IBM Security Directory Server 6.3.1 Console Installation for 64-bit

Installing SDS on SUSE Linux Enterprise Server 11 using only commands

*** Contact Valdemar Lemche <valdemar@lemche.net> for latest version ***

Environment Type (Mark with X):

Production	Testing	Sandbox
		X

Infrastructure Overview

Fill out anything applicable, and add additional rows and tables if needed.

Configuration Details

Users and groups:

General User Informations

Purpose	User name	Password
WAS Server ID	wasadmin	passw0rd
Portal Administrator	wpsadmin	passw0rd
Kerberos SPN user	krb_wp85@EXAMPLE.LOCAL	passw0rd
WAS Super user	iscadmin	passw0rd
WAS runtime user	was	*[none]*
IHS runtime user	ihs	*[none]*

General Group Informations

Purpose	Group name	Members
WAS Adminstrators	wasadmins	dilbert
Portal Administrators	wpsadmins	wasadmins
WAS runtime group	was	was
IHS runtime group	ihs	ihs

Network Environment:

Nodes:

Purpose	FQDN	Aliases
Primary Portal Server and deployment manager	sdsm.example.com	
Additional Portal Server	sdss.example.com	
Web Frontend	sdsproxy-1.dmz.example.com	
Web Frontend	sdsproxy-2.dmz.example.com	
Load Balancer	edge85-1.example.com	
Load Balancer	edge85-2.example.com	
Edge Cluster IP	ldap.example.com	

Domain Name System:

DNS server 1:	172.16.226.20
DNS server 2:	
Domain name:	example.com
Domain search list:	example.com

Networks:

Purpose	Public Interface	Heartbeat Interface	Backup Interface
Network Address:	172.16.226.0		
Netmask:	255.255.255.0		
Gateway:	172.16.226.254		
ldap	172.16.226.5		
sdsm	172.16.226.10		
sdss	172.16.226.11		
sdsproxy-1	172.16.226.31		
sdsproxy-2	172.16.226.32		
edge85-1	172.16.226.55		
edge85-2	172.16.226.56		

Database Servers:

DB2:

Host name:	db2.example.com
Port:	50001
Server Instance username:	db2inst1
Server Instance password:	passw0rd
Client Instance:	N/A
Client Instance user:	N/A
Client Instance user home directory:	N/A
JDBC String:	jdbc:db2://db2.example.com:50001/WP85DB:returnAlias=0;
JDBC driver class path:	/opt/ibm/db2/V10.5/java/db2jcc4.jar:/opt/ibm/db2/V10.5/java/db2jcc_license_cu.jar
JDBC java class:	com.ibm.db2.jcc.DB2Driver
JDBC provider name:	wpdbJDBC_db2
DB Configuration username:	db2usr1
DB Configuration password:	passw0rd

Databases:

Datasource:	Database:	Schema:	User:	Password:
wpdbDS	WP85DB	FDBKDB	db2run1	passw0rd

LDAP

LDAP host name:	ldap.example.com
LDAP port:	636

LDAP Type	IDS
BaseDN:	o=example
Bind DN:	uid=wpsbind,ou=users,o=example
Bind DN Password:	passw0rd
User Filter:	(objectclass=inetOrgPerson)
Group Filter:	(objectclass=groupOfNames)
WAS login attribute:	uid
WAS group member attribute:	member
WAS group member scope:	nested
WAS group membership attribute:	ibm-allGroups
WAS group membership scope:	nested

WebSphere Application Server

WAS Administrative Console

URL:	http://sdsm.example.com:9060/admin
User account repository	Federated repository
Cell name:	sdsmCell01

Clusters

Name	**Members**
PortalCluster	sdsmNode01/WebSphere_Portal sdssNode01/WebSphere_Portal

Nodes and servers

Server	**Node**	**WAS Server**
sdsm.example.com	sdsmCellManager01	dmgr
sdsm.example.com	sdsmNode01	nodeagent
sdsm.example.com	sdsmNode01	WebSphere_Portal
sdss.example.com	sdssNode01	nodeagent
sdss.example.com	sdssNode01	WebSphere_Portal

Server configurations

sdsmCellManager01/dmgr

Setting	**Value**
Heap size (min/max):	
Classpath:	
Generic JVM arguments:	
Run As User:	was

Run As Group:	was
Umask	0022

sdsmNode01/nodeagent

Setting	**Value**
Heap size (min/max):	
Classpath:	
Generic JVM arguments:	
Run As User:	was
Run As Group:	was
Umask	0022

sdsmNode01/WebSphere_Portal

Setting	**Value**
Heap size (min/max):	
Classpath:	
Generic JVM arguments:	
Run As User:	was
Run As Group:	was
Umask	0022

sdssNode01/nodeagent

Setting	**Value**
Heap size (min/max):	
Classpath:	
Generic JVM arguments:	
Run As User:	was
Run As Group:	was
Umask	0022

sdssNode01/WebSphere_Portal

Setting	**Value**
Heap size (min/max):	
Classpath:	
Generic JVM arguments:	
Run As User:	was
Run As Group:	was
Umask	0022

Server listening ports

sdsmCellManager01/dmgr

Port Name	Port
CELL_DISCOVERY_ADDRESS	7277
BOOTSTRAP_ADDRESS	9809
IPC_CONNECTOR_ADDRESS	9632
SOAP_CONNECTOR_ADDRESS	8879
ORB_LISTENER_ADDRESS	9100
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9401
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9402
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9403
WC_adminhost	9060
WC_adminhost_secure	9043
DCS_UNICAST_ADDRESS	9352
XDAGENT_PORT	7060
OVERLAY_UDP_LISTENER_ADDRESS	7060
OVERLAY_TCP_LISTENER_ADDRESS	11006
DataPowerMgr_inbound_secure	5555
STATUS_LISTENER_ADDRESS	9420

sdsmNode01/nodeagent

Port Name	Port
BOOTSTRAP_ADDRESS	2809
ORB_LISTENER_ADDRESS	9900
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9202
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9201
DCS_UNICAST_ADDRESS	9353
NODE_DISCOVERY_ADDRESS	7272
NODE_IPV6_MULTICAST_DISCOVERY_ADDRESS	5001
NODE_MULTICAST_DISCOVERY_ADDRESS	5000
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9901
SOAP_CONNECTOR_ADDRESS	8878
IPC_CONNECTOR_ADDRESS	9629
XDAGENT_PORT	7062
OVERLAY_TCP_LISTENER_ADDRESS	11004

OVERLAY_UDP_LISTENER_ADDRESS	11003

sdsmNode01/WebSphere_Portal

Port Name	Port
BOOTSTRAP_ADDRESS	10032
SOAP_CONNECTOR_ADDRESS	10033
ORB_LISTENER_ADDRESS	10034
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	10035
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	10036
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	10037
WC_adminhost	10038
WC_defaulthost	10039
DCS_UNICAST_ADDRESS	10040
WC_adminhost_secure	10041
WC_defaulthost_secure	10042
SIP_DEFAULTHOST	10043
SIP_DEFAULTHOST_SECURE	10044
OVERLAY_UDP_LISTENER_ADDRESS	10045
OVERLAY_TCP_LISTENER_ADDRESS	10046
IPC_CONNECTOR_ADDRESS	10047
SIB_ENDPOINT_ADDRESS	10048
SIB_ENDPOINT_SECURE_ADDRESS	10049
SIB_MQ_ENDPOINT_ADDRESS	10050
SIB_MQ_ENDPOINT_SECURE_ADDRESS	10051

sdssNode01/nodeagent

Port Name	Port
BOOTSTRAP_ADDRESS	2809
ORB_LISTENER_ADDRESS	9900
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9202
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9201
DCS_UNICAST_ADDRESS	9353
NODE_DISCOVERY_ADDRESS	7272
NODE_IPV6_MULTICAST_DISCOVERY_ADDRESS	5001
NODE_MULTICAST_DISCOVERY_ADDRESS	5000

SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9901
SOAP_CONNECTOR_ADDRESS	8878
IPC_CONNECTOR_ADDRESS	9629
XDAGENT_PORT	7062
OVERLAY_TCP_LISTENER_ADDRESS	11004
OVERLAY_UDP_LISTENER_ADDRESS	11003

sdssNode01/WebSphere_Portal

Port Name	**Port**
BOOTSTRAP_ADDRESS	10052
SOAP_CONNECTOR_ADDRESS	10053
ORB_LISTENER_ADDRESS	10054
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	10055
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	10056
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	10057
WC_adminhost	10058
WC_defaulthost	10059
DCS_UNICAST_ADDRESS	10060
WC_adminhost_secure	10061
WC_defaulthost_secure	10062
SIP_DEFAULTHOST	10063
SIP_DEFAULTHOST_SECURE	10064
OVERLAY_UDP_LISTENER_ADDRESS	10065
OVERLAY_TCP_LISTENER_ADDRESS	10066
IPC_CONNECTOR_ADDRESS	10067
SIB_ENDPOINT_ADDRESS	10068
SIB_ENDPOINT_SECURE_ADDRESS	10069
SIB_MQ_ENDPOINT_ADDRESS	10070
SIB_MQ_ENDPOINT_SECURE_ADDRESS	10071

Security

LTPA (NB: LTPA cookie names are case sensitive):

Key location:	/net/main/srv/common-setup/ltpa/example-ltpa.key
Key password:	passw0rd
Domain:	.example.com

Interoperability mode:	On
LTPA V1 cookie name:	LtpaToken
LTPA V2 cookie name:	LtpaToken2

SPNEGO

User account DN:	cn=krb_wp85,ou=Service Accounts,dc=example,dc=local
User account password:	passw0rd
Dynamically update SPNEGO:	true
Enable SPNEGO:	true
Allow fall back to application authentication mechanism:	true
Keytab file:	/opt/IBM/WebSphere/spnego/krb_wp85.keytab
Kerberos configuration file:	/opt/IBM/WebSphere/spnego/krb5.conf
Service principal name:	HTTP/wp85.example.com@EXAMPLE.LOCAL
Kerberos realm name:	EXAMPLE.LOCAL
Filter criteria:	request-url!=/seedlist/authverify;request-url!=/seedlist/server;request-url!=/seedlist/myserver;request-url!=noSPNEGO
Filter class:	
SPNEGO not supported error page URL:	http://wp85.example.com/TAIRedirect.html
NTLM token received error page URL:	http://wp85.example.com/TAIRedirect.html
Trim Kerberos realm from principal name:	true
Enable delegation of Kerberos credentials:	true

VMM Federated repositories

Realm:	ldap.example.com:636
Primary administrative user name (Server ID):	uid=wasadmin,ou=users,o=example
Server user identity	Automatically generated server identity
Use built-in repository	No

Federated repositories baseDN

WIM	LDAP	Repository Identifier
o=example	o=example	RepositoryID_ldap.example.com:636

WebSphere Variables

Only different and changed compared to the default variables are listed.

Name	Value	Scope
LOG_ROOT	/var/log/was/${WAS_CELL_NAME}	Node=sdsmCellManager01

LOG_ROOT	/var/log/was/${WAS_CELL_NAME}	Node=sdssNode01
LOG_ROOT	/var/log/was/${WAS_CELL_NAME}	Node=sdsmNode01
SERVER_LOG_ROOT	${LOG_ROOT}/${WAS_SERVER_NAME}	Node=sdsmCellManager01,Server=dmgr
SERVER_LOG_ROOT	${LOG_ROOT}/${WAS_SERVER_NAME}	Node=sdsproxy-2UnmanagedNode,Server=webserver1
SERVER_LOG_ROOT	${LOG_ROOT}/${WAS_SERVER_NAME}	Node=sdssNode01,Server=WebSphere_Portal
SERVER_LOG_ROOT	${LOG_ROOT}/${WAS_SERVER_NAME}	Node=sdsmNode01,Server=WebSphere_Portal
SERVER_LOG_ROOT	${LOG_ROOT}/${WAS_SERVER_NAME}	Node=sdsproxy-1UnmanagedNode,Server=webserver1

IBM WebSphere Portal / IBM Web Content Management

Custom Portlets	TBD
Custom EARs	TBD
Cache configuration	TBD

Mail

Mail exchange host:	
MX user:	
MX user password:	

IHS source keystores

sdsproxy-1.dmz.example.com

Database:	/opt/IBM/HTTPServer/conf/sdsproxy-1.dmz.example.com.kdb
Password:	passw0rd

sdsproxy-2.dmz.example.com

Database:	/opt/IBM/HTTPServer/conf/sdsproxy-2.dmz.example.com.kdb
Password:	passw0rd

FTP Server

FTP URL:	
FTP User:	
FTP User Password:	
FTP Server Home Directory:	
FTP Server Security Accounts:	

Permissions:	

NFS Servers

Installation files

URL:	files.example.com/srv
Mount point:	/net/files/srv (using automount)

Other files

URL:	main.example.com/srv
Mount point:	/net/main/srv (using automount)

Installation notes

Prepare installation files

Server	Task
sdsm sdss	Create the installation source directory `mkdir -p /net/files/srv/install/sds/6310/linux64`

sdsm
sdss
	Extract all images
	`tar xf ~/DownloadDirector/sds631-linux-x86-64-base.tar -C /net/files/srv/install/sds/6310/linux64`
	`tar xf ~/DownloadDirector/sds631_linux_x86-64_db2.tar -C /net/files/srv/install/sds/6310/linux64`
	`tar xf ~/DownloadDirector/sds631_linux_x86_64-ewas.tar -C /net/files/srv/install/sds/6310/linux64`
	`tar xf ~/DownloadDirector/sds631_linux_x86_64_gskit.tar -C /net/files/srv/install/sds/6310/linux64`
	`tar xf ~/DownloadDirector/sds631_linux_x86_64_IM.tar -C /net/files/srv/install/sds/6310/linux64`
	`tar xf ~/DownloadDirector/sds631_linux_x86_64_jdk.tar -C /net/files/srv/install/sds/6310/linux64`

Installation of components

OS preparation

Server	Task
sdsm sdss	Install pre-req packages `zypper install pam-32bit`
sdsm sdss	Set a installation source directory variable: `export INSTALL_DIR=/net/files/srv/install/sds/6310/linux64/sdsV6.3.1`

~~Installing IBM Installation Manager~~

~~Server~~	~~Task~~
~~sdsm~~ ~~sdss~~ ~~sdsproxy-1~~ ~~sdsproxy-2~~ ~~edge85-1~~ ~~edge85-2~~	~~`/net/files/srv/install/wp/8500/SETUP/IIM/linux_x86_64/installc\`~~ ~~`--acceptLicense\`~~ ~~`--log /tmp/install-im1710-log.xml\`~~ ~~`--showVerboseProgress`~~
~~sdsm~~ ~~sdss~~ ~~sdsproxy-1~~ ~~sdsproxy-2~~ ~~edge85-1~~ ~~edge85-2~~	~~Add imcl to PATH~~ ~~`cat > /etc/profile.d/imcl_path.sh << EOF`~~ ~~`#!/bin/sh`~~ ~~`test \$UID -lt 100 &&`~~ ~~`PATH=\${PATH}:/opt/IBM/InstallationManager/eclipse/tools`~~ ~~`export PATH`~~ ~~`EOF`~~ ~~`. /etc/profile.d/imcl_path.sh`~~

Install DB2 Enterprise Server Edition 9.7.0.6

Server	Task

sdsm sdss	Install the DB2 server that comes with SDS `$INSTALL_DIR/ibm_db2/db2_install -b /opt/ibm/db2/V9.7 -p ESE`

sdsm
sdss
	Add the DB2 IBM DB2 Restricted Enterprise Server Edition, V9.7 license
	`/opt/ibm/db2/V9.7/adm/db2licm -a`
	`$INSTALL_DIR/ibm_db2/db2/license/db2ese_o.lic`

Install Global Security Kit 8.0.14.26

Server	Task

sdsm
sdss
	`rpm -ivh \`
	`$INSTALL_DIR/ibm_gskit/gskcrypt64-8.0.14.26.linux.x86_64.rpm \`
	`$INSTALL_DIR/ibm_gskit/gskssl64-8.0.14.26.linux.x86_64.rpm`

Install IBM Java Development Kit for IBM Security Directory Server

Server	Task

sdsm
sdss
	None of the commands in this document will require the SDS JDK, but if you want to use any of the SDS GUI tools for instance administration, and configuration installed with **idsldap-cltjava631-6.3.1-0.x86_64.rpm**, then you need to extract this package using the command below.
	`mkdir -p /opt/ibm/ldap/V6.3.1`
	`tar -xf $INSTALL_DIR/ibm_jdk/ibm-java-16sr14-linux-64.tar -C /opt/ibm/ldap/V6.3.1/`
	
	However if you don't need any of the GUI tools, then don't install the JDK, and omit installing **idsldap-cltjava631-6.3.1-0.x86_64.rpm below.**

Install IBM Security Directory Server 6.3.1.0

Server	Task
sdsm sdss	The package **idsldap-license631-6.3.1-0.x86_64.rpm** checks for the license file **/opt/ibm/ldap/V6.3.1/LAPID**, before it allows to install. So install that first by running: `echo 1 | $INSTALL_DIR/license/idsLicense`

sdsm
sdss
	Then you can install the SDS packages.
	`rpm -ihv \`
	`$INSTALL_DIR/native/idsldap-license631-6.3.1-0.x86_64.rpm \`
	`$INSTALL_DIR/native/idsldap-cltbase631-6.3.1-0.x86_64.rpm \`
	`$INSTALL_DIR/native/idsldap-clt64bit631-6.3.1-0.x86_64.rpm \`
	`$INSTALL_DIR/native/idsldap-cltjava631-6.3.1-0.x86_64.rpm \`
	`$INSTALL_DIR/native/idsldap-srvbase64bit631-6.3.1-0.x86_64.rpm \`
	`$INSTALL_DIR/native/idsldap-srvproxy64bit631-6.3.1-0.x86_64.rpm \`
	`$INSTALL_DIR/native/idsldap-srv64bit631-6.3.1-0.x86_64.rpm \`
	`$INSTALL_DIR/native/idsldap-msg631-en-6.3.1-0.x86_64.rpm \`
	`$INSTALL_DIR/native/idsldap-ent631-6.3.1-0.x86_64.rpm`

sdsm
sdss
	Add SDS commands to PATH
	`cat > /etc/profile.d/sds_path.sh << EOF`
	`#!/bin/sh`
	`PATH=\${PATH}:/opt/IBM/ldap/V6.3.1/bin`
	`test \$UID -lt 100 && PATH=\${PATH}:/opt/IBM/ldap/V6.3.1/sbin`
	`export PATH`
	`EOF`

	`. /etc/profile.d/sds_path.sh`

Create default SDS instance

Server	Task

sdsm
sdss
	Create local system accounts for instance
	`groupadd -g 1001 grrdbm01`
	`useradd -m -u 1001 -g grrdbm01 -G idsldap -s /bin/ksh dsrdbm01`
	`echo passw0rd | passwd --stdin dsrdbm01`

sdsm
sdss
	Add root to groups
	`groupmod -A root idsldap`
	`groupmod -A root grrdbm01`

sdsm sdss	Create SDS instance - The "`1234567890qwertyuiopasdfghjklzxcvbnm`" number should be saved for later as its the encryption seed used for the directory encryption key. `idsicrt -I dsrdbm01 -e 1234567890qwertyuiopasdfghjklzxcvbnm -r "IBM Security Directory Server Instance V6.3.1" -n`
sdsm sdss	Define the instance server root (administrator) `idsdnpw -u cn=root -p passw0rd -n`
sdsm sdss	Configure a database to be used `idscfgdb -I dsrdbm01 -a dsrdbm01 -w passw0rd -t dsrdbm01 -l /home/dsrdbm01 -n`
sdsm sdss	Start default instance for the first time `idsslapd`

Enforce SSHA-256 password security algorithm

By default SDS uses the AES-256 security algorithm to store passwords, which is retrieved in clear format. I find that a little bit insecure as anyone with access to read userPassword can see all passwords in clear text. But if you change the default password security algorithm to SSHA-256, then password is retrieved in their hashed format.
SDS will still accept userPassword value of a different algorithm such as SSHA-512 if the client sends the raw value. But if SDS will default back to SSHA-256 if userPassword value is set unhashed.

