SGA API: A Database Statistics Application Programming Interface

This guide describes the system global area (SGA) application programming interface (API) that you can use to retrieve database performance statistics.

Revision/Update Information: Release 7.1

Copyright © Oracle Corporation, 1996. All rights reserved.

This software contains proprietary information of Oracle Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error-free.

Restricted Rights Legend Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the programs shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication and disclosure of the programs shall be subject to the restrictions in FAR 52.227-14, Rights in Data-General, including Alternate III (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065.

The programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It is the licensee's responsibility to take all appropriate fail-safe, back up, redundancy and other measures to ensure the safe use of such applications if the programs are used for such purposes, and Oracle disclaims liability for any damages caused by such use of the programs.

Oracle is a registered trademark of Oracle Corporation, Redwood City, California.

Oracle Rdb, Oracle RMU, Oracle RMUwin, Oracle SQL/Services, and Rdb7 are trademarks of Oracle Corporation, Redwood City, California.

All other company or product names are used for identification purposes only, and may be trademarks of their respective owners.

Contents

1 PREFACE 1
1.1 Purpose 1
1.2 Supported Oracle Rdb Releases 1
1.3 Sample Program 1
1.4 Files for the SGAAPI 1

CHAPTER 1 PROGRAMMING WITH THE SGA API 1–1

1.1 SGA API DESIGN 1–1
1.1.1 Program Routines 1–1
1.1.2 Program Flow 1–1
1.1.3 Sample Program 1–2
1.1.3.1 Compiling and linking the sample program • 1–2
1.1.3.2 Running the sample program • 1–2
1.1.4 Database Context 1–3
1.1.5 Information and Result Buffer Formats 1–3
1.1.5.1 Information Buffer • 1–4
1.1.6 Result Buffer 1–5
1.1.7 Continuation Context 1–6
1.1.8 Error Tags 1–7
1.1.9 General tags 1–7
1.1.10 Compiling and Linking 1–8

1.2 CLASS AND ITEM TAGS FOR ORACLE RDB DATABASES 1–8
1.2.1 Time and Date Tags (Oracle Rdb only) 1–8

1.3 RMUST_CT_AIJINFO - AIJ INFORMATION 1–9
1.3.1 Information Buffer Syntax 1–9
1.3.2 Result Buffer Syntax 1–10
1.3.3 Tag Descriptions 1–11
1.3.4 Version specific tags 1–15

1.4 RMUST_CT_AIJJNLS - AIJ JOURNAL CLASS 1–15
1.4.1 Information Buffer Syntax 1–16
1.4.2 Tag Descriptions 1–17

1.5 RMUST_CT_ARPMS_STATS - AREA PERFORMANCE STATISTICS 1–18

Contents

1.5.1 Buffer Syntax 1–18
1.5.2 Result Buffer Syntax 1–18
1.5.3 Tag Descriptions 1–19

1.6 RMUST_CT_DBINFO - DATABASE INFORMATION 1–19
1.6.1 Information Buffer Syntax 1–19
1.6.2 Result Buffer Syntax 1–20
1.6.3 Tag Descriptions 1–20

1.7 RMUST_CT_DBR - DATABASE RECOVERY PROCESS INFORMATION 1–22
1.7.1 Information Buffer Syntax 1–22
1.7.2 Result Buffer Syntax 1–22
1.7.3 Tag Descriptions 1–23

1.8 RMUST_CT_FILE - FILE INFORMATION 1–24
1.8.1 Information Buffer Syntax 1–24
1.8.2 Result Buffer Syntax 1–25

1.9 RMUST_CT_FILE_NODUP - FILE INFORMATION 1–27

1.10 RMUST_CT_PROCESS - PROCESS INFORMATION 1–27
1.10.1 Information Buffer Syntax 1–27
1.10.2 Result Buffer Syntax 1–28
1.10.3 Tag Descriptions 1–28

1.11 RMUST_CT_STALL - STALLED PROCESS INFORMATION, RMUST_CT_STALL_ACT - STALLED PROCESS INFORMATION 1–29
1.11.1 Information Buffer Syntax 1–29
1.11.2 Result Buffer Syntax 1–30
1.11.3 Tag Descriptions 1–31

1.12 RMUST_CT_RTPMS_STATS - RUN TIME PERFORMANCE STATISTICS 1–32
1.12.1 Buffer Syntax 1–32
1.12.2 Result Buffer Syntax 1–32
1.12.3 Tag Descriptions 1–32

CHAPTER 2 SGA API ENTRY POINT DESCRIPTIONS 2–1

2.1 KUSCX_ALLOCATE 2–1
2.1.1 Interface 2–1
2.1.2 Description 2–1
2.1.3 Parameters 2–1
2.1.4 Condition Values Returned 2–1
2.1.5 Example 2–1

2.2 KUSCX_FREE_HANDLE 2–1
2.2.1 Interface 2–2
2.2.2 Description 2–2
2.2.3 Parameters 2–2
2.2.4 Condition Values Returned 2–2
2.2.5 Example 2–2

2.3 KUSCX_INIT 2–2
2.3.1 Interface 2–2
2.3.2 Description 2–2
2.3.3 Parameters 2–2
2.3.4 Condition Values Returned 2–3
2.3.5 Example 2–3

2.4 KUSDB_ALLOCATE 2–3
2.4.1 Interface 2–3
2.4.2 Description 2–3
2.4.3 Parameters 2–3
2.4.4 Condition Values Returned 2–3
2.4.5 Example 2–3

2.5 KUSDB_CONNECT 2–4
2.5.1 Interface 2–4
2.5.2 Description 2–4
2.5.3 Parameters 2–4
2.5.4 Condition Values Returned 2–5
2.5.5 Example 2–5

2.6 KUSDB_DISCONNECT 2–5
2.6.1 Interface 2–5

2.6.2 Description 2–5
2.6.3 Parameters 2–5
2.6.4 Condition Values Returned 2–5
2.6.5 Example 2–5

2.7 KUSDB_FREE 2–5
2.7.1 Interface 2–6
2.7.2 Description 2–6
2.7.3 Parameters 2–6
2.7.4 Condition Values Returned 2–6
2.7.5 Example 2–6

2.8 KUSDB_ERROR_TEXT 2–6
2.8.1 Interface 2–6
2.8.2 Description 2–6
2.8.3 Parameters 2–6
2.8.4 Condition Values Returned 2–7
2.8.5 Example 2–7

2.9 KUSDB_GET_INFO 2–7
2.9.1 Interface 2–7
2.9.2 Description 2–7
2.9.3 Parameters 2–8
2.9.4 Condition Values Returned 2–8
2.9.5 Example 2–9

2.10 KUSTV_ALLOCATE_HANDLE 2–9
2.10.1 Interface 2–9
2.10.2 Description 2–9
2.10.3 Parameters 2–10
2.10.4 Condition Values Returned 2–10
2.10.5 Example 2–10

2.11 KUSTV_DUMP 2–10
2.11.1 Interface 2–10
2.11.2 Description 2–10
2.11.3 Parameters 2–11
2.11.4 Example 2–11

2.12 KUSTV_FREE_HANDLE 2–11

2.12.1 Interface 2–11
2.12.2 Description 2–11
2.12.3 Parameters 2–12
2.12.4 Condition Values Returned 2–12
2.12.5 Example 2–12

2.13 KUSTV_GET 2–12
2.13.1 Interface 2–12
2.13.2 Description 2–12
2.13.3 Parameters 2–12
2.13.4 Condition Values Returned 2–13
2.13.5 Example 2–13

2.14 KUSTV_OFFSET 2–13
2.14.1 Interface 2–13
2.14.2 Description 2–13
2.14.3 Parameters 2–13
2.14.4 Condition Values Returned 2–13
2.14.5 Example 2–14

2.15 KUSTV_PUT 2–14
2.15.1 Interface 2–14
2.15.2 Description 2–14
2.15.3 Parameters 2–14
2.15.4 Condition Values Returned 2–14
2.15.5 Example 2–15

2.16 KUSTV_REINIT_HANDLE 2–15
2.16.1 Interface 2–15
2.16.2 Description 2–15
2.16.3 Parameters 2–15
2.16.4 Condition Values Returned 2–15
2.16.5 Example 2–15

2.17 KUSTVP1_PUT_ONE_BYTE 2–16
2.17.1 Interface 2–16
2.17.2 Description 2–16
2.17.3 Parameters 2–16
2.17.4 Condition Values Returned 2–16
2.17.5 Example 2–16

2.18 KUSTVP2_PUT_TWO_BYTES 2–17
2.18.1 Interface 2–17
2.18.2 Description 2–17
2.18.3 Parameters 2–17
2.18.4 Condition Values Returned 2–17
2.18.5 Example 2–17

2.19 KUSTVP4_PUT_FOUR_BYTES 2–18
2.19.1 Interface 2–18
2.19.2 Description 2–18
2.19.3 Parameters 2–18
2.19.4 Condition Values Returned 2–18
2.19.5 Example 2–18

2.20 KUSTVPTG_PUT_TAG 2–19
2.20.1 Interface 2–19
2.20.2 Description 2–19
2.20.3 Parameters 2–19
2.20.4 Condition Values Returned 2–19
2.20.5 Example 2–19

TABLES

1 SGAAPI Files 1
1–1 SGA API Routines 1–1
1–2 Program Flow 1–2
1–3 Error Tags 1–7
1–4 General Tags 1–7
1–5 Classes for Oracle Rdb Databases 1–8
1–6 Date and Time Return Value Types 1–9
1–7 Formatted Date and Time Fields 1–9
1–8 RMUST_CT_AIJINFO Tags 1–11
1–9 AIJ Operator Notification Tags 1–13
1–10 Status values for hot standby log servers 1–14
1–11 Lowest version of Oracle Rdb required for tag support 1–15
1–12 AIJ File Information Tags 1–17
1–13 AIJ File State Tags 1–18
1–14 RMUST_CT_STATS Tags 1–19
1–15 RMUST_CT_DBINFO Tags 1–20
1–16 RMUST_CT_DBR Tags 1–23
1–17 DBR Activity Phase Tags 1–23
1–18 RMUST_CT_FILE Tags 1–25
1–19 File Access Mode Tags 1–27

1–20 RMUST_CT_PROCESS Tags 1–28
1–21 RMUST_CT_STALL Tags 1–31
1–22 RMUST_CT_RTPMS_STATS Tags 1–32

1 Preface

Oracle Rdb maintains an extensive set of online performance statistics that provide valuable dynamic information regarding the status of an active database. The system global area (SGA) application programming interface (API) described in this document provides a way to retrieve these database performance statistics.

1.1 Purpose

The SGA API automates retrieving database statistics available only through the RMU Show Statistics command. The SGA API provides the only way to retrieve statistics for Oracle Rdb databases from an application. Using the SGA API provides fast access to the data without effecting the execution of the server. In addition, the information in the database global areas is protected from corruption by the user.

1.2 Supported Oracle Rdb Releases

The SGA API supports Oracle Rdb Releases 6.0, 6.1, 7.0 and 7.1.

1.3 Sample Program

Refer to the $$$RDBSGA_API.README file included with the software kit for information about the RDBSGA_SAMPLE.C program. The sample program provides a working example of how to code the API. Information about compiling, linking, and running the program are included in the README file.

1.4 Files for the SGAAPI

The SGAAPI files are supplied by the Rdb installation procedure. Primarily they are stored in the RDM$DEMO directory though the KUSRMUSHRxx image is located in SYS$COMMON:[SYSLIB] (which is SYS$LIBRARY).

Table 1 SGAAPI Files

Location	File Name	Description
RDM$DEMO	KUSAPI.H	Header file containing interfaces to API entry points.
	KUSRMUAPI.H	Primary header file for Rdb V70 implementation of the API. This is the only file you need to include in your source code.
	KUSRMUSHRxx.OPT	Options file for linking your programs.
	RDBSGA_SAMPLE.C	Sample file showing how to use the API.
	KUSTAGS.H	Header file containing tag values passed to the API.
	RMUST_RTPMS.H	Rdb7 specific header file containing the RTPMS_t data structure returned from a request for RMUST_CT_RTPMS_STATS data. Also contains the FIOVEC_t structure returned from a RMUST_CT_ARPMS_STATS request.
	ORATYPES.H	Standard data types.
	SGAAPI.PS, SGAAPI.TXT, SGAAPI.HTML	Documentation for the API.
SYS$LIBRARY	KUSRMUSHRxx.EXE	VMS shareable image containing API implementation.

1 Programming with the SGA API

1.1 SGA API Design

1.1.1 Program Routines

Table 1 shows a list of routines supplied by the SGA API.

Table 1–1 SGA API Routines

Routine	Purpose
kuscx_allocate	Allocate continuation context
kuscx_free_handle	Free continuation context
kuscx_init	Initialize continuation context
kusdb_allocate	Allocate resources for a database handle
kusdb_connect	Connect to a database
kusdb_disconnect	Disconnect from a database
kusdb_error_text	Format an error message
kusdb_free	Free resources for a database handle
kusdb_get_info	Collect statistics for the database
kustv_allocate_handle	Allocate and initialize resource for a TLV handle
kustv_dump	Diagnostic dump routine for debugging purposes
kustv_free_handle	Free resources for a TLV handle
kustv_get	Retrieve next TLV from buffer
kustv_offset	Return current offset of read/write pointer
kustv_put	Insert a TLV triplet into buffer
kustv_reinit_handle	Reset offset for TLV handle
kustvp1_put_one_byte	Insert a TLV with associated byte passed by value
kustvp2_put_two_bytes	Insert a TLV with associated 2 bytes passed by value
kustvp4_put_four_bytes	Insert a TLV with associated 4 bytes passed by value
kustvptg_put_tag	Insert a TLV consisting of tag only (0 length)

1.1.2 Program Flow

The following table describes the order in which a program typically calls the SGA API functions:

Table 1–2 Program Flow

Step	Task	API Routine
1	Connect to the database	kusdb_connect
2	Build a tag length value (TLV) buffer, indicating the statistics you want to retrieve and allocate the input TLV handle	kustvptg_put_tag or kustvp4_put_four_bytes
3	Allocate the "wildcard context" (use this when the buffer might be too small for the data)	kuscx_allocate
4	Gather the requested statistics	kusdb_get_info
5	Initialize context for reading the TLV buffer	kustv_allocate_handle or kustv_reinit_handle
6	Loop-over call to get next TLV entry in the TLV buffer	kustv_get

7	Program flow continues as follows:
	• If the TLV buffer continues, program processing returns to step 4 to read the next buffer full of data
	• If the TLV buffer does not continue, the program either:
	  • Returns to step 2 to build a new request
	  • Disconnects
		kusdb_get_info kustvptg_put_tag kusdb_disconnect

1.1.3 Sample Program

You can get a feel for how to code against the API by looking at RDBSGA_SAMPLE.C in the RDM$DEMO directory.

