# CA Release Automation 4.7 thru 6.x

# Oracle Server

# Database Care and Feeding Guide

Author : Keith Puzey

Version: 2.0

Filename: CA Release Automation Database Care and Feeding Guide- Oracle.docx

Date: Monday, 25 April 2016

# Table of Contents

Overview ..... 3

Database Maintenance Guidelines ..... 3

Data Purging and Maintenance Guidelines ..... 4

- Overview ..... 4
  - Purging Offline Execution Jobs ..... 5
  - Purging Audit Records ..... 6

CA Release Automation Database Schema ..... 7

- Release Automation table overview ..... 7
- Schema Overview: ..... 7
  - ACL table details ..... 10
  - Auditing table details ..... 10
  - Parameter tree table details ..... 10
  - Action Table details ..... 11
  - Process design structure details ..... 11
  - Values table details ..... 12
  - Actions and flows under component details ..... 12
  - High availability table details ..... 12
  - Event table details ..... 13
  - Release structure table details ..... 13
  - Release parameter table details ..... 14
  - Servers table details ..... 14

## Overview

This document has been written to be used by a Database DBA to purge data from the CA Release Automation database. The tables being purged are the tables containing event information and temporary auditing information. The document includes a high level overview of the Database schema for information.

## Database Maintenance Guidelines

This document has been created as a Best Practice guide for the recommended maintenance tasks of the CA Release Automation (Nolio) Database. It is recommended for the DBA to read the References listed below:

- Develop a backup and recovery plan. Regular backups are critical to the safety of the database. Recovery exercise should be performed at least once a year.
- The Oracle SGA size should be set appropriately. Typically the more memory Oracle can utilize, the better the performance. SGA size can be set by the SGA_TARGET initialization parameter. For expert tuning, you can also set the size for each memory pool.
- It is recommended to turn on tablespace auto-extended option. Set auto-extend to an appropriate size and verify there is ample disk space available. If you choose to turn off auto-extend, ensure the tablespace has sufficient free space.
- Use the following tools to monitor and diagnose the database
  - AWR (Automatic Workload Repository) – collects, processes, and maintains performance statistics for problem detection and self-tuning purposes.
  - ADDM (Automatic Database Diagnostic Monitor) – analyzes the AWR data on a regular basis, then locates the root causes of performance problems, provides recommendations for correcting any problems, and identifies non-problem areas of the system.
  - STATSPACK – a set of performance monitoring and reporting utilities. Monitor the report over time. The ongoing STATSPACK reports should show any new performance problems.

  **Note**: Only users with licensed Oracle Enterprise Manager Diagnostic Pack are entitled to use AWR and ADDM. Other users should use STATSPACK.

- Turn on Automatic SQL Tuning, if available. This is an Oracle 11g feature and requires additional license.
- The query optimizer relies on statistical information about the data so it is crucial that table and index statistics are collected and up-to-date. If you have the license, setup AWR to gather statistics regularly during off hours. Alternatively, you can use the Oracle supplied *dbms_stats* package and setup a job to collect statistics regularly.
- Rebuild indexes if needed. See MOSC note 989093.1.
- Evaluate and install security patches regularly.

# Data Purging and Maintenance Guidelines

## Overview

The sp_purge_oracle.sql script included in supplied zip file provides an Oracle package with 2 stored procedures for purging 2 types of data in the Release Automation database:

Offline execution Jobs
Audit records

It is recommended that the stored procedures be executed during off hours though a scheduled job. The two stored procedures can be executed concurrently in different database sessions. However, do not execute the same stored procedure more than once simultaneously.

**Important**

Before executing the purge procedures, execute create_FK_indexes_oracle.sql script included in supplied zip file. The script creates indexes on all foreign key columns if they do not already exist. The indexes significantly improve performance for the purge script as well as the application.

**Important**

Due to the large amount of data that could be purged, rebuilding indexes may be beneficial. See MOSC note 989093.1 for details.

## Purging Offline Execution Jobs

The stored procedure purge_execution_jobs in sp_purge package purges offline execution jobs and its associated records one day at a time starting from the oldest records.

The following tables are purged:

offline_distribution_events
offline_exec_param_links
offline_exec_servers
offline_execution_jobs
offline_flow_events
offline_manual_operations
offline_parameter_requests
offline_parameters
offline_propagation_events
offline_step_events
offline_steps
offline_user_events

Parameters:

| | |
|---|---|
| retention_days: | Number of days to retain execution job records. Default is 90 days |
| time_limit: | Number of seconds after which no command is executed. The procedure will stop when the time limit is reached and the last command (started before the time limit) is complete. The default is no time limit (null). |
| time_delay: | Number of seconds to delay between each delete command. The default is null, no delay.* |
| delete_chunk_size: | Maximum number of rows to delete per command. The default is 10000 rows. |

Example:

The following example purges jobs older than 120 days, with a time limit so that no command is executed after 3600 seconds (1 hour)

```
begin
  sp_purge.purge_execution_jobs(120, 3600);
end;
```

Note: When using this purge procedure on large databases the number of days purged within the time period will vary depending on the amount of events, as an example this stored procedure tested on a large customer database purged 5 days in one hour.

