TS DataServer™ for MS SQL Server
Reference Manual

Version 8.8.0

46 Vreeland Drive, Suite 1 • Skillman, NJ 08558-2638
Telephone: 732-560-1377 • Outside NJ 800-524-0430
Fax: 732-560-1594

Internet address: **http://www.tbred.com**

Published by:
Thoroughbred Software International, Inc.
46 Vreeland Drive, Suite 1
Skillman, New Jersey 08558-2638

Document Number: TQ8.8.0M102

INTRODUCTION

TS DataServer for MS SQL Server is an integral part of the Thoroughbred Software network solution. With TS Environment 8.7.1+, data from SQL Server tables can be made available across platforms for use in Dictionary-IV, Report-IV, Query-IV, Script-IV, and Thoroughbred Basic applications.

Client Operating System Support: UNIX, Linux, OpenVMS, and Windows.
Server Operating System Support: Windows Servers
For specific information, please contact your Thoroughbred Sales Representative.

This guide first describes the Client side of TS DataServer for MS SQL Server, which is provided with every TS Environment installation. It then provides specific information for accessing SQL servers.

Your applications may require special privileges to access the database. You should consider the following issues:

- Setting privileges to access the database.
- Designing or modifying site procedures.
- Setting procedures for server transaction processing.

Requirements

To run TS DataServer for MS SQL Server your system must meet the following requirements:

- Network communications implemented with TCP/IP protocol
- TS Environment 8.7.1 or above
- SQL Server 2000 (Version 8.0 or higher)

INSTALLATION

All installations and activations must be run with Administrator privileges. Once fully installed and activated the products can be run using standard user accounts.

1. Login using an **Administrator** account.

2. We recommend closing all unnecessary applications before running the installation.

3. Insert the TS DataServer for MS SQL Server V8.8.0 CD. Browse the CD and run setup.exe located in the TSMSSQL folder. There is only one setup.exe for all platforms.

4. The InstallShield Wizard will start the installation process. You can exit this installation at anytime by selecting the Cancel button.

User Information

Enter your Name, Company and serial number supplied on the distribution media.

Choose Destination Location

Type and enter or browse and enter the path to the location for the installation. The default destination folder is C:\Program Files\TSSQLSVR. **We strongly recommend installing into the default folder.**

NOTE: If you are installing on Windows Vista, Windows Server 2008, Windows 7, Windows 8 or Windows Server 2012 you must install into the Program Files folder.

Once the Wizard completes, the TS DataServer for MS SQL Server service will need to be started.

IMPORTANT: The product must be activated prior to starting the service.

ACTIVATION

From the Start menu, select the **Thoroughbred Software** program group. Select **TS DataServer for MS SQL Server Activation**. The following screen will display:

Before you contact Thoroughbred Software International make sure you have the following information handy.

- Your Dealer Code.
- The software serial number, which is found on the label on the distribution media or on the displayed Activation screen.
- The Installation Code, which is also displayed on the Activation screen.

To get an Activation Key you can:

- Call Thoroughbred Software International at (800) 524-0430 or (732) 560-1377 and follow the prompts to obtain an Activation Key.
- Send a fax to Thoroughbred Software International at (732) 560-1594.

  Note: Please make sure that you include your fax number in the message so that Thoroughbred Software International can send the Activation Key to you.
- Visit our web page, www.tbred.com, and select the Activations option.

After Thoroughbred Software International provides the Activation Key, you must enter the value using the following procedure.

Enter the Key in the **Enter Activation Key** field and click the **Authorize** button.

Following a successful activation, the system responds with this screen:

Click **OK**. The server is now activated.

If an invalid Activation key is entered, the following screen will display:

Click **OK.** The system will return to the main Activation window.

To verify that the server is activated, execute the Server Activation program. The following screen will display:

Click **OK** to close the window.

Now the service can be started. From the **Start Menu**, select **Control Panel → Administrative Tools → Services**. Locate the TS DataServer for MS SQL Server Manager service. Click the **Start** button. A Service Control dialog will display while the system starts the service. Once started, the status field will be changed to "Started".

Server access is enabled by the addition of a number of configuration items. For more information see SERVER.MAP and IPLINPUT.

PRODUCT DESCRIPTION

TS DataServer for MS SQL Server Components

The TS DataServer for MS SQL Server consists of three components:

- A Client enabled TS Environment,
- A Manager Process that runs as a Windows Service (tsmssqlmgr), and
- A Server Process (tsmssql) for each connection.

Client Enabled TS Environment

The client side is enabled in the standard TS Environment. When configured for data server access, it detects requests for data that reside on the server. When this event occurs, it sends the I/O request to the server for processing. The server responds with the result set and processing continues.

Manager Process

The Manager program runs as a Windows Service. The Manager Program (tsmssqlmgr) creates Server Processes for clients as needed. When the Service detects a client attempting to connect, it tries to create a Server Process for the client. If a Server Process is created successfully, the client and server can perform client/server communications.

NOTE: The Service must be started before clients attempt to connect to it.

Server Process

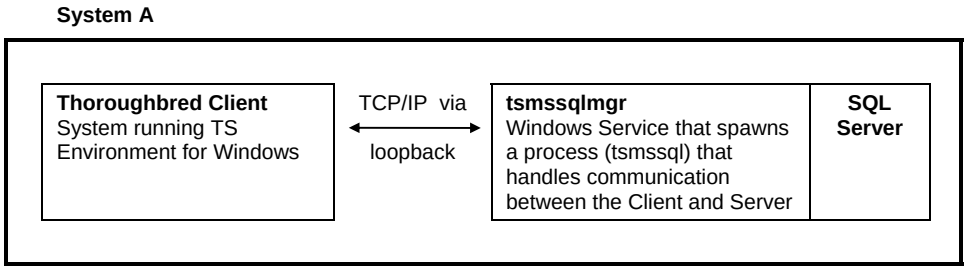
The Server Process (tsmssql) is triggered by requests from the client. Once a request is received, it is processed and the results returned to the client.

System Architecture

The following diagrams outline the core options for configuring the components of the Thoroughbred client.

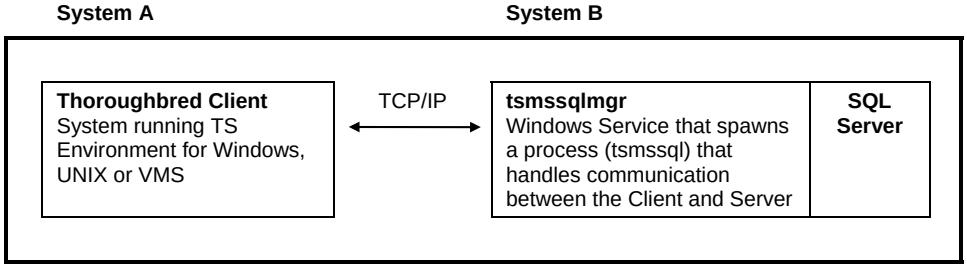
- Client and server on one system
- Client on one system - server on another

Client and Server Co-exist on One System



In the layout described by the above figure, the information is sent to the Server Process through the loopback TCP/IP address.

