Rocket Model 204 SQL Connectivity Guide

Version 7 Release 5.0

September 2014
204–75–SQLCON-01

Notices

Edition

Publication date: September 2014
Book number: 204–75–SQLCON-01
Product version:

Copyright

Trademarks

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Corporate Information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA

Contacting Technical Support

If you have current support and maintenance agreements with Rocket Software and CCA, contact Rocket Software Technical support by email or by telephone:

Email: m204support@rocketsoftware.com

Telephone :

North America	+1.800.755.4222
United Kingdom/Europe	+44 (0) 20 8867 6153

Alternatively, you can access the Rocket Customer Portal and report a problem, download an update, or read answers to FAQs. You will be prompted to log in with the credentials supplied as part of your product maintenance agreement.

To log in to the Rocket Customer Portal, go to:

www.rocketsoftware.com/support

Contents

About this Guide

The *Rocket Model 204 SQL Connectivity Guide* presents information for defining Model 204 intersystem communication using TCP/IP (Transmission Control Protocol/Internet Protocol), to connect to client workstations

The Connect★ Suite requires a TCP/IP network, after which its operation is transparent to Connect★ users.

Audience

This guide is targeted to individuals who work with the Model 204 external communication:

- Model 204 system administrator
- TCP/IP system administrator
- Administrator for your communications product(s)
- PC LAN administrator

A note about User Language and SOUL

Model 204 version 7.5 provides a significantly enhanced, object-oriented, version of User Language called SOUL. All existing User Language programs will continue to work under SOUL, so User Language can be considered to be a subset of SOUL, though the name "User Language" is now deprecated. In this guide, the name "User Language" has been replaced with "SOUL."

Model 204 documentation set

To access the Rocket Model 204 documentation, see the Rocket Documentation Library (http://docs.rocketsoftware.com/), or go directly to the Rocket Model 204 documentation wiki (http://m204wiki.rocketsoftware.com/).

Documentation conventions

This guide uses the following standard notation conventions in statement syntax and examples:

Convention	Description
`TABLE`	Uppercase represents a keyword that you must enter exactly as shown.
`TABLE` *`tablename`*	In text, italics are used for variables and for emphasis. In examples, italics denote a variable value that you must supply. In this example, you must supply a value for *tablename*.
`READ [SCREEN]`	Square brackets ([ ]) enclose an optional argument or portion of an argument. In this case, specify READ or READ SCREEN.
`UNIQUE | PRIMARY KEY`	A vertical bar (|) separates alternative options. In this example, specify either UNIQUE or PRIMARY KEY.
`TRUST | NOTRUST`	Underlining indicates the default. In this example, NOTRUST is the default.
`IS {NOT | LIKE}`	Braces ({ }) indicate that one of the enclosed alternatives is required. In this example, you must specify either IS NOT or IS LIKE.
`item ...`	An ellipsis (. . .) indicates that you can repeat the preceding item.
`item ,...`	An ellipsis preceded by a comma indicates that a comma is required to separate repeated items.
`All other symbols`	In syntax, all other symbols (such as parentheses) are literal syntactic elements and must appear as shown.
`nested-key` `::=` *`column_name`*	A double colon followed by an equal sign indicates an equivalence. In this case, *nested-key* is equivalent to *column_name*.
`Enter your account:` `sales11`	In examples that include both system-supplied and user-entered text, or system prompts and user commands, boldface indicates what you enter. In this example, the system prompts for an account and the user enters **sales11**.
File > Save As	A right angle bracket (>) identifies the sequence of actions that you perform to select a command from a pull-down menu. In this example, select the Save As command from the File menu.
E`DIT`	Partial bolding indicates a usable abbreviation, such as E for EDIT in this example.

1

Overview of Concepts and Terminology

Overview

This chapter presents terms used throughout this guide. Because the terminology used in this guide may differ from that to which you are accustomed, Technical Support recommends that all connectivity system administrators read this chapter.

Basic networking terms and concepts

In this guide, a computer *network* is defined as a collection of *nodes*, connected at the software level by a communication interface.

Different types of nodes exist on a network: application nodes, terminal nodes, printer nodes, and so on. When the term “node” is used in this guide, however, it refers to an *application node*—the point of connection to the network for some system that supports application programs.

Some examples of communication interfaces are:

- Virtual Telecommunication Access Method (VTAM)
- Transmission Control Protocol/Internet Protocol (TCP/IP)

Figure 1-1 is a diagram of a hypothetical network that consists of three nodes. The software communication interface used is TCP/IP.

The transactions originate from the *client nodes*—PCs and workstations (nodes A, B, and C)—which request information from the *server node*, where the Model 204 SQL process runs.

Figure 1-1. Network transactions

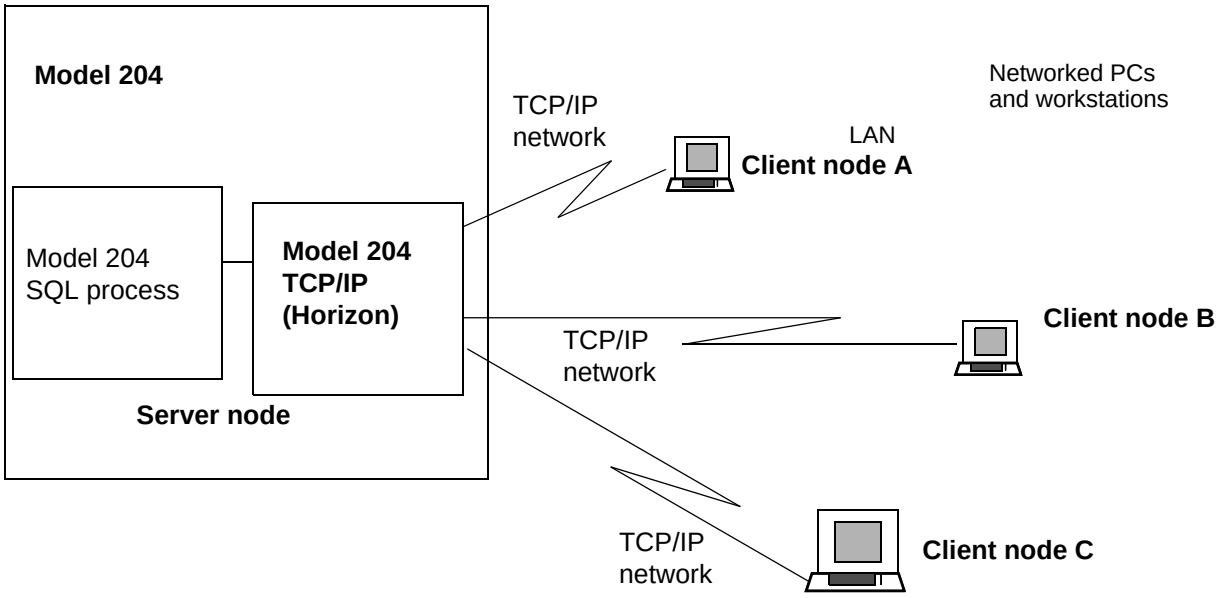


Figure description

Essential elements of Figure 1-1 on page 2 are:

- Exchange of data between two application programs at different nodes is referred to as a *conversation*.
- Two programs that converse are referred to as *conversation partners*.
- Client nodes—the PCs and the workstations—are also end users. An *end user* is defined as a user connected to a Model 204 terminal thread.
- For each end user in a Model 204 TCP/IP (Horizon) network, its operation is transparent.

TCP/IP concepts and terminology

The Transmission Control Protocol/Internet Protocol (TCP/IP) is designed for building network interconnections.

TCP

The *transmission control protocol* (TCP) controls the transfer of the data. This is the layer that provides logical connections between pairs of processes. In TCP, a connection is defined by a pair of sockets, processes that are exchanging information, and a conversation is the communication link between the two processes. TCP provides reliability in data transmission and controls the data flow.

IP

The *internet protocol* (IP) provides the routing mechanism for the data. The internet protocol creates a virtual network view and acts as a layer to hide the underlying physical network.