## Purging Audit Records

The stored procedure purge_execution_jobs in package sp_purge purges audit records.

The following tables are affected by the purging script:

auditingentry
revision_auditingentry
All table with _aud suffix

The following tables are excluded:

health_record_aud
health_round_samples_aud
health_sample_aud
health_threshold_aud

Parameters:

| | |
|---|---|
| time_limit: | Number of seconds after which no command is executed. The procedure will stop when the time limit is reached and the last command (started before the time limit) is complete. The default is no time limit (null). |
| time_delay: | Number of seconds to delay between each delete command. The default is null, no delay.* |
| delete_chunk_size: | Maximum number of rows to delete per command. The default is 10000 rows. |

Example:

The following example purges audit records with a time limit so that no command is executed after 3600 seconds (1 hour)

```
begin
  sp_purge.purge_audit(3600);
end;
```

*When time_delay parameter is specified, the procedure uses DBMS_LOCK.SLEEP to wait between commands. If time_delay is specified, the Nolio user must have EXECUTE privilege on SYS.DBMS_LOCK package. If time_delay is not specified or it is NULL, then the privilege is not required. Execute the following statement from a DBA user to grant the privilege.

```
grant execute on SYS.DBMS_LOCK to <nolio_owner>;
```

# CA Release Automation Database Schema

## Release Automation table overview

- All tables with “rc_” prefix belong to ROC entities
- All tables with “_aud” postfix are auditing tables that are used to track changes done in the corresponding entities (server -> servers_aud)
- All other tables are ASAP studio, management console and general system tables.

## Schema Overview:

The following table has the high level details for the Release Automation Database and the following section contains more details for some of the tables.

| Table Type | Table Name | Description |
|---|---|---|
| ACL tables | Tables with Prefix ACL_ | Acl table are used for security (spring security framework) |
| Auditing Tables | Tables with Suffix _aud | Auditing tables are used for reporting all changes in design process |
| Parameter Tree Tables | static_ parameter _folder | Represents all the built-in folders |
| | basic_parameter | Represents the parameters |
| | child_param_folders | Represents the folders nested under this folder |
| | basic_parameter | An abstract table and all relations is a concrete parameters |
| Action Tables | action_class_info | The action’s java class name |
| Values tables | Composite_value | A parameter value can contain values and parameters |
| Actions and flows under component | executable | Action and flow details |
| | pre_ executable & post_executable | Used by action loops |
| | order_link | Links between actions |
| Monitor tables | All tables with prefix health_ | Used for health monitor in the ROC |
| High availability tables | nac_nodes | Name of NAC nodes |
| | master_nac | Name of Master NAC and last heartbeat timestamp |
| Notification tables | All tables with prefix - “notification_”. | Used for notification in ASAP |
| Offline execution tables | All tables with prefix- “offline_”. | Used for reporting about execution processes |
| Event Tables | flow_events | All events related to execution process |
| | distribution_events | Events related to distribution |
| Artifacts Tables | All tables with prefix rc_artifact | Entry of artifacts in POC |

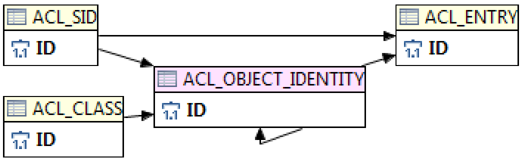
| Table Type | Table Name | Description |
|---|---|---|
| Release Structure Tables | rc_releases | Contains all releases |
|  | rc_stages | Release have two stages:<br>Init stage<br>Run Stage |
|  | rc_modules | Contains steps in release |
| Release Parameter Tables | All tables with a prefix rc_param | Release parameters |
| Template Tables | All tables with a prefix rc_template | Templates in ROC |
| Servers Tables | servers<br>server_ips<br>server_type_instance<br>access_by_jxta | Represent server and NES machines |
| Execution Tables | All tables with Prefix “execution” | All data for current execution jobs (After job is done the data is removed to the offline tables.) |
| Schedule Tables | All tables with prefix “schedule_” | ASAP Scheduling processes |
| Calendar Tables | All tables with prefix “rc_scheduled” | Calendar in ROC |

Note : Some of the DB entities have been renamed in the UI since its development. For example:

- Step -> Action
- Module -> Release Step

## ACL table details

Acl table are used for security (spring security framework)



Acl classes are system entities that require permission to access:

EnvironmentDetails, ProcessToEnvironmentAssignment, Application , TemplateOnEnvironmentProjection , EnvModuleTemplate, Module, Release , Template , TemplateModule.