Client on System A and Server on System B



In the above figure the information is sent over the network to the server system. Once the requests reach the Server Process it creates SQL statements that are interpreted by the server system. The results are then passed back to the client through the Server Process.

CLIENT AND SERVER ENVIRONMENT

To set up TS DataServer for MS SQL Server, the following files must be created or modified:

- SERVER.MAP
- IPLINPUT
- Registry Settings (optional)

SERVER.MAP

The SERVER.MAP text file establishes a relationship between a 2-character server ID used by Thoroughbred and the server system. This file is a simple text file.

NOTE: For UNIX Clients, this file should be placed in the **/usr/lib/basic** directory.

Thoroughbred uses the Server ID to reference the server system. Each line in the file represents an entry for one Server. The syntax for an entry is:

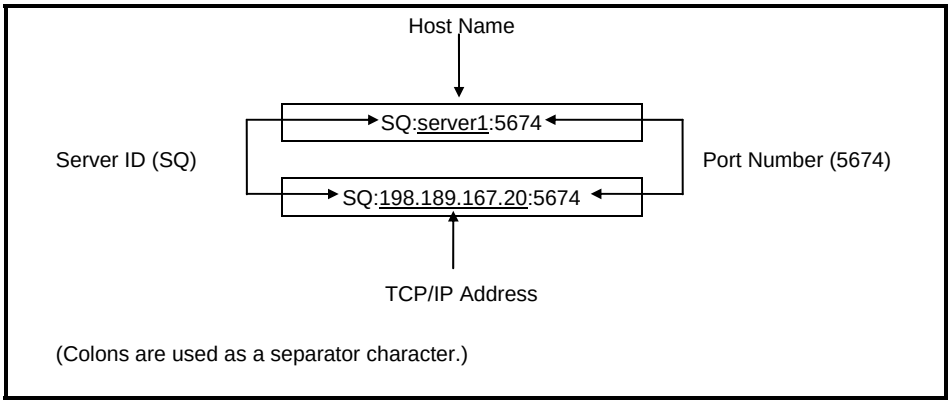
server-id*:[*host-name or TCP/IP-address*]:*TCP/IP-port* : *DataSource

server-id	is a 2-character alphanumeric string used within Basic to reference a particular server system.
host-name	is the host name for the server system. For installations where the Thoroughbred client and database server environments reside on the same physical system (Windows Network Basic) this field can contain the loop back address or host name.
TCP/IP-address	is the TCP/IP address for the server system. For installations where the Thoroughbred client and database server environments reside on the same physical system (Windows Network Basic) this field can contain the loop back address or host name.
TCP/IP-port	port number is an optional parameter. It is used to override the default port number (5674), in the event of a conflict with another process using TCP/IP. The transport protocol layer of the TCP/IP process uses the port number to deliver the packet data to the appropriate application. In the event of a conflict on a particular system, add the port number to the SERVER.MAP file and execute the server process using the port number as an argument. For more information see Server Process in the following section.
DataSource	is an optional setting. Data Sources are defined using the Microsoft Administration Tools. Data Sources allow multiple users to share the same settings (i.e. default SQL Server database). By default, the LocalServer Data Source is used. See Setting Up a Data Source section later in this document. DataSource names in the SERVER.MAP file may contain up to 32 alphanumeric characters (no dashes or underscores).

Example

The fields in a SERVER.MAP entry are separated by a colon (:). The following are examples of valid SERVER.MAP entries:

Server ID or TCP/IP Address



IPLINPUT

Once you establish the server ID you can make the required entries for the IPLINPUT file.

DEV

Each DEV line corresponds to a directory on the Server. The syntax for a DEV line is:

```
DEV device-name,1,,server-flag,authentication-type,logon-cache,case-
sensitivity,server-id:log/pswd
```

device-name	is device ID used to reference logical disk directories. Valid values range from D0 through D9, DA through DZ, and Da through Dz.
server-flag	indicates that this will be a client to the configured server system. The value for the SQL server is 4.
authentication-type	indicates whether logins to the SQL Server database are done with Windows authentication or SQL Server authentication. The default is Windows (blank or 0). To use SQL Server authentication, set this flag to 1.
logon-cache	configures a value for the maximum number of active logins to the SQL Server database. For more information see Logon Cache in the Performance Tuning section.
case-sensitivity	indicates whether or not to perform a case sensitive comparison of the key value returned from the SQL data server during a READ, KEY= operation. The default is to perform a case sensitive comparison (blank or 0). To have Basic perform a case insensitive comparison, set this flag to 1.
server-id	is the two-character alphanumeric code assigned to this server. The server ID on the DEV line must match an entry in the SERVER.MAP file.
log/pswd	when using SQL Server authentication, this is the SQL Server login and password used to access the database. The id and password are optional. If omitted, the system prompts for the information when the client is executed. You cannot mix authentication modes. If the authentication-type is 0, this field must be blank.

PRM

If multiple MS SQL Servers are configured for SQL Server authentication and the login/password are the same for each, the PRM MSSQL-LOGIN may be used as a shortcut to defining the login/password on each DEV line.

```
PRM MSSQL-LOGIN=login/password
```

login/password is the MS SQL Server login and password used to connect to the database. Only the login and slash (/) character are required. A prompt for the password will appear when the client connects.

PRM CREATWDBATTR

This parameter will create new tables using the attributes from the system dictionary. Mandatory fields (Entry Type 2 and 3 and the first key field) will be created with a NOT NULL constraint. In addition any fixed length fields (Entry Type 1 or 3) will be created with the CHAR data type so the field always contains data.

PRM ORA_NVLNULLS
PRM ORA_DONTWRITENULLS

These parameters control how the system orders records when the key field contains nulls. For more information see Null Processing.

PRM SQL_NUMERIC_NULLS

This parameter controls how numeric null values are stored in the database. When the data for a numeric field is sent over from a Thoroughbred Basic application as all spaces, if enabled, then this data will be written to the database as a null value, otherwise, it will be written to the database as a 0 value.

PRM UNIQUE-KEYS

This parameter will force all table creates to have unique secondary keys. This is accomplished by adding the Primary key to each sort definition. Some accommodations are made when the same data element name is used more than once in a sort definition. This is not supported by SQL Server.

Registry Settings

The following control parameters for the tsmssql server process are stored in the Windows registry:

HKEY_LOCAL_MACHINE\SOFTWARE\Thoroughbred Software International, Inc.\TS DataServer for MS SQL Server

The first time a tsmssql process is executed, the default values for these control parameters are created in the Windows registry.

Please do not edit the Windows Registry unless you are confident about doing so.

CLIENT-SYSTEM-INFO

This flag will enable process and host information to be logged. For more information see Using the Debugging Facility.

COMMIT-COUNT

This entry is used to specify the commit count. The maximum commit count is 65535. If the specified commit count is greater than the maximum, the maximum value will be used. The default commit count is 1.

The commit count can be overridden by SET CTC in Basic.

