

ASE 15.0.x Migration Checklist

Version 1.3

This document was prepared by Sybase Technical Support and is intended for Sybase DBAs who are preparing to migrate their existing Sybase Adaptive Server Enterprise (ASE) software to ASE 15.0.x. The purpose of this document is to provide a single source for the steps needed to help ensure an efficient and seamless migration to the latest Sybase ASE version. As all DBAs know, however, any enterprise level database migration is never a trivial task and this document is absolutely not meant to replace the rest of the Sybase documentation and resources. It is strongly recommended that you also closely review the following documents:

Planning Your ASE 15.0 Migration,

http://www.sybase.com/files/White_Papers/PlanningASE15Migration-040307.pdf

Migration Guide, Adaptive Server Enterprise Version 15.0,

http://infocenter.sybase.com/help/topic/com.sybase.dc34982_1500/pdf/mig_gde.pdf

Sybase ASE 15.0 Best Practices: Query Processing and Optimization,

http://www.sybase.com/files/White_Papers/ASE15-Optimizer-Best-Practices-v1.1-022008-wp.pdf

Semantic Partitions in ASE 15.0,

http://www.sybase.com/files/White_Papers/ASE-Partitions-040307-wp.pdf

Performance on Sybase Devices Using Direct I/O, Dsync and Raw Partitions,

<http://www.sybase.com/content/1043413/DirectIO-082906-wp.pdf>

Changes to Scope & Semantics of Session-Level Optimization Settings in ASE 15.0.2,

http://www.sybase.com/files/White_Papers/ASE1502_OptimizationCriteria_v1_0.pdf



Before going on, be sure and double check that you have the latest version of this document. Refer to the latest version at <http://www.sybase.com/detail?id=1058962>

Any migration plan typically comprises of 3 major phases:

- Planning
- Pre-migration Phase (including actual Migration)
- Post-migration Phase

In this document, we have put together a checklist of significant steps (and common issues encountered) that can be "checked" off. It will also serve as a ready reference to show your progress as you move from one phase of your migration to another. The pages following the checklist provide the details about each step.

ASE 15.x Migration Checklist

1. Pre-Migration Planning

- Step 1.1: Determine Your Method of Migrating
- Step 1.2: Evaluate New ASE 15.x License Requirements
- Step 1.3: Evaluate Increased ASE 15.x Resource Requirements
- Step 1.4: Consider Implementation of Semantic Partitioning
- Step 1.5: Determine Your ASE Database Device Attributes
- Step 1.6: Review Any Trace Flags Used at Startup
- Step 1.7: Ensure You Use the Latest ASE Release
- Step 1.8: Create a Tactical/Project Plan
- Step 1.9: Build a Test Plan

2. Pre-Migration Steps

- Step 2.1: Record Base Line Performance
- Step 2.2: Generate and Save Abstract Plans
- Step 2.3: Check Consistency of all User and Master Databases
- Step 2.4: Verify Sufficient Free Space in All Databases
- Step 2.5: Verify Sufficient Number of Locks Configured for Upgrade
- Step 2.6: Bulk Copy Out Key System Tables
- Step 2.7: Disable Replication
- Step 2.8: Dump Master Database
- Step 2.9: Unpartition All Partitioned Tables
- Step 2.10: Replace Update Statistics Command
- Step 2.11: Delete or Update Stale Statistics
- Step 2.12: Quiesce All Databases
- Step 2.13: Perform Migration

3. Post-Migration Steps

- Step 3.1: Recreate Table Partitions**
- Step 3.2: Reconfigure With Additional Resources**
- Step 3.3: Configure Statement Cache**
- Step 3.4: Recreate Optimizer Statistics**
- Step 3.5: Evaluate New Optimization Goals**
- Step 3.6: Understand The Role and Usage of Compatibility Mode**
- Step 3.7: Gather Post-Migration Baseline Performance Info**

Detailed Instructions for Each Step

Pre-Migration Planning Step Details

Step 1.1: Determine Your Method of Migrating

In the past, there were basically three methods of upgrading – 1) upgrade in place, 2) *dump/load*, or 3) *bcp* out and *bcp* back in. Because of the new *mount/unmount* feature added to ASE 12.5, there is now a fourth method that is likely considerably faster than *dump/load* for customers that use SAN disk technology.

The steps required for this new method are:

1. **Quiesce** the database using a manifest file
2. Copy/move the devices along with the manifest file to the ASE 15.0 host
3. Mount the database using the ***mount database*** command
4. Bring the database online via the ***online <database_name>*** command

It is during the online database step that the database migration is performed. Note that this method is only supported between machines from the same vendor and platform. Cross platform migrations are not supported.

Sybase Support recommends that, if you are not combining your upgrade with various schema changes that might necessitate a *bcp* in & out type of approach, then we suggest that you use either the ***dump & load*** or the ***mount & unmount*** techniques.