Server	Task

sdsm
sdss
	Set the password encryption to SSHA-256.
	`cat > /tmp/enable-ssha256.ldif << EOF`
	`dn: cn=configuration`
	`changetype: modify`
	`replace: ibm-slapdPWEncryption`
	`ibm-slapdPWEncryption: ssha256`
	`EOF`
	`idsldapmodify -D cn=root -w passw0rd -i /tmp/enable-ssha256.ldif`

sdsm
sdss
	Restart the instance
	`idsslapd -k`
	`idsslapd`

Create SysV scripts

The instances administration server is started using **/etc/inittab**, and the DB2 instance is started automatically when the LDAP server is started. So the SysV scripts for those two daemons are just for easier start-stop commands.

Server	Task

sdsm
sdss
	DB2 instance
	`cat > /etc/init.d/dsrdbm01 << EOF`
	`#!/bin/sh`
	`# Copyright (c) 2007 IBM Corp.`
	`#`
	`# Author: Valdemar Lemche (DK29563) <valdemar.lemche@dk.ibm.com>`
	`#`
	`# /etc/init.d/dsrdbm01`
	`#`
	`### BEGIN INIT INFO`
	`# Provides: dsrdbm01`
	`# Required-Start: \$network`
	`# Required-Stop:`
	`# Default-Start: 3 5`
	`# Default-Stop: 0 1 2 4 6`
	`# Description: db2start - Starts the DB2 instance dsrdbm01`
	`### END INIT INFO`
	
	`PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin`
	`RUNDIR=/var/run`
	`DB2INST=dsrdbm01`
	
	`case "\$1" in`
	`start)`
	`echo -n "Starting DB2 instance, \$DB2INST: db2start"`
	`su - \$DB2INST -c db2start >> /var/log/rc\$DB2INST.log 2>&1`
	`echo .`
	`;;`
	`stop)`
	`echo -n "Stopping DB2 instance, \$DB2INST: db2stop"`
	`su - \$DB2INST -c "db2 force application all" >>`
	`/var/log/rc\$DB2INST.log 2>&1`
	`su - \$DB2INST -c db2stop >> /var/log/rc\$DB2INST.log 2>&1`
	`echo .`
	`;;`
	`restart)`
	`\$0 stop`
	`\$0 start`
	`;;`
	`*)`
	`echo "Usage: \$0 {start|stop|restart}" >&2`
	`exit 1`
	`;;`
	`esac`
	`exit 0`
	`EOF`

sdsm
sdss
	SDS instance administration server
	`cat > /etc/init.d/ibmdiradm << EOF`
	`#!/bin/sh`
	`# Copyright (c) 2011 IBM Corp.`
	`#`
	`# Author: Valdemar Lemche (DK29563) <valdemar.lemche@dk.ibm.com>`
	`#`

```
# /etc/init.d/ibmdiradm
#
### BEGIN INIT INFO
# Provides:          ibmdiradm
# Required-Start:    \$network \$dsrdbm01
# Required-Stop:
# Default-Start:     3 5
# Default-Stop:      0 1 2 4 6
# Description: ibmdiradm - Starts a single IBM Security Directory Server
### END INIT INFO

PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
RUNDIR=/var/run
INSTANCE=dsrdbm01
TDS_HOME=/opt/ibm/ldap/V6.3.1

case "\$1" in
    start)
        echo -n "Starting IBM Security Directory Server: ibmdiradm".
        \$TDS_HOME/sbin/ibmdiradm -I \$INSTANCE -t >>
/var/log/rcibmdiradm-\$INSTANCE.log 2>&1
        echo .
        ;;
    stop)
        echo -n "Stopping IBM Security Directory Server: ibmdiradm"
        \$TDS_HOME/sbin/ibmdiradm -I \$INSTANCE -k >>
/var/log/rcibmdiradm-\$INSTANCE.log 2>&1
        echo .
        ;;
    restart)
        \$0 stop
        \$0 start
        ;;
    *)
        echo "Usage: \$0 {start|stop|restart}" >&2
        exit 1
        ;;
esac
exit 0
EOF
```

sdsm
sdss

SDS slapd daemon

```
cat > /etc/init.d/ibmslapd << EOF
#!/bin/sh
# Copyright (c) 2011 IBM Corp.
#
# Author: Valdemar Lemche (DK29563) <valdemar.lemche@dk.ibm.com>
#
# /etc/init.d/ibmslapd
#
### BEGIN INIT INFO
# Provides:          ibmslapd
# Required-Start:    \$network \$dsrdbm01
# Required-Stop:
# Default-Start:     3 5
# Default-Stop:      0 1 2 4 6
```

	`# Description: ibmslapd - Starts a single IBM Security Directory Server`
	`### END INIT INFO`
	
	`PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin`
	`RUNDIR=/var/run`
	`INSTANCE=dsrdbm01`
	`TDS_HOME=/opt/ibm/ldap/V6.3.1`
	
	`case "\$1" in`
	`start)`
	`echo -n "Starting IBM Security Directory Server: ibmslapd".`
	`\$TDS_HOME/sbin/ibmslapd -I \$INSTANCE -t >>`
	`/var/log/rcibmslapd-\$INSTANCE.log 2>&1`
	`echo .`
	`;;`
	`stop)`
	`echo -n "Stopping IBM Security Directory Server: ibmslapd"`
	`\$TDS_HOME/sbin/ibmslapd -I \$INSTANCE -k >>`
	`/var/log/rcibmslapd-\$INSTANCE.log 2>&1`
	`echo .`
	`;;`
	`restart)`
	`\$0 stop`
	`\$0 start`
	`;;`
	`*)`
	`echo "Usage: \$0 {start|stop|restart}" >&2`
	`exit 1`
	`;;`
	`esac`
	`exit 0`
	`EOF`

sdsm
sdss
	Make the scripts executable
	`chmod +x /etc/init.d/dsrdbm01`
	`chmod +x /etc/init.d/ibmdiradm`
	`chmod +x /etc/init.d/ibmslapd`

sdsm sdss	Tell ibmslapd to start at boot. `chkconfig ibmslapd on`

Create a sample directory

Server	**Task**
sdsm sdss	Stop the instance `idsslapd -k`
sdsm sdss	Create a sample suffix `idscfgsuf -s o=sample -n`
sdsm sdss	Import initial sample DIT `idsldif2db -i /opt/ibm/ldap/V6.3.1/examples/sample.ldif`
sdsm sdss	Start the instance `idsslapd`
sdsm	Create an administrative user

sdss	cat > /tmp/create-dsa-ou.ldif << EOF # Create a DSA OU to contain directory service accounts (DSA) dn: ou=DSA,O=SAMPLE changetype: add objectClass: organizationalUnit objectClass: top ou: DSA description: Directory System Agents EOF idsldapmodify -D cn=root -w passw0rd -i /tmp/create-dsa-ou.ldif
sdsm sdss	cat > /tmp/ldapadmin-dsa.ldif << EOF # Create a LDAP administrator DSA dn: cn=ldapadmin,ou=DSA,o=sample changetype: add objectClass: simpleAuthObject objectClass: top objectClass: dSA objectClass: applicationEntity userPassword: passw0rd presentationAddress: sdsm.example.com cn: ldapadmin EOF idsldapmodify -D cn=root -w passw0rd -i /tmp/ldapadmin-dsa.ldif
sdsm sdss	Enable the server administrative group cat > /tmp/enableAdminGroup.ldif << EOF dn: cn=Configuration changetype: modify replace: ibm-slapdAdminGroupEnabled #specify TRUE to enable or FALSE to disable the administrative group #TRUE has been preselected for you. ibm-slapdAdminGroupEnabled: TRUE EOF idsldapmodify -D cn=root -w passw0rd -i /tmp/enableAdminGroup.ldif
sdsm sdss	Re-read configuration idsldapexop -D cn=root -w passw0rd -op readconfig -scope single cn=Configuration ibm-slapdAdminGroupEnabled

sdsm
sdss
	Define ldapadmin as server administrator
	cat > /tmp/addMembersToAdminGroup.ldif << EOF
	# Create global administrator DSE that points to
	cn=ldapadmin,ou=DSA,O=SAMPLE
	dn: cn=ldapadmin,cn=AdminGroup,cn=Configuration
	changetype: add
	objectClass: top
	objectClass: ibm-slapdConfigEntry
	objectClass: ibm-slapdAdminGroupMember
	ibm-slapdAdminDN: cn=ldapadmin,ou=DSA,O=SAMPLE
	ibm-slapdAdminPW: passw0rd
	ibm-slapdAdminRole: DirDataAdmin
	ibm-slapdAdminRole: PasswordAdmin
	ibm-slapdAdminRole: SchemaAdmin
	ibm-slapdAdminRole: ServerConfigGroupMember
	ibm-slapdAdminRole: ServerStartStopAdmin
	ibm-slapdDigestAdminUser: ldapadmin

	cn: ldapadmin EOF idsldapadd -D cn=root -w passw0rd -i /tmp/addMembersToAdminGroup.ldif
sdsm sdss	Re-read the instance configuration (instead of restarting) idsldapexop -D cn=root -w passw0rd -op readconfig -scope subtree cn=AdminGroup,cn=Configuration

Enable LDAPS

Server	Task
sdsm sdss	Set variable which points to GSK command export GSK_CMD=/usr/local/ibm/gsk8_64/bin/gsk8capicmd_64
sdsm sdss	Create keystore database $GSK_CMD \ -keydb \ -create \ -db /home/dsrdbm01/idsslapd-dsrdbm01/etc/serverkey.kdb \ -pw passw0rd \ -type cms \ -stash
sdsm sdss	Add CA certificate to keystore $GSK_CMD \ -cert \ -add \ -db /home/dsrdbm01/idsslapd-dsrdbm01/etc/serverkey.kdb \ -pw passw0rd \ -type cms \ -label example-ca \ -file /net/main/srv/common-setup/ssl/cacert.pem \ -format ascii \ -trust enable
sdsm sdss	Import X.509 certificate, and RSA key (AKA personal certificate) into keystore $GSK_CMD \ -cert \ -import \ -db /net/main/srv/common-setup/ssl/$HOSTNAME.example.com-cert.p12 \ -target /home/dsrdbm01/idsslapd-dsrdbm01/etc/serverkey.kdb \ -pw passw0rd \ -target_pw passw0rd \ -type pkcs12 \ -label $HOSTNAME.example.com \ -target_type cms
sdsm sdss	Set the default certificate $GSK_CMD \ -cert \ -setdefault \ -db /home/dsrdbm01/idsslapd-dsrdbm01/etc/serverkey.kdb \ -label $HOSTNAME.example.com \ -pw passw0rd \ -type cms
sdsm sdss	List certificates in keystore $GSK_CMD \

	-cert \ -list \ -db /home/dsrdbm01/idsslapd-dsrdbm01/etc/serverkey.kdb \ -pw passw0rd
sdsm sdss	Change ownership to instance user chown dsrdbm01:grrdbm01 /home/dsrdbm01/idsslapd-dsrdbm01/etc/serverkey.*
sdsm sdss	Update renewed certificate (for when the certificate have expired in the keystore) $GSK_CMD \ -cert \ -receive \ -db /home/dsrdbm01/idsslapd-dsrdbm01/etc/serverkey.kdb \ -file "/net/main/srv/common-setup/ssl/$HOSTNAME.example.com-cert.pem" \ -pw passw0rd \ -type kdb \ -format ascii \ -default_cert yes
sdsm sdss	Tell server to enable SSL cat > /tmp/enableSSL.ldif << EOF dn: cn=SSL,cn=Configuration changetype: modify replace: ibm-slapdSSLAuth ibm-slapdSSLAuth: serverAuth - replace: ibm-slapdSecurity ibm-slapdSecurity: SSLTLS EOF idsldapmodify -D cn=root -w passw0rd -i /tmp/enableSSL.ldif
sdsm sdss	Define what GSK key database to use, and which password to use with it cat > /tmp/settingKeyDatabase.ldif << EOF dn: cn=SSL,cn=Configuration changetype: modify replace: ibm-slapdSSLKeyDatabase ibm-slapdSSLKeyDatabase: /home/dsrdbm01/idsslapd-dsrdbm01/etc/serverkey.kdb - replace: ibm-slapdSSLKeyDatabasePW ibm-slapdSSLKeyDatabasePW: passw0rd - replace: ibm-slapdSslCertificate ibm-slapdSslCertificate: $HOSTNAME.example.com EOF idsldapmodify -D cn=root -w passw0rd -i /tmp/settingKeyDatabase.ldif
sdsm sdss	Restrict the security protocols to TLSv1.2 cat > /tmp/restrictSslProtocolToTLS12.ldif << EOF dn: cn=SSL,cn=Configuration changetype: modify replace: ibm-slapdSecurityProtocol ibm-slapdsecurityprotocol: TLS12 EOF idsldapmodify -D cn=root -w passw0rd -i /tmp/restrictSslProtocolToTLS12.ldif
sdsm sdss	Enable TLSv1.2 cipher specifications cat > /tmp/enableTls12Ciphers.ldif << EOF

```
dn: cn=SSL,cn=Configuration
changetype: modify
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_RSA_WITH_RC4_128_SHA
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_RSA_WITH_3DES_EDE_CBC_SHA
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_RSA_WITH_AES_128_CBC_SHA
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_RSA_WITH_AES_128_GCM_SHA256
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_RSA_WITH_AES_256_CBC_SHA
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_RSA_WITH_AES_256_GCM_SHA384
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_RSA_WITH_AES_128_CBC_SHA256
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_RSA_WITH_AES_256_CBC_SHA256
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_ECDHE_RSA_WITH_RC4_128_SHA
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
-
add: ibm-slapdSslCipherSpec
ibm-slapdSslCipherSpec: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
-
```

	add: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA - add: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA - add: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 - add: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 - add: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 - add: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 EOF idsldapmodify -D cn=root -w passw0rd -i /tmp/enableTls12Ciphers.ldif
sdsm sdss	I like to use Apache Directory Studio, as its by far the best LDAP client available. But it does only support TLSv1.0. Futhermore SDS' own Web Administration Tool, and replication also only support protocols up to TLSv1.0. (At least we don't have to use SSLv3!). cat > /tmp/restrictSslProtocolToTLS10.ldif << EOF dn: cn=SSL,cn=Configuration changetype: modify replace: ibm-slapdSecurityProtocol ibm-slapdSecurityProtocol: TLS10 - add: ibm-slapdsecurityprotocol ibm-slapdsecurityprotocol: TLS12 EOF idsldapmodify -D cn=root -w passw0rd -i /tmp/restrictSslProtocolToTLS10.ldif But at least we should remove these old ciphers cat > /tmp/removeUnsafeCiphers.ldif << EOF dn: cn=SSL,cn=Configuration changetype: modify delete: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: RC4-128-MD5 - delete: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: RC4-128-SHA - delete: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: TripleDES-168 - delete: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: DES-56 - delete: ibm-slapdSslCipherSpec ibm-slapdSslCipherSpec: RC4-40-MD5 - delete: ibm-slapdSslCipherSpec

	ibm-slapdSslCipherSpec: RC2-40-MD5 EOF idsldapmodify -D cn=root -w passw0rd -i /tmp/removeUnsafeCiphers.ldif
sdsm sdss	Enable FIPS cat > /tmp/enableFIPS.ldif << EOF dn: cn=SSL,cn=Configuration changetype: modify replace: ibm-slapdSslFIPSModeEnabled ibm-slapdSslFIPSModeEnabled: true - replace: ibm-slapdSslFIPSProcessingMode ibm-slapdSslFIPSProcessingMode: true EOF idsldapmodify -D cn=root -w passw0rd -i /tmp/enableFIPS.ldif
sdsm sdss	Restart the default instance idsslapd -k idsslapd

Add custom CA signer certificate to default SSL truststore for command-line tools

Server	Task
sdsm sdss	Set variable which points to GSK command export GSK_CMD=/usr/local/ibm/gsk8_64/bin/gsk8capicmd_64
sdsm sdss	Add CA certificate to client tools keystore $GSK_CMD \ -cert \ -add \ -db /opt/ibm/ldap/V6.3.1/etc/ldapkey.kdb \ -pw ssl_password \ -type cms \ -label example-ca \ -file /net/main/srv/common-setup/ssl/cacert.pem \ -format ascii \ -trust enable
sdsm sdss	List certificates in keystore $GSK_CMD \ -cert \ -list \ -db /opt/ibm/ldap/V6.3.1/etc/ldapkey.kdb \ -pw ssl_password
sdsm sdss	Check that ids* command line tools can use SSL idsldapsearch -h sdsm.example.com **-p 636 -Z** -D cn=root -w passw0rd -b "" -s sub "(objectclass=organization)"

Configure peer to peer replication

A lot of the commands are run from the sdsm, but toward sdss. Its because the majority of the SDS configuration is actually done with LDIF commands, which can be executed from any client with LDAP tools.

