CA DataMinder

Database Guide

Release 14.6

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2014 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

CA Technologies Product References

This document references the following CA Technologies products:

- CA DataMinder™

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: About Databases

Before installing the CMS or a gateway server, your chosen database engine must already be installed and correctly configured on the target server. This guide contains guidelines for setting up your Oracle or SQL Server database engine. Later chapters provide general indexing advice and instructions for setting up database purges.

The key database requirements are:

Oracle

You must set up a database for CA DataMinder, a service name for the database, and Oracle accounts for CA DataMinder itself. You must also configure the listener and edit the sqlnet.ora file.

SQL Server

You must set up the correct method of server authentication and create logins for CA DataMinder. You also need to edit the database properties to prevent excessive use of disk space.

This section contains the following topics:

More information:

Contact CA Technologies (see page 3)

Log Files for Support

If you contact Technical Support, they may ask you to supply the following log files:

- The infrastructure log file, wgninfra.out.
- Any relevant system log files. These take the format: stderr_201001200945.log.

  Find these in CA's \data\log subfolder of the Windows All Users profile.

Supported Databases

CA DataMinder supports the following hardware and database software:

Database

CA DataMinder servers need sufficient memory and processing power to run your chosen database application. See your database documentation for details. The supported databases are:

- Oracle 10g (10.2.0.4) or 11g (11.1.0.7), 11g Release 2, or later

  Oracle 10g users may see improved search and report performance by applying the following Oracle fix 5765456:7 (see page 23). See the following My Oracle Support (formerly Oracle Metalink) notice: "Bug 5040753 Optimal index is not picked on simple query / Column group statistics."

- Microsoft SQL Server 2008
- Microsoft SQL Server 2008 R2
- Microsoft SQL Server 2008 Express Edition

  Not recommended for CMSs or FastStart base machines.

  Microsoft SQL Server 2012

Note: SQL Server is supported on Windows servers only. Verify that the SQL Server Browser service has started.

Hardware

CA DataMinder servers need sufficient memory and processing power to run your chosen database application. See your database documentation for details.

Windows servers

On all Windows CMSs and gateways, you also need Microsoft Data Access Components (MDAC) 2.6 or higher.

Configuring Your Database

Full configuration guidelines are provided for both Oracle and SQL Server databases.

Oracle

These guidelines cover configuration and performance tasks, plus details about the required Oracles users.

SQL Server

These guidelines cover configuration and performance tasks, details about the required SQL Server logins, plus SQL Server considerations when installing CA DataMinder CMSs and gateway servers.

Upgrading Your Database

Consider the following when upgrading your database:

Oracle

You must enable certain privileges (if you have not already done so) before upgrading your CA DataMinder server. These privileges permit CA DataMinder to create function indexes on key database columns.

SQL Server

After upgrading or installing a service pack, you must restart the relevant services.

For details about the issues that can arise when you upgrade your Oracle CMS or SQL Server CMS, see the *Upgrade Guide*; search the index for 'database, upgrade issues'.

Database Roles

When you install CA DataMinder on a server, the installation wizard creates the necessary user accounts and database tables for your chosen Database Engine. The wizard also creates the WGNMgmtGroupUser, WGNPolicyUser, WGNQueryUser and WGNReportingUser database roles, for both Oracle and SQL Server databases.

Note: See the requirements for Oracle roles if you use the Oracle Import and Export utilities (see page 37) to move, upgrade, or reconfigure an existing Oracle CMS database.

Security Models

Security models ensure that reviewers can only see events they are permitted to see when searching the CMS database.

You can choose which security models are available on your CMS. You can also have multiple security models active at the same time, though each reviewer is linked to a single model.

For example, some reviewers may only be permitted to see events linked to users in their own management group. Other reviewers may only be permitted to see specific types or categories of events.

CA DataMinder supports the following security models:

Management Group (Standard)

This is the default model, optimized to allow fast searching. It is based on the CA DataMinder user hierarchy.

It uses e-mail addresses (including synthesized addresses for participants in Web and Application Monitor events) to map participants to CA DataMinder users. Under this model, reviewers can only view events where at least one participant was in their management group when the event was captured.

You can also include this model in a hybrid with a Policy model (see below).

Management Group (Standard, Self-Exclude)

This model prevents reviewers from seeing their own events. As above, reviewers can only view events where at least one participant was in their management group. However, under this model the search results also exclude any events in which the 'logged-on user' (that is, the reviewer) was a participant.

You can also include this model in a hybrid with a Policy model (see below).

Management Group (Sender)

Under this model, when a reviewer runs an e-mail search, they can only view events where the e-mail sender was in their management group when the event was captured.

Important! This sender-centric security model is only appropriate for e-mail searches. Searches for other event types will return zero results.

You can also include this model in a hybrid with a Policy model (see below).

Management Group (Sender, Self-Exclude)

This model prevents reviewers from seeing their own e-mails (or any other events) when they run a search.

As above, reviewers can only view events where the e-mail sender was in their management group. However, under this model the search results also exclude any events in which the 'logged-on user' (that is, the reviewer) was a participant.

You can also include this model in a hybrid with a Policy model (see below).

Policy (Standard)

This model ensures that reviewers can only see specific types of event. For example, this model can be used to ensure that HR reviewers only see events that relate to HR issues such as employee behavior, while Legal reviewers only see events that relate to legal issues such as litigation threats or a breach of attorney client privilege.

The model is based on *policy classes*. For categorization purposes, you can associate individual triggers with a *policy class*, such as 'Employee Behavior' or 'Legal'. When a trigger fires, the policy class is stored with the associated event.

Likewise, each reviewer has a policy role. A *policy role* links a user to a collection of policy classes. In effect, the policy role determines which policy classes a user is permitted to see. When the user runs a search, the results only include events associated with these policy classes.

Before using this security model, you must define policy classes for triggers in your user policies, define your policy roles, and assign policy roles to your reviewers.

- Policy classes are described in the *Policy Guide*.
- Policy roles are described in the *Administration Guide*.

You can also include this model in a hybrid with a Management Group model (see below).

Policy (Standard, Self-Exclude)

This variant of the Policy model prevents reviewers from seeing their own events. As above, reviewers can see only specific types of event. However, the search results also exclude any events in which the reviewer was a participant

Before using this security model, you must define policy classes for triggers in your user policies, define your policy roles, and assign policy roles to your reviewers.

You can also include this model in a hybrid with a Management Group model (see below).

Policy (All Events, Restricted Triggers)

This variant of the Policy model allows reviewers to see any events in the CMS database when they run a search. That is, no events are excluded from the search results.

However, the reviewer can only see trigger and audit details for events covered by their policy role. Specifically, the Search Results screen only shows trigger and audit details for events associated with policy classes in the reviewer's policy role. If the search results include events associated with other policy classes, trigger and audit details for these events are hidden in the Search Results screen.

Before using this security model, you must define policy classes for triggers in your user policies, define your policy roles, and assign policy roles to your reviewers.

You can also include this model in a hybrid with a Management Group model (see below).

Hybrid Models: Management Group and Policy

If required, you can add a hybrid model on your CMS. This combines the Management Group and Policy models. Its effect is to restrict reviewers so they can only see specific types of event associated with users in their management group. For example, under this model a reviewer in the Legal team can only review legal events associated with members of their management group.

You can create hybrid models from any Management Group variant and any Policy variant. For example, you can create a hybrid from the 'Management Group (Self Exclude)' and the 'Policy (All Events, Restricted Triggers)' models. Here, a reviewer can only see events associated with users in their management group. But they cannot see events in which they were themself a participant and they cannot see trigger and audit details for events not covered by their policy role.

Before using this security model, you must define policy classes for triggers in your user policies, define your policy roles, and assign policy roles to your reviewers.

Unrestricted

This model is not subject to row level security (RLS). It permits reviewers to see any database items (events, users, triggers, and so on) when they run a database query. For example, Search results or reports are not restricted by policy class or the reviewer's management group. This model is required by:

- CA DataMinder administrators. Paradoxically, this security model *restricts* the extent to which administrators can edit the CMS database. Specifically, it prevents administrators from inadvertently updating the CMS database when they exercise their 'Admin: Allow Unrestricted SQL Searches' privilege.
- CA DataMinder user accounts set up explicitly for use by external reporting tools *that require full access* when searching the Data Warehouse for events.

  Note: If the user of an external reporting tool is subject to row level security, CA DataMinder applies that user's security model (typically a Management Group model) when the user runs a report.

Note: You can only assign the Unrestricted security model to a CA DataMinder user if you have the 'Admin: Disable security model filtering' administrative privilege.

Important! Certain reports and the Review Queue are not designed for use with Policy security models. See the reference below for details.

More information:

Manage Security Models (see page 15)
Policy Security Models Not Compatible With Some Reports or Review Queue (see page 16)
Security Model Limitations for Content Searches (see page 16)

Manage Security Models

Security models ensure that reviewers can only see events they are permitted to see when searching the CMS database for events.

CA DataMinder supports multiple security models, including models based on management groups, variants of this original model (for example, to prevent reviewers reviewing their own e-mails), and policy-based models. You can choose which models are active on your CMS and multiple models can be active at the same time. However, each reviewer can only be linked to a single model. For example, some reviewers may only be permitted to see events linked to users in their own management group. Other reviewers may only be to see specific types or categories of events.

When you first install CA DataMinder, only one security model is active. This is the default security model, Management Group (Standard). You must enable other security models before you can assign them to reviewers.

Important! CA DataMinder administrators require the Unrestricted security model. You *must* enable this model before you assign users accounts to the Administrator role.

You can manage security models in the Administration console.

To enable a security model on the CMS

1. Choose Tools, Manage Security Models.
2. In the Manage Security Models dialog, click Add.

   The Create Security Model dialog displays.
3. Set the Database User Name. CA DataMinder consoles use this database account to connect to the CMS database when searching for events under the new security model.
   a. Click the Set Credentials button.
   b. In the Set Model Credentials dialog, enter the name and password for a secure database account.

   Important! Each security model must use its own database account. Security models cannot share the same database account.
4. Choose the Model Type from the available types, such as Management Group (Sender).
5. (Optional) Create a hybrid security model. This combines two security models to filter the results when a reviewer searches for events.
   a. Select the Hybrid Model Type check box.
   b. Choose the second security frmo the available types.
6. Click OK.

To modify a security model

1. Choose Tools, Manage Security Models.
2. Select model you want and click Modify.

   You can now change the database user name or model type.

To remove a security model

1. Choose Tools, Manage Security Models.
2. Select model you want and click Remove.

   Any reviewers still assigned to this model will revert to the default security model. (Typically, this is the Management Group (Standard) model, although you can change the default model type.)

Policy Security Models Not Compatible With Some Reports or Review Queue

Certain reports, particularly the compliance reports such the Repeat Offender report and Compliance Audit Report, are not designed for use with Policy security models. This is also true for the Review Queue feature and the associated Reviewer search.

These reports and the Review Queue are explicitly designed to be run in conjunction with the Management Group security models. That is, they return data about users in specific user groups.

Important! We recommend that any users who need to run these reports or the Reviewer search are assigned to a Management Group security model, not to a Policy security model.

Security Model Limitations for Content Searches

Content searches only support the Management Group security model. If a reviewer has a different security model and runs a content search, the search returns zero events.

Also, unlike the Standard Search, if a reviewer has the top-level 'Users' group as their management group, a content search may return events where no participants map to CA DataMinder users. However, if a reviewer tries to view such events in the iConsole or Data Management console, CA DataMinder blocks access to these events.

Review Queue

The Review Queue (RQ) feature enables to reviewers to generate lists of events that they need to review or audit. Specifically, it provides an iConsole search that reviewers can run to retrieve all unreviewed events in their review queue. (Unreviewed events are assigned to reviewers based on their management groups.)

The RQ comprises a set of database components (such as stored procedures), plus an iConsole search and various iConsole reports. If your organization deploys the RQ, your DBAs will need to perform various depoyment tasks. The next section summarizes the required tasks.

Important! The Review Queue is not designed for use with Policy security models. See the reference below for details.

More information:

Policy Security Models Not Compatible With Some Reports or Review Queue (see page 16)

Deploy the Review Queue

The deployment procedure includes these steps:

1. **Install database components**

   The required RQ database components (such as stored procedures) are included in the server.msi installation package. They are installed automatically when you install or upgrade a CMS.

2. **Install iConsole search and reports**

   You can install the RQ search and reports when you install the iConsole Standard Searches and Reports component.

   The RQ search can be run by any reviewer to retrieve unreviewed events in their individual review queue. The RQ reports provide administrators with technical information about RQ database searches.

3. **Customize the Review Queue**

   RQ is designed to work out-of-the-box with simple defaults; no additional configuration is needed. However, in a production environment you will almost certainly want to customize the queue (for example, by changing the event selection criteria or the sampling rules).

4. **Populate the queue with events awaiting review**

   Finally, you must populate the queue with events awaiting review. Typically, you need to run or schedule a database job that calls the Populate_Queue procedure.

Details for steps 1, 3 and 4 are in the *iConsole Review Queue Configuration Guide*. Details for step 2 are in the *Platform Deployment Guide*.

Chapter 2: Oracle Guidelines

This section contains guidelines and requirements for an Oracle CMS or gateway server. It also provides details about the database accounts used by CA DataMinder, including the privileges and roles required by these Oracle users.

If you intend to use the Oracle Import and Export utilities to move, upgrade, or reconfigure an existing Oracle CMS database, check the requirements (see page 37)for these utilities before using them.

Note: To ensure optimum performance, you must also regularly maintain your database indexes. For full details about Oracle features, see your Oracle documentation.

This section contains the following topics:

Database Configuration

More information:

Oracle Database for CA DataMinder

Applicable to the CMS and gateways

Before installing the CMS or a gateway, you must create a new Oracle database for CA DataMinder. The installation wizard creates the necessary database tables when you install the CMS or gateway.

Multiple CMSs or Gateways Sharing a Database

Applicable to the CMS and gateways

If required, you can configure your CA DataMinder installation so that multiple CMSs and gateways share a single Oracle database service. To do this, you must create a separate Oracle primary user for each CMS or gateway.

Note: If running multiple CMSs and gateways on a single Oracle database service, we recommend you remove the DROP ANY ROLE privilege from all CA DataMinder users sharing that DBMS. The DBA must then manually remove the CA DataMinder roles when all instances have been installed.

More information:

Requirements for Oracle Users (see page 32)

Tablespaces for CA DataMinder

Applicable to the CMS and gateways

When you install the CMS or manually create Oracle users for CA DataMinder, you need to specify a tablespace for each Oracle user. For the primary and search users, this can be an existing, shared tablespace. But if you specify a separate schema owner, we recommend that you assign this user its own separate tablespace. This has several advantages:

- It allows precise monitoring of the CMS database, since no other products are using the tablespace.
- It permits off-line maintenance of your CA DataMinder tablespace without disrupting other products or databases that may be sharing your Oracle server.

You will reference these tablespaces when you install the CMS and if you manually create Oracle users for CA DataMinder.

Stripe Size Recommendation for RAID Devices

Applicable to CMS

To improve performance when implementing the CA DataMinder database on a RAID device, we recommend that you adjust the stripe size.

For the data files, we recommend that you set the stripe size to be a multiple of the database block size. As a minimum, the stripe size must be twice the block size.

For further details on optimizing RAID performance, please refer to your Oracle documentation.

Rollback Segments

If your rollback segments are not configured correctly, very large or complex database queries can, under certain conditions, result in an Oracle error:

```
ORA-01555 "snapshot too old (rollback segment too small)"
```

This occurs when rollback information is overwritten so that Oracle cannot roll back committed database changes to attain a sufficiently old enough version of the database block. For example, this could affect large jobs run using the CA DataMinder Event Import or Data Indexer utilities.

To prevent this error, you may need to increase the size of your existing rollback segments and create additional rollback segments. These steps allow pending database changes to be spread across more segments and reduce the risk of overwriting required rollback information. Although the exact size varies for each CA DataMinder installation, very large installations typically need rollback segments of 30Gb.

For guidelines about managing your rollback segments, see the relevant chapter in the *Oracle Administrators Guide*.

Initialization Parameters

We recommend these values for the following Oracle initialization parameters.

General Parameters

optimizer_dynamic_sampling

Set this parameter to 4.

db_file_multiblock_read_count

Set this parameter to 128.

open_cursors

Set this parameter to 500.

session_cached_cursors

Set this parameter to 100.

QUERY_REWRITE_ENABLED

Set this parameter to True.

QUERY_REWRITE_INTEGRITY

Set this parameter to Trusted.

Oracle 10g (64 bit) Parameters

The recommended settings are different for small deployments and large deployments. (A small deployment sees less 500,000 events per day; a large deployment sees over 500,000 events per day.)

Note: Do not perform final tuning of the SGA and PGA until you have executed wgn_stats and applied all fixes. By default, CA DataMinder generates poor query plans that require little PGA. The optimizer only generates good query plans that need a correspondingly larger PGA *after* wgn_stats has been deployed and all fixes have been applied (including Oracle fix 5765456:7).

sga_target

For small deployments, set this parameter to 5Gb.

For large deployments, set this parameter to 10Gb.

Note: Refer to the SGA Target Advisory in the AWR report.

pga_aggregate_target

For small deployments, set this parameter to 2Gb.

For large deployments, set this parameter to 5Gb.

Note: Refer to the PGA Target Advisory in the AWR report.

"_fix_control"

Set this parameter to:
`"_fix_control"="5764456:7"`

Note: The need to apply Oracle fix 5765456:7 is covered in the following section.

Oracle 11g (64 bit) Parameters

The recommended settings are different for small deployments and large deployments (see the Oracle 10g section above).

memory_target

For small deployments, set this parameter to 7Gb.