The *IP address* is a 32-bit address, usually represented in dotted decimal form. The internet protocol uses an IP address to specify two logical addresses:

- *Network address* represents the physical network within the internet
- *Local address* specifies an individual host within the physical network

Ports

A *port* is an identifier used by TCP/IP to distinguish one client node from another in a network connecting many client nodes to a single host (or server).

Identifying Model 204 to TCP/IP

In setting up a client node for the Model 204 TCP/IP (Horizon) network, you provide an IP address and port number combination to identify Model 204. This example shows a valid IP address/port number combination:

```
192.35.46.128:1003
```

where:

192.35.46.128	Is the dotted decimal notation address
1003	Is the port number

Connections

A TCP/IP *connection* transports requests and responses between the client nodes and the server node.

A *conversation* occurs over a connection. In this book, descriptions of Model 204 TCP/IP (Horizon) use conversation and connection interchangeably, because a connection is dropped when a conversation ends.

The Model 204 server process (CCARSQL) receives information from the client nodes, completes the specified action, and returns data to the clients.

Overview of Model 204 SQL processing

This section briefly reviews the components involved in Model 204 SQL processing. SQL processing transport methods are described in “SQL processing transport methods” on page 5. For a more complete discussion of all SQL processing components and related concepts, see the *Rocket Model 204 SQL Server User’s Guide*.

SQL Server

Client/server architecture allows the Model 204 to act as an SQL Server on the host machine, to service clients’ TCP/IP. The SQL Server invokes Model 204 database management system (DBMS) operations, and provides a combination of Model 204 and SQL database functionality. Its main components are:

SQL Engine	Compiles SQL syntax strings, checks SQL semantics, optimizes database access, generates code to accomplish SQL requests, and executes the generated code. Each SQL statement generates data, status information, or both for the requesting application.

SQL Server Front End (SSFE)	Serves as the access layer to the SQL Engine, accepting client request packets, processing each packet's requests for the SQL Engine, and returning the SQL processing result to the client.

The Model 204 SQL Server provides full SQL processing in one of these locations:

- Address space (z/OS)
- Virtual machine (VM)
- Partition (VSE)

SQL Server associated software

The SQL Server works in conjunction with the following Model 204 software:

SQL catalog	CCACAT is the Model 204 system file that houses the SQL catalog information for Model 204 files defined with SQL Data Definition Language (DDL).
SQL communications interface	Receives SQL requests from TCP/IP and passes them to the SQL Server Front End (SSFE). The interface receives result packets from the SSFE and routes them back to the client. This Model 204 module is used only for SQL processing.
SQL Client Front End (SCFE)	Groups and sends client requests, and receives and distributes results to the client. The SCFE is platform independent.

SQL Server supporting tools

These Model 204 tools support SQL Server processing:

Table Specification facility (TSF)	Model 204 subsystem (CCATSF) that provides an interactive, menu-driven facility for mapping existing Model 204 files to SQL tables and columns. The TSF is available on the mainframe only.
Catalog Reporting facility (CCACATREPT)	Model 204 subsystem that provides a menu-driven facility for generating DDL from and reports of the SQL catalog contents. With the CCACATREPT, you can review your SQL object definitions, names, and privileges or use the DDL that it generates to repopulate the SQL catalog. CCACATREPT is available on the mainframe only.

SQL processing transport methods

Transport of the data between the SQL Server and its clients relies on the communication layer: Model 204 TCP/IP (Horizon). These interfaces, which

define client to SQL Server connections, are described in this chapter and illustrated in Figure 1-2 on page 6.

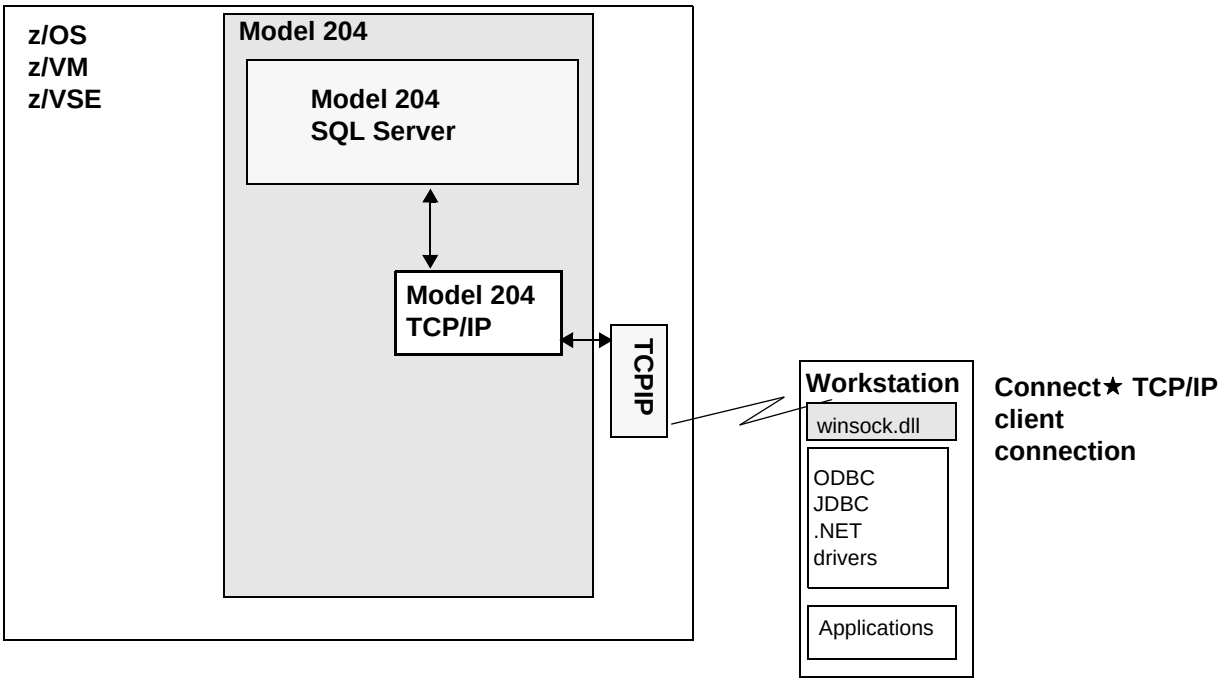
The illustrations include the workstation clients that can be configured with the Connect★ Suite for Model 204 drivers. You can configure Model 204 z/OS, z/VM, or z/VSE TCP/IP connections to workstation clients.

Application program interfaces

Connect★ application programming interfaces are shown in Figure 1-2 as elements in the client configurations. Brief descriptions of these interfaces follow the illustration.

Workstation clients can use the following 32-bit Connect★ Suite for Model 204 drivers, which lets you use dynamic SQL statements in any Windows application that uses Microsoft Open Database Connectivity (ODBC), JDBC, or .NET Framework function calls.

Figure 1-2. Model 204 SQL processing transport methods



Model 204 TCP/IP (Horizon)

Model 204 contains a TCP/IP intersystem processing facility that enables Model 204 applications to participate in program-to-program processing through the Connect★ Suite using the TCP/IP type of connection.

Connect★ requires that Model 204 TCP/IP (Horizon) be installed and running on the mainframe serving as the host machine in your network:

The *Rocket Model 204 Horizon: Intersystem Processing Guide* contains complete information about Horizon.

Along with IBM/TCP-IP software on the mainframe, Model 204 TCP/IP (Horizon) supplies a connection to TCP/IP software on workstations.

Windows Sockets (winsocket) support

A *socket* is a function that makes it possible for an application to access a communications protocol. Windows provides such a function, a Windows Sockets (winsocket) connection.

The Connect★ Suite provides winsocket connection support, which lets you connect to Model 204 using Connect★ and a Windows TCP/IP package that supports winsocket calls.

Interface product to support Windows Sockets

To use Model 204 TCP/IP (Horizon), you need installed and running on your PC or workstation an interface product that supports Windows Sockets.

Other requirements

To complete the TCP/IP connection, you must also make sure that your system meets these requirements:

- IBM/TCP-IP is running on the mainframe
- PCs are able to “talk to” the mainframe

  Use the ping command to check the network connection and make sure that each PC and workstation is communicating with the mainframe.