DEBUG-LEVEL

This entry is used to specify the debugging level. For more information see Using the Debugging Facility.

DEBUG-FLAGS

This entry is used to enable or disable additional information that can be written into the debug log file. The Flag values are:

Flag Number	Description
1	Disable logging of the process ID number
2	Enable logging of a time stamp
4	Enable logging of parameter values.
8	Enable logging of fetched values

LDA-CACHE

The entry is used to specify the logon cache count. For more information see Logon Cache.

NO-OPENLOCK

The flag will disable the use of the TS_TABLE_LOCK table.

READUNCOMMITTED

The flag is used to add 'WITH (NOLOCK)' to SQL statements when READing. This flag is enabled by default.

UNIQUE-SORTS

This flag will enable tables to be created with unique secondary keys.

Setup Example

This environment consists of two SQL Servers on two different host systems.

Existing server-name TCP/IP address information:

```
acct:125.32.1.1
dev:125.32.1.10
```

The SERVER.MAP file may be set up in one of the following ways:

- Using host names
  S1:acct
  S2:dev

- Using TCP/IP addresses
  S1:125.32.1.1
  S2:125.32.1.10

- Using a combination of host names and TCP/IP addresses
  S1:125.32.1.1
  S2:dev

- Specifying a DataSource name other than the default LocalServer
  S1:125.32.1.1::Production

In this example the login information to the SQL Server database (using SQL Server authentication) is as follows:

acct: login id = mary
password = mjs

dev: login id = kurt
password = ktc

The IPLINPUT should reflect the following:

```
DEV D8,1,,4,1,,,S1:mary/mjs

DEV D9,1,,4,1,,,S2:kurt/ktc
```

After Basic is executed, there will now be a tsmssql process on each of the server systems (acct and dev). If the Authentication Flag is not 1, Basic will display an error message.

Start the Windows Service (tsmssqlmgr)

The tsmssqlmgr manager is a system service that is installed on the server system. Each time a Basic client executes, a tsmssql process is started. The manager process is installed by default with a startup type of automatic. Each time the server restarts, this service will be automatically started. The Log On value for this service should be a valid Windows user (not the Local System account).

The service can be started with the following Start parameters:

-d Debug Logging

-p Override default port

The –d option is followed by a non-zero value indicating the debug level. The –p option is followed by a TCP/IP port number. By default, the manager will listen on port 5674. If this causes a conflict, use the –p option to override the default port. For more information using the –d option, see Using the Debugging Facility.

Setting Up a Data Source

The tsmssql manager requires a System data source. The System data source stores information about how to connect to the SQL Server database. The tsmssql manager will by default use a System data source called LocalServer. If the System data source is not named LocalServer, then it must be specified in either Registry or the SERVER.MAP file (for more information see the SERVER.MAP section earlier in this manual). System data sources can be added using the ODBC Data Source Administrator accessible via the Control Panel.

DICTIONARY-IV FORMATS AND LINKS

You must have a Format and Link defined to access an SQL Server table.

Format Definition

Although no changes are required to existing formats in order to use the client capabilities of the environment, you should be aware of the following information.

The data element names must not be SQL Server reserved words. For more information see the appropriate SQL Server documentation.

File Type

When Thoroughbred creates an SQL Server table, data types are translated as follows:

```
Alphanumeric:  VARCHAR or CHAR
Text field:    TEXT
Numeric:       DECIMAL
Date:          DATETIME
```

The VARCHAR and CHAR data types only support characters above hex 20 to be included in the data. Any character below hex 20 is converted to a space.

The CHAR type will be used when the Basic PRM CREATWDBATTR is configured and the data element is set as a fixed length field either mandatory or optional (Entry Type 1 or 3).

When records are written to this table, any field of the CHAR type will be assigned a value of at least one space. This will insure that the column is not null. For more information see PRM CREATWDBATTR in the IPLINPUT topic.

Entry Type

All mandatory fields (Entry Type 2 and 3) will be set as NOT NULL fields in the SQL Server tables when the Basic PRM CREATWDBATTR is set.

Data Elements – Naming

SQL Server only allows alphanumeric characters, the currency symbol ($), an underscore (_), and a pound sign (#) in column names. The first character of the column must start with a letter. Any deviation from this convention results in table creation failure.

The tsmssql program translates the hyphen (-) that is acceptable in Thoroughbred Dictionary-IV Formats to an underscore (_).

The use of SQL Server reserved words also results in table creation failure. For more information see your appropriate SQL Server reference material.

Data Elements – Multiple Occurrences

SQL Server does not support multiple occurrences for a single data element. For more information, see Multiple Occurrences in the Dictionary-IV Developer Guide.

The system creates separate columns in the SQL Server table from a multiple occurrence field. The following is the naming convention for multiple occurrences.

`data-element-name_seq#`

seq# is the range from one to the number of defined occurrences.

Binary Numeric Fields

Fields defined as binary in the Format are converted to decimal values and stored in SQL Server DECIMAL fields. The type of binary field determines the length of the field.

The following table displays the formulas for calculating the field width in the SQL Server table.

Formulas for Calculating Field Widths

Numeric Type	Field Width	Formula for SQL Server column Width L = Field Width
3,5	1 - 3	(L*2) + 1 + 1
	4 - 5	(L + 1)*2 = 1
	6 - 8	(L + 1)*2 + 1 + 1
4	1 - 2	(L*2) + 1
	3 - 4	(L + 1)*2
	5 - 7	(L + 1)*2 + 1
	8	(L + 2)*2
6	All	(L*2) + 1
7	10	16
	Odd Prec.	(((L*10) - 15)*2)/10
	Even Prec>	(((L*10) - 5)*2)/10 – 1
8	All	8
9	All	14
A	All	(L*2) - 1 + 1
B	All	(L*2) – 1
C	All	(L*2)
D	All	L + 1
E,F	All	L

In all cases: if the value contains a decimal, add one to the field width.

SQL Dates

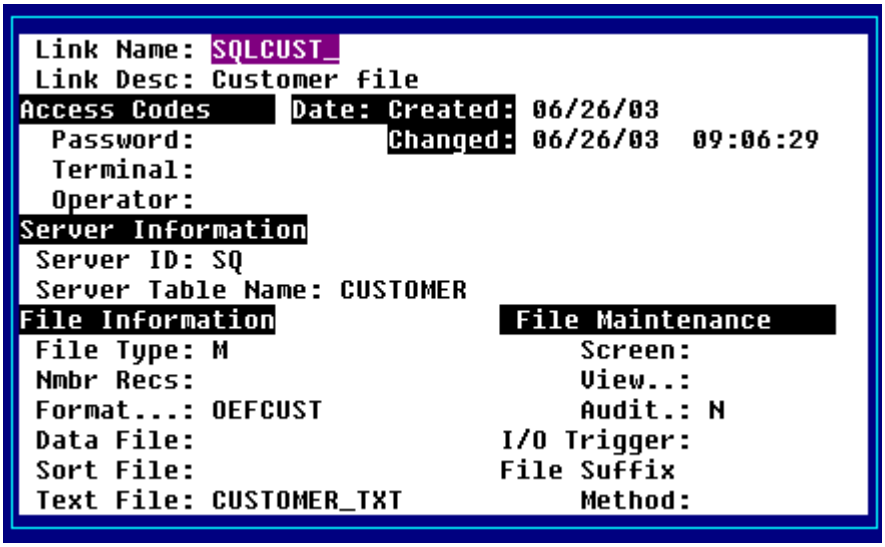
The system automatically translates dates stored in either of the SQL formats into the standard SQL Server DATETIME storage format as you update records.

