**(uploaded to github - done)**

Creating a **disaster recovery (DR) script** for **SAP Sybase ASE** (Adaptive Server Enterprise) is critical to ensure business continuity in case of a server crash, corruption, or other major failures.

A disaster recovery script typically automates or documents how to:

* Rebuild the server.
* Restore databases and configuration.
* Minimize downtime.

Here’s a **step-by-step approach** to create a **DR script** for Sybase ASE:

### **1. Pre-Requisites:**

* Ensure **regular backups** of:  
  + **Full database dumps** (dump database)
  + **Transaction logs** (dump transaction)
  + **Master database** and other system databases (dump database master)
* Backup:  
  + **RUN\_servername** file (the ASE startup file)
  + **Configuration file** ($SYBASE/ASE-\*/bin/\*.cfg)
  + **Interfaces file** (listening endpoints, $SYBASE/interfaces)
  + **Backup of Sybase license files** ($SYBASE/SYSAM-\*/licenses)
  + **Any external scripts (cron jobs, monitoring scripts)**

### **2. DR Script Main Sections:**

Here's a **template outline** for the script:

bash

CopyEdit

#!/bin/bash

# Disaster Recovery Script for Sybase ASE

# Author: [Your Name]

# Date: [Date]

### 1. Environment Setup

export SYBASE=/opt/sybase

export SYBASE\_ASE=ASE-16\_0

export SYBASE\_OCS=OCS-16\_0

export PATH=$SYBASE/$SYBASE\_ASE/bin:$PATH

export LD\_LIBRARY\_PATH=$SYBASE/$SYBASE\_ASE/lib:$LD\_LIBRARY\_PATH

export ASE\_SERVER=MY\_SERVER\_NAME

### 2. Start ASE server in minimal mode (if needed)

echo "Starting ASE server in single-user mode..."

startserver -f $SYBASE/$SYBASE\_ASE/install/RUN\_$ASE\_SERVER

### 3. Restore the master database (if corrupted)

echo "Restoring master database..."

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

shutdown with nowait

go

exit

EOF

# Assuming you have master database dump at /backup/master.dmp

dataserver -d/backup/master.dmp -i$SYBASE/$SYBASE\_ASE/install/RUN\_$ASE\_SERVER

### 4. Restart server normally

echo "Restarting ASE server normally..."

startserver -f $SYBASE/$SYBASE\_ASE/install/RUN\_$ASE\_SERVER

### 5. Restore system databases if necessary

echo "Restoring model, sybsystemdb, sybsystemprocs..."

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

load database model from "/backup/model.dmp"

go

load database sybsystemprocs from "/backup/sybsystemprocs.dmp"

go

load database sybsystemdb from "/backup/sybsystemdb.dmp"

go

online database model

go

online database sybsystemprocs

go

online database sybsystemdb

go

exit

EOF

### 6. Restore user databases

echo "Restoring user databases..."

for db in $(cat /backup/db\_list.txt); do

echo "Restoring database: $db"

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

load database $db from "/backup/$db.dmp"

go

online database $db

go

exit

EOF

done

### 7. Apply transaction logs (optional if logs exist)

# For each database, load transaction dumps if any

### 8. Validate Databases

echo "Running dbcc checks..."

for db in $(cat /backup/db\_list.txt); do

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

use $db

go

dbcc checkdb

go

exit

EOF

done

### 9. Final Configurations

# Rebuild any external devices, update configurations if needed.

echo "Disaster Recovery complete for ASE server: $ASE\_SERVER."

### **3. Important Notes:**

* **Password security:** Replace <password> with a secure method (environment variable, encrypted store).
* **Backup devices:** If your dumps are on a **dump device**, be sure you recreate the devices (disk init).
* **Transaction logs:** If your recovery point is after full backup, make sure you restore **transaction logs sequentially**.
* **Replication setups:** If you're using **Replication Server**, additional steps are needed.
* **Monitoring:** Always monitor during and after restoration using sp\_who, sp\_lock, and server logs.

### **4. Bonus Tips:**

* **Document exact backup locations** and update them dynamically in the script.
* **Test your DR scripts** quarterly in a test environment.
* **Use "load database with headeronly"** to verify dump files without restoring.
* **Consider automating** sending email alerts if steps fail (using mailx or similar).

**For full database recovery**, meaning:

* Load **full dumps**,
* Then optionally load **transaction logs** *completely*,
* Bring database **online**.

**For *Point-in-Time Recovery*** (PITR), **you'll need to modify** the script a little bit, because:

* You **load a full dump**,
* Then **sequentially load transaction log dumps**,
* **Stop** *right before* or *at* a specific point (for example, before a user error or corruption),
* Use load transaction with stop\_at (or until\_time).

### **Key Differences for Point-In-Time Recovery**

| **Full Recovery** | **Point-in-Time Recovery** |
| --- | --- |
| Load full dump, optionally logs | Load full dump + transaction logs carefully |
| Bring database online after all loads | Stop at a specific timestamp inside transaction logs |
| Simple script | Needs control: stop after reaching recovery point |

### **In the script for Point-in-Time recovery, you must:**

After loading the full database dump:

sql

CopyEdit

load database mydb from "/backup/mydb\_full.dmp"

go

Then load each transaction log **until** the point-in-time you want:

sql

CopyEdit

load transaction mydb from "/backup/mydb\_log1.trn"

go

load transaction mydb from "/backup/mydb\_log2.trn"

with stop\_at = "Apr 29, 2025 10:34:00AM"

go

After that Sybase will **truncate** at that time, and **bring the database online** automatically.