1.1.3.1 Compiling and linking the sample program

When writing your own application, include kusrmuapi.h in any modules which use the API.

To compile rdbsga_sample.c use a command similar to: *$ CC/DECC /INCLUDE=(RDM$DEMO) RDBSGA_SAMPLE.C*

To link the rdbsga_sample program, use a command similar to: *LINK RDBSGA_SAMPLE,RDM$DEMO:KUSRMUSHRxx.OPT/OPTIONS*

1.1.3.2 Running the sample program

The sample program must be run as a foreign command. This is some example output from RDBSGA_SAMPLE when run against the MF_PERSONNEL.RDB sample database:

```
$ rdbsga_sample == "$mydisk:[mydir]rdbsga_sample.exe
$ rdbsga_sample MF_PERSONNEL.RDB
================================================================
=                                                              =
=              Class RMUST_CT_DBINFO                           =
=                                                              =
================================================================
----- Beginning of group -----
Max user count: 50
Creation date: 25-OCT-1995 16:24:19.69
----- End of group -----
* Buffer ended normally *
================================================================
=                                                              =
=              Class RMUST_CT_ARPMS                            =
=                                                              =
================================================================
----- Beginning of group -----
Storage area ID: 4
ARPMS file I/O syread 0
----- End of group -----
* Buffer ended normally *
================================================================
=                                                              =
=              Class RMUST_CT_RTPMS                            =
=                                                              =
================================================================
----- Beginning of group -----
RTPMS stats root reads: 23
----- End of group -----
* Buffer ended normally *
    .
    .
    .
```

1.1.4 Database Context

A database handle is used to maintain context information for the database being examined. All routine names that begin with kusdb require a database handle as one of their arguments. The kusdb_allocate routine is used to allocate resources and initialize the handle so that it can be used by other routines. Typically, kusdb_connect is called to connect to a database. Once a handle has been used to successfully connect to a database, kusdb_get_info can be called to retrieve information from the database. When no more data is needed for the database, kusdb_disconnect can be called to disconnect from the database. When the handle is no longer needed, kusdb_free is called to free resources used by the handle. If kusdb_connect, kusdb_get_info or kusdb_disconnect ever return an error, then kusdb_error_text can be called to retrieve a text message associated with the error as well as any additional messages associated with the error.

1.1.5 Information and Result Buffer Formats

The most important structures used to communicate between the API and the caller of the API are two buffers that are called the information buffer and the result buffer. They are both structured as series of tag-length-value (TLV) entries. The tag and length are both unsigned words, ub2 (see the section titled Compiling and Linking on page 10 for more information on data types and header files). The length indicates the size, in bytes, of the value portion of the triplet. In many cases, a TLV entry will have a length of zero. For example, TLV entries in an information buffer usually have a length of 0.

1.1.5.1 Information Buffer

The information buffer is a buffer that is created and initialized by the code calling the API. This buffer contains TLVs that describe the information to be returned. Along with identifying fields to be returned, it also can specify how much data to return (for example, one TLV can specify that information for one process is to be returned while another TLV can specify that information for all active processes is to be returned). The syntax of the information buffer is shown below:

```
<info_buffer>          ::= <class tlv><item tlv>...<termination_word>

<class tlv>            ::= <class tag><length>[<value>]

<item tlv>             ::= <item tag><length>

<class tag>            ::= ub2, see below

<item tag>             ::= ub2, see below

<length>               ::= ub2

<value>                ::= byte buffer of size "length"

<termination_word>     ::= KUSGTBEND
```

The class TLV is used to indicate the type of fields that will be requested in subsequent item TLVs in this information buffer. In general terms, the class TLV specifies the underlying data structure that contains the data to be retrieved and the entries for each item TLV indicate a field to be returned from the underlying table. No value is specified for an item TLV in the information buffer (that is, the length field is zero). If data from different classes are desired, then separate information buffers will have to be created and separate calls will have be made for each class. The documentation for each of the classes describe in more detail the format of the information buffer for that class.

Most classes have multiple rows of information. The class TLV for these classes have an optional value which is a key value that identifies a subset of rows (usually only one row) in the class. If this value is specified, then only the data for the specified row are returned. If no value is specified (the length field is 0), then all rows for the class are returned. The example below shows an information buffer requesting the username and Oracle Rdb process ID of all Rdb7 processes. It is followed by an example of an information buffer requesting information for the Oracle Rdb process with PID 20.

Note that TLV buffers usually contain data of different types, and therefore, the TLV maintenance functions should be used to create and read the buffers. In this case, all data is of the same type, so an array is created to contain the data.

```
/*
   information buffer to return Oracle Rdb process ID
   and user name for all Oracle Rdb processes attached to
   an Rdb database
 */

ub2 proc_info[] = {
RMUST_CT_PROCESS, 0,
RMUST_T_PID, 0,
RMUST_T_USERNAME, 0,
KUSGTBEND, 0};
```

```
/*
   information buffer to return user name for Rdb
   process with PID 20
 */

ub2 proc_20_info[] = (
RMUST_CT_PROCESS, 4, 20,
RMUST_T_USERNAME, 0,
KUSGTBEND, 0};
```

The valid classes and tags for Oracle Rdb databases are listed in the section titled Class and Item Tags for Oracle Rdb Databases on page 11.

1.1.6 Result Buffer

The result buffer is a buffer allocated by the caller but filled in by the API with the data items specified by the information buffer. If the information buffer requests multiple rows of the same data (for example, return information for all active processes using the database), then all data for the first process is stored followed by data for the second process, and so on until all data for all processes have been stored. Tags are not guaranteed to appear in the result buffer in the same order they were specified in the information buffer. Specific information about the format of the result buffer for each possible class is documented as part of the class description.
The syntax for the result buffer is shown below:

```
<result_buffer> ::= <TLV group>... <termination_word>
<TLV group> ::= KUSGTBOG 0 <tlv>... [[<buffer_full_tlv>] | KUSGTEOG 0]

<tlv> ::= <tag> <length> [<value>]

<tag> ::= two unsigned bytes

<length> ::= two unsigned bytes

<value> ::= byte buffer of size "length"
<buffer_full_tlv> ::= {KUSGTTRN 0 | KUSGTBCON 0}

<termination_word> ::= { KUSGTBEND 0 | <buffer_full_tlv>}
```

The KUSGTBOG and KUSGTEOG tags are used to mark the beginning and end of a group of related data. Another way of looking at KUSGTBOG and KUSGTEOG is to view them as delimiting rows of data. For example, if an information buffer requested the user name, terminal name and PID of every Oracle process, then the result buffer would contain KUSGTBOG and KUSGTEOG tags around the set of fields for each Oracle process. Note that the KUSGTBOG and KUSGTEOG are used even if only one group of related fields is returned. An example of an output buffer with two 'rows' of data is shown below.

```
KUSGTBOG, 0,                            /* start of 1st row */
RMUST_CT_USERNAME, 10, "RDB_USER_1",
RMUST_CT_IMAGENAME, 7, "SQL.EXE",
RMUST_CT_PID, 4, 20,
KUSGTEOG, 0,                            /* end of 1st row */
KUSGTBOG, 0,                            /* start of 2nd row */
RMUST_CT_USERNAME, 10, "RDB_USER_2",
RMUST_CT_IMAGENAME, 7, "SQL.EXE",
RMUST_CT_PID, 4, 12,
KUSGTEOG, 0,                            /* end of 2nd row */
KUSGTBEND, 0                            /* end of buffer */
```

If a buffer is too small for the amount of data to be returned, the end of the buffer will containing the KUSGTBCNT or KUSGTBTRN TLVs. KUSGTBCNT is used when the user specified a continuation context in the call to kusdb_get_info and the remaining data can be returned in a subsequent call (see the following section titled Continuation Context for more information.) If no continuation context was specified by the caller and the data does not fit into the buffer, then KUSGTBTRN TLV is returned. If all requested data fits in the buffer, then KUSGTBEND is returned. Note that KUSTVBTRN and KUSTVBCONT can occur in the middle of a group of fields.

1.1.7 Continuation Context

Because the calling program cannot predict how much data will be returned by the API, the API provides a mechanism that allows the caller to make multiple calls to return information for one request. To enable this mechanism, the caller must first allocate a continuation context variable (with the kuscx_allocate function). Then, the continuation context variable is passed as an argument to the function that retrieves data (the kusdb_get_info function). When that function returns, the caller can determine if the buffer was too small by looking for the "continuation tag", KUSGTBCNT at the end of the result buffer. If the buffer was continued, then the caller can execute the call again to retrieve statistics (the kusdb_ get_info function) being sure to specify the continuation context variable. The caller continues to call kusdb_get_info until a buffer is returned that is not continued. When this happens, the caller then frees the continuation context information by calling kuscx_free_handle or reinitializes the continuation context information for reuse by calling kuscx_init.

As an example, consider what might happen if the caller specified a 40 byte buffer when retrieving data for the example above and the caller used a continuation context variable. The first call to kusdb_get_info might return the following result buffer (note that contents of the buffers may vary due to different alignment rules for data alignment on different platforms and compilers):

```
KUSGTBOG, 0,                            /* total 4 bytes */
RMUST_T_USERNAME, 13, "ORARDB_USER_1",  /* total 21 bytes */
RMUST_T_IMAGENAME,  10, "MY_PRG.EXE",   /* total 35 bytes */
KUSGTBCON, 0                            /* total 39 bytes */
```

No more TLVs can be inserted after the KUST39_TERMINAL TLV because there is not enough space in the buffer. Instead, a KUSGTBCON TLV is written. Because a continuation context variable was passed in by the caller, the rest of the data can still be retrieved. If the caller passes a 40 byte buffer to kusdb_get_info again, it would look like the following:

```
RMUST_T_PID, 4, 20,                     /* total 8 bytes */
KUSGTEOG, 0,                            /* total 12 bytes */
KUSGTBOG, 0,                            /* total 16 bytes */
RMUST_T_USERNAME, 13, "ORARDB_USER_2",  /* total 33 bytes */
KUSGTBCON 0                             /* total 37 bytes */
```

Another call to kusdb_get_info with another 40 byte buffer would return the following:

```
RMUST_T_IMAGENAME,  11, "MY_PRG2.EXE",  /* total 15 bytes */
RMUST_T_PID, 4, 12,                     /* total 23 bytes */
KUSGTEOG, 0,                            /* total 27 bytes */
KUSGTBEND, 0                            /* total 31 bytes */
```

KUSGTBEND indicates that all requested data has been returned.

It is possible for a TLV entry to not fit inside of the empty TLV buffer passed in by the caller. If the caller had specified a buffer size of 10 in the above example, the TLV for the user name field (KUST32_USERNAME) would not fit even if no other TLVs had been inserted into the buffer. If a TLV entry will not fit into a buffer and the buffer is empty, the KUSDB_ GET_INFO call will return status KUSSTBFSML.

1.1.8 Error Tags

If an unsupported or invalid tag is encountered by the kusdb_get_info routine while reading the information buffer, then a special error tag will be returned in the result buffer. An error tag will also be returned if the value for a class tag does not identify any rows. A list of the special error tags and the circumstances under which they are returned is listed in Table 2: Error Tags.

Table 1–3 Error Tags

tar	returned when...
KUSGTBDTG	this tag is not a known tag. The value for KUSGTBDTG is the unknown tag from the information buffer. Note that if an invalid class tag is specified. an error status is returned and the result buffer is empty.
KUSGTUNST	the tag is valid for this class, but not for the version of the connected database. The value of this tag is the unknown tag from the information buffer
KUSGTNP	the tag is not supported on the requested platform.

1.1.9 General tags

The SGA API defines a number of tags which are used in result buffers. These tags are described in Table 3: General Tags. The use of these tags has been discussed in previous sections.

Table 1–4 General Tags

tag code	meaning
KUSGTBEND	Indicates the end of both the request and result buffer.
KUSGTBTRN	Indicates that the result buffer was not large enough to contain the result. All information in the result buffer is valid, but one or more requested items is missing.

Table 1–4 (Cont.) General Tags

tag code	meaning
KUSGTBOG	Indicates the beginning of a set of related data items. Used with KUSGTEOG to delimit rows of data for classes that return multiple rows.
KUSGTEOG	Indicates the end of a set of related data items. KUSGTBOG is used to indicate the beginning of the set.
KUSGTBCON	Indicates that the result buffer contains a partial result. This tag can be returned only when a continuation context handle was specified in the kusdb_get_info call.

1.1.10 Compiling and Linking

For Oracle Rdb databases, a version varianted shareable image is generated for Version 6.0 through Version 7.1. The SGA API calls and API specific types are described in kusapi.h while the more generic types are described in oratypes.h.

1.2 Class and Item Tags for Oracle Rdb Databases

The valid class tags for Oracle Rdb classes and a description of the class are listed in Table 4: Classes for Oracle Rdb Databases. The table is followed by a description of the item tags that are valid for each class name.

Table 1–5 Classes for Oracle Rdb Databases

Class Name	Description
RMUST_CT_AIJINFO	General information on the state of the after-image journal subsystem
RMUST_CT_AIJJNL	Information on individual after image journals
RMUST_CT_DBINFO	General database information such as number of buffers, max number of users, database flags, etc.
RMUST_CT_DBR	Information on database recovery processes
RMUST_CT_FILE	Storage area file information; i.e. names, number of pages, blocks per page, etc.
RMUST_CT_FILE_NODUP	Storage area info, duplicate data removed for subsequent requests for info.
RMUST_CT_PROCESS	Information on database user processes
RMU_CT_STALL_ACT RMUST_CT_STALL	Stall message information
RMUST_CT_ARPMS_STATS	Storage area statistics
RMUST_CT_RTPMS_STATS	Run time statistics for users.

1.2.1 Time and Date Tags (Oracle Rdb only)

There are four suffixes for certain time and date item tags for Oracle Rdb. The suffix indicates the format to be returned for the date and time field. The suffixes and their descriptions are listed in Table 5: Date and Time Return Value Types.