For large deployments, set this parameter to 15Gb.

Note: Refer to the Memory Target Advisory in the AWR report.

Oracle 10.2 Fix 5765456:7

Oracle 10g users may see improved search and report performance by applying Oracle fix 5765456:7.

For details, see the My Oracle Support (formerly Oracle Metalink) notice "Bug 5040753 Optimal index is not picked on simple query / Column group statistics."

Note: This fix is included in Oracle 10.2.0.4, but it is disabled by default.

Create Partitions

Important! Before using CA DataMinder for the first time after installation, we recommend that you set up a purging strategy, especially for your gateway servers and client machines.

For guidelines on creating and purging table partitions, see the reference below.

More information:

Set Up Partitions and Purging (see page 81)

Statistics

Statistics collection is essential. See the Database Maintenance section .

More information:

Collecting Statistics (see page 51)
Database Maintenance (see page 51)
Oracle: Implement the Wgn_Stats Package (see page 51)

Console Support for Far Eastern Characters

Applicable to the CMS and gateways

CA DataMinder consoles can display captured or imported events and user names that contain Far Eastern characters. However, for this to work correctly your Oracle database must be correctly configured to provide Unicode support.

To configure your Oracle database, set the database character set to UTF-8. Specify the character set when you set up your Oracle database for CA DataMinder, not when you create an Oracle user for CA DataMinder.

Important! If you do not set the database character set to UTF-8, the Oracle character encoding scheme cannot convert and store Unicode characters correctly. For example, individual Far Eastern characters may display as question marks in the Administration console or iConsole.

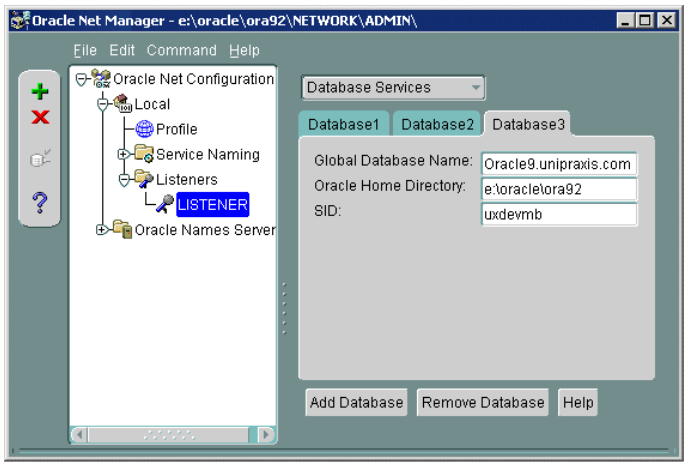
Listener Configuration

Applicable to the CMS and Gateways

To access an Oracle database across the network, you use a listener. The listener is an Oracle process that resides on the server and manages incoming connection requests from clients.

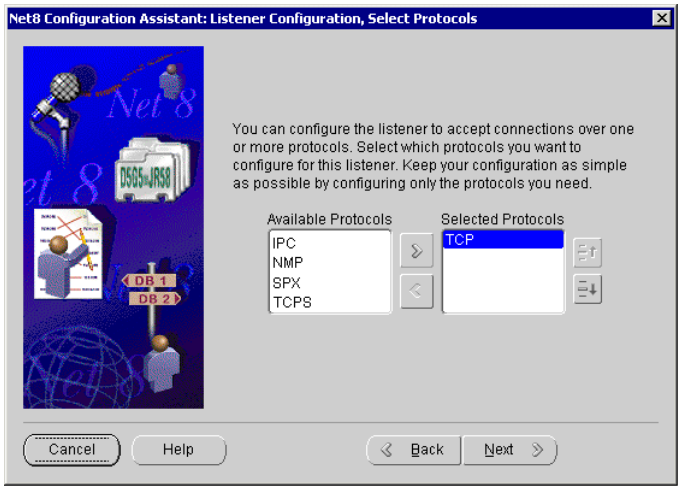
SID

Set the SID (system identifier) for the CA DataMinder database. You can do this using the Oracle Net Manager. The SID corresponds to the SID_NAME value in the listener.ora file. You must supply this SID when you install an Oracle CMS, gateway or utility machine.



Network protocol

You must configure the listener for the CA DataMinder database to use TCP. The listener uses TCP for incoming report requests from CA DataMinder. You may also need to configure the listener for any other protocols used for client-Oracle access. To specify the network protocols, use the Net Configuration Assistant.



Edit the Sqlnet.ora File

Applicable to the CMS and gateways.

Find sqlnet.ora in your Oracle installation directory. To prevent account authentication errors when you install a CA DataMinder CMS or gateway, remove or comment out the following line from this file. Do this before you install the CMS or gateway:

```
SQLNET.AUTHENTICATION_SERVICES = (NTS)
```

Oracle Users

CA DataMinder needs two Oracle accounts that it can use to access the CMS database. These are the Primary User and a Search User. If required, you can also specify additional Search Users and an account for the Schema Owner.

You can specify these users when you run the CMS installation wizard. Alternatively, you can manually create a primary user and schema owner before deploying the CMS (for example, you may want to do this as part of a native DDL script CMS installation).

These users are summarized as follows:

Schema Owner

This optional account owns the database schema. Some organizations choose to have separate accounts for the primary user and the database owner. This is typically for security reasons, for example, to ensure that employees cannot connect to the CMS database as the primary user and delete sensitive data or drop the underlying database objects.

Primary User

This is the main CA DataMinder database account. The infrastructure uses this account to access the CMS database. By default, this user also 'owns' the database schema unless a Schema Owner is specified.

Note: If a separate schema owner is specified, the primary user is also sometimes known as the 'shadow user'.

Search Users

CA DataMinder consoles use this database account when searching the CMS database for events. This is a secure account that is subject to row level security (RLS) when searching the database for events. This ensures that reviewers cannot see events that they are not permitted to see when they run a search. If multiple database security models are enabled on your CMS, specify a separate Search User database account for each security model.

You must specify a Search User when you install the CMS. This database account is automatically associated with the default database security model, Management Group (Standard). But if you enable additional security models on your CMS, each will require its own, unique Search User.

Note: 'Row level security' is a reference to event records in the relevant database tables.

Unrestricted Search User

This database account corresponds to the 'Unrestricted' security model. CA DataMinder consoles and external reporting tools can use this database account when searching the CA DataMinder Data Warehouse and CMS database for events. Unlike normal Search User database accounts, the Unrestricted Search User is *not* subject to row level security (RLS) when searching the database. If a reviewer has 'Unrestricted' security model, the reviewer can see any events when they run a search or report. Search results or reports are not restricted by policy class or the reviewer's management group.

You specify the Unrestricted Search User if you enable data warehousing when installing a CMS.

Reporting User

External reporting applications (such as BusinessObjects Enterprise) use this database account to connect to the Data Warehouse and CMS database.

You specify the Reporting User if you enable data warehousing when installing a CMS.

More information:

Manually Create a Schema Owner (see page 28)
Manually Create a Primary User (see page 29)
Manually Create a Search User (see page 30)
Manually Create an Unrestricted Search User (see page 31)
Manually Create a Reporting User (see page 32)
Requirements for Oracle Users (see page 32)
Password Expiry (see page 36)

Manually Create a Schema Owner

To manually create the Oracle schema owner before installing the CMS, you need the following files:

- CreateOracleUser.sql
- WgnSetupOwner.sql

The CreateOracleUser.sql script creates the schema owner and automatically grants the privileges that this Oracle user needs to access the database tables and manage the CMS database. To do this, the script calls and invokes a stored procedure in WgnSetupOwner.sql.

To manually create an Oracle schema owner

1. If you have not already done so, copy the following files to a local folder:
   - CreateOracleUser.sql
   - WgnSetupOwner.sql

   Find these files in the \Support folder of your CA DataMinder distribution image.
2. From a command line, change to the folder that you chose in step 1. Then run this command:
   `sqlplusw`

   This launches the Oracle SQL*Plus utility and sets the correct working directory for the script execution.
3. In SQL*Plus, connect to the CA DataMinder database as a user with sysdba privileges.
4. Run the following command:
   `@CreateOracleUser`

   The script prompts you for the following details:

   a. Enter a user name and password for the new user. You must supply these credentials when you install a CMS or gateway.

   b. Enter an existing tablespace. See also the recommendation in Tablespaces for CA DataMinder (see page 20).

   Finally, the script outputs a summary of the new account.
5. If you want to change the password or tablespace, rerun the script now.

   Important! You cannot change the password or tablespace after you have installed the CMS!

More information:

Tablespaces for CA DataMinder (see page 20)

Manually Create a Primary User

The procedure for manually creating the Oracle primary user is similar to that for creating a schema owner (see the previous section).

To manually create an Oracle primary user

1. If you have not already done so, copy the following files to a local folder:
   - CreateOracleUser.sql
   - WgnSetupOwner.sql

   Find these files in the \Support folder of your CA DataMinder distribution image.
2. From a command line, change to the folder that you chose in step 1. Then run this command:
   `sqlplusw`

   This launches the Oracle SQL*Plus utility and sets the correct working directory for the script execution.
3. In SQL*Plus, connect to the CA DataMinder database as a user with sysdba privileges.
4. Run the following command:
   `@CreateOracleUser`

   The script prompts you for the following details:

   a. Enter a user name and password for the new user. You must supply these credentials when you install a CMS or gateway.

   b. Enter an existing tablespace. See also the recommendation in Tablespaces for CA DataMinder (see page 20).

   Finally, the script outputs a summary of the new account.
5. If you want to change the password or tablespace, rerun the script now.

   Important! You cannot change the password or tablespace after you have installed the CMS!

More information:

Requirements for Oracle Users (see page 32)
Tablespaces for CA DataMinder (see page 20)

Manually Create a Search User

This section describes how to create a Search User for an Oracle CMS.

To manually create an Oracle Search User

1. If you have not already done so, copy the following files to a local folder:
   - CreateOracleRLSUser.sql
   - WgnSetupSearchUser.sql

   Find these files in the \Support folder of your CA DataMinder distribution image.
2. From a command line, change to the folder that you chose in step 1. Then run this command:
   `sqlplusw`

   This launches the Oracle SQL*Plus utility and sets the correct working directory for the script execution.
3. In SQL*Plus, connect to the CA DataMinder database as a user with sysdba privileges.
4. Run the following command:
   `@CreateOracleRLSUser`

   The script prompts you for the following details:

   a. Enter a user name and password for the new user. You must supply these credentials when you install a CMS or gateway.

   b. Enter an existing tablespace.

   Finally, the script outputs a summary of the new user account.
5. If you want to change the password or tablespace, rerun the script now or at a later time.

More information:

Requirements for Oracle Users (see page 32)
Tablespaces for CA DataMinder (see page 20)

Manually Create an Unrestricted Search User

The procedure for manually creating the Unrestricted Search User for an Oracle CMS is similar to that for creating the Search User (see the previous section).

To manually create an Oracle Unrestricted Search User

1. If you have not already done so, copy the following files to a local folder:
   - CreateOracleRLSUser.sql
   - WgnSetupSearchUser.sql

   Find these files in the \Support folder of your CA DataMinder distribution image.
2. From a command line, change to the folder that you chose in step 1. Then run this command:
   `sqlplusw`

   This launches the Oracle SQL*Plus utility and sets the correct working directory for the script execution.
3. In SQL*Plus, connect to the CA DataMinder database as a user with sysdba privileges.
4. Run the following command:
   `@CreateOracleRLSUser`

   The script prompts you for the following details:

   a. Enter a user name and password for the new user. You must supply these credentials if you enable the data warehouse when you install a CMS.

   b. Enter an existing tablespace.

   Finally, the script outputs a summary of the new user account.
5. If you want to change the password or tablespace, rerun the script now or at a later time.

Manually Create a Reporting User

The procedure for manually creating the Reporting User for an Oracle CMS is similar to that for creating the Search User (see the previous section).

To manually create an Oracle Reporting User

1. If you have not already done so, copy the following files to a local folder:
   - CreateOracleRLSUser.sql
   - WgnSetupSearchUser.sql

   Find these files in the \Support folder of your CA DataMinder distribution image.
2. From a command line, change to the folder that you chose in step 1. Then run this command:
   `sqlplusw`

   This launches the Oracle SQL*Plus utility and sets the correct working directory for the script execution.
3. In SQL*Plus, connect to the CA DataMinder database as a user with sysdba privileges.
4. Run the following command:
   `@CreateOracleRLSUser`

   The script prompts you for the following details:

   a. Enter a user name and password for the new user. You must supply these credentials if you enable the data warehouse when you install a CMS.

   b. Enter an existing tablespace.

   Finally, the script outputs a summary of the new user account.
5. If you want to change the password or tablespace, rerun the script now or at a later time.

Requirements for Oracle Users

The Oracle users for CA DataMinder require appropriate tablespaces, roles and privileges.

More information:

Tablespaces and Quotas (see page 33)
Required Privileges for Oracle Users (see page 33)
Privilege and Role Summary (see page 34)

Tablespaces and Quotas

Each Oracle user must be assigned a tablespace. You must also grant adequate tablespace to the schema owner or (if no schema owner is specified) to the primary user. By default, when you install the CMS or when you run the CreateOracleUser.sql script, this user is granted an 'Unlimited' quota. But if you subsequently want to grant a specific quota, ensure that this is sufficient to hold all your captured data.

More information:

Tablespaces for CA DataMinder (see page 20)

Required Privileges for Oracle Users

The Oracle primary user, search user and schema owner require appropriate privileges to manage the CMS database. For example, the primary user and schema owner need SELECT privileges on all data dictionary views; to allow this access, they must have the SELECT CATALOG role.

Automatically granted privileges

All required privileges are granted automatically when you install a new CMS or when you run the CreateOracleUser.sql script. But you must manually grant some privileges if upgrading your CMS.

Manually granted privileges

If upgrading from 12.0, you must grant:

- CREATE MATERIALIZED VIEW

If upgrading from 6.0, you must grant:

- CREATE JOB
- CREATE MATERIALIZED VIEW
- EXECUTE ON DBMS_LOCK
- SELECT ANY DICTIONARY

Note: These privileges are granted automatically when you install a new CMS.

More information:

Infrastructure-Based Purging (see page 57)

Privilege and Role Summary

The Oracle Primary User, Search User, and Schema Owner need the following privileges and roles:

Oracle Roles

SELECT CATALOG ROLE

Needed by the Primary User, Schema Owner, Search User, Unrestricted User and Reporting User.

Oracle System Privileges

ALTER SESSION

Needed for the Primary User, Schema Owner, Search User, Unrestricted User and Reporting /user.

CREATE CLUSTER

Needed by the Primary User and Schema Owner.

CREATE ANY CONTEXT

Needed by the Primary User and Schema Owner.

CREATE JOB

Needed by the Primary User and Schema Owner.

Grant manually if upgrading from 6.0.

CREATE MATERIALIZED VIEW

Needed by the Primary User and Schema Owner.

Grant manually if upgrading from 6.0 or r12.0.

CREATE PROCEDURE

Needed by the Primary User and Schema Owner.

CREATE ROLE

Needed by the Primary User and Schema Owner.

CREATE SEQUENCE

Needed by the Primary User and Schema Owner.

CREATE SESSION

Needed for the Primary User, Schema Owner, Search User, Unrestricted User and Reporting User.

Note: The CREATE SESSION privilege is granted directly to New Search User accounts created for CA DataMinder 14.1 or later. Search User accounts created for earlier versions of CA DLP were granted this privilege indirectly via the CONNECT ROLE.

CREATE SYNONYM

Needed for the Primary User, Search User and Schema Owner.

CREATE TABLE

Needed for the Primary User and Schema Owner.

CREATE TRIGGER

Needed for the Primary User and Schema Owner.

CREATE VIEW

Needed for the Primary User and Schema Owner.

CREATE TYPE

Needed for the Primary User and Schema Owner.

DROP ANY ROLE

Only needed by the Schema Owner for de-installation.

DROP ANY CONTEXT

Only needed by the Schema Owner for de-installation.

QUERY REWRITE

Needed for the Primary User and Schema Owner.

Needed to create function indexes.

SELECT ANY DICTIONARY

Needed for the Primary User and Schema Owner.

Grant manually if upgrading from 6.0.

Oracle Object Privilege

EXECUTE ON DBMS_LOCK

Needed by the Primary User and Schema Owner.

Grant manually if upgrading *and* you intend to partition the database or you want to run Review Queue database searches.

More information:

Infrastructure-Based Purging (see page 57)
Requirements for Oracle Import and Export Utilities (see page 37)

Password Expiry

By default, Oracle 11g enables password expiry for a default period of 180 days. To prevent unexpected suspension of the CMS when the password expires, and because CA DataMinder user accounts are not 'interactive' user accounts (that is, they are only used by the application software), we recommend that the DBAs set password expiry to a value of UNLIMITED for the CA DataMinder Oracle DBMS accounts.

Requirements for Oracle Import and Export Utilities

(Applies only to export operations in Table Mode, User Mode or Tablespace Mode. If you run export operations in Full Database Mode, the following requirements do not apply.)

Oracle 11g Release 2 (11.2) and earlier ships with Import and Export utilities. The Export utility writes data from an Oracle database into an operating system file in binary format. This file is stored outside the database and it can be read into another Oracle database using the Import utility.

If you use the Import and Export utilities to move CA DataMinder data into a receptacle database where the objects were created by the native scripts, follow these steps:

1. Drop the database sequences WGN3SEQ and Wgn3CpID before running the import operation.

   The native scripts create WGN3SEQ and Wgn3CpID. Drop these sequences before importing, otherwise these sequences are assigned default values instead of the source database values.

