F

# Background Processes

An Oracle Database **background process** is defined as any process that is listed in `V$PROCESS` and has a non-null value in the `PNAME` column.

Table F-1 describes Oracle Database background processes.

The External Properties column lists the type of instance in which the process runs. If the process is specific to a particular feature, then the column names the feature.

> **Note:**
>
> When the `THREADED_EXECUTION` initialization parameter is set to `TRUE` on Linux and UNIX, the `DBW`, `PMON`, `PSP`, and `VKTM` background processes run as operating system processes, and the other background processes run as operating system threads.
>
> See "THREADED_EXECUTION" for more information about the `THREADED_EXECUTION` initialization parameter.

**Table F-1 Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| ABMR | Auto BMR Background Process | Coordinates execution of tasks such as filtering duplicate block media recovery requests and performing flood control | When a process submits a block media recovery request to ABMR, it dynamically spawns worker processes (BMR*n*) to perform the recovery. ABMR and BMR*n* terminate after being idle for a long time.<br>**See Also:** *Oracle Database Backup and Recovery User's Guide* | Database instances |
| ACFS | Oracle Advanced Cluster File System (Oracle ACFS) CSS Process | Tracks the cluster membership in CSS and informs the file system driver of membership changes | The ACFS process delivers CSS membership changes to the cluster file system. These membership changes are required for the file system to maintain file system consistency within the cluster. | Oracle ASM instances, Oracle RAC |
| ACMS | Atomic Control File to Memory Service Process | Coordinates consistent updates to a control file resource with its SGA counterpart on all instances in an Oracle RAC environment | The ACMS process works with a coordinating caller to ensure that an operation is executed on every instance in Oracle RAC despite failures. ACMS is the process in which a distributed operation is called. As a result, this process can exhibit a variety of behaviors. In general, ACMS is limited to small, nonblocking state changes for a limited set of cross-instance operations. | Database instances, Oracle RAC |
| AMB*n* | **See** ASMB, AMB*n* | | | |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| AP*nn* | Replication Apply Process Coordinator Process | Obtains transactions from the reader server and passes them to apply servers | The coordinator process name is AP*nn*, where *nn* can include letters and numbers.<br>For more information about the coordinator process, see `V$XSTREAM_APPLY_COORDINATOR` for XStream and `V$GG_APPLY_COORDINATOR` for Oracle GoldenGate.<br>**See Also:** *Oracle Database XStream Guide* | Database instances, Logical Standby, XStream Inbound servers, XStream Outbound servers, GoldenGate Integrated Replicat |
| AQPC | AQ Process Coordinator | Per instance AQ global coordinator | AQPC is responsible for performing administrative tasks for AQ Primary Class Processes including commands like starting, stopping, and other administrative tasks. This process is automatically started on instance startup. | Database instances Advanced Queueing |
| ARB0 | ASM Rebalance Process | Rebalances data extents within an Oracle ASM disk group | ARB0 uses the value of the `ASM_POWER_LIMIT` initialization parameter for the Oracle ASM instance as the maximum power for disk rebalancing. | Oracle ASM instances |
| ARC*n* | Archiver Process | Copies the redo log files to archival storage when they are full or an online redo log switch occurs | ARC*n* processes exist only when the database is in `ARCHIVELOG` mode and automatic archiving is enabled, in which case ARC*n* automatically archives online redo log files. LGWR cannot reuse and overwrite an online redo log group until it has been archived.<br>The database starts multiple archiver processes as needed to ensure that the archiving of filled online redo logs does not fall behind. Possible processes are ARC0-ARC9 and ARCa-ARCt.<br>The `LOG_ARCHIVE_MAX_PROCESSES` initialization parameter specifies the number of ARC*n* processes that the database initially invokes.<br>**See Also:** *Oracle Database Concepts* and *Oracle Database Administrator's Guide* | Database instances |
| ARS*n* | ASM Recovery Worker Process | Recovers ASM transactional operations | The ASM RBAL background process coordinates and spawns one or more of these worker processes to recover terminated ASM transactional operations. These processes run only in the Oracle ASM instance.<br>Possible processes are ARS0-ARS9. | Oracle ASM instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| ASMB, AMB*n* | ASM Background Process | Communicates with an Oracle ASM instance, managing storage and providing statistics | In a database instance, the ASMB and AMB*n* processes enable the database instance to connect to an Oracle ASM instance in order to access Oracle ASM disk groups. Possible processes are ASMB and AMB1-AMB3.<br>In an Oracle ASM instance, the ASMB process runs when the ASMCMD `cp` command runs, or when a database instance first starts if the server parameter file is stored in Oracle ASM. ASMB also runs with Oracle Cluster Registry on Oracle ASM. The only possible process is ASMB; AMB*n* processes do not run in Oracle ASM instances.<br>In an Oracle IOServer (IOS) instance, the ASMB process enables the IOS instance to connect to an Oracle ASM instance in order to access Oracle ASM disk groups. The only possible process is ASMB; AMB*n* processes do not run in IOS instances. | Database instances, Oracle ASM instances, Oracle IOServer (IOS) instances |
| AS*nn* | Replication Apply Reader or Apply Server | Computes dependencies between logical change records (LCRs) and assembles messages into transactions (Reader Server)<br>Applies LCRs to database objects or passes LCRs and user messages to their appropriate apply handlers (Apply Server) | When the reader server finishes computing dependencies between LCRs and assembling transactions, it returns the assembled transactions to the coordinator process. Query `V$STREAMS_APPLY_READER`, `V$XSTREAM_APPLY_READER`, and `V$GG_APPLY_READER` for information about the reader server background process.<br>An apply server receives the transactions from the coordinator background process, and either applies database changes in LCRs or sends LCRs or messages to apply handlers. Apply servers can also enqueue a queue. If an apply server encounters an error, then it then tries to resolve the error with a user-specified conflict handler or error handler. If an apply server cannot resolve an error, then it rolls back the transaction and places the entire transaction, including all of its messages, in the error queue. When an apply server commits a completed transaction, this transaction has been applied. When an apply server places a transaction in the error queue and commits, this transaction also has been applied. Query `V$STREAMS_APPLY_SERVER` for information about the apply server background process. For XStream Inbound servers, query `V$XSTREAM_APPLY_SERVER`. For GoldenGate Integrated Replicat, query `V$GG_APPLY_SERVER`.<br>The coordinator process name is AS*nn*, where *nn* can include letters and numbers. | Database instances, XStream Outbound servers, XStream Inbound servers, GoldenGate Integrated Replicat |
| BG*nn* | Background Scheduler Group Process | Runs assorted background maintenance actions in the database | The database instance runs various background maintenance tasks that are necessary for database operation. BG*nn* runs assorted background actions for these maintenance tasks. | Database instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| BMR*n* | Automatic Block Media Recovery Worker Pool Process | Fetches blocks from a real-time readable standby database | When a process submits a block media recovery request to ABMR, it dynamically spawns worker processes (BMR*n*) to perform the recovery. BMR*n* processes fetch blocks from a real-time readable standby database. ABMR and BMR*n* terminate after being idle for a long time.<br>**See Also:** *Oracle Database Backup and Recovery User's Guide* | Database instances |
| B*nnn* | ASM Blocking Worker Process for GMON | Performs maintenance actions on Oracle ASM disk groups | B*nnn* performs actions that require waiting for resources on behalf of GMON. GMON must be highly available and cannot wait.<br>A B*nnn* worker is spawned when a disk is taken offline in an Oracle ASM disk group. Offline timer processing and drop of the disk are performed in this worker. Up to five process (B000 to B004) can exist depending on the load. | Oracle ASM instances |