Procedures for defining the network

The Rocket Model 204 documentation wiki command pages describe the DEFINE LINK, DEFINE PROCESS, and DEFINE PROCESSGROUP commands that you use to define the TCP/IP network to Model 204 for PCs and workstations:

http://m204wiki.rocketsoftware.com/index.php/Category:Commands

For more information

For more information about installing Connect★ on a PC or workstation running Windows and TCP/IP software, consult the appropriate section in the *Connect★ Installation and Programming Guide.*

Information to be gathered

To set up the Model 204 TCP/IP (Horizon) network, you might need to obtain information from the TCP/IP administrator. See Table 4-1 on page 24.

Tasks summary

Table 1-1 summarizes the general tasks that you perform as you progress through this guide.

Table 1-1. List of general tasks

Step	Task	See...
1.	Include SQL-specific files in your online environment.	page 9
2.	Set CCAIN parameters to support Model 204 SQL processing.	page 10
3.	Size the CCATEMP file.	page 12
4.	Define SQL and RCL IODEV threads.	page 12
5.	Size the fixed server area and the variable server area.	page 14
6.	Size SQL buffers.	page 16
7.	Review network definition entities.	page 17
8.	Using Model 204 TCP/IP (Horizon), issue DEFINE commands for: • Link • Processgroup • Process	

2

Specifying the Runtime Environment

Overview

This chapter discusses changes to the job or EXEC that brings up your Model 204 Online—these changes are required or recommended to support Model 204 SQL and RCL processing. This chapter also provides information about sizing SQL buffers.

You also need to specify DEFINE commands for Model 204 TCP/IP (Horizon). These DEFINE commands are described in the Rocket Model 204 documentation wiki command pages:

http://m204wiki.rocketsoftware.com/index.php/Category:Commands

SQL processing files

Table 2-1 lists SQL-specific files you need to include in your Online environment.

Add DD statements or FILEDEFs for these files to the job that brings up your Model 204 Online. Model 204 installation or reinstallation allocates the files.

Table 2-1. SQL files for Online environment

File	Contents
CATPROC	Procedures *and* data: SOUL procedures to run CCACATREPT subsystem and to create the CCACAT file for initial installation and reorganizations; and Help files for CCACATREPT.

Table 2-1. SQL files for Online environment

File	Contents
CCACAT	*Required file*. Model 204 SQL catalog for storage of all SQL DDL definitions.
TSFDATA	Model 204 file for work records that permit you to retain TSF (CCATSF subsystem) working data across sessions.
TSFPROC	Procedures *and* data: SOUL procedures to run CCATSF subsystem, and Help files for the TSF.

CCAIN parameters to support Model 204 SQL processing

Table 2-2 summarizes the recommended values for CCAIN parameters that support Model 204 SQL processing. For complete details on what these parameters support and how to adjust their values, please see the Model 204 documentation wiki parameter pages:

http://m204wiki.rocketsoftware.com/index.php/List_of_Model_204_parameters

When any of these parameters are required for RCL processing, the SQL settings are used as defaults.

Table 2-2. CCAIN parameters affected by SQL processing

Parameter	Recommendation
CUSTOM	Set to 8 to allow RCL clients to issue the LOGIN command. This allows users to logout and login without closing the current connection and opening a new one.
LCPDLST	Use the default, 2176
LHEAP	Set to 200000, then monitor and adjust
LIBUFF	Set to 5000; adjust to match SQLBUFSZ
LNTBL	Monitor NTBL and adjust
LOBUFF	Set to 5000
LPDLST	Set to 32760
LQTBL	Monitor QTBL and adjust
LSTBL	Monitor STBL and adjust
LTTBL	Set to 2000, then monitor TTBL, then adjust
LVTBL	Monitor VTBL and adjust
MAXBUF	Minimum value is 18

Table 2-2. CCAIN parameters affected by SQL processing (Continued)

Parameter	Recommendation
MINBUF	Minimum value is 18. See the Model 204 documentation wiki MINBUF parameter page for the formula used to calculate the appropriate value for MINBUF in SQL processing: http://m204wiki.rocketsoftware.com/index.php/MINBUF_parameter
NSUBTKS	Increase by 3 for first open Model 204 TCP/IP (Horizon) link; by 2 for each subsequent link
SERVSIZE	Initially set this to at least 350K and adjust. For processing SQL statements that are unusually large, the recommended value is between 600K and 700K.
SQLBSCAN	Use default, then adjust
SQLBUFSZ	Set to 100000 on first SQL IODEV line. RCL IODEV uses SQL IODEV setting (default).
SQLCNVER	Review the 0 and 1 settings of the SQLCNVER parameter in the Rocket Model 204 wiki documentation to choose what is best for your application: http://m204wiki.rocketsoftware.com/index.php/SQLCNVER_parameter
SQLFILE	Set to 1 to have Model 204 SQL processing generate a unique file name for the file to which your CREATE TABLE table is mapped in the SQL catalog
SQLIQBSZ	Set to 32752, then adjust
SQLLPLIM	Use default, then adjust

Handling SQL statements greater than 32K bytes

When handling an SQL statement that is greater than 32K, you must consider the following parameters:

- The LCPDLST parameter is the length of the C pushdown list. If its value is not large enough, the error message invoked includes the name of the relevant parameter.
- The SQLBUFSZ parameter is the buffer to collect the entire SQL statement. It must be slightly greater than the maximum size of the largest SQL statement, which is the value of SQLI in the audit trail. For example, set SQLBUFSZ=100000 to run a DDL statement that is 52K.
- The SQLIQBSZ parameter is the buffer for the interface with IFAM. For example, set SQLIQBSZ=32752 to run an IFGET result that requires a buffer greater than 31K.

- Other pertinent parameters to review are VTBL, LHEAP, and SERVSIZE. If their values are not large enough, the error message invoked includes the name of the relevant parameter.

  To allow LHEAP and VTBL to be as large as possible, conserve address space by setting the buffer size parameters as low as possible. Alternatively, you can place CCASERVR in memory to eliminate size restrictions on SERVSIZE.

CCATEMP file size

SQL query processing uses CCATEMP space for sorting rows and for storing subquery results.

Depending on the amount of data being accessed, your typical CCATEMP requirements might increase. You can set CCATEMP as high as 16 million pages. Overallocating CCATEMP wastes disk space and uses excessive memory, so only allocate additional pages if necessary.

Defining SQL and RCL IODEV threads

Model 204 requires a pool of threads to support server control of SQL connections. The system manager must define these threads in the CCAIN stream as part of the Online startup configuration.

For more information about CCAIN and IODEV threads, see the Rocket Model 204 documentation wiki system manager pages:

http://m204stage.rocketsoftware.com/index.php/Defining_the_User_Environment_(CCAIN)

You must define IODEV threads for all Connect★ configurations:

- *IODEV=19* defines a Model 204 TCP/IP SQL thread for TCP/IP.
- *IODEV=49* defines a Model 204 TCP/IP RCL thread for TCP/IP.

Coding the IODEV line

Your definition of an IODEV 19 or 49 thread involves the specification or adjustment of the NOTHREAD, SQLBUFSZ, and LHEAP parameters described in this section. For examples, see “Runtime examples” on page 13.

RCL and SQL multithreaded environment

When defining IODEV=49 (RCL) and IODEV=19 (SQL) threads:

- In the IODEV=49 line and in the IODEV=19, set the total number of concurrent threads (NOTHREAD) and identify the number of each thread.
- In the IODEV=19, set other IODEV parameters such as SQLBUFSZ, LHEAP, LIBUFF, and SQLIQBSZ. The RCL thread uses these SQL thread settings as defaults.

NOTHREAD

You can insert one or more IODEV=19 or 49 lines in the CCAIN stream. On the first line defining the SQL or RCL thread, include the NOTHREAD parameter to indicate the number of threads to be allocated. The number of threads of types 19 and 49 is controlled by the CPUID zap and NOTHREAD must be less than the number of threads defined against the CPUID zap for your site.

Set NOTHREAD to the maximum number of concurrent Connect★ conversations to be supported. Make sure that your Model 204 NUSERS parameter setting accommodates these additional SQL and RCL threads.