Define global replication configuration

Server	Task

sdsm	Change the server ID's to something a little more human readable than the default UUID.
	`cat > /tmp/sdsm-changeServerId.ldif << EOF`
	`dn: cn=Configuration`
	`changetype: modify`
	`replace: ibm-slapdServerId`
	`ibm-slapdServerId: sdsm`
	`EOF`
	`idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/sdsm-changeServerId.ldif`
	
	`cat > /tmp/sdss-changeServerId.ldif << EOF`
	`dn: cn=Configuration`
	`changetype: modify`
	`replace: ibm-slapdServerId`
	`ibm-slapdServerId: sdss`
	`EOF`
	`idsldapmodify -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/sdss-changeServerId.ldif`

sdsm	Define replication partners
	`cat > /tmp/sdsm-addReplicationPartnerEntry.ldif << EOF`
	`dn: cn=Master server,cn=configuration`
	`cn: master server`
	`ibm-slapdMasterDN: cn=master`
	`ibm-slapdMasterPW: passw0rd`
	`ibm-slapdMasterReferral: ldaps://sdss.example.com:636`
	`objectclass: ibm-slapdReplication`
	`EOF`
	`idsldapadd -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/sdsm-addReplicationPartnerEntry.ldif`
	
	`cat > /tmp/sdss-addReplicationPartnerEntry.ldif << EOF`
	`dn: cn=Master server,cn=configuration`
	`cn: master server`
	`ibm-slapdMasterDN: cn=master`
	`ibm-slapdMasterPW: passw0rd`
	`ibm-slapdMasterReferral: ldaps://sdsm.example.com:636`
	`objectclass: ibm-slapdReplication`
	`EOF`
	`idsldapadd -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/sdss-addReplicationPartnerEntry.ldif`

sdsm
sdss
	Restart TDS’es - You have to restart both servers now, its not enough to just tell them to re-read their configurations
	`idsslapd -k`
	`idsslapd`

Create replication configuration for "o=sample" suffix

Server	**Task**

sdsm	First you need to add a replication context objectClass to a suffix
	`cat > /tmp/addReplicationcontextObjectclassToSuffix.ldif << EOF`
	`# Add a replication context OC`
	`dn: o=sample`
	`changetype: modify`
	`add: objectclass`

	objectclass: ibm-replicationContext EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/addReplicationcontextObjectclassToSuffix.ldif
sdsm	Create a replication bind user cat > /tmp/addReplicationBindUser.ldif << EOF dn: cn=ReplicaBindCredentials,o=sample changetype: add objectclass: ibm-replicationCredentialsSimple cn: ReplicaBindCredentials replicaBindDN: cn=master replicaCredentials: passw0rd EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/addReplicationBindUser.ldif
sdsm	Create a replication group cat > /tmp/addReplicationGroup.ldif << EOF dn: ibm-replicaGroup=default,o=sample changetype: add objectclass: top objectclass: ibm-replicaGroup ibm-replicaGroup: default EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/addReplicationGroup.ldif
sdsm	Add both servers to replication group cat > /tmp/addServersToReplicationGroup.ldif << EOF dn: ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample changetype: add objectclass: top objectclass: ibm-replicaSubentry ibm-replicaServerId: sdsm ibm-replicationServerIsMaster: true cn: sdsm dn: ibm-replicaServerId=sdss,ibm-replicaGroup=default,o=sample changetype: add objectclass: top objectclass: ibm-replicaSubentry ibm-replicaServerId: sdss ibm-replicationServerIsMaster: true cn: sdss EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/addServersToReplicationGroup.ldif
sdsm	Create a replication agreements to the servers in the replication group cat > /tmp/addReplicationAgreement.ldif << EOF dn: cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample changetype: add objectclass: top objectclass: ibm-replicationAgreement cn: sdss ibm-replicaConsumerId: sdss

	ibm-replicaUrl: ldaps://sdss.example.com:636 ibm-replicaCredentialsDN: cn=ReplicaBindCredentials,o=sample dn: cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,o=sample changetype: add objectclass: top objectclass: ibm-replicationAgreement cn: sdsm ibm-replicaConsumerId: sdsm ibm-replicaUrl: ldaps://sdsm.example.com:636 ibm-replicaCredentialsDN: cn=ReplicaBindCredentials,o=sample EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/addReplicationAgreement.ldif
sdsm	Propagate replication topology to sdss idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op repltopology -rc o=sample
sdsm	Test that sdsm have propagated the replication configuration to sdss' o=sample suffix. idsldapsearch -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -b "" -s sub objectclass=ibm-repl*

Create replication configuration for "cn=ibmpolicies"

To replicate the password policies, account locks, and schema changes, then you also need to configure replication of the **cn=ibmpolicies** suffix.

A replicationContext objectclass, as well as a replication group DSE have already been created during installation. So you just need to setup a replication user, and define which servers that should participate in the replication.

Server	**Task**
sdsm	Create a replication bind user cat > /tmp/addReplicationBindUserToPolicies.ldif << EOF dn: cn=ReplicaBindCredentials,cn=ibmpolicies changetype: add objectclass: ibm-replicationCredentialsSimple cn: ReplicaBindCredentials replicaBindDN: cn=master replicaCredentials: passw0rd EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/addReplicationBindUserToPolicies.ldif
sdsm	Add both servers to replication group cat > /tmp/addServersToReplicationGroupInPolicies.ldif << EOF dn: ibm-replicaServerId=sdsm,ibm-replicaGroup=default,cn=ibmpolicies changetype: add objectclass: top objectclass: ibm-replicaSubentry ibm-replicaServerId: sdsm ibm-replicationServerIsMaster: true cn: sdsm dn: ibm-replicaServerId=sdss,ibm-replicaGroup=default,cn=ibmpolicies changetype: add objectclass: top

	objectclass: ibm-replicaSubentry ibm-replicaServerId: sdss ibm-replicationServerIsMaster: true cn: sdss EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/addServersToReplicationGroupInPolicies.ldif
sdsm	Create a replication agreements to the servers in the replication group cat > /tmp/addReplicationAgreementToPolicies.ldif << EOF dn: cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,cn=ibmpolicies changetype: add objectclass: top objectclass: ibm-replicationAgreement cn: sdss ibm-replicaConsumerId: sdss ibm-replicaUrl: ldaps://sdss.example.com:636 ibm-replicaCredentialsDN: cn=ReplicaBindCredentials,cn=ibmpolicies dn: cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,cn=ibmpolicies changetype: add objectclass: top objectclass: ibm-replicationAgreement cn: sdsm ibm-replicaConsumerId: sdsm ibm-replicaUrl: ldaps://sdsm.example.com:636 ibm-replicaCredentialsDN: cn=ReplicaBindCredentials,cn=ibmpolicies EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/addReplicationAgreementToPolicies.ldif
sdsm	Propagate replication topology to sdss idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op repltopology -rc cn=ibmpolicies

Export/Import DIT from sdsm.example.com to sdss.example.com

From here on sdsm.example.com, and sdss.example.com will send DIT updates to each other. But what about existing DSE's in sdsm.example.com? Well here's a howto on how to safely replace the entire DIT on sdss.example.com with the DIT from sdsm.example.com. This is also a useful howto to solve a situation that either of your SDS' have become corrupted, so that you aren't able to replicate between them.

Server	**Task**
sdss	Stop the server idsslapd -k
sdsm	Quiesce o=sample idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op quiesce -rc o=sample
sdsm	Skip all blocking entries for the queue from sdsm.example.com to sdss.example.com and opposite idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op controlqueue -skip all -ra cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample

sdsm	Export entire DIT `idsdb2ldif -o /net/files/srv/backup/sdsm.ldif`
sdsm	Unquiesce o=sample `idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op quiesce -rc o=sample -end`
sdss	Delete database on sdss.example.com `idsucfgdb -I dsrdbm01 -r -s -n`
sdss	Copy crypto sync key from sdsm.example.com `scp dsrdbm01@sdsm:/home/dsrdbm01/idsslapd-dsrdbm01/etc/ibmslapddir.ksf /home/dsrdbm01/idsslapd-dsrdbm01/etc`
sdss	Copy schema files from sdsm.example.com `scp dsrdbm01@sdsm:/home/dsrdbm01/idsslapd-dsrdbm01/etc/V3.* /home/dsrdbm01/idsslapd-dsrdbm01/etc`
sdss	Ensure they are owned by the instance owner, and group `chown dsrdbm01:grrdbm01 /home/dsrdbm01/idsslapd-dsrdbm01/etc/V3.*`
sdss	Configure a database to be used `idscfgdb -I dsrdbm01 -a dsrdbm01 -w passw0rd -t dsrdbm01 -l /home/dsrdbm01 -n`

sdss	Import the DIT from sdsm.example.com
	`idsbulkload -I dsrdbm01 -i /net/files/srv/backup/sdsm.ldif`
	
	NB: idsbulkload is faster than idsldif2db, but if sdsm.example.com have been populated with additional suffixes, that doesn't exist on sdss.example.com, then you have to use idsldif2db instead of idsbulkload.
	`idsldif2db -I dsrdbm01 -r yes -i /net/files/srv/backup/sdsm.ldif`

sdss	Optimize the database `idsrunstats -I dsrdbm01`
sdss	Start SDS to resume normal operation `idsslapd`

Configure Edge Load Balancer

Configuring SDS servers

Server	**Task**

sdsm
sdss
	First create the SysV scripts for IBM WebSphere Edge ULB loopback configuration
	`cat > /etc/init.d/edge-ulb-loopback << EOF`
	`#!/bin/sh`
	`#`
	`# Copyright (c) 2014 International Business Machines Corp.`
	`#`
	`# AUTHOR: Valdemar Lemche (DK20563) <valdemar.lemche@dk.ibm.com>`
	`#`
	`# DESCRIPTION:`
	`# SysV init scripts for add/remove of loopback address for IBM WebSphere Edge ULB.`
	`#`
	`### BEGIN INIT INFO`
	`# Provides: edge-ulb-loopback`

```
# Required-Start: \$network
# X-UnitedLinux-Should-Start: network
# Required-Stop:  \$network
# Default-Start:  2 3 5
# Default-Stop:   0 1 6
# Description:    IBM WebSphere Edge ULB loopback configuration
### END INIT INFO

### CHANGE THESE VARIABLES TO FIT YOUR ENVIRONMENT ###
ULB_HOST="ldap.example.com"

### DON'T CHANGE ANYTHING BELOW UNLESS YOU KNOW WHAT YOU'RE DOING ###
if [ ! -x /sbin/ip ]; then
       echo "/sbin/ip is missing";
       exit 1;
fi;

LOG="/var/log/rcedge-ulb-loopback.log"
ULB_HOST_IP=\$(host \$ULB_HOST | sed 's/^.*has address //g')

case "\$1" in
    start)
        echo "\`date\` *** start requested ***" > \${LOG} 2>&1
        printf "Adding Edge Cluter IP to loopback:"
        printf " ip"
        ip -4 addr add \${ULB_HOST_IP}/32 dev lo >> \${LOG} 2>&1
        echo "."
        echo "\`date\` *** start completed ***" >> \${LOG} 2>&1
        ;;
    stop)
        echo "\`date\` *** shutdown requested ***" > \${LOG} 2>&1
        printf "Removing Edge Cluter IP to loopback:"
        printf " ip"
        ip -4 addr delete \${ULB_HOST_IP}/32 dev lo >> \${LOG} 2>&1
        echo "."
        echo "\`date\` *** shutdown completed ***" >> \${LOG} 2>&1
        ;;
    restart)
        \$0 stop
        \$0 start
        ;;
    status)
        /sbin/ip address list lo
        ;;
    *)
        echo "Usage: \$0 {start|stop|restart|status}"
        exit 1
        ;;
esac

exit 0
EOF
```

Make it executable, and run the script

```
chmod +x  /etc/init.d/edge-ulb-loopback
/etc/init.d/edge-ulb-loopback start
```

	And finally define that they should start at system boot `chkconfig edge-ulb-loopback on`

On the Edge servers

Server	Task

sdsm
sdss
	Add the LDAP LB configuration to the dispatcher configuration file
	`cat >> /opt/IBM/WebSphere/Edge/ULB/servers/configurations/dispatcher/default.cfg << EOT`
	`dscontrol set loglevel 1`
	`dscontrol set logsize 10485760`
	
	`dscontrol executor start`
	`dscontrol executor set nfa ${HOSTNAME}.example.com`
	`dscontrol executor nicconfig auto`
	
	`dscontrol cluster add ldap.example.com`
	`dscontrol cluster set ldap.example.com proportions 49 50 1 0`
	
	`dscontrol port add ldap.example.com@389 selectionalgorithm connection`
	`dscontrol port set ldap.example.com@389 staletimeout 6400`
	`dscontrol port set ldap.example.com@389 weightbound 20`
	
	`dscontrol server add ldap.example.com@389@sdsm.example.com`
	`dscontrol server set ldap.example.com@389@sdsm.example.com weight 5`
	`dscontrol server add ldap.example.com@389@sdss.example.com`
	`dscontrol server set ldap.example.com@389@sdss.example.com weight 14`
	
	`dscontrol port add ldap.example.com@636 selectionalgorithm connection`
	`dscontrol port set ldap.example.com@636 staletimeout 6400`
	`dscontrol port set ldap.example.com@636 weightbound 20`
	
	`dscontrol server add ldap.example.com@636@sdsm.example.com`
	`dscontrol server set ldap.example.com@636@sdsm.example.com weight 5`
	`dscontrol server add ldap.example.com@636@sdss.example.com`
	`dscontrol server set ldap.example.com@636@sdss.example.com weight 14`
	
	`dscontrol highavailability heartbeat add edge85-1.example.com edge85-2.example.com`
	`dscontrol highavailability backup add primary=edge85-1.example.com auto 389`
	
	`dscontrol manager start manager.log 10004`
	
	`dscontrol advisor start ldap 389 ldap_389.log`
	`dscontrol advisor interval ldap 389 3`
	`dscontrol advisor timeout ldap 389 30`
	`dscontrol advisor connecttimeout ldap 389 9`
	`dscontrol advisor receivetimeout ldap 389 9`
	`dscontrol advisor retry ldap 389 1`
	
	`dscontrol advisor start ldaps 636 ldaps_636.log`
	`dscontrol advisor interval ldaps 636 3`
	`dscontrol advisor timeout ldaps 636 30`
	`dscontrol advisor connecttimeout ldaps 636 9`
	`dscontrol advisor receivetimeout ldaps 636 9`

	dscontrol advisor retry ldaps 636 1 EOT
	Restart Edge ULB /etc/init.d/edge-ulb restart
	Check if the LB is up and running /opt/IBM/WebSphere/Edge/ULB/bin/dscontrol status /opt/IBM/WebSphere/Edge/ULB/bin/dscontrol server report ldap.example.com@389@ /opt/IBM/WebSphere/Edge/ULB/bin/dscontrol server report ldap.example.com@636@

Testing that it works

Server	Task
sdsm sdss	From sdsm or sdss run idsldapsearch -h ldap.example.com -p 389 -D cn=root -w passw0rd -b "" -s sub "(objectclass=organization)" idsldapsearch -h ldap.example.com -p 636 -Z -D cn=root -w passw0rd -b "" -s sub "(objectclass=organization)"

Install, and configure embedded WAS server, and Web Administration Tool

I personally don't really use the SDS GUI except for the sake of reference, but here's how to install it anyhow.
Also you only need one Web Administration Tool (or two if you need redundancy) installed to manage all TDS's in your infrastructure.