| BW*nn* | Database Writer Process | Writes modified blocks from the database buffer cache to the data files | See the **Long Description** for the DBW*n* process in this table for more information about the BW*nn* process. | Database instances |
| CJQ0 | Job Queue Coordinator Process | Selects jobs that need to be run from the data dictionary and spawns job queue worker processes (J*nnn*) to run the jobs | CJQ0 is automatically started and stopped as needed by Oracle Scheduler.<br>The `JOB_QUEUE_PROCESSES` initialization parameter specifies the maximum number of processes that can be created for the execution of jobs. CJQ0 starts only as many job queue processes as required by the number of jobs to run and available resources.<br>**See Also:** *Oracle Database Concepts* and *Oracle Database Administrator's Guide* | Database instances |
| CKPT | Checkpoint Process | Signals DBW*n* at checkpoints and updates all the data files and control files of the database to indicate the most recent checkpoint | At specific times CKPT starts a checkpoint request by messaging DBW*n* to begin writing dirty buffers. On completion of individual checkpoint requests, CKPT updates data file headers and control files to record most recent checkpoint.<br>CKPT checks every three seconds to see whether the amount of memory exceeds the value of the `PGA_AGGREGATE_LIMIT` initialization parameter, and if so, takes the action described in "PGA_AGGREGATE_LIMIT".<br>**See Also:** *Oracle Database Concepts* | Database instances, Oracle ASM instances |
| CL*nn* | Cleanup Worker Process | Performs cleanup of terminated processes | Cleanup workers assist in the cleanup of terminated processes and terminated sessions. The number of workers will be proportional to the amount of cleanup work to be done and the current efficiency of cleanup. | Database instances, Oracle ASM instances |
| CLG | Persistent Cluster Flash Cache Background Process | For Oracle Data Appliance only, this process performs actions related to recovery of a terminated instance's database flash cache | For Oracle Data Appliance only, in the event of an instance crash, the running instance will recover the crashed instance's database flash cache. The CLG process will perform actions related to scanning the crashed instance's database flash cache and claim flash blocks mastered by the crashed instance. | Database instances, Oracle RAC |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| CLMN | Cleanup Main Process | Performs cleanup of terminated processes, terminated sessions, terminated transactions, and terminated network connections | CLMN periodically performs cleanup of all the following: terminated processes, terminated sessions, transactions, network connections, idle sessions, detached transactions, and detached network connections that have exceeded their idle timeout. | Database instances, Oracle ASM instances |
| CP*nn* | Replication Capture Process | Captures database changes from the redo log by using the infrastructure of LogMiner | The capture process name is CP*nn*, where *nn* can include letters and numbers. The underlying LogMiner process name is MS*nn*, where *nn* can include letters and numbers. The capture process includes one reader server that reads the redo log and divides it into regions, one or more preparer servers that scan the redo log, and one builder server that merges redo records from the preparer servers. Each reader server, preparer server, and builder server is a process. Query the `V$XSTREAM_CAPTURE` and `V$GOLDENGATE_CAPTURE` views for information about this background process.<br>**See Also:***Oracle Database XStream Guide* | Database instances, XStream Outbound Servers |
| CR*nn* | LMS CR Worker Process | Offloads the work from LMS so that blocks that require lots of UNDO to be applied do not block the LMS. Such requests are passed on to the worker so that the LMS is not stalled | There can be a maximum of eight CR processes per LMS process, with names from CR00 to CR07. Each LMS has its own set with similar name. The CR*nn* processes are threads and the process ID part will be the same as the owning LMS's process ID.<br>The names for CRnn processes will have the format CR0*n*_<spawning process id>_<thread id>. | Oracle RAC |
| CS*nn* | I/O Calibration Process | Issues I/Os to storage as part of storage calibration. | CS*nn* worker processes are started on execution of the `DBMS_RESOURCE_MANAGER.CALIBRATE_IO()` procedure. There is one worker process per CPU on each node of the database. | Database instances, Oracle RAC |
| CTWR | Change Tracking Writer Process | Tracks changed data blocks as part of the Recovery Manager block change tracking feature | CTWR tracks changed blocks as redo is generated at a primary database and as redo is applied at a standby database. The process is slightly different depending on the type of database.<br>**See Also:** *Oracle Database Backup and Recovery User's Guide* | Database instances |
| CX*nn* | Replication Capture Worker Process | Sends captured LCRs to a receiver, such as an XStream Outbound Server | The capture worker process name is CX*nn*, where *nn* can include letters and numbers. | Database instances, XStream Outbound Server |
| DBRM | Database Resource Manager Process | Sets resource plans and performs other tasks related to the Database Resource Manager | If a resource plan is not enabled, then this process is idle.<br>**See Also:** *Oracle Database Administrator's Guide* | Database instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| DBW*n* | Database Writer Process | Writes modified blocks from the database buffer cache to the data files | The primary responsibility of the Database Writer Process is to write data blocks to disk. It also handles checkpoints, file open synchronization, and logging of Block Written records.<br>In many cases the blocks that the Database Writer Process writes are scattered throughout the disk. Thus, the writes tend to be slower than the sequential writes performed by LGWR. The Database Writer Process performs multiblock writes when possible to improve efficiency. The number of blocks written in a multiblock write varies by operating system.<br>The `DB_WRITER_PROCESSES` initialization parameter specifies the number of Database Writer Processes. There can be 1 to 100 Database Writer Processes. The names of the first 36 Database Writer Processes are DBW0-DBW9 and DBWa-DBWz. The names of the 37th through 100th Database Writer Processes are BW36-BW99. The database selects an appropriate default setting for the `DB_WRITER_PROCESSES` parameter or adjusts a user-specified setting based on the number of CPUs and processor groups.<br>**See Also**: "DB_WRITER_PROCESSES" | Database instances |
| DIA0 | Diagnostic Process | Detects and resolves hangs and deadlocks | | Database instances, Oracle ASM instances |
| DIAG | Diagnostic Capture Process | Performs diagnostic dumps | DIAG performs diagnostic dumps requested by other processes and dumps triggered by process or instance termination. In Oracle RAC, DIAG performs global diagnostic dumps requested by remote instances. | Database instances, Oracle ASM instances |
| DM*nn* | Data Pump Control Job Process | Coordinates the Data Pump job tasks performed by Data Pump worker processes and handles client interactions | The Data Pump control job process is started during job creation and coordinates all tasks performed by the Data Pump job. It handles all client interactions and communication, establishes all job contexts, and coordinates all worker process activities on behalf of the job. | Database instances, Data Pump |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| DMON | Data Guard Broker Monitor Process | Manages and monitors a database that is part of a Data Guard broker configuration | When you start the Data Guard broker, a DMON process is created. DMON runs for every database instance that is managed by the broker. DMON interacts with the local database and the DMON processes of the other databases to perform the requested function. DMON also monitors the health of the broker configuration and ensures that every database has a consistent description of the configuration.<br>DMON maintains profiles about all database objects in the broker configuration in a binary configuration file. A copy of this file is maintained by the DMON process for each of the databases that belong to the broker configuration. The process is created when the `DG_BROKER_START` initialization parameter is set to `true`.<br>**See Also:** *Oracle Data Guard Broker* | Database instances, Data Guard |