Step 1.2: Evaluate New ASE 15.x License Requirements

ASE 15.x uses the new SYSAM 2.0 system. Some of the new features that you need to evaluate/estimate for your server are as follows:

1. Determine which Product Edition you will need. The licenses that are generated for your product are Edition Specific. Be sure that you know your product edition (PE) prior to installing the software.
2. Determine which License Type you will need. The licenses that are generated for your product are Type Specific. Be sure that you know your license type (LT) prior to installing the software.
3. If your server is unable to check out a license, it will start in Grace Mode and can function for 30 days until you have a valid/working license in place.
4. Licenses are made available and generated at the Sybase Product Download Center (SPDC) at <https://sybase.subscribenet.com/> . If you do not see your product, you will need to contact your Sales Representative to have your profile updated on this website.
5. Some features of ASE (such as XML, Java, Encryption, and LDAP) require a license to be generated. These licenses can be contained in a single file to simplify maintenance (the website offers an option to save a file with all licenses for a particular host).
6. Licenses are not generated specific to a Host Machine. You must have your Host ID (the website contains a checklist/instructions for each platform) when generating the license. These licenses cannot be directly swapped between machines. You have to check the license back in at the website and regenerate it for the new host if moving.
7. Determine the type of licensing, Served or Un-served, that best suits your environment prior to generating the license files. There are guidelines to help

you with this process at

http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc00530_0200/html/sysamug/BABFGHCJ.htm

8. For more specific assistance with the new SYSAM process, please review the online User Guide for Sybase Software Asset Management.

Step 1.3: Evaluate Increased ASE 15.x Resource Requirements

This step is provided to give you early and advanced notice that ASE 15.x requires additional resources. Your current hardware platform may need to be reconfigured to support these additional requirements under ASE 15.x. Steps 3.2 and 3.3 provided more detailed information on how to specifically reconfigure various ASE parameters.

Step 1.4: Consider Implementation of Semantic Partitioning

ASE 15.0 now supports horizontal partitioning, in which a selection of table rows can be distributed among partitions on different disk devices. Individual table or index rows are assigned to a partition according to a semantic or to a round-robin partitioning strategy.

This strategy allows breaking up large data sets into smaller, manageable chunks, known as partitions. Data in each partition can be processed independently of other partitions. DBA tasks such as update statistics, index creation, truncate, reorg, dbcc's, bcp, etc., can be broken down into smaller sub-tasks. This represents a divide-and-conquer strategy to manage large and increasing volumes of data

Semantic partitioning strategies use the data values in specified key columns in each row to determine the partition assignment of that row. The round-robin partitioning strategy assigns rows randomly without reference to data values.

Partitioning strategies are:

- Hash partitioning (semantic) – a system-supplied hash function determines the partition assignment for each row.
- List partitioning (semantic) – values in key columns are compared with sets of user-supplied values specific to each partition. Exact matches determine the partition assignment.
- Range partitioning (semantic) – values in key columns are compared with a user-supplied set of upper and lower bounds associated with each partition. Key column values falling within the stated bounds determine the partition assignment.
- Round-robin partitioning – rows are assigned randomly to partitions in a round-robin manner so that each partition contains a more or less equal number of rows. This is the default strategy.

You can create partitions when you create a table or index using the CREATE TABLE and CREATE INDEX, ALTER TABLE can change the partitioning strategy. You can add a partition to an existing table with add partition. ASE 15.0 implements two different types of indexing - local and global indexes. Local indexes are good for parallelizing queries. For implementing uniqueness, global indexes have to be used.

Step 1.5: Determine Your ASE Database Device Attributes

Prior to ASE 15.0, Sybase recommended that the *dsync* setting be turned on for devices initialized on UNIX operating system files. The dsync flag ensures that

writes to the device file occur directly on the physical storage media. Each write will wait for both the file data and file status to be physically updated, so that ASE can recover data on the device in the event of a system failure.

This functionality, however, yields much slower performance during write operations, when compared to using raw partitions.

In ASE 15, the `directio` parameter allows you to configure ASE to transfer data directly to disk, bypassing the operating system buffer cache. File system devices with `directio on` perform IO in the same manner and provide the same performance benefit as raw devices .

The `directio` option is a feature of the file system in which writes directly go to the storage device. Direct I/O is invoked by opening a file with the `O_DIRECT` flag. The direct I/O parameter is used with ***disk init***, ***disk reinit***, and ***sp_deviceattr*** commands.

In summary, using the `directio` option is recommended whenever file system devices are used.

Advantages over *dsync*:

- No double buffering
- Improved performance for write and cold reads since the overhead of the FS cache is eliminated
- Gives performance on par with raw with the manageability of a file system (significantly better than ***dsync***),

Note: The `directio` and `dsync` parameters are mutually exclusive. If a device has `dsync` set to "true," you cannot set `directio` to "true" for this device. To enable `directio` for a device, you must first reset `dsync` to "false."