2. Delete the values in the WgnID database table before running the import operation.

   The native scripts populate the table WGNID with default values. Delete these values before importing, otherwise the import results in duplicate key errors and the IDs are assigned default values instead of the source database values.

3. Verify that all roles granted to CA DataMinder database accounts are enabled by default. This requirement applies to all Oracle users used by CA DataMinder and includes, for example, the Schema Owner, Primary User, and Search Users.

   a. After the import operation has completed, and before you start the CA DataMinder infrastructure, run the following command using the SQL*Plus utility:

      ```
      ALTER USER <user_name> DEFAULT ROLE ALL;
      ```

      Where <user_name> is a CA DataMinder database account.

   b. Repeat this command for each CA DataMinder database account.

   Note: The Export utility only sets roles to be enabled by default if the export runs in Full Database Mode.

For more information on Oracle database utilities, please refer to your Oracle documentation.

Chapter 3: SQL Server Guidelines

This chapter provides SQL Server configuration and performance guidelines. If you intend to use SQL Server as your CA DataMinder database engine, you must address various requirements before installing the CMS or a gateway. This chapter also provides details about the various SQL Server logins required by CA DataMinder and the requirements when installing CMSs and gateways that use a SQL Server database.

For details about SQL Server features, see your Microsoft SQL Server documentation.

Note: To ensure optimum performance, you must also regularly maintain your database indexes.

This section contains the following topics:

Database Configuration

More information:

SQL Server Database for CA DataMinder

(Applicable to the CMS and gateways)

When you install a CMS or gateway, the installation wizard creates a SQL Server database and the necessary database tables plus, optionally, the required logins. For details about installing a CMS or gateway, see the *Platform Deployment Guide*.

For CMSs and gateways running a SQL Server database, the installation wizard prompts you for:

Host Server

When you select the server hosting the database, the wizard sets the IP port and database name automatically. You can change these details, if necessary.

Primary User

This account is the main CA DataMinder database login. For SQL Server databases, the Primary User owns the schema.

Search User

The CA DataMinder consoles use this login when running event searches.

Manually Creating a Database Before Installing a CMS or Gateway

(Optional) You can manually create a database before you install the CMS or gateway:

- When you specify the database owner, choose a login that you want to use as the CA DataMinder Primary User.

  (If you do not specify a login, the database owner defaults to the current login.)
- When you install a CMS or gateway:
  - Supply the installation wizard with the name of the database that you manually created.
  - Specify the database owner login as the Primary User.

More information:

SQL Server Logins (see page 48)

Multiple CMSs and Gateways Sharing a Database

If required, you can configure your CA DataMinder installation so that multiple CMSs and gateways share a single SQL Server database server. There are two possible methods.

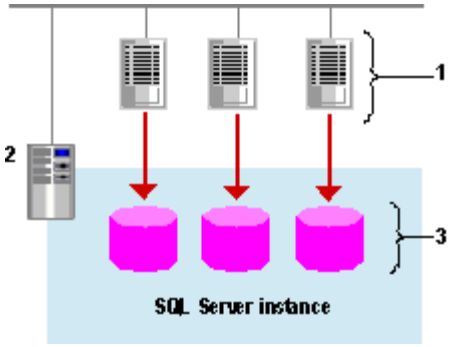
Sharing a Single SQL Server Instance

Specifically, each CMS or gateway uses a separate database hosted in the same instance of a SQL Server database engine. Each database requires a unique name. For this reason, CA DataMinder automatically assigns a unique database name when you install the CMS or a gateway.

For each SQL Server database, you will also need to create a login for CA DataMinder to use.

Example: Sharing a single SQL Server instance

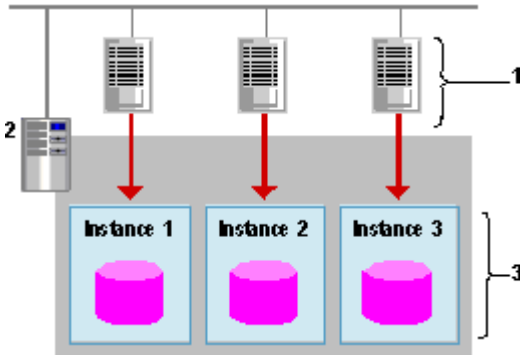
The following is an illustration where multiple CMSs or gateways share a single SQL Server instance:



1 CMSs or gateways. **2** Server hosting a single instance of SQL Server. **3** Separate SQL Server databases for each CMS or gateway.

Example: Separate instances of SQL Server for each CMS

The following is an illustration where separate instances of SQL Server instance are used by each CMS or gateway:



1 CMSs or gateways. **2** Server hosting multiple instances of SQL Server. **3** Separate SQL Server instance for each CMS or gateway.

More information:

Server Authentication (see page 45)
Network Configuration (see page 45)
Controlling Disk Space Usage (see page 43)

Multiple Instances of SQL Server

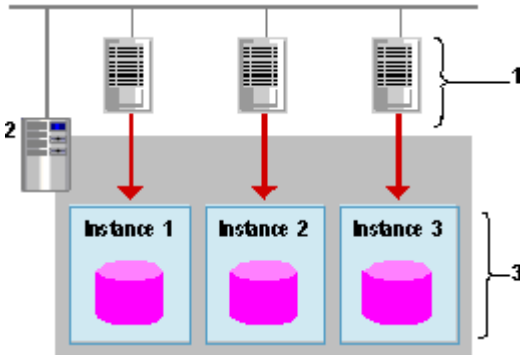
With this method each CMS or gateway uses a separate instance of SQL Server, all hosted on the same machine. You will need to create each SQL Server instance before you install the intended CMS or gateway (the installation wizard lets you choose which instance to use.

For each SQL Server instance service, you will need to create a login for CA DataMinder to use. Note also that each CMS or gateway must use a separate port number to communicate with its own instance of SQL Server.

Note: This method offers superior security where your database server supports not just CA DataMinder but other applications too. This is because it allows you to separately control access to system tables and management tasks for each instance.

Example: Separate instances of SQL Server for each CMS

The following is an illustration where separate instances of SQL Server instance are used by each CMS or gateway:



1 CMSs or gateways. **2** Server hosting multiple instances of SQL Server. **3** Separate SQL Server instance for each CMS or gateway.

More information:

Server Authentication (see page 45)
Network Configuration (see page 45)
Controlling Disk Space Usage (see page 43)

Controlling Disk Space Usage

(Applicable to the CMS and gateways)

Note: Even with the disk space configurations listed below in place, we strongly recommend that you implement your own procedures to monitor and maintain your CA DataMinder databases to ensure efficient operation.

When you create a SQL Server database, its default properties may subsequently cause it to use excessive disk space. To prevent this, you need to edit the database properties. In the SQL Server Enterprise Manager, edit the properties of the CA DataMinder database as follows:

Data Files

Because CA DataMinder databases can become very large, we strongly recommend that you review the autogrow setting for data and transaction files. If this is set too small, disk fragmentation may result; if set too large, users may experience lengthy response delays if automatic growth does occur.

The optimum growth size depends on the type of CA DataMinder server and can range from 10 MB for a gateway server generating small amounts of data (for example, policy updates) to 250 MB for an enterprise-wide CMS capturing all user data.

Transaction Log

Ensure that the database is configured to automatically grow the transaction log file. You can specify how much the log file grows at each increment: the default is 10%. For a more efficient allocation of server disk space, change this to, for example, 10 MB.

Options

In the Options tab, you need to consider these settings:

Recovery Model

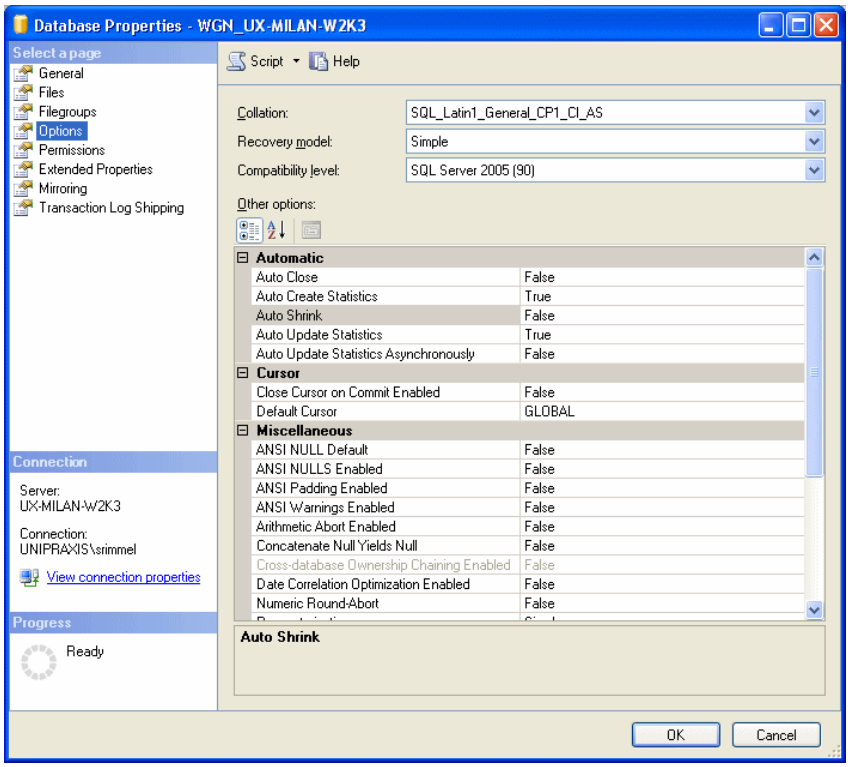
We recommend that you set this to 'Simple' to prevent the Transaction Log growing excessively (see the screenshot on the previous page). Although Simple recovery may appear to offer less protection than Full recovery, in practise this is not so. Even if you choose Full recovery, you can only restore CA DataMinder to your most recent backup anyway.

Auto Shrink

We strongly recommend that this is turned off. Your DBA must ensure that other mechanisms are put in place to monitor and control the database files.

Note: Based on experience in the field, we would expect database growth to quickly level off on gateways and Event Import servers, assuming ingest and replication rates remain consistent.

The following screen shows the Microsoft SQL Server: Database Properties dialog, Options page with the recommended settings:



More information:

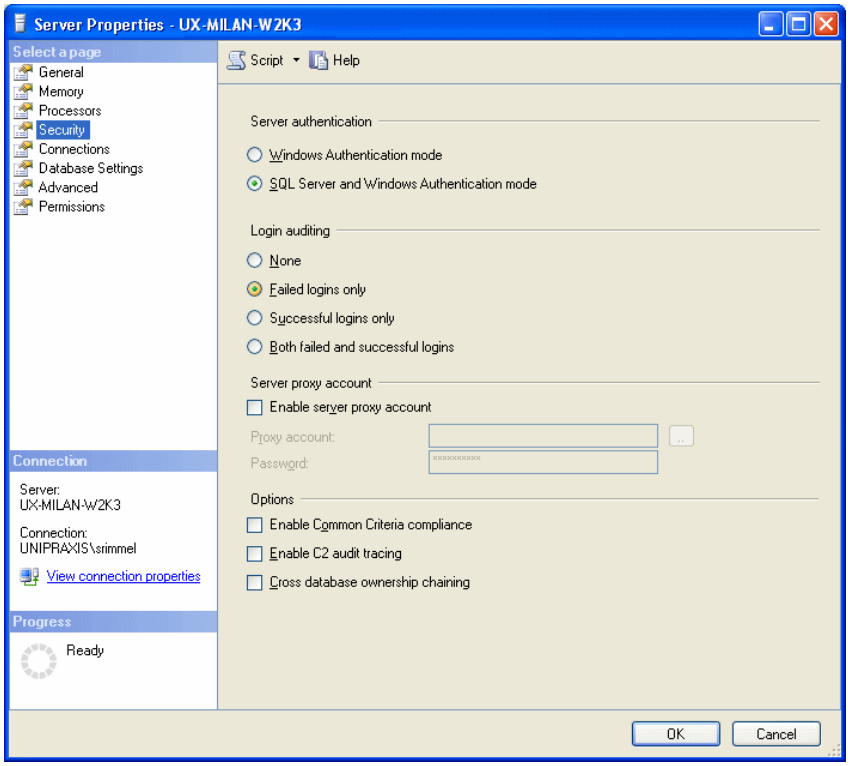
Server Authentication (see page 45)

Server Authentication

(Applicable to the CMS and gateways)

You must up set up the target CMS or gateway server to use the correct authentication method, otherwise it will be unable to connect to the SQL Server database.

In the SQL Server Enterprise Manager, edit the Security properties of the target CMS server so it uses 'SQL Server and Windows' authentication. For example, in SQL Server 2005 you do this in the Security page of the Server Properties dialog; see the screenshot below.



Network Configuration

(Applicable to the CMS and gateways)

You must ensure that SQL Server can be connected to over a network. For example, you may need to enable connections to SQL Server though TCP/IP.

SQL Server Requirements

For SQL Server 2008:

- Ensure that the SQL Server Browser service has started. By default, this service is off. We recommend that you set it to start automatically.
- You must enable the TCP/IP protocol. By default, this protocol is disabled. When you enable it, you must restart the service SQL Server (MSSQLSERVER) for this change to take effect.
- Ensure that your SQL Server database uses static IP ports (see page 46).

We recommend that you use SQL Server Configuration Manager to make these changes.

Use Static IP Ports

Your CA DataMinder servers (CMS and gateways) connect to the DBMS using static ports. If your database host server is configured to use dynamic ports, you must change this so that the SQL Server instance hosting the CA DataMinder database uses static ports.

CA DataMinder Server Requirements

After setting up your SQL Server database for CA DataMinder, note the following requirements when you install your CMS and gateway servers.

TCP/IP Port Number

When you install the CMS or a gateway server, the installation wizard prompts you for the TCP/IP port number used by SQL Server. If you have multiple instances of SQL Server running on a single machine, each must have its own port number. Make sure that you enter the port number associated with the CA DataMinder database when you run the installation wizard.

Using a Remote \Data Folder

If you want to use a SQL Server database while storing your captured data in a remote \Data folder, you must:

- Use a network-attached storage (NAS) device, or
- Use a storage area network (SAN), or
- Create an empty WGN database on the host database server before you run the installation wizard.

  Note: This is necessary because, if the WGN database does not already exist, the CA DataMinder installation wizard will create the necessary database files in the remote \Data folder, causing database access errors.

These limitations are due to a known SQL Server issue that prevents you storing your database files on a network file share. For details, see MS Knowledge Base article Q304261.

SQL Server Logins

CA DataMinder needs two SQL Server logins that it can use to access the CMS database. These are the Primary User and Search User.

You can specify and, if required, create these users when you run the CMS installation wizard. Alternatively, you can manually create a Primary User and Search User before deploying the CMS (for example, you may want to do this as part of a native DDL script CMS installation)

Primary User

This is the main CA DataMinder database account. The infrastructure uses this account to access the CMS database. For SQL Server databases, the primary user owns the schema.

Search Users

CA DataMinder consoles use this database account when searching the CMS database for events. This is a secure account that is subject to row level security (RLS) when searching the database for events. This ensures that reviewers cannot see events that they are not permitted to see when they run a search. If multiple database security models are enabled on your CMS, specify a separate Search User database account for each security model.

You must specify a Search User when you install the CMS. This database account is automatically associated with the default database security model, Management Group (Standard). But if you enable additional security models on your CMS, each will require its own, unique Search User.

Note: 'Row level security' is a reference to event records in the relevant database tables.

Unrestricted Search User

This database account corresponds to the 'Unrestricted' security model. CA DataMinder consoles and external reporting tools can use this database account when searching the CA DataMinder Data Warehouse and CMS database for events. Unlike normal Search User database accounts, the Unrestricted Search User is *not* subject to row level security (RLS) when searching the database. If a reviewer has 'Unrestricted' security model, the reviewer can see any events when they run a search or report. Search results or reports are not restricted by policy class or the reviewer's management group.

You specify the Unrestricted Search User if you enable data warehousing when installing a CMS.

Reporting User

External reporting applications (such as BusinessObjects Enterprise) use this database account to connect to the Data Warehouse and CMS database.

You specify the Reporting User if you enable data warehousing when installing a CMS.

Note: The Primary User and Search User logins must use 'SQL Server Authentication'. You specify the authentication method in the Login Properties dialog in SQL Server Enterprise Manager.

Manually Create a SQL Server Login

(Applicable to the CMS)

You can manually create the required SQL Server logins before installing the CMS. Use SQL Server Enterprise Manager to create logins for the: Primary User; Search User; Unrestricted Search User; and Reporting User.

Language Requirement

(Applies only to Data Management console users)

If you manually create a SQL Server login, you *must* set the default language to US English. Specifically, set the language name to:

```
us_english (language alias: 'English')
```

If the login uses any other language, the Data Management console fails to run because of incompatible date formats. For example, do *not* set the language to:

```
british (language alias: 'British English')
```

If you need to reset the language assigned to a login, run the following command as a system admin account such as 'sa':

```
ALTER LOGIN [WGNUSER] WITH DEFAULT_LANGUAGE=[us_english]
```

Follow these steps:

1. In Enterprise Manager, go to the Security, Logins folder of the CMS server:
2. In the General tab do the following:
   a. Enter a login name in the Name field. This login must be able to connect to the 'master' database. The CMS installation wizard will ask for this login name.
   b. Select 'SQL Server Authentication' and specify a password.
3. In the Server Roles tab, grant the dbcreator role to the new login. (SQL Server uses server roles to grant server-wide security permissions to logins.)
4. Click OK.

