Oracle 常见后台进程及其功能汇总

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 slave processes (BMR*n*) to perform the recovery. ABMR and BMR*n* terminate after being idle for a long time. See Also: *Oracle Database Backup and Recovery User's Guide*	Database instance
ACFS	ASM Cluster File System CSS Process	Tracks the cluster membership in CSS and informs the file system driver of membership changes	ACFS delivers CSS membership changes to the Oracle cluster file system. These membership changes are required for the file system to maintain file system consistency within the cluster.	ASM instance, Oracle RAC
ACMS	Atomic Control File to Memory Service Process	Coordinates consistent updates to a control file resource with its SGA counterpart	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 instance, Oracle RAC

		on all instances in an Oracle RAC environment		
AP*nn*	Logical Standby / Streams 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. **See Also:** *Oracle Streams Concepts and Administration*	Database instance, Data Guard, Oracle Streams
ARB*n*	ASM Rebalance Process	Rebalances data extents within an ASM disk group	Possible processes are ARB0-ARB9 and ARBA.	ASM instance
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 LGWR cannot reuse and overwrite an online redo log group until it has been archived. The database starts multiple archiver processes as needed to ensure that the archiving of filled online redo logs does not fall behind. Possible processes include ARC0-ARC9 and ARCa-ARCt. The LOG_ARCHIVE_MAX_PROCESSES initialization parameter specifies the number of ARC*n* processes that the database initially invokes. **See Also:** *Oracle Database Concepts* and *Oracle Database Administrator's Guide*	Database instance
ASMB	ASM Background Process	Communicates with the ASM instance, managing storage	ASMB runs in ASM instances when the ASMCMD cp command runs or when the database instance first starts if the server parameter file is stored in ASM. ASMB also runs with Oracle Cluster Registry on ASM.	Database and ASM instances

		and providing statistics		
AS*nn*	Logical Standby / Streams Apply Process Reader Server or Apply Server	• Computes dependencies between logical change records (LCRs) and assembles messages into transactions (Reader Server) • 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$STREAMS_APPLY_READER for information about the reader server background process. An apply server receives the transactions from the coordinator background process, and either applies database changes in LCRs or sends LCRs or messages to apply handlers. Apply servers can also enqueue a queue. If an apply server encounters an error, then it then tries to resolve the error with a user-specified conflict handler or error handler. If an apply server cannot resolve an error, then it rolls back the transaction and places the entire transaction, including all of its messages, in the error queue. When an apply server commits a completed transaction, this transaction has been applied. When an apply server places a transaction in the error queue and commits, this transaction also has been applied. Query V$STREAMS_APPLY_SERVER for information about the apply server background process. The coordinator process name is AS*nn*, where *nn* can include letters and numbers.	Database instance
BMR*n*	Automatic Block	Fetches blocks	When a process submits a block media recovery request to ABMR, it dynamically spawns	Database

	Media Recovery Slave Pool Process	from a real-time readable standby database	slave 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. **See Also:** *Oracle Database Backup and Recovery User's Guide*	instance
B*nnn*	ASM Blocking Slave Process for GMON	Performs maintenance actions on ASM disk groups	B*nnn* performs actions that require waiting for resources on behalf of GMON. GMON must be highly available and cannot wait. A B*nnn* slave is spawned when a disk is taken offline in an ASM disk group. Offline timer processing and drop of the disk are performed in this slave. Up to five process (B000 to B004) can exist depending on the load.	ASM instance
CJQ0	Job Queue Coordinator Process	Selects jobs that need to be run from the data dictionary and spawns job queue slave processes (J*nnn*) to run the jobs	CJQ0 is automatically started and stopped as needed by Oracle Scheduler. 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. **See Also:** *Oracle Database Concepts* and *Oracle Database Administrator's Guide*	Database instance
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. **See Also:** *Oracle Database Concepts*	Database and ASM instances
CP*nn*	Streams Capture	Captures database	The capture process name is CP*nn*, where *nn* can include letters and numbers. The	Database