Link Definition

The Link definition contains the configuration parameters necessary to describe a server table to Dictionary-IV. To access the SQL Server tables, you must enter the following information in the Link definition:

- File Type: M
- Server ID
- Table Name

Below is a sample Link definition screen:



Server ID
The server ID is a unique identifier that references the host via the definition in the SERVER.MAP file. It will be validated against the list of server devices configured in the IPLINPUT file. Although no ID will be disallowed, a warning will be displayed.

Server Table Name
The Table Name is the name used to create a table on the server.

Text File
A Text File value is required if the Format contains Text fields. In the example above, the Text Field data will be maintained in an SQL Server table named CUSTOMER_TXT. If the Text File name had a .TXT extension, the SQL Server table name would be converted to use _TXT.

Both the table and text file names must conform to SQL Server table naming conventions. For more information see the appropriate SQL Server document to review the restrictions.

Links must be defined as a file type M (MSORT). While the physical file is not stored as an MSORT file, it will be processed as one in Dictionary-IV for sorts and text field storage. For more information see Using Sorts and Text Fields for SQL Server Data.

For Dictionary-IV to detect that a Link references a server-based table, File Type, Server ID, and Table Name must all contain valid information. If they do not a local file will be used, referenced by the name entered in the Data File field.

The system supports file suffixes. The file suffix character (@) must be placed in the Table Name field for server files, rather than the Data File field. Where text fields are configured and a file suffix is used, the text file name must include the suffix character. For more information refer to the Dictionary-IV User Guide.

Sort definition occurs in the same way for server Links as for local Links and should be saved and updated with **F8** processing. Even though no physical rebuild takes place, this will properly update the Link header. For more information see the Dictionary-IV Developer Guide.

If sort definitions exist in the Link and the table is recreated, the indexes will be defined automatically on the server as part of the create process.

Under the current Dictionary-IV methodology, first build a Link header and create the table. Modify the Link header to define sorts. The next time the table is created, the indexes for all sorts will be defined on the server.

Once the Link header is configured, multiple and single record maintenance, CONNECTs, reports, queries, and recompiled scripts can access server tables. For more information about modifying Basic programs to support access to server tables see Using Basic.

While the data and sort file names may be populated during the Link header creation, neither is referenced during server access. However, the data file name is used to generate the text file name for text field storage. Therefore be sure to either enter the data file name for the automatic generation of a text file name or manually enter the field.

SQL SERVER TABLES

The following describes how to process SQL Server tables from the TS Environment.

Accessing SQL Server Tables

The following describes how to create SQL Server tables and then access these tables from the TS Environment. For more information specific to Script-IV and Dictionary-IV see Script-IV/Dictionary-IV Tables.

Using Basic

Existing server files are accessed using the following syntax:

```
OPEN(CH,OPT="LINK")"link-name"
```

"LINK" specifies that the file to be opened on this channel is found in the system dictionary in the link-name record.

"link-name" specifies a valid link-name containing the necessary server references.

New server files may be created from the Basic environment as follows:

```
OPEN(CH,OPT="LINK|CREATE")"link-name"
```

"CREATE" is a parameter that causes Basic to generate the server commands necessary to build the file. The SQL Server table definition is based on the format listed in the Link Definition. If the table already exists on the SQL Server, the CREATE is ignored and the table opened.

A special OPEN syntax is used for access to text field data. This syntax is used internally by the Dictionary-IV system for accessing text field data. For more information see the Text Fields for SQL Server section later in this manual.

```
OPEN(CH,OPT="LINK|TEXT")"link-name"
```

"TEXT" specifies that the text file from the link is to be opened.

If multiple SQL servers are configured, existing server files on different servers can be accessed using the following syntax:

```
OPEN(CH,OPT="LINK|SID='server-id'")"link-name"
```

"SID=" is a parameter that causes Basic to access the specified server instead of the server ID defined in the link definition.

"server-id" specifies the id of one of the configured SQL servers.

Server files with names different from the names defined in a link may be accessed using the following syntax:

```
OPEN(CH,OPT="LINK|TABLE-NAME='table-name'|TEXT-FILE='text-table-name'")"link-name"
```

"TABLE-NAME="	is a parameter that causes Basic to use the specified table name instead of the one defined in the link definition.
"table-name"	specifies the table name.
"TEXT-FILE="	is a parameter that causes Basic to use the specified text field table name instead of the one defined in the link definition.
"text-table-name"	specifies the text field table name.

The following code fragments show multiple ways of using the new OPEN syntax to access files. The examples assume a Link definition with information in the Server ID and Table Name fields and a file type of M.

Example 1

```
OPEN/READ server file into IOLIST (variables)

10       IOLIST CUST_CODE$,CUST_NAME$,ADDRESS$

100      CH=UNT;
         OPEN(CH,OPT="LINK") "UTCUST"
         READ(CH,KEY="0001")IOL=10
```

This example assumes a Format containing field separators.

Example 2

```
OPEN/READ server file into FORMAT

10       CH=UNT;
         OPEN(CH,OPT="LINK") "UTCUST";
         FORMAT INCLUDE #UTCUST;
         READ(CH,KEY="0001")#UTCUST
```

This example can be used on any format with or without field separators.

Example 3

```
OPEN/READ server file into record variable

10       CH=UNT;
         OPEN(CH,OPT="LINK") "UTCUST";
         READRECORD(CH,KEY="CUST0001)REC$
```

This example shows access using a Format without field separators. The data in REC$ could be used to populate the format data area.

Under certain circumstances an error 167 may be returned when writing to a Link opened with OPT="LINK". When employing a Format directly or a Format that contains field separators, the system assigns that data to the data element name.

If this data is invalid for the particular data element name (invalid value for Yes/No fields, blank mandatory fields, or invalid data in a date field) an error 167 occurs. Printing the DNE system variable at the instance of the error will display the data element name for which the assignment failed.

Example 4

```
OPEN a link's table on a different server

10       CH=UNT;
         OPEN(CH,OPT="LINK|SID=Q2") "UTCUST"
```

Example 5

```
CREATE and OPEN a link's tables using different table names

10       OV$="|TABLE-NAME=BUCUST|TEXT-FILE=BUCUST_TXT";
         DCHL=UNT;
         OPEN(DCHL,OPT="LINK|CREATE"+OV$) "UTCUST";
         XCHL=UNT;
         OPEN(DCHL,OPT="LINK|TEXT"+OV$) "UTCUST
```

ERASE

Server tables are deleted from within the Basic environment by specifying the Link name and OPT="LINK".

```
ERASE"UTCUST",OPT="LINK"
```

This deletes the table from the server referenced in the Link UTCUST. It also deletes the text field table if any text fields were defined.