SQLBUFSZ

SQLBUFSZ is a required user parameter that defines the maximum incoming SQL message length that can be presented to the SQL Server. The recommended initial value for this parameter is 100000. Specify it on the first IODEV 19 line.

For more information about sizing SQL buffers, see the Rocket Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/SQLBUFSZ_parameter

LHEAP

LHEAP specifies the size of HEAP, a dynamic storage area required for the processing of SQL C routines. HEAP is in the variable portion of the server area allocated to each user, and its size is added to the variable size calculation of the server area. An overall working space of 200,000 bytes is usually sufficient for SQL operations.

The recommended initial value is 200000 for SQL threads. Specify it on the first IODEV 19 line. Do not specify it with the User 0 parameters, because this wastes space. For non-SQL threads, which do not require HEAP space, you can set or reset LHEAP to zero, the default.

You can monitor the high-water mark for use of this storage area with the HEAP statistic, described in “HEAP and PDL high-water marks” on page 28.

As the query complexity increases, you must increase the LHEAP value accordingly, as you encounter SQL error messages. Many SQL error messages offer specific advice. However, if you continue to get SQL error messages after increasing the LHEAP value, your query is too complex for RSQL to handle. You should rewrite your query into smaller, less complex parts.

Runtime examples

Shown in these examples are sample settings of some of the CCAIN parameters described earlier in this chapter, the first two IODEV lines for each SQL IODEV, and Model 204 SQL DEFINE commands.

For all operating systems using TCP/IP RCL and SQL threads

The following example is an excerpt from a job that brings up a Model 204 Online for SQL processing, using a TCP/IP connection.

```
••• LIBUFF=3000,LNTBL=600,LOBUFF=5000,LPDLST=32760,
LQTBL=2000,LSTBL=12000,LTTBL=150,LVTBL=300,
MINBUF=18,MAXBUF=200,NSUBTKS=15,•••
SERVSIZE=700000,•••
   •
   •
   •
IODEV=49,POLLNO=1,NOTHREAD=4
IODEV=49,POLLNO=2
IODEV=49,POLLNO=3
IODEV=49,POLLNO=4
IODEV=19,POLLNO=1,NOTHREAD=4,SQLBUFSZ=100000,LHEAP=200000, -
LIBUFF=6048,SQLIQBSZ=32752
IODEV=19,POLLNO=2
IODEV=19,POLLNO=3
IODEV=19,POLLNO=4
* above IODEV=49 is for RCL connections
* above IODEV=19 is for SQL CONNECT* connections
*-------------------------------------------------------*  /
*  DEFINE CONNECT * SQL AND RCL LINK ETC...              *  /
*-------------------------------------------------------*  /
*********************************************
DEFINE LINK TCPSQL WITH SCOPE=SYSTEM TRANSPORT=TCPSE -
  PROTOCOL=IP LOCALID=ANY INBUFSIZE=4096 CONNECTIONS=8 -
  SERVPORT=2132
DEFINE PROCESSGROUP ANY192   WITH SCOPE=SYSTEM LINK=TCPSQL -
  INLIMIT=8 OUTLIMIT=8 REMOTEID=192.0.0.0  LOGIN=NOTRUST   -
  GUESTUSER=REJECT  MASK=255.0.0.0
DEFINE PROCESSGROUP ANY204  WITH SCOPE=SYSTEM LINK=TCPSQL  -
  INLIMIT=8 OUTLIMIT=8 REMOTEID=204.0.0.0  LOGIN=NOTRUST   -
  GUESTUDEFINE PROCESS CCARSQL WITH SCOPE = SYSTEM -
DATALEN = 32763 FROM = (ANY192, ANY204)
SER=REJECT  MASK=255.0.0.0
*****************************************************
```

Server area sizing

Server areas are the internal work areas allocated to each user. The server area includes a fixed space portion and a variable space portion. This section lists the fixed and variable portions of the server area that are affected significantly by SQL Server processing.

The SERVSIZE system parameter specifies the size of the server area available for all users for the Online run or per IODEV thread. SERVSIZE is the sum of the fixed and variable portions of the server area.

You can calculate SERVSIZE according to formulas provided in the Rocket Model 204 documentation wiki system manager pages:

http://m204stage.rocketsoftware.com/index.php/Defining_the_runtime_environment_(CCAIN)#Server_areas

The fixed and variable portions of the server area are calculated separately. To help you estimate SERVSIZE for Model 204 SQL processing, the following sections list the server area parameters and tables that are affected by SQL processing.

Adjust your typical estimates of these parameters and tables as recommended, and include them in your usual SERVSIZE calculation along with the parameters and tables not affected by SQL processing.

The recommended *minimum* value of SERVSIZE for SQL processing is 350K. For processing SQL statements that are unusually large, the recommended value is between 600K and 700K.

Fixed server area

LIBUFF and LOBUFF are parameters that contribute to the fixed portion of the server area calculation. Their typical settings must be increased to accommodate SQL processing of certain kinds of long input statements or returned data. The maximum for both is 32K. Technical Support recommends that you set LIBUFF close to the value of SQLBUFSZ.

In addition to these parameters, the fixed portion of the server area includes an area (called C PRV in the audit trail) that is used internally by the SQL Engine for storage of C global variables. SQL IODEVs must allow 6K bytes of fixed server size for the C PRV; non-SQL IODEVs do not need to allocate server space for the C PRV.

Variable server area

The variable portion of the server area includes server tables and a dynamic storage area for the processing of SQL C routines.

Server tables are sections of the server area used by the compiler and evaluator to store all the information necessary to run either SOUL or SQL requests. Some server tables are also used by the editor and Host Language Interface (HLI) functions.

Table 2-3 on page 16 lists the sections of the server area used for SQL processing. You might need to resize the controlling parameters to conform to SQL application requirements. Estimate as usual the sizes of the server tables not affected by SQL processing.

For more information about monitoring server tables and the parameters with which you control their size, see the Rocket Model 204 documentation wiki system management pages:

http://m204stage.rocketsoftware.com/index.php/Defining_the_runtime_environment_(CCAIN)#Server_areas

For more information about HEAP, see the discussion of SQL IODEV threads in “Defining SQL and RCL IODEV threads” on page 12.

Table 2-3. Variable server area sections affected by SQL

Section	Contents	Controlling parameter
HEAP	C routine processing	LHEAP
LPDLST	User pushdown list	LPDLST
NTBL	Statement labels, list names, and variables	LNTBL
QTBL	Statements in internal form (quadruples)	LQTBL
STBL	Character strings	LSTBL
TTBL	Temporary work pages	LTTBL
VTBL	Compiler variables	LVTBL

Setting initial sizes for SQL buffers

The system manager must set three SQL buffer parameters for Connect★, which are described in detail in the Model 204 documentation wiki:

- INBUFSIZE parameter of DEFINE LINK,
- DATALEN parameter of DEFINE PROCESS:

  http://m204wiki.rocketsoftware.com/index.php/Category:Commands
- SQLBUFSZ user parameter set in the IODEV line:

  http://m204wiki.rocketsoftware.com/index.php/SQLBUFSZ_parameter

INBUFSIZE

At the mainframe, Model 204 TCP/IP receives the TCP/IP request unit (TU) into a buffer whose size is set by the DEFINE LINK parameter INBUFSIZE. The buffer receives and passes one TU at a time. For Model 204 TCP/IP set INBUFSIZE to 4096 or the TCP/IP TU size in your network.

DATALEN

In Model 204 DATALEN has a maximum setting of 32K. It should be set to the average amount of data in any request in your network.

3

Defining Network Entities

Overview

This overview chapter introduces the network entities that you define to establish Model 204 TCP/IP (Horizon) networks:

- Link
- Processgroup
- Process

Determining the necessary connections

The Model 204 TCP/IP (Horizon) connections that you need to establish are described in this section.