	Process	changes from the redo log by using the infrastructure of LogMiner	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$STREAMS_CAPTURE` view for information about this background process. **See Also:** *Oracle Streams Concepts and Administration*	instance, Oracle Streams
CS*nn*	Streams Propagation Sender Process	Sends LCRs to a propagation receiver	The propagation sender process name is CS*nn*, where *nn* can include letters and numbers. In an Oracle Streams combined capture and apply optimization, the propagation sender sends LCRs directly to the propagation receiver to improve performance. The propagation receiver passes the LCRs to an apply process. Query `V$PROPAGATION_SENDER` for information about a propagation sender.	Database instance, Oracle Streams
CS*nn*	I/O Calibration Process	Issues I/Os to storage as part of storage calibration.	CS*nn* slave processes are started on execution of the`DBMS_RESOURCE_MANAGER.CALIBRATE_IO()` procedure. There is one slave process per CPU on each node of the database.	Database instance, 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. **See Also:** *Oracle Database Backup and Recovery User's Guide*	Database instance
DBRM	Database Resource Manager Process	Sets resource plans and performs other tasks related to the Database	If a resource plan is not enabled, then this process is idle. **See Also:** *Oracle Database Administrator's Guide*	Database instance

		Resource Manager		
DBW*n*	Database Writer Process	Writes modified blocks from the database buffer cache to the data files	The primary responsibility of DBW*n* is to write data blocks to disk. DBW*n* also handles checkpoints, file open synchronization, and logging of Block Written records. In many cases the blocks that DBW*n* writes are scattered throughout the disk. Thus, the writes tend to be slower than the sequential writes performed by LGWR. DBW*n* performs multiblock writes when possible to improve efficiency. The number of blocks written in a multiblock write varies by operating system. The DB_WRITER_PROCESSES initialization parameter specifies the number of DBW*n* processes (DBW0-DBW9 and DBWa-DBWz). The database selects an appropriate default setting for this parameter or adjusts a user-specified setting based on the number of CPUs and processor groups. **See Also:** *Oracle Database Concepts* and *Oracle Database Performance Tuning Guide*	Database instance
DIA0	Diagnostic Process	Detects and resolves hangs and deadlocks		ASM and Database 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.	ASM and Database instances
DM*nn*	Data Pump Master Process	Coordinates the Data Pump job tasks performed by Data Pump worker processes	The Data Pump master (control) 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 instance, Data Pump

		and handles client interactions		
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. 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`. **See Also:** *Oracle Data Guard Broker*	Database instance, 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. Several initialization parameters relate to shared servers. The principal parameters are: `DISPATCHERS`, `SHARED_SERVERS`, `MAX_SHARED_SERVERS`, `LOCAL_LISTENER`, `REMOTE_LISTEN`	Database instance, shared servers

			ER. See Also: *Oracle Database Concepts*	
DR*nn*	ASM Disk Resynchronization Slave Process	Resynchronizes the contents of an offline disk	When a disk online SQL command is issued on a disk or disks that are offline, ASM spawns DR*nn*. Depending on the load, more than one slave may be spawned.	ASM Instance
DSKM	Slave Diskmon Process	Acts as the conduit between the database, ASM instances, and the Master 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.	ASM instance, Exadata
DW*nn*	Data Pump Worker Process	Performs Data Pump tasks as assigned by the Data Pump master process	The Data Pump worker process is responsible for performing tasks that are assigned by the Data Pump master process, such as the loading and unloading of metadata and data.	Database instance
EMNC	EMON Coordinator Process	Coordinates database event management and notifications	EMNC coordinates event management and notification activity in the database, including Streams Event Notifications, Continuous Query Notifications, and Fast Application Notifications.	Database and ASM instances
E*nnn*	EMON Slave Process	Performs database event management and notifications	The database event management and notification load is distributed among the EMON slave 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 and ASM 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. FDBA maintains metadata on the current rows and tracks how much data has been archived. 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. **See Also:** *Oracle Database Advanced Application Developer's Guide*	Database and 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. FMON is started by the database whenever the `FILE_MAPPING` initialization parameter is set to `true`.	Database and 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 instance, Data Guard
GCR*n*[Foot 1]	Global Conflict Resolution Slave Process	Performs synchronous tasks on behalf of LMHB	GCR*n* processes are transient slaves that are started and stopped as required by LMHB to perform synchronous or resource intensive tasks.	Database and ASM instances, Oracle