Step 1.6: Review Any Trace Flags Used at Startup

ASE 15 changed the way many trace flags are used. Many trace flags that you may have been using in your run server file (e.g. `RUN_<servername>`) have been changed. Some have been removed while others are now controlled via the `sp_configure`, `sp_dboption` or other mechanisms.

To determine which trace flags you may be using, you should do both of the following:

- Review your run server file(s) used for your production servers and make a note of any trace flags in the file. These flags will be following the **`-T`** flag.
- Run the following commands on your production ASE prior to upgrading:

```
1> dbcc traceon(3604)
2> go
1> dbcc traceflags
2> go
```

This command will list any traceflags that are enabled for that server & session. If you are running your pre-ASE 15.x environment(s) with trace flags, refer to you ASE 15.x documentation, or check with Sybase Technical Support for information on how to move forward.

Step 1.7: Ensure You Use the Latest ASE Release and ESD

Given the continual improvements being made to ASE, both in new features (For example, the MDA tables which are now installed by default in 15.0.2 and are vital in identification and analysis of poorly running queries) and new bug fixes (many performance issues resolved), it is imperative that when you begin your migration, that you be sure to use the most recent, feature rich and stable version of ASE 15.0.2 or above should be the minimum installation version.

To be sure and do so, reference the chart at the following link and be sure to access the most recently released software that is available on your platform. The link is below:

<https://login.sybase.com/login/userLogin.do?referer=http%3A%2F%2Fdownloads.sybase.com%2Fswd%2Fbase.do%3Fclient%3Dsupport>

Step 1.8: Create a Tactical/Project Plan

Having determined the specific upgrade method in Step 1.1, develop a Tactical Plan, that outlines the specific steps to be taken, dates and owner/responsibilities assigned for each step. Include any inter-step dependencies and rollback plan in case something goes wrong. As part of the Tactical Plan, evaluate all aspects of the environment that might be affected by the upgrade/migration, for example:

- Evaluate the compatibility/certification of 3rd party products
 - You may need certification/upgrades to the client interfaces like OCS, ODBC, jdbc etc.
- Evaluate interfaces with upstream/downstream applications to determine impact
 - Review versions and feature compatibility
 - The interfaces may need to be upgraded as well
- Evaluate data/batch needs for all the interfaces
 - Downtime required for migration/upgrade and post migration checks/tests
 - Changes to system functions/commands affecting the upstream / downstream applications
 - Pay close attention to data-type conversion and format issues
- Evaluate specific ASE features in use for special needs
 - CIS, JAVA, XML, Replication, Parallelism, DTM/ASTC/rpc etc.
 - Review default settings and document any extra/special steps required to ensure that the functionality and data are not impacted
- Evaluate Rollback strategy
 - Include dump and load times required to restore the system to pre-migration state.
 - Set clear "GO-no-GO" decision timeframes

Step 1.9: Build a Test Plan

While the Tactical plan covers steps to be followed on migration day, the test plan must cover functional and performance testing of the critical parts of the application environment. It should also define success or failure criteria for each critical module. Continued testing should be followed until all the criteria can be met. Knowing what and how to test will avoid business impacting surprises in production.

- Evaluate the kind of test strategy you want to follow. Select an appropriate method for testing the application. Several different strategies are documented under Chapter 6 Ensuring Stability and Performance --- Summary of testing techniques, at the following link:
http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc34982_1500/html/mig_gde/X21853.htm
- Identify the functional areas of the application that are critical to the success of the migration. These could include:
 - Periodic batches (nightly, weekly, monthly, etc).
 - Functional modules including but not limited to :
 - Online functionality
 - Upstream/downstream interfaces, data feeds etc.
- Determine success criteria for each module to determine acceptable variances in performance or resource utilization after the upgrade/migration
- “Single user” functionality testing should be done for all (or at least the critical) modules. The plan must include at least one end-to-end test of all the critical modules identified.
- Plan for a second test cycle using any later ESDs released by Sybase since the start of testing. These may include fixes for issues encountered during testing saving valuable time later.

Pre-Migration Steps

Step 2.1: Record Base Line Performance

In order to determine how well ASE is performing after the upgrade, it is crucial to have a quantitative understanding of how it is performing prior to the upgrade. One way of measuring this base line performance is to execute your key stored procedures and queries in a test mode and to measure the amount of time or logical IOs they require to complete.

There are various ways of accomplishing this. You can use some canned stored procedures to query the MDA tables, you can use a "wrapper" type of stored procedure that executes and times your compiled objects or you can use your own method of measuring performance. The important thing is that you use some method of quantitatively measuring how your system runs, at as low a level as possible, before you begin the upgrade.