Table 1–6 Date and Time Return Value Types

Suffix	Meaning
ADT	Operating system specific binary date and time format.
FDT	Formatted Date and Time. The time and date are formatted into an ASCII result buffer in YYYYNNDDHHMMSSGG format as described in Table 6.
RT	Relative Time. The time is specified in 1/100 second increments specified in an unsigned longword. The relative time is used as both an information and result buffer tag.
TDT	Text Date and Time. Returns the date and time in an operating system specific form which can be used for display.
TT	Text Time. Returns the time component of the time and date stamp in an operating system specific form which can be used for display.

Descriptions of the date and time field components used in Table 5 are listed in Table 6.

Table 1–7 Formatted Date and Time Fields

String	Meaning
YYYY	Four digits of the year, between 1857 and 9999
NN	Two digits of month, including leading zero for months between January through September. Valid range is 01 - 12
DD	Two digits for day of month, 01 through 31, right-justified and zero-filled
HH	Two digits for hour of day, 00 through 23, right-justified and zero-filled
MM	Two digits for minute of hour, 00 through 59, right-justified and zero-filled
SS	Two digits for second of minute, 00 through 59, right-justified and zero-filled
GG	Two digits for hundredths of second, 00 through 99, right-justified and zero-filled

1.3 RMUST_CT_AIJINFO - AIJ Information

The AIJ information class provides information on the after-image journaling subsystem.

1.3.1 Information Buffer Syntax

```
<aijinfo_info_buffer> ::= RMUST_CT_AIJINFO 0 <aijinfo_tlv>... KUSGTBEND
<aijinfo_tlv> ::= {
  RMUST_T_AIJ_AUTO_SYNC 0
| RMUST_T_AIJ_CUR_MSN 0
| RMUST_T_AIJ_DATA_SYNC 0
| RMUST_T_AIJ_ENABLED_FLG 0
| RMUST_T_AIJ_LEOF 0
| RMUST_T_AIJ_PEOF 0
| RMUST_T_AIJ_STALLED_MSN 0
| RMUST_T_AIJ_STATUS 0
| RMUST_T_AIJ_VNO 0
| RMUST_T_AUTO_BKUP_FLG 0
| RMUST_T_CACHE_FILENAME 0
| RMUST_T_CACHE_STATUS 0
| RMUST_T_CUR_ACTIVE_AIJ 0
| RMUST_T_CUR_BKUP_VNO 0
| RMUST_T_CUR_RCVR_VNO 0
| RMUST_T_DATA_CMIT_FLG 0
| RMUST_T_DB_BKUP_VNO 0
| RMUST_T_DEF_ALLOCATION 0
| RMUST_T_DEF_BKUP_FILENAME 0
| RMUST_T_DEF_EXTENSION 0
| RMUST_T_GOVERNOR_ENABLED 0
| RMUST_T_HARD_DATA_LOSS_FLG 0
| RMUST_T_JOURNAL_CNT 0
| RMUST_T_LCS_ACTIVE 0
| RMUST_T_LRS_ACTIVE 0
| RMUST_T_LRS_STATE 0
| RMUST_T_LSS_ACTIVE 0
| RMUST_T_LSS_AIJ_VBN 0
| RMUST_T_LSS_AIJ_VNO 0
| RMUST_T_LSS_LAG_FDT 0
| RMUST_T_LSS_REF_COUNT 0
| RMUST_T_LSS_STATE 0
| RMUST_T_NEW_VERSION_FLG 0
| RMUST_T_OPER_CLASS 0
| RMUST_T_OVERWRITE_FLG 0
| RMUST_T_OVERWRITTEN_FLG 0
| RMUST_T_SHUTDOWN_TIME 0
| RMUST_T_STBY_RT_FILNAM 0 }
```

1.3.2 Result Buffer Syntax

```
<aij_result_buffer> ::= KUSGTBOG <aij_result_tlv> KUSGTEOG <end_word>
<aij_result_tlv> ::= {
  RMUST_T_AIJ_AUTO_SYNC 4 <syncMode>
| RMUST_T_AIJ_CUR_MSN 4 <count value>
| RMUST_T_AIJ_DATA_SYNC 4 <syncMode>
| RMUST_T_AIJ_ENABLED_FLG 4 <boolean>
| RMUST_T_AIJ_LEOF 4 <count value>
| RMUST_T_AIJ_PEOF 4 <count value>
| RMUST_T_AIJ_STALLED_MSN 4 <count value>
| RMUST_T_AIJ_STATUS 4 <aij_status_value>
| RMUST_T_AIJ_VNO 4 <count_value>
| RMUST_T_AUTO_BKUP_FLG 4 <boolean>
| RMUST_T_CACHE_FILENAME <length> <text>
| RMUST_T_CACHE_STATUS 4 <cacheStatus>
| RMUST_T_CUR_ACTIVE_AIJ 4 <count_value>
| RMUST_T_CUR_BKUP_VNO 4 <count_value>
| RMUST_T_CUR_RCVR_VNO 4 <count_value>
| RMUST_T_DATA_CMIT_FLG 4 <boolean>
| RMUST_T_DB_BKUP_VNO 4 <count_value>
| RMUST_T_DEF_ALLOCATION 4 <count_value>
| RMUST_T_DEF_BKUP_FILENAME <length> <text>
| RMUST_T_DEF_EXTENSION 4 <count_value>
| RMUST_T_GOVERNOR_ENABLED 4 <boolean>
| RMUST_T_HARD_DATA_LOSS_FLG 4 <boolean>
| RMUST_T_JOURNAL_CNT 4 <count_value>
```

```
| RMUST_T_LCS_ACTIVE 4 <boolean>
| RMUST_T_LRS_ACTIVE 4 <boolean>
| RMUST_T_LRS_STATE 4 <server_state>
| RMUST_T_LSS_ACTIVE 4 <boolean>
| RMUST_T_LSS_AIJ_VBN 4 <count value>
| RMUST_T_LSS_AIJ_VNO 4 <count value>
| RMUST_T_LSS_LAG_FDT <length> <text>
| RMUST_T_LSS_REF_COUNT 4 <count value>
| RMUST_T_LSS_STATE 4 <server_state>
| RMUST_T_NEW_VERSION_FLG 4 <boolean>
| RMUST_T_OPER_CLASS 4 <count_value>
| RMUST_T_OVERWRITE_FLG 4 <boolean>
| RMUST_T_OVERWRITTEN_FLG 4 <boolean>
| RMUST_T_SHUTDOWN_TIME 4 <count_value>
| RMUST_T_STBY_RT_FILNAM <length> <text> }

<aij_status_value> ::= { RMUST_K_AIJ_NORMAL | RMUST_K_AIJ_CORRUPT |
                         RMUST_K_UNKNOWN}
<sync_mode> ::= {
      RMUST_K_SYNC_COLD
    | RMUST_K_SYNC_COMMIT
    | RMUST_K_SYNC_HOT
    | RMUST_K_SYNC_WARM
    | RMUST_K_UNKNOWN }

<server_state> ::= {
      RMUST_K_LRS_COMPLETION
    | RMUST_K_LRS_RESTART
    | RMUST_K_LRS_RESUMPTION
    | RMUST_K_LRS_SUSPENDED
    | RMUST_K_LRS_SYNC_REDO
    | RMUST_K_LSS_ACTIVATE
    | RMUST_K_LSS_CONNECT
    | RMUST_K_LSS_DB_BIND
    | RMUST_K_LSS_DB_UNBIND
    | RMUST_K_LSS_DB_SYNC
    | RMUST_K_LSS_INACTIVE
    | RMUST_K_LSS_NET_BIND
    | RMUST_K_LSS_NET_UNBIND
    | RMUST_K_LSS_RECOVERY
    | RMUST_K_LSS_REPLICATE
    | RMUST_K_LSS_SHUTDOWN }
```

1.3.3 Tag Descriptions

Descriptions of the item tags for class RMUST_CT_AIJINFO are shown in Table 7.

Table 1–8 RMUST_CT_AIJINFO Tags

Tar	Meaning
RMUST_T_AIJ_ENABLED_ FLG	If TRUE AIJ is enabled, FALSE otherwise.
RMUST_T_AIJ_AUTO_ SYNC	Hot standby - database replication sync mode.
RMUST_T_AIJ_CUR_MSN	Hot standby - current message sequence number.
RMUST_T_AIJ_DATA_SYNC	Hot standby - database replication sync mode.
RMUST_T_AIJ_LEOF	Current AIJ logical EOF.
RMUST_T_AIJ_PEOF	Current AIJ physical EOF.
RMUST_T_AIJ_SRV_NAME	Hot Standby: name if replication server connection

Table 1–8 (Cont.) RMUST_CT_AIJINFO Tags

Tar	Meaning
RMUST_T_AIJ_STALLED_ MSN	Hot standby - stalled message sequence number.
RMUST_T_AIJ_STATUS	If not RMUST_T_AIJDB_NORMAL, then some process failed writing to the AIJ file; AIJ DATA LOSS HAS OCCURRED. Accordingly, ALL active processes will detach from the database, and DBR will not attempt to access the AIJ file. Also new BINDs will be prevented. This field can be cleared by disabling or modifying the AIJ file.
RMUST_T_AIJ_VNO	Current AIJ sequence number.
RMUST_T_AUTO_BKUP_ FLG	If TRUE backup is automatic, FALSE otherwise.
RMUST_T_CACHE_ FILENAME	This field contains the electronic "non-volatile" after-image journal cache (ACE) filename (if any) associated with the database.
RMUST_T_CACHE_STATUS	If not RMUST_K_ACE_NORMAL, some process has failed writing to the ACE. AIJ data loss has NOT occurred.
RMUST_T_CUR_ACTIVE_ AIJ	current active AIJ journal index.
RMUST_T_CUR_BKUP_ VNO	AIJ current backup version number.
RMUST_T_CUR_RCVR_ VNO	AIJ current recovery version number.
RMUST_T_DATA_CMIT_FLG	If TRUE DB changes have been made when AIJ was disabled, FALSE otherwise.
RMUST_T_DB_BKUP_VNO	Full Database backup version number.
RMUST_T_DEF_ ALLOCATION	default AIJ allocation in blocks.
RMUST_T_DEF_BKUP_ FILENAME	This field contains the default after-image journal backup filename (if any) associated with the database.
RMUST_T_DEF_ EXTENSION	default AIJ extension in blocks.
RMUST_T_GOVERNOR_ ENABLED	Hot standby - TRUE if replication governor enabled.
RMUST_T_HARD_DATA_ LOSS_FLG	If TRUE hard data loss resulted from a failover, FALSE otherwise.
RMUST_T_JOURNAL_CNT	number of allocated AIJ File Blocks.
RMUST_T_LCS_ACTIVE	Hot standby - TRUE if AIJ log catch-up server started.
RMUST_T_LRS_ACTIVE	Hot standby - TRUE if AIJ log roll-forward server started.
RMUST_T_LRS_STATE	Hot standby - LRS state. Refer to Table 9 for valid return values.
RMUST_T_LSS_ACTIVE	Hot standby - TRUE if AIJ log ship server started.
RMUST_T_LSS_AIJ_VBN	Hot standby - Master/standby AIJ VBN.
RMUST_T_LSS_AIJ_VNO	Hot standby - Master/standby AIJ VNO.
RMUST_T_LSS_LAG_FDT	Hot standby - Master/standby lag time.
RMUST_T_LSS_REF_ COUNT	Hot standby - Master/standby # of LSS participants.

Table 1–8 (Cont.) RMUST_CT_AIJINFO Tags

Tar	Meaning
RMUST_T_LSS_STATE	Hot standby - LSS state. Refer to Table 9 for valid return values.
RMUST_T_NEW_VERSION_ FLG	If TRUE backup creates a new journal version, FALSE otherwise.
RMUST_T_OPER_CLASS	OpenVMS operator notification classes. Refer to Table 8: AIJ Operator Notification Tags for valid values
RMUST_T_OVERWRITE_ FLG	If TRUE journals are overwritten on wraparound, FALSE otherwise.
RMUST_T_OVERWRITTEN_ FLG	If TRUE one or more journals have been overwritten, FALSE otherwise.
RMUST_T_SHUTDOWN_ TIME	Time (minutes) to database shutdown after the AIJ subsystem becomes stalled. i.e. all journals consumed.
RMUST_T_STBY_RT FILNAM	Live or standby database filename.

The valid values for tag RMUST_T_OPER_CLASS are shown in Table 8.

Table 1–9 AIJ Operator Notification Tags

Tag	Value	Meaning
RMUST_K_AIJ_ OPR_DISABLED	0	No operator class activated.
RMUST_K_AIJ_ OPR_CENTRAL	1	Central operator.
RMUST_K_AIJ_ OPR_DISKS	8	Disk operator.
RMUST_K_AIJ_ OPR_CLUSTER	128	Cluster operator.
RMUST_K_AIJ_ OPR_SECURITY	256	Security operator.
RMUST_K_AIJ_ OPR_OPER1	4096	User operator #1.
RMUST_K_AIJ_ OPR_OPER2	8192	User operator #2.
RMUST_K_AIJ_ OPR_OPER3	16384	User operator #3.
RMUST_K_AIJ_ OPR_OPER4	32768	User operator #4.
RMUST_K_AIJ_ OPR_OPER5	65536	User operator #5.
RMUST_K_AIJ_ OPR_OPER6	131072	User operator #6.
RMUST_K_AIJ_ OPR_OPER7	262144	User operator #7.

Table 1–9 (Cont.) AIJ Operator Notification Tags

Tag	Value	Meaning
RMUST_K_AIJ_OPR_OPER8	524288	User operator #8.
RMUST_K_AIJ_OPR_OPER9	1048576	User operator #9.
RMUST_K_AIJ_OPR_OPER10	2097152	User operator #10.
RMUST_K_AIJ_OPR_OPER11	4194304	User operator #11.
RMUST_K_AIJ_OPR_OPER12	8388608	User operator #12.

The valid values for tag RMUST_T_LRS_STATUS, RMUST_T_LSS_STATUS are shown in Table 9.

