VTS Server Guide

Release 5.2

Table of Contents

Preface

What This Manual is About

This manual describes the tableBASE Virtual Table Share (VTS) interface.

Whom This Manual is For

This manual is intended for individuals responsible for the operation of the VTS environment.

What You Should Know to Use This Manual

You should be familiar with tableBASE Concepts and Facilities, the tableBASE Programmer's Guide, the tableBASE Installation Guide and the batch MVS environment.

Additional tableBASE References

This manual is one of several that describe tableBASE; others include:

- Concepts and Facilities Manual
- tableBASE Installation Guide
- tableBASE Administrator's Guide
- tableBASE Programmer's Guide
- tableBASE Batch Utilities Manual
- tablesONLINE/CICS Users Manual
- tablesONLINE/ISPF Users Manual
- Quick Reference Guide

Chapter 1
Overview

The VTS interface enables multiple MVS applications to obtain information from tables in a shared memory environment, enabling each application to share an identical image of a table. As with other tableBASE components, tables to be managed by VTS reside in a tableSPACE region (TSR). The VTS TSR is an MVS/ESA shared dataspace that contains the tables to be shared among many applications.

VTS relies upon the MVS subsystem interface mechanism. Thus, in order to create a tableBASE Virtual TSR, a VTS Server subsystem must be started.

In a parallel sysplex environment, a VTS server subsystem with the same name must be started on all images that need to have access to the shared tables.

Access to the tableBASE VTS Interface is supported under MVS Batch, TSO, IMS, and CICS. Tables managed by VTS can be accessed without changing tableBASE calls in existing programs.

System Description

Figure 1-1 on the following page describes the tableBASE VTS Interface. There are two main processing components:

- VTS Server - the virtual TSR owning task
- VTS Client - the user application.

A VTS Server is identified by a unique four-character subsystem name defined to MVS. This name provides a logical connection between the Virtual TSR and the user application.

The tableBASE VTS Interface also has the following data components:

- Virtual tableSPACE Region (TSR) - an MVS/ESA dataspace that contains the in-memory copies of the tables
- On the Server side
  - tableBASE libraries (e.g., MAINLIB)
  - a dataset defining VTS Server input parameters (DDNAME of TBOPTV)
  - a dataset defining VTS Server commands (DDNAME of CMD)
  - operator console
- On the Client side

- a dataset defining local TSR and Client options (DDNAME of TBOPT)
- a local TSR.

The VTS Interface has a feature that allows the contents of a Virtual tableSPACE Region to be dynamically reconfigured by the issuance of an operator command or submission of a special job. This feature is known as REFRESH and is a sub-function of the Server address space.

You may have several differently named VTS Servers active simultaneously, allowing you to access multiple shared Virtual tableSPACE Regions, each of which may be as large as 2 gigabytes. Clients may be any mix of application programs using the batch, TSO, CICS or IMS interfaces.

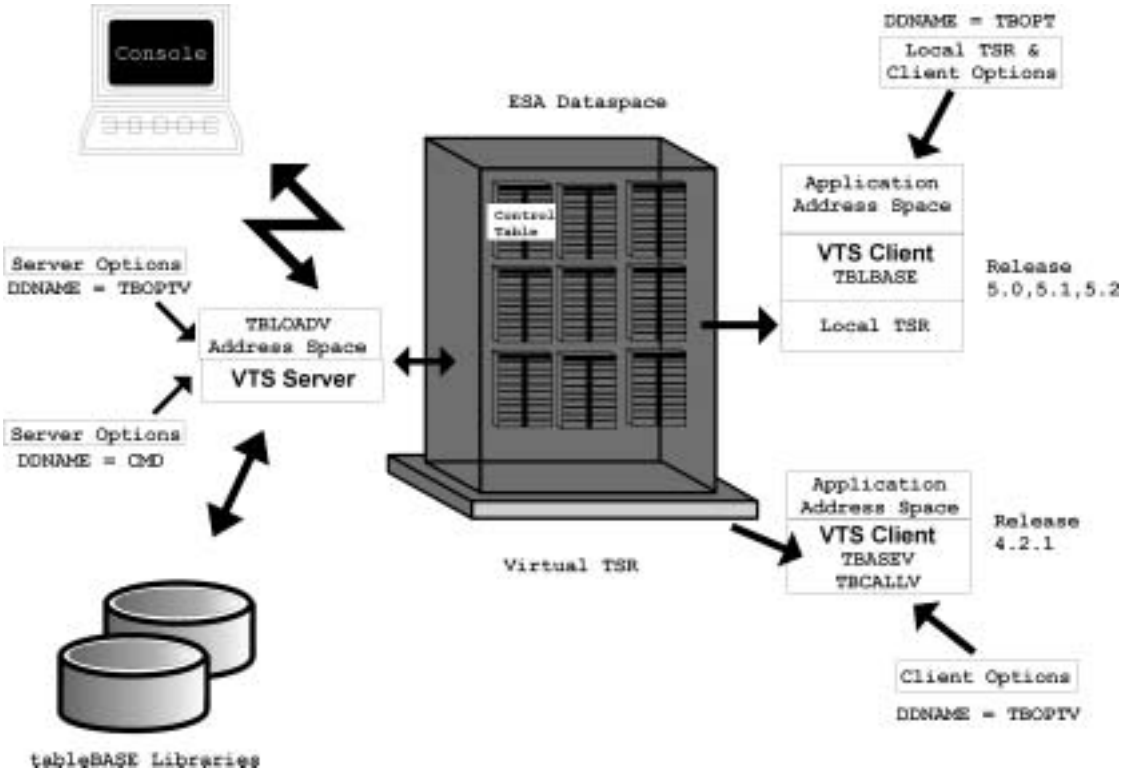

Figure 1-1 The tableBASE Virtual Table Share Interface

VTS in a Parallel Sysplex Environment

Each MVS image in a parallel sysplex that requires access to tableBASE tables must have a VTS Server active.