Three network entities

You define the following entities for the Model 204 TCP/IP (Horizon) network:

- Link
- Processgroup
- Process

Definitions you provide for these entities set the physical and logical characteristics of network conversations. For detailed information on

defining these entities, see the DEFINE commands in the Rocket Model 204 documentation wiki command pages:

http://m204wiki.rocketsoftware.com/index.php/Category:Commands

Link

A *link* represents a connection between Model 204 and a network. The link definition specifies the transport mechanism and protocol to use for communications

Processgroup

A processgroup connects one or more processes to a specific link, and groups processes according to certain attributes to facilitate resource allocation. The processgroup definition also specifies the remote partner with which the processes can communicate.

Process

The process definition associates the CCARSQL process (the Model 204 SQL process) with one or more processgroups; the processgroup(s) associate the process with one or more links.

CCARSQL always receives the SQL client's request for a conversation. This same CCARSQL process is used for all connections to the Model 204 SQL Server—one definition of CCARSQL must apply to *all* SQL and RCL connections.

Defining connections

Table 3-1 lists the connections that you must define and the chapter in this book that describes how to define each connection.

You might have to define multiple connections. For example, if you are using Connect★, you must define a TCP/IP connection.

Table 3-1. Connections to define for Connect★ users

If you are...	Define a connection for...	See...
Running SQL or RCL on a PC or workstation using 32-bit Connect★	Model 204 TCP/IP (Horizon)	Chapter 4

Single and multiple definitions

For Connect★, specify DEFINE commands for each of the SQL network entities: link, processgroup, and process.

See the excerpt from the sample Model 204 Online job in “Runtime examples” on page 13.

Where to specify DEFINE commands

The DEFINE commands can be:

- Coded in the CCAIN stream of the job or EXEC you use to bring up your Model 204 Online
- Stored in a SOUL procedure that is invoked by the Online at runtime
- Issued Online from Model 204 command level by a user with system manager privileges

Network control commands

Use network control commands to manipulate the Model 204 TCP/IP (Horizon) network entities (link, processgroup, and process) once you have defined them.

You can place these commands in the User 0 stream or in a Model 204 procedure, or issue them at command level. You must have User 0, system administrator, or system manager privileges.

These commands are listed in Table 3-2 and described in the Rocket Model 204 documentation wiki command pages:

http://m204wiki.rocketsoftware.com/index.php/Category:Commands

Table 3-2. Network control commands

Command	Function
OPEN LINK *name*	Opens the link; required for connection
CLOSE LINK *name*	Disables an opened link
START *type name*	Starts a stopped link or processgroup
STOP *type name*	Stops a link or processgroup
MONITOR *type name*	Displays current usage of a network entity
MODIFY PROCESSGROUP *name*	Modifies a current processgroup definition

4

Using Model 204 TCP/IP (Horizon)

Overview

Model 204 TCP/IP (Horizon) provides a **connection** between Model 204 and PCs and workstations over a TCP/IP network. This connection transports requests and responses between the workstation clients and the Model 204 SQL Server on the mainframe.

You can use Model 204 TCP/IP (Horizon) and Windows Sockets (winsockets) to establish a client connection from a PC or workstation to the TCP/IP software on the mainframe. To use Model 204 TCP/IP (Horizon), you need installed and running on your PC or workstation an interface product that supports Windows Sockets.

Preparing the environment

To implement the Model 204 TCP/IP connection, prepare the Model 204 Online environment by defining TCP/IP SQL and RCL threads and adjusting buffer sizes. For information about preparation of the Model 204 Online, see Chapter 2.

In addition, you must specify the Model 204 TCP/IP link, processgroup, and process entities with Model 204 DEFINE commands. The procedures for creating these definitions are provided in this chapter.

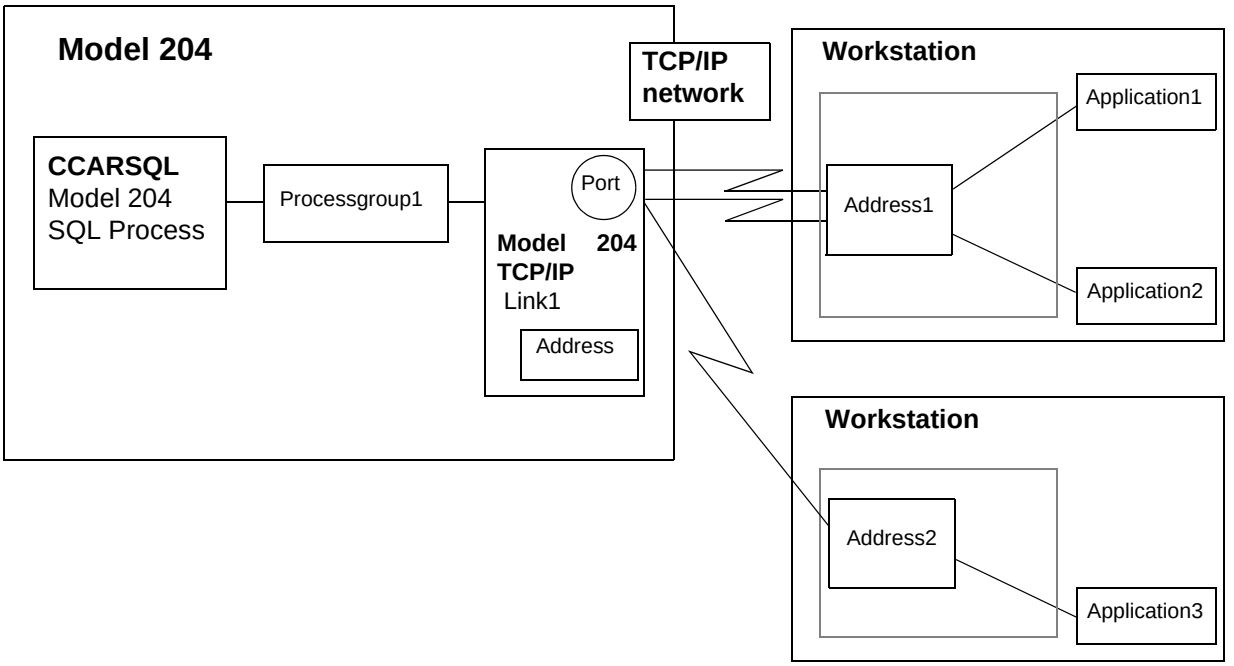
Transparent operation

Once the network is properly defined, the operation of Model 204 TCP/IP is transparent to Connect★ users. For example, the execution of socket verbs is handled internally.

Connecting to Model 204 TCP/IP (Horizon)

Figure 4-1 shows a Model 204 TCP/IP link with three connections to Connect★ clients. Note the interrelated —process, processgroup, and link—that support TCP/IP connections.

Figure 4-1. Horizon for TCP/IP connection support



In Model 204 TCP/IP, a *conversation* occurs over a connection in which the IP protocol for exchanging messages is observed.

See the DEFINE LINK command in the Rocket Model 204 documentation wiki for information about the IP protocol:

http://m204wiki.rocketsoftware.com/index.php/DEFINE_LINK_command:_Horizon_for_TCP/IP

In this chapter, descriptions of Model 204 TCP/IP use *conversation* and *connection* interchangeably, because a connection is dropped when a conversation ends. Model 204 TCP/IP conversation exchanges are always initiated by the Connect★ for an application program. CCARSQL is always the server.

Guidelines

- Model 204 SQL process accepts connection requests only from pre-specified TCP/IP clients.
- Model 204 TCP/IP processgroup definition specifies the network address or addresses of those clients with whom connections are possible.
- Model 204 link definition specifies the network address and port by which the Model 204 SQL server is known to the TCP/IP network.
- A single Model 204 TCP/IP link can support as many as 1999 connections at a time.

DEFINE commands

In Model 204 terminology, the Model 204 SQL Server is the server process, or program, that communicates through a conversation with a client partner (SQL application). The Model 204 DEFINE commands specify the physical and logical characteristics of the conversation and name the clients that can access the SQL Server.

The commands—DEFINE LINK, DEFINE PROCESSGROUP, and DEFINE PROCESS—are all necessary to define a Horizon for TCP/IP connection. These commands are documented in the Rocket Model 204 documentation wiki command pages:

http://m204wiki.rocketsoftware.com/index.php/Category:Commands

Issuing DEFINE commands

The DEFINE commands are typically stored in a SOUL procedure that is invoked by the Model 204 Online job or EXEC at runtime or coded in the Online job or EXEC CCAIN stream. For example,

```
DEFINE PROCESS CCARSQL WITH SCOPE = SYSTEM  -
DATALEN = 32763 FROM = (ANY192, ANY204)
```

The CCAIN settings defined for the SQL thread(s) are used as default settings for the RCL thread(s). They can also be issued Online from Model 204 command level by a user with system manager privileges.