Table 1–10 Status values for hot standby log servers

Tag	Meaning
RMUST_K_LRS_COMPLETION	Replication completion.
RMUST_K_LRS_RESTART	Replication restart activity.
RMUST_K_LRS_RESUMPTION	Replication resumed (catch-up).
RMUST_K_LRS_SUSPENDED	Replication suspended.
RMUST_K_LRS_SYNC_REDO	LRS synchronization redo completion.
RMUST_K_LSS_ACTIVATE	LSS server activation.
RMUST_K_LSS_CONNECT	Waiting for LCS to connect.
RMUST_K_LSS_DB_BIND	Binding to database.
RMUST_K_LSS_DB_UNBIND	Unbinding from database.
RMUST_K_LSS_DB_SYNC	Database synchronization.
RMUST_K_LSS_INACTIVE	Inactive (server not running).
RMUST_K_LSS_NET_BIND	Binding to network.
RMUST_K_LSS_NET_UNBIND	Unbinding from network.
RMUST_K_LSS_RECOVERY	Recovering from failure.
RMUST_K_LSS_REPLICATE	Database replication.
RMUST_K_LSS_SHUTDOWN	Replication cleanup.

1.3.4 Version specific tags

Table 1–11 Lowest version of Oracle Rdb required for tag support

Tag	Version of Oracle Rdb required
RMUST_T_AIJ_AUTO_ SYNC	7.0
RMUST_T_AIJ_DATA_SYNC	7.0
RMUST_T_AIJ_CUR_MSN	7.0
RMUST_T_AIJ_DATA_SYNC	7.0
RMUST_T_AIJ_SRV_NAME	7.0
RMUST_T_AIJ_STALLED_ MSN	7.0
RMUST_T_GOVERNOR_ ENABLED	7.0
RMUST_T_LCS_ACTIVE	7.0
RMUST_T_LRS_ACTIVE	7.0
RMUST_T_LRS_STATE	7.0
RMUST_T_LSS_ACTIVE	7.0
RMUST_T_LSS_AIJ_VBN	7.0
RMUST_T_LSS_AIJ_VNO	7.0
RMUST_T_LSS_LAG_FDT	7.0
RMUST_T_LSS_REF_ COUNT	7.0
RMUST_T_LSS_STATE	7.0
RMUST_T_STBY_RT_ FILNAM	7.0
RMUST_T_AIJ_VNO	6.1
RMUST_T_CACHE_STATUS	6.1

1.4 RMUST_CT_AIJJNLS - AIJ Journal Class

The AIJ journal class provides information about after-image journal instances. This class has multiple rows. All rows will be returned if a length of 0 is specified for the TLV entry containing the class tag. A single row can be retrieved by specifying the AIJFB index for the AIJ journal file of interest as the value of the TLV entry for the class tag.

1.4.1 Information Buffer Syntax

```
<aijjnls_info_buffer> ::= RMUST_CT_AIJJNLS{0 | {4, <aijfb_index_number>}}
              <aij_jnls_tlv>... KUSGTBEND
<aij_file_tlv> ::= {
  RMUST_T_ACTIVE_FLG 0
| RMUST_T_AIJFB_INDEX 0
| RMUST_T_ALLOCATION 0
| RMUST_T_BACKUP_DATE_ADT 0
| RMUST_T_BACKUP_DATE_FDT 0
| RMUST_T_BACKUP_DATE_TDT 0
| RMUST_T_BKUP_ACTIVE_FLG 0
| RMUST_T_BKUP_VNO 0
| RMUST_T_CUR_AIJ_FILENAME 0
| RMUST_T_CUR_BKUP_FILENAME 0
| RMUST_T_DEF_AIJ_FILENAME 0
| RMUST_T_EXTENSION 0
| RMUST_T_HARD_DATA_LOSS_FLG 0
| RMUST_T_INIT_ACTIVE_FLG 0
| RMUST_T_JOURNAL_NAME 0
| RMUST_T_JOURNAL_STATUS 0
| RMUST_T_LAST_BKUP_VNO 0
| RMUST_T_MODIFIED_FLG 0
| RMUST_T_NEW_VERSION_FLG 0
| RMUST_T_OVERWRITTEN_FLG 0
| RMUST_T_RESTORED_FLG 0
| RMUST_T_SOFT_DATA_LOSS_FLG 0 }
```

Result Buffer Syntax

```
<aijjnls_result_buffer> ::= {KUSGTBOG <aijjnls_result_tlv>...
           KUSGTEOG } ... <end_word>
<aijjnls_result_tlv> ::= {
  RMUST_T_ACTIVE_FLG 4 <boolean>
    | RMUST_T_AIJFB_INDEX 4 <AIJFB index number, 0 based>
| RMUST_T_ALLOCATION 4 <count_value>
| RMUST_T_BACKUP_DATE_ADT 8 <absolute_date>
| RMUST_T_BACKUP_DATE_FDT <length> <text>
| RMUST_T_BACKUP_DATE_TDT <length> <text>
| RMUST_T_BKUP_ACTIVE_FLG 4 <boolean>
| RMUST_T_BKUP_VNO 4 <count_value>
| RMUST_T_CUR_AIJ_FILENAME <length> <text>
| RMUST_T_CUR_BKUP_FILENAME <length> <text>
| RMUST_T_DEF_AIJ_FILENAME <length> <text>
| RMUST_T_EXTENSION 4 <count_value>
| RMUST_T_HARD_DATA_LOSS_FLG 4 <boolean>
| RMUST_T_INIT_ACTIVE_FLG 4 <boolean>
| RMUST_T_JOURNAL_NAME <length> <text>
| RMUST_T_JOURNAL_STATUS 4 <aij_file_status>
| RMUST_T_LAST_BKUP_VNO 4 <count_value>
| RMUST_T_MODIFIED_FLG 4 <boolean>
| RMUST_T_NEW_VERSION_FLG 4 <boolean>
| RMUST_T_OVERWRITTEN_FLG 4 <boolean>
| RMUST_T_RESTORED_FLG 4 <boolean>
| RMUST_T_SOFT_DATA_LOSS_FLG 4 <boolean>
}

<aij_file_status> ::= {
      RMUST_K_JNL_NORMAL          | RMUST_K_JNL_FILACCERR
    | RMUST_K_JNL_CORRUPT         | RMUST_K_JNL_EMPTY
    | RMUST_K_JNL_OPCODE          | RMUST_K_JNL_FULL
    | RMUST_K_JNL_DISABLED        | RMUST_K_JNL_NOT_EMPTY
    | RMUST_K_JNL_DBR             | RMUST_K_UNKNOWN }
```

1.4.2 Tag Descriptions

Descriptions of the valid item tags for the RMUST_CT_AIJJNL class are shown in Table 11: AIJ File Information Tags.

Table 1–12 AIJ File Information Tags

Tag	Meaning
RMUST_T_ACTIVE_FLG	AIJ file block has been assigned.
RMUST_T_AIJFB_INDEX	Unique 0-based integer index to identify each journal file
RMUST_T_ALLOCATION	AIJ journal allocation in blocks.
RMUST_T_BACKUP_DATE_ ADT	TAD of last AIJ backup. (ADT Format)
RMUST_T_BACKUP_DATE_ FDT	TAD of last AIJ backup. (FDT Format)
RMUST_T_BACKUP_TIME_ TDT	TAD of last AIJ backup. (TDT Format)
RMUST_T_BKUP_DATE_ FLG	AIJ Backup in-progress.
RMUST_T_BKUP_VNO	AIJ current backup version number.
RMUST_T_CUR_AIJ_ FILENAME	Name of after-image journal file for database.
RMUST_T_CUR_BKUP_ FILENAME	This field contains the resultant filename of the after-image journal backup file (if any) associated with this journal entry.
RMUST_T_DEF_AIJ_ FILENAME	This field contains the default after-image journal filename associated with this journal entry.
RMUST_T_EXTENSION AIJ	journal extension in blocks.
RMUST_T_HARD_DATA_ LOSS_FLG	Hard data loss resulted from failover.
RMUST_T_INIT_ACTIVE_ FLG	AIJ Initialization in-progress.
RMUST_T_JOURNAL_ NAME	This field contains the symbolic name of the after-image journal file (if any) associated with the database.
RMUST_T_JOURNAL_ STATUS	Refer to Table 12, AIJ File State Tags for valid values
RMUST_T_LAST_BKUP_ VNO	Last version number backed up.
RMUST_T_MODIFIED_FLG	AIJ file needs to be backed up.
RMUST_T_NEW_VERSION_ FLG	Backup creates new journal version.
RMUST_T_OVERWRITTEN_ FLG	Journal has been overwritten.
RMUST_T_RESTORED_FLG AIJFB	restored from existing file.
RMUST_T_SOFT_DATA_ LOSS_FLG	Soft data loss resulted from failover.

Table 1–13 AIJ File State Tags

Tag	Meaning
RMUST_K_JNL_CORRUPT	Journal file is corrupt.
RMUST_K_JNL_DBRDBR	invoked during suspended switchover.
RMUST_K_JNL_DISABLED	Journal entry has been disabled.
RMUST_K_JNL_EMPTY	Journal file is empty.
RMUST_K_JNL_ FILACCERR	Journal has been possibly deleted or moved.
RMUST_K_JNL_FULL	Journal file is full.
RMUST_K_JNL_NORMAL	Journal status is normal.
RMUST_K_JNL_NOT_ EMPTY	Journal file contains transaction data.
RMUST_K_JNL_OPCODE	Journal contains unknown opcode.

1.5 RMUST_CT_ARPMS_STATS - Area Performance Statistics

The RMUST_CT_ARPMS_STATS class is used to return run-time statistics maintained by Oracle Rdb for databases on a particular node. This request can be used to return the statistics buffer maintained for each storage area.

1.5.1 Buffer Syntax

```
arpms_stats_request ::=     RMUST_CT_ARPMS {0 | {4, <dbid>}}
                    <arpms_rqst_tlv> KUSGTBEND
<arpms_rqst_tlv> ::= {
          RMUST_T_ARPMS 0
        | RMUST_T_DBID 0 }
```

1.5.2 Result Buffer Syntax

```
arpms_stats_response ::= <arpms_result> ... <end_word>
<end_word> ::= { KUSGTBEND | KUSGTBTRN | KUSGTBCON }
<arpms_result> ::=    {KUSGTBOG <arpms_tlv>... KUSGTEOG}...
           <end_word>

<arpms_tlv> ::=       { RMUST_T_DBID 4 <dbid>
                      | RMUST_T_ARPMS <len> <area_stats> }

<len> ::= 16 longwords

<area_stats> ::= Pointer to an FIOVEC_t structure for the area.
```

1.5.3 Tag Descriptions

Table 1–14 RMUST_CT_STATS Tags

Tag Code	Meaning
RMUST_T_ARPMS	Storage area performance measurement statistics returned in FIOVEC_t structure. Look in RMUST_RTPMS.H for a declaration of the structure.
RMUST_T_DBID	Storage area ID

1.6 RMUST_CT_DBINFO - Database Information

The database information class provides access to data stored in the KODA root (KROOT) structure.

1.6.1 Information Buffer Syntax

```
<dbinfo_info_buffer> ::= RMUST_CT_DBINFO 0 <dbinfo_rqst_tlv>... KUSGTBEND
<dbinfo_rqst_tlv> ::= {
  RMUST_T_AIJ_COMMIT_FLG 0
| RMUST_T_AIJ_SHUTDOWN 0
| RMUST_T_ALARM_ENABLED_FLG 0
| RMUST_T_ALS_ACTIVE 0
| RMUST_T_ALS_MODE_FLG 0
| RMUST_T_AUDIT_ENABLED_FLG 0
| RMUST_T_AUDIT_FIRST_FLG 0
| RMUST_T_AUDIT_FLUSH_FLG 0
| RMUST_T_BACKUP_DATE_ADT 0
| RMUST_T_BACKUP_DATE_FDT 0
| RMUST_T_BACKUP_DATE_TDT 0
| RMUST_T_BUF_BLOCK_CNT 0
| RMUST_T_CKPT_BLOCKS 0
| RMUST_T_CKPT_TIME 0
| RMUST_T_CREATE_DATE_ADT 0
| RMUST_T_CREATE_DATE_FDT 0
| RMUST_T_CREATE_DATE_TDT 0
| RMUST_T_DBR_BUF_CNT 0
| RMUST_T_DEF_BUF_CNT 0
| RMUST_T_DEFERRED_SNAP_FLG 0
| RMUST_T_DELTA_HSEC 0
| RMUST_T_FAST_COMMIT_FLG 0
| RMUST_T_FILID_CNT 0
| RMUST_T_FULL_BACKUP_TSN 0
| RMUST_T_GB_CNT 0
| RMUST_T_GB_ENABLED_FLG 0
| RMUST_T_LCKOPT_ENABLED_FLG 0
| RMUST_T_LOCK_TIMEOUT 0
| RMUST_T_MAX_GB_PER_USER 0
| RMUST_T_MAX_NODE_CNT 0
| RMUST_T_MAX_USER_CNT 0
| RMUST_T_OPEN_MODE_FLG 0
| RMUST_T_PLATFORM 0
| RMUST_T_SFDB_FLG 0
| RMUST_T_STATS_ENABLED_FLG 0
| RMUST_T_TSN_INTERVAL 0 }
```

1.6.2 Result Buffer Syntax

```
<dbinfo_result_buffer> ::= KUSGTBOG <dbinfo_tlv>... KUSGTEOG <end_word>
<dbinfo_tlv> ::= {
 RMUST_T_AIJ_COMMIT_FLG 4 <boolean>
| RMUST_T_AIJ_SHUTDOWN 4 <count value>
| RMUST_T_ALARM_ENABLED_FLG 4 <boolean>
| RMUST_T_ALS_ACTIVE 4 <boolean>
| RMUST_T_ALS_MODE_FLG 4 <boolean>
| RMUST_T_AUDIT_ENABLED_FLG 4 <boolean>
| RMUST_T_AUDIT_FIRST_FLG 4 <boolean>
| RMUST_T_AUDIT_FLUSH_FLG 4 <boolean>
| RMUST_T_BACKUP_DATE_ADT 8 <absolute_date>
| RMUST_T_BACKUP_DATE_FDT <length> <text)
| RMUST_T_BACKUP_DATE_TDT <length> <text>
| RMUST_T_BUF_BLOCK_CNT <count_value>
| RMUST_T_CKPT_BLOCKS 4 <count_value>
| RMUST_T_CKPT_TIME 4 <count_value>
| RMUST_T_CREATE_DATE_ADT 8 <absolute_date>
| RMUST_T_CREATE_DATE_FDT <length> <text>
| RMUST_T_CREATE_DATE_TDT <length> <text>
| RMUST_T_DBR_BUF_CNT 4 <count_value>
| RMUST_T_DEF_BUF_CNT 4 <count_value>
| RMUST_T_DEFERRED_SNAP_FLG 4 <boolean>
| RMUST_T_DELTA_HSEC 4 <count_value>
| RMUST_T_FAST_COMMIT_FLG 4 <boolean>
| RMUST_T_FILID_CNT 4 <count_value>
| RMUST_T_FULL_BACKUP_TSN 4 <count_value>
| RMUST_T_GB_CNT 4 <count_value>
| RMUST_T_GB_ENABLED_FLG 4 <boolean>
| RMUST_T_LCKOPT_ENABLED_FLG 4 <boolean>
| RMUST_T_LOCK_TIMEOUT 4 <count_value>
| RMUST_T_MAX_GB_PER_USER 4 <count_value>
| RMUST_T_MAX_NODE_CNT 4 <count_value>
| RMUST_T_MAX_USER_CNT 4 <count_value>
| RMUST_T_OPEN_MODE_FLG 4 <boolean>
| RMUST_T_PLATFORM 4 <platform_value>
| RMUST_T_SFDB_FLG 4 <boolean>
| RMUST_T_STATS_ENABLED_FLG 4 <boolean>
| RMUST_T_TSN_INTERVAL 4 <count_value> }
```

1.6.3 Tag Descriptions

Descriptions of the valid item tags for the RMUST_CT_DBINFO class are shown below.