Restart Services after Upgrading

If you upgrade SQL Server or install a service pack, you must restart the following services in the order shown:

1. MSSQLSERVER
2. CA DataMinder Infrastructure

To restart these services, use the Services applet: find this in Administrative Tools.

Chapter 4: Database Maintenance

This section contains the following topics:

Collecting Statistics

Important! You must regularly collect statistics for all tables, columns and indexes in the CA DataMinder database. Failure to do so may significantly slow down searches for captured data.

For optimum performance, indexes require additional maintenance. The most important aspect of index maintenance is statistics. Both Oracle and SQL Server use a number of statistics, including 'column value distribution' statistics, to determine the suitability of an index for a particular database query. You **must** keep these statistics up to date to verify the query processor operates reliably. See your database documentation for information about maintaining database statistics.

Oracle: Implement the Wgn_Stats Package

For Oracle databases, you must deploy the wgn_stats statistics-gathering package. This ensures appropriate statistics are gathered for optimal performance. See also the *Database Statistics Reference Guide*.

We acknowledge that many DBAs prefer to use their own statistics collection scripts. If you do use your own script, it is essential that you understand what the wgn_stats package is designed to do. Specifically, wgn_stats:

- Gathers all the required global, partition and index statistics.
- Sets key table statistics that work around limitations in the Oracle optimizer.

  Without these statistics in place, the optimizer significantly underestimates certain key join cardinalities and produces inappropriate query plans that result in poor performance.

- Removes and locks statistics from global temporary tables.
- Generates histograms on key date columns to compensate for skew data generated by spam.

Note: The wgn_stats package only gathers statistics on standard CA DataMinder tables. If you have implemented any custom tables within the CA DataMinder schema, ensure that statistics are also gathered on these tables.

Typical Wgn_Stats Implementation

When you first deploy wgn_stats, you must execute wgn_stats to gather statistics on *all* partitions.

To gather initial statistics on all partitions

In SQL Plus, connect to the CA DataMinder database as the schema owner and execute the following command:

```
execute WGN_STATS.GATHER_ALL_STATS(
  P_def_audit_samp_pct=>50,
  P_def_non_cap_samp_pct=>100,
  P_def_cap_samp_pct=>10,
  P_num_partitions=>-1, -- All partitions
  P_degree=>8 -- or maximum available
  );
```

Recommended Weekly and Daily Statistics-gathering Configurations

You must set up daily and weekly statistics gathering. This section suggests suitable wgn_stats configurations, depending on the partitioning scheme implemented. For further information see the *Database Statistics Reference Guide*.

Daily Partitions

Daily wgn_stats call

```
execute
WGN_STATS.GATHER_ALL_STATS(
P_def_audit_samp_pct=>100,
P_def_non_cap_samp_pct=>100,
P_def_cap_samp_pct=>10,
P_num_partitions=>2, --Last 2 partitions
P_degree=>1
);
```

Weekly wgn_stats call

```
execute
WGN_STATS.GATHER_ALL_STATS(
P_def_audit_samp_pct=>100,
P_def_non_cap_samp_pct=>100,
P_def_cap_samp_pct=>10,
P_num_partitions=>7, -- Last 7 partitions
P_degree=>8 -- or maximum available
);
```

Weekly Partitions

Daily wgn_stats call

```
execute
WGN_STATS.GATHER_ALL_STATS(
P_def_audit_samp_pct=>100,
P_def_non_cap_samp_pct=>100,
P_def_cap_samp_pct=>10,
P_num_partitions=>1, --Last partition
P_degree=>1
);
```

Weekly wgn_stats call

```
execute
WGN_STATS.GATHER_ALL_STATS(
P_def_audit_samp_pct=>100,
P_def_non_cap_samp_pct=>100,
P_def_cap_samp_pct=>10,
P_num_partitions=>2, -- Last 2 partitions
P_degree=>8 -- or maximum available
);
```

Monthly or longer, or non-partitioned

Daily wgn_stats call

```
execute
WGN_STATS.GATHER_ALL_STATS(
P_def_audit_samp_pct=>100,
P_def_non_cap_samp_pct=>100,
P_def_cap_samp_pct=>10,
P_num_partitions=>1, --Last partition
P_degree=>1
);
```

Weekly wgn_stats call

```
execute
WGN_STATS.GATHER_ALL_STATS(
P_def_audit_samp_pct=>100,
P_def_non_cap_samp_pct=>100,
P_def_cap_samp_pct=>10,
P_num_partitions=>2, -- Last 2 partitions
P_degree=>8 -- or maximum available
);
```

Note: If statistics gathering takes too long for larger installations, reduce sample percentage *P_def_cap_samp_pct*.

Do not Collect Statistics for Temporary Tables

Important! For Oracle databases, do not collect statistics on the temporary tables such as TMP_Wgn3RLS.

The temporary database tables are dynamic tables, with rows continually being added or removed. Any statistics gathered on these tables are never accurate. For example, if you collect statistics for a temporary table while it is empty, the query optimizer will fail to identify the most efficient query mechanism. This can seriously undermine performance and event searches filtered by management group may be significantly slower.

Instead of collecting statistics on temporary database tables, set the following Oracle initialization parameter to **4**. This ensures that statistics are collected immediately before a query runs, allowing searches to be optimized correctly.

```
optimizer_dynamic_sampling
```

Note: Wgn_stats deletes statistics from *all* temporary tables.

SQL Server: Maintain Indexes

Your CA DataMinder database automatically includes a set of base indexes. These were chosen for their ability to speed up common queries such as captured data searches based on event, trigger or user properties. To ensure optimum performance, these indexes must be properly maintained.

Defragment the Database Indexes

For SQL Server databases, you must regularly defragment the database pages that hold index data and keep your database statistics up to date.

In particular, we strongly recommend that you defragment your indexes after running the Wgn_Queryflags_Migrate stored procedure (SP). This is an optional post-upgrade SP that runs on the CMS to optimize CA DataMinder search and report performance. After the SP has completed, some database indexes will be heavily fragmented becuase of the data updates. We strongly recommend that you run your standard index maintenance routines at the earliest convenient opportunity to remove any temporary performance degradation due to this index fragmentation.

For full details about maintaining indexes, see your SQL Server documentation.

More information:

Collecting Statistics (see page 51)

Chapter 5: Infrastructure-Based Purging

Important! Before using CA DataMinder for the first time after installation, we recommend that you set up a purging strategy, especially for your gateway servers and client machines.

This section describes the various strategies available for purging events from CA DataMinder databases. Briefly, you edit the machine policy to specify the frequency and scope of purges on an individual CA DataMinder computer. This section provides of an overview of how to configure purging. For full details, see the *Administration Guide.*

This chapter also provides an overview of how the CA DataMinder infrastructure invokes stored procedures (SPs) when purging the database. CA DataMinder provides various public purge SPs, or you can design your own custom purge SPs, including SPs to run immediately before or after a purge.

Note: This section focuses solely on purges driven by the CA DataMinder infrastructure. Partition-based purges are documented later in this guide.

This section contains the following topics:

What Data Is Purged?

Although this chapter refers to 'database purging', these infrastructure-based purges actually remove eligible database records plus any corresponding blob (Binary Large Object) files, if the blob files are stored in the \Data folder or EMC Centera. This is because each CA DataMinder event comprises metadata, written to the local CA DataMinder database, and a blob file, saved on physical media:

- **A database entry** contains the event metadata. For example, database fields specify an email's delivery date, 'envelope' details, what policy triggers were applied, and so on. Later sections in this chapter focus on purging these database entries.
- **A blob file** contains the text content of a captured file, e-mail or Web page, stored in CA DataMinder format. The blob file is written to disk and saved locally in CA's \data subfolder of the Windows All Users profile (or migrated to EMC Centera, if required).

Purging Strategies

After installing CA DataMinder, you need to turn on infrastructure-based purging. You need a separate purging strategy for your CMS, which holds captured data for your entire organization, and one or more strategies for your gateways and client machines.

More information:

Manual Purges (see page 66)

Infrastructure-Based or Partition-Based Purging?

This chapter focuses on database purges driven by the CA DataMinder infrastructure. That is, the scheduled purges are configured in the machine policy; the infrastructure then invokes the relevant database stored procedures (SPs). This contrasts with the partition-based purges described in 'Set up partition-based purging (see page 93)'.

However, a purging strategy for a partitioned database will typically require both. This is because partition-based purging will drop individual partitions but not the corresponding blob files. You must therefore schedule infrastructure-based purges to delete the residual blobs, using a custom 'no operation' SP to prevent these purges from trying to delete database records that may have been already removed by a partition-based purge.

Important! Infrastructure-based purging is **not** suitable for high volume Oracle CMS installations. You **must** use partition-based purges on these CMSs.

More information:

Partition-Based Purges (see page 63)
Set Up Partition-Based Purging (see page 93)

CMS Purges

Compliance-seeking organizations need to implement a CMS purging strategy that meets regulatory requirements on the storage and retention of historical communications. Typically, this strategy requires scheduled database purges. These run at regular intervals and are configurable in the CMS machine policy.

If you set up scheduled purging, you must also specify the minimum retention period for captured items before they are eligible for purging. For example, you may be required to retain emails for a minimum of three years.

More information:

Minimum Retention Period (see page 59)

Gateway and Client Machine Purges

Events stored in the local database of a gateway or client machine are eventually replicated up to the parent server and ultimately to the CMS. The main reason for database purging on gateways and client machines is therefore to prevent free disk space falling to dangerously low levels on these machines (with the attendant risk of the CA DataMinder infrastructure being suspended).

The simplest strategy is to implement purging after replication. This purges captured data from the local database as soon as it has been replicated to the parent server. Under this strategy, individual items of captured data are automatically excluded from purges until they have been replicated to the parent server. Only items that have already been replicated can become eligible for purging.

By default, database purging is turned off in the common client machine and common gateway policies.

Configure Purge Settings in Machine Policy

You configure purges by editing the machine policy. You can configure purges to run immediately after the data has been replicated, or you can schedule purges to run at regular intervals.

Other policy settings provide further control over purge operations. For example, you can choose to suspend the CA DataMinder infrastructure during purge operations or you can specify a purging timeout. Full details about the relevant policy settings and instructions for setting up database purging are provided in the Administration Console online help; search the index for 'purging events'.

Minimum Retention Period

CA DataMinder events become eligible for purging from the CMS when their minimum retention period expires. The retention period is measured in whole days, from midnight to midnight. For example, a 1000 day retention period implies that captured events must be retained in the CA DataMinder database for at least one thousand whole days before they can be purged.

Calculating the Age of an Event

An event's eligibility for purging therefore depends on its age. For:

- **A captured event**, that is, an email or web page captured by a CA DataMinder client or server agent, its age is calculated from the time the trigger activated.
- **Imported emails**, the age is determined by the EMail.EventDateFromEMail parameter. This specifies whether the emails capture date is set from the date in the email itself or the date when it was imported. This parameter is described in the *Platform Deployment Guide*; search for 'parameters: Event Import'.

  Note: The default retention period can be overwritten by reviewers and policy administrators—see below.

More information:

Overriding the Default Retention Period (see page 60)

Overriding the Default Retention Period

CA DataMinder permits reviewers and policy administrators to override the default minimum retention period. For example, a reviewer may need to put an unauthorized email on litigation hold. They can do this in the Audit screen of the Data Management console by specifying that email's retention period never expires. Likewise, policy administrators can set a custom retention period for all events captured by a specific trigger. For example, they may want to retain events captured by an Application Monitor trigger for one month only, but retain events captured by an email trigger for three years.

For details about overriding the default retention period, see the Administration Console online help; search for 'minimum retention period', then follow the 'triggers' sub-entries.

Purges and Stored Procedures

For both SQL Server and Oracle databases, CA DataMinder supports two types of stored procedures (SPs):

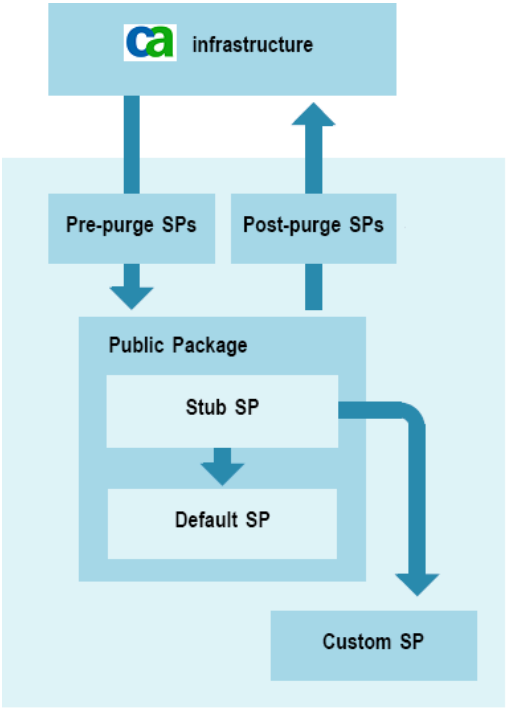
Public SPs

These are provided by CA DataMinder and provide default purging functionality. Public SPs can be overridden.

Custom SPs

These do not exist by default, but can be created by DBAs on demand.

The diagram below shows SPs are invoked when the CA DataMinder infrastructure runs a database purge:



When the CA DataMinder infrastructure runs a database purge, it does the following:

1. It checks for, and invokes, any pre-purge SPs.
2. The Stub SP checks for, and invokes, a Custom SP.
3. If no Custom SP is defined, the Stub SP invokes the Default SP containing the default purging functionality.
4. When the purge completes or terminates, the infrastructure checks for, and invokes, any post-purge SPs.

Public SPs

The public SPs provided by CA DataMinder include Stub SPs and Default SPs. The Stub SP is invoked directly by the CA DataMinder infrastructure. First, it checks for a 'well known' Custom SP, which it invokes if one exists. If no Custom SP is defined, the Stub SP invokes the Default SP.

Public SPs

The public SPs are summarized below. For details about each public SP (for example, parameter listings), refer to comments within the SP itself. Each SP can be customized, but the SP for manual event purges does not check for pre-purge or post-purge SPs. Note also the SP naming requirements.

PUB_WGN_PURGE_STUB_1

Performs policy-determined purges.

Customizable.

Pre-purge and post-purge SPs supported.

PUB_WGN_MAN_PURGE_STUB_1

Performs policy-determined purges.

Customizable.

Pre-purge and post-purge SPs are **not** supported.

Oracle Public SPs

For Oracle databases, the public Stub and Default SPs are implemented as a single 'package'. The package names are:

PUB_WGN_PURGE

Package for policy-determined purges

PUB_WGN_MAN_PURGE

Package for manually-specified purges

More information:

SP Names (see page 64)

Custom SPs

To override the Default SP, you can implement a Custom SP, with appropriate parameters, in the CA DataMinder database schema. The Custom SP is then invoked by the public Stub SP.

Important! For Oracle databases, do not implement the Custom SP in the 'package' containing the Default SP, otherwise your Custom SP will be overwritten during any future upgrade.

More information:

SP Names (see page 64)

Partition-Based Purges

If you implement database partitions, you will typically need to run partition-based purges. But unlike the infrastructure-based purges described in this chapter, partition-based purges may not automatically delete the corresponding blob files.

Specifically, if you have separate database accounts for the primary user and schema owner, blob files for expired events are **not** automatically deleted. Therefore, to ensure that blobs *are* correctly deleted, you can either use a custom SP to invoke partition-based purges when an infrastructure-based purge runs, or you can use a third party scheduler. The recommended procedure depends on whether the primary user also owns the database schema:

Primary user also owns the schema

You need to implement a custom SP to execute the wgn_roll_partitions procedure.

Separate database accounts for primary user and schema owner

We recommend you use a third-party scheduler to run each step in the purge procedure.

If using a scheduler is not practicable, you need to implement a 'no operation' custom SP—this will prevent infrastructure-based purges from trying to delete the corresponding event records in the CMS database. This is necessary because these records may have been already removed by a partition-based purge.

More information:

Link to a Scheduled Infrastructure-based Purge Using a Custom SP (see page 100)
Using a Third-party Scheduler (see page 100)

SP Names

SP names must adhere to these rules:

Stub SPs

Names must follow this pattern:
`PUB_WGN_<NAME>_STUB_<INTERFACE #>`

For example: PUB_WGN_PURGE_STUB_1

Default SPs

Names must follow this pattern:
`PUB_WGN_<NAME>_DEF_<INTERFACE #>`

For example: PUB_WGN_PURGE_DEF_1

Custom SPs

Names for custom SPs that override default SPs must follow this pattern:
`PUB_WGN_<NAME>_CUSTOM_<INTERFACE #>`

For example: PUB_WGN_PURGE_CUSTOM_1

Pre-purge SPs

When a purge runs, CA DataMinder checks for the following pre-purge SP. If found, this SP is invoked before the purge:
`WGN_PRE_PURGE_ACTION`

Post-purge SPs

When a purge completes or terminates, CA DataMinder checks for the following post-purge SPs. If found, this SP is invoked:
`WGN_POST_PURGE_ACTION`

Note: SP name checking is case sensitive.

Pre-Purge and Post-Purge SPs

CA DataMinder does not provide pre-purge or post-purge SPs, but you can implement your own custom SPs and the public Stub SP provided by CA DataMinder will support them. When a purge runs (as defined in the machine policy), CA DataMinder first checks for pre-purge and post-purge SPs.

More information:

SP Names (see page 64)

What Tasks Are Needed?

The two principal factors that determine which pre-purge and post-purge tasks to run are the total database size and the size of the purge (that is, the total amount of data that you expect to be purged). The actual range of required tasks will vary from organization to organization.