| D*nnn* | Dispatcher Process | Performs network communication in the shared server architecture | In the shared server architecture, clients connect to a dispatcher process, which creates a virtual circuit for each connection. When the client sends data to the server, the dispatcher receives the data into the virtual circuit and places the active circuit on the common queue to be picked up by an idle shared server. The shared server then reads the data from the virtual circuit and performs the database work necessary to complete the request. When the shared server must send data to the client, the server writes the data back into the virtual circuit and the dispatcher sends the data to the client. After the shared server completes the client request, the server releases the virtual circuit back to the dispatcher and is free to handle other clients.<br>Several initialization parameters relate to shared servers. The principal parameters are: `DISPATCHERS`, `SHARED_SERVERS`, `MAX_SHARED_SERVERS`, `LOCAL_LISTENER`, `REMOTE_LISTENER`.<br>**See Also:** *Oracle Database Concepts* | Database instances, shared servers |
| DSKM | Worker Diskmon Process | Acts as the conduit between the database, Oracle ASM instances, and the Primary Diskmon daemon to communicate information to Exadata storage | This process is active only if Exadata Storage is used. DSKM performs operations related to Exadata I/O fencing and Exadata cell failure handling. | Oracle ASM instances, Exadata |
| DW*nn* | Data Pump Worker Process | Performs Data Pump tasks as assigned by the Data Pump control job process | The Data Pump worker process is responsible for performing tasks that are assigned by the Data Pump control job process, such as the loading and unloading of metadata and data. | Database instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| EMNC | EMON Coordinator Process | Coordinates database event management and notifications | EMNC is a primary background process that coordinates event management and notification activity in the database, including Streams Event Notifications, Continuous Query Notifications, and Fast Application Notifications. | Database instances |
| E*nnn* | EMON Worker Process | Performs database event management and notifications | The database event management and notification load is distributed among the EMON worker processes. These processes work on the system notifications in parallel, offering a capability to process a larger volume of notifications, a faster response time, and a lower shared memory use for staging notifications. | Database instances |
| FBDA | Flashback Data Archiver Process | Archives historical rows for tracked tables into flashback data archives and manages archive space, organization, and retention | When a transaction that modifies a tracked table commits, FBDA stores the pre-image of the rows in the archive. FBDA maintains metadata on the current rows and tracks how much data has been archived.<br>FBDA is also responsible for automatically managing the flashback data archive for space, organization (partitioning tablespaces), and retention. FBDA also keeps track of how far the archiving of tracked transactions has progressed.<br>**See Also:** *Oracle Database Development Guide* | Database instances |
| FD*nn* | Oracle ASM Stale FD Cleanup Worker Process | Cleans up Oracle ASM stale file descriptors on foreground processes | This process cleans up Oracle ASM stale file descriptors on foreground processes if an Oracle ASM disk is globally closed. | Database and Oracle ASM instances |
| FENC | Fence Monitor Process | Processes fence requests for RDBMS instances which are using Oracle ASM instances | CSS monitors RDBMS instances which are connected to the Oracle ASM instance and constantly doing I/Os. When the RDBMS instance terminates due to a failure, all the outstanding I/O's from the RDBMS instance should be drained and any new I/O's rejected. FENC receives and processes the fence request from CSSD. | Oracle ASM instances |
| FMON | File Mapping Monitor Process | Manages mapping information for the Oracle Database file mapping interface | The `DBMS_STORAGE_MAP` package enables you to control the mapping operations. When instructed by the user, FMON builds mapping information and stores it in the SGA, refreshes the information when a change occurs, saves the information to the data dictionary, and restores it to the SGA at instance startup.<br>FMON is started by the database whenever the `FILE_MAPPING` initialization parameter is set to `true`. | Database instances, Oracle ASM instances |
| FSFP | Data Guard Broker Fast Start Failover Pinger Process | Maintains fast-start failover state between the primary and target standby databases | FSFP is created when fast-start failover is enabled. | Database instances, Data Guard |
| GCR*n* | Global Conflict Resolution Worker Process | Performs synchronous tasks on behalf of LMHB | GCR*n* processes are transient worker processes that are started and stopped as required by LMHB to perform synchronous or resource intensive tasks. | Database instances, Oracle ASM instances, Oracle RAC |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| GCW*n* | GCR Monitor processes (LMHB) | GCR(DRF) monitor processes (LMHB) | Infrastructure helper processes for LMHB. Monitors the LMHB process for stalls, and restarts if necessary. | Database instances, Oracle ASM instances, Oracle RAC instances (prior to Oracle Database 23ai only) |
| GEN0 | General Task Execution Process | Performs required tasks including SQL and DML | | Database instances, Oracle ASM instances, Oracle ASM Proxy instances |
| GMON | ASM Disk Group Monitor Process | Monitors all mounted Oracle ASM disk groups | GMON monitors all the disk groups mounted in an Oracle ASM instance and is responsible for maintaining consistent disk membership and status information. Membership changes result from adding and dropping disks, whereas disk status changes result from taking disks offline or bringing them online. | Oracle ASM instances |
| GTX*n* | Global Transaction Process | Provides transparent support for XA global transactions in an Oracle RAC environment | These processes help maintain the global information about XA global transactions throughout the cluster. Also, the processes help perform two-phase commit for global transactions anywhere in the cluster so that an Oracle RAC database behaves as a single system to the externally coordinated distributed transactions.<br>The `GLOBAL_TXN_PROCESSES` initialization parameter specifies the number of GTX*n* processes, where *n* is 0-9 or a-j. The database automatically tunes the number of these processes based on the workload of XA global transactions. You can disable these processes by setting the parameter to 0. If you try to run XA global transactions with these processes disabled, an error is returned.<br>**See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* | Database instances, Oracle RAC |
| I*nnn* | Disk and Tape I/O Worker Process | Serves as an I/O worker process spawned on behalf of DBWR, LGWR, or an RMAN backup session | I/O worker process can be configured on platforms where asynchronous I/O support is not available. These workers are started by setting the corresponding worker enable parameter in the server parameter file. The I/O workers simulate the asynchronous I/O behavior when the underlying platform does not have native support for asynchronous I/O. | Database instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| IMCO | In-Memory Coordinator | Initiates background population and repopulation of in-memory enabled objects | The IMCO background process initiates population (prepopulation) of in-memory enabled objects with priority LOW/MEDIUM/HIGH/CRITICAL. In-memory enabled objects with priority NONE will not be prepopulated but will be populated on demand via W*nnn* processes when queried. The IMCO background process can also initiate repopulation of in-memory objects.<br>Starting with Oracle Database 19c, IMXT (In-Memory External Table) segments are dropped by the IMCO background process. In previous releases, IMXT segments were dropped by foreground processes. | Database instances |