Installing embedded WAS, and Web Administration Tool

Server	Task
sdsm sdss	First set a installation source directory variable: export INSTALL_DIR=/net/files/srv/install/sds/6310/linux64/sdsV6.3.1
sdsm sdss	If you're not installing on one of your SDS servers, then you first need to install the license. echo 1 | $INSTALL_DIR/license/idsLicense rpm -ihv \ $INSTALL_DIR/native/idsldap-license631-6.3.1-0.x86_64.rpm
sdsm sdss	Installing Embedded WAS (cd $INSTALL_DIR/ibm_ewas; ./install.sh -installRoot /opt/ibm/ldap/V6.3.1/appsrv) NB ... I don't know ... perhaps one ought to ensure that the WAS only run as an unprivileged user ...
sdsm sdss	Installing EAR application, and deployment scripts rpm -ihv \ $INSTALL_DIR/native/idsldap-webadmin631-6.3.1-0.x86_64.rpm
sdsm sdss	Deploy Web Administration application cd /opt/ibm/ldap/V6.3.1/idstools/ ./deploy_IDSWebApp
sdsm sdss	Verify Web Administration Tool deployment /opt/ibm/ldap/V6.3.1/idstools/deploy_IDSWebApp -v

Starting, and stopping Web Administration application

Server	Task
sdsm sdss	Starting `/opt/ibm/ldap/V6.3.1/idstools/bin/startWebadminApp`
sdsm sdss	Stopping `/opt/ibm/ldap/V6.3.1/appsrv/profiles/TDSWebAdminProfile/bin/stopServer.sh server1`

Prepare SSL communication with LDAP servers

Server	Task

sdsm
sdss
	Create SSL server keystore
	`/opt/ibm/ldap/V6.3.1/appsrv/java/bin/keytool \`
	`-importkeystore \`
	`-srckeystore /net/main/srv/common-setup/ssl/$HOSTNAME.example.com-cert.p12 \`
	`-srcstoretype pkcs12 \`
	`-srcstorepass passw0rd \`
	`-destkeystore /opt/ibm/ldap/V6.3.1/appsrv/profiles/TDSWebAdminProfile/IDSKeystores/key.jks \`
	`-storepass passw0rd`

sdsm
sdss
	Update renewed certificate (when the time comes)
	`/opt/ibm/ldap/V6.3.1/appsrv/java/bin/keytool \`
	`-import \`
	`-trustcacerts \`
	`-alias $HOSTNAME.example.com \`
	`-keystore /opt/ibm/ldap/V6.3.1/appsrv/profiles/TDSWebAdminProfile/IDSKeystores/key.jks \`
	`-file "/net/main/srv/common-setup/ssl/$HOSTNAME.example.com-cert.pem" \`
	`-storepass passw0rd`

sdsm sdss	Create SSL server truststore We'll just cheat, and copy the trust keystore, so we don't have to create a new. `cp /opt/ibm/ldap/V6.3.1/appsrv/profiles/TDSWebAdminProfile/IDSKeystores/key.jks /opt/ibm/ldap/V6.3.1/appsrv/profiles/TDSWebAdminProfile/IDSKeystores/trust.jks`

sdsm
sdss
	Update renewed certificate (when the time comes)
	`/opt/ibm/ldap/V6.3.1/appsrv/java/bin/keytool \`
	`-delete \`
	`-alias $HOSTNAME.example.com \`
	`-keystore /opt/ibm/ldap/V6.3.1/appsrv/profiles/TDSWebAdminProfile/IDSKeystores/trust.jks \`
	`-storepass passw0rd`

sdsm
sdss
	Just check that only the CA certificate exists in truststore
	`/opt/ibm/ldap/V6.3.1/appsrv/java/bin/keytool \`
	`-list \`

	`-keystore /opt/ibm/ldap/V6.3.1/appsrv/profiles/TDSWebAdminProfile/IDSKeystores/trust.jks \` `-storepass passw0rd`

Managing Web Administration Tool

Create some DSE's that WebAdmin requires to manage the DIT

Server	Task

sdsm
sdss
	Realm template user
	`cat > /tmp/create-template-user.ldif << EOF`
	`cn=template-user,o=sample`
	`javaSerializedData=NOT ASCII`
	`objectClass=top`
	`objectClass=javaContainer`
	`objectClass=javaObject`
	`objectClass=javaSerializedObject`
	`javaClassNames=com.ibm.ldap.admin.usrAdmin.TemplateContainer`
	`javaClassNames=java.lang.Object`
	`javaClassNames=java.io.Serializable`
	`javaClassName=com.ibm.ldap.admin.usrAdmin.TemplateContainer`
	`cn=template-user`
	`EOF`
	`idsldapadd -D cn=root -w passw0rd -i /tmp/create-template-user.ldif`

sdsm
sdss
	Create a SDS realm
	`cat > /tmp/create-realm.ldif << EOF`
	`cn=sample,o=sample`
	`cn=sample`
	`ibm-realmAdminGroup=cn=sample,o=sample`
	`ibm-realmGroupContainer=ou=Groups,o=sample`
	`ibm-realmUserContainer=o=sample`
	`ibm-realmUserSearchFilter=(objectClass=person)`
	`ibm-realmUserTemplate=cn=template-user,o=sample`
	`objectclass=ibm-realm`
	`objectclass=ibm-staticGroup`
	`objectclass=top`
	`EOF`
	`idsldapadd -D cn=root -w passw0rd -i /tmp/create-realm.ldif`

First time configuration

Server	Task

sdsm
sdss
	Change default password for superadmin, and add a SDS instance
	● Open http://sdsm.example.com:12100/IDSWebApp/IDSjsp/Login.jsp, and log in with superadmin:secret.
	● Go to Console Administration > Change console administrator password, and define a new password.
	● Console administration > Manage console properties > SSL key database, and define full path for the keystore, and trust store created above, and click on OK.
	● Go to Console Administration > Manage console servers, and add a server with the
	  ○ Label: sdsm
	  ○ Hostname: sdsm.example.com

	○ Port: 636 ○ Enable SSL encryption: [checked] And click on OK, and add the remaining servers the same way. ● Logout, and now you can log into the sdsm profile, using cn=root, and passw0rd.

Create SysV script for IDS Web Administration Tool

Server	Task

sdsm
sdss
	`/opt/ibm/ldap/V6.3.1/appsrv/bin/wasservice.sh \`
	`-add tdsWebAdministrationTool \`
	`-serverName server1 \`
	`-profilePath /opt/ibm/ldap/V6.3.1/appsrv/profiles/TDSWebAdminProfile`
	
	If you want to disable automatic start of the service at boot
	`chkconfig tdsWebAdministrationTool_was.init off`

SDS performance optimization

SDS has its own tool to analyse its usage, and optimize both SDS, and the DB2 instance. Ref knowledge center chapter regarding The performance tuning tool.

And for more information about tuning please refer to LDAP, and DB2 tuning chapter in the SDS Knowledge Center, and for even more in depth knowledge I recommend the Performance Tuning for IBM Security

But in essence ...

Server	Task

sdsm
sdss
	You should run basic performance tuning after the SDS environment have been completed, but before it's put in production. And afterwards at regular intervals.
	`cat > /usr/local/bin/sds_basic_tuning.sh << EOF`
	`#!/bin/sh`
	`for dsinst in \$(idsilist 2>&1 | grep -v 'Directory server instance(s):');`
	`do`
	`  if [ ! -f /home/\${dsinst}/idsslapd-\${dsinst}/logs/perftune_stat.log ];`
	`then`
	`    su \$dsinst -c "touch`
	`/home/\${dsinst}/idsslapd-\${dsinst}/logs/perftune_stat.log"`
	`  fi`
	`  echo "start SDS basic tuning, and turn on DB2 monitor switches"`
	`  idsperftune -I \$dsinst -B -s -m || exit 1`
	`  echo "wait 5 mins, and 5 seconds to capture one full database snapshot"`
	`  sleep 305`
	`  echo "update DB2 and SDS cache configuration settings, update`
	`perftune_input.conf"`
	`  echo "and turn off DB2 monitor switches"`
	`  idsperftune -I \$dsinst -B -u -s -o || exit 1`
	`done`
	`EOF`
	`chgrp grrdbm01 /usr/local/bin/sds_basic_tuning.sh`
	`chmod 0750 /usr/local/bin/sds_basic_tuning.sh`
	
	`/usr/local/bin/sds_basic_tuning.sh`

sdsm
sdss
	Advanced tuning should only be run after the environment have been running for a while, and run it while typical load have been reached. (And don't run it too often)
	`cat > /usr/local/bin/sds_advanced_tuning.sh << EOF`
	`#!/bin/sh`
	`for dsinst in \$(idsilist 2>&1 | grep -v 'Directory server instance(s):');`
	`do`
	`  if [ ! -f /home/\${dsinst}/idsslapd-\${dsinst}/logs/perftune_stat.log ];`
	`then`
	`    su \$dsinst -c "touch`
	`/home/\${dsinst}/idsslapd-\${dsinst}/logs/perftune_stat.log"`
	`  fi`
	`  echo "start SDS advanced tuning, and turn on DB2 monitor switches"`
	`  idsperftune -I \$dsinst -A -s -m || exit 1`
	`  echo "wait 15 mins, and 5 seconds to capture one full database snapshot"`
	`  sleep 905`
	`  echo "update DB2 and SDS cache configuration settings, update`
	`perftune_input.conf"`
	`  echo "and turn off DB2 monitor switches"`

	`  idsperftune -I \$dsinst -A -s -u -o || exit 1`
	`done`
	`EOF`
	`chgrp grrdbm01 /usr/local/bin/sds_advanced_tuning.sh`
	`chmod 0750 /usr/local/bin/sds_advanced_tuning.sh`
	
	`/usr/local/bin/sds_advanced_tuning.sh`

sdsm sdss	**A comment about group caching!** You should be aware that group cache size is not automatically updated. By default only 25 groups, and only 25,000 members are cached, and you should properly update `TDS_GROUP_CACHE`, and `TDS_GROUP_MEMBER` in **/home/dsrdbm01/idsslapd-dsrdbm01/etc/perftune_input.conf**, and rerun either `/usr/local/bin/sds_basic_tuning.sh`, or `/usr/local/bin/sds_advanced_tuning.sh` Its mostly common to use either the OC groupOfNames, or groupOfUniqueNames, so a quick way to count the groups in the entire directory would be. `idsldapsearch -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -s sub "(|(objectClass=groupOfUniqueNames)(objectClass=groupOfNames))" dn | grep -v ^$ | wc -l` By default **idsperftune** sets the entry cache size to 25% of the total number entries, so I suppose you should set the group cache size accordingly.

sdsm
sdss
	Create a crontab for root to schedule maintenance
	Idsrunstats, and **idsdbmaint** is just SDS commands that runs DB2 runstats, and DB2 reorg.
	
	`cat >> ~/.crontab << EOF`
	`# Run basic tuning Mon, Tue, Thu, and Fri at 8:30 AM`
	`30 8 * * 1,2,4-5 /usr/local/bin/sds_basic_tuning.sh`
	`# Run advanced tuning Wed at 8:30 AM`
	`30 8 * * 4 /usr/local/bin/sds_advanced_tuning.sh`
	`# Run daily runstat, and reorg of databases.`
	`0 0 * * 1-6 idsrunstats -I dsrdbm01 && su - dsrdbm01 -c /usr/local/bin/db2_reorg_all.sh`
	`# Run weekly off line maintenance - DON'T RUN THIS TASK AS THE SAME TIME ON ALL NODES`
	`0 0 * * 7 idsslapd -I dsrdbm01 -k && idsdbmaint -I dsrdbm01 -i && idsdbmaint -I dsrdbm01 -r && idsslapd -I dsrdbm01`
	`EOF`
	`crontab ~/.crontab`
	
	NB: Obviously don’t run these cron jobs at the same time on both replication peers …

	Tuning caches directly If you find that idsperftune doesn't increase the entry, filter, and group cache sufficiently, or you want to set up a external process to adjust cache sizes dynamically, then here's a few tricks. Firstly you can set group cache size as described above, and adding the idsperftune -E <percent of entries to be cached>, and -F <total amount of filters cached> switches in my scripts above.

Alternatively you also retrieve cn=monitor attributes to check cache thresholds:

```
idsldapsearch -h sdsm.example.com -p 636 -Z -s base -b "cn=monitor" -D
"cn=root" -w passw0rd "(objectclass=*)" | grep cache
```

If it for instance prints an output such as the one below, then when you compare `*_cache_size`, `*_cache_current`, and `*_cache_miss`, you can see for instance see the discrepancy between `entry_cache_size`, `entry_cache_current`, and `entry_cache_miss`.

```
filter_cache_size: 0
filter_cache_current: 0
filter_cache_hit: 0
filter_cache_miss: 193770
filter_cache_bypass_limit: 100
entry_cache_size: 8000
entry_cache_current: 8000
entry_cache_hit: 3750668
entry_cache_miss: 67861
group_members_cache_size: 25
group_members_cache_current: 25
group_members_cache_hit: 2
group_members_cache_miss: 319
group_members_cache_bypass: 25000
acl_cache: TRUE
acl_cache_size: 25000
cached_attribute_total_size: 0
cached_attribute_configured_size: 0
cached_attribute_auto_adjust: FALSE
cached_attribute_auto_adjust_time: T000000
cached_attribute_auto_adjust_time_interval: 2
```

You can then update the **ibmslapd.conf** directly through LDAP.

```
cat > /tmp/set-caches.ldif << EOF
dn: cn=Front End, cn=Configuration
changetype: modify
replace: ibm-slapdEntryCacheSize
# maximum number of elements in entry cache
ibm-slapdEntryCacheSize: 100000
-
replace: ibm-slapdFilterCacheSize
# maximum number of elements in search filter cache
ibm-slapdFilterCacheSize: 10000

dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
changetype: modify
replace: ibm-slapdGroupMembersCacheBypassLimit
# maximum number of members in a group that can be cached
ibm-slapdGroupMembersCacheBypassLimit: 100000
-
replace: ibm-slapdGroupMembersCacheSize
# maximum number of groups in cache
ibm-slapdGroupMembersCacheSize: 10000
```

	EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/set-caches.ldif Read the configuration idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op readconfig -scope subtree "cn=Front End, cn=Configuration" idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op readconfig -scope subtree "cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration"
sdsm sdss	**Enable attribute cache** This is properly also a good idea to enable. cat > /tmp/enable-attribute-cache.ldif << EOF dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration changetype: modify replace: ibm-slapdCachedAttributeAutoAdjust # Specifies if autonomic attribute caching is to be enabled. ibm-slapdCachedAttributeAutoAdjust: true - replace: ibm-slapdCachedAttributeAutoAdjustTime # Time to start autonomic attribute cache processing. # Values are in the form of Thhmmss where hh is hours, # mm is minutes and ss is seconds, using a 24 hour clock. ibm-slapdCachedAttributeAutoAdjustTime: T000000 - replace: ibm-slapdCachedAttributeAutoAdjustTimeInterval # Specifies the time interval, in hours, for autonomic attribute # cache processing. ibm-slapdCachedAttributeAutoAdjustTimeInterval: 2 - replace: ibm-slapdCachedAttributeSize # Amount of memory, in kilobytes, that can be used by the attribute # cache. A value of 0 indicates not to use an attribute cache. ibm-slapdCachedAttributeSize: 1024 EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/enable-attribute-cache.ldif Re-read the configuration idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op readconfig -scope subtree "cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration"

Super-duper performance tuning script

The script below somewhat summarizes how I recommend performing an initial performance tuning. The script is meant to be run before going live, but after populating the DIT with all initial DSE's, and should be run during a stress test.
The script can also be run after go-live, but please run it during the most busy time of the day.
The script will ramp up cache sizes, what to be cached, and DB2 performance.

Server	Task
sdsm sdss	cat > /usr/local/bin/sds_initial_performance_tuning.sh << EOF #!/bin/sh

```
if ! which idsldapsearch > /dev/null; then
 echo "*** No idsldapsearch binary found in path"
 exit 1
fi

if ! which idsslapd > /dev/null; then
 echo "*** No idsslapd binary found in path"
 exit 1
fi

USAGE="usage: \$0 -I instance -i <hostname> -p <port> [-Z] -D <cn=root DN>
-w <cn=root PW> -c <cache percent> [-h]"

while getopts :I:i:p:ZD:w:c:h opt; do
 case \$opt in
  I)
   if [[ \$OPTARG = -* ]]; then
    ((OPTIND--))
    continue
   fi
   INSTANCE=\$OPTARG
   ;;
  i)
   if [[ \$OPTARG = -* ]]; then
    ((OPTIND--))
    continue
   fi
   HOST=\$OPTARG
   ;;
  p)
   if [[ \$OPTARG = -* ]]; then
    ((OPTIND--))
    continue
   fi
   PORT=\$OPTARG
   ;;
  Z)
   SSL="-Z"
   ;;
  D)
   if [[ \$OPTARG = -* ]]; then
    ((OPTIND--))
    continue
   fi
   BIND_DN=\$OPTARG
   ;;
  w)
   if [[ \$OPTARG = -* ]]; then
    ((OPTIND--))
    continue
   fi
   BIND_PW=\$OPTARG
   ;;
  c)
   if [[ \$OPTARG = -* ]]; then
```

```
    ((OPTIND--))
    continue
   fi
   CACHE_PERCENT=\$OPTARG
   ;;
  h)
   echo \$USAGE
   exit 1
   ;;
 esac
done

if [ "\$INSTANCE" = "" ]; then echo; echo "missing instance!"; echo; echo
\$USAGE; exit 1; fi
if [ "\$HOST" = "" ]; then echo; echo "missing hostname!"; echo; echo
\$USAGE; exit 1; fi
if [ "\$PORT" = "" ]; then echo; echo "missing port!"; echo; echo \$USAGE;
exit 1; fi
if [ "\$BIND_DN" = "" ]; then echo "missing bind DN!"; echo; echo \$USAGE;
exit 1; fi
if [ "\$BIND_PW" = "" ]; then echo "missing bind PW!"; echo; echo \$USAGE;
exit 1; fi
if [ "\$CACHE_PERCENT" = "" ]; then echo "missing cache percent!"; echo;
echo \$USAGE; exit 1; fi

IDSLDAPSEARCH_CONNECTION_SWITCHES="-h \$HOST -p \$PORT \$SSL -D \$BIND_DN
-w \$BIND_PW"

TOTAL_ENTRIES=\$(idsldapsearch \$IDSLDAPSEARCH_CONNECTION_SWITCHES -s sub
"(objectClass=*)" dn | grep -v ^\$ | wc -l)
echo "*** total entries in instance, \$INSTANCE, counted to
\$TOTAL_ENTRIES"

TOTAL_GROUPS=\$(idsldapsearch \$IDSLDAPSEARCH_CONNECTION_SWITCHES -s sub
"(|(objectClass=groupOfUniqueNames)(objectClass=groupOfNames))" dn | grep
-v ^\$ | wc -l)
echo "*** total groups in instance, \$INSTANCE, counted to \$TOTAL_GROUPS"

ENTRY_CACHE_SIZE=\$(awk "BEGIN{percent = \$CACHE_PERCENT/100;
entry_cache_size = \$TOTAL_ENTRIES * percent; print int(entry_cache_size +
0.5)}")
echo "*** entry cache size calculated to \$ENTRY_CACHE_SIZE"

GROUP_CACHE_SIZE=\$(awk "BEGIN{percent = \$CACHE_PERCENT/100;
group_cache_size = \$TOTAL_GROUPS * percent; print int(group_cache_size +
0.5)}")
echo "*** group cache size calculated to \$GROUP_CACHE_SIZE"

if [ ! -f /home/\$INSTANCE/idsslapd-\$INSTANCE/logs/perftune_stat.log ];
then
  echo "*** /home/\$INSTANCE/idsslapd-\$INSTANCE/logs/perftune_stat.log
doesn't exist - creating empty file"
  su - \$INSTANCE -c "touch
/home/\$INSTANCE/idsslapd-\$INSTANCE/logs/perftune_stat.log"
fi
```

```
echo "*** setting TDS_GROUP_CACHE in
/home/\$INSTANCE/idsslapd-\$INSTANCE/etc/perftune_input.conf to
\$GROUP_CACHE_SIZE"
if grep ^TDS_GROUP_CACHE
/home/\$INSTANCE/idsslapd-\$INSTANCE/etc/perftune_input.conf > /dev/null;
then
  sed -i "s/TDS_GROUP_CACHE=.*/TDS_GROUP_CACHE=\$GROUP_CACHE_SIZE/"
/home/\$INSTANCE/idsslapd-\$INSTANCE/etc/perftune_input.conf
else
  echo TDS_GROUP_CACHE=\$GROUP_CACHE_SIZE >>
/home/\$INSTANCE/idsslapd-\$INSTANCE/etc/perftune_input.conf
fi

cat > /tmp/set-caches.ldif << EOF
dn: cn=Front End, cn=Configuration
changetype: modify
replace: ibm-slapdEntryCacheSize
# maximum number of elements in entry cache
ibm-slapdEntryCacheSize: \$ENTRY_CACHE_SIZE
-
replace: ibm-slapdFilterCacheSize
# maximum number of elements in search filter cache
ibm-slapdFilterCacheSize: \$ENTRY_CACHE_SIZE
-
replace: ibm-slapdFilterCacheBypassLimit
# don't impose a bypass limit on filter searches
ibm-slapdFilterCacheBypassLimit: 0

dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
changetype: modify
replace: ibm-slapdGroupMembersCacheBypassLimit
# maximum number of members in a group that can be cached
ibm-slapdGroupMembersCacheBypassLimit: 25000
-
replace: ibm-slapdGroupMembersCacheSize
# maximum number of groups in cache
ibm-slapdGroupMembersCacheSize: \$GROUP_CACHE_SIZE
EOF

echo "*** setting entry cache size to \$ENTRY_CACHE_SIZE, filter cache size
to \$ENTRY_CACHE_SIZE, and group cache size to \$GROUP_CACHE_SIZE"
idsldapmodify \$IDSLDAPSEARCH_CONNECTION_SWITCHES -i /tmp/set-caches.ldif

echo "*** re-read the configuration from ibmslapd.conf"
idsldapexop \$IDSLDAPSEARCH_CONNECTION_SWITCHES -op readconfig -scope
subtree "cn=Front End, cn=Configuration"
idsldapexop \$IDSLDAPSEARCH_CONNECTION_SWITCHES -op readconfig -scope
subtree "cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration"

cat > /tmp/enable-attribute-cache.ldif << EOF
dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
changetype: modify
replace: ibm-slapdCachedAttributeAutoAdjust
```

```
# Specifies if autonomic attribute caching is to be enabled.
ibm-slapdCachedAttributeAutoAdjust: true
-
replace: ibm-slapdCachedAttributeAutoAdjustTime
# Time to start autonomic attribute cache processing.
# Values are in the form of Thhmmss where hh is hours,
# mm is minutes and ss is seconds, using a 24 hour clock.
ibm-slapdCachedAttributeAutoAdjustTime: T000000
-
replace: ibm-slapdCachedAttributeAutoAdjustTimeInterval
# Specifies the time interval, in hours, for autonomic attribute
# cache processing.
ibm-slapdCachedAttributeAutoAdjustTimeInterval: 2
-
replace: ibm-slapdCachedAttributeSize
# Amount of memory, in kilobytes, that can be used by the attribute
# cache.  A value of 0 indicates not to use an attribute cache.
ibm-slapdCachedAttributeSize: 1024
EOF

echo "*** enabling attribute caching"
idsldapmodify \$IDSLDAPSEARCH_CONNECTION_SWITCHES -i
/tmp/enable-attribute-cache.ldif

echo "*** re-read the configuration from ibmslapd.conf"
idsldapexop \$IDSLDAPSEARCH_CONNECTION_SWITCHES -op readconfig -scope
subtree "cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration"

echo "*** start SDS advanced tuning, and turn on DB2 monitor switches"
idsperftune -A -E \$CACHE_PERCENT -F \$ENTRY_CACHE_SIZE -I \$INSTANCE -m -p
\$PORT -s || exit 1

echo "*** wait 15 mins, and 5 seconds to capture one full database
snapshot"
sleep 905

echo "*** update DB2 and SDS cache configuration settings, update
perftune_input.conf"
echo "and turn off DB2 monitor switches"
idsperftune -A -E \$CACHE_PERCENT -F \$ENTRY_CACHE_SIZE -I \$INSTANCE -o -p
\$PORT -s -u || exit 1

echo -n "!!! restart ibmslapd, and DB2 for new parameters to take place?
(Y/n) "
read ANSWER
if [ "\\$ANSWER" == "n" ]; then
  echo "OK, but remember to restart both at a later point - ITS CRUCIAL!"
  exit 0
fi

idsslapd -I \$INSTANCE -k && \
su - \$INSTANCE -c "db2 force application all && db2stop force && db2start"
&& \
idsslapd -I \$INSTANCE
EOF
```

