SDM 12.9 Advanced Availability

Document purpose

The purpose of this document is to collect and group information about the new SDM 12.9 AA setup.
The informations were collected on a bootcamp held in August 2014 in Ismaning/Germany.
Any comments are welcome. Please feel free to add any valuable information and let us know, if you find this document helpfull, or what you are missing.

Motivation for an AA setup

Two basic benefits:

- Advanced (higher) availablity
  - A better availablilty can be achieved by introducing additional redundant infrastructure components.
  - User sessions are higher available because of more than one application server.
  - Singleton processes, those who can't be redundant, are highly available by a so called "hot standby" server.
- Rolling Maintenance
  - Reducing maintencane downtime by implementing changes server by server, with minimum impact for the user.

AA: what it does and what it doesn't

AA is not HA

- Certain components are not improved by AA in terms of availability
  - The HA of the database has to be handled separately.
  - The HA of File Repositories have to be handled separately.
  - A 3rd Party Load Balancer must be implented and made HA.
- The so called singleton processes are not redundant.
- The monitoring and failover has to be initialized by seperate HA means. Special hardware or other CA or 3rd party software products (i.e. CA Spectrum) maybe used for this purpose.
- The system must be shutdown completely, if
  - an mdb patch needs to be installed.
  - OS patches needs be installed.
  - Security patches needs to be installed.
  - ...
- High Avalability is also achievable by other technologies like virtualization, and maybe the better choice in simple setups.

Minimum and possible HW setups

At least four (4) servers are required for the simplest SDM AA environment:

- 1 Background Server
- 1 Standby Server (Hot Standby)
- 2 Application Server

Additional HW is required for remote DB (local DB does't make sense anymore), Loadbalancer and File sharing infrastructure for Knowlegde index files and attachmants. This additional HW has to be high available also.

There can be any number of standby servers (chain standby) and application server (redundancy and scalability).

Migration from Conventional to AA

Depending on different scenarios, there are different migration strategies available.
In general, there is no direct migration path from a conventional setup to a AA setup. While in a conventional setup, a distributed environment was primarily based on scalability, an AA setup is primarily used for higher availablity. Of course, an AA setup can also be used for scalability. But you have to invest in HW without improvement of scalability or performance.

Single box pre 12.9 setup

- At least three additional servers are needed.
- A pre 12.9 primary server will be migrated to 12.9 as usual.
- setup all other servers as described in the following chapter.

Distributed primary/secondary pre 12.9 setup

Plan the architecture first!

A simple migration procedure would be:

- migrate the primary server first.
- Configure (pdm_configure) the server as the background server.
- Start this server and add all standby and application server to the Server list, assigning the corresponding roles.
- After creating the server entries you are able to migrate your pre 12.9 pdm_startup procedure to the new distributed environment by executing the "pdm_edit_migration.bat|sh" script. This will create approporiate server confguration settings in the DB.
- You may create those configurations for each server manually or modify the migrated ones (adding(/ removing domsrvrs/webegines, etc.).
- After this initial setup, install and configure all other servers.
- A migration of a secondary server is not realy neccessary. They could be installed as a fresh SDM 12.9 server. (remove old installation first)

Administration

- All servers has to be manually created as a Server record (Administration->System->Servers)
- Configuration can be used to add additional processes like domsrvrs, webengines, webdirectors and others for each server. (Administration->System->Configurations)
- There could be more than one configuartion for each server to reflect different setups.
- These Configurations can be selected when configuring (pdm_configure) a server.
- startup files, web.cfg files and other config files are generated automatically based on the selected configuration during the pdm_configure.

AA scenarios

Rolling maintencance process

To implements changes like patches, customizations or configurations there is a prefered procedure to follow:

- Stop standby server (pdm_halt).