Each acl entry represents a user or a group with permission to a specific entity.

## Auditing table details

Auditing tables are used for reporting all changes in design process which Include:

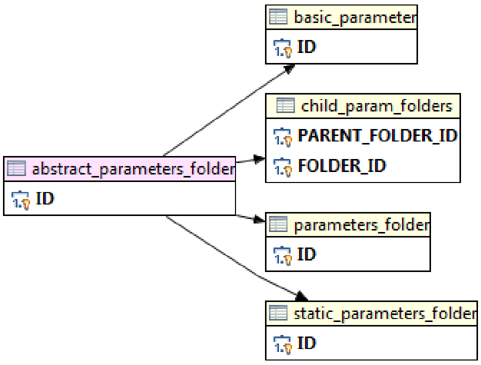
- all tables with suffix “_aud” .
- auditingentry table.
- Auditreportentry table.

The data in the “_aud” tables is generated immediately after changes done to the corresponding entity. The internal auditing service runs a scheduled task to process and transfer the data from these “aud” tables to the main audit report table – “auditreportentry”

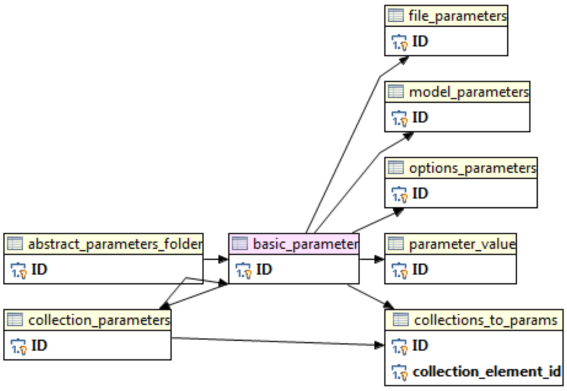
## Parameter tree table details

Parameter tree – represent the parameter tree in ASAP.

- “static_ parameter _folder” - represents all the built-in folders.
- “basic_parameter” - represents the parameter itself.
- “child_param_folders” - represents the folders nested under this folder.



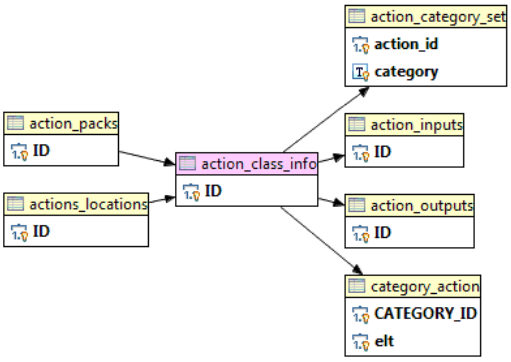
“basic_parameter” - is an abstract table and all relations is a concrete parameters.



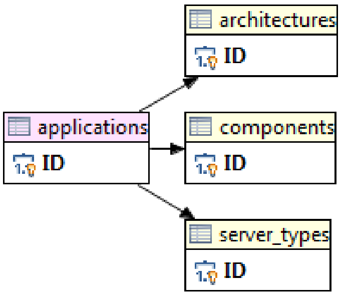
## Action Table details

These tables hold the metadata of the all the actions loaded to the system. The actions themselves are defined in java classes, with special annotations (Action name and description, inputs and outputs etc...) These tables are populated when the action packs are loaded to the system.

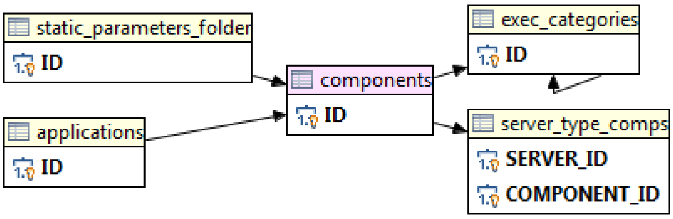
“action_class_info” –The action’s java class name.