If a VTS client, such as a transaction in a CICSplex spanning multiple MVS images of a parallel sysplex, requires identical access to a VTS server on more than one MVS image, the VTS servers must be initialized identically on

each of those images. See “Parallel Sysplex Considerations” in Chapter 2, *Server Operations*, for implementation details.

However, if the clients accessing tables through the VTS server on one image use different tableBASE libraries from those accessed on other images, then the planning and implementation of the servers is no different from multiple VTS servers on a single image.

Chapter 2
Server Operations

This chapter discusses considerations for the operation of the VTS Server. It is intended for use by tableBASE administrators for planning purposes, and by those responsible for day to day computer operations for reference.

The chapter discusses the following topics:

- Starting the VTS Server
- Refreshing the Server
- Shutting the Server Down
- Restarting the Server
- Parallel Sysplex Considerations
- Operations JCL

VTS Server Startup

The VTS Server may be initiated either as a batch job (by submission of the startup JCL) or as an MVS Started Task (STC). This job/STC would usually be included in your installation's daily startup procedures. In the startup JCL shown below, two datasets - TBOPTV and CMD - are of special importance and are described here. If identical VTS servers are to be initiated on multiple images in a parallel sysplex environment, ensure that the TBOPTV options and CMD file input are identical for all VTS server startup procedures. See "Parallel Sysplex Considerations" in this chapter for additional information.

TBOPTV

The TBOPTV DDNAME is used to supply override startup options to the VTS Server control program, TBLOADV.

Valid parameters are:

VTSNAME=xxxx	Where 'xxxx' is the four-character MVS subsystem name to be used to identify this VTS Server. VTS Clients use this name to make a logical connection to the Server. The default value is DKL1.
VTSTEMP=xxx	Where 'xxx' is either YES or NO. YES allows the VTS Server to dynamically define to MVS the subsystem named by the VTSNAME parameter. NO requires that the subsystem named by the VTSNAME parameter be previously explicitly defined to MVS. The default value is YES.
VTSSIZE=nnnnnnn or nnnnnnnK or nnnnM or nG	Where 'n' is numeric. This specifies the amount of virtual storage (ESA dataspace size) to be allocated for holding table data. It needs to be large enough to contain all the tables that are to be made available by the server. If the parameter ends with K, M, or G, then that number of Kilobytes, Megabytes or Gigabytes is allocated. The default as supplied by Data Kinetics is 2M.
VTSNMTAB=nnnnn	Where 'n' is numeric, ranging from 1 to 65,535. This value should be larger than the maximum number of tables that the server will have open concurrently. The default value is 500. A calculation to give an accurate minimum value for VTSNMTAB is: 1 + T + A + LA where: T is the total of concurrently open tables A is the total of concurrently open alternate

indexes for those tables

LA is the largest number of alternate indexes for any open table that has alternate indexes.

CMD

The CMD DDNAME is used to pass Server control commands to the VTS Server. It is this input that controls exactly which tables are to be made available to Clients. The command file is processed at server startup, at a refresh operation and, optionally, at server termination. See Chapter 3, *Server Commands*, for details of the various commands that may be used.

VTS Startup Sample JCL

JCL must be prepared to startup the VTS server. In this example, the server to be started will be known as DKL1; as part of its initialization, the DKL1 Server reads its CMD file, and carries out the commands found there. In this example, it opens all tables found in two libraries.

```
//LOADCMDS EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT2   DD DSN=*your.prefix*.VTS.DKL1.VCMDS,DISP=SHR
//SYSUT1   DD *
*  Open all tables and invoke any alternates
ML,BAS1LIB,ALT1LIB
AR,*
//*
//TBLOADV  EXEC PGM=TBLOADV,COND=(0,LT),TIME=1440
//STEPLIB  DD DISP=SHR,DSN=your.authorized.VTS.loadlib
//SYSOUT   DD SYSOUT=*
//MAINLIB  DD DSN=*your.prefix*.VTS.DKL1.MAINLIB,DISP=SHR
//BAS1LIB  DD DSN=*your.prefix*.VTS.DKL1.BAS1LIB,DISP=SHR
//ALT1LIB  DD DSN=*your.prefix*.VTS.DKL1.ALT1LIB,DISP=SHR
//TBOPTV   DD *
 VTSNAME=DKL1
 VTSSIZE=20M
 VTSTEMP=YES;
/*
//CMD      DD DSN=*your.prefix*.VTS.DKL1.VCMDS,DISP=SHR
```

Note that the SYSUT2 dataset, changed by the LOADCMDS step, is the Server's CMD dataset.

