE_CHANGE, CLIENT_MOVE, SHARE_MOVE.
  ▶ It reconnects the SMB3 connection if required
  ▶ But it does not call Register[Ex]() for the new connection

▶ We use a trick in order to force a re-registration
  ▶ 5 seconds after a RESOURCE_CHANGE, CLIENT_MOVE, SHARE_MOVE.
  ▶ we return AsyncNotify with STATUS_NOT_FOUND
  ▶ This triggers a re-registration
Windows clients behave in strange ways (Part 3)

- After a AsyncNotify response there is no re-registration
  - A Windows client reacts on a RESOURCE_CHANGE, CLIENT_MOVE, SHARE_MOVE.
  - It reconnects the SMB3 connection if required
  - But it does not call RegisterEx() for the new connection

- We use a trick in order to force a re-registration
  - 5 seconds after a RESOURCE_CHANGE, CLIENT_MOVE, SHARE_MOVE.
  - we return AsyncNotify with STATUS_NOT_FOUND
  - This triggers a re-registration
Basic smb.conf options for rpcd_witness

net conf list output:

```
[global]
  netbios name = ubcluster
  idmap config * : backend = autorid
  idmap config * : range = 1000000-1999999
  security = ADS
  workgroup = W2022-L7
  realm = W2022-L7.BASE
  rpc start on demand helpers = no
  smb3 share cap:continuous availability = yes

[shm]
  path = /dev/shm
  read only = no
```
There is a 47.samba-dcerpcd script for ctdbd

- `ctdb event script enable legacy 47.samba-dcerpcd`
- This tries to start the samba-dcerpd (systemd service)
- This is needed for `rpc start on demand helpers = no`
net witness commands

- **net witness list**
  - List witness registrations from rpcd_witness_registration.tdb

- **net witness client-move**
  - Generate client move notifications for witness registrations to a new ip or node

- **net witness share-move**
  - Generate share move notifications for witness registrations to a new ip or node

- **net witness force-unregister**
  - Force unregistrations for witness registrations

- **net witness force-response**
  - Force an AsyncNotify response based on json input (mostly for testing)
net witness commands

- net witness list
  - List witness registrations from rpcd_witness_registration.tdb
- net witness client-move
  - Generate client move notifications for witness registrations to a new ip or node
- net witness share-move
  - Generate share move notifications for witness registrations to a new ip or node
- net witness force-unregister
  - Force unregistrations for witness registrations
- net witness force-response
  - Force an AsyncNotify response based on json input (mostly for testing)
net witness commands

▶ net witness list
  ▶ List witness registrations from rpcd_witness_registration.tdb
▶ net witness client-move
  ▶ Generate client move notifications for witness registrations to a new ip or node
▶ net witness share-move
  ▶ Generate share move notifications for witness registrations to a new ip or node
▶ net witness force-unregister
  ▶ Force unregistrations for witness registrations
▶ net witness force-response
  ▶ Force an AsyncNotify response based on json input (mostly for testing)
net witness commands

- net witness list
  - List witness registrations from rpcd_witness_registration.tdb
- net witness client-move
  - Generate client move notifications for witness registrations to a new ip or node
- net witness share-move
  - Generate share move notifications for witness registrations to a new ip or node
- net witness force-unregister
  - Force unregistrations for witness registrations
- net witness force-response
  - Force an AsyncNotify response based on json input (mostly for testing)
net witness commands

- net witness list
  - List witness registrations from rpcd_witness_registration.tdb
- net witness client-move
  - Generate client move notifications for witness registrations to a new ip or node
- net witness share-move
  - Generate share move notifications for witness registrations to a new ip or node
- net witness force-unregister
  - Force unregistrations for witness registrations
- net witness force-response
  - Force an AsyncNotify response based on json input (mostly for testing)
net witness list example

```bash
root@ub1704-166:--# net witness list
Registration-UUID:   NetName                             ShareName         IpAddress          ClientComputerName
--------------------:--------------------------------------:-----------------:---------------:--------------------------
c10b4d0b-758a-4918-b1fa-37916c4465c  ubcluster.w2022-l7.base  ' '                172.31.99.167    w2022-118.w2022-l7.base

root@ub1704-166:--# net witness list --json --witness-regISTRATION=c10b4d0b-758a-4918-b1fa-37916c4465c | jq 'registrations'

```

```json
{
  "c10b4d0b-758a-4918-b1fa-37916c4465c": {
    "version": "0x00020000",
    "net_name": "ubcluster.w2022-l7.base",
    "share_name": null,
    "ip_address": "172.31.99.167",
    "client_computer_name": "w2022-118.w2022-l7.base",
    "flags": {
      "WITNESS_REGISTER_IP_NOTIFICATION": false,
      "int": 0,
      "hex": "0x00000000"
    },
    "timeout": 120,
    "context_handle": {
      "handle_type": 1,
      "uuid": "c10b4d0b-758a-4918-b1fa-37916c4465c"
    },
    "server_id": {
      "pid": 25488,
      "task_id": 0,
      "vmm": 0,
      "unique_id": 177883242780630300
    },
    "auth": {
      "account_name": "W2022-118$",
      "domain_name": "W2022-L7",
      "account_sid": "S-1-5-21-133451344-1126667713-3548050118-1000"
    },
    "connection": {
      "local_address": "ipv4:172.31.99.166:49154",
      "remote_address": "ipv4:172.31.9.118:64996"
    },
    "registration_time": "2024-04-15T14:23:51.526021+0200"
  }
}
```
net witness client-move examples

Example 1: with given registration id

```bash
root@ub1704-166:--# net witness client-move --witness-registration=c10b4d0b-758a-4918-b1fa-3791e6c4465c --witness-new-node=0
CLIENT_MOVE_TO_NODE: 0
Registration-UUID: NetName ShareName IpAddress ClientComputerName
------------------:----------:--------:-----------:------------------
c10b4d0b-758a-4918-b1fa-3791e6c4465c ubcluster.w2022-l7.base '' 172.31.99.167 w2022-118.w2022-l7.base

Example 2: apply to all

```bash
root@ub1704-166:--# net witness list
Registration-UUID: NetName ShareName IpAddress ClientComputerName
------------------:----------:--------:-----------:------------------
b217fc5d-a0c7-44a9-90f9-13228365bd21 ubcluster.w2022-l7.base '' 172.31.99.167 w2022-118.w2022-l7.base
```

Stefan Metzmacher

SMB Witness Service

(17/19)
Samba 4.20.0 and Windows clients

- Samba 4.20.0 contains all changes
- We should hope that Windows clients get a fix
  - So that SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY without SMB2_CAP_PERSISTENT_HANDLES does not flood the clients event log
Samba 4.20.0 and Windows clients

- Samba 4.20.0 contains all changes
- We should hope that Windows clients get a fix
  - So that `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY` without `SMB2_CAP_PERSISTENT_HANDLES` does not flood the clients event log
Questions? Feedback!

- Stefan Metzmacher, metze@samba.org
- https://www.sernet.com
- https://samba.plus

Slides: https://samba.org/~metze/presentations/2024/SambaXP/