| IMR0 | Instance Membership Recovery Worker Process | Performs synchronous tasks on behalf of LMON | The IMR0 background process performs the Instance Member Recovery synchronous operations on behalf of LMON | Oracle RAC, Database instances, Oracle ASM instances |
| INSV | Data Guard Broker Instance Worker Process | Performs Data Guard broker communication among instances in an Oracle RAC environment | INSV is created when the `DG_BROKER_START` initialization parameter is set to `true`. | Database instances, Data Guard |
| IPC0 | IPC Service Background Process | Common background server for basic messaging and RDMA primitives based on IPC (Inter-process communication) methods. | IPC0 handles very high rates of incoming connect requests, as well as, completing reconfigurations to support basic messaging and RDMA primitives over several transports such as UDP, RDS, InfiniBand and RC. | Oracle RAC |
| IR*nn* | Text Index Asynchronous Maintenance | Performs deferred DML maintenance for all Text Indexes in the instance | The IR*nn* (Information Retrieval) processes perform deferred DML maintenance for all Text Indexes in the instance. They perform all stages of index sync and index optimization. These worker processes are shared across all Text indexes and index partitions for all PDBs. The processes are usually initiated as a result of a `COMMIT` after a DML on a table with a Text Index. | Database instances |
| J*nnn* | Job Queue Worker Process | Executes jobs assigned by the job coordinator | Job worker processes are created or awakened by the job coordinator when it is time for a job to be executed.<br>Job workers gather all the metadata required to run the job from the data dictionary. The worker processes start a database session as the owner of the job, execute triggers, and then execute the job. After the job is complete, the worker processes commit and then execute appropriate triggers and close the session. The worker can repeat this operation in case additional jobs need to be run. | Database instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| JP*n* | Java Patching Worker Process | Patches and updates the Java in the database classes | JP*n* patches and updates the Java in the database classes. It is only started for Oracle Real Application Clusters (Oracle RAC) databases, and one of the database instances is responsible for patching the Java in the database objects. For mulitenant container databases (CDBs), the process updates each pluggable database (PDB) individually. JP*n* is started automatically and does not require user intervention. | Oracle RAC |
| LCK*n* | Lock Process | Manages global enqueue requests and cross-instance broadcasts | The process handles all requests for resources other than data blocks. For examples, LCK*n* manages library and row cache requests.<br>Possible processes are LCK0 and LCK1. | Database instances, Oracle ASM instances, Oracle RAC |
| LDD*n* | Global Enqueue Service Daemon Helper Worker | Helps the LMD*n* processes with various tasks | LDD*n* processes are worker processes spawned on demand by LMD*n* processes. They are spawned to help the dedicated LMD*n* processes with various tasks when certain workloads start creating performance bottlenecks. These worker processes are transient as they are started on demand and they can be shutdown when no longer needed. There can be up to 36 of these worker processes (LDD0-LDDz). | Database instances, Oracle ASM instances, Oracle RAC |
| LG*nn* | Log Writer Worker | Writes redo log | On multiprocessor systems, LGWR creates worker processes to improve the performance of writing to the redo log. LGWR workers are not used when there is a SYNC standby destination.<br>Possible processes are LG00-LG99. | Database instances |
| LGWR | Log Writer Process | Writes redo entries to the online redo log | Redo log entries are generated in the redo log buffer of the system global area (SGA). LGWR writes the redo log entries sequentially into a redo log file. If the database has a multiplexed redo log, then LGWR writes the redo log entries to a group of redo log files.<br>**See Also:** *Oracle Database Concepts* and *Oracle Database Administrator's Guide* | Database instances, Oracle ASM instances |
| LM*nn* | **See** LMSn, LMnn | | | |
| LMD*n* | Global Enqueue Service Daemon Process | Manages incoming remote resource requests from other instances | LMD*n* processes enqueue resources managed under Global Enqueue Service. In particular, they process incoming enqueue request messages and control access to global enqueues. They also perform distributed deadlock detections. There can be up to 36 of these processes (LMD0-LMDz). | Database instances, Oracle ASM instances, Oracle RAC |
| LMFC | Lock Manager Flash Cache Process | For Oracle Database Appliance only, performs actions related to recovery of a terminated instance's database flash cache. | For Oracle Database Appliance only, in the event of a instance crash, the running instance will recover the crashed instance's database flash cache. The LMFC process will perform actions related to scanning the crashed instance's database flash cache and claim flash blocks mastered by the crashed instance. | Database instances, Oracle RAC |
| LMHB | Global Cache/ Enqueue Service Heartbeat Monitor | Monitor the heartbeat of several processes | LMHB monitors the CKPT, DIA*n*, LCK*n*, LG*nn*, LGWR, LMD*n*, LMON, LMS*n* , and RMS*n* processes to ensure they are running normally without blocking or spinning. | Database instances, Oracle ASM instances, Oracle RAC |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| LMON | Global Enqueue Service Monitor Process | Monitors an Oracle RAC cluster to manage global resources | LMON maintains instance membership within Oracle RAC. The process detects instance transitions and performs reconfiguration of GES and GCS resources.<br>**See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* | Database instances, Oracle ASM instances, Oracle RAC |
| LMS*n*, LM*nn* | Global Cache Service Process | Manages resources and provides resource control among Oracle RAC instances | LMS*n* and LM*nn* processes maintain a lock database for Global Cache Service (GCS) and buffer cache resources. These processes receive, process, and send GCS requests, block transfers, and other GCS-related messages. There can be up to 100 of these processes, named as follows:<br>LMS0-LMS9<br>LMSA-LMSZ<br>LM10-LM19<br>LM1A-LM1Z<br>LM20-LM29<br>LM2A-LM2R<br>**See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* | Database instances, Oracle ASM instances, Oracle RAC |
| LREG | Listener Registration Process | Registers the instance with the listeners | LREG notifies the listeners about instances, services, handlers, and endpoint. | Database instances, Oracle ASM instances, Oracle RAC |
| LSP0 | Logical Standby Coordinator Process | Schedules transactions for Data Guard SQL Apply | LSP0 is the initial process created upon startup of Data Guard SQL Apply. In addition to managing LogMiner and Apply processes, LSP0 is responsible for maintaining inter-transaction dependencies and appropriately scheduling transactions with applier processes. LSP0 is also responsible for detecting and enabling run-time parameter changes for the SQL Apply product as a whole. | Database instances, Data Guard |
| LSP1 | Logical Standby Dictionary Build Process | Performs a logical standby dictionary build on a primary database | The LSP1 process is spawned on a logical standby database that is intended to become the new primary database. A logical standby database becomes a primary database because of switchover or failover. The dictionary is necessary for logical standby databases to interpret the redo of the new primary database. | Database instances, Data Guard |
| LSP2 | Logical Standby Set Guard Process | Determines which database objects will be protected by the database guard | The LSP2 process is created as needed during startup of SQL Apply to update the list of objects that are protected by the database guard. | Database instances, Data Guard |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| L*nnn* | Pooled Server Process | Handles client requests in Database Resident Connection Pooling | In Database Resident Connection Pooling, clients connect to a connection broker process. When a connection becomes active, the connection broker hands off the connection to a compatible pooled server process. The pooled server process performs network communication directly on the client connection and processes requests until the client releases the server. After being released, the connection is returned to the broker for monitoring, leaving the server free to handle other clients.<br>**See Also:** *Oracle Database Concepts* | Database instances, Database Resident Connection Pooling |