Table 1–15 RMUST_CT_DBINFO Tags

Tag	Meaning
RMUST_T_AIJ_COMMIT_ FLG	If TRUE then Commit to Journal is enabled for the database. FALSE otherwise.
RMUST_T_AIJ_SHUTDOWN	Number of minutes until AIJ shutdown.
RMUST_T_ALARM_ ENABLED	If TRUE then security alarms are enabled for the database. FALSE otherwise.
RMUST_T_ALS_ ACTIVETRUE	if AIJ log server started.
RMUST_T_ALS_MODE_FLG	0 = ALS mode is AUTOMATIC. 1 = ALS mode is MANUAL.
RMUST_T_AUDIT_ ENABLED_FLG	If TRUE then security auditing is enabled for the database. FALSE otherwise.

Table 1–15 (Cont.) RMUST_CT_DBINFO Tags

Tag	Meaning
RMUST_T_AUDIT_FIRST_ FLG	If TRUE then audits and/or alarms are generated only for the first access to a security object. If FALSE, all accesses to security objects generate audits or alarms.
RMUST_T_AUDIT_FLUSH_ FLG	If TRUE then security audits are flushed immediately to the security audit log file.
RMUST_T_BACKUP_DATE_ ADT	TAD of last completed full backup.
RMUST_T_BACKUP_DATE_ FDT	Formatted Date/Time of last completed full backup.
RMUST_T_BACKUP_DATE_ TDT	Text Date/Time of last completed full backup.
RMUST_T_BUF_BLK_CNT	Length in blocks of each page buffer.
RMUST_T_CKPT_BLOCKS	This field contains the number of blocks the AIJ file must increase in size before a user is forced to perform another checkpoint.
RMUST_T_CKPT_TIME	This field contains the number of seconds which pass before a user is forced to perform another checkpoint.
RMUST_T_CREATE_DATE_ ADT	Time and date the database was created.
RMUST_T_CREATE_DATE_ FDT	Formatted time and date the database was created.
RMUST_T_CREATE_DATE_ TDT	Text time and date the database was created.
RMUST_T_DBR_BUF_CNT	This field contains the default number of page buffers to be allocated to each DBR process.
RMUST_T_DEF_BUF_CNT	This field contains the default number of page buffers to be allocated to each bound user. This can be overridden by a logical name bind parameter.
RMUST_T_DEFERRED_ SNAP_FLG	If TRUE, then snapshots will only be written when necessary (a snapshot transaction is in progress). If FALSE, snapshots are always written.
RMUST_T_DELTA_HSEC	Hundredths of seconds since start of statistics collection.
RMUST_T_FAST_COMMIT	_FLGIf TRUE then Fast Commit is enabled for the database. FALSE otherwise.
RMUST_T_FILID_CNT	Number of storage area slots allocated.
RMUST_T_FULL_BACKUP_ TSN	TSN of last completed full backup.
RMUST_T_GB_CNT	number of global buffers per node.
RMUST_T_GB_ENABLED_ FLG	If TRUE then global buffers are enabled. If FALSE global buffers are disabled.
RMUST_T_LCKOPT_ ENABLED	If TRUE then carry over lock optimization is enabled. FALSE otherwise.
RMUST_T_LOCK_TIMEOUT	lock timeout interval in seconds.
RMUST_T_MAX_GB_PER_ USER	max # of allocate set blocks per user.
RMUST_T_MAX_NODE_ CNT	This field contains the maximum number of nodes that can be bound to the database at one time.

Table 1–15 (Cont.) RMUST_CT_DBINFO Tags

Tag	Meaning
RMUST_T_MAX_USER_ CNT	This field contains the maximum number of users that can be bound to the database (cluster wide) at one time.
RMUST_T_OPEN_MODE_ FLG	0 = Database Open mode is AUTOMATIC. 1 = Open mode is MANUAL.
RMUST_T_PLATFORM	Current platform - RMUST_K_VMS or RMUST_K_OSF
RMUST_T_SFDB	If TRUE, this is an ORACLE RDB single-file database. If FALSE, this is an ORACLE RDB multi-file database.
RMUST_T_STATS_ ENABLED_FLG	If TRUE then statistics are maintained for the database. FALSE otherwise.
RMUST_T_TSN_INTERVAL	AIJ commit optimization TSN interval.

1.7 RMUST_CT_DBR - Database Recovery Process Information

This class returns information regarding active database recovery processes. This class can return multiple rows. If information for all recovery processes is desired, then the TLV for the class tag should contain no value. If information for only one recovery process is desired, then the TLV for the class tag should contain the DBR index for the recovery process.

1.7.1 Information Buffer Syntax

```
<dbr_info_buffer> ::= RMUST_CT_DBR{0 | {4, <dbr index>}} <dbr_rqst_tlv>...
           KUSGTBEND

<dbr_rqst_tlv> ::= {
  RMUST_T_ASCII_STALL 0
| RMUST_T_DBR_ACTIVITY 0
| RMUST_T_DBR_INDEX 0
| RMUST_T_LKID 0
| RMUST_T_PID 0
| RMUST_T_VBN 0 }
```

1.7.2 Result Buffer Syntax

```
<dbr_result_buffer> ::= {KUSGTBOG <dbr_tlv> ... KUSGTEOG}...
<end_word>
<dbr_tlv> ::= {
  RMUST_T_ASCII_STALL <length> <text>
| RMUST_T_DBR_ACTIVITY 4 <dbr_activity_code>
| RMUST_T_DBR_INDEX 4 <rtupb index>
| RMUST_T_LKID 4 <lkid>
| RMUST_T_PID 4 <pid>
| RMUST_T_VBN 4 <vbn> }
```

```
<dbr_activity_code> ::= {
| RMUST_K_DBR_AIJ_RCVR
| RMUST_K_DBR_BIND
| RMUST_K_DBR_FLUSH
| RMUST_K_DBR_GB_RCVR
| RMUST_K_DBR_INACTIVE
| RMUST_K_DBR_OPT_CMT
| RMUST_K_DBR_RDC_RELOAD
| RMUST_K_DBR_REDO
| RMUST_K_DBR_REFETCH
| RMUST_K_DBR_SETUP
| RMUST_K_DBR_TSN_UPDT
| RMUST_K_DBR_UNBIND
| RMUST_K_DBR_UNDO
| RMUST_K_UNKNOWN}

<pid> ::= process ID
<lkid> ::= lock ID
<vbn> ::= virtual block number
<length> ::= length, in bytes, of text
<dbr index> ::= zero-based index into RTUPB table for this process
```

1.7.3 Tag Descriptions

Description of the valid item tags for the RMUST_CT_DBR class is shown in Table 15.

Table 1–16 RMUST_CT_DBR Tags

Tag Code	Meaning
RMUST_T_ASCII_STALL	Stall text.
RMUST_T_DBR_ACTIVITY	Activity code associated with DBR state. Refer to Table 16: DBR Activity Phase Tags for valid values.
RMUST_T_DBR_INDEX	Index for this DBR process.
RMUST_T_LKID	Lock ID DBR is waiting on (0 if not blocked on lock).
RMUST_T_PID	Process ID of DBR process.
RMUST_T_VBN	Current Virtual Block Number (VBN) in RUJ file.

An enumeration of the valid values for RMUST_T_DBR_ACTIVITY is shown in Table 16. Note that not all values are returned for all versions of Oracle Rdb.

Table 1–17 DBR Activity Phase Tags

Tag Code	Meaning
RMUST_K_DBR_AIJ_RCVR	DBR is performing AIJ recovery.
RMUST_K_DBR_BIND	DBR is binding to database.
RMUST_K_DBR_FLUSH	DBR is performing buffer flush.
RMUST_K_DBR_GB_RCVR	DBR is recovering global buffer context.
RMUST_K_DBR_INACTIVE	DBR is initializing.
RMUST_K_OPT_CMT	DBR is scanning AIJ for optimistic commit.
RMUST_K_RDC_RELOAD	DBR is rebuilding the cache.
RMUST_K_DBR_REDO	DBR is performing transaction REDO.
RMUST_K_DBR_REFETCH	DBR is refetching transferred pages.

Table 1–17 (Cont.) DBR Activity Phase Tags

Tag Code	Meaning
RMUST_K_DBR_SETUP	DBR is setting up for recovery.
RMUST_K_DBR_TSN_UPDT	DBR is updating root structures.
RMUST_K_DBR_UNBIND	DBR is unbinding.
RMUST_K_DBR_UNDO	is performing transaction UNDO.
DBR	

1.8 RMUST_CT_FILE - File Information

The file information class provide data from the FILID block which contains attributes of storage area. This class can return multiple rows of data. If information about all storage areas is desired, then the length field in the TLV entry for this class should be zero. If information for a specific storage area is desired, then the value of the TLV for this class should be the DBID for that storage area. (DBIDs are returned by the RMUST_T_DBID tag.)

1.8.1 Information Buffer Syntax

```
<file_rqst> ::= RMUST_CT_FILE {0 | {4, <dbid>}}<file_rqst_tlv>... KUSGTBEND
<file_rqst_tlv> ::= {
  RMUST_T_ACCESS_MODE 0
| RMUST_T_ACTIVE_FLG 0
| RMUST_T_BACKUP_DATE_ADT 0
| RMUST_T_BACKUP_DATE_FDT 0
| RMUST_T_BACKUP_DATE_TDT 0
| RMUST_T_CALC_PNO 0
| RMUST_T_CORRUPT_FLG 0
| RMUST_T_DBID 0
| RMUST_T_EXT_COUNT 0
| RMUST_T_EXT_ENABLED_FLG 0
| RMUST_T_EXT_PERCENT 0
| RMUST_T_FILENAME 0
| RMUST_T_FILID_SLOTNUM 0
| RMUST_T_INC_RES_DATE_ADT 0
| RMUST_T_INC_RES_DATE_FDT 0
| RMUST_T_INC_RES_DATE_TDT 0
| RMUST_T_INCONSISTENT_FLG 0
| RMUST_T_LINKAGE_DBID 0
| RMUST_T_MAX_EXT_PAGCNT 0
| RMUST_T_MAX_PNO 0
| RMUST_T_MIN_EXT_PAGCNT 0
| RMUST_T_MIXED_FMT_FLG 0
| RMUST_T_PAG_BLKCNT 0
| RMUST_T_SNAPS_ALLOWED_FLG 0
| RMUST_T_SNAPS_ENABLED_FLG 0
| RMUST_T_SNAPSHOT_FLG 0
| RMUST_T_SPAMS_ENABLED_FLG 0
| RMUST_T_SPAMS_FLG 0
| RMUST_T_STAREA_NAME 0
| RMUST_T_THRESHOLD 0 }
```

1.8.2 Result Buffer Syntax

```
<file_result> ::= {KUSGTBOG <file_tlv> ... KUSGTEOG}...
<end_word>
<file_tlv> ::= {
  RMUST_T_ACCESS_MODE 4 <access_mode>
| RMUST_T_ACTIVE_FLG 4 <boolean>
| RMUST_T_BACKUP_DATE_ADT 8 <absolute_date>
| RMUST_T_BACKUP_DATE_FDT <length> <text>
| RMUST_T_BACKUP_DATE_TDT <length> <text>
| RMUST_T_CALC_PNO 4 <count_value>
| RMUST_T_CORRUPT_FLG 4 <boolean>
| RMUST_T_DBID 4 <dbid>
| RMUST_T_EXT_COUNT 4 <count_value>
| RMUST_T_EXT_ENABLED_FLG 4 <boolean>
| RMUST_T_EXT_PERCENT 4 <count_value>
| RMUST_T_FILENAME <length> <text>
| RMUST_T_FILID_SLOTNUM 4 <count_value>
| RMUST_T_INC_RES_DATE_ADT 8 <absolute_date>
| RMUST_T_INC_RES_DATE_FDT <length> <text>
| RMUST_T_INC_RES_DATE_TDT <length> <text>
| RMUST_T_INCONSISTENT_FLG 4 <boolean>
| RMUST_T_LINKAGE_DBID 4 <dbid>
| RMUST_T_MAX_EXT_PAGCNT 4 <count_value>
| RMUST_T_MAX_PNO 4 <count_value>
| RMUST_T_MIN_EXT_PAGCNT 4 <count_value>
| RMUST_T_MIXED_FMT_FLG 4 <boolean>
| RMUST_T_PAG_BLKCNT 4 <count_value>
| RMUST_T_SNAPS_ALLOWED_FLG 4 <boolean>
| RMUST_T_SNAPS_ENABLED_FLG 4 <boolean>
| RMUST_T_SNAPSHOT_FLG 4 <boolean>
| RMUST_T_SPAMS_ENABLED_FLG 4 <boolean>
| RMUST_T_SPAMS_FLG 4 <boolean>
| RMUST_T_STAREA_NAME <length> <text>
| RMUST_T_THRESHOLD <length> <threshold_array>

<dbid> ::= 32 bit unsigned DBID of storage area. This
value should be used to specify an area number for
the area statistics request.
<access_mode> ::= {
  RMUST_K_ACCESS_READ_WRITE
| RMUST_K_ACCESS_READ_ONLY
| RMUST_K_ACCESS_WORM }
<threshold_array> ::= array of three bytes
```

The descriptions of the valid item tags for the RMUST_CT_FILE class are listed in Table 17.