The MDA and Wrapper stored procedures are available at the following links:

[MDA Link](#)

[Wrapper Link](#)

Step 2.2: Generate and Save Abstract Plans

Since ASE 12.0, abstract plan capture can be used to intercept client-submitted SQL text. This text is stored in the *sysqueryplans* table along with its abstract query plan. The captured query text can be retrieved directly from this table. In ASE 15, the new feature of query metrics capture uses the same underlying mechanism to capture execution metrics such as I/O counts and execution times, along with the SQL text. These metrics are also stored in *sysqueryplans*, but should be accessed through a view named *sysquerymetrics*. Sybase recommends that you capture and save query plans from your prior version for later comparison or import, if necessary into your 15.x server. Keep in mind that many of the Abstract Plans will not import to 15.x because of the optimizer differences.

For more information about query metrics capture, see the ASE Performance and Tuning Guide, volume "Query Processing and Abstract Plans".

Step 2.3: Check Consistency of Master and All User Databases

General Consistency Checks

Attempting to upgrade a corrupted database is a recipe for a failed upgrade. Therefore, it is strongly recommended that sufficient consistency checking be completed prior to attempting the upgrade to ASE 15.x. More specifically, we strongly recommend running ***dbcc checkstorage*** on all user databases as well as on the master database. Of course, any issues reported by ***dbcc checkstorage*** should be resolved before moving on.

If you prefer not to run ***dbcc checkstorage***, it is probably sufficient to run ***dbcc checkdb*** and ***checkalloc*** on the master and all user databases.

Check System Tables

You should manually review the *sysusers* and the *sysroles* tables to ensure that any manually inserted user or role ID's are within the normal Sybase user range. If the upgrade process attempts to add a new user or role and one already exists with the chosen ID, then the attempt will fail and the entire upgrade could be in question. This can occur because; in ASE 12.5.x some of the user roles (such as

dtm_tm_role, ha_role) were not automatically installed in user databases by ASE. If these roles were added in 12.5.x to the database as a user defined roles, the *lrid* and *uid* of this role can conflict with the system roles that are automatically created in databases by ASE 15.

```
select * from master..sysrvroles
select * from dbname..sysroles
select * from dbname..sysusers
```

The relationships are:

```
db.sysusers.uid <-> db..sysroles.lrid
db.sysroles.id <-> master.sysrvroles.srid
```

The workaround is to remove the entries of mismatched roles.

Step 2.4: Verify Sufficient Free Space in All Databases

User Databases

Make sure that you have plenty of free space within the databases to be upgraded. While the pre-upgrade utility will help verify that ample space is present, it is always advisable to have at least 25% of free space available within each user database

You can verify this by reviewing the output of either ***dbcc checkstorage*** or ***dbcc checkalloc*** (from the previous step) which report the amount of free space.

System Databases

Due to the increased space required for the system catalog, Sybase recommends that system databases such as *master* and *sybsystemprocs* be no more than about 50% full.

Step 2.5: Verify Sufficient Number of Locks Configured for Upgrade

In order to help ensure that ASE is configured with enough lock structures for the upgrade process to complete successfully, it is strongly recommended that you perform the following steps:

1. In every user database, execute the following query:

```
1> select 2 * (count(*)) + 100 from systabstats
2> go
```
2. Record the highest value returned
3. Run the following command from any database:

```
1> sp_configure "number of locks"
2> go
```
4. If the value from Step 2 above is less than the current number of configured lock structures, then increase the configured value to the value from Step 2, as shown below:

```
1> sp_configure 'number of locks', <new value>
2> go
```

Step 2.6: Bulk Copy Out Key System Tables

We recommend that you bulk copy the contents of each of the following system tables from the master database and have them available after the migration has been made.

- | | |
|------------------------|--------------------------------|
| - <i>sysdevices</i> | Optional: (depending on usage) |
| - <i>sysconfigures</i> | - <i>sysservers</i> |
| - <i>syscharsets</i> | - <i>sysremotelogins</i> |
| - <i>syslogins</i> | - <i>sysloginroles</i> |
| - <i>sysusages</i> | |
| - <i>sysdatabases</i> | |

Step 2.7: Disable Replication

You can use Sybase Central or the procedure below to quiesce a system consisting of several Replication Servers.

To quiesce a replication system:

1. Execute the ***suspend log transfer from all*** command at each Replication Server. This prevents RepAgent from connecting to the Replication Servers.
2. Execute ***admin quiesce_force_rsi*** at each Replication Server.

This command forces Replication Server to deliver all queued messages to other Replication Servers, then reports whether the system is successfully quiesced.

Quiescing occurs most efficiently if you follow the flow of the data. For example, if data flows from TOKYO_RS to MANILA_RS to SYDNEY_RS, quiesce the Replication Servers in that order.

1. Check that the Replication Server is quiescent using ***admin quiesce_check***. If necessary, repeat steps 2 and 3 until all Replication Servers are quiescent.
2. After all Replication Servers are quiescent, ***execute admin quiesce_force_rsi*** once more at each Replication Server. Check that each Replication Server is quiescent using ***admin quiesce_check***. If necessary, repeat this step until all Replication Servers are quiescent.

