New Features Guide

Adaptive Server® Enterprise

15.7 ESD #4

DOCUMENT ID: DC00641-01-1574-01
LAST REVISED: May 2013

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

CHAPTER 1 Adaptive Server ESD #4.2

Improved Rollback Reporting

Adaptive Server® 15.7 ESD #4.2 onwards provides enhanced reporting using the **kill with statusonly** command.

The **kill with statusonly** command provides information about the amount of log space scanned, and to be scanned, for a full transaction rollback. It differentiates different types of rollbacks, and reports a subset of information in cases where the rollback is not a full transaction rollback.

CHAPTER 2 Adaptive Server ESD #4

Changes to sp_addthreshold and Roles

System and user defined roles that are active when a threshold procedure is created are preserved in `systhresholds`.

In Adaptive Server 15.7 ESD#3 and earlier, **sp_addthreshold** preserves the system roles that are active when the threshold procedure is created. However, only directly granted system roles that have not been revoked since the threshold was created are activated when the threshold fires. Indirectly granted system roles and user-defined roles are not activated.

This restriction has been relaxed and all system and user defined roles active when the threshold procedure is created are preserved in `systhresholds`. All system and user defined roles that the user had at time of creating the threshold and that have not been revoked will be activated when the threshold fires.

Configuring the Number of Rows Waiting in the Send Buffer

Adaptive Server assembles data and accumulates rows that wait in the send buffer, but does not send them until the buffer has reached its configured size.

If the packet size is large and if the query is one that takes a long time to execute, these rows may sit for a while before they get sent. When Adaptive Server does send the rows, they arrive at the client simultaneously, and the client then spends time processing them individually.

For large data transfers, the larger the value for the network packet size the more efficient the overall transmission, but there is a time lag during which the client is idle while the Adaptive Server waits for the first buffer to fill. Reducing the size of the network packet size can improve response time, but at the cost of efficiency.

The **number of early send rows** and **early row send increment** configuration parameters allow you to configure the number of rows that sit in the send buffer, and can reduce the amount of wait time:

- `number of early send rows` – initial number of rows Adaptive Server sends to a client.
- `early row send increment` – number by which Adaptive Server increments the value for number of early send rows when the packet fills, but there are still rows waiting in the send buffer

number of early send rows allows Adaptive Server to send results to the client before the packet has reached its configured size. Each subsequent packet increments the number of rows until the packets has reached its configured size (as set by the **network packet size** parameter). This gives the benefit of the reduced latency for the first few rows without losing overall transmission efficiency. The cycle continues for each new result set, starting with smaller packets, then growing to larger ones.

For example, If you set the value for **number of early send rows** to 2, Adaptive Server sends the rows in the buffer when it contains 2 rows. The client receives the initial rows more quickly and processes them while Adaptive Server is working on the next set of rows in the query.

If you subsequently set the value for **early row send increment** to 20,000, Adaptive Server sends the second set of rows in the buffer when it contains 2 + 20,000 rows and adds the value 20,000 to the value for number of early send rows for each subsequent buffer: that is, 2 + 20,000 + 20,000 for the third buffer and so on.

early row send increment

Table 1. Summary Information

Default value	2147483647
Range of values	1 – 2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	Network Communication

Configures the additional number of rows sent in the second and subsequent packets of a result set (subject to the maximum packet size).

number of early send rows

Table 2. Summary Information

Default value	0
Range of values	0 – 2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	Network Communication

Configures the number of rows Adaptive Server sends to the client in the first packet of a new result set .

Remote Backup Server Maximum Name Length

The remote backup server name length limit for **dump** and **load** commands has been increased from 30 characters to 255 characters.

Backup Server Supports the mountwait Parameter

Backup Server supports the Tivoli Storage Manager **mountwait -M** parameter for the **dump database**, **dump transaction**, **load database**, and **load transaction** commands.

Backup Server supports the Tivoli Storage Manager **mountwait -M** parameter for the **dump database**, **dump transaction**, **load database**, and **load transaction** commands.

The syntax is:

```
[dump database | dump tran] to "syb_tsm::-M::object_name"
```

and:

```
[load database | load tran] from "syb_tsm::-M::object_name"
```

For example:

```
dump database to "syb_tsm::-M::object_name"
load database from "syb_tsm::-M::object_name"
```

The **mountwait -M** parameter is disabled by default.

Support for Query Plan Pinning

The configuration option **dynamic SQL plan pinning** has been introduced in Adaptive Server 15.7 ESD #4 to improve Adaptive Server performance by reducing the time spent by server connections waiting for access to the query plan manager.