Exchanging Horizon for TCP/IP connection parameters

The Model 204 system manager and TCP/IP system administrator must exchange or jointly determine certain parameter values required to define their connection. Connection characteristics are provided by default.

As summarized in Table 4-1, Model 204 DEFINE command parameters must specify server and client network addresses, the server port number, and connection buffer sizes. For 32-bit Connect★ interfaces, ODBC SQLConnect

and SQLDriverConnect function calls also specify the server's network address and port number, and might specify a Model 204 user ID and password.

Table 4-1. Model 204 TCP/IP connection parameters

Value provided	Model 204 DEFINE command parameter	TCP/IP client specification
Server node (Model 204 SQL) address	DEFINE LINK LOCALID	Host (Server) Name or IP Address parameter in ODBC, JDBC or .NET connection string
Server port number	DEFINE LINK SERVPORT	Port number parameter in ODBC, JDBC or .NET connection string
Client node (TCP/IP Workstation) address	DEFINE PROCESSGROUP REMOTEID MASK	
Model 204 user ID and password for client user (might not be required)	Login depends on setting of: DEFINE PROCESSGROUP LOGIN GUESTUSER	User ID and password in ODBC, JDBC or .NET connection string

For details about TCP/IP client specification, see the *Connect ✭ Suite Installation and Programming Guide*.

DEFINE command interconnections

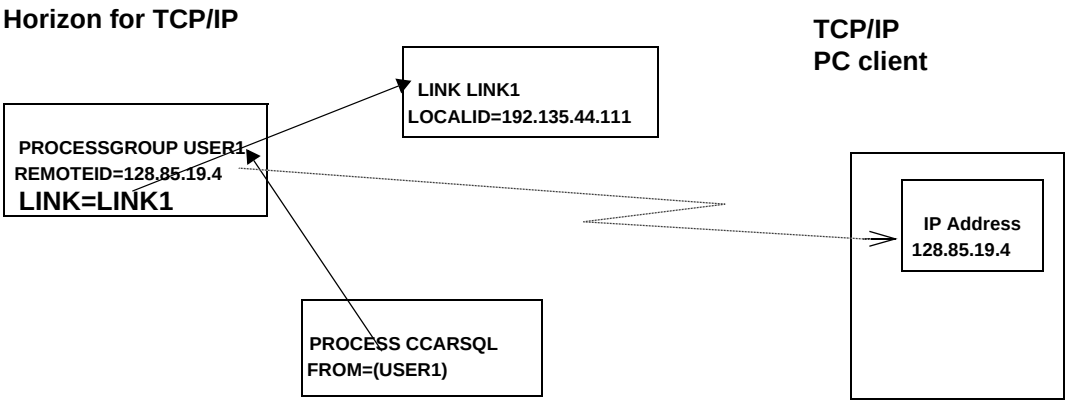
Specify a DEFINE command for each Horizon network entity. Each command has multiple parameters, some of which are optional. The Horizon network entities are:

Link	Identifies the local node (with LOCALID parameter and SERVPORT parameter)
Processgroup	Points to the link (with LINK) and points to the client node (with REMOTEID and, optionally, MASK)
Process	Model 204 SQL process (CCARSQL) points to the processgroups (with FROM)

Figure 4-2 on page 25 shows the parameters with which the commands point to each other and to the client partner. Descriptions of the individual entity

definitions with all their parameters are provided in previous sections of this chapter.

Figure 4-2. Interconnecting parameters of network entity definitions



OPEN LINK command

After you have issued the required entity definition commands, you must issue the OPEN LINK command to enable processing.

The OPEN LINK command prepares a port to accept connection requests. Connection requests are accepted and processed only if the address they come from matches the (masked) address in a processgroup definition that points to an open link.

5

Statistics and Communications Errors

Overview

This chapter describes Model 204 statistics that are specific to Model 204 SQL processing. For complete information on statistics and journal formats, see the Rocket Model 204 wiki system management pages:

http://m204wiki.rocketsoftware.com/index.php/Tracking_system_activity_(CCAJRNL,_CCAAUDIT,_CCAJLOG)

SQL statement statistics

Model 204 SQL processing generates a since-last statistics line after each Prepare, Execute, Open Cursor, logoff, and terminate.

The PROC field in the statistics output contains the SQL statement name. The LAST label indicates the type of statement:

This label...	Indicates...
LAST = PREP	SQL Prepare statement (compilation of an SQL statement)
LAST = EXEC	One of the following: • SQL Execute (execution of a compiled statement) • Execute Immediate (combined compilation and execution) • Open Cursor (execution of the SELECT statements associated with a cursor) statement • SQL user logoff

HEAP and PDL high-water marks

The HEAP statistic indicates the high-water mark for dynamic memory allocation (MALLOC) for processing of C routines. This space is used by the SQL compiler/code generator. The SQL evaluator also uses the C heap for its runtime stack. Because dynamic memory is allocated before the system calls the SQL evaluator, the HEAP high-water mark does not change during a unit of SQL statement execution.

For user subtype ‘01’ entries, the offset of the HEAP statistic is X‘58’, immediately following the OUTPB statistic.

Interpreting RECDS and STRECDS statistics

The RECDS statistic (also used to count records processed by Model 204 FOR and SORT statements) indicates the number of physical cursor advances in an SQL base table and the number of records input to sort. The STRECDS statistic indicates the number of records input to SQL sort.

Multiple fetches of the same physical record without intervening cursor movement count as one read. For example, a sequential fetch of several nested table rows within a parent row increments the RECDS statistic by one.

Output sort records from SQL sort are not counted in the RECDS and STRECDS statistics.

Interpreting the PBRSFLT statistic

The Model 204 SQL sort uses a DKBM private buffer feature to increase the number of concurrent open buffers a user can hold. Use of the private buffer requires prior reservation. The PBRSFLT statistic indicates how many times a user failed to get a private buffer reservation.

In Model 204 SQL, reservation of this buffer is conditional, and the PBRSFLT counter is incremented only for unconditional reservations. Therefore, PBRSFLT is always zero for current SQL users. Failure to obtain this buffer is not reflected in the statistic, but rather is indicated by a negative SQL code and message.

In the record layouts, PBRSFLT follows the SVPAGES statistic and is generated for system subtypes ‘00’ and ‘01’. For example, the offset of PBRSFLT for system type ‘00’ entries is X‘1E0’.

Interpreting the SQLI and SQLO statistics

The SQLI and SQLO since-last statistics record the high-water marks for the SQL logical input and output record lengths, respectively. They indicate the size of the largest SQL request to the Model 204 SQL Server and the size of the largest response from the SQL Server.

You can use SQLI and SQLO to size the buffers that receive and transfer Model 204 SQL requests and responses. These buffer sizes are set by the Model 204 CCAIN parameter SQLBUFSZ and the Model 204 DEFINE command parameters INBUFSIZE.

Interpreting communications errors

When you use or attempt to use Connect★, Model 204 generates messages that track the network activity. Model 204 writes these messages in the audit trail.

If an error occurs involving Connect★, conversation processing is neither started nor stopped, and Model 204 automatically displays status codes and sometimes an error message on the operator's console. Model 204 also writes the error message in the audit trail.

Connect★ errors that involve the underlying communications layer are handled differently from application-level errors.

Communications error display format

Each Model 204 error condition involving the underlying communications layer is displayed as one or two error messages. One of these messages contains the status codes. The other line, if any, contains a Model 204 message that helps to describe the condition that caused the error.

An error message that includes the status codes has the following format:

```
M204.nnnn: COMM ERROR STATUS=nn, STATUSD=nn
```

The STATUS value refers to the status of the communications statement last executed. This value is a number value greater than or equal to zero. A value of 12 or greater indicates that a communications error occurred and identifies the error as belonging to a particular category.