	`chmod +x /usr/local/bin/sds_initial_performance_tuning.sh`
	
	Run the script against the instance dsrdbm01
	`sds_initial_performance_tuning.sh -I dsrdbm01 -i $HOSTNAME.example.com -p 636 -Z -D cn=root -w passw0rd -c 50`

Backup, and restore

You have two backup methods, *basic*, and *enhanced*. AFAIK the only real advantage of enhanced is that you can initiate the backup remotely using ldapexop (LDAP EXtended OPerations) from the SDS client tools, and that it backs up the schema. But since it doesn't back up all required files such as your keystores, the directory encryption key, and instance list, then you still have to backup those files manually. So here's two scripts - 1st to *configure* backup of all instances, 2nd to *perform* backups of all instances.

For more information about backup, and restore please refer to the SDS Knowledge Center Backing up and restoring directory server, and IBM Tivoli® Directory Server backup using IBM Tivoli® Storage Manager.

Setup backup

Server	Task

sdsm
sdss
	Script to configure backup of all instances
	`cat > /usr/local/bin/sds_initialize_backup_all.sh << EOF`
	`#!/bin/sh`
	
	`BACKUP_DIR=/net/files/srv/backup/\$HOSTNAME`
	
	`for dsinst in \`idsilist 2>&1 | grep -v 'Directory server instance(s):'\`;`
	`do`
	`  echo -n "*** WARNING! This script will stop each instance during this initialization of backup configuration - Continue? (Y/n)"`
	`  read ANSWER`
	`  if [ "\$ANSWER" == "n" ]; then`
	`  exit 1`
	`  fi`
	`  if [ -d \$BACKUP_DIR/\$dsinst ]; then`
	`  if ls \$BACKUP_DIR/\$dsinst\/\* > /dev/null 2>&1; then`
	`  echo "*** \$BACKUP_DIR/\$dsinst exists, and isn't empty - delete existing, and re-create? (Y/N)"`
	`  read ANSWER`
	`  if [ "\$ANSWER" \!= "n" ]; then`
	`  echo "*** deleting \$BACKUP_DIR/\$dsinst"`
	`  rm -fr \$BACKUP_DIR/\$dsinst`
	`  else`
	`  exit 1`
	`  fi`
	`  fi`
	`  fi`
	`  echo "*** creating backup directory"`
	`  mkdir -p \$BACKUP_DIR/\$dsinst || exit 1`
	`  echo "*** Stopping idsslapd for \$dsinst"`
	`  idsslapd -k || exit 1`
	`  echo "*** creating initial backup configuration for \$dsinst"`
	`  idsdbback -I \$dsinst -k \$BACKUP_DIR/\$dsinst -n || exit 1`
	`  echo "*** creating online backup configuration for \$dsinst"`
	`  idsdbback -I \$dsinst -k \$BACKUP_DIR/\$dsinst -u -n || exit 1`
	`  echo "*** starting idsslap for \$dsinst"`
	`  idsslapd || exit 1`
	`done`
	`EOF`
	`chmod +x /usr/local/bin/sds_initialize_backup_all.sh`

sdsm sdss	Now run it for the first time on our default instance (WARNING this script will restart the instance after configuration): `/usr/local/bin/sds_initialize_backup_all.sh`

sdsm
sdss
	Create a script to perform backups
	`cat > /usr/local/bin/sds_backup_all.sh << EOF`
	`#!/bin/sh`
	
	`BACKUP_DIR=/net/files/srv/backup/\$HOSTNAME`
	
	`for dsinst in \`idsilist 2>&1 | grep -v 'Directory server instance(s):'\`;`
	`do`
	`  if [ ! -d \$BACKUP_DIR/\$dsinst ]; then`
	`  echo "*** \$BACKUP_DIR/\$dsinst doesn't exist - run`
	`/usr/local/bin/sds_initialize_backup_all.sh to initialize backup`
	`configuration for instance"`
	`  exit 1`
	`  fi`
	`  echo "*** starting online backup for \$dsinst"`
	`  idsdbback -I \$dsinst -k \$BACKUP_DIR/\$dsinst -u -n`
	`  echo "*** creating archive file of instance"`
	`  tar --create --file=\${BACKUP_DIR}/\${dsinst}.tar`
	`--directory=/home/\${dsinst}/idsslapd-\${dsinst} adworkdir db2instance etc`
	`idsprofile logs tmp userprofile workdir`
	`done`
	`cp -fr /opt/ibm/ldap/idsinstinfo \$BACKUP_DIR`
	`EOF`
	`chmod +x /usr/local/bin/sds_backup_all.sh`

sdsm
sdss
	Configure cron to run backup to run every day at 6 AM
	`cat >> ~/.crontab << EOT`
	`0 6 * * * /usr/local/bin/sds_backup_all.sh`
	`EOT`
	`crontab ~/.crontab`
	
	Make sure that your file backup completes before 6 AM.

How to restore

Server	Task

sdsm
sdss
	Stop ibmslapd if running
	`idsslapd -I dsrdbm01 -k`
	
	Extract the etc directory from the tarball, so we get the correct configuration files, certificates, keys etc before restoring the database
	`tar --extract --file=/net/files/srv/backup/${HOSTNAME}/dsrdbm01.tar`
	`--directory=/home/dsrdbm01/idsslapd-dsrdbm01 etc`
	
	Restore the database
	`idsdbrestore -I dsrdbm01 -r -k /net/files/srv/backup/${HOSTNAME}/dsrdbm01`
	`-n`
	
	And finally start ibmslapd again
	`idsslapd -I dsrdbm01`

Other tips, tricks, and best practices

Enable log rotation

We'll just use the default log rotation tool in Linux

Server	Task

sdsm
sdss
	cat > /etc/logrotate.d/dsrdbm01 << EOF
	/home/dsrdbm01/idsslapd-dsrdbm01/logs/adminaudit.log
	/home/dsrdbm01/idsslapd-dsrdbm01/logs/bulkload.log
	/home/dsrdbm01/idsslapd-dsrdbm01/logs/db2cli.log
	/home/dsrdbm01/idsslapd-dsrdbm01/logs/db2clicmds.log
	/home/dsrdbm01/idsslapd-dsrdbm01/logs/ibmdiradm.log
	/home/dsrdbm01/idsslapd-dsrdbm01/logs/ibmslapd.log
	/home/dsrdbm01/idsslapd-dsrdbm01/logs/idstools.log
	/home/dsrdbm01/idsslapd-dsrdbm01/logs/lostandfound.log {
	compress
	daily
	dateext
	maxage 365
	rotate 365
	missingok
	create 640 dsrdbm01 grrdbm01
	su dsrdbm01 grrdbm01
	}
	
	EOF

Multi-threaded replication

If you need to replicate large amount of entries, then switching to a multi-thread replication method will improve performance significantly over the default single-threaded method - but it will also make the replication process less error tolerant, as replication will freeze if *any* problems occurs, and an operator needs to force replication to start again (see below).

Server	Task
sdsm	Enabling multi-threaded replication cat > /tmp/enable-multithreaded-replication.ldif << EOF # Create a replication agreements to the servers in the replication group dn: cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample changetype: modify add: ibm-replicaconsumerconnections ibm-replicaconsumerconnections: 2 - add: ibm-replicamethod ibm-replicamethod: 2 dn: cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,O=SAMPLE changetype: modify add: ibm-replicaconsumerconnections ibm-replicaconsumerconnections: 2 - add: ibm-replicamethod ibm-replicamethod: 2 EOF

	idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/enable-multithreaded-replication.ldif
sdsm	Propagate replication topology to sdss idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op repltopology -rc o=sample
sdsm	Restart servers You have to restart the servers to make the effect take place. idsslapd -k idsslapd
sdsm	Forcing replication Multi-threaded replication is happening asynchronously, so if you want to force a replication to start, then run the following command. idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op controlrepl -action replnow -rc o=sample

Disable anonymous binds

Server	Task
sdsm	If you require that any query must be authenticated, then create the following configuration change. cat > /tmp/restrict-anon-access.ldif << EOF dn: cn=Connection Management,cn=Front End, cn=Configuration changetype: modify replace: ibm-slapdAllowAnon ibm-slapdAllowAnon: FALSE EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/restrict-anon-access.ldif
sdsm sdss	Restart (both) servers It isn't sufficient to just reread the configuration, but you have to restart the daemons idsslapd -k idsslapd

Allow binds with only username

A DN is a long string to remember when binding. So you can chose an attribute to which to bind with instead. So setup SDS so that users can use their uid value as bind DN rather than their entire DN.

Server	Task
sdsm	cat >> /tmp/setBindAttr.ldif << EOF # Configuring an attribute with a unique value for bind operations dn: cn=Configuration changetype: modify add: ibm-slapdUniqueAttrForBindWithValue ibm-slapdUniqueAttrForBindWithValue: uid EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/setBindAttr.ldif
sdsm	And if you're using a replication partner you need to do add this to the configuration as well there

	`idsldapmodify -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/setBindAttr.ldif`

sdsm
sdss
	Restart (both) servers
	`idsslapd -k`
	`idsslapd`

sdsm	Let's test. `idsldapsearch -h sdsm.example.com -p 636 -Z -D` **`root`** `-w passw0rd -b "" -s sub "(objectclass=organization)"`

Group membership attribute

Typically you define members of a group by creating a group DSE, and adding `uniqueMember`, or `member` attributes, depending on which objectclass the DSE is using, i.e. `groupOfUniqueNames` or `groupOfNames` - *You can use both with the same DSE, but typically you only use one of them*.
The values of member attributes can be the DNs to any DSE, but typically you either define a DN to a user or a group DSE.

Now if you have a user DSE, which is a member of a group DSE, and that group DSE is a member of another group DSE, which is then a member of another group DSE etc-etc, then if an application needs to look up all the groups that a user DSE is a member of, then it must intelligently follow all references to groups, that the user DSE is a member of, and all references to groups that those groups are members of. This means that an application will often need to perform perhaps a 100 queries to look up a single user's group memberships!

This is a performance killer, which is a design problem of LDAP. Except for Active Directory most LDAP server have various solutions on how to solve this, and here is how to solve it in SDS.

How to configure activate

So the solution is to configure SDS to index group memberships automatically, so that the group memberships appears as the attribute `ibm-allGroups` on the user DSE.

Server	**Task**

sdsm	So create some user DSEs - **userA**, and **userB**
	`cat >> /tmp/createUserAandB.ldif << EOF`
	`cn: userA`
	`objectclass: top`
	`objectclass: person`
	`objectclass: organizationalPerson`
	`sn: A`
	
	`dn: cn=userB,ou=Austin,O=SAMPLE`
	`cn: userB`
	`objectclass: top`
	`objectclass: person`
	`objectclass: organizationalPerson`
	`sn: B`
	`EOF`
	`idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/createUserAandB.ldif`

sdsm	Then create a group DSE, **groupA**, that **userA** is a direct member of: `cat >> /tmp/createGroupA.ldif << EOF`

	dn: cn=groupA,ou=Groups,O=SAMPLE cn: groupA objectclass: top objectclass: groupOfNames member: cn=userA,ou=Austin,O=SAMPLE EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/createGroupA.ldif
sdsm	Then finally create a group DSE, **groupB**, that both **groupA**, and **userB** is a direct member of cat >> /tmp/createGroupB.ldif << EOF dn: cn=groupB,ou=Groups,O=SAMPLE cn: groupB objectclass: top objectclass: groupOfNames objectclass: ibm-nestedGroup ibm-memberGroup: cn=groupA,ou=Groups,O=SAMPLE member: cn=userB,ou=Austin,O=SAMPLE idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/createGroupB.ldif

Notice how groupB have the additional lines?
`objectclass=ibm-nestedGroup`
`ibm-memberGroup: cn=groupA,ou=Groups,O=SAMPLE`

These tell SDS to index *direct* members as well as *indirect* members thus making **userA** an indirect member of **groupB** through **groupA**. The OC `ibm-nestedGroup` allows the use of the attribute, `ibm-memberGroup`, where the values are other groups, that are members of this group.

Server	**Task**
sdsm	The following query of **userA** now shows with the attribute `ibm-allGroups`, that **userA** is both member of **groupA**, and **groupB**: idsldapsearch -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -b o=sample "(cn=userA)" ibm-allGroups Output: cn=userA,ou=Austin,O=SAMPLE ibm-allGroups=cn=groupA,ou=Groups,O=SAMPLE ibm-allGroups=cn=groupB,ou=Groups,O=SAMPLE

So now the application only needs to make a single query to list all the groups that a user is a member of.

This indexing also works the other way - i.e. for instance for a group DSE the `member` or `uniqueMember` attribute only shows the direct members of the group. But using `ibm-nestedGroup` OC if you want to see all indirect members of a group, then you can ask the query to return the `ibm-allMembers` attribute.

Server	**Task**
sdsm	idsldapsearch -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -b o=sample "(cn=groupB)" ibm-allMembers

	Output:
	`cn=groupB,ou=Groups,O=SAMPLE`
	`ibm-allMembers=cn=userB,ou=Austin,O=SAMPLE`
	`ibm-allMembers=cn=userA,ou=Austin,O=SAMPLE`

You might argue that its cumbersome that you have to use one attribute (`member`, `uniqueMember`) to add *user* DSEs as members of a group, and another attribute (`ibm-memberGroup`) to add *group* DSEs as members of a group, but thats how its designed. For any application provisioning the DSE's in the directory it should be an easy task to configure it to distinguish between the types of DSE's, and for applications using the directory as clients, then its easy to configure, as shown in the next section.

Configuring an application to use group membership attribute.

I'm going to use WebSphere Application Server as an example, because its configured very elegantly in here.