Temporarily Suspend the Infrastructure

You must suspend the CA DataMinder infrastructure during purges to guarantee that all replication activity is temporarily stopped. For example, if you remove foreign key constraints during purges, suspending the infrastructure during purges prevents new data being written to the database while these constraints are removed.

Optional Tasks

The following list includes other typical tasks that you may want to consider:

Statistics

You may want to recompile the database statistics. Will the purge perform better if the statistics are calculated beforehand? What effect will it have on the statistics used by the query optimizer?

Indexes

You may want to rebuild or defragment the indexes. Will the purge perform better if the indexes are defragmented beforehand? What effect will deleting this amount of data have on the indexes?

Temporarily remove indexes and key constraints

You may want to remove indexes and key constraints while the purge runs and reinstate them afterwards. If deleting significant proportion of the data, would it be more efficient to 'drop' non-essential indexes and foreign key constraints during the purge process?

Compaction

Shrinking or compacting the database.

Reclaiming freed space

You may want to reclaim DBMS space freed by the purge. Does this newly free space need to be released to the file system immediately?

Manual Purges

Note: By default, CA DataMinder does not support manual purges. To amend your CA DataMinder license to support manual purges, please contact Technical Support.

If permitted by your CA DataMinder license, you can manually purge all eligible events from the CMS database. The command below removes all event data (both database records and blob files) from the local server. When the purge completes, an entry is written to the CA DataMinder activity log, indicating how many events were deleted. The command syntax is:

```
wgninfra -exec wigan/schema/Schema RemoveAppData <password> [parameters]
```

Where:

<password>

Is the password for the database account used by CA DataMinder to access the CMS database.

[parameters]

Are optional date parameters:

Before=<Date>

Purges all events captured or imported into the CMS before the specified date. Events captured before midnight at the start of the specified day are purged. The <date> format is yyyy-mm-dd.

For example, this parameter purges events captured before 23:59:59 on January 17 2012:

```
Before=2012-01-18
```

After=<Date>

Purges all events captured or imported into the CMS after the specified date. Events captured after midnight at the end of the specified day are purged. The <date> format is yyyy-mm-dd.

For example, this parameter purges events captured after 23:59:59 on January 18 2012:

```
After=2012-01-18
```

Note: In the above command, the command elements wigan/schema/Schema and RemoveAppData *are* case-sensitive.

More information:

Log Files for Support (see page 9)

Chapter 6: Database Storage and Partitioning

The following sections describe how to implement a custom database storage and (for Oracle only) partitioning and purging scheme based on template SQL scripts. For Oracle databases, these scripts can also be customized to separate the 'primary user' and 'schema owner' database accounts.

Notes

- At the time of writing, partitioning support is not available for SQL Server.
- Scheduled event purges driven by the CA DataMinder infrastructure are described in the previous chapter.
- If you intend to use the Oracle Import and Export utilities to move, upgrade, or reconfigure an existing Oracle CMS database, refer to Requirements (see page 37) section before using these utilities.

This section contains the following topics:

Overview

Implementing a database storage scheme based on multiple tablespaces or file groups provides various benefits. For example, you can optimize input and output operations, while backup and maintenance operations are easier to manage.

The CA DataMinder database schema also provides support for Oracle database partitioning based on event time stamps. Partitioning benefits include faster queries, improved database manageability and availability. For example, this allows DBAs to quickly purge partitions containing data that no longer needs to be retained without incurring the cost of row-level operations.

Which Tables Are Partitioned and Purged?

The default partitioning and purging scheme supported by CA DataMinder creates partitions in: the primary event table plus all related tables in the CMS database; and the associated tables in the data warehouse. These tables are all partitioned by event timestamp using exactly the same partition boundaries. Partition-based purges affect the same tables.

The tables affected by partitioning and purging are:

CMS

- Wgn3Event
- Wgn3EA
- Wgn3EventAudit
- Wgn3EventIssue
- Wgn3EventIssueParticipant
- Wgn3EventParticipant
- Wgn3IssueTrigger
- Wgn3Trigger
- Wgn3Transaction
- WgnUEexport

Data Warehouse

- WgnDW_Audit_Curr_Fact
- WgnDW_Audit_Tran_Fact
- WgnDW_Event_Audit_Status_Fact
- WgnDW_Event_Fact
- WgnDW_Event_Participant_Fact
- WgnDW_Trigger_Fact

In some tables, the volume of accumulated data may not be large enough to warrant a purge operation, but the specified partition is purged anyway to maintain consistency with other partitioned tables.

Partitioning and Purging Features

Applicable to Oracle databases only

CA DataMinder support for database partitioning includes the following benefits:

Clear separation between core and customizable partitioning and purging features

All customizable purging and partitioning features are implemented in a separate database package from the core features that ship with CA DataMinder. This formal separation of customizable and core features makes them much easier to maintain. In particular, it ensures that any customizations made within the appropriate package body are not overwritten during subsequent upgrades or reinstallations of CA DataMinder.

For example, any future enhancements to core purging procedures provided by CA DataMinder can be easily rolled out to all customers without affecting any existing customizations already in use. Likewise, customers can add their own customizations in the future without affecting core purge and partitioning functionality.

Create timestamp-based partitions

The CA DataMinder database schema provides support for table partitioning based on event time stamps. For example, you can set up 7 day or 30 day partitions.

Configure partition-based purging

Partition-based purging is dramatically faster than relying solely on infrastructure-based purging. After setting up partitions, typically, you will set up a purge strategy based on rolling partitions.

Handling for failed purges

If a partition-based purge fails to complete, you can use information recorded in the CMS database to locate and fix the problem, and then restart the purge (using a database function provided by CA DataMinder) from the point where it failed.

Storage and Partitioning Procedure

Implementing a custom database storage and partitioning scheme involves the following steps:

1. Obtain the native DDL scripts.
2. Customize the DDL scripts.
3. Prepare the database.
4. Run the DDL scripts.
5. Install the CA DataMinder CMS.
6. Set up partitioning and purging.
7. Customize your partitioning scheme.
8. Set up partitions.
9. Set up partition-based purging.

Obtain the Native DDL Scripts

The required DDL scripts are distributed with CA DataMinder. Find them in the following folder in the CA DataMinder distribution image:

\Support\NativeSchemaSQL\ORACLE\Partitioned\Install

What Scripts Are Supplied?

The full list of scripts required varies according to the type of database and whether you are installing a wholly new CA DataMinder installation or upgrading an existing installation. In all cases, however, the core scripts include:

- DWSynonyms.sql
- DWGrant_User.sql
- Grant_user.sql (Oracle only)
- Native_ApplyUtilSPs.sql
- Native_CreateTables.sql
- Native_DWWrapper.sql
- Native_DWCreateTables.sql
- Native_MSSQLSecurity.sql (SQL Server only)
- Native_OracleSecurity.sql (Oracle only)
- Native_PopulateTables.sql
- Native_Wrapper.sql (Oracle only)
- Synonyms.sql (Oracle only)

Partitioning: Default Configuration

(Applicable to Oracle databases only) Partitioning is only supported for captured or imported events, and only tables in the specified tablespace are partitioned. By default, this is the WGNDATA tablespace.

Notes:

- Tables are configured for RANGE partitioning on the EventTimestamp column.
- Tables are sub-partitioned by HASH on the EventUID column. By default, the scripts specify four subpartitions.
- A single default MAXVALUE partition is created by default in the specified tablespace.

More information:

Customizable Names for Partitions, Temporary Tables and Tablespaces (see page 105)

Storage: Default Configuration

Note: When assigning storage to tables and indexes, SQL Server uses 'File Groups' while Oracle uses 'Tablespaces'. In the following sections, the term 'tablespace' is used as a generic reference to either of these terms.

By default, the generated SQL scripts assume that four tablespaces will be created for use by CA DataMinder:

WGN

This tablespace holds all tables containing administration data, that is, user, group, machine, and policy data.

WGNIDX

This tablespace holds the indexes associated with the tables in the WGN tablespace.

WGNDATA

This tablespace holds all tables containing captured or imported event data, including event audit details, participants, and trigger data.

WGNDATAIDX

This holds the indexes associated with the tables in the WGNDATA tablespace.

Customize the DDL Scripts

If required, you can customize the SQL scripts. For example, you can use tablespaces of a different name or refine the storage specifications.

Important! We strongly recommend that any changes to the Native*.sql scripts are made only under the guidance of CA Support at http://ca.com/support.

Modify the Grant_user and Synonyms scripts

(Applicable to Oracle databases only) If you want to use separate database accounts for the CA DataMinder primary user and the schema owner, modify the followng scripts by adding the actual name of the Oracle user.

Note: In all cases, specify the user name in upper case, for example, MY_DLP_USER.

Synonyms.sql

Replace all '<user>' references with the schema owner. This Oracle user owns the CA DataMinder schema tables.

Grant_user.sql

Replace all '<user>' references with the primary user. The CA DataMinder infrastructure uses this Oracle user to access the CMS database.

DWSynonynms

(Optional) Modify this script if you intend to deploy the CA DataMinder data warehouse.

Replace all '<user>' references with the schema owner. This Oracle user owns the CA DataMinder schema tables.

DWGrant_user.sql

(Optional) Modify this script if you intend to deploy the CA DataMinder data warehouse.

Replace all '<user>' references with the primary user. The CA DataMinder infrastructure uses this Oracle user to access the CMS database.

More information:

Obtain the Native DDL Scripts (see page 70)

Prepare the Database

You must install and configure the basic DBMS before the scripts can be installed.

Run the DDL Scripts

The script installation procedure differs for Oracle and SQL Server. These are described below.

Run the SQL Server Scripts

This procedure installs the main database tables for the CMS.

Run the SQL Server scripts using the OSQL utility. For each generated script, run an OSQL command while connecting to the database as the 'CA DataMinder database user'. This is the account that CA DataMinder will use to access the CMS database.

Install the CMS database tables

1. Create a database to host the tables.
2. Create the database user account or accounts needed to install CA DataMinder.

   Ensure that all necessary privileges and roles are assigned to these accounts.
3. From a command prompt, go to the folder containing the Native*.sql scripts.
4. For each script, run an OSQL command. An example is shown below:

   ```
   OSQL -U sa -P password -d WGN
     -i NativeCreateTables.sql -o NativeCreateTables.log
   ```

   Where:

   -U

   Specifies the SQL Server login for CA DataMinder.

   -P

   Specifies the password for the SQL Server login.

   -d

   Specifies the database name for the CA DataMinder database.

   -i

   Specifies the SQL script.

   -o

   Specifies the output log file.

   Note: Parameters are case-sensitive.
5. After these scripts have run, use SQL Server Enterprise Manager to verify that the CA DataMinder schema has been created and exists in the appropriate file groups.
6. This completes the script installation. You now need to install your CMS.

Run the Oracle Scripts: Main Tables

This procedure installs the main database tables for the CMS. There is a separate procedure for installing the Data Warehouse tables.

You run the Oracle scripts from the SQL*Plus utility.

Install the CMS database tables

1. Create the tablespaces required by the scripts, typically:
   - WGN
   - WGNIDX
   - WGNDATA
   - WGNIDX
2. Create the Primary User and (optional) Schema Owner database accounts that you need to install CA DataMinder.

   Verify that all necessary privileges and roles are assigned to these accounts. For details about creating these accounts, see the references at the end of this section.
3. Verify the schema owner and primary user accounts have access to and sufficient quota on the tablespaces created in step 1.
4. Verify that you have modified the Grant_user.sql script to reference the primary user.

   (Optional) Verify that you have modified the Synonyms.sql script to reference the schema owner.

   For details, see Modify the Grant_User and Synonyms Scripts.
5. Copy the modified script files to a location where SQL*Plus can access them.
6. Browse to the folder containing the script files and open Native_Wrapper.sql.
7. Start SQL*Plus, and connect to the database as the user who will own the CA DataMinder schema tables. That is, you must connect as the:
   - **Schema owner** if you **have** created a separate schema owner, or
   - **Primary user** if you have **not** created a separate schema owner.
8. From the SQL*Plus prompt, type:
   ```
   start native_wrapper
   ```

   This creates the tables and indexes and populates the tables with basic data.
9. Review the native_wrapper.log file to confirm that the script executed correctly.
10. If you **have** created separate Primary User and Schema Owner accounts, go to step 11.

    If you do not have separate accounts (that is, the primary user also owns the CA DataMinder schema tables), this completes the script installation.

11. Grant the primary user access to the tables you created in step 8, and create synonyms for this user.
    a. Verify that you are connected to the database as the schema owner.
    b. In SQL*Plus, type this command:
       `start Grant_user`
    c. Still in SQL*Plus, reconnect to the database as the primary user. See step 7 for command details.
    d. To create the Oracle synonyms against the primary user, type this command:
       `start synonyms`

    This completes the native script installation for the CMS.
12. (Optional) Run a native script installation for the data warehouse.

    This procedure installs the database tables for the data warehouse. See the next section for details. Complete the data warehouse installation using native scripts before you continue to step 13.

    If you do not want to install the data warehouse using native scripts, you can install the data warehouse when you run the CMS installation wizard (see step 13).
13. Install your CMS. See the reference below for CMS installation details.

    Note: The CMS installation wizard allows you to install the CMS and data warehouse at the same time.

    Important! When you run the CMS installation wizard, you specify the Primary User, Schema Owner accounts in the Database Accounts screen. Do not click the Create User checkboxes for the these accounts! However, you can create an account for the Search User when you run the wizard (you do not need to specify an existing account).

More information:

Manually Create a Schema Owner (see page 28)
Manually Create a Primary User (see page 29)

Run the Oracle Scripts: Data Warehouse

This procedure installs the data warehouse tables. There is a separate procedure for installing main database tables for the CMS.

Run the Oracle scripts from the SQL*Plus utility.

Note: You do not need to create a separate database schema owner and primary user account. The data warehouse uses the same accounts that you created when running the 'main tables' script, native_wrapper.sql.

Install the data warehouse tables

1. Create the tablespaces required by the scripts. The default tablespaces are:
   - **WGNDWTSS:** Holds the staging area tables.
   - **WGNDWTSF:** Holds the large data warehouse fact tables.
   - **WGNDWTSD:** Holds the data warehouse dimension tables.

   If you want to use your own tablespace names, edit Native_DWWrapper.sql and replace the default tablespace names with your new ones.
2. Verify the schema owner and primary user accounts have access to and sufficient quota on the tablespaces created in step 1.
3. Verify that you have modified the DWGrant_user.sql script to reference the primary user.

   (Optional) Verify that you have modified the DWSynonyms.sql script to reference the schema owner.

   For details, see Modify the Grant_User and Synonyms Scripts.
4. Copy the modified script files to a location where SQL*Plus can access them.
5. Browse to the folder containing the script files and open Native_DWWrapper.sql.
6. Start SQL*Plus, and connect to the database as the user who will own the CA DataMinder schema tables. That is, you must connect as the:
   - **Schema owner** if you **have** created a separate schema owner, or

     Primary user if you have **not** created a separate schema owner.
7. From the SQL*Plus prompt, type:

   ```
   start native_dwwrapper
   ```

   This creates the tables and indexes and populates the tables with basic data.
8. Review the native-dwwrapper.log file to confirm that the script executed correctly.
9. If you **have** created separate Primary User and Schema Owner accounts, go to step 10.

   If you do not have separate accounts (that is, the primary user also owns the CA DataMinder schema tables), this completes the script installation.

10. You now need to grant the primary user access to the tables you created in step 7, and create synonyms for this user.

    a. Verify that you are connected to the database as the schema owner.

    b. In SQL*Plus, type this command:

    ```
    start DWGrant_user
    ```

    c. Still in SQL*Plus, reconnect to the database as the primary user. See step 6 for command details.

    d. To create the Oracle synonyms against the primary user, type this command:

    ```
    start dwsynonyms
    ```

    This completes the native script installation for the data warehouse.

11. If you have not yet installed the CMS, run the CMS installation wizard now before you contine to step 12.

    Note: The CMS installation wizard allows you to install the CMS and data warehouse at the same time.

12. Install the data warehouse.

    See the *Platform Deployment Guide* for details.

Install the CMS

After installing the SQL scripts, you can install the CA DataMinder central management server (CMS) using the server installation wizard. For details, see 'Installing a CMS' in the *Platform Deployment Guide*.

In particular, note the following wizard screens:

Database Identification screen

Specify the SID or service identifier (for Oracle) or the database name (for SQL Server).

Database Accounts screen**,**

Enter the name and password for the various CA DataMinder database accounts: the primary user, search user and, optionally for Oracle CMSs, the schema owner.

Schema Owner

(Oracle CMSs only) This optional account owns the database schema. Some organizations choose to have separate accounts for the primary user and the database owner. This is typically for security reasons, for example, to ensure that employees cannot connect to the CMS database as the primary user and delete sensitive data or drop the underlying database objects.

Primary User

This is the main CA DataMinder database account. The infrastructure uses this account to access the CMS database. By default, this user also 'owns' the database schema unless a Schema Owner is specified.

Search User

CA DataMinder consoles use this database account when searching the CMS database for events. This is a secure account that is subject to row level security (RLS) when searching the database for events. This ensures that reviewers cannot see events that they are not permitted to see when they run a search. If multiple database security models are enabled on your CMS, specify a separate Search User database account for each security model.

You created the primary user and (optionally) schema owner accounts when you prepared the database, so you do not need to create these accounts now. But you may need to create the Search User account.

Note: For details about the requirements for your Oracle users or SQL Server logins, see the references at the end of this section.

Data Warehouse Configuration screen

Specify whether to enable data warehousing and whether to collect event participant data.

The iConsole dashboard and BusinessObjects reports for CA DataMinder require the data warehouse. If you want to run reports or dashboards that show results broken down by user group, you must also collect event participant data.