INITFILE

Server tables may be cleared of all data by using the INITFILE directive. This directive is useful for clearing a table without the extra step of performing an erase.

```
INITFILE "UTCUST",OPT="LINK"
```

This clears the table on the server referenced in the Link UTCUST by doing a DROP followed by an ADD. It also clears the text field table if any fields are defined.

RENAME
The rename function allows a table name to be changed on the server from the client process.

```
RENAME "UTCUST","UTTEMP",OPT="LINK"
```

This renames the table referenced in Link UTCUST to the table UTTEMP. No changes are made to the original Link.

FID Function
When the FID is executed on a channel where a Link referencing a server table is opened, the resulting string contains (in addition to the standard FID information) the following:

Byte 1 Value is "S". Indicates that this is a channel opened to a link referencing a server table.

Byte 2 - 3 The two character Server ID from the Link.

The bytes that normally contain the file name now contain the Link name.

TCB Function
Error 150 has been added to the Basic error list to reflect errors resulting from server operations. To return the specific server error, an argument has been added to the TCB( ) function. The TCB(20) now returns the server error in the same way that the TCB(3) returns the operating system error. The value returned for the TCB(20) can be interpreted by referencing the Microsoft knowledge-base. Often Basic directives will generate multiple SQL commands. Therefore it is possible for multiple SQL errors to occur. See the Virtual Error Table in the Error Message section of this manual.

DUMP Directive
Several of the DUMP options have been changed to be aware of server access.

DUMP CHANNELS now display the link name when a link has been OPENed with OPT="LINK".

DUMP IPLDEVS will specify that devices configured as a server will return "Access using server xx."

XFD Function

The following describes the current values returned for server tables:

Option = 0

Bytes	*Description*
1-2	Unused
3-10	N/A
11-35	Unused
36-38	N/A
38-41	N/A
42-47	Unused
48	"N"
49-53	Unused
54-59	N/A
60-65	N/A
66-71	N/A
72-85	Unused
86+	N/A

Option = 1

As documented in the Thoroughbred Basic Reference Manual.

Option = 3

Bytes	*Description*
1	05
2	83
3	Number of sorts (binary)
4	Current sort number (binary)
5-12	"NNY" (bytes 8-12 Unused)
13-14	N/A
15-16	COBOL only
17-42	Unused
45+	Sort definition parameters (For more information see the Thoroughbred Basic Reference Manual.)

Option = 4

As documented in the Thoroughbred Basic Reference Manual.

Option = 6

Bytes	*Description*
1	06
2	00
3-6	00000000
7-8	Bytes per record (binary)
9+	Link name

Option = 10

Returns -1

FST Function
This function is not supported for server tables.

IND Function
SQL Server does not support an IND function. IND for a channel opened to an SQL Server table will always return 1. Errors will be reported when trying to access SQL Server tables using IND.

NOTE: The string following OPT= can be a variable in any of the commands where it is specified. The value does not need to be hard-coded into the program.

Using Sorts

Sorts, also known as indexes, are handled in substantially the same manner for server tables as they are for local files. Created in the Link definition, they can be used for access in any operation that currently supports sorts. As with MSORT, if the Link references an existing file, the link definition can be updated to include sorts defined in the file without rebuilding the file. The differences occur in the way that sorting is applied to the server table.

Indexes do not need to physically exist in the server environment to use Link sorts anywhere in Dictionary-IV. As long as the sort definition is established in the Link definition, Dictionary-IV can use it to access the SQL tables. Basic will expand the sort statement defined in the Link into the appropriate field names, rather than passing only the sort number.

SQL sort statements will refer to these field (column) names, not index names, when defining an ORDER BY statement. So if the index does not exist, SQL performs the sorting operation in-line. However, if an SQL index exists for the columns in the ORDER BY statement, the index is used to access the data file. However, the Indexes will be used only if one of the columns has been defined with a NOT NULL constraint. For more information see the Performance Tuning section later in this manual.

If sort definitions exist in the Link and the table is recreated, the indexes will be defined automatically on the server as part of the create process.

Under the current Dictionary-IV methodology first build a Link header and create the table. Modify the Link header to define the sorts. Erase and recreate the table. The indexes for all sorts are now defined on the server.

SQL Server does not support sorts on substrings of fields.

The sorts (indexes) are created using the following naming convention:

"I_" + *table-name* + " _ " + *Dictionary-IV sort number*

Example:
For sort number 2 on a Link with a table-name defined as TSISERVER, the index would be created as "I_TSISERVER_2".

Adding and deleting sorts is not supported from within the Basic client (ADDSORT/REMSORT). These functions are reserved for the database administrator.

Locking Records

The locking facilities available with SQL Server provide the developer with resources to insure data consistency. Many of the features designed for these goals occur automatically. For more information refer to the appropriate Microsoft SQL Server documentation.

This section explains how these locking mechanisms correlate to Thoroughbred directives and how to relate locking that occurs in the Thoroughbred application to that, which occurs in an SQL application.

The following table summarizes the relationship of Thoroughbred directives to SQL statements and the level of locking associated with each.

Thoroughbred Directives/SQL Statements

Thoroughbred Directive	**SQL Statement**	**Type of Lock**	**Lock Modes Permitted (see legend below chart for description)**				
			RS	**RX**	**S**	**SRX**	**X**
READ, FIND, PREAD, and all key functions	SELECT... FROM table...	none	Y	Y	Y	Y	Y
WRITE, if new	INSERT INTO table...	RX	Y	Y	N	N	N
WRITE, if exist	UPDATE table...	RX	Y*	Y*	N	N	N
ERASE, INITFILE	DELETE FROM table...	RX	Y*	Y*	N	N	N
EXTRACT, WRITE	SELECT... FROM table... FOR UPDATE OF...	RS	Y*	Y*	Y*	Y*	N
OPEN, UNLOCK	SELECT … FROM table … WITH (HOLDLOCK) WHERE 1=2	RS	Y	Y	Y	Y	N
LOCK	SELECT … FROM table … WITH (HOLDLOCK,TABLOCKX) WHERE 1=2	X	N	N	N	N	N

RS	Row Share	**RX**	Row Exclusive	**S**	Share	**SRX**	Share Row Exclusive	**X**	Exclusive

* Valid as long as no conflicting records are involved.

For more information about the WRITE statement see the Generated SQL Code section later in this manual.

The Lock Modes Permitted column indicates what type of lock may be placed on the file (table) or record (row) while the particular directive is in effect. For example, if a record is EXTRACTed in Thoroughbred, exclusive (X) locks by another client or an SQL task are not allowed and produce an error. All other lock types are valid as long as a different row is locked.

Since the query-type functions (READ, FIND, etc.) do not generate a lock, it will not fail under any locking mode the table or record may currently have. This means that READing locked records will not require PRM READONLY to access tables or rows locked by either an SQL task or another Thoroughbred client process.