This step is necessary because, although a Replication Server may be quiescent, it may have recently sent messages to another Replication Server. These messages may initiate more communication between these two Replication Servers or between several Replication Servers in the replication system. Repeating steps 2 and 3 ensures that you have quiesced the entire replication system.

Step 2.8: Dump Master Database

Backing up the *master* database is the cornerstone of any backup and recovery plan. The *master* database contains details about the structure of your entire database system. It keeps track of the Adaptive Server databases, devices, and device fragments that make up those databases. Because Adaptive Server needs this information during recovery, it is crucial that you maintain an up-to-date backup copy of the *master* database at all times.

To ensure that your backup of *master* is always up to date, back up the database after each command that affects disks, storage, databases, or segments. This means you should back up *master* after performing any of the following procedures:

- Creating or deleting databases
- Initializing new database devices
- Adding new dump devices

- Using any device mirroring command
- Creating or dropping system stored procedures, if they are stored in *master*
- Creating, dropping, or modifying a segment
- Adding new Adaptive Server logins

To back up *master* to a tape device, start **isql** and enter the command, where *tape_device* is the name of the tape device (for example, */dev/rmt0*):

```
1> dump database master to "tape_device"
2> go
```

Step 2.9: Unpartition All Partitioned Tables

As a result of system table changes, and the changes made to support semantic partitioning, tables currently partitioned on pre-ASE 15.x servers will be unpartitioned during the migration process.

Prior to unpartitioning any tables, please be sure to record exactly what tables were partitioned and to what degree, prior to removing the partitions. This can be done by running the below procedure:

```
1> sp_helppartition
2> go
```

There are two ways to achieve this step. You can manually execute an ***alter table <table_name> unpartition*** command for every partitioned user table, or you can execute the following stored procedure which will do it for you:

```
1> sp_unpartition <dbname>
2> go
```

Note that the *sp_unpartition* procedure is not a normal Sybase system stored procedure but it is at the following link: [sp_unpartition](#).

Step 2.10: Consider Replacing the Update Statistics Command

One aspect of the new Query Optimizer in ASE 15.x is that it is now much more capable of analyzing more statistical attributes of a data table when it is deciding which query path to choose. In particular, prior to ASE 15.x, the optimizer only used statistics for the leading column of all multi-column/composite indices. The ***update statistics*** command was therefore designed to only create statistics for the leading column of each index.

Given this, we strongly recommend running ***update index statistics <tablename>*** whenever possible and, if feasible, to replace your ***update statistics*** commands with ***update index statistics***. This command will generate statistics on all of the columns of each composite index and will help ASE make the best decisions.

Note however that because the ***update index statistics*** command is doing more work, it will definitely take longer to update a table's statistics (at least for indices with more than one column). On very large datasets, you may need to consider alternate ways of gathering statistics due to maintenance windows and resource constraints.

Some of these methods include:

- Running ***update statistics <table> <optional index>*** to do the leading column of one or all indices as well as densities of all pre-fix permutations of columns in composite indices, then run ***update statistics <table> <column(s)>*** for all unique columns in the table that are contained in one or more indices. These can be run in parallel given sufficient CPU, memory, and *tempdb* space.
- Running update index statistics with **Sampling** percentage, but keep in mind, there are no recommended values for the sampling, you would need to test, then balance between too low a percentage rendering the statistics ineffective, and too high a percentage not yielding any significant savings in time.
- Running update statistics as needed, determined by using the ***datachange()*** function, with the understanding that the percentage can vary depending on cardinality of underlying data

Importing current statistics from a Disaster Recovery or Development server with the same data using ***optdiag***. For more information and examples, see the Migration or Administrators guide.

Step 2.11: Delete or Update Stale Statistics

In this step, you will identify old statistics that have been populated via the *update statistics*, *update index statistics*, *update all statistics* or *update statistics tablename (<columnname>)* commands.

Use the following query to identify stale statistics. The output is ordered by oldest to newest statistics.

```
use <database>
go
select "Oldest to Newest Statistics" =
    convert(varchar(40), object_name(sc.id) + "." + sc.name),
    "Last updated On " = convert(varchar,ss.moddate,0)
from syscolumns sc, sysstatistics ss
where ss.id = sc.id
    and ss.colidarray = convert(varbinary,sc.colid)
    and ss.formatid = 100
    and ss.id > 99
order by ss.moddate asc, sc.name
```

Having stale statistics may negatively impact performance because the histogram, density and selectivity values may no longer be representative of the data contained in the columns. Having current statistics or removing statistics that are no longer needed will help the optimizer choose the most efficient access plan.