VTS Server Refresh

Once the VTS Server control program has completed initialization, it goes into a wait state. VTS Clients may access the available table data without any further involvement from the Server program. However, the Server program can be "woken up" to perform a "Refresh" operation at any time. If identical VTS servers are active in multiple MVS images in a parallel sysplex, "Refresh" operations must be coordinated on all images. See "Parallel Sysplex Considerations" in this chapter for further details.

When triggered to perform a refresh, the server control program reopens its CMD input file and processes all the Server control commands in that file. See Chapter 3, *Server Commands*, for details of the available commands. Note that a refresh does **not** completely re-initialize the Server. The effects of the CMD input at startup and any subsequent refreshes are cumulative. It is therefore expected that the contents of the Server's CMD input file will be changed (e.g., by using IEBGENER) prior to triggering a refresh.

To ensure consistent access to VTS tables, we recommend that all programs and transactions accessing tables being refreshed be disabled until the refresh is complete. If this is not practical, see "VTS Refresh Precautions" in this chapter for considerations.

During refresh processing, tables are updated without any delay in client processing, if possible. If the refresh process causes a table to be reloaded, the previous version of the table is kept available for access until the new one is built. If the refresh causes the VTS server to re-organize the TSR, client requests must be inhibited while the server does the re-organization. To allow for this, the server waits for 2 seconds for active client requests to complete referencing the TSR. Any new VTS client requests are queued until the VTS internal processing completes. If a queued request is from CICS, the entire CICS region could wait for up to two seconds.

To avoid the possibility of queued requests, ensure that the VTSNMTAB parameter value in the TBOPTV file is sufficiently large. A formula for calculating a least value for VTSNMTAB is given above under "VTS Server Startup".

In a parallel sysplex environment, even if a VTS refresh is initiated on the servers of all MVS images at the same time, varying workloads could cause significant differences in the refresh completion times. If transactions accessing a VTS table are dispatched on different images in quick succession, there is a possibility that a transaction could access back-level data on an image that has not yet completed the refresh.

The refresh process can be triggered in one of two ways, as follows:

- By issuing, from an MVS console, a modify command to the VTS Server job/STC of the form:

  MODIFY/F job/STCname,REFRESH

- By running program TBPOSTV in batch. Sample JCL to run this program to trigger a refresh follows:

```
//REFRESH    EXEC PGM=TBPOSTV
//STEPLIB    DD     DISP=SHR,DSN=your.authorized.VTS.loadlib
//SYSOUT     DD     SYSOUT=*
//TBOPTV     DD     *
   VTSNAME=xxxx
   VTSCNTL=REFRESH
/*
```

  where 'xxxx' is the subsystem name of the server to be refreshed; the default value is DKL1.

The following JCL provides an example of a job that updates a tableBASE table, then ensures that the updated table is made available via VTS by refreshing the VTS Server. No other tables in that server's TSR are affected.

VTS Refresh - Sample JCL

```
//CHGDATA   EXEC PGM=TBEXEC
//STEPLIB   DD DSN=*your.prefix*.TBASE.LOAD,DISP=SHR
//TBMSG     DD  SYSOUT=*
//TBRPT     DD  SYSOUT=*
//BAS1LIB   DD  DSN=*your.prefix*.VTS.DKL1.BAS1LIB,DISP=SHR
//CNTLCARD  DD *
   CHANGE TBL=EXAMPLE LIB=BAS1LIB
                   ORG=D  MTHD=C;
//*
//LOADCMDS EXEC PGM=IEBGENER,COND=(0,LT)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT2   DD DSN=*your.prefix*.VTS.DKL1.VCMDS,DISP=SHR
//SYSUT1   DD *
*    Change ML list, refresh table EXAMPLE and any of its alternates
ML,BAS1LIB,ALT1LIB,MAINLIB
RF,EXAMPLE
GD,EXAM*
PR,EXAM*,0
//*
//REFRESH   EXEC PGM=TBPOSTV,COND=(0,LT)
//STEPLIB   DD DISP=SHR,DSN=your.authorized.VTS.loadlib
//SYSOUT    DD SYSOUT=*
//TBOPTV    DD *
VTSNAME=DKL1
VTSCNTL=REFRESH
//*
```

VTS Refresh JCL Notes

The CHGDATA step modifies the EXAMPLE table using tableBASE utility program TBEXEC. The CNTLCARD input changes the organization and search method of the table so that any subsequent item accesses will be in descending order.

The LOADCMDS step modifies the VTS Server input dataset to execute a 'Refresh' command (RF) against the EXAMPLE table.

The REFRESH step, executing program TBPOSTV, requests the VTS Server to process all commands found in the Server's CMD dataset (RF,EXAMPLE, etc.), thus refreshing the subsystem.

The Server will load this (potentially new) table EXAMPLE into the Virtual TSR, re-chain whatever alternate indexes are currently associated with the table, then reset the TSR directory for use by the VTS Clients. While the Refresh operation is in process, clients accessing this VTS Server will continue to be able to access all open tables. The original version of a table being refreshed will be available until the updated table is completely refreshed.

VTS Refresh Precautions

While the VTS server refreshes a table, the old version of the table is available to clients until the table is rebuilt, at which point clients are switched to the new version. This means a client accessing a table with GN, GP, or FC processing across the refresh could skip rows or retrieve duplicate rows if the refresh process added or deleted rows in the table. Key-based access can also be affected if keys are updated, added, or deleted.