				RAC
GEN0	General Task Execution Process	Performs required tasks including SQL and DML		Database and ASM instances
GMON	ASM Disk Group Monitor Process	Monitors all mounted ASM disk groups	GMON monitors all the disk groups mounted in an 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.	ASM instance
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. 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 process disabled, an error is returned. **See Also:** *Oracle Real Application Clusters Administration and Deployment Guide*	Database instance, Oracle RAC
I*nnn*	Disk and Tape I/O Slave Process	Serves as an I/O slave process spawned on behalf of DBWR, LGWR, or an RMAN backup session	I/O slave process can be configured on platforms where asynchronous I/O support is not available. These slaves are started by setting the corresponding slave enable parameter in the server parameter file. The I/O slaves simulate the asynchronous I/O behavior when the underlying platform does not have native support for asynchronous I/O.	Database instance
INSV	Data Guard Broker Instance Slave	Performs Data Guard broker	INSV is created when the `DG_BROKER_START` initialization parameter is set to `true`.	Database instance,

	Process	communication among instances in an Oracle RAC environment		Data Guard
Jnnn	Job Queue Slave Process	Executes jobs assigned by the job coordinator	Job slave processes are created or awakened by the job coordinator when it is time for a job to be executed. Job slaves gather all the metadata required to run the job from the data dictionary. The slave processes start a database session as the owner of the job, execute triggers, and then execute the job. After the job is complete, the slave processes commit and then execute appropriate triggers and close the session. The slave can repeat this operation in case additional jobs need to be run.	Database instance
LCK0	Instance Enqueue Background Process	Manages global enqueue requests and cross-instance broadcasts	The process handles all requests for resources other than data blocks. For examples, LCK0 manages library and row cache requests.	Database and ASM instances, Oracle RAC
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. **See Also:** *Oracle Database Concepts* and *Oracle Database Administrator's Guide*	Database and ASM instances
LMD0	Global Enqueue Service Daemon 0 Process	Manages incoming remote resource requests from other instances	LMD0 processes enqueue resources managed under Global Enqueue Service. In particular, LMD0 processes incoming enqueue request messages and controls access to global enqueues. It also performs distributed deadlock detections.	Database and ASM instances, Oracle RAC

LMHB	Global Cache/Enqueue Service Heartbeat Monitor	Monitor the heartbeat of LMON, LMD, and LMS*n* processes	LMHB monitors LMON, LMD, and LMS*n* processes to ensure they are running normally without blocking or spinning.	Database and ASM instances, Oracle RAC
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. **See Also:** *Oracle Real Application Clusters Administration and Deployment Guide*	Database and ASM instances, Oracle RAC
LMS*n*	Global Cache Service Process	Manages resources and provides resource control among Oracle RAC instances	LMS, where *n* is 0-9 or a-z, maintains a lock database for Global Cache Service (GCS) and buffer cache resources. This process receives, processes, and sends GCS requests, block transfers, and other GCS-related messages. **See Also:** *Oracle Real Application Clusters Administration and Deployment Guide*	Database and 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 runtime parameter changes for the SQL Apply product as a whole.	Database instance, 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 by means of switchover or failover. The dictionary is necessary for logical standby databases to interpret the redo of the new primary database.	Database instance, Data Guard
LSP2	Logical Standby Set Guard Process	Determines which database objects	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 instance,

		will be protected by the database guard		Data Guard
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. **See Also:** *Oracle Database Concepts*	Database instance, 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 ASM instance. If required, MARK can also be started on demand when disks go offline in the ASM redundancy disk group.	Database and ASM instances
MMAN	Memory Manager Process	Serves as the instance memory manager	This process performs the resizing of memory components on the instance.	Database and 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 and 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 and ASM instances
M*nnn*	MMON Slave Process	Performs manageability tasks on behalf of MMON	M*nnn* performs manageability tasks dispatched to them by MMON. Tasks performed include taking Automatic Workload Repository snapshots and Automatic Database Diagnostic Monitor analysis.	Database and 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. **See Also:** *Oracle Data Guard Concepts and Administration*	Database instance, 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. There may be more than one such group, for example, Downstream Capture sessions.	Database instance, Logical Standby, Oracle Streams
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. **See Also:** *Oracle Database Concepts*	Database instance, Database Resident Connection Pooling
NSA*n*	Redo Transport	Ships redo from	NSA*n* can run as multiple processes, where *n* is 1-9 or A-V.	Database