- Log into your admin console
- Go to Global security > Federated repositories > <federated LDAP repository> > Group attribute definition
- Set value for "*Name of group membership attribute*" to `ibm-allGroups`
- Set the radio button to "*Nested - Contains direct members and members nested within subgroups of this group*"
- Click on "**Apply**"
- Click on "**Member attributes**"
- Click on "**New**"
- Set the value for "*Name of member attribute*" to `ibm-allMembers`
- Leave "Object class" empty
- Set the radio button to "*Nested - Contains direct members and members nested within subgroups of this group*"
- Click on OK
- Click on Save
- Restart WAS cell

Now you have configure WAS VMM to only look up a user's groupmemberships using one query, and one query to lookup the members of a group.

The corresponding python commands for wsadmin are:

```
AdminTask.setIdMgrLDAPGroupConfig('[-id <federated LDAP repository> -name ibm-allGroups
-scope nested]')
AdminTask.addIdMgrLDAPGroupMemberAttr('[-id <federated LDAP repository> -name
ibm-allMembers -objectClass -scope nested]')
AdminConfig.save()
```

Setting password policies

SDS isn't able to set very complicated policies out of box (see User password policy settings). However in Appendix J, you can read how to install an enhanced password policy plugin. But let me just take you through the basics of the out of box features.

Enable password policies

Server	Task
sdsm	First you must enable password policies globally (You can also create specific policies that you associate with groups, but I'll show that later) `cat > /tmp/enablePwdPolicy.ldif << EOF`

	`# Enable password policies`
	`dn: cn=pwdpolicy,CN=IBMPOLICIES`
	`changetype: modify`
	`replace: ibm-pwdPolicy`
	`ibm-pwdPolicy: TRUE`
	`EOF`
	`idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/enablePwdPolicy.ldif`

Global policy

So when thats done, lets play with the policies.

- Let set a minimum length to 8 characters
- It will expire after 5184000 secs (60 days)
- Users can reuse a password after 6 changes
- Password must contain at least 2 alphabetical characters
- Password must contain at least 1 "special" character
- Password only allows two consecutive occurrences of the same character

Server	Task

sdsm	`cat > /tmp/setPwdPolicy.ldif << EOF`
	`# Enable password policies, password lockout, and`
	`dn: cn=pwdpolicy,CN=IBMPOLICIES`
	`changetype: modify`
	`replace: pwdMinLength`
	`pwdMinLength: 8`
	`-`
	`replace: pwdMaxAge`
	`pwdMaxAge: 5184000`
	`-`
	`replace: pwdInHistory`
	`pwdInHistory: 6`
	`-`
	`replace: passwordMinAlphaChars`
	`passwordMinAlphaChars: 2`
	`-`
	`replace: passwordMinOtherChars`
	`passwordMinOtherChars: 1`
	`-`
	`replace: passwordMaxConsecutiveRepeatedChars`
	`passwordMaxConsecutiveRepeatedChars: 2`
	`EOF`
	`idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/setPwdPolicy.ldif`

Prevent brute-force attacks

Let see if we can't prevent brute-force attacks on users.

This LDIF will ...

- allow 3 failed authentication attempts
- lock a user's password when exceeding pwdMaxFailure value
- release the lock after 600 seconds

Server	Task
sdsm	`cat > /tmp/preventBruteForceAttacks.ldif << EOF`

	dn: cn=pwdpolicy,CN=IBMPOLICIES changetype: modify replace: pwdMaxFailure pwdMaxFailure: 3 - replace: pwdLockout pwdLockout: TRUE - replace: pwdLockoutDuration pwdLockoutDuration: 600 - EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/preventBruteForceAttacks.ldif

Group bound policies

Server	Task
sdsm	Lets create another policy for just for service accounts where we won't expire passwords, but will require stronger ones. cat > /tmp/createServiceAccountPwdPolicy.ldif << EOF dn: cn=serviceAccounts,cn=ibmpolicies objectclass: container objectclass: ibm-pwdPolicyExt objectclass: pwdPolicy objectclass: top cn: serviceAccounts ibm-pwdPolicy: true passwordMaxConsecutiveRepeatedChars: 0 passwordMaxRepeatedChars: 1 passwordMinAlphaChars: 2 passwordMinDiffChars: 2 passwordMinOtherChars: 3 pwdAttribute: userPassword pwdInHistory: 30 pwdMaxAge: 0 pwdMaxFailure: 0 pwdMinLength: 12 EOF idsldapadd -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/createServiceAccountPwdPolicy.ldif
sdsm	Create a service accounts group cat > /tmp/createServiceAccountGroup.ldif << EOF dn: cn=serviceAccounts,ou=groups,O=EXAMPLE objectClass: groupOfNames objectClass: top cn: serviceAccounts member: cn=ldapbind,ou=DSA,O=EXAMPLE EOF idsldapadd -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/createServiceAccountGroup.ldif
sdsm	Add the password policy above to the group cat > /tmp/addPolicyToGroup.ldif << EOF dn: cn=serviceAccounts,ou=groups,O=EXAMPLE

	changetype: modify add: ibm-pwdGroupPolicyDN ibm-pwdGroupPolicyDN: cn=serviceAccounts,cn=ibmpolicies EOF idsldapmodify -k -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/addPolicyToGroup.ldif

Restricting access using "Access Control Lists"

Server	Task
sdsm	By default SDS allow anonymous access to read all normal, system, and restricted attributes, but lets say that you only want to authenticated users to be able to read any data. cat >> /tmp/acl-restrict-read-to-authenticated-users.ldif << EOF dn: o=sample changetype: modify add: aclEntry aclentry: group: cn=Authenticated:normal:rsc:system:rsc:restricted:rsc EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/acl-restrict-read-to-authenticated-users.ldif
sdsm	Now you find out that you actually only want to allow a specific group to be able to read data. cat >> /tmp/acl-restrict-read-to-group.ldif << EOF # Create LDAPSupporters group, and make Henry the owner, and Kyle a member dn: cn=LDAPSupporters ,ou=Groups,o=sample changetype: add objectclass: top objectclass: groupOfNames description: LDAP Supporters cn: LDAPSupporters owner: cn=Henry Nguyen,ou=Austin,o=sample member: cn=Kyle Nguyen,ou=Austin,o=sample # Replace all current ACLs with one new that defines that only LDAPAdmins can read data dn: o=sample changetype: modify replace: aclEntry aclentry: group:CN=LDAPSUPPORTERS,OU=GROUPS,O=SAMPLE:normal:rsc:system:rsc:restricted:rsc EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/acl-restrict-read-to-group.ldif
sdsm	Well you might still want to permit users to read their own data. cat >> /tmp/acl-allow-users-to-read-own-data.ldif << EOF # Allow an authenticated user to read the user's own data dn: o=sample changetype: add

	aclentry: access-id:CN=THIS:normal:rsc:system:rsc:restricted:rsc EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/acl-allow-users-to-read-own-data.ldif
sdsm	And perhaps it would be cool if users could also change their own password. cat >> /tmp/acl-allow-users-to-change-password.ldif << EOF # Replace current CN=THIS with an ACL that also include write access to the userPassword attribute. dn: o=sample changetype: modify delete: aclEntry aclentry: access-id:CN=THIS:normal:rsc:system:rsc:restricted:rsc - add: aclEntry aclentry: access-id:CN=THIS:normal:rsc:system:rsc:restricted:rsc:at.userPassword:rwsc EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/acl-allow-users-to-change-password.ldif
sdsm	And finally lets also create a administrative group that is allowed to manage the whole o=sample suffix. cat >> /tmp/acl-create-administrative-group.ldif << EOF # Create LDAPAdmins group, and make Henry the owner, and member dn: cn=LDAPAdmins,ou=Groups,o=sample changetype: add objectclass: top objectclass: groupOfNames description: LDAP Administrators cn: LDAPAdmins owner: cn=Henry Nguyen,ou=Austin,o=sample member: cn=Henry Nguyen,ou=Austin,o=sample # Replace all current ACLs with a new that defines only LDAPAdmins can read data dn: o=sample changetype: modify add: aclEntry aclentry: group:CN=LDAPADMINS,OU=GROUPS,O=SAMPLE:system:rsc:restricted:rwsc:object:ad:normal:rwsc:sensitive:rwsc:critical:rwsc EOF idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/acl-create-administrative-group.ldif

Enable audit logging

Server	Task
sdsm	This will enable the audit log **/home/dsrdbm01/idsslapd-dsrdbm01/logs/audit.log**, as well as enable a few more triggers cat > /tmp/enableAuditLog.ldif << EOF dn: cn=Audit, cn=Log Management, cn=Configuration

	changetype: modify replace: ibm-audit ibm-audit: true - replace: ibm-auditAdd ibm-auditAdd: true - replace: ibm-auditDelete ibm-auditDelete: true - replace: ibm-auditExtOp ibm-auditExtOp: true - replace: ibm-auditExtOPEvent ibm-auditExtOPEvent: true - replace: ibm-auditGroupsOnGroupControl ibm-auditGroupsOnGroupControl: true - replace: ibm-auditModify ibm-auditModify: true - replace: ibm-auditModifyDN ibm-auditModifyDN: true EOF /opt/ibm/ldap/V6.3.1/bin/idsldapmodify -k -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/enableAuditLog.ldif
sdsm	/opt/ibm/ldap/V6.3.1/bin/idsldapmodify -k -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/enableAuditLog.ldif
sdsm	Re-read the instance configuration (instead of restarting) idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op readconfig -scope subtree "cn=Audit, cn=Log Management, cn=Configuration"
sdsm	idsldapexop -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -op readconfig -scope subtree "cn=Audit, cn=Log Management, cn=Configuration"

sdsm
sdss
	cat >> /etc/logrotate.d/dsrdbm01 << EOT
	/home/dsrdbm01/idsslapd-dsrdbm01/logs/audit.log {
	compress
	daily
	dateext
	maxage 365
	rotate 365
	missingok
	create 640 dsrdbm01 grrdbm01
	su dsrdbm01 grrdbm01
	}
	
	EOT

Enable Global Administration Group

If you don't want to only use cn=root for administrative purposes, then you can enable an global administration group, and add any DSE - but typically dedicated administrative accounts - to it. The global administrator group is created directly under cn=ibmpolicies, and its members will all privileges but to the audit log, and or any entries under cn=configuration - aka the server configuration.

Ref: Adding a user as a member of the Global Administrative Group

Server	Task

sdsm	Create an administrative account.
	`cat > /tmp/addAdministrationUser.ldif << EOF`
	`dn: cn=ldapGlobalAdmin,cn=users,cn=ibmpolicies`
	`changetype: add`
	`userPassword: passw0rd`
	`objectClass: accessRole`
	`objectClass: simpleAuthObject`
	`objectClass: top`
	`cn: ldapGlobalAdmin`
	`EOF`
	`/opt/ibm/ldap/V6.3.1/bin/idsldapmodify -k -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/addAdministrationUsers.ldif`

sdsm	Then create the ibm-globalAdminGroup group.
	`cat >> /tmp/addIbm-globalAdminGroup.ldif << EOF`
	`dn: globalGroupName=GlobalAdminGroup,CN=IBMPOLICIES`
	`changetype: add`
	`objectClass: ibm-globalAdminGroup`
	`objectClass: top`
	`globalGroupName: GlobalAdminGroup`
	`member: cn=ldapGlobalAdmin,cn=users,cn=ibmpolicies`
	`EOF`
	`idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/addIbm-globalAdminGroup.ldif`

Extending schema

Server	Task

sdsm	### Adding attribute and objectclass to schema
	You always have to create the attributes before creating the objectclass that references them.
	`cat > /tmp/extendSchema.ldif << EOF`
	`# create attribute favoriteColor`
	`dn: cn=schema`
	`changetype: modify`
	`add: attributetypes`
	`attributetypes: ( favoriteColor-oid NAME ( 'favoriteColor' )`
	`DESC 'Which color?'`
	`EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15`
	`{200} USAGE userApplications )`
	`-`
	`add: ibmattributetypes`
	`ibmattributetypes: ( favoriteColor-oid DBNAME ( 'favoriteColorTable' 'favoriteColorColumn' ) ACCESS-CLASS normal LENGTH 200 )`
	
	`# create attribute preferredSwallow`
	`dn: cn=schema`
	`changetype: modify`
	`add: attributetypes`
	`attributetypes: ( preferredSwallow-oid NAME ( 'preferredSwallow' )`
	`DESC 'Which swallow - An African or European?'`

	EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 {200} USAGE userApplications) - add: ibmattributetypes ibmattributetypes: (preferredSwallow-oid DBNAME ('preferredSwallowTable' 'preferredSwallowColumn') ACCESS-CLASS normal LENGTH 200) # create an auxiliary objectclass that allows favoriteColor, and preferredSwallow dn: cn=Schema changetype: modify add: objectclasses objectclasses: (examplePerson-oid NAME 'examplePerson' DESC 'objectClass for typical Example Ltd. employees' SUP top AUXILIARY MUST (objectClass \$ employeeNumber) MAY (favoriteColor \$ preferredSwallow \$ cn)) EOF idsldapadd -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/extendSchema.ldif
sdsm	**Check that the schema have been extended** idsldapsearch -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -b cn=schema -s base "(objectclass=*)" objectClasses | grep examplePerson idsldapsearch -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -b cn=schema -s base "(objectclass=*)" attributeTypes | grep favoriteColor idsldapsearch -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -b cn=schema -s base "(objectclass=*)" attributeTypes | grep preferredSwallow
sdsm	Check that the schema have been replicated if replication have been configure idsldapsearch -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -b cn=schema -s base "(objectclass=*)" objectClasses | grep examplePerson idsldapsearch -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -b cn=schema -s base "(objectclass=*)" attributeTypes | grep favoriteColor idsldapsearch -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -b cn=schema -s base "(objectclass=*)" attributeTypes | grep preferredSwallow
sdsm	**Deleting attributes, and objectclass from schema** The order of operations is also important here - you have to delete the objectclass referencing the attributes before deleting them. cat > /tmp/deleteOcFromschema.ldif << EOF dn: cn=Schema changetype: modify delete: objectclasses objectclasses: (examplePerson-oid) dn: cn=schema changetype: modify delete: attributetypes attributetypes: (preferredSwallow-oid) - delete: ibmattributetypes ibmattributetypes: (preferredSwallow-oid) dn: cn=schema

	changetype: modify delete: attributetypes attributetypes: (favoriteColor-oid) - delete: ibmattributetypes ibmattributetypes: (favoriteColor-oid) EOF idsldapadd -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -i /tmp/deleteOcFromschema.ldif

Monitoring, and troubleshooting

Monitoring

Besides monitoring OS, database etc, the following commands can help monitoring SDS

Server	Task

sdsm
sdss
	Check for available worker threads
	This command is a quick way to check if your SDS is able to properly serve clients, and should be used by the monitoring system to check the health of the SDS instance.
	`idsldapsearch -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -s base -b "cn=monitor" objectclass=* | grep -i available_workers | awk -F= '{print $2}'`
	
	Output:
	`14`

sdsm	Check replication state
	`idsldapsearch -h sdsm.example.com -p 636 -Z -b "O=SAMPLE" -D "cn=root" -w passw0rd "(objectclass=ibm-replicationagreement)" ibm-replicationstate | grep ibm-replicationstate | awk -F= '{print $2}'`
	
	Output:
	`error log full`
	
	If output is "`ready`", then there's no problem from SDS' perspective

sdsm	Also check replication state of sdss
	`idsldapsearch -h sdss.example.com -p 636 -Z -b "O=SAMPLE" -D "cn=root" -w passw0rd "(objectclass=ibm-replicationagreement)" ibm-replicationstate | grep ibm-replicationstate | awk -F= '{print $2}'`
	
	Output:
	`error log full`

sdsm	Check replicated data between two DITs
	
	ldapdiff only supports JKS keystores, so you need to create one with your signer CAs
	
	Create keystore database
	`/opt/ibm/ldap/V6.3.1/java/jre/bin/ikeycmd \`
	`-keydb \`
	`-create \`
	`-db /opt/ibm/ldap/V6.3.1/etc/ldapkey.jks \`
	`-pw ssl_password \`
	`-type jks`
	
	Add CA certificate to keystore
	`/opt/ibm/ldap/V6.3.1/java/jre/bin/ikeycmd \`
	` -cert \`
	` -add \`
	` -db /opt/ibm/ldap/V6.3.1/etc/ldapkey.jks \`
	` -pw ssl_password \`
	` -type jks \`
	` -label example-ca \`
	` -file /net/main/srv/common-setup/ssl/cacert.pem \`

```
 -format ascii \
 -trust enable
```

Create a script, so you don't have to recall the switches each time

```
cat > /usr/local/bin/sds_compare_dits.sh << EOF
#!/bin/sh

# this is a workaround because of a broken ldapdiff that wasn't fixed until
FP8.
export LDAPDIFF_PAGE_SIZE=2147483647

SDSDIR="/opt/IBM/ldap/V6.3.1"
BASEDN="o=sample"
BINDDN="cn=root"
BINDPW="passw0rd"
SUPPLIER_SERVER=sdsm.example.com
SUPPLIER_PORT=636
CONSUMER_SERVER=sdss.example.com
CONSUMER_PORT=636
KEYSTORE_FILE="/opt/ibm/ldap/V6.3.1/etc/ldapkey.jks"
KEYSTORE_PW="ssl_password"

OPTIONS=" -S \\
 -b "\${BASEDN}" \\
 -sh \${SUPPLIER_SERVER} \\
 -sp \${SUPPLIER_PORT} \\
 -sD \${BINDDN} \\
 -sw \${BINDPW} \\
 -sZ \\
 -sK \${KEYSTORE_FILE} \\
 -sP \${KEYSTORE_PW} \\
 -sN jks \\
 -sT \${KEYSTORE_FILE} \\
 -sY \${KEYSTORE_PW} \\
 -st jks \\
 -ch \${CONSUMER_SERVER} \\
 -cp \${CONSUMER_PORT} \\
 -cD \${BINDDN} \\
 -cw \${BINDPW} \\
 -cZ \\
 -cK \${KEYSTORE_FILE} \\
 -cP \${KEYSTORE_PW} \\
 -cN jks \\
 -cT \${KEYSTORE_FILE} \\
 -cY \${KEYSTORE_PW} \\
 -ct jks"

\$SDSDIR/bin/ldapdiff \$OPTIONS

EOF
chmod +x /usr/local/bin/sds_compare_dits.sh
```

Now run the script

```
/usr/local/bin/sds_compare_dits.sh
```

Output:

	GLPJBP058I Security protocol set to SSL_TLSv2 GLPJBP058I Security protocol set to SSL_TLSv2 GLPJBP008I Traversing the tree on both the servers... GLPJBP009I Successfully finished traversing the tree on both the servers. GLPJBP004I Schema compare is in progress... This might take a few minutes... GLPJBP003I Schema compare is complete.