In addition, once a new version of a table is built, the space occupied by the old version is released for reuse, possibly by the new version of a subsequently refreshed table. If a client were suspended by MVS after gaining access to a table but before transferring the row data from the virtual TSR to its own storage while a refresh of this table was occurring, it is extremely unlikely, but not impossible, for incorrect data to be returned to the client. This could occur if the MVS dispatching priority of the VTS server were much higher than that of the VTS client, resulting in the client not being redispatched until the server had refreshed the table being accessed and rebuilt another refreshed table in the original space. Although this is highly improbable (since the VTS server requires I/O to complete a refresh and there is no I/O during a VTS client tableBASE call), as a precaution, we recommend that transactions and programs accessing tables being refreshed be disabled until the refresh is complete.

VTS Server Shutdown

The VTS Server may be shutdown normally by one of the following methods:

- By issuing, from an MVS console, a stop command to the VTS Server job/STC. The form of the command is

  STOP/P job/STCname

- By issuing, from an MVS console, a modify command to the VTS Server job/STC. The form of the command is

  MODIFY/F job/STCname,function

- By running program TBPOSTV in batch. Sample JCL for this program to cause a server to shut down follows:

```
//SHUTDOWN   EXEC PGM=TBPOSTV
//STEPLIB    DD     DISP=SHR,DSN=your.authorized.VTS.loadlib
//SYSOUT     DD     SYSOUT=*
//SYSPRINT   DD     SYSOUT=*
//TBOPTV     DD     *
   VTSNAME=xxxx
   VTSCNTL=function
/*
```

  'xxxx' is the subsystem name of the server to be shut down (the default subsystem name is DKL1)

'function' in the MODIFY command and sample JCl above is one of the following:

SHUTDOWN instructs the server to process its CMD file prior to terminating

STOP terminates the server without processing its CMD file.

STOP may be faster than SHUTDOWN, depending on the contents of the CMD file at the time of termination.

Any Client attempting to access a terminating server will receive an error code of 1072.

Note: The MVS operator STOP/P command also terminates the server without processing its CMD file.

VTS Server Recovery

Should the VTS Server abend during execution for any reason, the VTS Server job/STC may be resubmitted to MVS. While the Server is re-initializing, client applications will not be able to access tables via the Server, and will receive an error code 1072 indicating that the Server is unavailable.

It should be noted that, when a VTS Server is restarted, it uses its CMD file for the restart as it appears at the time of the restart. This may be different than the CMD contents at the original startup, e.g., because of changes made to it for refresh operations, or new generations of tables to be loaded into the TSR may have been created. Thus, the contents of the TSR after a restart may be different than before the abend. So caution is advised when restarting a failed VTS Server. In particular, restart on one MVS image of a sysplex independently of the other MVS images could lead to TSR contents different from other images.

Parallel Sysplex Considerations

Since VTS supports read-only access to tables managed by it, parallel sysplex support for VTS does not require any considerations in coupling facility policies or structure definitions. The three areas which must be addressed are Startup (Initialization), Refresh Processing, and Shutdown.

Startup

If identical VTS table access is to be maintained across multiple MVS images, each MVS image must start the VTS server under the same subsystem name (VTSNAME parameter). Each VTS subsystem must load the same tables from the same library (or exact copies of the library) in the same order. We recommend using the same command file to ensure this. Access to a VTS subsystem on any image should not be allowed until initialization is complete on all images (after message "TBV0885I tableBASE/VTS (xxxx/########) Startup complete" is issued by each server).

One method for the installation to coordinate the initialization would be to issue the MVS ROUTE command with the *ALL operand to initiate the VTS started task:

```
RO *ALL,S  vtsjobstream
```

The "TBV0885I tableBASE/VTS (xxxx/########) Startup complete" message can then be used by a job control system to initiate jobstreams (such as online regions) that are dependent on the availability of VTS tables.

No transactions or jobs accessing the VTS subsystem should be released until VTS is initialized on all images, especially if the system allows batch jobs or transactions to be scheduled on any image. If transactions are released early, inconsistent results are possible, due to differing startup times on the various images.

Refresh

If a VTS table is to be refreshed, all programs and transactions on all MVS images that use the table should be disabled until the refresh is completed across all MVS images. To initiate the refresh across all images at the same time, you can issue the command:

```
RO *ALL,F job/STCname,REFRESH
```

Once again we recommend using the same VTS command file on all images. When the VTS message "TBV0887I tableBASE/VTS (xxxx/########) Refresh complete" has been issued on all MVS images, programs and transactions using the table can be re-enabled. If transactions and programs are not disabled until the completion of VTS refresh, inconsistent results are possible, as explained above under *VTS Server Refresh*.

Shutdown

Similarly, we recommend that VTS servers on all MVS images be shutdown at the same time. Disabling all programs and transactions on all MVS images before the shutdown will assure that clients do not receive inconsistent access results due to differing actual shutdown completions on the various images. To initiate shutdown of VTS on all images at the same time, you can issue the command:

```
RO *ALL,F job/STCname,STOP
```

VTS Operations JCL

This section describes how JCL may be composed to handle the starting, refreshing, and stopping of VTS servers. Three cases will be considered.