### **Modified Recovery Script Section (for PITR)**

# Load full database dump

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

load database $db from "/backup/$db\_full.dmp"

go

EOF

# Load transaction logs

for logdump in /backup/$db\_log\*.trn; do

echo "Loading transaction log: $logdump"

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

load transaction $db from "$logdump"

go

EOF

done

# Final transaction log with STOP\_AT

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

load transaction $db from "/backup/$db\_log\_final.trn"

with stop\_at = "Apr 29, 2025 10:34:00AM"

go

EOF

# No need to ONLINE database manually if stop\_at is used.

### **Important Notes for PITR:**

* Your transaction log dumps must be **sequential** (no gaps).
* Time format for stop\_at must match your server's **datetime** format.
* If stop\_at is earlier than the end of a transaction log, Sybase **applies transactions until that point** and **truncates the rest**.
* Use load transaction with headeronly to check what's inside a .trn file before loading.
* If restoring master database and system databases, you can't use point-in-time — only user databases allow PITR.

### **In summary:**

* **No, you cannot use the full recovery script exactly as-is for Point-in-Time Recovery**.
* **You need slight changes** — mainly inserting stop\_at when loading the final transaction log.
* **You must know the timestamp** you are recovering to.

| **Type** | **Description** | **When Used** |
| --- | --- | --- |
| **Full Disaster Recovery (Full Recovery)** | Restore everything (master, system, user databases) from full database backups (and all transaction logs) | Server crash, data center failure, hardware failure |
| **Point-in-Time Recovery (PITR)** | Restore a database **up to a specific moment** (before corruption, bad update, or user error) | Logical errors, accidental data deletion, application bugs |

**Disaster Recovery** in the traditional sense **almost always means Full Recovery** — **getting the whole server or environment back to last good state** (usually from the latest available full backups).

**Point-in-Time Recovery** is a **specialized form** of recovery **mainly used for a single database** when you want to "rewind" it to just before a disaster **inside** the database — **not the server itself**.

### **In simple words:**

* If your **whole Sybase server** (hardware, VM, OS, ASE installation) **crashed** →  
   ➔ **Full Disaster Recovery (Full Backup + All Logs)**.
* If someone **dropped a table** or **corrupted data at 10:34AM**, but server is fine →  
   ➔ **Point-in-Time Recovery** for that **one database**.

### **Why are they different?**

| **Full Disaster Recovery** | **Point-in-Time Recovery** |
| --- | --- |
| "Bring everything back" | "Fix a database to before a mistake" |
| Focused on infrastructure + databases | Focused only on data/time inside the database |
| Needs full server/filesystem rebuild sometimes | No server rebuild needed |

### **Quick Examples:**

* Server caught fire or crashed:  
   ➔ **Full disaster recovery**.
* User accidentally runs DELETE FROM orders at 2PM:  
   ➔ **Point-in-time recovery to 1:59 PM**.

### **Therefore:**

**Disaster Recovery usually means Full Recovery**,  
 **Point-in-Time Recovery is a database-level recovery strategy** — and is sometimes part of disaster recovery when only specific data is affected.

**Disaster Recovery (DR) Flow Diagram** that shows the difference between **Full Recovery** and **Point-in-Time Recovery** (PITR) for Sybase ASE:

# **Disaster Recovery Flow**

Server or Database Failure Detected

|

V

+---------------------------+

| What failed exactly? |

+---------------------------+

|

+-----------------------------------------+

| |

Whole ASE Server Down Only One Database Corrupted

| |

V V

+----------------------------+ +------------------------------+

| Full Disaster | | Point-in-Time Recovery |

| Recovery | +------------------------------+

+----------------------------+ |

| |

Backup master, system dbs, Full database dump

full + log backups (all dbs) + transaction logs

| |

Rebuild Server Environment? Identify exact "recovery point"

(OS, ASE, configs, devices) (timestamp)

| |

Restore master, model, sybsystemprocs Load full dump

Restore all user databases Load transaction logs

Load transaction logs fully Load transaction logs

| |

Bring server and all dbs online Use `load transaction with stop\_at`

| |

Application reconnects Database brought online

| |

Business Resumes Application reconnects

# 

|  | **Full Disaster Recovery** | **Point-in-Time Recovery** |
| --- | --- | --- |
| Focus | Entire Sybase ASE server | Single database |
| Reason | System-wide crash, hardware loss | Logical mistake, data corruption |
| Includes | OS restore, ASE restore, device rebuild, full db restores | Only db-level restore and stop\_at |
| Complexity | Higher | Medium |
| Time taken | Longer (depends on system size) | Shorter (1 db only) |

# **Also**

**Full DR** = Crash + Server rebuild + All DBs recovered  
 **PITR** = Mistake in DB + Only *one* DB rewind to timestamp

DR Document Plan should be created.