For full details about these deploymet options, see the Data Warehouse chapter in the *Platform Deployment Guide*.

Data Warehouse Database Accounts screen

(Required only you enable the data warehouse.) Enter the name and password for the following database accounts:

Data Warehouse User

External reporting applications (such as BusinessObjects Enterprise) use this database account to connect to the Data Warehouse and CMS database.

Unrestricted Search User

This database account corresponds to the 'Unrestricted' security model. CA DataMinder consoles and external reporting tools can use this database account when searching the CA DataMinder Data Warehouse and CMS database for events. Unlike normal Search User database accounts, the Unrestricted Search User is *not* subject to row level security (RLS) when searching the database. If a reviewer has 'Unrestricted' security model, the reviewer can see any events when they run a search or report. Search results or reports are not restricted by policy class or the reviewer's management group.

More information:

Oracle Users (see page 26)
SQL Server Logins (see page 48)

Set Up Partitions and Purging

(Oracle databases only)

Your database is now ready for you to implement partitions and a partition-based purging scheme. Specifically, you need to:

Create timestamp-based partitions

The CA DataMinder database schema provides support for table partitioning based on event time stamps. For example, you can set up 7 day or 30 day partitions.

Set up partition-based purging

Partition-based purging is dramatically faster than relying solely on infrastructure-based purging. After setting up partitions, you will need to configure the purging behavior. Typically, you will set up a purge strategy based on rolling partitions.

Run or schedule a purge

When you schedule or manually run a partition-based purge, your purge procedure must include steps to suspend the CMS and delete any corresponding blob files.

Customize your purges

You can fully customize your partitioning and purging operations. For example, you can define the retention conditions for records due to be purged. You can also configure custom operations to run, for example, when a purge operation fails.

You can use functions in the wgn_partition_util package to specify the partition time ranges, create partitions, configure partition-based purging, and restart failed purges. You can customize the retention conditions and configure other custom operations using functions and procedures in the wgn_partition_cust package.

More information:

Partitioning Packages (see page 82)
Create Database Partitions (see page 83)
Set Up Partition-Based Purging (see page 93)
Run or Schedule a Partition-based Purge (see page 99)
Customize Your Partitioning Scheme (see page 103)
Diagnostic Support (see page 110)
Example Purge Sequence (see page 111)

Partitioning Packages

Customized purging and partitioning procedures are implemented in a separate package from the core procedures that ship with CA DataMinder.

Core Package: wgn_partition_util

This package is installed automatically on all Oracle-based CA DataMinder servers. It contains the core purging and partitioning functionality.

It includes the package spec, which defines the interfaces used by the core partitioning procedures, and the package body, which contains the actual functions and procedures, configured by you and which are used to create a partitioning scheme and configure partition-based purging.

Important! You must not edit the package spec!

More information:

Set Up Partition-Based Purging (see page 93)
Create Database Partitions (see page 83)

Customizable Package: wgn_partition_cust

For this customizable package, only the package spec is installed automatically on (Oracle-based) CA DataMinder servers. This spec defines the interfaces you can use in the package body to customize your own purging and partitioning procedures. These include the function you use to customize the record retention conditions. You can also customize other procedures defined in the spec. For example, you may want to send alerts to your DBAs if a partition fails to be created.

These customizable interfaces are invoked directly by the wgn_partition_util package. At run time, CA DataMinder checks for customized versions of the relevant functions and procedures in the wgn_partition_cust package body. If these are not found, it invokes the default behavior defined in the wgn_partition_util package.

A sample body package, wgn_partition_cust_body.sql, is available in the \Support\CustomSQLTemplates subfolder on your CA DataMinder distribution media.

More information:

Specify the Retention Condition (see page 94)

Create Database Partitions

To create database partitions, you need to execute procedures in the wgn_partition_util package.

More information:

Which Tables Are Partitioned?

Partitioning affects the primary event table plus all related tables in the CMS database.

Required Date Format

When specifying date ranges in the wgn_partition_util partitioning package, you **must** enter dates in the correct format. By default, you must use this format:

```
'dd-mon-yyyy'
```

You must use this format even if the local system uses American dates (mon-dd-yyyy). Alternatively, you can customize the required format. But for all formats, you must enclose the date in single quotes.

Add a Series of Partitions

To add a series of partitions, you must execute the wgn_partition_tables procedure as the schema owner. The prototype is shown below:

```
wgn_partition_tables(
  date_range IN VARCHAR2 ,
  num_partitions IN NUMBER ,
  num_days_per_partition IN NUMBER ,
  db_type IN NUMBER DEFAULT wgn_db_type_CMS ,
  Partition_Options IN NUMBER DEFAULT 0 /*
    1 = wgn_opt_Partition_NonPart_Tabs,
    2 = wgn_opt_Convert_to_Partitioned,
    3 = wgn_opt_Convert_to_NonPartition */ ,
  new_tablespace IN VARCHAR2 DEFAULT NULL ,
  new_indextablespace IN VARCHAR2 DEFAULT NULL ,
  pdebug IN BOOLEAN DEFAULT FALSE
  );
```

Where:

date_range

Is the end date of the earliest partition.

By default, the date format must be dd-mmm-yyyy (for example, 08-mar-2011).

num_partitions

Specifies the number of partitions you want to create.

num_days_per_partition

Is the number of days in each partition.

db_type

Specifies the database that you want to partition:

wgn_partition_util.wgn_db_type_CMS

(Default) Specifies the CMS database tables.

wgn_partition_util.wgn_db_type_DW

Specifies the data warehouse tables.

wgn_partition_util.wgn_db_type_BOTH

Specifies both databases. That is, you want to partition tables in the CMS database and the data warehouse.

Partition_Options

Specifies how you want to partition the database:

0

(Default) Creates partitions only in existing partitioned tables.

wgn_partition_util.wgn_opt_Partition_NonPart_Tabs

Converts all non-partitioned tables to partitioned tables.

wgn_partition_util.wgn_opt_Convert_to_Part

Converts existing partitioned tables to partitioned tables based on the new partitioning scheme.

wgn_partition_util.wgn_opt_Convert_to_NonPart

Converts existing partitioned tables to non-partitioned tables.

Important! If the Partition_Options parameter is set to a non-default value, existing tables are copied and replaced with new tables partitioned as specified. The old copies of the tables are renamed BK<x>_<TableName>. This operation creates a complete copy of the data and consumes a significant amount of storage!

new_tablespace

Specifies which tablespace to use. (If a custom tablespace is specified, the custom tablespace takes precedence.)

new_indextablespace

Specifies which tablespace to use for new indexes.

pdebug

(Optional) Specifies whether a debug trace is produced. It defaults to false.

Examples

This command creates 12 partitions for the CMS tables, starting on 31 January 2011, and each 30 days long:

```
SQL> EXEC wgn_partition_util.wgn_partition_tables(
  date_range =>'31-jan-2011' ,
  num_partitions =>12 ,
  num_days_per_partition =>30 ,
  db_type =>wgn_partition_util.wgn_db_type_cms
  );
```

This command creates the same partition structure for the data warehouse tables:

```
SQL> EXEC wgn_partition_util.wgn_partition_tables(
  date_range =>'31-jan-2011' ,
  num_partitions =>12 ,
  num_days_per_partition =>30 ,
  db_type =>wgn_partition_util.wgn_db_type_dw
  );
```

This command creates the same partition structure for the data warehouse tables, but also copies and replaces the existing tables with new partitioned tables. This command specifies WGNDWTSF tabelspace for the tables and WGNDWTSFIDX tablespace for the indexes:

```
SQL> EXEC wgn_partition_util.wgn_partition_tables(
  date_range =>'31-jan-2011' ,
  num_partitions =>12 ,
  num_days_per_partition =>30 ,
  db_type =>wgn_partition_util.wgn_db_type_DW ,
  Partition_Options =>wgn_partition_util.wgn_opt_Convert_to_Partitioned ,
  new_tablespace =>'WGNDWTSF' ,
  new_indextablespace =>'WGNDWTSFIDX'
  );
```

Add a Series of Partitions Using Dates

You can also add a series of partitions using dates. As before, you must execute the wgn_partition_tables procedure as the schema owner. The prototype is shown below:

```
wgn_partition_tables(
  partition_dates IN WGN_PARTITION_DATES_TAB ,
  db_type IN NUMBER DEFAULT wgn_db_type_CMS ,
  Partition_Options IN NUMBER DEFAULT 0 /*
    1 = wgn_opt_Partition_NonPart_Tabs,
    2 = wgn_opt_Convert_to_Part,
    3 = wgn_opt_Convert_to_NonPart */ ,
  new_tablespace IN VARCHAR2 DEFAULT NULL ,
  new_indextablespace IN VARCHAR2 DEFAULT NULL ,
  pdebug IN BOOLEAN DEFAULT FALSE
  );
```

Where:

partition_dates

Is a WGN_PARTITION_DATES_TAB table of partition dates.

db_type

Specifies the database that you want to partition:

wgn_partition_util.wgn_db_type_CMS

(Default) Specifies the CMS database tables.

wgn_partition_util.wgn_db_type_DW

Specifies the data warehouse tables.

wgn_partition_util.wgn_db_type_BOTH

Specifies both databases. That is, you want to partition tables in the CMS database and the data warehouse.

Partition_Options

Specifies how you want to partition the database:

0

(Default) Creates partitions only in existing partitioned tables.

wgn_partition_util.wgn_opt_Partition_NonPart_Tabs

Converts all non-partitioned tables to partitioned tables.

wgn_partition_util.wgn_opt_Convert_to_Part

Converts existing partitioned tables to partitioned tables based on the new partitioning scheme.

wgn_partition_util.wgn_opt_Convert_to_NonPart

Converts existing partitioned tables to non-partitioned tables.

Important! If the Partition_Options parameter is set to a non-default value, existing tables are copied and replaced with new tables partitioned as specified. The old copies of the tables are renamed BK<x>_<TableName>. This operation creates a complete copy of the data and consumes a significant amount of storage!

new_tablespace

Specifies which tablespace to use. (If a custom tablespace is specified, the custom tablespace takes precedence.)

new_indextablespace

Specifies which tablespace to use for new indexes.

pdebug

(Optional) Specifies whether a debug trace is produced. It defaults to false.

Examples

This command creates two partitions for CMS tables, with dates 25 June 2011 and 29 June 2011.

```
SQL> EXEC Wgn_Partition_Util.wgn_partition_tables(
  partition_dates =>
    WGN_PARTITION_DATES_TAB(
      WGN_PARTITION_DATE(TO_DATE('25 Jun 2011'), NULL, NULL) ,
      WGN_PARTITION_DATE(TO_DATE('29 Jun 2011'), NULL, NULL)
      ) ,
  db_type => Wgn_Partition_Util.wgn_db_type_CMS ,
  pdebug => True
  );
```

This command uses the Wgn_Partition_Util.get_partition_dates function to return the CMS partition dates. The command then uses these dates to partition the data warehouse using dates that match the CMS partition dates:

```
SQL> EXEC Wgn_Partition_Util.wgn_partition_tables(
  partition_dates =>
Wgn_Partition_Util.get_partition_dates(Wgn_Partition_Util.wgn_db_type_cms)
  , db_type => Wgn_Partition_Util.wgn_db_type_DW
  , Partition_Options => Wgn_Partition_Util.wgn_opt_Convert_to_Partitioned
  , new_tablespace => 'WGNDWTSF'
  , new_indextablespace => 'WGNDWTSFIDX'
  pdebug => True
  );
```

Add a Specific Partition

To add a specific partition, you must execute the wgn_add_partition procedure as the schema owner. The prototype is:

```
wgn_add_partition(
  date_range IN VARCHAR2
  db_type IN NUMBER
  );
```

Where:

date_range

Specifies the end date for the new partition.

By default, the date format must be dd-mmm-yyyy (for example, 08-mar-2011).

db_type

Specifies the database that you want to partition:

wgn_partition_util.wgn_db_type_CMS

(Default) Specifies the CMS database tables.

wgn_partition_util.wgn_db_type_DW

Specifies the data warehouse tables.

wgn_partition_util.wgn_db_type_BOTH

Specifies both databases. That is, you want to partition tables in the CMS database and the data warehouse.

Examples

A database is segmented into partitions 30 days long. If the latest existing partition has an end date of March 7, this command adds a new partition to store events with capture dates between March 8 and April 6:

```
SQL> EXEC wgn_partition_util.wgn_add_partition (
  date_range =>'06-apr-2011',
  db_type => Wgn_Partition_Util.wgn_db_type_CMS
  );
```

This command adds the same partition for the data warehouse:

```
SQL> EXEC wgn_partition_util.wgn_add_partition (
  date_range =>'06-apr-2011',
  db_type => Wgn_Partition_Util.wgn_db_type_DW
  );
```

Specify the Tablespace for Individual Objects

When specifying a partition scheme, you can also specify the tablespace for an individual table or index object for the current session.

If you do not specify the tablespace for an individual object, CA DataMinder assigns tables to tablespaces in a specific order of precedence.

Example

The following Wgn_Partition_Util.Set_Obj_Tablespace commands specify:

- The TS_WGN3EVENT tablespace for the WGN3EVENT table.
- The TS_IX_WGN3EVENT_EVENTTIMESTAMP tablespace for the IX_WGN3EVENT_EVENTTIMESTAMP index.

```
SQL> EXEC Wgn_Partition_Util.Set_Obj_Tablespace('WGN3EVENT', 'TS_WGN3EVENT');
SQL> EXEC Wgn_Partition_Util.Set_Obj_Tablespace('IX_WGN3EVENT_EVENTTIMESTAMP',
'TS_IX_WGN3EVENT_EVENTTIMESTAMP');
SQL> EXEC Wgn_Partition_Util.wgn_partition_tables(
  date_range =>'31-jan-2010'
  , num_partitions =>12
  , num_days_per_partition =>30
  , db_type => Wgn_Partition_Util.wgn_db_type_CMS
  , Partition_Options => Wgn_Partition_Util.wgn_opt_Convert_to_Partitioned
  , new_tablespace => 'WGNDWTSF'
  , new_indextablespace => 'WGNDWTSFIDX'
  );
```

Tablespace Order of Precedence

CA DataMinder supports the following methods for assigning tables to tablespaces. CA DataMinder applies these methods in the following order of precedence:

1. Use the Wgn_Partition_Util.Set_Obj_Tablespace function.

   If this function is used, the specified tablespaces override all other methods.

2. Use custom tablespaces.

   If custom tablespaces are used, they override the parameters in method 3.

3. Use the tablespace parameters for the wgn_partition_tables procedure.
   `new_tablespace`
   `new_Indextablespace parameters`

4. Use the existing tablespaces.

   If methods 1, 2 and 3 are not specified but a table is already assigned to a tablespace, CA DataMinder uses the existing tablespace.

5. Use the WGNDATA default tablespace.

   CA DataMinder assigns a table to the default WGNDATA tablespace if no other sources are specified.

Delete a Specific Partition

A partitioned database cannot have gaps in the partition date range values. Therefore, in order to delete a specific partition, you must purge the partition you want to delete and then merge this partition with the previous partition. The previous partition acquires the high value of the purged partition.

To do this, execute the wgn_roll_partition procedure, but also specify the perm_data_date_range as the previous partition date range. The prototype is:

```
EXEC wgn_roll_partitions(
  new_partition_date_range IN VARCHAR2,
  date_range_to_purge IN VARCHAR2,
  perm_data_date_range IN VARCHAR2,
  db_type IN NUMBER
  );
```

Where:

new_partition_date_range

Specifies the end date for the new partition.

This parameter adds the new partition into the existing tablespace. In the Wgn3PurgeHistory table, this date is listed in the P1 column. You can also set this parameter to NULL to purge a specific partition without adding a new one.

date_range_to_purge

Specifies the end date for the partition that you want to purge. Date formats are as above.

This parameter purges all non-permanent data from the partition specified by the date_range_to_purge parameter. In the Wgn3PurgeHistory table, this date is listed in the P2 column.

perm_data_date_range

Specifies the end date for the historic partition. Date formats are as above.

db_type

Specifies the database that you want to partition:

wgn_partition_util.wgn_db_type_CMS

(Default) Specifies the CMS database tables.

wgn_partition_util.wgn_db_type_DW

Specifies the data warehouse tables.

wgn_partition_util.wgn_db_type_BOTH

Specifies both databases. That is, you want to partition tables in the CMS database and the data warehouse.

Example

The following command purges a CMS partition with an end date of 31 January 2011. It then merges the 31 December partition into this partition:

```
EXEC wgn_partition_util.wgn_roll_partitions(
  new_partition_date_range => null ,
  date_range_to_purge => '31-jan-2011' ,
  perm_data_date_range => '31-dec-2010' ,
  db_type => wgn_partition_util.wgn_db_type_CMS
  );
```

More information:

Purge a Specific Partition (see page 97)

Additional Procedures and Functions

Further partitioning functions and procedures are included in the wgn_partition_util package spec, but not documented in this guide. These are provided primarily for testing purposes and must only be used in a production environment under the guidance of CA Technical Support.

Set Up Partition-Based Purging

To implement a partition-based purging scheme, you need to execute functions and procedures in the wgn_partition_util and wgn_partition_cust packages. You may also want to customize the default purge operations, for example, to change the default retention conditions.

Note: Scheduled event purges driven are by the CA DataMinder infrastructure.

Important! When specifying a partition-based purge, you must do so as the schema owner!

More information:

Infrastructure-Based Purging (see page 57)

What Data Is Purged?

Partition-based purges remove eligible database records but **not** any corresponding blob (Binary Large Object) files. Instead, you must incorporate into your partition-based purging scheme steps to delete these blob files.