1. Simple Server

This is a very simple VTS application. A server loads some tables to a virtual TSR, keeping them available for client access until it is decided to stop the server. It may subsequently be started again, with the same or different tables by appropriate changes to the CMD file. The Server, once started, is never to be refreshed.

The simplified JCL to handle this case is as follows.

VTSSTART

```
//VTSDKL1   EXEC PGM=TBLOADV,TIME=1440
//STEPLIB   DD   DISP=SHR,DSN=your.authorized.VTS.loadlib  <=== Change STEPLIB
//SYSUDUMP  DD   SYSOUT=*
//TBOPT     DD   *
//TBOPTV    DD   *
  VTSNAME=DKL1
  VTSSIZE=2M
  VTSTEMP=YES
//CMD       DD   *
*  Sample CMD file to open all tables starting with TBL.  Change as appropriate.
ML,MAINLIB
OA,TBL*
//MAINLIB   DD  DISP=SHR,DSN=...   <=== Specify Server's MAINLIB
//OTHERLIB  DD  DISP=SHR,DSN=...   <=== DDs as needed for more libraries
```

VTSSTOP

```
//TBPOSTV   EXEC PGM=TBPOSTV
//STEPLIB   DD   DISP=SHR,DSN=your.authorized.VTS.loadlib   <=== Change STEPLIB
//SYSUDUMP  DD   SYSOUT=*
//TBOPTV    DD   *
  VTSNAME=DKL1
  VTSCNTL=STOP
/*
```

Procedures VTSSTART and VTSSTOP are provided as samples in TBASE.CNTL should you wish to use them.

2. General Server

In this example, the Server will be refreshed, possibly multiple times, possibly with different CMD input at each refresh, and with a CMD list at shutdown.

This is the fairly general situation, and we have provided a scheme for handling it. Each server that you will run requires a PDS and a sequential dataset. Suppose the server is called xxxx; the datasets are named

your.prefix.TBASE.VTS.xxxx.XCMDS and
your.prefix.TBASE.VTS.xxxx.VCMDS.

PDS xxxx.XCMDS contains members that are used to control the server, file xxxx.VCMDS is used as the server's input CMD file and is typically revised as required by copying members of xxxx.XCMDS to it.

The members of xxxx.XCMDS are:

STARTJCL

```
//*  JCL TO START VTS SERVER
//*
//SRVRSTRT PROC SERVER=
//*
//*  FIRST, COPY STARTUP COMMANDS TO SERVER'S CMD FILE
//*
//GENERCMD EXEC PGM=IEBGENER
//SYSPRINT   DD  SYSOUT=*
//SYSIN      DD  DUMMY
//SYSUT1     DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..XCMDS(STARTUP)  <===
//SYSUT2     DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..VCMDS           <===
//*
//*  THEN FIRE UP THE SERVER
//*
//VTSSRVER EXEC PGM=TBLOADV,TIME=1440,COND=(0,NE)
//STEPLIB   DD  DISP=SHR,DSN=YOUR.TBASE.VTS.LOAD                       <===
//SYSUDUMP  DD  SYSOUT=*
//SYSOUT    DD  SYSOUT=*
//TBOPT     DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..XCMDS(TBOPT)      <===
//TBOPTV    DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..XCMDS(SRVRNAME)   <===
//          DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..XCMDS(TBOPTV)     <===
//CMD       DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..VCMDS             <===
//*
//          PEND
//*
//          EXEC SRVRSTRT,SERVER=DKL1
//MAINLIB   DD  DISP=SHR,DSN=...  <====  Specify  Server's MAINLIB
//MORELIBS  DD  DISP=SHR,DSN=...  <==== DDs as needed for more libraries
```

STARTUP

```
*  THESE ARE THE COMMANDS THE SERVER WILL RUN AT  SERVER STARTUP TIME.
*  EACH INSTALLATION/SERVER WILL HAVE ITS OWN SET OF COMMANDS.
*  MODIFY THE COMMANDS BELOW AS REQUIRED FOR YOUR INSTALLATION.
ML,MAINLIB,MORELIBS
AR,TABL*
LT
```

TBOPT

```
DELIM=*;   EXAMPLE OF HOW WE MIGHT OVERRIDE SYSTEM PARAMETERS
```

SRVRNAME

```
VTSNAME=DKL1;    DEFAULT NAME
```

TBOPTV

```
VTSSIZE=2M;      DEFAULT IS 2M
VTSTEMP=YES;     DEFAULT IS YES
```

RFRSHJCL

```
//*  JCL TO REFRESH VTS SERVER
//*
//SRVRRFSH PROC SERVER=
//*
//*  FIRST, COPY REFRESH COMMANDS TO SERVER'S CMD FILE
//*
//GENERCMD EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DUMMY
//SYSUT1   DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..XCMDS(REFRESH)  <===
//SYSUT2   DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..VCMDS           <===
//*
//*  THEN GET TBPOSTV TO ACTIVATE SERVER
//*
//TBPOSTV   EXEC PGM=TBPOSTV,COND=(0,NE)
//STEPLIB   DD  DISP=SHR,DSN=YOUR.TBASE.VTS.LOAD                     <===
//SYSUDUMP  DD  SYSOUT=*
//TBOPTV    DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..XCMDS(SRVRNAME)  <===
//          DD  DDNAME=SYSIN
//*
//          PEND
//*
//          EXEC SRVRRFSH,SERVER=DKL1
//TBPOSTV.SYSIN DD *
  VTSCNTL=REFRESH;
/*
```