| MARK | Mark AU for Resynchronization Coordinator Process | Marks ASM allocation units as stale following a missed write to an offline disk | MARK essentially tracks which extents require resynchronization for offline disks. This process runs in the database instance and is started when the database instance first begins using the Oracle ASM instance. If required, MARK can also be started on demand when disks go offline in the Oracle ASM redundancy disk group. | Database instances, Oracle ASM instances |
| MMAN | Memory Manager Process | Serves as the instance memory manager | This process performs the resizing of memory components on the instance. | Database instances, Oracle ASM instances |
| MMNL | Manageability Monitor Lite Process | Performs tasks relating to manageability, including active session history sampling and metrics computation | MMNL performs many tasks relating to manageability, including session history capture and metrics computation. | Database instances, Oracle ASM instances |
| MMON | Manageability Monitor Process | Performs or schedules many manageability tasks | MMON performs many tasks related to manageability, including taking Automatic Workload Repository snapshots and performing Automatic Database Diagnostic Monitor analysis. | Database instances, Oracle ASM instances |
| M*nnn* | Shared MMON Worker Process | Performs manageability tasks on behalf of MMON | The M*nnn* processes are a pool of worker processes that can be shared by multiple MZ*nn* processes.<br>See the **Long Description** for MZ*nn* in this table for more information about the MZ*nn* processes. | Database instances, Oracle ASM instances |
| MRP0 | Managed Standby Recovery Process | Coordinates the application of redo on a physical standby database | MRP0 is spawned at the start of redo apply on a physical standby database. This process handles the extraction of redo and coordinates the application of that redo on a physical standby database.<br>**See Also:** *Oracle Data Guard Concepts and Administration* | Database instances, Data Guard |
| MS*nn* | LogMiner Worker Process | Reads redo log files and translates and assembles into transactions | Multiple MS*nn* processes can exists, where *n* is 0-9 or a-Z. A minimum of three MS*nn* processes work as a group to provide transactions to a LogMiner client, for example, a logical standby database or a database capture. There may be more than one such group, for example, multiple capture processes configured for either local or downstream capture in a database. | Database instances, Logical Standby, XStream Outbound servers, Oracle GoldenGate |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| MZ*nn* | Dedicated MMON Worker Process | Performs manageability tasks on behalf of MMON | MZ*nn* is a dedicated process for a single MMON worker action. It performs manageability tasks dispatched by MMON, which include taking Automatic Workload Repository snapshots and performing Automatic Database Diagnostic Monitor analysis. | Database instances, Oracle ASM instances |
| N*nnn* | Connection Broker Process | Monitors idle connections and hands off active connections in Database Resident Connection Pooling | In Database Resident Connection Pooling, clients connect to a connection broker process. When a connection becomes active, the connection broker hands off the connection to a compatible pooled server process. The pooled server process performs network communication directly on the client connection and processes requests until the client releases the server. After being released, the connection is returned to the broker for monitoring, leaving the server free to handle other clients.<br>**See Also:** *Oracle Database Concepts* | Database instances, Database Resident Connection Pooling |
| NFS*n* | Direct NFS Dispatcher IO Worker Process | Performs direct NFS I/O for database processes | The dispatcher worker processes enable scaling of Direct NFS connections to a clustered NAS storage. These dedicated set of workers will be used to perform Direct NFS I/Os on behalf of database processes. The dispatcher processes are enabled by the `ENABLE_DNFS_DISPATCHER` initialization parameter. NFSn is spawned only if Direct NFS library is enabled for I/O to NFS servers. The number of worker processes spawned is based on the `CPU_COUNT` value.<br>**See Also:** "ENABLE_DNFS_DISPATCHER" | Database instances |
| NSS*n* | Network Server SYNC Process | Transfers redo from current online redo logs to remote standby destinations configured for SYNC transport | NSS*n* can run as multiple processes, where *n* is 1-9 or A.<br>**See Also:** *Oracle Data Guard Concepts and Administration* | Database instances, Data Guard |
| NSV*n* | Data Guard Broker NetSlave Process | Performs broker network communications between databases in a Data Guard environment | NSV*n* is created when a Data Guard broker configuration is enabled. There can be as many NSV*n* processes (where *n* is 0- 9 and A-U) created as there are databases in the Data Guard broker configuration. | Database instances, Data Guard |
| OCF*n* | ASM CF Connection Pool Process | Maintains a connection to the Oracle ASM instance for metadata operations | | Database instances, Oracle ASM instances |
| OFSD | Oracle File Server Background Process | Serves file system requests submitted to an Oracle instance | This background process listens for new file system requests, both management (like mount, unmount, and export) and I/O requests, and executes them using Oracle threads. | Database instances, Oracle RAC |
| OF*nn* | Oracle File Server Background Process Thread | Serves file system requests submitted to an Oracle instance | This is a thread for the OFSD background process<br>This background process thread is available only on Linux systems. | Database instances, Oracle RAC |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| O*nnn* | ASM Connection Pool Process | Maintains a connection to the Oracle ASM instance for metadata operations | O*nnn* worker processes are spawned on demand. These processes communicate with the Oracle ASM instance. | Database instances, Oracle ASM instances |
| PING | Interconnect Latency Measurement Process | Assesses latencies associated with communications for each pair of cluster instances | Every few seconds, the process in one instance sends messages to each instance. The message is received by PING on the target instance. The time for the round trip is measured and collected. | Database instances, Oracle ASM instances, Oracle RAC |
| PMAN | Process Manager | Manages several background processes including shared servers, pooled servers, and job queue processes | PMAN monitors, spawns, and stops the following as needed:<br>• dispatcher and shared server processes<br>• connection broker and pooled server processes for database resident connection pools<br>• job queue processes<br>• restartable background processes | Database instances, Oracle ASM instances, Oracle ASM Proxy instances |
| PMON | Process Monitor | Scans for terminated processes and coordinates cleanup | PMON periodically scans all processes to find any that have terminated in a nonstandard way. PMON is then responsible for coordinating cleanup performed by the CLMN process and the CL*nn* workers.<br>**See Also:** *Oracle Database Concepts* and *Oracle Database Net Services Administrator's Guide* | Database instances, Oracle ASM instances, Oracle ASM Proxy instances |
| P*nnn* | Parallel Query Worker Process | Perform parallel execution of a SQL statement (query, DML, or DDL) | Parallel Query has two components: a foreground process that acts as query coordinator and a set of parallel workers (P*nnn*) that are background processes. These background processes are spawned or reused during the start of a parallel statement. They receive and perform units of work sent from the query coordinator.<br>The maximum number of P*nnn* processes is controlled by the initialization parameter `PARALLEL_MAX_SERVERS`. Worker processes are numbered from 0 to the `PARALLEL_MAX_SERVERS` setting. If the query is a `GV$` query, then these background processes are numbered backward, starting from PPA7. | Database instances, Oracle ASM instances |