*About **dynamic SQL plan pinning***

When a program sends a dynamic prepared SQL statement to Adaptive Server, Adaptive Server internally creates a stored procedure containing the prepared SQL statement. This stored procedure is similar to a user-created stored procedure, except that it has no system catalog entries, that is, it exists in memory only. The first time that the prepared statement is executed, a query plan is compiled and executed. At the end of execution, the query plan is released to the query plan manager in Adaptive Server for re-use. When the same statement is executed again, Adaptive Server calls to the query plan manager to see if a query plan is available, and if so, the query plan manager returns the query plan to the Adaptive Server connection to execute. At the end of execution, the query plan is returned to the query plan manager.

All Adaptive Server connections can access the query plan manager to ask for available query plans as well as to store new query plans that they have compiled and finished executing. However, only one Adaptive Server connection can access the query plan manager at a time, to avoid multiple connections getting the same query plan at the same time (only a single connection can execute a given query plan at a time). Each connection will access the query plan manager twice for each dynamic prepared SQL statement that it executes: Once to acquire the query plan and once to release it for re-use.

In a highly concurrent environment (many Adaptive Server connections running dynamic prepared SQL statements at the same time), Adaptive Server performance may be degraded because each connection must wait its turn to access the query plan manager when retrieving or storing a query plan. The dynamic SQL plan pinning feature was introduced to improve Adaptive Server performance by reducing the time spent by server connections waiting for access to the query plan manager. When query plan pinning is enabled, each Adaptive Server connection compiles a query plan for each dynamic prepared statement that it executes and does not release it to the query plan manager for re-use. Each connection keeps all query plans that it compiles for its own exclusive re-use, thus, avoiding the need to access to the query plan manager on the second and subsequent executions of the same dynamic prepared SQL statement.

Enabling ***dynamic SQL plan pinning***

To enable **dynamic SQL plan pinning**, use:

```
sp_configure 'dynamic SQL plan pinning', 1
```

Once enabled, dynamic SQL plan pinning will only take effect if one of the following options is also enabled:

```
sp_configure 'streamlined dynamic SQL', 1
```

```
sp_configure 'enable functionality group', 1
```

The Impact of ***dynamic SQL plan pinning***

Because each Adaptive Server connection keeps its own copy of each query plan and query plans are created from the procedure cache memory pool, this pool may need to be configured to a larger size when **dynamic SQL plan pinning** is enabled. Exactly how much larger the procedure cache needs to be depends upon the number of concurrent Adaptive Server connections executing dynamic SQL prepared statements: In extreme environments with small procedure cache sizes, 2-to-3 times larger may be required.

Note: By default, **dynamic SQL plan pinning** is not enabled.

Support for AF_UNIX

Adaptive Server version 15.7 ESD #4 and later support the AF_UNIX named pipe communication, which avoids some of the overhead incurred by the TCP protocol.

Include the AF_UNIX protocol in the interfaces file; the syntax is:

```
master afunix unused //host_name/pipe_name
```

where:

- *host_name* is the name of the local machine on which you start Adaptive Server.
- *pipe_name* uses this format:

  ```
  /folder_name/pipe_name
  ```

  You can use named pipes only for local communications.

Adaptive Server configured for AF_UNIX supports these clients:

- **isql**
- **bcp**
- **optdiag**
- Clients that use Sybase® ODBC

Adaptive Server configured for AF_UNIX does not support:

- Local clients using jConnect™ applications that are not local
- Local clients using JConnect to connect to the dataserver

Note: Using Adaptive Server to Adaptive Server RPCs requires at least one entry in the interfaces file be a TCP entry.

If the interfaces file includes an invalid AF_UNIX entry followed by a valid TCP entry, Adaptive Server listens on the valid TCP port. Adaptive Server does not start unless there is a valid entry in the interfaces file.

Adaptive Server uses information in the interfaces file entry to create a named pipe in the specified location. AF_UNIX supports communication between the client and server only if both are present on the same machine.

Use the actual hostname of your local machine (you cannot use the network alias of the hostname). Use the **hostname** or **uname -n** UNIX commands to return the hostname.

This example shows a portion of a `$SYBASE/interfaces` file with a database named MYSERVER and an AF_UNIX entry for a named pipe named `/tmp/big_pipe` on a host named `big_server`:

```
MYSERVER
    master afunix unused //big_server/tmp/big_pipe
    query afunix unused //big_server/tmp/big_pipe
```

Managing Named Pipes

Use **sp_listener** to manage an AF_UNIX interfaces file entry. The syntax to start a named pipe listener is:

```
sp_listener "start", "afunix:machinename:pipename"
```

This example starts a connection on the machine named `big_server` and creates a pipe named `/tmp/big_pipe` in the /tmp directory of this machine (Adaptive Server must be running on machine `big_server`).

```
sp_listener "start", "afunix:big_server:/tmp/big_pipe"
```

The syntax to stop a named pipe listener is:

```
sp_listener "stop", "afunix:machinename:pipename"
```

This example stops a previously started named pipe connection (named `big_pipe`) on the machine named big_server:

```
sp_listener "stop", "afunix:big_server:/tmp/big_pipe"
```

Index