The ***delete statistics*** command is provided as a tool to remove statistics not used by the optimizer. Dropping an index does not drop the statistics for that index. If the distribution of keys in the columns change after the index is dropped, but the statistics are still used for some queries, the outdated statistics can affect query plans.

Example: `delete statistics <table_name> (<column_name>)`

The ***delete statistics*** command, when used with just the table name, removes all statistics for a table, even where indexes exist, in which case, you must run ***update index statistics*** on the table to restore the statistics for the index.

Step 2.12: Quiesce All Databases

This step is to ensure that no uncommitted transactions cause any problems. Also, it helps to ensure overall consistency and speed of the process. To quiesce the database, we use the same process as is used for Cross-Platform Dump/Load:

1. Execute ***dbcc checkdb*** and ***dbcc checkalloc*** to verify the database runs cleanly.
2. To prevent concurrent updates from open transactions by other processes during **dump database**, use **sp_dboption** to place the database in a single- user mode.
3. Flush statistics to *systabstats* using **sp_flushstats**.
4. Wait for 10 to 30 seconds, depending on the database size and activity.
5. Run **checkpoint** against the database to flush updated pages.
6. Run **dump database**.

Step 2.13: Perform Migration

When you reach this step it is time to execute the method of migration that was determined in Pre-Migration Planning Step #1.

Post-Migration Steps

Step 3.1. Recreate Table Partitions

Since partitioned tables in pre-15 will be un-partitioned after upgrading to 15, you will need to re-partition.

The pre-15 syntax for partitioning remains valid, and results in the same as the 'round robin' partitioning syntax. The only exception is: alter table...un-partition is no longer allowed when the table has an index, and actually, un-partitioned tables don't really exist anymore in 15. Each table has at least 1 round robin partition.

Each partition has its own partition ID in the page header, so the new partition id needs to be re-stamped on every page for the object, so even though no data is being moved, every page of the table needs to be touched, modified and logged which involves additional time and resource overhead.

Step 3.2. Reconfigure With Additional Resources

Procedure Cache

One of the consequences of new query processing engine in ASE 15 is that ASE requires more procedure cache. This increased memory requirement applies to optimization and execution of queries. ASE has many more join methods and data access algorithms to consider than prior to version 15. This is especially true for ***allrows_dss***, which will generally tend towards consuming more procedure cache for query optimization than ***allrows_mix***, which in turn usually needs more procedure cache than ***allrows_oltp***.

It is difficult to predict exactly how much more memory ASE 15 will need compared with 12.5. As a rule of thumb, Sybase recommends to plan for sizing of the procedure cache in ASE 15 between 2 to 6 times larger than in ASE 12.5. Use the 'procedure cache' section of sp_sysmon to monitor procedure cache usage.

Procedure Cache usage limitation in 15.0.2 ESD#2

In ASE 15.0.2 ESD#2, a limitation was implemented on the amount of procedure cache that can be used by the ASE query optimizer. This is based on the setting of configuration parameter ***max resource granularity*** (settable on session level with ***set resource granularity***). This setting is an integer between 1-100, reflecting a percentage. The default setting is 10.

Tempdb Usage

ASE 15 requires increased *tempdb* space over prior versions. Part of this increase is a result of a larger work space requirement by the Query Processing layer, as there are more join methods and access algorithms in 15. There are also new features in 15 that if used, will increase *tempdb* utilization. Some examples of these features are scrollable cursors, computed columns, functional indexes, etc.

We recommend that you keep this increased utilization in mind when planning disk space, and observing I/O rates in *tempdb*. The amount of the increase will vary, depending the optimization goal used, your specific workload mix and the features you use.

Other resource usage aspects of ASE 15

One of the new aspects of ASE 15 is the feature of table partitions. Although semantic data partitioning is an optional license feature, it matters even when you

do not use semantic data partitioning: in ASE 15, each table and index always consists of at least one partition. Consequently, the configuration parameter number of open partitions must be set high enough to avoid unnecessary performance overhead in reuse of this resource.

Sybase recommends using *sp_countmetadata* to estimate the required setting for this parameter (keep in mind this is a starting point, as temp tables are not considered), then use *sp_monitorconfig* to monitor usage.

Step 3.3. Configure Statement Cache

Due to the increased complexity of query optimization in ASE 15, the query optimizer might use significantly more resources (compilation time and procedure cache) over prior versions of ASE.

So, if large numbers of identical or very similar SQL queries were are being run, where the difference lies in the search argument, enabling the statement cache as well as literal auto-parameterization (an enhancement to the statement cache in ASE 15.0.1) is recommended.

These settings may significantly reduce the time and resources spent on query optimization and therefore improve overall performance.

When the statement cache is enabled, a query's plan will be cached, so that an exactly identical query does not need to be compiled again, and the cached plan can be used instead, saving time and resources. With literal auto-parameterization enabled, this caching is extended to almost-identical queries that differ only in a constant value, and are likely to end up with the same plan.