LDAP replication troubleshooting

Fixing single threaded replication problems

Single threaded replication is quite robust, and will nearly always recover when servers have come out of sync, and is able to reconnect. So if there's problems with single threaded replication its nearly always either a network problem, or a authentication problem.

Server	Task

sdsm	**Network problem**
	Check that each node can reach other on 389/tcp
	`echo close | telnet sdss.example.com 389`
	
	Good output:
	`Trying 172.16.226.11...`
	`Connected to sdss.example.com.`
	`Escape character is '^]'.`
	`Connection closed by foreign host.`
	
	Bad output:
	`Trying 172.16.226.11...`
	`telnet: connect to address 172.16.226.11: Connection refused`
	
	Check that each node can reach other on 636/tcp, and perform a SSL handshake
	`echo Q | openssl s_client -connect sdss.example.com:636`
	
	Good output:
	`CONNECTED(00000003)`
	`depth=1 /C=US/ST=Somewhere in the US/L=Some US metropolis/O=Large Technology Company (EXAMPLE)/OU=Certifying Authority/CN=EXAMPLE CA/emailAddress=ca@example.com`
	`verify return:1`
	`depth=0 /C=SE/ST=Somewhere in the US/L=Some US Metropolis/O=Large Technology Company (EXAMPLE.com)/OU=Technical/CN=sdss.example.com/emailAddress=root@sdss.example.com`
	`verify return:1`
	`---`
	`Certificate chain`
	`0 s:/C=SE/ST=Somewhere in the US/L=Some US Metropolis/O=Large Technology Company (EXAMPLE.com)/OU=Technical/CN=sdss.example.com/emailAddress=root@sdss.example.com`

```
   i:/C=US/ST=Somewhere in the US/L=Some US metropolis/O=Large Technology
Company (EXAMPLE)/OU=Certifying Authority/CN=EXAMPLE
CA/emailAddress=ca@example.com
 1 s:/C=US/ST=Somewhere in the US/L=Some US metropolis/O=Large Technology
Company (EXAMPLE)/OU=Certifying Authority/CN=EXAMPLE
CA/emailAddress=ca@example.com
   i:/C=US/ST=Somewhere in the US/L=Some US metropolis/O=Large Technology
Company (EXAMPLE)/OU=Certifying Authority/CN=EXAMPLE
CA/emailAddress=ca@example.com
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIFljCCBH6gAwIBAgIBSDANBgkqhkiG9w0BAQUFADCByDELMAkGA1UEBhMCVVMx
HDAaBgNVBAgME1NvbWV3aGVyZSBpbiB0aGUgVVMxGzAZBgNVBAcMElNvbWUgVVMg
bWV0cm9wb2xpczErMCkGA1UECgwiTGFyZ2UgVGVjaG5vbG9neSBDb21wYW55IChF
WEFNUExFKTEdMBsGA1UECwwUQ2VydGlmeWluZyBBdXRob3JpdHkxEzARBgNVBAMM
CkVYQU1QTEUgQ0ExHTAbBgkqhkiG9w0BCQEWDmNhQGV4YW1wbGUuY29tMB4XDTE3
MDIwMTEzNTIzM1oXDTE4MDIwMTEzNTIzM1owgc4xCzAJBgNVBAYTAlNFMRwwGgYD
VQQIExNTb21ld2hlcmUgaW4gdGhlIFVTMRswGQYDVQQHExJTb21lIFVTIE1ldHJv
cG9saXMxLzAtBgNVBAoTJkxhcmdlIFRlY2hub2xvZ3kgQ29tcGFueSAoRVhBTVBM
RS5jb20pMRIwEAYDVQQLEwlUZWNobmljYWwxGTAXBgNVBAMTEHNkc3MuZXhhbXBs
ZS5jb20xJDAiBgkqhkiG9w0BCQEWFXJvb3RAc2Rzcy5leGFtcGxlLmNvbTCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAwOlWWSBG5lZRMQVBWSvEIuee4yTLfG/v
UBthrIQOyKGLSmgr2qlMHRjOaFK3C+QljKlC7TGp4NWv6Pw5DvoauSiFr+LHb31c
aosm31/+NDlMUS1m3TCmLhSv6AkUGruIdgfg6qJfMISsgHE78kTikldUll0Y0wxI
fMt/gcOh1u8CAwEAAaOCAgUwggIBMAkGA1UdEwQCMAAwLgYJYIZIAYb4QgENBCEW
H21rY2VydC5wbCBnZW5lcmF0ZWQgY2VydGlmaWNhdGUwHQYDVR0OBBYEFHPNdeuE
CaX18vmQJfAjwe7MQrJDMB8GA1UdIwQYMBaAFDt62jR8x8q3zemVpNFIYQPY9OXQ
MCkGCWCGSAGG+EIBAgQcFhpodHRwOi8vY2EuZXhhbXBsZS5jb20vc3NsLzAVBglg
hkgBhvhCAQQECBYGY2EuY3JsMB4GCWCGSAGG+EIBAwQRFg9yZXZvY2F0aW9uLmh0
bWwwGwYJYIZIAYb4QgEHBA4WDHJlbmV3YWwuaHRtbDAaBglghkgBhvhCAQgEDRYL
cG9saWN5Lmh0bWwwLwYDVR0SBCgwJoYkaHR0cDovL2NhLmV4YW1wbGUuY29tL3Nz
bC9jYWNlcnQuZGVyMDEGA1UdHwQqMCgwJqAkoCKGIGh0dHA6Ly9jYS5leGFtcGxl
LmNvbS9zc2wvY2EuY3JsMIGEBgNVHREEfTB7ghBzZHNzLmV4YW1wbGUuY29tghB0
ZHNzLmV4YW1wbGUuY29tghBpZHNzLmV4YW1wbGUuY29tgg9zZHMuZXhhbXBsZS5j
b22CD3Rkcy5leGFtcGxlLmNvbYIPaWRzLmV4YW1wbGUuY29tghBsZGFwLmV4YW1w
bGUuY29tMA0GCSqGSIb3DQEBBQUAA4IBAQAPMpwikcip2KM/6EeG2updp1fysAtY
IsyDm+/G5mAPq1meb0kaZDV8xN+jmJtDCjHid5JR5pqcsA1emEZ8csN8Yzc/grLz
VTk3QLwN1FimqHveIz7o3NBd0YzDWXEG47keaRtSIUP6OE/g7yamIXqXyHb+ek3C
1SQYaWdDXqR3XV5r8f+6AuoMpIy8ac5DHzrlddbtUszyFDFlfxcM52H10KsyepqT
TkaUd7uX23vvsXcKnfmpeea9WFqT8wbfuoN938ib01RVtVjwSMrU/qQAXLqHmrBp
QVjQxHbAZObL3dg4C6SDURrXGlz+6uLqG/91jw2542AkHVTXJEvfDas4
-----END CERTIFICATE-----
subject=/C=SE/ST=Somewhere in the US/L=Some US Metropolis/O=Large
Technology Company
(EXAMPLE.com)/OU=Technical/CN=sdss.example.com/emailAddress=root@sdss.examp
le.com
issuer=/C=US/ST=Somewhere in the US/L=Some US metropolis/O=Large Technology
Company (EXAMPLE)/OU=Certifying Authority/CN=EXAMPLE
CA/emailAddress=ca@example.com
---
No client certificate CA names sent
---
SSL handshake has read 2725 bytes and written 326 bytes
---
New, TLSv1/SSLv3, Cipher is AES128-SHA
```

	Server public key is 1024 bit Secure Renegotiation IS supported Compression: NONE Expansion: NONE SSL-Session: Protocol : TLSv1 Cipher : AES128-SHA Session-ID: EBD750C7F240C35306DA521443435F0F134796B50486FD63869925157E3731A1 Session-ID-ctx: Master-Key: 36C4E8F2E8E83B29C9778D077DB8A883EED145C78BFE7DD25057A479979E937F95EF1C547F4B447FE2F5DF7844327E1A Key-Arg : None Start Time: 1491386209 Timeout : 300 (sec) Verify return code: 0 (ok) --- DONE Bad output: CONNECTED(00000003) 24564:error:140790E5:SSL routines:SSL23_WRITE:ssl handshake failure:s23_lib.c:188: Another bad output: socket: Connection refused connect:errno=111 This will both check if each hostname is resolvable in DNS, and if the firewalls or route rules are correct.
sdss	Check the same on sdss echo close | telnet sdsm.example.com 389 echo Q | openssl s_client -connect sdsm .example.com:636
sdsm	**Authentication problems** Check that cn=master can authenticate on both servers idsldapsearch -h sdsm.example.com -p 636 -Z -D cn=master -w passw0rd -b "o=sample" -s base "(objectclass=organization)" Output: o=sample objectclass=top objectclass=organization objectclass=ibm-replicationContext o=sample
sdss	idsldapsearch -h sdss.example.com -p 636 -Z -D cn=master -w passw0rd -b "o=sample" -s base "(objectclass=organization)" Output: o=sample

	objectclass=top objectclass=organization objectclass=ibm-replicationContext o=sample

How to solve a typical multi-threaded replication problem

These steps should only be carried out after solving any technical problem that prevented the SDS servers from communicating.

Server	Task

sdsm	First set a password.
	`export HISTCONTROL=ignoreboth`
	`PASSWORD=passw0rd`

sdsm	Check replication state of sdsm
	`idsldapsearch -h sdsm.example.com -p 636 -Z -b "O=SAMPLE" -D "cn=root" -w passw0rd "(objectclass=ibm-replicationagreement)" ibm-replicationpendingchangecount ibm-replicationstate`
	
	Output:
	`cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample`
	`ibm-replicationpendingchangecount=320`
	`ibm-replicationstate=error log full`
	
	`cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,o=sample`

sdsm	Check replication state of sdss
	`idsldapsearch -h sdss.example.com -p 636 -Z -b "O=SAMPLE" -D "cn=root" -w $PASSWORD"(objectclass=ibm-replicationagreement)" ibm-replicationpendingchangecount ibm-replicationstate`
	
	Output:
	`cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample`
	
	`cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,o=sample`
	`ibm-replicationpendingchangecount=202`
	`ibm-replicationstate=error log full`

sdsm	Retry all faults on both nodes.
	`idsldapexop -h sdsm -D cn=root -w $PASSWORD -op controlreplerr -retry 0 -ra "cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,O=SAMPLE"`
	
	Output:
	`Operation completed successfully.`
	
	`idsldapexop -h sdss -D cn=root -w $PASSWORD -op controlreplerr -retry 0 -ra "cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,O=SAMPLE"`
	
	Output:
	`Operation completed successfully.`

sdsm	Wait a few minutes, and check the replication state again.
	`idsldapsearch -h sdsm -b "O=SAMPLE" -D "cn=root" -w $PASSWORD "(objectclass=ibm-replicationagreement)" ibm-replicationpendingchangecount ibm-replicationstate`
	
	Output:
	`cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample`
	`ibm-replicationpendingchangecount=40`
	`ibm-replicationstate=error log full`
	
	`cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,o=sample`

sdsm	`idsldapsearch -h sdss -b "O=SAMPLE" -D "cn=root" -w $PASSWORD "(objectclass=ibm-replicationagreement)" ibm-replicationpendingchangecount ibm-replicationstate`
	
	Output:
	`cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample`
	
	`cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,o=sample`
	`ibm-replicationpendingchangecount=202`
	`ibm-replicationstate=error log full`

sdsm	Delete all faults on both nodes - If it doesn't help to retry anymore, then you basically have to delete the all errors entries in the log, as the replication stops if any error has occurred in the log. So to delete all faults on both nodes run.
	`idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op controlreplerr -delete 0 -ra "cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample"`
	
	Output:
	`Operation completed successfully.`
	
	`idsldapexop -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -op controlreplerr -delete 0 -ra "cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,o=sample"`
	
	Output:
	`Operation completed successfully.`

sdsm	"Force" a new replication on both nodes
	`idsldapexop -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -op controlrepl -action replnow -ra "cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,o=sample"`
	
	Output:
	`Operation completed successfully.`

	idsldapexop -h sdss.example.com -p 636 -Z -D cn=root -w passw0rd -op controlrepl -action replnow -ra "cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample" Output: Operation completed successfully.
sdsm	Wait 15 mins, and check the replication state again idsldapsearch -h sdss.example.com -p 636 -Z -b "O=SAMPLE" -D "cn=root" -w passw0rd "(objectclass=ibm-replicationagreement)" ibm-replicationpendingchangecount ibm-replicationstate Output: cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,o=sample ibm-replicationpendingchangecount=0 ibm-replicationstate=ready
sdsm	idsldapsearch -h sdss.example.com -p 636 -Z -b "O=SAMPLE" -D "cn=root" -w passw0rd "(objectclass=ibm-replicationagreement)" ibm-replicationpendingchangecount ibm-replicationstate Output: cn=sdss,ibm-replicaServerId=sdsm,ibm-replicaGroup=default,o=sample cn=sdsm,ibm-replicaServerId=sdss,ibm-replicaGroup=default,o=sample ibm-replicationpendingchangecount=0 ibm-replicationstate=ready

Solving different DITs

If your replication reports that all changes are replicated, but you still find that there are differences between DITs, then you can use ldapdiff to correct any changes between them.

Server	**Task**
sdsm	Reusing the JKS keystore created previously, we'll make a script with slightly different switches. cat > /usr/local/bin/sds_fix_dits.sh << EOF #!/bin/sh # this is a workaround because of a broken ldapdiff that wasn't fixed until FP8. export LDAPDIFF_PAGE_SIZE=2147483647 SDSDIR="/opt/IBM/ldap/V6.3.1" BASEDN="o=sample" BINDDN="cn=root" BINDPW="passw0rd" SUPPLIER_SERVER=sdsm.example.com SUPPLIER_PORT=636 CONSUMER_SERVER=sdss.example.com CONSUMER_PORT=636 KEYSTORE_FILE="/opt/ibm/ldap/V6.3.1/etc/ldapkey.jks" KEYSTORE_PW="ssl_password"

```
OPTIONS=" -b "\${BASEDN}" \\
 -sh \${SUPPLIER_SERVER} \\
 -sp \${SUPPLIER_PORT} \\
 -sD \${BINDDN} \\
 -sw \${BINDPW} \\
 -sZ \\
 -sK \${KEYSTORE_FILE} \\
 -sP \${KEYSTORE_PW} \\
 -sN jks \\
 -sT \${KEYSTORE_FILE} \\
 -sY \${KEYSTORE_PW} \\
 -st jks \\
 -ch \${CONSUMER_SERVER} \\
 -cp \${CONSUMER_PORT} \\
 -cD \${BINDDN} \\
 -cw \${BINDPW} \\
 -cZ \\
 -cK \${KEYSTORE_FILE} \\
 -cP \${KEYSTORE_PW} \\
 -cN jks \\
 -cT \${KEYSTORE_FILE} \\
 -cY \${KEYSTORE_PW} \\
 -ct jks \\
 -a \\
 -F"

\$SDSDIR/bin/ldapdiff \$OPTIONS
EOF
chmod +x /usr/local/bin/sds_fix_dits.sh
```

And run the script

```
/usr/local/bin/sds_fix_dits.sh
```

The output is the same as when ldapdiff just checked the DITs. But now it will have mirrored the base DN DIT from sdsm to sdss

Installing plugins

Tivoli Directory Server Plug-in for Custom Password Policy Rules

SDS doesn't really support a lot of complex password policies, so there's an IBM provided but unsupported plugin that will add a few additional policies.
The plug-in can be found at https://www-304.ibm.com/software/brandcatalog/ismlibrary/details?catalog.label=1TW10TD0B, download, and install it as following.

We'll will add the following policy rules.

`pwdNoSpaces` Specifies that password must not contain spaces. Valid values are true and false.

`pwdNoUserId` Specifies that password must not be same as or contain value of cn and uid attributes. Valid values are true and false

`pwdMinSpecialChars` Specifies the minimum number of special characters that a password must contain. Valid value is a number. If 0 is specified, rule is disabled.

`pwdMinNumericChars` Specifies the minimum number of numeric characters that a password must contain. Valid value is a number. If 0 is specified, rule is disabled.

`pwdMinLowercaseChars` Specifies the minimum number of lowercase characters that a password must contain. Valid value is a number. If 0 is specified, rule is disabled.

`pwdMinUppercaseChars` Specifies the minimum number of uppercase characters that a password must contain. Valid value is a number. If 0 is specified, rule is disabled.

However the following doesn't work (properly because the plugin was compiled for 6.3, and not 6.3.1).

`pwdMaxAscChars` Specifies the maximum number of ascending characters that a password can contain, beyond which server will throw error. The characters could be alphabetic or numeric. Valid value is a number.

`pwdMaxDscChars` Specifies the maximum number of descending characters that a password can contain, beyond which server will throw error. The characters could be alphabetic or numeric. Valid value is a number.

Server	Task

sdsm
sdss
	Goto SDS installation directory, extract the zip file, and link in the library to the SDS library directory.
	`cd /opt/ibm/ldap/V6.3.1`
	`unzip /net/files/srv/install/sdsplugins/custppolicy.zip`
	`ln -s /opt/ibm/ldap/V6.3.1/custppolicy/libcustppolicy_x64linux.so /opt/ibm/ldap/V6.3.1/lib64`

sdsm	Add the plugin to the SDS configuration
	`cat >> /tmp/custppolicy.ldif << EOF`
	`# Add additional password policies plug-in`
	`dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration`
	`changetype: modify`
	`add: ibm-slapdPlugin`
	`ibm-slapdPlugin: preoperation libcustppolicy_x64linux.so customPwdPolicyInit pwdNoSpaces=true pwdNoUserId=true pwdMinSpecialChars=0 pwdMinNumericChars=1 pwdMinLowercaseChars=1 pwdMinUppercaseChars=1`
	`EOF`
	`idsldapmodify -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -f /tmp/custppolicy.ldif`

sdsm
sdss
	Then restart SDS.
	