Note: Each CA DataMinder event comprises metadata, written to the local CA DataMinder database, and a blob file, saved on physical media.

More information:

What Data Is Purged? (see page 57)

Which Tables Are Purged?

Partition-based purges affect the primary event table plus all related tables in the CMS database and the data warehouse.

Suspending the CMS and Deleting Blob Files

Important! Partition-based purging will purge individual partitions but **not** the corresponding blob files! You must also suspend the CMS while the purge runs.

More information:

Run or Schedule a Partition-based Purge (see page 99)

Historic Partitions

CA DataMinder support for partition-based purging allows you to configure a 'historic' partition. This is a special partition for holding records that must be retained permanently (that is, exempted from purging). Any permanent records from the purged partition are merged into this historic partition if you specify one.

Purge Dates

Important! When specifying a purge operation, date ranges must correspond to partition boundaries.

When creating partitions, the wgn_partition_util package uses partition end dates to generate partition names. Likewise, when purging partitions, the same package uses the specified date range to identify the partition by name.

Recording Purge Progress and History

The progress and status of partition-based purge operations are recorded in two views, WGN_V_PURGESTATE and WGN_V_PURGEHISTORY.

If a partition-based purge fails to complete, you can use information recorded in these views to diagnose the failed purge. You can then fix the problem and restart the purge

More information:

Diagnostic Support (see page 110)

Specify the Retention Condition

By default, each purge operation purges the specified partition but retains any events marked as 'Do Not Delete'. However, you may want to customize the purge to retain other categories of events (for example, all events whose status is not 'Closed').

To customize your retention conditions, you must implement the function wgn_get_retention_condition in the wgn_partition_cust package. This function is called during the purge process.

More information:

Customizable Retention Conditions (see page 104)

Set up Rolling Partitions

In a typical RANGE partitioning model, the purge operation periodically purges the specified partition and adds a new partition.

To define your rolling partitions, you must execute the wgn_roll_partitions procedure in the wgn_partition_util package. The prototype is shown below:

```
wgn_roll_partitions(
  new_partition_date_range IN VARCHAR2,
  date_range_to_purge IN VARCHAR2,
  perm_data_date_range IN VARCHAR2 := null,
  db_type IN NUMBER
  pdebug IN BOOLEAN DEFAULT FALSE
  );
```

Where:

new_partition_date_range

Specifies the end date for the new partition.

This parameter adds the new partition into the existing tablespace. In the Wgn3PurgeHistory table, this date is listed in the P1 column. You can also set this parameter to NULL to purge a specific partition without adding a new one—see the next section.

By default, the date format must be dd-mmm-yyyy (for example, 08-mar-2011).

date_range_to_purge

Specifies the end date for the partition that you want to purge. Date formats are as above.

This parameter purges all non-permanent data from the partition specified by the date_range_to_purge parameter. In the Wgn3PurgeHistory table, this date is listed in the P2 column.

perm_data_date_range

(Optional) Specifies the end date for the historic partition. Date formats are as above.

If this parameter is not specified, permanent records stay in their original partition. In the Wgn3PurgeHistory table, this date is listed in the P3 column.

db_type

Specifies the database that you want to partition:

wgn_partition_util.wgn_db_type_CMS

(Default) Specifies the CMS database tables.

wgn_partition_util.wgn_db_type_DW

Specifies the data warehouse tables.

wgn_partition_util.wgn_db_type_BOTH

Specifies both databases. That is, you want to partition tables in the CMS database and the data warehouse.

pdebug

(Optional) Specifies whether a debug trace is produced. It defaults to false.

Example

The command below creates a new partition for the CMS, starting 31 January 2011 and purges the partition 31 January 2010:

```
SQL> EXEC wgn_partition_util.wgn_roll_partitions (
  new_partition_date_range =>'31-jan-2011',
  date_range_to_purge =>'31-jan-2010',
  db_type => Wgn_Partition_Util.wgn_db_type_CMS
  );
```

This command adds and purges the same partitions for the data warehouse:

```
SQL> EXEC wgn_partition_util.wgn_roll_partitions (
  new_partition_date_range =>'31-jan-2011',
  date_range_to_purge =>'31-jan-2010',
  db_type => Wgn_Partition_Util.wgn_db_type_DW
  );
```

More information:

Wgn3PurgeHistory (see page 111)
Historic Partitions (see page 94)
Example Purge Sequence (see page 111)

Purge a Specific Partition

You can use the wgn_roll_partitions procedure to purge a specific partition by setting the new_partition_date_range parameter to NULL. The prototype is shown below:

```
wgn_roll_partitions(
  new_partition_date_range IN VARCHAR2,
  date_range_to_purge IN VARCHAR2,
  db_type IN NUMBER
  );
```

Where:

new_partition_date_range

Specifies the end date for the new partition.

This parameter adds the new partition into the existing tablespace. In the Wgn3PurgeHistory table, this date is listed in the P1 column.

By default, the date format must be dd-mmm-yyyy (for example, 08-mar-2011).

date_range_to_purge

Specifies the end date for the partition that you want to purge. Date formats are as above.

This parameter purges all non-permanent data from the partition specified by the date_range_to_purge parameter. In the Wgn3PurgeHistory table, this date is listed in the P2 column.

db_type

Specifies the database that you want to partition:

wgn_partition_util.wgn_db_type_CMS

(Default) Specifies the CMS database tables.

wgn_partition_util.wgn_db_type_DW

Specifies the data warehouse tables.

wgn_partition_util.wgn_db_type_BOTH

Specifies both databases. That is, you want to partition tables in the CMS database and the data warehouse.

Note: When you purge a partition, the partition is not automatically deleted. For instructions on deleting a specific partition, see the reference below.

Example

This command purges a CMS partition with an end date of 31 January 2011:

```
EXEC wgn_partition_util.wgn_roll_partitions (
  new_partition_date_range => null,
  date_range_to_purge => '31-jan-2011',
  db_type => Wgn_Partition_Util.wgn_db_type_CMS
  );
```

More information:

Delete a Specific Partition (see page 91)

Restart a Failed Purge

CA DataMinder does not permit a purge to be left incomplete and in a failed state. If a purge fails to complete, after fixing the problem you can use the wgn_restart_purge procedure to restart the purge. Note also that restarted purges do not repeat or omit any purge steps.

```
begin
wgn_partition_util.wgn_restart_purge;
end;
```

Because you cannot start a new purge while the latest purge is in a failed state, the Restart procedure does not need to take any parameters; by definition, any failed purge that needs to be restarted must be the most recent purge.

More information:

Diagnostic Support (see page 110)

Additional Procedures and Functions

Further purging functions and procedures are supported in the wgn_partition_util package spec, but not documented in this guide. These are provided primarily for testing purposes and must only be used in a production environment under the guidance of CA Technical Support.

Run or Schedule a Partition-based Purge

When you run or schedule a partition-based purge, your purge procedure must incorporate steps to suspend the CMS and delete any corresponding blob files.

Suspend the CMS

The CMS must be suspended for the duration of any partition-based purge and then resumed when the purge is complete. This is because some key constraints are disabled during the purge, and suspending the CMS prevents new data being written to the database while these constraints are removed; suspending the infrastructure guarantees that all replication activity is paused for the duration of the purge.

Important! CA DataMinder consoles can still connect the CMS while it is suspended. For example, you can use the Administration console to amend policies on a suspended CMS.

Delete residual blob files

Partition-based purging will purge individual partitions but **not** the corresponding blob files. Although the associated blob files are not deleted by these purges, they are marked for deletion in the Wgn3BlobDeletion database table. This enables you to run a CA DataMinder command to delete these blobs after the partition-based purge completes.

The following sections describe how to incorporate these steps into procedures for:

- Scheduling a purge by linking it to an infrastructure-based purge (see the next section)
- Scheduling a purge by using a third party scheduler such as Unicenter Autosys
- Manually running a purge.

Link to a Scheduled Infrastructure-based Purge Using a Custom SP

Infrastructure-based purges automatically delete any corresponding blob files. You can therefore link your partition-based purge to a scheduled infrastructure-based purge. This involves the following procedure:

1. **Configure CMS machine policy**

   You need to configure the machine policy to suspend the CMS during event purges. In the Data Management policy folder, set the Suspend Infrastructure During Purge? setting to True.

2. **Purge the partitions**

   The infrastructure-based purge must be configured to invoke a custom SP. The recommended method depends on which database account owns the schema:

   Primary user also owns the schema

   You need to implement a custom SP to execute the wgn_roll_partitions procedure.

   Separate database accounts for primary user and schema owner

   We do not recommend using this method; instead, we recommend using a third-party scheduler—see the next section. But if you do not have a third party scheduler, you will need to use this more complex method. Specifically, you need a mechanism to automatically run a partition-based purge before the scheduled infrastructure purge, which itself must invoke a 'no operation' custom SP. This SP prevents the infrastructure-based purge from trying to delete event records in the CMS database that will have already been removed by the partition-based purge.

3. **Delete the blob files**

   The infrastructure-based purge then automatically deletes any residual blob files marked for deletion.

More information:

Purges and Stored Procedures (see page 61)
Partition-Based Purges (see page 63)
Run or Schedule a Partition-based Purge (see page 99)

Using a Third-party Scheduler

If you have a separate schema owner for your CA DataMinder database, this is the method we recommend for scheduling a partition-based purge.

Using a third-party schedule such as Unicenter Autosys, you can run a sequence of CA DataMinder commands and Oracle scripts:

1. **Suspend the CMS**

   Add this command to suspend the CMS. This command must be run from the \System subfolder of the CA DataMinder installation folder on the CMS:

   ```
   wgninfra -exec wigan/infrastruct/Infrastruct Suspend
   ```

   If successful, the suspend mechanism returns zero; if unsuccessful, it returns a non-zero value, indicating an error. Errors are recorded in wgnlee.log; find this logfile on the CMS in the \System subfolder of the CA DataMinder installation folder.

2. **Run the purge**

   Add an Oracle script to purge the specified partitions. This script needs to execute the wgn_roll_partitions procedure.

   Automation

   Wgn_roll_partitions takes several date parameters. Your scheduler script must pass the correct parameters to wgn_roll_partitions to ensure the correct partitions are added and removed. If required, your script can derive these parameters from the Wgn3PurgeHistory database table.

   Purge failure

   If the purge fails to complete, you must also abort the remaining steps. That is, do not resume the CMS (see step 3) or delete the blobs (see step 4). After fixing the problem (see below), you can restart the purge from the point where it failed and then proceed to steps 3 and 4.

   Note: You can customize the 'on failure' procedure, for example, to alert your DBA if a purge fail.

   Purge failure continued

   To locate and fix the problem before restarting the failed purge, use information recorded in the Wgn3PurgeState and Wgn3PurgeHistory database tables.

   Most importantly, your scheduler script must include error handling to re-enable key constraints if it detects an error from wgn_roll_partitions. This ensures that data integrity is preserved. If your scheduler script fails to re-enable key constraints, you must do so manually—see page 51 for instructions.

   Note: Wgn_roll_partitions will re-enable key constraints if it detects a purge error, but we recommend that your scheduler script also includes error handling as a further safeguard.

3. **Resume the CMS**

   Add the following command. As for step 1, this must be run from the \System folder on the CMS:

   ```
   wgninfra -exec wigan/infrastruct/Infrastruct Resume
   ```

   If successful, the resume mechanism returns zero; if unsuccessful, it aborts and returns a non-zero value, indicating an error. As for step 1, errors are recorded in wgnlee.log.

4. **Delete the blob files**

   Add the following command. As before, run this from the \System folder on the CMS:

   ```
   wgninfra -exec wigan/infrastruct/database/DBManagement PurgeBLOBs
   ```

   If successful, the blob purge mechanism returns zero; if unsuccessful, it returns a non-zero value, indicating an error. If an error occurs, the scheduler can retry calling this command to delete the blob files. As for step 1, errors are recorded in wgnlee.log.

More information:

Manual Purges (see page 66)
Diagnostic Support (see page 110)

Manually Run a Purge

This is the simplest method and is suitable for, say, evaluation and testing purposes.

1. **Suspend the CMS**

   You can do this manually in the Administration console (see the online help for details) or you can run a command. From a command prompt in the \system subfolder in the CA DataMinder installation folder on the CMS, run:

   ```
   wgninfra -exec wigan/infrastruct/Infrastruct Suspend
   ```

2. **Run the purge**

   Execute the wgn_roll_partitions procedure. The procedure prototype is shown in Set up rolling partitions section.

3. **Resume the CMS**

   As before, you can do this in the Administration console or you can run this command:

   ```
   wgninfra -exec wigan/infrastruct/Infrastruct Resume
   ```

4. **Delete the blob files**

   Finally, you need to purge the residual blob files. Run this command to purge the blob files marked for deletion:

   ```
   wgninfra -exec wigan/infrastruct/database/DBManagement PurgeBLOBs
   ```

If a Purge Fails to Complete

If a partition-based purge fails to complete (for example, when using a third party scheduler), be aware of the following:

Manually re-enable the key constraints

If a purge fails to complete, we recommend that you manually re-enable key constraints. To do this, execute the relevant following procedures in the wgn_partition_util package:

```
wgn_enable_pk_constraints (log_progress BOOLEAN:=TRUE);
wgn_enable_fk_constraints (log_progress BOOLEAN:=TRUE);
```

For example, run the following command to enable the primary and foreign key constraints respectively.

```
begin
wgn_partition_util.
wgn_enable_fk_constraints;
wgn_partition_util.
wgn_enable_fk_constraints;
end;
```

Customize the 'on failure' procedure

If a purge fails, the wgn_on_stmt_fail procedure is invoked immediately, see Customizable SQL statement procedures. This procedure is designed to be customizable, allowing you to implement a mechanism to notify your DBA (for example, by pager or email) about the purge failure.

You customize wgn_on_stmt_fail in the package body of wgn_partition_cust.

More information:

Customizable SQL Statement Procedures (see page 109)

Customize Your Partitioning Scheme

Any customizations made to the partitioning package body in wgn_partition_cust must adhere to the interfaces defined in the package spec. These functions and procedures permit you to add additional functionality if required. These are summarized below and on the following pages. You customize these functions in the package body of wgn_partition_cust.

More information:

Customizable Date Formats

You can customize the date format for partition names and the required format for dates supplied as parameters to functions or procedures.

The following function defines the date format used for naming the actual partitions in the database:

```
FUNCTION wgn_get_pn_date_fmt
RETURN VARCHAR2;
```

By default, the wgn_partition_util package uses ‘yyyy/mm/dd’ dates (for example, 2007/01/31). This ensures that partitions are listed in chronological order in tools such as OEm or TOAD.

If you set a new date format, you must enclose it in single quotes.

The following function defines the date format for parameters supplied to functions and procedures in the package bodies for wgn_partition_util and wgn_partition_cust. You **must** enter dates in the correct format.

```
FUNCTION wgn_get_ip_date_fmt
RETURN VARCHAR2;
```

By default, the wgn_partition_util package requires that you use ‘dd-mmm-yyyy’ dates (for example, 01-jan-2011), regardless of how the local system is configured to display dates.

If required, you can reset this to, for example, 'mmm-dd-yyyy'. If you set a new date format, you must enclose it in single quotes.

Customizable Retention Conditions

By default, each purge operation purges the specified partition but retains any events marked as 'Do Not Delete' in the Wgn3Event table. However, you can customize the purge to retain other categories of events, such as all events whose status is not 'Closed'.

To define a retention condition, customize the following function:

```
FUNCTION wgn_get_retention_condition
(table_name IN VARCHAR2,
date_range IN VARCHAR2)
RETURN VARCHAR2;
```

Where:

table_name

Specifies the database table being checked for events that must be retained.

date_range

Specifies the end date for the partition being purged.

Customizable Names for Partitions, Temporary Tables and Tablespaces

You can customize the naming conventions for partitions, the MAXVALUE partition, temporary tables and the tablespace for partitioned tables. You customize these functions in the package body of wgn_partition_cust.

Partition Names

By default, partition names include: a 'TP_' prefix; the name of the database table being partitioned; and the partition end date (the date format is defined by the wgn_get_pn_date_fmt function. Partition names use this pattern:

```
TP_<table name><end date>
```

For example:

```
TP_WGN3EVENT2006/01/31
```

If required, you can customize the partition naming convention. The function prototype is shown below:

```
FUNCTION wgn_get_partition_name
(table_name IN VARCHAR2,
date_range IN VARCHAR2)
RETURN VARCHAR2;
```

MAXVALUE Partition Name

The MAXVALUE partition contains any 'anomalous' records whose time-stamp date does not match an existing partition. For example, if the latest partition has an end date of 31 January 2007, the MAXVALUE partition accepts all events with a timestamp of 1 February 2007 or later. If a new partition is created subsequently with a date range that matches these anomalous records, they are automatically migrated out of the MAXVALUE partition and into the new partition.

The MAXVALUE partition name uses this pattern: :

```
<table name>_DEF
```

For example:

```
WGN3EVENT_DEF
```

If required, you can customize this partition name. The function prototype is shown below:

```
FUNCTION wgn_get_maxvalue_name
(table_name IN VARCHAR2)
RETURN VARCHAR2;
```

Temporary Table Name

By default, the names of temporary tables include a '_TEMP' suffix:

```
<table name>_TEMP
```

For example,

```
WGN3EVENT_TEMP
```

If required, you can customize the partition naming convention. The function prototype is shown below:

```
FUNCTION wgn_get_tmp_tab_name
(table_name IN VARCHAR2)
RETURN VARCHAR2;
```

Tablespace Name

Only CA DataMinder tables containing captured or imported events can be partitioned. By default, these tables are in the WGNDATA tablespace, but you can specify a different tablespace. The function prototype is shown below:

```
FNCTION wgn_get_tablespace_name
(table_name IN VARCHAR2)
RETURN VARCHAR2
```

Note: The wgn_get_tablespace_name function assumes that all partitioned tables are in the same tablespace. But if you have separated these tables into separate tablespaces (for example, assigning event, auditing and trigger tables to separate tablespaces), then you will need to implement wgn_get_tablespace_name with a custom function that reflects your own site-specific database deployment.