## Process design structure details



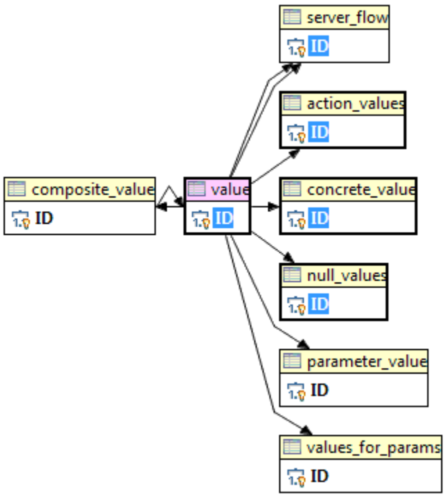
The tree is exactly like an ASAP



Components can be assigned to “server_types”

## Values table details

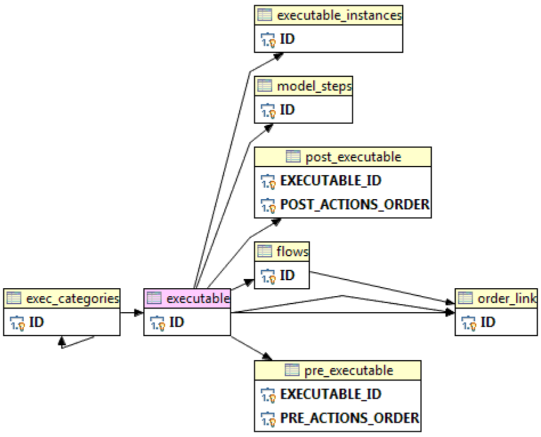
Represent type and concrete value in the parameters.



"composite_value" - is a parameter value can contain values and parameters.

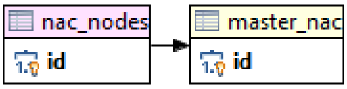
## Actions and flows under component details

| Table Name | Description |
|---|---|
| Executable | Action and flow |
| pre_ executable & post_executable | Used by action loops. |
| order_link | Links between actions. |



## High availability table details

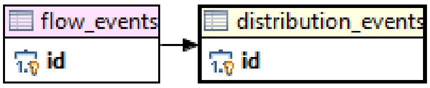
Indicates which server is active/passive.



## Event table details

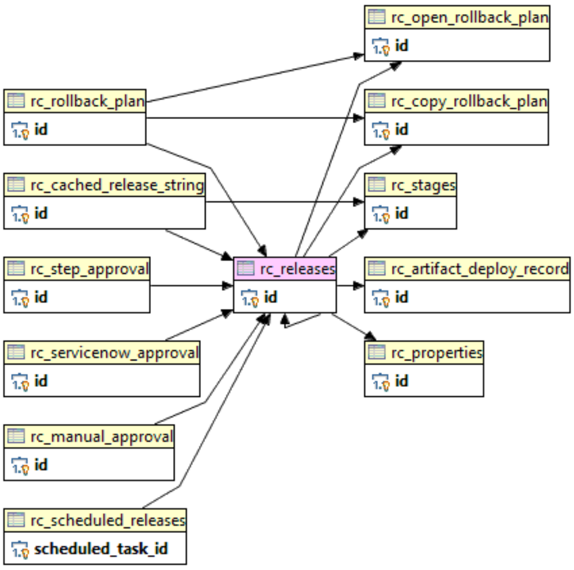
Tables used to manage the events in the application.

| Table Name | Description |
|---|---|
| Flow_events | All events related to execution process. |
| distribution_event | Events related to distribution. |



## Release structure table details

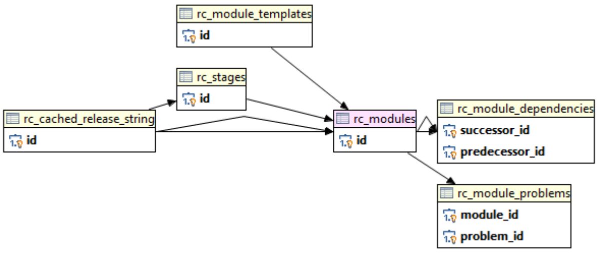
Represent structure of release in ROC



"rc_releases" - contain all releases

"rc_stages" – all releases have some stages. Currently there are two:

- Init stage - for init step.
- Run stage – for running steps.

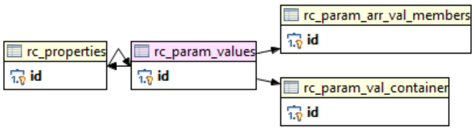


"rc_modules" – contains steps in release.

"rc_modules_template" – is an abstract for "rc_modules".

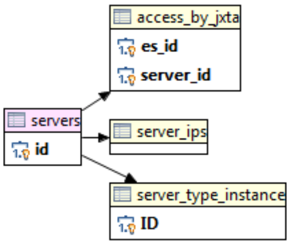
## Release parameter table details

Represent release parameters.



## Servers table details

Represent agent and NES machines


access_by_jxta
es_id
server_id
servers
id
server_ips
server_type_instance
ID