STATUSD is set to the return code that details the specific error condition within a particular STATUS category.

STATUS and STATUSD refer to the Model 204 SOUL $STATUS and $STATUSD function values described in Table 5-1.

$STATUS and $STATUSD codes

Table 5-1 lists the possible combinations of return values of the $STATUS and $STATUSD functions and a brief description of each combination.

The combinations are listed by ascending $STATUSD value (SD column) for each $STATUS value (S column).

Table 5-1. $STATUS and $STATUSD codes

S	SD	Description
0	0	Normal completion; no unusual conditions.
1	0	Normal completion; check RESULT variable.
1	1	Specified event(s) not pending (WAIT or TEST).
1	2	Specified event(s) still pending (TEST).
1	3	WAIT completed due to timeout.
2	2	SEND ERROR statement issued by partner.
3	3	State check: verb issued in wrong conversation state.
4	0	Conversation ended normally by partner process. The partner process has finished executing, or it has ended the conversation by issuing a CLOSE PROCESS statement.
4	1	Partner closed conversation abnormally.
5	1	Password is required. If the USERID keyword parameter is provided in the OPEN PROCESS statement, then PASSWORD parameter must also be provided.
5	2	Process is already open. The OPEN PROCESS statement being executed specifies a process name or CID for a process that is already open.
5	3	Initial (PIP) data incorrectly specified.
5	4	Entity (process, processgroup, or link) not defined.
5	5	Process has not been opened.
5	6	Statement or option not supported.
5	11	Process can be opened only within an application subsystem.
5	12	Process definition requires ìOPENî for security parameters. The OPEN PROCESS statement being executed contains an ACCOUNT or USERID parameter. This is valid only when the process definition specifies ACCTSOURCE=OPEN or UIDSOURCE=OPEN (PROFILE and PROFSOURCE can be written instead of ACCOUNT and ACCTSOURCE).
5	13	Security violation.
5	14	SESPARMS was specified on the processgroup definition but is not valid for Horizon or Horizon links.
5	15	OPEN type conflicts with PROCESS type.
5	16	Reserved names cannot be used in OPEN PROCESS.

Table 5-1. $STATUS and $STATUSD codes (Continued)

S	SD	Description
5	17	Process name or CID is too long.
5	18	Synchronization level not supported. CONFIRM verb requires the process definition to have the CONFIRM parameter.
5	19	Required parameter not specified.
5	20	Invalid duration value specified for WAIT statement.
10	1	Insufficient main storage available at local node.
10	2	Reserved.
10	3	Local link is not open or is being stopped or closed.
10	4	Local processgroup is being stopped.
11	1	Partner process temporarily cannot be started.
11	3	Remote allocation failure that you can retry.
12	1	Link failure.
13	1	Session or connection failure.
13	2	Reserved.
13	3	Connection failure for parallel session setup.
50	1	Local session or connection limit has been reached.
50	2	Local conversation limit has been reached.
50	3	Server process subsystem could not be started because APSY is not initialized.
50	4	Subsystem is not available. The requested server process subsystem is not defined in CCASYS.
50	5	Input buffer too small for SUBSYSPARM. The SUBSYSPARM value specified in the process definition was larger than the size in the IODEV 27 command line.
50	6	Synchronization level not supported by local process. Requesting process and local process must agree on CONFIRM or NOCONFIRM.
50	7	Server security violation.
51	1	Server process not available. Examine messages sent to operatorís console or audit trail at server node to diagnose.
51	2	Synchronization level not supported by server. The server process definition does not specify the same synchronization level as that of the local process (CONFIRM/NOCONFIRM).
52	1	Link failure.
53	1	Session or connection failure. Check sense code in $ERRMSG.

Table 5-1. $STATUS and $STATUSD codes (Continued)

S	SD	Description
53	2	Conversation timed out waiting for response from partner.
53	3	Session or connection not established due to incorrect session or connection parameters.

6

Security Considerations

Overview

This chapter presents the security issues to consider in establishing a network connection to Model 204.

The server system in a Horizon network is the Model 204 SQL Server. You establish security for the server system through DEFINE commands, by specifying answers to these questions:

- Which systems can access the Model 204 server?
- Which users in those systems can access the Model 204 server?
- Which programs on the Model 204 server can these users access?

Server system security

The following issues are of concern to the manager of the server system:

- Controlling which systems can make requests of the server

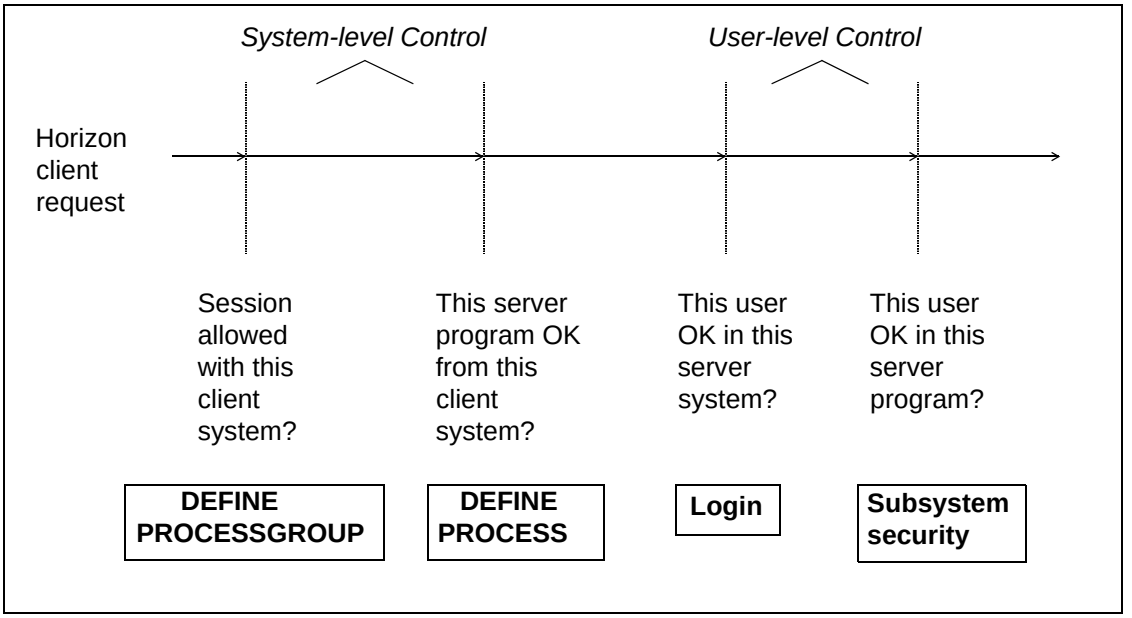
  The processgroup and process definitions used to do this are discussed in this section.

- Specifying which individual users from a given system can access the server, and which server programs they can use

  The login security measures used to accomplish these tasks are described in the next section of this chapter, “Model 204 login security” on page 35.

Figure 6-1 shows how these issues represent layers of protection for the server system, and how access is controlled. Refer to it during the discussion of access to a server system.

Figure 6-1. Lines of defense for a server system



Processgroup definition

For a client system to establish a TCP/IP session with a Model 204 server, the client's network node name must be coded in the REMOTEID parameter of a server's DEFINE PROCESSGROUP command.

Because this command requires system manager privileges, a given client system must be specifically authorized by a server's system manager to gain access to the server.

Process definition

For a given server program (application subsystem or procedure) to be accessible from any client system, the server system manager must include the name of a processgroup that defines that client system in the FROM parameter of the server program's DEFINE PROCESS command.

The system manager is thus required to tell Model 204 explicitly which nodes in the network can access any server program.

For information on DEFINE commands

To learn how to define processgroups and processes, see the appropriate DEFINE commands in the Rocket Model 204 documentation wiki command pages*:*

http://m204wiki.rocketsoftware.com/index.php/Category:Commands

Model 204 login security

Login security verifies a client end user's identity before allowing a Connect★ conversation. After verification, the user receives privileges (file access and command authority) according to user ID or default. The privileges are those defined in CCASTAT (the Model 204 password table) or the external security package CA-ACF2, CA-Top Secret, or Security Server, if a security interface is active.