Literal auto-parameterization can greatly improve the effectiveness of the statement cache, and therefore it is recommended to experiment with this feature if your applications use many almost-identical SQL queries that are not part of stored procedures and are not run in execute-immediate.

The statement cache is enabled sever-wide with the configuration parameter:

```
1> sp_configure "statement cache size"  
2> go
```

At the session level, the statement cache can be disabled with ***set statement cache off***.

Literal auto-parameterization is enabled server-wide with the configuration parameter:

```
1> sp_configure "enable literal autoparam"  
2> go
```

And at the session level with the set option ***literal_autoparam*** on (or off). It applies only when the statement cache is enabled.

Statement Cache Sizing: A cached statement requires approximately 1K memory in the statement cache, depending on the length of the SQL text, and 2k for the cached plan, plus a few hundred bytes of overhead per plan. There can be duplicates of the same plan if two or more users request a plan concurrently. So it depends on how many and the size of cached statements. ***Sp_sysmon*** has a

new section now for statement cache. You can monitor with *sp_sysmon* to see the utilization and effectiveness to size appropriately.

Step 3.4. Recreate Optimizer Statistics

Given that the ASE 15 query optimizer is much more able to use statistical information describing columns in a query than in previous versions, it is extremely important to give the optimizer all the data that it can use. To check for missing statistics, ASE 15.0.2 provides a new *set* option, ***show_missing_stats on***. This option will display a short message for each column or column group where availability of statistics would have helped the optimizer. Below are examples of how it may be used both by a user with *sa_role* and a user without *sa_role*:

User with *sa_role*:

```
set option show_missing_stats on
dbcc traceon(3604)
go
```

This option requires either Trace Flag 3604 or 3605 for all users!

User with *sa_role* granting permission to DevUser:

```
grant set tracing to DevUser
go
```

User DevUser:

```
set option show_missing_stats on
dbcc traceon(3605)
go
```

Trace Flag 3605 sends messages to the error log.

In addition to using the new set option above, we also recommend making the following configuration related changes:

```
1> sp_configure "number of histogram steps", 200
2> go
1> sp_configure "histogram tuning factor", 10
2> go
```

Estimate Ideal Step Counts

For all user tables on the ASE in question, estimate the ideal new value for "step count" using the following process.

1. For each large/key table, use the *data_pages* function to determine the number of data pages that the table currently occupies. Then divide this number by 10,000. The rationale for doing this is that Sybase estimates that each table should have *about* 1 histogram step per 10,000 data pages, never exceed 2,000 steps.

```
1> select data_pages(db_id(), object_id("<table_name>") ) / 10000
2> go
```

2. If the result from the above query is:
 - i. > 2,000, use 2,000 histogram steps for indices on that table.
 - ii. < 20, use 20 histogram steps for indices on that table.

3. For each key table, execute the following command:

```
1> update index statistics <table_name> using <# steps> values
2> go
```

Step 3.5. Evaluate New Optimization Goals

Optimization Goals are an important new concept in ASE 15.0. The Query Optimizer is given a hint as to the type of queries that you run, and makes optimization choices on that basis. The goal you choose will affect plan choices and the time spent on optimization.

The three supported optimization goals are detailed below:

- ***allrows_mix*** – Best for mixed workload applications. OLTP style, simple point queries with some DSS style complex queries NLJ, SMJ, sort/hash based grouping and parallelism support.
- ***allrows_oltp*** – For systems using primarily OLTP style simple point queries. NLJ, sort based grouping.
- ***allrows_dss*** - Primarily for Decision Support (DSS) style complex queries. NLJ, SMJ, HJ sort/hash based grouping, bushy tree plans and parallelism.

The default Optimization Goal (*allrows_mix*) is intended to accommodate the majority of queries efficiently, however, depending on your processing mix, you may need to adjust the optimization goal at the server, session or individual query level.

Examples:

Server Level: ***sp_configure "optimization goal", 0, "allrows_dss"***

Session Level: ***set plan optgoal allrows_mix***

Query Level: ***select * from blah plan "(use optgoal allrows_oltp)"***

Individual optimization algorithms that are used under various optgoals can be turned on and off at the session level, overriding the server optgoal.

Examples:

set nl_join ON/OFF

set parallel_query ON/OFF

set merge_join ON/OFF

If you run into problem queries post upgrade, it will be useful to compare captured plans from the prior version and adjust strategies accordingly on ASE 15.x.

There are other options that can be enabled/disabled that may be useful in very specific situations. Sybase Technical Support can advise on their usage.

Keep in mind that there may be behavior changes in optgoals between ESDs. Consider re-visiting your strategy, should you run into any issues.

Step 3.6: Understand The Role and Usage of Compatibility Mode

In most cases, ASE 15.0 will perform better than prior versions, especially for complex queries. However, there may be cases where a query does not perform as well as it did prior to 15.0. For these cases, we have introduced *Compatibility Mode*.