	NSA1 Process	current online redo logs to remote standby destinations configured for ASYNC transport	**See Also:** *Oracle Data Guard Concepts and Administration*	instance, Data Guard
NSS*n*	Redo Transport NSS1 Process	Acts as a slave for LGWR when SYNC transport is configured for a remote standby destination	NSS*n* can run as multiple processes, where *n* is 1-9 or A-V. **See Also:** *Oracle Data Guard Concepts and Administration*	Database instance, 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 instance, Data Guard
OCF*n*	ASM CF Connection Pool Process	Maintains a connection to the ASM instance for metadata operations		Database and ASM instances
O*nnn*	ASM Connection Pool Process	Maintains a connection to the ASM instance for metadata operations	O*nnn* slave processes are spawned on demand. These processes communicate with the ASM instance.	Database and 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 and ASM instances, Oracle RAC
PMON	Process Monitor	Monitors the other background processes and performs process recovery when a server or dispatcher process terminates abnormally	PMON periodically performs cleanup of all the following: • Processes that died abnormally • Sessions that were killed • Detached transactions that have exceeded their idle timeout • Detached network connections which have exceeded their idle timeout In addition, PMON monitors, spawns, and stops the following as needed: • Dispatcher and shared server processes • Job queue processes • Pooled server processes for database resident connection pooling • Restartable background processes PMON is also responsible for registering information about the instance and dispatcher processes with the network listener. **See Also:** *Oracle Database Concepts* and *Oracle Database Net Services Administrator's Guide*	Database and ASM instances
P*nnn*	Parallel Query Slave 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 slaves (P*nnn*) that are background processes. These background processes are spawned or reused during the start of a parallel statement. They receive and carry out units of work sent from the query coordinator.	Database and ASM instances

			The maximum number of P*nnn* processes is controlled by the initialization parameter PARALLEL_MAX_SERVERS. Slave 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 PZ99.	
PR*nn*	Parallel Recovery Process	Performs tasks assigned by the coordinator process performing parallel recovery	PR*nn* serves as a slave 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 instance
PSP0	Process Spawner Process	Spawns Oracle background processes after initial instance startup		Database and ASM instances
QMNC	AQ Coordinator Process	Monitors AQ	QMNC is responsible for facilitating various background activities required by AQ and Oracle Streams: 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. 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 instance, Advanced Queuing
Q*nnn*	AQ Server Class Process	Performs various AQ-related background task for QMNC	Q*nnn* acts as a slave process for QMNC and carry out tasks assigned by QMNC. The number of these processes is dynamically managed by QMNC based on load.	Database instance

RBAL	ASM Rebalance Master Process	Coordinates rebalance activity	In an ASM instance, it coordinates rebalance activity for disk groups. In a database instances, it manages ASM disk groups.	Database and 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 instance, 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. **See Also:** *Oracle Database Concepts* and *Oracle Database Net Services Administrator's Guide*	Database instance
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. **See Also:** *Oracle Real Application Clusters Administration and Deployment Guide*	Database instance, Oracle RAC
R*nnn*	ASM Block Remap Slave Process	Remaps a block with a read error	A database instance reading from an ASM disk group can encounter an error during a read. If possible, ASM asynchronously schedules a R*nnn*slave process to remap this bad block from a mirror copy.	ASM instance
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. 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	Database instance

			its parent process.
			
			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:
			
			`SELECT VALUE`
			`FROM V$PARAMETER`
			`WHERE NAME='cpu_count';`
			
			When `parallel_level` is `1`, no worker processes are spawned.	

RSMO	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 instance, Data Guard
RSMN	Remote Slave Monitor Process	Manages background slave process creation and communication on remote instances in Oracle RAC	This background process manages the creation of slave processes and the communication with their coordinators and peers. These background slave processes perform tasks on behalf of a coordinating process running in another cluster instance.	Database instance, 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 instance, Flashback Database

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 slave processes (W*nnn*) to implement these tasks.	Database instance
SMON	System Monitor Process	Performs critical tasks such as instance recovery and dead 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: • Creates and manages the temporary tablespace metadata • Reclaims space used by orphaned temporary segments • Maintains the undo tablespace by onlining, offlining, and shrinking the undo segments based on undo space usage statistics • Cleans up the data dictionary when it is in a transient and inconsistent state • Maintains the SCN to time mapping table used to support Oracle Flashback features In an Oracle RAC database, the SMON process of one instance can perform instance recovery for other instances that have failed. SMON is resilient to internal and external errors raised during background activities. **See Also:** *Oracle Database Concepts*	Database instance
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	Database instance, shared servers