Table 1–18 RMUST_CT_FILE Tags

Tag Code	Meaning
RMUST_T__ACCESS_ MODE	This field specifies the access mode of the area. Valid values are defined in Table 18: File Access Mode Tags.
RMUST_T_ACTIVE_FLG	If TRUE, this area is valid and can be accessed.
RMUST_T_BACKUP_DATE_ ADT	If this field contains zero, then area has not been backed up.
RMUST_T_BACKUP_DATE_ FDT	Backup time and date (FDT Format)
RMUST_T_BACKUP_DATE_ TDT	Backup time and date (TDT Format)
RMUST_T_CALC_PNO	Number of pages in the CALC range for this storage area.

Table 1–18 (Cont.) RMUST_CT_FILE Tags

Tag Code	Meaning
RMUST_T_CORRUPT_FLG	If true, then this storage area may have been corrupted by an aborted batch update transaction.
RMUST_T_DBID	Storage area DBID of the storage area described by this FILID block. DBIDs are always one-based.
RMUST_T_EXT_COUNT	This field contains the number of times that a storage area has been extended. This field is set back to zero during a restore operation.
RMUST_T_EXT_ENABLED_FLG	If true, then this storage area may be extended. If false, then an exception is raised when the storage area becomes full.
RMUST_T_EXT_PERCENT	This field contains a percentage factor by which the storage area is to be extended.
RMUST_T_FILENAME	This field contains the resultant filename of the storage area file. If the storage area is COUPLED or RELATED, then this field contains a zero-length name.
RMUST_T_FILID_SLOTNUM	Index for this area into the filid vector.
RMUST_T_INC_RES_DATE_ADT	This field contains the time-and-date stamp of the most recent incremental area restore that was applied to this area.
RMUST_T_INC_RES_DATE_FDT	This field contains the time-and-date stamp of the most recent incremental area restore that was applied to this area. (FDT Format)
RMUST_T_INC_RES_DATE_TDT	This field contains the time-and-date stamp of the most recent incremental area restore that was applied to this area. (TDT Format)
RMUST_T_INCONSISTENT_FLG	If TRUE, this area is inconsistent with the last commit TSN of the database.
RMUST_T_LINKAGE_DBID	Linkage DBID. If this is a live storage area that has snapshots allowed, then this field contains the DBID of the snapshot area for this live storage area. If this is a snapshot area, then this field contains the DBID of the live storage area for this snapshot area.
RMUST_T_MAX_EXT_PAGCNT	This field contains the maximum number of pages by which the storage area is to be extended.
RMUST_T_MAX_PNO	Page number of the highest initialized page in the storage area.
RMUST_T_MIN_EXT_PAGCNT	This field contains the minimum number of pages by which the storage area is to be extended.
RMUST_T_MIXED_FMT_FLG	If true, this is an Oracle Rdb mixed page format area.
RMUST_T_PAG_BLKCNT	Length of each area page in blocks.
RMUST_T_SNAPS_ALLOWED_FLG	If true, then this is a live storage area that has a corresponding snapshot file.
RMUST_T_SNAPS_ENABLED_FLG	If true, then this is a live storage area that has snapshots enabled.
RMUST_T_SNAPSHOT_FLG	If true, then this is a snapshot area. If false, then this is a live storage area.
RMUST_T_SPAMS_ENABLED_FLG	If TRUE, then this area has space management pages and they are enabled. Spams can only be disabled for mixed format areas.
RMUST_T_SPAMS_FLG	If true, then this is a live storage area that has space management pages.
RMUST_T_STAREA_NAME	This field contain a name by which this storage area can be referenced. This eliminates the need of having to reference the area by name by its full file-spec.

Table 1–18 (Cont.) RMUST_CT_FILE Tags

Tag Code	Meaning
RMUST_T_THRESHOL	If this area has space management pages, this field contains the percentage factors for the space management thresholds.

Table 1–19 File Access Mode Tags

File Access Constant	Meaning
RMUST_K_ACCESS_ READ_WRITE	Storage area is marked READ-WRITE
RMUST_K_ACCESS_ READ_ONLY	Storage area is marked READ-ONLY
RMUST_K_ACCESS_WORM	Storage area is marked as Write Once Read Many

1.9 RMUST_CT_FILE_NODUP - File Information

The RMUST_CT_FILE_NODUP class provides the same information as the RMUST_CT_FILE class. However, the RMUST_CT_FILE class returns all the data requested, no matter how many times the kusdb_ get_info call is made. RMUST_CT_FILE_NODUP returns all the data requested on the first call to kusdb_get_info. On subsequent calls to kusdb_get_info, only data for those storage areas which have changed are returned. Also, data for any storage areas which have been added or dropped are returned. Note that a request for data for a single storage area is not allowed for the RMUST_CT_FILE_NODUP class.

1.10 RMUST_CT_PROCESS - Process Information

The process information class provide information about currently active database users. This class can return multiple rows of data. To return information for all active process, specify a length of zero in the class TLV. To return information for one process, specify the process index for the process of interest. Note that the RMUST_CT_PROCESS class is only valid for OpenVMS.

1.10.1 Information Buffer Syntax

```
<process_rqst> ::=    RMUST_CT_PROCESS{0 | {4 <proc index>}}
          <process_rqst_tlv>... KUSGTBEND
<process_rqst_tlv> ::= {
  RMUST_T_BIOCNT 0
| RMUST_T_CPUTIM_RT 0
| RMUST_T_DIOCNT 0
| RMUST_T_ENQCNT 0
| RMUST_T_IMAGENAME 0
| RMUST_T_NUMBIO 0
| RMUST_T_NUMDIO 0
| RMUST_T_PGFCNT 0
| RMUST_T_PGFLTS 0
| RMUST_T_PID 0
| RMUST_T_PROC_INDEX 0
| RMUST_T_PROCNAME 0
| RMUST_T_PSTATE 0
| RMUST_T_STID 0
| RMUST_T_USERNAME 0
| RMUST_T_VMSIZE 0
| RMUST_T_WSSIZE 0 }
```

1.10.2 Result Buffer Syntax

```
<process_result> ::= {KUSGTBOG <process_tlv> ... KUSGTEOG} ... |
<end_word>
<process_tlv> ::= {
  RMUST_T_BIOCNT 4 <count_value> }
| RMUST_T_CPUTIM_RT 4 <count_value>
| RMUST_T_DIOCNT 4 <count_value>
| RMUST_T_ENQCNT 4 <count_value>
| RMUST_T_IMAGENAME <length> <text>
| RMUST_T_NUMBIO 4 <count_value>
| RMUST_T_NUMDIO 4 <count_value>
| RMUST_T_PGFCNT 4 <count_value>
| RMUST_T_PGFLTS 4 <count_value>
| RMUST_T_PID 4 <value>
| RMUST_T_PROC_INDEX 4 <proc index>
| RMUST_T_PROCNAME <length> <text>
| RMUST_T_PSTATE 4 <count_value>
| RMUST_T_STID 4 <value>
| RMUST_T_USERNAME <length> <text>
| RMUST_T_VMSIZE 4 <count_value>
| RMUST_T_WSSIZE 4 <count_value> }

<proc index> ::= zero-based index into RTUPB table for this process
```

1.10.3 Tag Descriptions

Description of the item tags for the RMUST_CT_PROCESS class is shown in Table 19.

Table 1–20 RMUST_CT_PROCESS Tags

Tag	Meaning
RMUST_T_BIOCNT	Remaining buffered I/O quota.
RMUST_T_CPUTIM_RT	OS Specific CPU time consumed in 1/100 seconds.
RMUST_T_DIOCNT	Remaining direct I/O quota.
RMUST_T_ENQCNT	Remaining lock request quota.
RMUST_T_IMAGENAME	OS specific name of user program.

Table 1–20 (Cont.) RMUST_CT_PROCESS Tags

Tag	Meaning
RMUST_T_NUMBIO	Number of buffered I/O´s incurred by process.
RMUST_T_NUMDIO	Number of direct I/O´s incurred by process.
RMUST_T_PGFCNT	Remaining page file quota.
RMUST_T_PGFLTS	Number of page faults incurred by process.
RMUST_T_PID	OS specific process ID.
RMUST_T_PROC_ INDEXProcess	index.
RMUST_T_PROCNAME	OS specific process name.
RMUST_T_PSTATE	OS Specific process state string.
RMUST_T_STID	Stream ID of user process.
RMUST_T_USERNAME	OS specific user name.
RMUST_T_VMSIZE	Current virtual memory size.
RMUST_T_WSSIZE	Current working set size.

1.11 RMUST_CT_STALL - Stalled Process Information, RMUST_CT_STALL_ACT - Stalled Process Information

The RMUST_CT_STALL class is used to return information about database processes which are stalled in the database. RMUST_CT_ STALL and RMUST_CT_STALL_ACT do the same work, only RMUST_ CT_STALL_ACT returns information only for those processes that are currently stalled. RMUST_CT_STALL returns information for a process regardless of whether it is stalled. These classes can return information about multiple processes. To retrieve information about all users, specify a length of zero in the TLV for the class tag. To retrieve information for a specific stalled process, specify the stall index for that process.

1.11.1 Information Buffer Syntax

```
<stall_rqst> ::=      RMUST_CT_STALL{0 | {4 <stall index>}}
                      <stall_rqst_tlv> ... KUSGTBEND
<stall_rqst_tlv> ::= {
  RMUST_T_ASCII_DEADLOCK 0
| RMUST_T_ASCII_STALL 0
| RMUST_T_ASCII_TIMEOUT 0
| RMUST_T_DEADLOCK_SEQ 0
| RMUST_T_DEADLOCK_SINCE_ADT 0
| RMUST_T_DEADLOCK_SINCE_TT 0
| RMUST_T_DEADLOCK_SINCE_RT 0
| RMUST_T_IS_STALLED 0
| RMUST_T_LKID 0
| RMUST_T_PID 0
| RMUST_T_STALL_INDEX 0
| RMUST_T_STALL_SINCE_ADT 0
| RMUST_T_STALL_SINCE_TT 0
| RMUST_T_STALL_SINCE_RT 0
| RMUST_T_STID 0
| RMUST_T_TIMEOUT_SEQ 0
| RMUST_T_TIMEOUT_SINCE_ADT 0
| RMUST_T_TIMEOUT_SINCE_TT 0
| RMUST_T_TIMEOUT_SINCE_RT 0 }
```

1.11.2 Result Buffer Syntax

```
<stall_result> ::= { KUSGTBOG <stall_tlv> ... KUSGTEOG }... <end_word>
<stall_tlv> ::= {
  RMUST_T_ASCII_DEADLOCK <length> <text>
| RMUST_T_ASCII_STALL <length> <text>
| RMUST_T_ASCII_TIMEOUT <length> <text>
| RMUST_T_DEADLOCK_SEQ 4 <deadlock_seq>
| RMUST_T_DEADLOCK_SINCE_ADT 8 <absolute_time>
| RMUST_T_DEADLOCK_SINCE_TT <length> <time_string>
| RMUST_T_DEADLOCK_SINCE_RT <delta_time>
| RMUST_T_IS_STALLED 4 <boolean>
| RMUST_T_LKID 4 <lkid>
| RMUST_T_PID 4 <pid>
| RMUST_T_STALL_INDEX 4 <stall index>
| RMUST_T_STALL_SINCE_ADT 8 <absolute_time>
| RMUST_T_STALL_SINCE_TT <length> <time_string>
| RMUST_T_STALL_SINCE_RT 4 <delta_time>
| RMUST_T_STID 4 <count_value>
| RMUST_T_TIMEOUT_SEQ 4 <timeout_seq>
| RMUST_T_TIMEOUT_SINCE_ADT 8 <absolute_time>
| RMUST_T_TIMEOUT_SINCE_TT <length> <time_string>
| RMUST_T_TIMEOUT_SINCE_RT 4 <delta_time> }
<stall index> ::= zero-based index into RTUPB table for this process
<pid> ::= process ID
<lkid> ::= lock ID
<timeout_seq> ::= timeout sequence number
<deadlock_seq> ::= deadlock sequence number
<boolean> ::= TRUE (1) or FALSE (0)
<length> ::= length, in bytes of subsequent data
<text> ::= ASCII text string of <length> bytes
<delta_time> ::= Time value, expressed in 1/100´s of seconds.
<absolute_time>_::= QUADWORD time in VMS system format.
<time_string>_::= Formatted ASCII time string.
```

1.11.3 Tag Descriptions

Table 1–21 RMUST_CT_STALL Tags

Tag Code	Meaning
RMUST_T_ASCII_ DEADLOCK	String indicating the lock resource for the last deadlocked lock request.
RMUST_T_ASCII_STALL	String indicating the last reason that the associated process was stalled for or the current stall reason if the
RMUST_T_IS_STALLED	flag is TRUE.
RMUST_T_ASCII_TIMEOUT	String indicating the lock resource for the last lock timeout.
RMUST_T_DEADLOCK_ SEQ	Indicates the number of deadlocks that have occurred against the corresponding process since it attached to the database.
RMUST_T_DEADLOCK_ SINCE_ADT	Time and date of the last deadlock for the associated process.
RMUST_T_DEADLOCK_ SINCE_TT	Time string of the last deadlock for the associated process.
RMUST_T_DEADLOCK_ SINCE_RT	Relative Time of the last deadlock for the associated process.
RMUST_T_IS_STALLED	Boolean value indicating whether the associated process is actively blocked waiting for a database resource.
RMUST_T_LKID	Indicates the lock ID number associated with the stall, deadlock or lock timeout.
RMUST_T_PID	Specifies the process ID of the process involved in the stall, deadlock or lock timeout.
RMUST_T_STALL_ INDEXStall	index.
RMUST_T_STALL_SINCE_ ADT	Time and date of the start of the active stall.
RMUST_T_STALL_SINCE_ RT	Relative time of the start of the active stall.
RMUST_T_STALL_SINCE_ TT	Time string of the start of the active stall.
RMUST_T_STIDStream	ID of the process.
RMUST_T_TIMEOUT_SEQ	Indicates the number of lock timeouts that have occurred against the corresponding process since it attached to the database.
RMUST_T_TIMEOUT_ SINCE_ADT	Time and date of the last lock timeout for the associated process.
RMUST_T_TIMEOUT_ SINCE_TT	Time string of the last lock timeout for the associated process.
RMUST_T_TIMEOUT_ SINCE_RT	Relative time of the last lock timeout for the associated process.