Null Processing

By default, null key values will be returned to the Thoroughbred Basic application following all other records for ascending sorts and before all records for descending sorts. This is an MS SQL Server convention and cannot be changed. However, two options exist to alter this behavior. Both methods involve the modification of the null data so that the sorting mechanism treats the null key value as non-null data. This is more consistent with a Thoroughbred Basic application where null keys appear at the top of an ascending sort and at the bottom of a descending sort.

In the first method, the null data is logically changed. This is accomplished by using the MS SQL Server ISNULL function in the ORDER BY clause of the SELECT statement. Each column listed in the ORDER BY clause is passed through this function which will map null values to the value supplied to the function. In this case, the value is a single space. This value will then be passed to the sorting process so that these records will sort at the appropriate end of the result set. While this method makes no modification to the physical data, it may cause sorts to perform poorly for composite indexes. To activate this method, specify the PRM ORA_NVLNULLS in the IPLINPUT file.

For single column indexes that allow null values, performance may not be an issue since this index would not be utilized.

In the second method, the data is changed physically. Whenever a null character field is written or updated to an MS SQL Server table, it will be changed to a single blank. All subsequent accesses with indexes will return the data with the blank fields sorted appropriately. This method will allow the index to be used for queries so that the result set is returned efficiently, but the data is modified. It will be important for other applications to maintain the space in the data. To activate this method, specify the PRM ORA_DONTWRITENULLS in the IPLINPUT file.

As an alternative to the above methods, the table could be created so that columns used as indexes can never contain null values. This will happen if the field is defined in Dictionary-IV as a fixed length type, either optional or mandatory (Entry Type 1 or 3). This attribute will insure that the data for that column will not be null.

For access to existing tables, all of the options are available, however, care must be taken if new data is written to a table when the ORA_DONTWRITENULLS parameter is specified. This will create a situation where only the new records with null keys appear in the proper sorted order.

Processing existing SQL Server Tables

To access existing SQL Server tables you must build a Format and Link with the appropriate table information. Be sure to use the exact spelling for data element names and the table name.

SQL Server supports many data types. Not all of the SQL data types have a one to one Thoroughbred equivalent. Many of the data types can be derived using Basic code. For example SQL Server supports 4 different integers (bigint, int, smallint, and tinyint). Each of these data types are binary byte swapped. Using DEC and SWP you should be able to determine their values. Refer to your Microsoft documentation for specific information on the SQL Server supported data types.

Be aware of the following SQL Server constructs:

- Variable length fields where data greater than 255 characters.
- Numeric fields exceeding Thoroughbred's maximum of 14 significant digits.
- Column names exceeding 20 characters in length.

You may be able to access a table that contains the above characteristics for reading and updating. By defining a format that contains a partial list of the fields defined in the table, data can be read and updated without affecting the fields not included in the format.

Text Fields for SQL Server Data

Dictionary-IV treats server tables internally as MSORT for I/O processing. Therefore text field data associated with servers must be stored in a separate file as is the current convention with MSORT type files. This file, referenced as the text file in the Link header, must be stored on the server with the main table to maintain data integrity, to provide access to all clients, and for backing up.

If the Format defines one or more text fields then a separate table for the text data is automatically generated when the table for the record data is created. The record data is stored in the table referenced by the Table Name field and the text field data is stored in a table referenced by the Text File field. If present, a dot (.) in the file name is converted to an underscore (_).

The definition of the text table is derived from the record data and a number of preset fields that are used internally by Dictionary-IV. This does not appear as a Dictionary-IV Format but exists as an SQL Server table. A logical Format called IDSV1 has been created to aid in the handling of this information. It is used by the system and is not associated with any specific Link.

The SQL Server table layout is as follows:

```
KEY_FIELD      varchar (len=length of primary key)
TEXT_ID        varchar (len=1)
CREATE_DATE    datetime
CHANGE_DATE    datetime
TEXT_WIDTH     decimal (3,0)
TEXT_FIELD     text
```

KEY_FIELD
Contains the key to the record. This is similar to the current key structure of text field data, except there is no need for a sequence counter since the entire text field is stored in one record. This is possible because the text data type has a maximum storage capacity of 2 gigabytes.

TEXT_ID
Contains the text field identifier. (See Format definition).

CREATE_DATE
Specifies the creation date.

CHANGE_DATE
Specifies the date of the last change.

TEXT_WIDTH
Used by Dictionary-IV for compatibility to local MSORT processing.

TEXT_FIELD
Because this text data has been made available as an SQL table, the data is comprised of purely printable characters (except line terminating $0A$). Therefore any color or attribute information is stripped prior to storage in the table. Graphic characters are converted to a standard ASCII character representation. Changes made from an SQL Server application are available to the Thoroughbred application.

All file level access (create, erase, and clear) performed on the record table is also executed on the text table. However all record level access requires a distinct OPEN of the text table. This is consistent with the current MSORT text field operation.

During I/O processing a FF is implied in the first byte of the key. This byte does not appear in the data but is required for programming consistency. The byte is required for all key access and is returned in all key functions (KEY, FKY, LKY, and PKY).

The Text field data is automatically maintained when accessed via Dictionary-IV. No changes are required to CONNECTs, Scripts, Reports, or Queries that currently use Text fields.

PERFORMANCE TUNING

The following describes how to maximize the throughput using sorts and process cache.

Sorts

To maximize the throughput for a number of the Basic I/O operations that operate on server tables, you must carefully plan the design of the table. The client process creates generic SQL Server tables. It makes no assumption about the data to be contained in the record other than what it determines from the Dictionary-IV Format. This is sufficient to describe the table to SQL Server, but under some conditions, sorting, for example this may not be enough.

SQL Server creates result sets that it generates from statements that specify ordering whether or not an index exists for the columns listed in the ORDER BY clause. Without an index this operation is very time consuming for large tables. Within the Basic environment this is multiplied since each access to the table may require an in-line sort. Basic saves the result set from a particular query and uses it if the subsequent query matches the one that generated the result. If it does not match, the result set must be regenerated the next time the query is executed.

The solution is to create an index for the columns to sort and define at least one column as NOT NULL (a mandatory field in Dictionary-IV). This must be done or the index is ignored for ORDER BY processing.

The create process translates mandatory fields and the first column of the primary key into columns with the NOT NULL constraint if PRM CREATWDBATTR is configured. If these columns are part of indices you need no further changes.

Degradation due to indices being ignored becomes evident in:

Basic
using LKY( ), PKY( ), and I/O with sorts

Dictionary-IV
File access with sorts (multi-record maintenance, view maintenance, sort order change) and backward reads in files (multi-record maintenance up arrow and page up)

Script-IV
READ PREVIOUS, LAST TO FIRST, sorts, etc.

Logon Cache

All access to the SQL Server database occurs through one login to the server using multiple cursors. When a LOCK, EXTRACT, or WRITE is performed in Basic a new login is executed to preserve key pointers to the other tables opened on the original login (LOCKs, EXTRACTs, and WRITE will perform commits and will clear all record locks). Once the LOCK, EXTRACT, or WRITE is complete, this login is terminated. The logon cache count parameter in the IPLINPUT file is used to specify the number of login processes to retain.