- Perform the changes on the standby server.
- Suppress version control between standby and background server
  - pdm_server_control –v (on standby server)
- Start standby server (pdm_init / net start ....)
- Promote standby server to background server
  - pdm_server_control -b (standby server)
  - switches the server roles. Previous background server becomes standby server and vice versa.
- Stop the new standby server and implement your changes if applicable. Many changes maybe deployed by version control when restarting the standby server.
- Start new standby server
- You may promote the current standby server back to the background server, but there is no need to do so.
- Do changes as needed on all other standby server.
- Do changes as need on all application server.
  - Choose application server with lowest user session count.
  - Initiate quiesce of this app server (delayed shutdown)
    - pdm_server_control –q interval (–s server_name)
      - interval : seconds to wait before stopping the service
    - User gets notified inside WEB-GUI nad has a chance to finnish his work and may reconnect to a different app server.
  - After the app server has stopped, do the appropriate changes.
  - And start the app server again
  - Do the last three steps for all other app servers

Failover

A failover can be initiated to switch the server role from a standby server to the background server. The SDM services of the previous background server will be shutdown when initiating failover!
This is different from a rolling maintenance scenario, where both server role gets exchanged. Both server will be active after a "pdm_server_control -b", while a failover will stop the provious background server.

In general there are two functionalities available for a failover automation:

- Monitoring critical processes on a server
- Initiating the failover in certain error conditions

The monitoring of critical processes on server can be used to

- check the health status of the background server (used to decide, if failover is needed)
- check the availability of an application server (used by loadbalancer to dispatch new sessions)

These functionalities are provided by a separate Servlet implementation, called HealthServlet. This servlet gets delivered as a war package in $NX_ROOT/samples/HealthServlet/HealthServlet.war. CA recommends to install this servler in a separate tomcat installation to be independent on any SDM relevant processes.

To install the servlet, just copy the war package to the "webapps" directory of your tomcat installation. Typically the HealthServlet will be installed on all servers.

Monitoring

- The monitoring servlet is accessible by http://host:port/HealthServlet/GetHealth
- The Servlet will respond with an HTTP Status of 200 for a healthy system. Additionaly it will return some information about the server role.
  - EV-Server-Status: All OK!

    EV-Server-Role: BG
- If critical processes are not running, or the server is in quiesce mode, the servlet will respond with an HTTP status of 503.
  - EV-Server-Status: NOT OK!

    EV-Server-Role: BG
- Additional information maybe provided in the response.
- It may be necessary to enhance the default configuration of the HealthServlet to include additional critical processes. Take a look at the configuration file : <tomcat-dir>/webapps/HealthServlet/WEB-INF/classes/health.xml

Initiating failover

- To initiate the failover, send a simple GET request to the failover servlet by using the following URL:

  https://<standby_server>:<port>/HealthServlet/FailoverServlet
- The failover will promote the standby server to the new background server and stop the old background server.
- The failover servlet should be secured by https and access restriction using available authentication mechanism, like basic authenticaion.

Facts

- pdm_edit.pl is gone. Startup definitions are configured through the WEB-GUI as Server Configurations.
- A Web-Director can only dispatch sessions to webengines to its own server.
- A Background Server Login(WEB-GUI) is only allowed fo users with access type having an Access-Level of “Admin”.
- pdm_webstat only show local information. There is no central information source for all running processes on all servers, or all sessions on all servers. Each server has its own slump_nxd running.
- Repository directories should be on HA file server/share.
- After a failover, the previous standby server becomes the new background server and cannot be distinguished from the original background server. It now is a full functional background server and can remain in this state.
- There are two different scenarios, depending on failover and rolling maintenance situation:
  - A failover is triggered externaly through the HealthServlet. This leads to the promotion of the standby server to the background server, and a service shutdown of the prevoious background server.

    - In a rolling maintenance situation, the standby server is promoted to the new background server, and the background server gets demoted to the active standby server. This is initialized by a manual "pdm_server_control -b" on the standby server.
- An applicaiton server is completely independent of a background server, while missing the services of the singleton processes. The application server can run even if the background server is not available.
- Background singleton processes are serving:
    - Event handling (animator_nxd)
    - KT daemons
    - Mail eater
    - Mail sender
    - Archive and Purge
    - DB Monitor
    - LDAP daemon
    - TNG Converter/Filter
    - Version Control
    - Authentication, if configured to run on BG server
    - TTV
    - WSP
- Authentication and session handling is separated in two different daemons. bopauth_nxd is used for authentication. It usualy runs on the BG server, but can be moved to an application server by an Options Manager setting. The boplgin daemon now runs on each server and is responsible for session handling only.
- You should always start the background server before all others servers.

Open Questions

- Only one web.cfg.tpl seems to be available. How about using webengines with different web.cfg settings on the same server?
-