| PR*nn* | Parallel Recovery Process | Performs tasks assigned by the coordinator process performing parallel recovery | PR*nn* serves as a worker process for the coordinator process performing parallel media recovery and carries out tasks assigned by the coordinator. The default number of these processes is based on number of CPUs. | Database instances |
| PSP0 | Process Spawner Process | Spawns Oracle background processes after initial instance startup | | Database instances, Oracle ASM instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| PXMN | Parallel Execution Monitor | Spawns parallel server processes on local instances in an Oracle RAC environment for Query Coordinator in remote instances. | | Database instances |
| QMNC | Non-sharded queue primary process | Monitors AQ | QMNC is the non-sharded queue primary process responsible for facilitating various background activities required by AQ: time management of messages, management of nonpersistent queues, cleanup of resources, and so on. QMNC dynamically spawns Q*nnn* processes as needed for performing these tasks.<br>Note that if the `AQ_TM_PROCESSES` initialization parameter is set to 0, this process will not start. The database writes the following message to the alert log: `WARNING: AQ_TM_PROCESSES is set to 0. System might be adversely affected.` | Database instances Advanced Queueing |
| QM*nn* | AQ Primary Class Process | Per instance per AQ Primary Class Process | Each of this type of process represents a single class of work item such as AQ notification, queue monitors, and cross process. | Database instances Advanced Queueing |
| Q*nnn* | AQ Server Class Process | Per AQ Primary Class server process | Each server class process acts on behalf of an AQ primary class process. This relationship is maintained until the primary requires services of a particular service process. Once released, the server class processes are moved to a free server pool. | Database instances Advanced Queueing |
| R*nnn* | ASM Block Remap Worker Process | Remaps a block with a read error | A database instance reading from an Oracle ASM disk group can encounter an error during a read. If possible, Oracle ASM asynchronously schedules a R*nnn* worker process to remap this bad block from a mirror copy. | Oracle ASM instances |
| RBAL | ASM Rebalance Primary Process | Coordinates rebalance activity | In an Oracle ASM instance, it coordinates rebalance activity for disk groups. In a database instance, it manages Oracle ASM disk groups. | Database instances, Oracle ASM instances |
| RCBG | Result Cache Background Process | Handles result cache messages | This process is used for handling invalidation and other messages generated by server processes attached to other instances in Oracle RAC. | Database instances, Oracle RAC |
| RECO | Recoverer Process | Resolves distributed transactions that are pending because of a network or system failure in a distributed database | RECO uses the information in the pending transaction table to finalize the status of in-doubt transactions. At timed intervals, the local RECO attempts to connect to remote databases and automatically complete the commit or rollback of the local portion of any pending distributed transactions. All transactions automatically resolved by RECO are removed from the pending transaction table.<br>**See Also:** *Oracle Database Concepts* and *Oracle Database Net Services Administrator's Guide* | Database instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| RL*nn* | ResetLogs Process | Clear online redo logs when performing open resetlogs and converting to physical standby | RL*nn* processes are spawned to clear online redo logs. These workers are terminated after the online redo logs are cleared, and the session does not persist.<br>Possible processes are RL00-RL31. | Database instances |
| RM | RAT Masking Worker Process | Extracts and masks bind values from workloads like SQL tuning sets and DB Replay capture files | This background process is used with Data Masking and Real Application Testing. | Database instances |
| RMON | Rolling Migration Monitor Process | Manages the rolling migration procedure for an Oracle ASM cluster | The RMON process is spawned on demand to run the protocol for transitioning an ASM cluster in and out of rolling migration mode. | Oracle ASM instance, Oracle RAC |
| RMS*n* | Oracle RAC Management Process | Performs manageability tasks for Oracle RAC | RMS*n* performs a variety of tasks, including creating resources related to Oracle RAC when new instances are added to a cluster.<br>**See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* | Database instances, Oracle RAC |
| RP*nn* | Capture Processing Worker Process | Processes a set of workload capture files | RP*nn* are worker processes spawned by calling `DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE(capture_dir,parallel_level)`. Each worker process is assigned a set of workload capture files to process.<br>Worker processes execute in parallel without needing to communicate with each other. After each process is finished processing its assigned files, it exits and informs its parent process.<br>The number of worker processes is controlled by the `parallel_level` parameter of `DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE`. By default, `parallel_level` is null. Then, the number of worker processes is computed as follows:<br>`SELECT VALUE FROM V$PARAMETER WHERE NAME='cpu_count';`<br>When `parallel_level` is `1`, no worker processes are spawned. | Database instances |
| RPOP | Instant Recovery Repopulation Daemon | Responsible for re-creating and/or repopulating data files from snapshot files and backup files | The RPOP process is responsible for re-creating and repopulating data files from snapshots files. It works with the instant recovery feature to ensure immediate data file access. The local instance has immediate access to the remote snapshot file's data, while repopulation of the recovered primary data files happens concurrently. Any changes in the data are managed between the instance's DBW processes and RPOP to ensure the latest copy of the data is returned to the user. | Database instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| RS*nn* | Global Cache Service Remaster Process | Performs remastering for cluster reconfiguration and dynamic remastering | Each RS*nn* process is a worker process for LMS*n* to handle remastering work. They are also helper processes for LMS to handle non-critical work from global cache service.<br>The RS*nn* processes were named RMV*n* in Oracle Database 12c and earlier releases. | Database instances, Oracle RAC |
| RSM0 | Data Guard Broker Worker Process | Performs monitoring management tasks related to Data Guard on behalf of DMON | The process is created when a Data Guard broker configuration is enabled. | Database instances, Data Guard |
| RSMN | Remote Worker Monitor Process | Manages background worker process creation and communication on remote instances in Oracle RAC | This background process manages the creation of worker processes and the communication with their coordinators and peers. These background worker processes perform tasks on behalf of a coordinating process running in another cluster instance. | Database instances, Oracle RAC |
| RVWR | Recovery Writer Process | Writes flashback data to the flashback logs in the fast recovery area | RVWR writes flashback data from the flashback buffer in the SGA to the flashback logs. RVWR also creates flashback logs and performs some tasks for flashback log automatic management. | Database instances, Flashback Database |
| S*nnn* | Shared Server Process | Handles client requests in the shared server architecture | In the shared server architecture, clients connect to a dispatcher process, which creates a virtual circuit for each connection. When the client sends data to the server, the dispatcher receives the data into the virtual circuit and places the active circuit on the common queue to be picked up by an idle shared server. The shared server then reads the data from the virtual circuit and performs the database work necessary to complete the request. When the shared server must send data to the client, the server writes the data back into the virtual circuit and the dispatcher sends the data to the client. After the shared server completes the client request, the server releases the virtual circuit back to the dispatcher and is free to handle other clients.<br>Several initialization parameters relate to shared servers. The principal parameters are: `DISPATCHERS`, `SHARED_SERVERS`, `MAX_SHARED_SERVERS`, `LOCAL_LISTENER`, `REMOTE_LISTENER`.<br>**See Also:** *Oracle Database Concepts* | Database instances, shared servers |