For example, if the logon cache count is set to 5, a pool of up to 5 SQL login processes will remain open. The next LOCK, EXTRACT, or WRITE executed by Basic will attach itself to one of these 5 processes bypassing the overhead to login and establish a new process.

By configuring this value properly a balanced maximum can be maintained between excessive login processing and the number of active logins. A maximum of 20 logins can be cached.

Registry Settings

The first Basic client connection will create registry settings, if they do not already exist. These settings can be used to optimize performance. To view the registry settings, run regedit from the Windows start menu. The Thoroughbred entries can be found in: \HEY_LOCAL_MACHINE\SOFTWARE\Thoroughbred Software International\TS DataServer for SQL Server. The registry settings can be used as follows:

CLIENT-SYSTEM-INFO — Not used.

COMMIT COUNT — Database commits are done for all actions (add, drop, insert, update, etc.). For batch processing, you may want to adjust this value so commits are done less often. Set this value to the number of transactions between database commits. For example, if this value is 1000, a database commit will be done after every 1000 transactions.

DATA-SOURCE — Basic will use this Data Source for all SQL connections. This allows users to connect to a common database. The default system Data Source is LocalServer. See SERVER.MAP for additional capabilities.

DEBUG-FLAGS — This entry is used to enable or disable additional information that can be written into the debug log file. The flag values are:

Flag Number	Description
1	Disable logging of the process ID number
2	Enable logging of a time stamp
4	Enable logging of parameter values.
8	Enable logging of fetched values

DEBUG-LEVEL — If this value is non-zero, all SQL statements will be logged. The log file (tsmssql.log) can be found in the installed directory on the Server.

LDA-CACHE — This entry can be set to change the number of Logon Data Areas to cache. Depending activity (read, extract, sort access, etc.), as many as 6 LDAs may be associated with an open Basic channel. See the OPEN-TABLE-CACHE comments below regarding resource usage.

LOGGING — This sets the logging group to something other than the default-logging group.

NO-OPENLOCK	If this entry is non-zero, no Basic locks are done during OPEN. The Basic OPEN will verify the table exists. There is no concept of OPENing a table in SQL. The default lock helps Basic return an Error=0 when a user tries to ERASE a table opened by another user.
OPEN-TABLE-CACHE	A numeric value indicating how many open Basic channels to cache. This will enhance performance in the area of table opens. But, this setting should be used with caution. If this value is set too high, the tsmssql process will become a burden on system resources.
Path	This entry contains the install path and is used for updates.
READUNCOMMITTED	If this entry is non-zero, “WITH (NOLOCK)” will be added to generated SQL statements when READing. This control is enabled by default.
UNIQUE-SORTS	Any non-zero value will indicate all secondary sorts are unique. This will speed up the fetch process. Only set this value, if you’re absolutely sure your sort definitions will generate unique keys. See PRM UNIQUE-KEYS for additional tuning.

USING THE DEBUGGING FACILITY

SQL Output

A debugging facility is available from the tsmssql process. The debugger is activated by either setting the registry entry DEBUG-LEVEL or starting the Manager service with a –dn parameter.

n Currently the only debug level is 1. This sends the generated SQL statements to the log file (tsmssql.log). The output in the log file contains a separate line for each generated SQL statement.

While debugger is active, output for all clients connected to the same SQL Server Group is sent to one tsmssql.log file.

If the tsmssql.log file is copied, output will continue to be placed in the renamed file until the Basic session is terminated. All subsequent Basic sessions would then use a new debug log.

GENERATED SQL CODE

This section shows the resulting SQL syntax for Thoroughbred functions (Italic typeface). They are presented in alphabetical order.

SQL Syntax for Thoroughbred Functions

CLOSE (channel)

COMMIT
This is performed via a direct OCI call, not as a literal SQL statement.

ERASE

DROP TABLE table-name
If text fields are configured: DROP TABLE text table-name

EXTRACT[RECORD] (channel)

SELECT columns FROM table-name WHERE key-conditions >=
current-key ORDER BY sort-information
Get key from first record or current SELECT
SELECT columns FROM table-name WHERE key-conditions =
current-key ORDER BY sort-information FOR UPDATE NOWAIT

EXTRACT[RECORD] (channel,KEY=)

SELECT columns FROM table-name WHERE key-conditions =
current-key ORDER BY sort-information FOR UPDATE NOWAIT

FKY( )

SELECT columns FROM table-name WHERE key-conditions
ORDER BY sort-information
Get the first record.
Extract the key information from the record.

INITFILE

DELETE FROM table-name
If text fields are configured: DELETE FROM text-table-name

KEY( )

If a record is not extracted
SELECT columns FROM table-name WHERE key >
    key of last record read ORDER BY sort-information
Otherwise: Extract the key information from the record.

LKY( )

SELECT columns FROM table-name ORDER BY reverse-sort
Get the first record.
Extract the key information from the record.

LOCK (channel)

SELECT 1 FROM table-name WITH (HOLDLOCK, TABLOCKX) WHERE 1=2

OPEN (channel,OPT="LINK|CREATE")

CREATE TABLE table-name
If text fields are configured: CREATE TABLE text-table-name

For sort0:
CREATE UNIQUE INDEX I_table-name_0

For sortn:
CREATE INDEX I_table-name_n

OPEN (channel, OPT="LINK")

SELECT 1 FROM table-name WITH (HOLDLOCK) WHERE 1=2

OPEN (channel, OPT="LINK|TEXT")

SELECT 1 FROM table-name WITH (HOLDLOCK) WHERE 1=2

PKY( )

If a record is not extracted
SELECT /*+ INDEX_DESC(table-name) */ columns FROM table-name
    WHERE key-conditions ORDER BY reverse-sort
Otherwise: Extract the key information from the record.

PREAD[RECORD] (channel)

If a SELECT is in memory get the next record
Else
SELECT /*+ INDEX_DESC(table-name) */columns FROM table-name
    WHERE key-conditions ORDER BY reverse-sort

READ[RECORD] (channel,KEY=)
FIND[RECORD](channel,KEY=)

If a SELECT is in memory get the next record
    Else
    SELECT columns FROM table-name WHERE key-conditions >=
        current key ORDER BY sort-information

REMOVE (channel,KEY=)

SELECT columns FROM table-name WHERE key-conditions
    FOR UPDATE NOWAIT
DELETE FROM table-name WHERE key-conditions

RENAME

A table rename requires an SQL stored procedure.

UNLOCK (channel)

No SQL equivalent. Data will be committed and Basic's internal lock on the channel will be released.

WRITE[RECORD] (channel,KEY=)

Get key value from the record data
SELECT columns FROM table-name WHERE key-conditions =
    current-key FOR UPDATE NOWAIT
If the record exists:
UPDATE table-name SET column = value,...
Else:
INSERT INTO table-name column ... VALUES (value...)
fi
COMMIT
This is performed via a direct SQL call, not as a literal SQL statement.