	`idsslapd -k`
	`idsslapd`
	
	Check the output for
	`GLPCOM021I The preoperation plugin is successfully loaded from libcustppolicy_x64linux.so.`
	
	If you see the message
	`GLPCOM012E Failed to load plugin from libcustppolicy_x64linux.so.`
	
	The check **/home/dsrdbm01/idsslapd-dsrdbm01/logs/ibmslapd.log** for why.

Updating software components

Please refer to [Recommended fixes for IBM Security Directory Server (LDAP)] for latest fixes.

NB: Btw remember that if you update an existing instance, then you'll need to run through all the pre, and post installation steps for each software component.

Scripts used during updating the databases

Server	Task

sdsm
sdss
	Offline backup of all databases in instance
	`cat > /usr/local/bin/db2_offline_backup_all.sh << EOF`
	`#!/bin/sh`
	`BACKUP_DIR=/net/files/srv/backup/db2`
	
	`if [ ! -d \$BACKUP_DIR/\$HOST/\$DB2INSTANCE ]; then`
	`  mkdir -p \$BACKUP_DIR/\$HOST/\$DB2INSTANCE || exit 1`
	`fi`
	
	`for db in \`db2 LIST DATABASE DIRECTORY
		grep "Database name"	awk '{print \$4}'\`; do`
			`  TIMESTAMP=\$(date +%Y%m%d%H%M%S)`
			`  echo "********** starting offline backup of \$db at \$TIMESTAMP**********"`
			`  db2 CONNECT TO \$db`
			`  db2 QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS`
			`  db2 CONNECT RESET`
			`  db2 DEACTIVATE DATABASE \$db`
			`  db2 BACKUP DATABASE \$db TO "\$BACKUP_DIR/\$HOST/\$DB2INSTANCE" EXCLUDE LOGS WITHOUT PROMPTING`
			`  db2 CONNECT TO \$db`
			`  db2 UNQUIESCE DATABASE`
			`  db2 CONNECT RESET`
			`  echo "********** check backup of \$db **********"`
			`  db2ckbkp \$(ls -1 \$BACKUP_DIR/\$HOST/\$DB2INSTANCE/\${db}* | tail -1) || exit 1`
			`  echo "********** prune history before \$TIMESTAMP **********"`
			`  db2 CONNECT TO \$db`
			`  db2 PRUNE HISTORY \$TIMESTAMP AND DELETE`
			`  db2 CONNECT RESET`
			`done`
			`EOF`
			`chgrp grrdbm01 /usr/local/bin/db2_offline_backup_all.sh`
			`chmod 0750 /usr/local/bin/db2_offline_backup_all.sh`

sdsm
sdss
	Online backup of all databases in instance
	`cat > /usr/local/bin/db2_online_backup_all.sh << EOF`
	`#!/bin/sh`
	`BACKUP_DIR=/net/files/srv/backup/db2`
	
	`if [ ! -d \$BACKUP_DIR/\$HOST/\$DB2INSTANCE ]; then`
	`  mkdir -p \$BACKUP_DIR/\$HOST/\$DB2INSTANCE || exit 1`
	`fi`

	for db in \`db2 LIST DATABASE DIRECTORY | grep "Database name" | awk '{print \$4}'\`; do
	  TIMESTAMP=\$(date +%Y%m%d%H%M%S)
	  echo "********** starting online backup of \$db at \$TIMESTAMP**********"
	  db2 BACKUP DATABASE \$db ONLINE TO "\$BACKUP_DIR/\$HOST/\$DB2INSTANCE" INCLUDE LOGS WITHOUT PROMPTING
	  echo "********** check backup of \$db **********"
	  db2ckbkp \$(ls -1 \$BACKUP_DIR/\$HOST/\$DB2INSTANCE/\${db}* | tail -1) || exit 1
	  echo "********** prune history before \$TIMESTAMP **********"
	  db2 CONNECT TO \$db
	  db2 PRUNE HISTORY \$TIMESTAMP AND DELETE
	  db2 CONNECT RESET
	done
	EOF
	chgrp grrdbm01 /usr/local/bin/db2_online_backup_all.sh
	chmod 0750 /usr/local/bin/db2_online_backup_all.sh

sdsm
sdss
	Reorg all databases in instance
	cat > /usr/local/bin/db2_reorg_all.sh << EOF
	#!/bin/sh
	for db in \`db2 LIST DATABASE DIRECTORY | grep "Database name" | awk '{print \$4}'\`; do
	  echo "********** checking \$db **********"
	  db2 CONNECT TO \$db
	  db2 REORGCHK UPDATE STATISTICS ON TABLE ALL | grep -B1 * | grep 'Table:' | awk '{print \$2}' > /tmp/tables
	  for table in \`cat /tmp/tables\`; do
	  echo "********** running reorg on \$table **********"
	  db2 REORG TABLE \$table ALLOW NO ACCESS
	  db2 REORG INDEXES ALL FOR TABLE \$table ALLOW NO ACCESS
	  db2 RUNSTATS ON TABLE \$table AND INDEXES ALL
	  done
	  db2 TERMINATE
	  rm /tmp/tables
	done
	EOF
	chgrp grrdbm01 /usr/local/bin/db2_reorg_all.sh
	chmod 0750 /usr/local/bin/db2_reorg_all.sh

sdsm
sdss
	Update all databases in instance
	cat > /usr/local/bin/db2_update_all.sh << EOF
	#!/bin/sh
	for db in \`db2 LIST DATABASE DIRECTORY | grep "Database name" | awk '{print \$4}'\`; do
	  echo "********** updating \$db **********"
	  db2updv97 -d \$db
	done
	EOF
	chgrp grrdbm01 /usr/local/bin/db2_update_all.sh
	chmod 0750 /usr/local/bin/db2_update_all.sh

sdsm sdss	Rebind all databases cat > /usr/local/bin/db2_rebind_all.sh << EOF

	#!/bin/sh
	for db in \`db2 LIST DATABASE DIRECTORY | grep "Database name" | awk '{print \$4}'\`; do
	  echo "********** rebinding \$db **********"
	  db2 CONNECT TO \$db
	  db2 BIND \$HOME/sqllib/bnd/db2schema.bnd BLOCKING ALL GRANT PUBLIC SQLERROR CONTINUE
	  db2 BIND \$HOME/sqllib/bnd/@db2ubind.lst BLOCKING ALL GRANT PUBLIC ACTION ADD
	  db2 BIND \$HOME/sqllib/bnd/@db2cli.lst BLOCKING ALL GRANT PUBLIC ACTION ADD
	  db2 TERMINATE
	  db2rbind \$db -l \${db}.rbind_out all
	done
	EOF
	chgrp grrdbm01 /usr/local/bin/db2_rebind_all.sh
	chmod 0750 /usr/local/bin/db2_rebind_all.sh

sdsm
sdss
	Prune logs in all databases
	cat > /usr/local/bin/db2_prune_all.sh << EOF
	#!/bin/sh
	for db in \$(db2 LIST DATABASE DIRECTORY | grep "Database name" | awk '{print \$4}'); do
	  echo "********** prune logs for \$db **********"
	  db2 CONNECT TO \$db
	  # Apparently GET DB CONFIG doesn't print to stdout, so you can't pipe the output to a grep in a command substitution - i.e. \$()
	  db2 GET DB CONFIG > /tmp/db_config
	  db2 PRUNE LOGFILE PRIOR TO \$(cat /tmp/db_config | grep "First active log file" | awk -F= '{print \$2}')
	  rm -f /tmp/db_config
	  db2 CONNECT RESET
	done
	EOF
	chgrp grrdbm01 /usr/local/bin/db2_prune_all.sh
	chmod 0750 /usr/local/bin/db2_prune_all.sh

Updating IBM Security Directory Server

Extract packages for FP 6.3.1.19

Server	Task
files	Set Fix Pack 6.3.1.19 location export INSTALL_DIR=/net/files/srv/install/sdsfps/63119/linux64
files	Create installation source directory mkdir -p $INSTALL_DIR cd $INSTALL_DIR
files	Extract GSK FP 8.0.50.67 tar zxf ~/DownloadDirector/8.0.50.67-ISS-GSKIT-LinuxX64-FP0067.tar.gz
files	Extract DB2 9.7.0.11 Universal FP mkdir v9.7fp11_linuxx64_universal_fixpack tar zxf ~/DownloadDirector/v9.7fp11_linuxx64_universal_fixpack.tar.gz -C v9.7fp11_linuxx64_universal_fixpack

files	Copy SDS JDK 6.0.16.0 (6.0 SR-26) to install directory
	`mkdir 6.0.16.26-ISS-JAVA-LinuxX64-FP0026`
	`cp ~/DownloadDirector/6.0.16.26-ISS-JAVA-LinuxX64-FP0026.tar 6.0.16.26-ISS-JAVA-LinuxX64-FP0026`

files	Extract SDS FP 6.3.1.19 `tar zxf ~/DownloadDirector/6.3.1.19-ISS-ISDS-LinuxX64-IF0019.tar.gz`

files	Copy WAS FP 7.0.0.41 to install directory
	`mkdir 7.0.0.41-ISS-WASEmbeded-LinuxX64-FP0000041`
	`cp ~/Downloads/7.0.0.41-WS-UPDI-LinuxAMD64.zip \`
	`~/Downloads/7.0.0-WS-WASEmbeded-LinuxX64-FP0000041.pak \`
	`~/Downloads/7.0.0-WS-WASSDK-LinuxX64-FP0000041.pak \`
	`$INSTALL_DIR/7.0.0.41-ISS-WASEmbeded-LinuxX64-FP0000041`

Stop … everything

Server	Task

sdsm
sdss
	Shutting down SDS slapd services
	`for dsinst in `idsilist 2>&1 | grep -v 'Directory server instance(s):'`; do`
	`  idsslapd -I $dsinst -k`
	`done`

sdsm
sdss
	Stopping, and disable start up of all DB2 instances
	`for dbinst in `/opt/ibm/db2/V9.7/instance/db2ilist`; do`
	`  su -l $dbinst -c "db2 force applications all; db2 terminate; db2stop; db2licd -end"`
	`done`

sdsm
sdss
	Disable DB2 Fault Monitor
	`/opt/ibm/db2/V9.7/bin/db2fmcu -d`
	`telinit q`

sdsm
sdss
	Clean memory for any DB2 related libraries
	`for dbinst in `/opt/ibm/db2/V9.7/instance/db2ilist`; do`
	`  su -l $dbinst -c "ipclean"`
	`done`

sdsm sdss	Check that no DB2 libraries are in memory `lsof +D /opt/ibm/db2/V9.7`

sdsm
sdss
	Shutting down SDS administrative services
	`for dsinst in `idsilist 2>&1 | grep -v 'Directory server instance(s):'`; do`
	`  ibmdiradm -I $dsinst -k`
	`done`

Update packages

This chapter doesn't include updating eWAS, and the Web Administration Tool. Please refer to "Appendix B - Installing Embedded WAS server, and Web Administration Tool" for how to update.

Server	Task
sdsm sdss	Set Fix Pack 6.3.1.19 location `export INSTALL_DIR=/net/files/srv/install/sdsfps/63119/linux64`

sdsm
sdss
	Update GSK to 8.0.50.67
	`rpm -Uvh \`
	`$INSTALL_DIR/8.0.50.67-ISS-GSKIT-LinuxX64-FP0067/64/gskcrypt64-8.0.50.67.linux.x86_64.rpm \`
	`$INSTALL_DIR/8.0.50.67-ISS-GSKIT-LinuxX64-FP0067/64/gskssl64-8.0.50.67.linux.x86_64.rpm`

sdsm sdss	Update DB2 ESE to 9.7.0.11 `$INSTALL_DIR/v9.7fp11_linuxx64_universal_fixpack/universal/installFixPack -b /opt/ibm/db2/V9.7`

sdsm
sdss
	Update instances
	`for dbinst in `/opt/ibm/db2/V9.7/instance/db2ilist`; do`
	`  /opt/ibm/db2/V9.7/instance/db2iupdt $dbinst`
	`done`

sdsm
sdss
	Update and rebind all databases in all instances
	`for dbinst in `/opt/ibm/db2/V9.7/instance/db2ilist`; do`
	`  su -l $dbinst -c "db2start"`
	`  su -l $dbinst -c "/usr/local/bin/db2_offline_backup_all.sh"`
	`  su -l $dbinst -c "/usr/local/bin/db2_update_all.sh"`
	`  su -l $dbinst -c "/usr/local/bin/db2_rebind_all.sh"`
	`done`

sdsm
sdss
	Enable automatic start of services through DB2 Fault Monitor
	`/opt/ibm/db2/V9.7/bin/db2fmcu -u -p /opt/ibm/db2/V9.7/bin/db2fmcd`
	`telinit q`

sdsm sdss	Update SDS to 6.3.1.19 `(cd $INSTALL_DIR/6.3.1.19-ISS-ISDS-LinuxX64-IF0019; echo 1 | ./idsinstall -u)`

sdsm
sdss
	Update SDS JDK to 6.0.26.0
	`mv /opt/ibm/ldap/V6.3.1/java /opt/ibm/ldap/V6.3.1/java_old`
	`tar xf`
	`$INSTALL_DIR/6.0.16.26-ISS-JAVA-LinuxX64-FP0026/6.0.16.26-ISS-JAVA-LinuxX64-FP0026.tar -C /opt/ibm/ldap/V6.3.1`

Start … everything

Server	Task

sdsm
sdss
	Starting SDS administrative services
	`for dsinst in `idsilist 2>&1 | grep -v 'Directory server instance(s):'`; do`
	`  ibmdiradm -I $dsinst`
	`done`
	
	NB: DB2 is started automatically by the administrative services

sdsm
sdss
	Starting SDS slapd services
	`for dsinst in `idsilist 2>&1 | grep -v 'Directory server instance(s):'`; do`
	`  idsslapd -I $dsinst`
	`done`

Post configuration

Server	Task

sdsm
sdss
	Add custom CA signer certificate to default SSL truststore for command-line tools
	The SDS client tools keystore have been overwritten during the update, so you need to reimport your custom CA again.
	`/usr/local/ibm/gsk8_64/bin/gsk8capicmd_64 -cert -add -db /opt/ibm/ldap/V6.3.1/etc/ldapkey.kdb -pw ssl_password -type cms -label example-ca -file /net/main/srv/common-setup/ssl/cacert.pem -format ascii -trust enable`
	`/usr/local/ibm/gsk8_64/bin/gsk8capicmd_64 -cert -list -db /opt/ibm/ldap/V6.3.1/etc/ldapkey.kdb -pw ssl_password`

sdsm sdss	Check that ids* command line tools can use SSL `idsldapsearch -h sdsm.example.com -p 636 -Z -D cn=root -w passw0rd -b "" -s sub "(objectclass=organization)"`

Updating Embedded WAS server, and Web Administration Tool

Server	Task
sdsm sdss	Set Fix Pack 6.3.1.19 location `export INSTALL_DIR=/net/files/srv/install/sdsfps/63119/linux64`
sdsm sdss	Stop server `/opt/ibm/ldap/V6.3.1/appsrv/profiles/TDSWebAdminProfile/bin/stopServer.sh server1`

sdsm
sdss
	Update Embedded WAS
	`if [ -d /tmp/updi ]; then rm -fr /tmp/updi/*; else mkdir /tmp/updi; fi`
	`(cd /tmp/updi; unzip $INSTALL_DIR/7.0.0.41-ISS-WASEmbeded-LinuxX64-FP0000041/7.0.0.41-WS-UPDI-LinuxAMD64.zip)`
	`/tmp/updi/UpdateInstaller/install -OPT silentInstallLicenseAcceptance="true" -OPT allowNonRootSilentInstall="true" -OPT disableOSPrereqChecking="true" -OPT disableEarlyPrereqChecking="true" -OPT installLocation="/opt/IBM/WebSphere/UpdateInstaller" -silent`
	`/opt/IBM/WebSphere/UpdateInstaller/update.sh -OPT checkFilePermissions="true" -W maintenance.package="$INSTALL_DIR/7.0.0.41-ISS-WASEmbeded-LinuxX64-FP0000041" -OPT disableNonBlockingPrereqChecking="true" -W product.location="/opt/ibm/ldap/V6.3.1/appsrv"  -W update.type="install" -silent`

sdsm sdss	Verify eWAS version `/opt/ibm/ldap/V6.3.1/appsrv/bin/versionInfo.sh`
sdsm sdss	Clean up `rm -fr /tmp/updi`

sdsm sdss	Update Web Administration Tool application `(cd $INSTALL_DIR/6.3.1.19-ISS-ISDS-LinuxX64-IF0019 && echo 1 | ./idsinstall -u)`

sdsm
sdss
	Redeploy Web Administration application
	`cd /opt/ibm/ldap/V6.3.1/idstools/`
	`./deploy_IDSWebApp`

sdsm sdss	Verify Web Administration Tool deployment `/opt/ibm/ldap/V6.3.1/idstools/deploy_IDSWebApp -v`

sdsm
sdss
	Disable node administrative security
	FP19 forgets to disable security on the profile, so you have to disable it yourself
	`/opt/ibm/ldap/V6.3.1/appsrv/profiles/TDSWebAdminProfile/bin/stopServer.sh server1 -username wasadmin -password secret`
	`/opt/ibm/ldap/V6.3.1/appsrv/bin/wsadmin.sh -conntype NONE -profileName TDSWebAdminProfile -c "$AdminApp securityoff"`

sdsm sdss	Start server `/opt/ibm/ldap/V6.3.1/idstools/bin/startWebadminApp`

References

- IBM Security Directory Server V6.3.1 Knowledge Center
- IBM Security Directory Server V6.3.1.5 Knowledge Center
- IBM Tivoli Directory Server for z/OS V1R12.0 - Creating a peer-to-peer replication topology (This has better configuration examples than the infocenter for distributed systems)
- IBM TN1396012: Syncing data between two servers
- Enable SHA-256 as password encryption algorithm in Tivoli Directory Server 6.3
- Debugging Replication in IBM Tivoli Directory Server
- Approaches for solving problems with Tivoli Directory Server synchronization
- Recommended fixes for IBM Security Directory Server (LDAP)
- Tivoli Directory Server Plug-in for Custom Password Policy Rules
- IBM Tivoli® Directory Server backup using IBM Tivoli® Storage Manager
- Performance Tuning for IBM Security Directory Server