1.12 RMUST_CT_RTPMS_STATS - Run Time Performance Statistics

The RMUST_CT_RTPM_STATS class is used to return run-time statistics maintained by Oracle Rdb for databases on a particular node. This request is used to return the complete run-time performance measurement statistics (RTPMS) data structure.

1.12.1 Buffer Syntax

```
rtpms_stats_request ::= RMUST_T_RTPMS {0} KUSGTBEND
```

1.12.2 Result Buffer Syntax

```
rtpms stats_response ::= KUSGTBOG RMUST_T_RTPMS {4096} <rtpms_buffer> KUSTGTEOG
<end_word>
<end_word> ::= { KUSGTBEND | KUSGTBTRN | KUSGTBCON }
<rtpms_buffer> ::= buffer containing RTPMS values.
```

1.12.3 Tag Descriptions

Table 1–22 RMUST_CT_RTPMS_STATS Tags

Tag Code	Meaning
RMUST_T_RTPMS	Tag used to indicate the complete RTPMS data structure is to be returned. The data returned is a 4096 byte data structure. The file RMUST_RTPMS.H contains a declaration for the RTPMS_t structure.

2 SGA API Entry Point Descriptions

This section describes the entry points that comprise the SGA API. Each call contains a short example. A more extensive example that uses many of the calls presented in this section can be found in the sample program listing. The README file provides further instructions about obtaining the sample program.

2.1 kuscx_allocate

2.1.1 Interface

```
kusst kuscx_allocate
  (
    kuscx_handle *ctx_handle      /* (in/out) context handle */)
  )
```

2.1.2 Description

This routine allocates and initializes a continuation context handle which is used to maintain a context across multiple calls to kusdb_get_info to receive groups of logical records such as process information records. Handles are released with the kuscx_free_handle call. A handle can also be reinitialized for use with the kuscx_init call.

2.1.3 Parameters

ctx_handle - Address to contain the handle.

2.1.4 Condition Values Returned

KUSSTOK - Normal successful completion.
KUSSTALLC - Error allocating memory.

2.1.5 Example

```
kuscx_handle ctx_handle;
kusst status;

status = kuscx_allocate (&ctx_handle);
if (status != KUSSTOK))  /* Do error handling... */
```

2.2 kuscx_free_handle

2.2.1 Interface

```
kusst kuscx_free_handle
  (
    kuscx_handle *ctx_handle      /* (in/out) continuation handle     */
  )
```

2.2.2 Description

Releases a continuation context handle previously allocated through a call to kuscx_init.

2.2.3 Parameters

ctx_handle - Address of the handle to be freed.

2.2.4 Condition Values Returned

KUSSTOK - Normal successful completion.
KUSSTICH - Invalid context handle supplied

2.2.5 Example

```
kusst status;
kuscx_handle ctx_handle;

status = kuscx_free_handle (&ctx_handle)
if (status != KUSSTOK)  /* Do error handling ... */
```

2.3 kuscx_init

2.3.1 Interface

```
kusst kuscx_init
  (
    kuscx_handle *ctx_handle      /* (in/out) continuation handle     */
  )
```

2.3.2 Description

Reinitializes a continuation context handle previously allocated through a call to kuscx_allocate. A call to kuscx_init prepares a continuation context handle for a subsequent use in a continuation context operation. It is semantically equivalent to issuing a call to kuscx_free_handle on a valid continuation context handle followed by a call to kuscx_allocate. However calling kuscx_init instead avoids the overhead of freeing memory and reallocating it.

2.3.3 Parameters

ctx_handle - Address of the handle to be initialized.

2.3.4 Condition Values Returned

KUSSTOK - Normal successful completion.
KUSSTICH - Invalid context handle supplied.

2.3.5 Example

```
kuscx_handle ctx_handle;
kusst status;

    status = kuscx_allocate (&ctx_handle);
          .
          .
          .
    status = kuscx_init (&ctx_handle);
    if (status != KUSSTOK)  /* Do error handling ... */
```

2.4 kusdb_allocate

2.4.1 Interface

```
kusst kusdb_allocate
  (
    kusdb_handle      *handle           /* (in/out) Database Handle */
  )
```

2.4.2 Description

This routine allocates memory for a DB handle. This handle is used in other kusdb calls. Typically, after calling kusdb_allocate, the caller will next call kusdb_connect. When a user is finished with the DB handle, kusdb_free must be called to free resources used by the handle.

2.4.3 Parameters

- handle - Address of a database handle This handle is updated by kusdb_connect and should be passed in subsequent calls to functions that should use this connection.

2.4.4 Condition Values Returned

- KUSSTOK - Normal successful value
- KUSSTALLC - Error allocating memory

2.4.5 Example

```
kusdb_handle db_handle;
kusst status;

  status = kusdb_allocate (&db_handle);
  if (status != KUSSTOK) ...
```

2.5 kusdb_connect

2.5.1 Interface

```
kusst kusdb_connect
  (
    kusdb_handle handle,                /* (in/out) Database Handle   */
    ub4          database_name_length,  /* (in) Length of db name     */
    char         *database_name,         /* (in) Name of database      */
    ub4          security_length,       /* (in, opt) Len of security  */
                                        /*   info 0 if not supplied   */
    char         *security_info         /* (in, opt) Security info    */
                                        /*   NULL if not supplied     */
  }
```

Note: The ub4 (security length) and char (security info) are ignored for Oracle Rdb.

2.5.2 Description

This routine connects to the specified database and then allocates and initializes the resources necessary to support gathering of statistics. The companion routine, kusdb_disconnect should be used to deallocate and release resources seized by this routine. If an error status is returned, kusdb_error_text can be called to get more information about the error.

Note: The specified database must be local to the node running the process calling kusdb_connect. Connecting to remote databases is not supported.

For Oracle Rdb databases, this call performs a ˆutility bindˆ to the database. This process does not become a user of the database as far as Oracle Rdb is concerned.

2.5.3 Parameters

- handle - Database handle This handle must have been allocated by kusdb_allocate.
- database_name_length - Name and length of the name of the database to connect.
- database_name - A database name must be specified for an Oracle Rdb database.
- security_length, security_info - Length and value of security information. If no security information is specified (the value of security_info_length is 0), then kusdb_connect uses the user name of the process.

  These fields must be 0. The Oracle Rdb API uses the name of the user running the program to access the database.

2.5.4 Condition Values Returned

- KUSSTOK - Normal successful value
- KUSSTALLC - Unable to allocate memory
- KUSSTIDH - Invalid database handle supplied

2.5.5 Example

```
kusdb_handle db_handle;
kusst status;
char db_name = "mf_personnel";
status = kusdb_connect (&db_handle, (dbname *)NULL, 0, strlen(dbname),
                        strlen(0_info));
if (status != KUSSTOK) ...
```

2.6 kusdb_disconnect

2.6.1 Interface

```
kusst kusdb_disconnect
  (
    kusdb_handle db_handle        /* (in/out) database handle */
  }
```

2.6.2 Description

This routine disconnects from a database that had been connected with a call to kusdb_connect. If the handle is no longer needed after this call, kusdb_free should be called to free resources used by the handle.

2.6.3 Parameters

- db_handle - The handle of the database that is no longer needed.

2.6.4 Condition Values Returned

- KUSSTOK - Normal successful completion
- KUSSTIDH - Invalid handle

2.6.5 Example

```
kusdb_handle db_handle;
kusst status;

status = kusdb_disconnect (db_handle);
```

2.7 kusdb_free

2.7.1 Interface

```
kusst kusdb_free
  (
    kusdb_handle      *handle           /* (in/out) Database Handle */
  }
```

2.7.2 Description

This routine frees memory associated with a DB handle as a result of calling kusdb_allocate.

2.7.3 Parameters

- handle - Address of a database handle

2.7.4 Condition Values Returned

- KUSSTOK - Normal successful value
- KUSSTIDH - Invalid database handle supplied

2.7.5 Example

```
kusdb_handle db_handle;
kusst status;

status = kusdb_free (&db_handle);
if (status != KUSSTOK) ...
```

2.8 kusdb_error_text

2.8.1 Interface

```
kusst kusdb_error_text
  (
    kusdb_handle db_handle        /* (IN) handle of database */
    char        *text_buf,       /* (IN) Ptr to user-supplied buffer */
    ub4         text_buf_len     /* (IN) Length of supplied buffer   */
    ub4          *text_len_out    /* (OUT) Number of bytes written */
  )
```

2.8.2 Description

Formats the status of the last kusdb_get_info or kusdb_connect call into text and inserts it into a user supplied buffer.

2.8.3 Parameters

- db_handle - Handle of the database whose last error is to be displayed.
- text_buf - Address of an allocated piece of memory that will contain the formatted message.
- text_buf_len - The number of bytes allocated in text_buf.

- text_len_out - The number of bytes written into text_buf.

2.8.4 Condition Values Returned

- KUSSTOK - Normal successful completion
- KUSSTALLC - Unable to allocate memory
- KUSSTIDH - Invalid database handle supplied

2.8.5 Example

```
text err_buffer [512];
ub4 err_buffer_len, err_buffer_len;
kusst status;

if (KUSSTOK != kusdb_get_info (db_handle†);
{
  if (KUSSTOK == kusdb_error_text (db_handle, err_buffer,
                                   sizeof (err_buffer),&err_buffer_len))
    fprintf (stderr, "ERROR: %.*s\n", err_buffer_len, err_buffer);
  else
    fprintf (stderr, "ERROR FORMATTING STATUS\n");
}
```

2.9 kusdb_get_info

2.9.1 Interface

```
kusst kusdb_get_info
  (
    kusdb_handle db_handle,       /* (in)database handle              */
    kuscx_handle cont_handle,     /* (in,opt) Continuation handle     */
                                  /*      NULL if not supplied        */
    ub1          *info_buffer,    /* (in) address of info buffer      */
    ub4         info_buffer_length,
                                  /* (in) bytes in info buffer        */
    ub1         *result_buffer,  /* (in/out) address of result buffer*/
    ub4         *result_buffer_length,
                                  /* (in/out) bytes in result buffer  */
    kustvs      *tlv_state       /* (out) termination tag for        */
                                  /*      result buffer               */
  )
```

2.9.2 Description

This call implements the main statistics gathering interface. Communication with the statistics interface is performed by building a request buffer to specify the information items you´re interested in retrieving. The kusdb_get_info call uses the information buffer to determine what information to return in the result buffer. Information on the request and response buffer formats is presented on page 6, Information and Result Buffer Formats.

2.9.3 Parameters

- db_handle - An initialized handle returned by a call to kusdb_connect.
- cont_handle - An initialized continuation context handle returned by a call to either kuscx_allocate or kuscx_init.
- info_buffer - Address of an allocated and initialized buffer that indicates the type of information to be retrieved.
- info_buffer_length - Number of valid bytes in info_buffer.
- result_buffer - Address of a caller allocated before that kusdb_get_info uses to return information to the caller.
- result_buffer_length - When kusdb_get_info is called, this parameter contains the number of bytes allocates by the caller for the return_buf parameter. When the routine has completed execution, this parameter contains the number of bytes that were written by the routine into return_buf.
- tlv_state - Address of a variable to hold the state of the termination of the result buffer. Valid values are: KUSTVS_CONTBuffer ended with KUSGTBCNT tag. Returned only if cont_handle was specified and there is more data to be returned for the request. To get more of the requested data, call kusdb_get_info again specifying the same info_buffer and cont_handle. KUSTVS_ENDBuffer ended with KUSGTBEND tag. Returned if all requested data fit in the result_buffer. KUSTVS_TRUNCBuffer ended with KUSGTTRNC tag. Returned only if cont_handle was not specified and there was more data to be returned for the request. KUSTVS_UNKBuffer not terminated or TLV handle has not been used to read or write the last tag. This value is returned if an error occured while processing the request.

2.9.4 Condition Values Returned

- KUSSTOK - Normal successful completion
- KUSSTALLC - Unable to allocate memory
- KUSSTBDKY - Bad index value for class tag
- KUSSTBFSML - Buffer too small for data
- KUSSTBINF - Information buffer contains a bad tag
- KUSSTDIS - Statistics collection disabled for this database
- KUSSTIBE - Information buffer is empty
- KUSSTICH - Invalid context handle supplied
- KUSSTIDH - Invalid database handle supplied
- KUSSTUNKC - Unknown class tag specified

2.9.5 Example

```
#define INFO_BUFFER_LEN 1000
#define RESULT_BUFFER_LEN 10000

kusdb_handle db_handle;
kuscx_handle cont_handle;
kusst status;
kustv_handle info_tlv_handle;
kustvs tlv_state = KUSTVS_CONT;
ub1 info_buffer[INFO_BUFFER_LEN], data_buffer[RESULT_BUFFER_LEN];
ub4 result_buffer_length=RESULT_BUFFER_LEN

/* allocate a db handle and connect to the database */

status = kusdb_allocate (&db_handle);
status = kusdb_connect (db_handle...);

/* initialize the context handle */

status = kuscx_allocate (&cont_handle);

/* initialize TLV handle for info buffer and call kustv_put to load it */

status = kustv_allocate_handle (&info_tlv_handle, info_buffer,
                        INFO_BUFFER_LEN);
status = kustv_put (info_tlv_handle...);
...

/* collect information */

while (tlv_state == KUSTVS_CONT)
{
  status = kusdb_get_info (db_handle, cont_handle, info_buffer,
                           INFO_BUFFER_LEN, data_buffer,
                           &result_buffer_len, &tlv_state);
  /* process data_buffer */
  . . .
}
```

2.10 kustv_allocate_handle

2.10.1 Interface

```
kusst kustv_allocate_handle
  (
    kustv_handle      *tlv_handle,      /* (in/out) handle to init    */
    ub1               *tlv_buffer,      /* (in) TLV buffer            */
    ub4               buf_size)        /* (in) bytes in buffer       */
  )
```

2.10.2 Description

Associates a TLV handle with a TLV buffer allocated by the user. kustv_allocate_handle must be called to initialize the handle prior to using any other TLV support routine. kustv_reinit_handle can be used to reinitialize a handle so that it points to the beginning of the buffer.

If the size is not known, specify a value of -1 for buf_size. This should be done only with buffers that are known to have been terminated with a valid termination tag (KUSGTBEND, KUSGTBTRN, or KUSGTBCNT) such as result buffers written by calls to kusdb_get_info. If -1 is specified for buf_size, then tlv_handle can be used only to read from the buffer. Any attempt to write to such a buffer will fail with a status of KUSSTNWR.