| SA*nn* | SGA Allocator | Allocates SGA | A small fraction of SGA is allocated during instance startup. The SA*nn* process allocates the rest of SGA in small chunks. The process exits upon completion of SGA allocation.<br>Possible processes are SA00-SAzz. | Database instances |
| SCC*n* | ASM Disk Scrubbing Worker Check Process | Performs Oracle ASM disk scrubbing check operation | SCC*n* acts as a worker process for SCRB and performs the checking operations.<br>Possible processes are SCC0-SCC9. | Oracle ASM instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| SCM0 | DLM Statistics Collection and Management Worker | Collects and manages statistics related to global enqueue service (GES) and global cache service (GCS) | The DLM Statistics Collection and Management worker (SCM0) is responsible for collecting and managing the statistics related to global enqueue service (GES) and global cache service (GCS). This worker exists only if DLM statistics collection is enabled. | Database instances |
| SCMN | Statistics Collection and Management | Main thread for a multithreaded process in a threaded Oracle RAC architecture | All other threads are spawned within the processes through SCMN, based on the requests in the instance. SCMN is an idle main thread, which waits for any requests, especially spawn threads, and takes care of them, along with performing periodic maintenance operations. | Oracle RAC |
| SCRB | ASM Disk Scrubbing Primary Process | Coordinates Oracle ASM disk scrubbing operations | SCRB runs in an Oracle ASM instance and coordinates Oracle ASM disk scrubbing operations. | Oracle ASM instances |
| SCR*n* | ASM Disk Scrubbing Worker Repair Process | Performs Oracle ASM disk scrubbing repair operation | SCR*n* acts as a worker process for SCRB and performs the repairing operations.<br>Possible processes are SCR0-SCR9. | Oracle ASM instances |
| SCV*n* | ASM Disk Scrubbing Worker Verify Process | Performs Oracle ASM disk scrubbing verify operation | SCV*n* acts as a worker process for SCRB and performs the verifying operations.<br>Possible processes are SCV0-SCV9. | Oracle ASM instances |
| SMCO | Space Management Coordinator Process | Coordinates the execution of various space management tasks | This background process coordinates the execution of various space management tasks, including proactive space allocation and space reclamation. SMCO dynamically spawns worker processes (W*nnn*) to implement these tasks. | Database instances |
| SMON | System Monitor Process | Performs critical tasks such as instance recovery and terminated transaction recovery, and maintenance tasks such as temporary space reclamation, data dictionary cleanup, and undo tablespace management | SMON performs many database maintenance tasks, including the following:<br>• Reclaims space used by orphaned temporary segments<br>• Maintains the undo tablespace by onlining, offlining, and shrinking the undo segments based on undo space usage statistics<br>• Cleans up the data dictionary when it is in a transient and inconsistent state<br>• Maintains the SCN to time mapping table used to support Oracle Flashback features<br>In an Oracle RAC database, the SMON process of one instance can perform instance recovery for other instances that have failed.<br>SMON is resilient to internal and external errors raised during background activities.<br>**See Also:** *Oracle Database Concepts* | Database instances |
| SP | SPA Exec Worker | Analyzes single SQL statements sent from SQL Performance Analyzer (SPA) | Executions of SPA tasks created from a SQL tuning set use this worker to analyze the SQL statements of the SQL tuning set concurrently. | Database instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| SVCB | Service Background Process | Provides database service run-time load balancing and topology information to clients. | Every 30 seconds the process processes and publishes run-time load-balancing information and keeps the topology information current. This process is started only if Oracle Real Application Clusters (Oracle RAC) is enabled. | Oracle RAC |
| TEM*n* | ASM disk Test Error Emulation Process | Emulates I/O errors on Oracle ASM disks through named events | I/O errors can be emulated on Oracle ASM disk I/O through named events. The scope can be the process, instance, or even cluster. Optionally, a set of AUs can be chosen for error emulation. | Oracle ASM instances |
| TT*nn* | Redo Transport Worker Process | Ships redo from current online and standby redo logs to remote standby destinations configured for ASYNC transport | TT*nn* can run as multiple processes, where *nn* is 00 to ZZ.<br>**See Also:** *Oracle Data Guard Concepts and Administration* | Database instances, Data Guard |
| U*nnn* | Container process for threads | Host processes where database processes execute as threads. | U*nnn* processes are database container operating system processes where database backgrounds processes like SMON, CJQ0, and database foreground processes run. The `V$PROCESS` view lists database processes running in these container processes. These container processes are created only when the `THREADED_EXECUTION` initialization parameter is set to `TRUE`. The number of these processes vary depending on the active database processes. On a host with multiple NUMA nodes, there will be at least one U*nnn* process per NUMA node.<br>These processes are irrecoverable processes; if any of them is terminated, it will result in instance termination. These processes exit when the instance is shut down or terminated. | Database instances |
| VBG*n* | Volume Background Process | Communicates between the Oracle ASM instance and the operating system volume driver | VBG*n* handles messages originating from the volume driver in the operating system and sends them to the Oracle ASM instance.<br>VBG*n* can run as multiple processes, where *n* is 0-9. | Oracle ASM instances, Oracle ASM Proxy instances |
| VDBG | Volume Driver Process | Forwards Oracle ASM requests to perform various volume-related tasks | VDBG handles requests to lock or unlock an extent for rebalancing, volume resize, disk offline, add or drop a disk, force and dismount disk group to the Dynamic Volume Manager driver. | Oracle ASM instances, Oracle ASM Proxy instances |
| VI*nn* | Volume I/O | Route ADVM volume I/O for ASM instances on compute nodes within an Exadata | These processes handle requests for I/Os targeted at storage not locally accessible. They are used for Exadata targeted storage as well. These background processes only start when an ASM Volume is created and set up to be used. One process will start for each NUMA node on target machines. Under normal operation on non-Exadata hardware and on Exadata hardware that is not utilizing ASM volumes, these processes will not be started.<br>There can be up to 32 VI processes, and they are named sequentially from VI00 to VI31. | Oracle ASM Proxy instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| VKRM | Virtual Scheduler for Resource Manager Process | Serves as centralized scheduler for Resource Manager activity | VKRM manages the CPU scheduling for all managed Oracle processes. The process schedules managed processes in accordance with an active resource plan. | Database instances |
| VKTM | Virtual Keeper of Time Process | Provides a wall clock time and reference time for time interval measurements | VKTM acts as a time publisher for an Oracle instance. VKTM publishes two sets of time: a wall clock time using a seconds interval and a higher resolution time (which is not wall clock time) for interval measurements. The VKTM timer service centralizes time tracking and offloads multiple timer calls from other clients. | Database instances, Oracle ASM instances |
| VMB0 | Volume Membership Process | Maintains cluster membership on behalf of the Oracle ASM volume driver | This process membership in the cluster as an I/O-capable client on behalf of the Oracle ASM volume driver. | Oracle ASM instances, Oracle ASM Proxy instances |
| VOSD | Virtual Operating System Daemon | Executes time-bound Oracle database service actions | This process is spawned on instance startup and is responsible for executing system service actions critical for the database. | Database instances, Oracle ASM instances, Oracle RAC |
| VUBG | Volume drive Umbilicus Background | Relays messages between Oracle ASM instance and Oracle ASM Proxy instance that is used by ADVM (for ACFS) | | Oracle ASM instances, Oracle ASM Proxy instances |