			to handle other clients. Several initialization parameters relate to shared servers. The principal parameters are: DISPATCHERS, SHARED_SERVERS, MAX_SHARED_SERVERS, LOCAL_LISTENER, REMOTE_LISTENER. **See Also:** *Oracle Database Concepts*	
TEM*n*	ASM disk Test Error Emulation Process	Emulates I/O errors on ASM disks through named events	I/O errors can be emulated on 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.	ASM instance
VBG*n*	Volume Background Process	Communicates between the 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 ASM instance. VBG*n* can run as multiple processes, where *n* is 0-9.	ASM instance
VDBG	Volume Driver Process	Forwards 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.	ASM instance
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 instance
VKTM	Virtual Keeper of Time Process	Provides a wall clock time and	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	Database and ASM

		reference time for time interval measurements	is not wall clock time) for interval measurements. The VKTM timer service centralizes time tracking and offloads multiple timer calls from other clients.	instances
VMB0	Volume Membership Process	Maintains cluster membership on behalf of the ASM volume driver	This process membership in the cluster as an I/O-capable client on behalf of the ASM volume driver.	ASM instance
V*nnn*	ASM Volume I/O Slave Process	Initializes ASM volume contents during creation	This process is responsible for initializing the ASM volume during creation.	ASM instance
W*nnn*	Space Management Slave Process	Performs various background space management tasks, including proactive space allocation and space reclamation	W*nnn* processes are slave 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. At most 10 W*nnn* slaves can run on one database instance. After being started, the slave 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.	Database instance
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 inaccessible disks and cells and when they become accessible again, and to initiate the ASM ONLINE operation. The ONLINE operation is handled by XDWK.	ASM instance, Exadata
XDWK	Exadata Automation Manager	Performs automation tasks requested by XDMG	XDWK gets started when asynchronous actions such as ONLINE, DROP, and ADD an ASM disk are requested by XDMG. After a 5 minute period of inactivity, this process will shut itself down.	ASM instance, Exadata
X*nnn*	ASM Disk Expel Slave Process	Performs ASM post-rebalance	This process expels dropped disks at the end of an ASM rebalance.	ASM instance

		activities		

* **ACMS** (atomic controlfile to memory service) per-instance process is an agent that contributes to ensuring a distributed SGA memory update is either globally committed on success or globally aborted in the event of a failure in an Oracle RAC environment.

* **DBRM** (database resource manager) process is responsible for setting resource plans and other resource manager related tasks.

* **DIA0** (diagnosability process 0) (only 0 is currently being used) is responsible for hang detection and deadlock resolution.

* **DIAG** (diagnosability) process performs diagnostic dumps and executes global oradebug commands.
* **EMNC** (event monitor coordinator) is the background server process used for database event management and notifications.

* **FBDA** (flashback data archiver process) archives the historical rows of tracked tables into flashback data archives. Tracked tables are tables which are enabled for flashback archive. When a transaction containing DML on a tracked table commits, this process stores the pre-image of the rows into the flashback archive. It also keeps metadata on the current rows.

FBDA is also responsible for automatically managing the flashback data archive for space, organization, and retention and keeps track of how far the archiving of tracked transactions has occurred.

* **GTX0-j** (global transaction) processes provide transparent support for XA global transactions in an Oracle RAC environment. The database autotunes the number of these processes based on the workload of XA global transactions. Global transaction processes are only seen in an Oracle RAC environment.

* **KATE** performs proxy I/O to an ASM metafile when a disk goes offline.

* **MARK** marks ASM allocation units as stale following a missed write to an offline disk.

* **SMCO** (space management coordinator) process coordinates the execution of various space management related tasks, such as proactive space allocation and space reclamation. It dynamically spawns slave processes (Wnnn) to implement the task.

* **VKTM** (virtual keeper of time) is responsible for providing a wall-clock time (updated every second) and reference-time counter (updated every 20 ms and available only when running at elevated priority)

* **PZ** (PQ slaves used for global Views) are RAC Parallel Server Slave processes, but they are not normal parallel slave processes, PZnn processes (starting at 99) are used to query GV$ views which is done using Parallel Execution on all instances, if more than one PZ process is needed, then PZ98, PZ97,... (in that order) are created automatically.

* **O00** (ASM slave processes) A group of slave processes establish connections to the ASM instance. Through this connection pool database processes can send messages to the ASM instance. For example opening a file sends the open request to the ASM instance via a slave. However slaves are not used for long running operations such as creating a file. The use slave (pool) connections eliminate the overhead of logging into the ASM instance for short requests

* **x000** - Slave used to expell disks after diskgroup reconfiguration