Multiple Select Statements

The following Basic commands may produce multiple select statements when sequentially accessing tables with multiple key fields defined in the Format:

READ[RECORD]
FIND[RECORD]
PREAD[RECORD]
EXTRACT[RECORD]
KEY( )
PKY( )
FKY( )

Unsupported Basic Directives

There is no equivalent SQL code for the following Basic directives:

ADD
ADDSORT
FILE
REMSORT

These items are not supported when applied to server tables.

TECHNICAL REFERENCE

The following provides information specific to Script-IV, Dictionary-IV, Thoroughbred Basic, and SQL Server views.

Script-IV/Dictionary-IV

The following information is specific to Script-IV and Dictionary-IV

DATA-FILE IS, SORT-FILE IS

This feature will be supported but the object referenced by this command must be a Link rather than a local data file. The Link can then reference either a server table or a local file. The file type set in this Link must be "M".

Access Subset of Columns

By defining a format that contains a subset of the columns that are defined in a table, access to tables that contain unsupported data types (BIGINT,IMAGE,NTEXT, etc.) is possible. All of the possible I/O directives are supported with no effect on the unreferenced fields.

Basic

Soft Includes

When an OPEN is executed on a server Link, the Format referenced in the Link header is soft included. This means that the Format data area is defined in memory. However, the Format name will not appear in the FMTNL list and will not interfere with any hard includes that take place. The soft include is executed internally as a FORMAT INCLUDE #format-name, OPT=”NONE” so that any data defined in a Format data area at the time of an OPEN of a server Link that references this Format is preserved.

If however a READ or a WRITE is executed using an IOLIST, the Format data area will be modified since the I/O operation uses the Format data area to parse the data into the proper variable values.

Enhanced Access to Tables

Where possible sequential access to tables may provide faster access times during READ operations. This is possible because a particular I/O statement will generate a result set on the server. Subsequent READ operations will simply move through the result set and return the data to Basic. The area where this is most obvious is when a table is sorted by Basic in a way that no index exists in SQL Server. The initial READ will force the sort to take place, but all READS that follow will get the data from the result set without the need to sort again. If any other I/O statement had occurred between READs another result set would have been created forcing a re-sort on the next READ statements. The following is NOT considered a sequential READ:

K$ = KEY(CH); READ (CH,KEY=K$)

This code generates two separate result sets. Each READ that specifies a specific key generates a new result set. A sequential READ looks like:

READ (CH,ERC=2)

Tables

SQL Server Views

Where possible the use of an SQL Server view as the table in Link definition should be avoided. Views will not support all of the SQL statements that are generated by Basic statements. The most common of these will be the SELECT FOR UPDATE which is the EXTRACT directive in Basic. Therefore it is possible that server errors (ERR=150) will occur accessing Views which will interrupt normal program functioning.

Also for the View data to be returned in a sorted order, the sort must be explicitly defined in the definition of the view in SQL Server. Merely defining key fields will not produce sorted results. A sort must be defined in the link header and then used in all I/O statements.

ERROR MESSAGES

The following describes client error messages.

Client/Server Basic Error Messages

150

Explanation: Server system error. For more information refer to TCB(20) to get the server error number. Also see Virtual Error Table below.

171

Explanation: Undefined Link.

Virtual Error Table

Explanation: Errors accessing SQL Server tables will generate text errors in a Virtual Error Table. The Virtual Error Table is defined as the Link: OOLSQLER. Before using the Virtual Error Table, you will need to update this Link to reflect your Server ID.

After any error trying to access SQL Server data, you can check the Virtual Error Table for more detailed SQL Server error messages. Often this table will contain more than one error message, so you will need to read records until you reach end of file. The Virtual Error Table is accessed like an SQL Server table but it is stored in Thoroughbred memory. It doesn't actually make an SQL Server call. The following is sample code on accessing the Virtual Error Table:

```
CH = UNT;
FORMAT INCLUDE #OOFSQLER
OPEN (CH,OPT="LINK") "OOLSQLER";
CLEAR ERC;
WHILE ERC <> 2;
    READ(CH,ERC=2) #OOFSQLER;
    IF ERC = 0
        PRINT #OOFSQLER
    FI;
WEND
```

Thoroughbred Basic Startup Messages

No match for server id in SERVER.MAP

Explanation: The two-character server id defined in the IPLINPUT file cannot be located in the SERVER.MAP file.

Action: Verify contents of both files.

Could not resolve Network address for Thoroughbred foreign file server

Explanation: System does not understand the Network address configured in the SERVER.MAP file.

Action: Verify that the address or host name in the SERVER.MAP file is a valid value for the system. On systems that use DNS, be sure the system is configured.

Invalid Thoroughbred foreign file server type

Explanation: The system found an invalid value in the server type parameter in the IPLINPUT server device line.

Action: Verify that the server device line is defined properly with a Server type of 4.

Cannot mix SQL and Windows Authentication for SQL Server

Explanation: The IPLINPUT file indicates Windows Authentication but also includes an SQL Server Login/Password Authentication.

Action: Either set the Authentication Type to 1 (SQL Server) or remove the SQL Server Login/Password value.

Could not connect to Thoroughbred foreign file server

Explanation: Thoroughbred Basic could not establish communication to the tsmssqlmgr manager.

Action: Verify that the Thoroughbred Manager service is started.

TROUBLESHOOTING

The following provides troubleshooting information specific to SQL Server.

Link Error

Error 0 occurs opening the Link and no locks seem to be active in Basic or from an SQL application.

Since the Link header is read internally by Basic when opening the Link, this record needs to be accessible at open time. If it is extracted through Dictionary-IV, for example, the error 0 occurs. Setting the PRM READONLY parameter in the IPLINPUT file will clear this situation.

Server Table Error

Errors opening server tables.

- Verify that the DEV line for the server where the table resides is present in the appropriate IPLINPUT file.
- Verify the user has SQL Server credentials for the Database and Table.
- Verify the Link has a Table name. If the associated Format has text fields defined, verify that the Link also has a Text File Name.

Unexpected Sorting Sequence or Errors Accessing SQL Tables

- If a Link header has been configured to access a server table defined on the server as a view, unexpected results ranging from incorrectly sorted data to tsmssql process failures may result. It is suggested that access to the server be limited to objects defined as tables. SQL does not support certain operations on views such as SELECT ...FOR UPDATE.
- Be sure that the primary key index is present.
- Check the collation sequence for the SQL Server Database. The default collation sequence may not be ASCII.

Hanging on a WRITE

Under certain circumstances a WRITE may hang if the row is locked in SQL Server using a share (S) or share row exclusive (SRX) type lock. This occurs because the INSERT/UPDATE part of the SQL code for a WRITE statement does not support NOWAIT and will wait until the lock is removed by the SQL Server process. The share and share row exclusive type locks are not generated by any Basic code, and therefore this situation cannot occur between Basic tasks.

Some Access Using Sorts is Very Slow

Sorts that contain references to substrings of fields may not perform as well as sorts on full fields. This is because SQL Server has no facility to define indexes on portions of fields. For more information see the Performance Tuning section of this manual.