No explicit LOGIN command or password prompt is used. Connect★ client users enter user IDs and passwords on the Model 204 ODBC Driver - Configure Data Service screen or in an SQLConnect or SQLDriverConnect function for ODBC, or in the User and Password parameters of the JDBC connection URL, or in the Login ID and Password parameters of the .NET connection string. When a client's initial SQL request is evaluated, Model 204 verifies the user identification the client shipped.

You define login security with:

- LOGIN and GUESTUSER parameters of the DEFINE PROCESSGROUP command
- Model 204 SYSOPT parameter

Types of login security

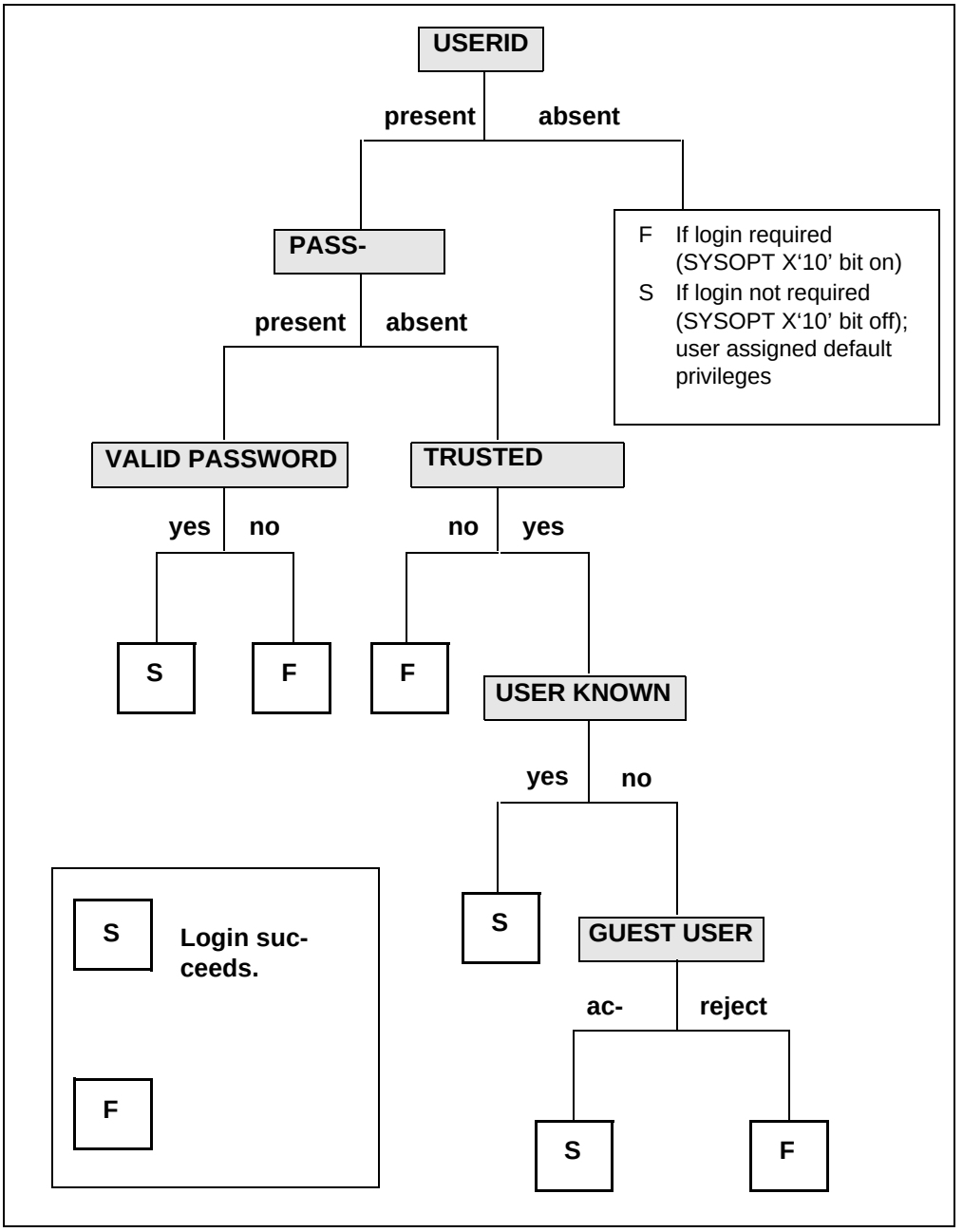
You can define the following types (not mutually exclusive) of login security:

Login security type	Requirements
No trust (default)	Requires a known user ID and valid password
Trusted	**Does not require a password, but must be defined to Model 204**
Guest	Does not require a password and does not need to be defined to Model 204
Login not required	Does not require a user ID

Login success depends on the type of login security in effect and whether the client presents a user ID, a password, or both.

The outcomes that result from the various combinations of processgroup parameter settings, SYSOPT settings, and client choices of user ID and password are shown in Figure 6-2.

Figure 6-2. Login security processing decision tree



Trusted login

The LOGIN and GUESTUSER parameters offer extensions to Model 204 login security. If you trust that a client node reliably verifies the validity of its users, you can define the node to be either:

- Trusted (LOGIN=TRUST)
- Guest (GUESTUSER=ACCEPT)

Users from the node can then be logged in for SQL Server access without having to present a password.

Known user ID

A user from a trusted node must have a known user ID but no password to be logged in successfully. A *known user ID* is one that is defined in either the Model 204 password table (CCASTAT) or in CA-ACF2.

If a user ID is not defined in CCASTAT but is defined in Security Server (formerly RACF) or CA-Top Secret, it is considered unknown. Horizon gives the trusted user the login user privileges defined for their user ID.

Guest users

A user from a guest node must have a user ID (known or not) but no password to be logged in successfully. Allowing guest users relieves the Model 204 system manager from having to define every client user ID. Guest users receive minimal (X‘00’) Model 204 user privileges. Connect★ software on the client node automatically satisfies the requirement that logins from trusted and guest clients send no password to Model 204. If an end user enters a password, it is stripped when the client request is shipped.

Negotiating trust or guest

For client users to be trusted, make complementary specifications for Model 204 and Connect★. For Model 204 TCP/IP, Model 204 requires a processgroup definition that specifies LOGIN=TRUST. For Connect★, the client requires a password specified as a null string (“ ") in the client connection string.

Login not required

When the system manager sets the SYSOPT parameter X‘10’ bit off, Model 204 login is not required. A conversation can occur even if no user ID is present on the client request. When login is not required, the outcome of Horizon login processing depends on whether a user ID is present:

- If a user ID is not present, Model 204 allows the conversation and assigns default user privileges: a user ID of “NO USERID,” an account of “NO ACCOUNT,” and Superuser privileges.
- If a user ID is present, Model 204 login security processing proceeds just as if login were required:

If login requirements are...	Then...
Met	User is assigned the user ID passed in the request and the login privileges that the system manager has assigned to that user ID.
Not met	Login fails; no conversation is allowed.

Client system security

In general, follow the security procedures outlined in documentation for your client system’s operating system.

The system manager of the client system must also use the Model 204 DEFINE PROCESSGROUP and DEFINE PROCESS commands to define the server systems and programs with which the client can communicate.

SQL security user exits

This feature allows clients to replace the SQL statement security provided by the Model 204 SQL catalog (CCACAT) with the security of an external package such as CA-Top Secret (or Security Server or CA-ACF2) to control the use of SQL data definition language (DDL) and access to SQL objects (DML). Instead of the SQL Server checking privileges, clients can call external security “user exits” that they have written.

Whenever the SQL Server needs to check DDL or DML privileges, it first calls a user exit. The call is executed only if an external security package is active for SQL. Otherwise, standard Model 204 SQL privilege checking is performed.

The SQL Server passes to the user exit all the information necessary to perform privilege checking identical to the Model 204 SQL privilege checking. The extent of the checking done is an option of the user exit.

For more information about SQL security user exits, see the Rocket Model 204 documentation wiki security interfaces pages:

http://m204wiki.rocketsoftware.com/index.php/SQL_security_exits

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X