Customizable Pre-and Post-Operation Procedures

Customizable pre- and post-operation procedures are invoked by operations in the purge process. You can define customized procedures in the package body of wgn_partition_cust. If no customized procedures are defined, CA DataMinder invokes default procedures. These default procedures do nothing. The procedure prototypes are listed below.

Creating a temporary table

These procedures are invoked immediately before or after creating a temporary table to hold events that need to be retained.

Pre-operation

```
PROCEDURE wgn_pre_create_temp_table
(table_name IN VARCHAR2,
date_range IN VARCHAR2);
```

Post-operation

```
PROCEDURE wgn_post_create_temp_table
(table_name IN VARCHAR2,
date_range IN VARCHAR2);
```

Populating the blob deletion queue

These procedures are invoked immediately before or after populating the blob deletion queue.

Pre-operation

```
PROCEDURE wgn_pre_pop_blob_deletion
(date_range IN VARCHAR2);
```

Post-operation

```
PROCEDURE wgn_post_pop_blob_deletion
(date_range IN VARCHAR2);
```

Truncating a partition

These procedures are invoked immediately before or after truncating a partition.

Pre-operation

```
PROCEDURE wgn_pre_truncate_partition
(table_name IN VARCHAR2,
date_range IN VARCHAR2);
```

Post-operation

```
PROCEDURE wgn_post_truncate_partition
(table_name IN VARCHAR2,
date_range IN VARCHAR2);
```

Exchanging a temporary table with a purged partition

These procedures are invoked immediately before or after exchanging a temporary table (holding retained events) with the partition being purged. In effect, these procedures can be invoked immediately before or after merging retained events back into the original partition.

Pre-operation

```
PROCEDURE wgn_pre_exchange_partition
(table_name IN VARCHAR2,
date_range IN VARCHAR2);
```

Post-operation

```
PROCEDURE wgn_post_exchange_partition
(table_name IN VARCHAR2,
date_range IN VARCHAR2);
```

Merging a temporary table into a historic partition

These procedures are invoked immediately before or after merging a temporary table (holding retained events) into a historic partition—a special partition for holding records that must be retained permanently. Any permanent records from the purged partition are merged into this historic partition.

Pre-operation

```
PROCEDURE wgn_pre_merge_partitions
(table_name IN VARCHAR2,
historic_partition IN VARCHAR2,
date_range IN VARCHAR2);
```

Post-operation

```
PROCEDURE wgn_post_merge_partitions
(table_name IN VARCHAR2,
historic_partition IN VARCHAR2,
date_range IN VARCHAR2);
```

Customizable SQL Statement Procedures

These customizable procedures are invoked when executing SQL statements during a purge operation. You can use them, for example, to provide additional auditing capabilities to your partitioning and purging scheme. You customize these procedures in the package body of wgn_partition_cust. The prototypes are listed below.

- This procedure is invoked immediately before executing each SQL statement.
  ```
  PROCEDURE wgn_on_stmt_start
  (stmt IN VARCHAR2);
  ```
- This procedure is invoked immediately after successfully executing each SQL statement.
  ```
  PROCEDURE wgn_on_stmt_end
  (stmt IN VARCHAR2);
  ```
- This procedure is invoked immediately after failing to execute any SQL statement. It is passed the actual statement plus the associated error.
  ```
  PROCEDURE wgn_on_stmt_fail
  (stmt IN VARCHAR2,
  sql_error IN VARCHAR2);
  ```

Managing Centera Blob Files

Be careful when you purge events stored in Centera clips. Multiple blobs (binary large object files) can be stored in a single clip, so do not delete the clip until *all* its blobs can be discarded or have been requeued for storage in Centera.

To requeue events, you must add a record referencing each event to be retained to the Wgn3EventQueue table. Typically, you add custom requeue code to the wgn_pre_blob_deletion custom stored procedure.

Example

```
PROCEDURE wgn_pre_pop_blob_deletion(date_range IN VARCHAR2)
AS
BEGIN
   EXECUTE IMMEDIATE 'INSERT INTO Wgn3EventQueue (
     EventUID, eventtimestamp, StoreType, Location
     )
     SELECT eventuid, eventtimestamp, 2 AS StoreType, bloblocation
     FROM Wgn3Event
PARTITION('||wgn_get_partition_name('WGN3EVENT',date_range)||') E
     WHERE blobtype = 2 AND (eventuid,eventtimestamp)
        IN ( SELECT eventuid,eventtimestamp FROM
'||wgn_get_tmp_tab_name('WGN3EVENT')||')'
        || ' AND E.blobtype = 2 AND E.ispermanent = 0 AND E.BlobLocation IS NOT NULL'
        || ' AND NOT EXISTS(SELECT 1 FROM Wgn3EventQueue eq
             WHERE eq.EventUID = E.EventUID AND eq.eventtimestamp =
E.eventtimestamp)';
END;
```

When the infrastructure blob purge executes, the procedure purges any blobs listed in the Wgn3BlobDeletion table but skips any blobs that also exist in the Wgn3EventQueue table. Skipping these blobs gives the requeue mechanism time to reload any requeued blobs into a new Centera clip before the original clip is deleted.

Note: Clip date range boundaries and partition date range boundaries may not exactly match. Therefore, to avoid losing blob data you may need to extend the 'requeue events' query so that it extends beyond the database partition being purged.

Diagnostic Support

If a partition-based purge fails to complete, you can use information recorded in the CMS database to locate and fix the problem, and to then restart the purge from the point where it failed. Specifically, the progress and status of purge operations are recorded in two database tables, Wgn3PurgeState and Wgn3PurgeHistory. These are created automatically when you install the CMS.

More information:

Wgn3PurgeState (see page 110)
Wgn3PurgeHistory (see page 111)
If a Purge Fails to Complete (see page 103)

Wgn3PurgeState

This table records operation details for the current purge, including the progress of each purge step and associated statement. It records the start and completion times of each statement, and its status (‘Started’, ‘Complete’ or an error message). These details are deleted from the table when the next purge starts.

Wgn3PurgeHistory

This table provides provide a permanent record of all partition-based purges, showing when each purge ran and whether it completed successfully. In effect, it provides an audit trail of CMS purge operations, recording the start and end times of all previous purges, the parameters associated with each purge, plus an indication of whether the most recent purge completed successfully or not.

The parameters associated with each purge are listed in the P1, P2 and P3 columns of the Wgn3PurgeHistory table and correspond to the parameters passed to the wgn_roll_partitions function. They indicate:

P1

The end date of the new partition added. A null entry indicates that no new partition was added.

P2

The end date of the partition that was purged.

P3

The end date for the historic partition, if used. This holds records that must be retained permanently. A null entry indicates that permanent records were retained in their original partition, not in a historic partition.

Example Purge Sequence

The example below shows a series of rolling partitions, with a new partition added as the specified partition is purged. The tables below show the total number of records plus the number of permanent records in each partition, based on two retention strategies:

Strategy A

Uses a historic partition to retain permanent records (that is, events flagged as 'Do Not Delete'). Compared to strategy B, this approach typically makes it easier to manage your database because it results in fewer table partitions.

Strategy B

Does not use a historic partition. Instead, permanent records (see above) are retained in their original partition. The disadvantage of this strategy is that over time a large number of partitions can accumulate in the database.

More information:

1. Create Partitions (see page 112)
2. Run First Purge (see page 112)
3. Run Second Purge (see page 113)
Historic Partitions (see page 94)

1. Create Partitions

This example shows a database segmented into four weekly partitions. That is, each partition holds events captured over a seven day period.

	Events per partition	
Partition	**Total**	**Permanent**
Week 4	0	0
Week 3	25,000	400
Week 2	24,000	300
Week 1	26,000	500

2. Run First Purge

When the first purge runs, the Week 1 partition is purged and the Week 5 partition added. Strategy A specifies a historic partition to retain the permanent records from the Week 1 partition; Strategy B simply retains these permanent records in their original partition. At this stage, there is no practical difference between strategies A and B.

	Strategy A	
Partition	**Total**	**Permanent**
Week 5	0	0
Week 4	23,000	300
Week 3	25,000	400
Week 2	24,000	300
Historic	500	500

	Strategy B	
Partition	**Total**	**Permanent**
Week 5	0	0
Week 4	23,000	300
Week 3	25,000	400
Week 2	24,000	300
Week 1	500	500

3. Run Second Purge

When the second purge runs, the Week 2 partition is purged and the Week 6 partition added. Under strategy A, the historic partition now contains permanent records from both the Week 1 and Week 2 partition; Strategy B simply retains these permanent records in their respective original partitions.

	Strategy A	
Partition	**Total**	**Permanent**
Week 6	0	0
Week 5	26,000	200
Week 4	23,000	300
Week 3	25,000	400
Historic	800	800

	Strategy B	
Partition	**Total**	**Permanent**
Week 6	0	0
Week 5	26,000	200
Week 4	23,000	300
Week 3	25,000	400
Week 2	300	300
week 1	500	500

Enable Multiple Concurrent Block Connections

Applicable to the CMS and gateways

To improve database performance, for example when importing large volumes of captured emails, we recommend that you reconfigure the value of the initrans property to enable multiple concurrent block connections.

When Oracle performs row level locking (an enqueue), it creates a 'slot' entry in the relevant block header. By default, there is one slot per block. This can become a contention point if multiple connections try to concurrently modify the same block. For this reason, we recommend that you increase the number of available slots for database tables associated with captured events. You also need to increase the slots in the respective table indexes.

What is a Suitable Setting for Initrans?

This depends on your CA DataMinder deployment. For example, if you are running extensive Event Import jobs, our experience in the field indicates that you should set initrans to be 'the number of Event Import instances plus two'. For further guidance, please contact CA Support at http://ca.com/support.

Which Tables and Indexes?

Table	Index
Wgn3Event	IX_WGN3EVENT_EXTERNALID IX_WGN3EVENT_EVENTTIMESTAMP IX_WGN3EVENT_UPDATETIMESTAMP PK_WGN3EVENT UK_WGN3EVENT_SEQIDM
Wgn3EA	IX_WGN3EA_TYPE PK_WGN3EA
Wgn3EventParticipant	IX_WGN3EVENTPARTY_ADDRID IX_WGN3EVENTPARTY_LOGINID IX_WGN3EVENTPARTY_TIMESTAMP PK_WGN3EVENTPARTICIPANT

Chapter 7: Database Backups

We recommend that you make a full back up of your CA DataMinder database on the CMS at least once per week, and incremental backups on a daily basis. This chapter gives an overview of the backup and restore procedures for the CMS, for both Microsoft SQL Server and Oracle database engines. First, it outlines the general CMS backup tasks.

This section contains the following topics:

General Backup Tasks

When you back up the CMS, you must perform these backup tasks irrespective of the type of database engine used on the CMS.

Back up the Data folder

For all CMSs. This folder holds all the configuration data and captured data used by your CA DataMinder installation. We recommend you back up this folder a minimum of once a week, but a daily backup is preferential. By default, when you install CA DataMinder this folder is added as a Data subfolder in the installation folder, but you can rename it and locate it anywhere suitable on your network.

There are two subfolders inside the Data folder which require special handling in the backup:

cache

This subfolder stores cache files of all processed data. It does not need to be backed up.

e

This subfolder stores all captured data. We recommend that you carry out a partial back up of this subfolder.

The e subfolder needs special handling because of its potentially large size. You can either back up the entire folder, or back up selected events by date range using the dated folder names. Typically, you may only need to back up BLOB files associated with events from the last 2-7 days.

It is good practice to have a separate strategy in place for backing up BLOB files associated with events more than a week old. For example, to complete a full backup of the e subfolder once a month. Assuming that the daily backup collects data from the last seven days, then a monthly backup means that no more than 1 full restore and at most 4 daily backups are required to restore the event folder to any day within the last month.

Back up the master encryption key

The CMS uses a password-protected key to provide highly secure data management. If you need to restore the CMS, you will need to restore the key. For this purpose, CA DataMinder provides a data management utility for exporting and re-importing these keys.

For full details about these general backup tasks, see the *Platform Deployment Guide*; search for 'backups'.

Backup Tasks for Oracle

If you use Oracle as your database engine, the backup procedure has three steps. First, you must back up essential registry keys on your CMS. Second, you must set up a backup schedule for your Oracle database. Third, you must add the CA DataMinder data folder into your existing backup regime.

1. **Back up the master encryption key**

   See the general backup tasks.

2. **Cold backups only: Suspend the CMS**

   If you are running a cold backup, you must first suspend the CMS. See the next section for details. Note that we do not recommend this method.

3. **Back up your Oracle database**

   Note: This section describes the basic procedure for backing up an Oracle database. Experienced Oracle administrators may prefer other methods for configuring backup jobs.

   a. In the Oracle DBA Studio, launch the Backup wizard and choose your backup strategy. This can be predefined, or you can define a custom strategy.

   b. Next define the backup frequency. A typical CA DataMinder database will be large and frequently updated, so we recommend that you choose a full back up of your CMS every week, with incremental backups every day.

   c. Specify what time the backups run. You can monitor the status of completed backup jobs in the Oracle Enterprise Manager Console.

   d. Specify the backup configuration. If you have already defined a custom backup configuration, you can choose that configuration now.

   e. Finally, choose the destination database for the backup job. In this case, submit the job to the CA DataMinder database.

4. **Back up your CA DataMinder data folder**

   See the previous section, ‘Back up the Data folder: For all CMSs. This folder holds all the configuration data and captured data used by your CA DataMinder installation. We recommend you back up this folder a minimum of once a week, but a daily backup is preferential. By default, when you install CA DataMinder this folder is added as a Data subfolder in the installation folder, but you can rename it and locate it anywhere suitable on your network.’.

5. **Cold backups only**

   Resume the CMS

More information:

Backup Tasks for SQL Server (see page 119)
General Backup Tasks (see page 116)
Cold Backups (see page 118)

Cold Backups

We do not recommend cold backups. However, if you intend to run cold backups of your Oracle database, you must suspend the CMS before starting the database backup and restart it when the backup is complete. This prevents error messages being written to the CA DataMinder Activity log (viewable in the Administration console).

To suspend or resume the CMS

1. In the Administration console, expand the My Servers folder and select the CMS whose database you are about to backup.
2. Expand the Machine Administration tree and right-click the CMS.
3. Right-click the CMS and click Suspend or Resume.

Backup Tasks for SQL Server

If you use SQL Server as your database engine, the backup procedure has four steps. First, you must back up essential registry keys on your CMS. Second, you must set up a maintenance plan for your system databases. Third, you set up a similar plan for your CA DataMinder database. Finally, you must add the CA DataMinder data folder into your existing backup regime.

1. **Back up the data management registry key**

   See the general backup tasks.

2. **Set up a maintenance plan for your system databases**

   If your organization already uses SQL Server databases for other purposes unrelated to CA DataMinder, you have probably already set up a maintenance plan for your system databases. If not, you must up set up a maintenance plan for your system databases that incorporates a backup plan.

   a. In the SQL Server Enterprise Manager, expand the Management branch of your CMS and launch the Database Maintenance Plan Wizard.

      Note: The following steps describe where specific settings are required for a successful CA DataMinder backup. If a wzard screen is not mentioned, you can safely assume that settings on that screen do not affect CA DataMinder backups.

   b. Create a maintenance plan for all system databases (master, model and msdb).

   c. Schedule a database integrity check.

      Important! The Integrity Check wizard screen includes an option to 'Attempt to repair any minor problems'. Because of a known SQL Server issue, do not select this option because it may cause related backup jobs to fail. For details, see the MS Knowledge Base article Q290622.

   d. Create a database backup plan. You need to:

      - Verify the integrity of the completed backup.
      - Choose a backup location (tape or disk) for the backup file.
      - Schedule when the backup runs.

   e. When prompted by the Wizard, do not specify a transaction log backup plan for your system databases.

      Important! You will specify a transaction log backup plan for your CA DataMinder database (see next section, step 3.4). Because of a known SQL Server issue, do not specify a transaction log backup plan for your system databases. For details, see the MS Knowledge Base article Q285288.

   f. In the remaining screens of the Database Maintenance Plan Wizard, there are no specific settings needed for CA DataMinder backups.

3. **Set up a maintenance plan for your CA DataMinder database**

The procedure is the same as for your system databases, except this time the maintenance plan does include a transaction log backup plan. See the following steps for details.

a. Select your CMS in the SQL Server Enterprise Manager and launch the Database Maintenance Plan Wizard. As before, the following steps highlight only those areas where specific settings are required for a successful CA DataMinder backup.

b. Schedule a database integrity check.

c. Create a database backup plan. You need to:

   - Verify the integrity of the completed backup.
   - Choose a backup location (tape or disk) for the backup file.
   - Schedule when the backup runs. We recommend that you back up your CMS every week and that the backups for your system databases and CA DataMinder database not overlap.

d. When prompted by the Wizard, do specify a transaction log backup plan for your system databases.

   Note: When you schedule the transaction log backup plan, make sure that it does not overlap with the scheduled backup of your CA DataMinder database.

e. In the remaining screens of the Database Maintenance Plan Wizard, there are no specific settings needed for CA DataMinder backups.

4. **Back up your CA DataMinder data folder**

More information:

General Backup Tasks (see page 116)

Index

R

S

T

U

W