**Table F-1 (Cont.) Background Processes**

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| W*nnn* | Space Management Worker Process | Performs various background space management tasks, including proactive space allocation and space reclamation | W*nnn* worker processes perform work on behalf of Space Management and on behalf of the Oracle Database In-Memory option.<br>• When performing work on behalf of Space Management, W*nnn* processes are worker processes dynamically spawned by SMCO to perform space management tasks in the background. These tasks include preallocating space into locally managed tablespace and SecureFiles segments based on space usage growth analysis, and reclaiming space from dropped segments. After being started, the worker acts as an autonomous agent. After it finishes task execution, it automatically picks up another task from the queue. The process terminates itself after being idle for a long time.<br>• When performing work on behalf of the Oracle Database In-Memory option, W*nnn* processes execute tasks for population or repopulation of objects that are enabled for the In-Memory column store (IM columns store), and tasks that drop in-memory segments when an object is disabled for the IM columns store.<br>For in-memory population and repopulation, both the IMCO background process and foreground processes will utilize W*nnn* workers. W*nnn* processes are utilized by the IMCO background process for prepopulation of in-memory enabled objects with priority LOW/MEDIUM/HIGH/CRITICAL, and for repopulation of in-memory objects. In-memory populate and repopulate tasks running on W*nnn* workers are also initiated from foreground processes in response to queries and DMLs that reference in-memory enabled objects. | Database instances |
| XDMG | Exadata Automation Manager | Initiates automation tasks involved in managing Exadata storage | XDMG monitors all configured Exadata cells for state changes, such as a bad disk getting replaced, and performs the required tasks for such events. Its primary tasks are to watch for when inaccessible disks and cells become accessible again, and to initiate the ASM ONLINE operation. The ONLINE operation is handled by XDWK. | Oracle ASM instances, Exadata |
| XDWK | Exadata Automation Manager | Performs automation tasks requested by XDMG | XDWK gets started when asynchronous actions such as ONLINE, DROP, and ADD an Oracle ASM disk are requested by XDMG. After a 5 minute period of inactivity, this process will shut itself down. | Oracle ASM instances, Exadata |
| X*nnn* | ASM Disk Expel Worker Process | Performs Oracle ASM post-rebalance activities | This process expels dropped disks after an Oracle ASM rebalance. | Oracle ASM instances |

# Index

## A

## B



## C

## D

## E

## F



## G

## H



## I

## J



## L

## M



## N



## O

## P

## Q



## R

## S

## T

## U

## V

## W

## X