REFRESH

```
*  THESE ARE THE COMMANDS THE SERVER WILL RUN WHEN IT IS ASKED TO
*  REFRESH.  THESE MAY CHANGE FROM TIME TO TIME, BUT, IF A "STANDARD"
*  OPERATING CYCLE IS IN USE, THE COMMANDS MAY BE FAIRLY STATIC.
*  FOR DISTRIBUTION SAMPLE, WE WILL ASSUME THE SERVER SHOULD SIMPLY
*  REFRESH ALL TABLES IT CURRENTLY HAS OPEN.
*  MODIFY THESE COMMANDS AS REQUIRED FOR YOUR INSTALLATION.
RF,*
LT
```

SHUTJCL

```
/*  JCL TO SHUTDOWN VTS SERVER
//*
//SRVRSHDN PROC SERVER=
//*
//*  FIRST, COPY SHUTDOWN COMMANDS TO SERVER'S CMD FILE
//*
//GENERCMD EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DUMMY
//SYSUT1   DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..XCMDS(SHUTDOWN)  <===
//SYSUT2   DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..VCMDS            <===
//*
//*  THEN GET TBPOSTV TO ACTIVATE SERVER
//*
//TBPOSTV  EXEC PGM=TBPOSTV,COND=(0,NE)
//STEPLIB  DD  DISP=SHR,DSN=YOUR.TBASE.VTS.LOAD                      <===
//SYSUDUMP DD  SYSOUT=*
//TBOPTV   DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..XCMDS(SRVRNAME)   <===
//         DD  DDNAME=SYSIN
//*
//         PEND
//*
//         EXEC SRVRSHDN,SERVER=DKL1
//TBPOSTV.SYSIN DD *
  VTSCNTL=SHUTDOWN;
/*
```

SHUTDOWN

```
*  THESE COMMANDS WILL BE RUN BY THE SERVER IF SHUTDOWN IS REQUESTED.
*  LIKELY, WE DON'T WANT IT TO DO ANYTHING, BUT THERE MAY BE SPECIAL
*  OCCASIONS WHEN WE WANT SOME COMMANDS TO BE RUN.  MODIFY THESE
*  COMMANDS AS REQUIRED FOR YOUR INSTALLATION.  WE USE LT AS A SAMPLE.
LT
```

STOPJCL

```
//*  JCL TO STOP VTS SERVER
//*
//SRVRSTOP PROC SERVER=
//*
//*  GET TBPOSTV TO ACTIVATE SERVER
//*
//TBPOSTV  EXEC PGM=TBPOSTV
//STEPLIB  DD  DISP=SHR,DSN=YOUR.TBASE.VTS.LOAD                      <===
//SYSUDUMP DD  SYSOUT=*
//TBOPTV   DD  DISP=SHR,DSN=YOUR.TBASE.VTS.&SERVER..XCMDS(SRVRNAME)  <===
//         DD  DDNAME=SYSIN
//*
//         PEND
//*
//         EXEC SRVRSTOP,SERVER=DKL1
//TBPOSTV.SYSIN  DD  *
  VTSCNTL=STOP;
/*
```

If you find this scheme suitable for your VTS server operations, the job VTSCR8OP in TBASE.CNTL will allocate and set up the XCMDS and VCMDS datasets for any server xxxx. You will have to make simple changes to VTSCR8OP to allow for your dataset naming and job card conventions. The changes are described at the top of VTSCR8OP.

3. Roll Your Own

Having seen the simple and general cases, you may choose to define your own scheme for operating your VTS server(s). The essential things to be aware of are these:

- The server (TBLOADV) reads and processes its CMD file at startup, refresh, and shutdown (but not at a stop);
- the refresh, shutdown, or stop notification may be given to the server either by an operator Modify (or for stop, operator Stop) command or by a batch (TBPOSTV) job.

Chapter 3
Server Commands

The VTS Server provides a memory-resident repository for table data that can be accessed by VTS Clients using the Client API command set described in Chapter 3 of the ***tableBASE Programmer's Guide***.

For any Client access to a VTS Server table to succeed, the table must first be "opened" within the Server. The VTS Server command input file (DDNAME CMD) controls the actions that the Server performs at startup and, optionally, at termination and refresh. These actions should include opening all required tables.

VTS Server command input follows the same syntax rules as those for TBDRIVER (see the ***tableBASE Programmer's Guide***). Note that some commands provide a "wildcard" capability; i.e., the command parameters may refer to more than one table provided the tables have certain common characters in their names. For example:

OR,B*	Opens all tables beginning with the letter 'B'
OR,*T*	Opens all tables having the letter 'T' as the second character
OR,*C	Opens all tables where the table name is two characters in length with the letter 'C' as the second character

Common Commands for the VTS Server

The commands to make tables available in the Server are as follows:

Command	Function	Wildcard Available?
ML	Define the library list that the Server will use to process subsequent commands in the command input.	No
AL	Dynamically allocate a library	No
UL	Dynamically free (un-allocate) a library	No
DE	Disengage all table libraries	No
OR	Open table for Read	Yes
CN	Change name of a table in memory only	No
IA	Invoke alternate index	Yes
AR	Open base table and invoke all associated alternate indexes	Yes
OA	Open Any. If table is a base table, open table for read. If table is an alternate, open base table and invoke the alternate.	Yes
RF	Refresh currently open table and alternates, if any, from copy in library. Note that the table will always be refreshed even if it the currently open version is the latest version of the table.	Yes
CL	Close table	Yes

Note: With the exception of the CL (Close Table) and RF (Refresh) commands, the “wildcard” applies to tables in the current library list. Any “wildcard” characters in the CL and RF commands apply to tables in the Virtual TSR.