2.10.3 Parameters

- tlv_handle - Handle to be initialized for this TLV buffer. This handle is passed in other calls to manipulate the TLV.
- tlv_buffer - Address of the TLV buffer to be associated with tlv_handle.
- buf_size - Number of bytes in tlv_buffer.

2.10.4 Condition Values Returned

- KUSSTOK - Success
- KUSSTALLC - Error allocating memory

2.10.5 Example

```
char ret_buf [8192];
kustv_handle ret_tlv;
kusst status;

status = kustv_allocate_handle (&ret_tlv, ret_buf, sizeof (ret_buf));
```

2.11 kustv_dump

2.11.1 Interface

```
kusst kustv_dump
  (
    kustv_handle      *tlv_handle,      /* (in) TLV buffer to dump    */
    CONST char        *header,          /* (in, opt) header           */
    int               data_flg,        /* (in) print values?        */
    int               *append_flg,      /* (in) append to output file */
    CONST char        *directory_name,  /* (in) directory for file    */
    CONST char        *file_name        /* (in) File to dump into     */
  )
```

2.11.2 Description

Prints a formatted display of the specified TLV buffer described by tlv_handle to the file specified by file_name in the directory specified by directory_name. The parameters file_name and directory_name are both optional. If file_name is not specified, the output is sent to stderr. If the append_flg parameter is true, then the output is appended to an existing file. Otherwise, a new file is created. If the append_flag is non-zero and the file specified by file_name does not exist, then the KUSSTDMPOP status is returned.

This routine is useful for debugging purposes. The optional header argument can be used to display a descriptive title in the header output. If data_flg is non-zero, a hex dump of the values associated with each TLV entry will be displayed along with the buffer format.

2.11.3 Parameters

- tlv_handle - Handle of the TLV buffer whose contents are to be dumped.
- header - Pointer to ASCIZ string containing text to be printed before the dump is performed.
- data_flg - Flag used to indicate if the data in each TLV entry should be displayed along with the tag and the length. If set to a non-zero value, the data for each TLV is displayed. If set to zero, only the tag and length are displayed.
- append_flg - Flag to indicate if an existing instance of the output file should be appended to (or replaced)
- directory - ASCIZ string containing a directory specification for the output file. If an empty string is specified, the current default directory is used.
- file_name - ASCIZ string containing a file name for the output file. If absent, stderr is used and append_flg and directory are ignored. If specified without an extension, the Oracle extension for an output file on the given platform is used.

2.11.4 Example

```
kustv_handle tlv_handle;
kusst status;
ub1 tlv_buff[1000];

status = kustv_allocate_handle (&tlv_handle, tlv_buff, 1000);
status = kustv_put (tlv_handle, . . .);

status = kustv_dump (tlv_handle, "Dump of TLV buffer just filled", 1,
                     1, "", NULL);
```

2.12 kustv_free_handle

2.12.1 Interface

```
kusst kustv_free_handle
  (
    kustv_handle      *tlv_handle,      /* (in/out) handle to initialize*/
  )
```

2.12.2 Description

Frees resources used by a TLV handle. This routine is called when the buffer described by a handle is no longer needed. Note that calling kustv_reinit_handle is the semantic equivalent of calling kustv_free followed by kustv_allocate.

2.12.3 Parameters

- tlv_handle - Handle whose resources are to be freed.

2.12.4 Condition Values Returned

- KUSSTOK - Success
- KUSSTITH - Invalid TLV handle

2.12.5 Example

```
char ret_buf [8192];
kustv_handle ret_tlv;
kusst status;

status = kustv_allocate_handle (&ret_tlv, ret_buf, sizeof (ret_buf));
    . . .
status = kustv_free_handle (&ret_tlv);
```

2.13 kustv_get

2.13.1 Interface

```
kusst kustv_get
  (
    kustv_handle tlv_hndl,  /* (in) handle for TLV structure          */
    ub2         *tag,            /* (out) Tag of next entry          */
    ub2         *length,   /* (out) Length of value field in bytes   */
    dvoid       *value     /* (out) Pointer to value field           */
  )
```

2.13.2 Description

Returns the next TLV set associated with the TLV handle and updates tlv_hndl to point to the next TLV in the buffer. If the return value is not KUSSTOK, then the output variables (tag, length, and value) do not contain valid data.

Note that kustv_get should be used only on TLV buffers that have been terminated by TLVs with tags KUSGTBEND, KUSGTBTRN, or KUSGTBCNT. These tags are the only way for kustv_get to know when the last TLV has been read. The kusdb_get_info function always ends the results buffer with one of these tags, unless an error occurred.

2.13.3 Parameters

- tlv_hndl - Handle for TLV whose offset is to be returned.
- tag - Value of tag field in next TLV entry.
- length - Value of length field in next TLV entry.
- value - Pointer to value in the TLV entry. If value returned for length is 0, then value should not be used (it may not point to valid memory.)

2.13.4 Condition Values Returned

- KUSSTOK - Success
- KUSSTCONT - Buffer continuation tag was read by a previous get
- KUSSTEOB - End of buffer encountered by a previous get
- KUSSTITH - Illegal tag handle
- KUSSTTRNC - Buffer truncation tag was read by a previous get

2.13.5 Example

```
kustv_handle ret_tlv;
unsigned short tag;
unsigned short length;
dvoid *pvalue;
kusst status;

status = kustv_get (ret_tlv, &tag, &length, &pvalue);
```

2.14 kustv_offset

2.14.1 Interface

```
kusst kustv_offset
  (
    kustv_handle tlv_hndl;  /* (in) handle for TLV structure          */
    ub4 *offset;                  /* (out) current offset for tlv_hndl*/
  )
```

2.14.2 Description

Returns the current offset of tlv_hndl into the TLV structure. The current offset is the byte offset at which the next read or write operation will occur. The current offset can be reinitialized by calling kustv_reinit_handle. This routine can be useful for debugging purposes.

2.14.3 Parameters

- tlv_hndl - Handle for TLV whose offset is to be returned
- offset - Offset into the TLV buffer of the next TLV entry

2.14.4 Condition Values Returned

- KUSSTOK - Success
- KUSSTITH - Illegal tag handle

2.14.5 Example

```
ub4 offset;
text buff[10000];
kustv_handle tlv_hndl;
kusst status;

...

status = kustv_offset (tlv_hndl, &offset);
```

2.15 kustv_put

2.15.1 Interface

```
kusst kustv_put
  (
    kustv_handle tlv_handle,      /* (in/out) handle                  */
    ub2          tag,             /* (in) Tag                         */
    ub2          length           /* (in) length of value buffer      */
    dvoid        *value           /* (in) address of value            */
  )
```

2.15.2 Description

Adds a TLV into a TLV buffer. Before this TLV entry is written, kustv_put makes sure there is enough room for this TLV (plus enough room for a TLV containing a termination tag if this TLV does not contain a termination tag). If there is no room for this TLV entry, then a TLV with tag KUSGTBTRN is written into the buffer. If a termination tag has already been written to the TLV, then the TLV data is not added and the return status indicates how the buffer was terminated.

2.15.3 Parameters

- tlv_handle - Handle of the TLV buffer that gets this TLV entry
- tag - tag for the new TLV entry
- length - length of the value to be inserted into the TLV
- value - address of value to be copied into the TLV buffer. If length is zero, then value is ignored.

2.15.4 Condition Values Returned

- KUSSTOK - Success
- KUSSTBFSML - Buffer is empty and this TLV is too large to fit into it. No TLV was written into the buffer.
- KUSSTCONT - Buffer is terminated with KUSGTBCNT. TLV was not added.
- KUSSTEOB - Buffer is terminated with KUSGTBEND. TLV was not added.
- KUSSTNWR - TLV put operation attempted on a TLV buffer that doesn't have a length associated with it.

- KUSSTTRNC - Buffer is terminated with KUSGTBTRN. TLV was not added.

2.15.5 Example

```
kusst status;
kustv_handle info_tlv;
char *area_name = ^Area1~;
char info_buf[1000];

status = kustv_allocate (&info_tlv, info_buf, 1000);
status = kustv_put (info_tlv, RMUST_T_STAREA_NAME,
                    strlen (area_name), area_name);
```

2.16 kustv_reinit_handle

2.16.1 Interface

```
kusst kustv_reinit_handle
  (
    kustv_handle      tlv_handle,/* (in/out) handle to initialize    */
    ub1               *buffer,   /* (in) buffer for the handle       */
    ub4               buf_size   /* (in) size of buffer              */
  )
```

2.16.2 Description

Reset the handle so that the offset points to the beginning of the specified buffer. This is useful when rereading or reusing a buffer.

2.16.3 Parameters

- tlv_handle - Handle of the TLV buffer to be reset
- buffer - address of user allocated buffer for this handle
- buf_size - number of bytes in buffer

2.16.4 Condition Values Returned

- KUSSTOK - Success
- KUSSTITH - Invalid TLV handle

2.16.5 Example

```
kustv_handle ret_tlv;
ub1 tlv_buff[1000];
kusst status;

status = kustv_reinit_handle (ret_tlv, tlv_buff, 1000);
```

2.17 kustvp1_put_one_byte

2.17.1 Interface

```
kusst kustvp1_put_one_byte
  (
    kustv_handle tlv_handle,      /* (in/out) TLV handle        */
    ub2          tag,             /* (in) Tag                        */
    ub1          value            /* (in) Value                      */
  )
```

2.17.2 Description

Adds a TLV with a value field of a single byte into a TLV buffer. See documentation for kustv_put for information about what happens when the TLV to be added will not fit into the buffer.

2.17.3 Parameters

- tlv_handle - handle of the TLV buffer that gets this TLV entry added
- tag - tag for the new TLV entry
- value - 1-byte value of the new TLV entry

2.17.4 Condition Values Returned

- KUSSTOK - Success
- KUSSTBFSML - Buffer is empty and this TLV is too large to fit into it. No TLV was written into the buffer.
- KUSSTCONT - Buffer is terminated with KUSGTBCNT. TLV was not added.
- KUSSTEOB - Buffer is terminated with KUSGTBEND. TLV was not added.
- KUSSTNWR - TLV put operation attempted on a TLV buffer that doesn't have a length associated with it.
- KUSSTTRNC - Buffer is terminated with KUSGTBTRN. TLV was not added.

2.17.5 Example

```
ub1 my_c;
ub2 tag;
kustv_handle tlv_hndl;
kusst status;

    ...

status = kustvp1_put_one_byte (tlv_hndl, tag, my_c);
```

2.18 kustvp2_put_two_bytes

2.18.1 Interface

```
kusst kustvp2_put_two_bytes
  (
    kustv_handle tlv_handle,      /* (in/out) TLV handle        */
    ub2          tag,             /* (in)Tag                          */
    ub2          value            /* (in) value to place in the TLV   */
  )
```

2.18.2 Description

Adds a TLV with a value field of two bytes into a TLV buffer. See documentation for kustv_put for information about what happens when the TLV to be added will not fit into the buffer.

2.18.3 Parameters

- tlv_handle - handle of the TLV buffer that gets this TLV entry added
- tag - tag for the new TLV entry
- value - 2-byte value of the new TLV entry

2.18.4 Condition Values Returned

- KUSSTOK - Success
- KUSSTBFSML - Buffer is empty and this TLV is too large to fit into it. No TLV was written into the buffer.
- KUSSTCONT - Buffer is terminated with KUSGTBCNT. TLV was not added.
- KUSSTEOB - Buffer is terminated with KUSGTBEND. TLV was not added .
- KUSSTNWR - TLV put operation attempted on a TLV buffer that doesn't have a length associated with it.
- KUSSTTRNC - Buffer is terminated with KUSGTBTRN. TLV was not added.

2.18.5 Example

```
ub2 my_w;
ub2 tag;
kustv_handle tlv_hndl;
kusst status;
    ...

    status = kustvp2_put_two_bytes (tlv_hndl, tag, my_w);
```

2.19 kustvp4_put_four_bytes

2.19.1 Interface

```
kusst kustvp4_put_four_bytes
  (
    kustv_handle tlv_handle,/* (in/out) TLV handle              */
    ub2          tag,       /* (in) Tag                              */
    ub4          value      /* (in) Value                            */
  )
```

2.19.2 Description

Adds a TLV entry with a value field of 4 bytes into a TLV buffer. See documentation for kustv_put for information about what happens when the TLV to be added will not fit into the buffer.

2.19.3 Parameters

- tlv_handle - handle of the TLV buffer that gets this TLV entry added
- tag - tag for the new TLV entry
- value - 4-byte value of the new TLV entry

2.19.4 Condition Values Returned

- KUSSTOK - Success
- KUSSTBFSML - Buffer is empty and this TLV is too large to fit into it. No TLV was written into the buffer.
- KUSSTCONT - Buffer is terminated with KUSGTBCNT. TLV was not added.
- KUSSTEOB - Buffer is terminated with KUSGTBEND. TLV was not added.
- KUSSTNWR - TLV put operation attempted on a TLV buffer that doesn't have a length associated with it.
- KUSSTTRNC - Buffer is terminated with KUSGTBTRN. TLV was not added.

2.19.5 Example

```
ub4 my_c;
ub2 tag;
kustv_handle tlv_hndl;
kusst status;
    ...

    status = kustvp4_put_four_bytes (tlv_hndl, tag, my_c);
```

2.20 kustvptg_put_tag

2.20.1 Interface

```
kusst kustvptg_put_tag
  (
    kustv_handle tlv_handle,      /* (in/out) handle            */
    ub2          tag              /* (in) Tag                   */
  )
```

2.20.2 Description

Write a TLV entry that contains a tag with no value into the TLV described by tlv_handle. See documentation for kustv_put for information about what happens when the TLV to be added will not fit into the buffer.

2.20.3 Parameters

- tlv_handle - handle of the TLV buffer that gets this TLV entry added
- tag - tag for the new TLV entry

2.20.4 Condition Values Returned

- KUSSTOK - Success
- KUSSTBFSML - Buffer is empty and this TLV is too large to fit into it. No TLV was written into the buffer.
- KUSSTCONT - Buffer is terminated with KUSGTBCNT. TLV was not added.
- KUSSTEOB - Buffer is terminated with KUSGTBEND. TLV was not added.
- KUSSTNWR - TLV put operation attempted on a TLV buffer that doesn't have a length associated with it.
- KUSSTTRNC - Buffer is terminated with KUSGTBTRN. TLV was not added.

2.20.5 Example

```
kustv_handle tlv_hndl;
kusst status;
    ...
    status = kustvptg_put_tag (tlv_hndl, KUSGTBEND);
```