For example, if you will not be using new features in ASE 15.0, or your application is very sensitive to increased query compilation time, and statement cache or stored procedures don't address this as well as required, Compatibility Mode provides a way to generate and execute a similar query plan as in pre ASE 15,

yielding similar performance characteristics. This can also be useful for applications which have been tuned for the ASE 12.5 optimizer, possibly preventing the ASE 15 optimizer from working as well as 12.5. Compatibility Mode can move re-tuning of applications for the ASE 15 optimizer out of the migration process, taking advantage of ASE 12.5 tuning efforts.

To enable Compatibility Mode

Server-wide: (not usually recommended)

```
sp_configure 'enable compatibility mode', [1 | 0]
```

Session-level: (recommended)

```
set compatibility_mode [on | off]
```

The session setting will override the server-wide setting.

Compatibility Mode is only available in ASE 15.0.3 ESD #1 and later releases. There are other restrictions on whether Compatibility Mode can be used, and at what level. See Table 3.1 in the 15.0.3 Migration Guide for specifics:

<http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc00967.1503/html/MigrationGuide/title.htm>

SQL statements will be processed in one of three modes:

- Full Compatibility Mode
Optimized by an ASE 12.5.4-like optimizer, code generated into ASE 12.5.4 query plans and executed by the ASE 12.5.4 query execution engine.
- Restricted Compatibility Mode
Optimized by an ASE 12.5.4-like optimizer to generate an abstract query plan that will be processed by the ASE 15 optimizer and query execution engine.
- Default Mode
Optimized by the ASE 15 optimizer and executed by the ASE 15 query execution engine.

To see which mode was used to execute the previous query:

```
select @@qpmode
```

0– Non-optimizable query (create index, set rowcount, ...)

1– Full Compatibility Mode.

2 – Restricted Compatibility Mode.

3 – Compatibility Mode could not be applied.

Available trace flags:

441 - Allow all parameterized literals in Compatibility Mode.

446 - Skip Restricted Compatibility Mode

477 - Prints the following :

- Prints the reason why a query was not processed using Full Compatibility Mode.
- Prints whether a query was chosen for Full Compatibility Mode or not.
- Prints the query text (if available).
- Use trace flag 3604/3605 to direct the output.

Enabling compatibility mode may not always have an effect. There will be no effect on plans that are already cached.

Step 3.7: Gather Post Migration Baseline Performance Info

Now that your ASE software has been migrated, it is time to evaluate how it is performing. You should (at a minimum) use the same procedures that you used in Step 2.1 to gather post-migration performance information and then compare it with the pre-migration numbers.

If you find one or more queries, stored procedures or other compiled objects that have worsened in performance, we recommend that you:

- a) Use the new QPTune tool, discussed below
- b) Review Steps 2.9, 2.10 and 3.4, regarding table and index-level statistics and ensure that the tables and indices involved all have up-to-date statistics.
- c) Review Step 3.5 and re-run the query/procedure under each of the different optimization goals to see if may generate a better plan than the others.

QPTune

Starting in ASE version 15.0.3 ESD#1 a new tool has been introduced called QPTune, which stands for Query Processing Tuning. This is a query analysis tool which identifies *poorly performing* queries. There are two basic functions in QPTune, 1) capturing missing statistics and, 2) query tuning.

The first step is the statistics advisory function. This will identify and optionally create missing statistics. Statistics are absolutely critical to the ASE 15 optimizer choosing the correct plans. Once all the statistics are in place the second step is to do tuning on any queries still not performing optimally. QPTune does this by capturing query text and a variety of metrics which may include elapsed time, logical or physical I/O.

The most important part of the tool is that it allows you to compare metrics across multiple runs, and to tune ASE at either the server level or at the query level. This part of the process is very quick and gives valuable information on which portions of the query perform best using which optgoal.

If you continue to have slow performing queries that you are unable to resolve, then it may be time to contact Technical Support. Before you do, however, it will save time if you and make the entire process faster if you can generate some basic trace information for Technical Support to review.

If you have narrowed a performance problem down to a particular stored procedure or query, then it is a great idea to generate the following information *prior* to contacting Sybase Technical Support:

- 1) Optdiag Statistics for each table involved in the query. For example:
unix% optdiag statistics <dbname>..<tablename> -Uxx -Pxx > outputfile
- 2) Sp_help for each table (or DDL for table & indices)
- 3) Set the following and run the query in question:
set statistics time, io on
set showplan on
set option show_missing_stats on

```
set option show_ljo_costing on
set option show_search_engine on
```

The **sa_role** is required for some of the options above. Note for stored procedures, you may need to drop and recreate the procedure first in order to generate output from some of the above trace commands.

- 4) Output from `select @@version`
- 5) Copy of ASE's .cfg file

This information may not be all that Tech Support needs, but it will most certainly get the diagnostic effort off to a fast start.