The results for each command processed are written to the Server's SYSOUT file.

Diagnostic Commands for the VTS Server

In addition to the above, the following commands are supported as input to the Server. They may be useful for diagnostic or audit purposes. Output from these commands will be written to the SYSOUT file of the VTS Server job.

Command	Function	Wildcard Available? *
SK	Search for (first) matching Key	Yes
FK	Fetch (first) matching Key	Yes
FC	Fetch matching Count	Yes
FG	Fetch Generic key	Yes
GF	Get First item in table	Yes
GL	Get Last item in table	Yes
GP	Get Previous item in table	No
GN	Get Next item in table	No
GD	Get table Definition	Yes
NX	Get NeXt table name	No
LL	List Library search order	No
CS	Change tableBASE (VTS Server) Status	No
LS	List tableBASE (VTS Server) Status	No
LT	List Tables currently open in the Virtual TSR	No
BN	Banner display (tableBASE vendor/client data)	No
PR	Print items in table	Yes
DR	Print directory	No
VX	Execution of a program in VTS (See note)	No

* The “wildcard” applies to tables in the Virtual TSR.

Note: The format of the VX command is

VX,pgmname,text

where: “pgmname” is the name of a user-written program

“text” is data to be passed to the program (a comma imbedded in the text will terminate scanning of the text)

On entry to the user-written program,

R0 contains the current “Error Level” status of the VTS server

R1 points to the first byte of “text”

The program must be in an authorized library in the VTS server STEPLIB list.

Sample programs TBVXUSRL and TBVXGATE are provided in TBASE.SRC.

Chapter 4
Considerations

Accessing the tableSPACE Region

The following comparison highlights the differences between the tableBASE interfaces that access a local TSR in the application region and the tableBASE VTS Interface that accesses a Virtual TSR.

Non-VTS Interface	VTS Interface
Tables stored in local storage	Tables stored in an MVS/ESA dataspace
Tables may be opened for read implicitly	Tables must be explicitly opened in the Server
Paged tables are supported.	Paged tables are not supported.
TableBASE table rollouts are supported.	tableBASE table rollouts are not supported.
A table can be updated.	Refresh process must be used to change a table.
All tables must be opened individually and all alternates must be invoked individually by name.	Command 'OA*' opens any base or alternate plus base tables. Command AR opens base tables and all associated alternate indexes.
A table cannot be refreshed. It may be closed and re-opened	Command 'RF,tablename' refreshes a table that is open in the TSR.

Implementing VTS without modifying user programs

By using the "VTSFIRST" or "VTSLAST" options in the TBOPT dataset, tables can be accessed on VTS servers instead of local libraries without any changes to existing programs. If a VTS server is implemented this way, be aware of the following:

- An ERROR CODE 1072 is returned if VTS is not available. This can occur if VTS is not up, not fully initialized or is terminating. (Under prior releases of VTS, error code 1072 was also returned during refresh processing.) The VTS server message "TBV0885I tableBASE/VTS (xxxx) Startup Complete" or the VTS VX command can be used to coordinate scheduling of dependent programs. (See the VX command description in Chapter 3, *Diagnostic Commands for the VTS Server*, and the examples TBVXUSRL and TBVXGATE in the TBASE.SRC dataset).
- If a VTS table must be refreshed, to ensure consistent access to the data, we recommend that all programs or transactions that access the table be

disabled until the refresh has completed. See Chapter 3, *VTS Server Refresh*, for details.

- If VTS is refreshed while programs or transactions are accessing it, internal processing may cause the programs or transactions to experience a delay (see note in Chapter 3, *VTS Server Refresh*, for details).

Capacity

The tableBASE VTS Interface loads tables into ESA Virtual Storage segments called dataspaces. Each dataspace may be defined up to a maximum of 2 gigabytes . For every 100 tables defined by parm VTSNMTAB in TBOPTV, approximately 16K of the dataspace is reserved for table directory information.

Only one dataspace is supported by each VTS subsystem. However, a VTS Client application may refer to any number of subsystems consecutively by specifying the subsystem name in the TB-SUBSYSTEM field of TB-PARM parameter of the TBLBASE interface or by using a suitable ML list with "VTS:" entries and/or using the VTSFIRST or VTSLAST TBOPT parameter.

Chapter 5
VTS Messages

In general, messages ending with an 'I' are informational, messages ending with an 'E' are user errors, and messages ending with an 'S' are system errors.

Serious Errors

The following message is always printed in the event of internal logic errors.:

```
SERIOUS TABLE BASE INTERNAL ERROR: CALL YOUR S.E.
TBL BASE ERROR ABEND XXXX, COMMAND=XX,XXXXXX
```

A VTS Client will not abend; however, the task will not be able to use the VTS system from that point on; clients will receive error code 1072.

VTS Client Messages

tableBASE/VTS Client error messages are the same as those issued by tableBASE when accessing a local TSR. Error codes are set in the tableBASE command area's ERROR-CODE field.

Level 1000 Errors

Some error messages have been upgraded (by adding '1000' to the error code) in order to reflect a specific VTS environmental problem. Code 1072 is the only one that will be presented to Clients.

1027	Storage Mode Code of 'P' is not supported (Paged tables are not supported by the VTS Server.)
1072	VTS Server is not available (is not running, has not finished initializing, is terminating, or is not compatible with the client)
1089	A request to create a dataspace was rejected by MVS

VTS Server Messages

Messages issued from the VTS Server vary from function to function. Any 'S' (System Errors) messages should be referred to your tableBASE support person(s). In messages below, "xxxx" represents the VTS subsystem name and "########" represents the MVS system name.

TBV0000	**I**	tableBASE/VTS (xxxx) Initializing
TBV0001	**I**	tableBASE Virtual Table Share active

TBV0002	**I**	tableBASE command processor available
TBV0010	**I**	tableBASE VTS (xxxx/########) Initializing
TBV0015	**S**	Option errors - tableBASE VTS terminating
TBV0016	**S**	A copy of subsystem xxxx is already running
TBV0017	**S**	Another subsystem named xxxx is already running
TBV0018	**S**	Logic error creating subsystem CVT
TBV0019	**S**	Logic error creating anchor block
TBV0020	**I**	Command is ********
TBV0021	**S**	Logic error deleting anchor block
TBV0022	**E**	Invalid command ******** ignored
TBV0023	**S**	Logic error deleting SSCT
TBV0025	**I**	Initializing dataspaces
TBV0033	**I**	******** ******** FAILED RC=**, REASON=********
TBV0040	**I**	Requested dataspace not found
TBV0041	**I**	SSCVT anchor table not found
TBV0042	**I**	Subsystem xxxx init incomplete, retry in 10 secs
TBV0043	**I**	Subsystem xxxx quiescing, cannot access anchor
TBV0044	**I**	Subsystem xxxx not found
TBV0055	**I**	SSCVT anchor table found
TBV0056	**I**	SSCVT anchor table already exists
TBV0066	**I**	Subsystem CVT entry does not exist
TBV0067	**I**	Subsystem anchor not found
TBV0070	**I**	Command post initialization in progress
TBV0071	**I**	Command = ********
TBV0083	**S**	ASCB not found
TBV0084	**S**	Transition to non-swappable failed
TBV0086	**I**	TBLOADV is now non-swappable
TBV0091	**S**	Anchor table not found
TBV0092	**E**	TBLOADV initialization not complete
TBV0093	**S**	Option errors - tableBASE post terminating
TBV0094	**E**	Post had already completed
TBV0095	**S**	Subsystem xxxx does not exist
TBV0096	**E**	Subsystem quiescing, command not posted
TBV0097	**E**	Subsystem not waiting for command
TBV0098	**I**	Subsystem busy, will retry command
TBV0099	**S**	Unable to locate ASCB for TBLOADV
TBV0101	**I**	STOP command received
TBV0123	**E**	Invalid command specified

TBV0567	**I**	Anchor has been released
TBV0568	**S**	Logic error deleting anchor
TBV0569	**S**	CVERASE logic error deleting SSCT
TBV0570	**I**	CVERASE SSCT deleted
TBV0701	**I**	Console interface is available
TBV0721	**I**	Console interface is not available
TBV0874	**I**	TBLOADV startup in progress
TBV0875	**I**	TBLOADV startup complete
TBV0876	**I**	TBLOADV refresh in progress
TBV0877	**I**	TBLOADV refresh complete
TBV0878	**I**	TBLOADV shutdown in progress
TBV0879	**I**	TBLOADV shutdown is complete
TBV0884	**I**	tableBASE/VTS (xxxx/########) Startup in progress
TBV0885	**I**	tableBASE/VTS (xxxx/########) Startup complete
TBV0886	**I**	tableBASE/VTS (xxxx/########) Refresh in progress
TBV0887	**I**	tableBASE/VTS (xxxx/########) Refresh complete
TBV0888	**I**	tableBASE/VTS (xxxx/########) Shutdown in progress
TBV0889	**I**	tableBASE/VTS (xxxx/########) Shutdown complete
TBV0901	**I**	Post error recovery entered
TBV0902	**I**	Entered with SDWA
TBV0903	**I**	Entered with no SDWA

VTS Server Diagnostic Messages

In the event of an abend, the VTS Server is required to restore the environment back to the way it was found. An MVS extended ESTAE is therefore given control. The following messages are issued from this procedure.

TBV1001	**S**	ESTAE cannot locate SSCT, no cleanup performed
TBV1002	**S**	ESTAE cannot find anchor, no cleanup performed
TBV1003	**I**	ESTAE entered with S****, U****
TBV1021	**S**	ESTAE logic error deleting anchor block
TBV1022	**S**	ESTAE logic error deleting SSCT
TBV1029	**S**	ESTAE entered without PARMLIST, cleanup impossible
TBV1300	**I**	ESTAE ALE deleted
TBV1301	**I**	ESTAE dataspace deleted
TBV1302	**I**	ESTAE No dataspaces to delete
TBV1303	**I**	ESTAE anchor deleted
TBV1305	**I**	ESTAE SSCT deleted

INDEX