Universal CMDB

Software Version: 10.33

Database Guide

Document Release Date: July 2017
Software Release Date: July 2017

Hewlett Packard Enterprise

Documentation Updates

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hpe.com/.

This site requires that you register for an HPE Passport and to sign in. To register for an HPE Passport ID, click **Register** on the HPE Software Support site or click **Create an Account** on the HPE Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HPE sales representative for details.

Support

Visit the HPE Software Support site at: https://softwaresupport.hpe.com/.

This website provides contact information and details about the products, services, and support that HPE Software offers.

HPE Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support website to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HPE support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HPE Passport user and to sign in. Many also require a support contract. To register for an HPE Passport ID, click **Register** on the HPE Support site or click **Create an Account** on the HPE Passport login page.

To find more information about access levels, go to: https://softwaresupport.hpe.com/web/softwaresupport/access-levels.

HPE Software Integration Catalog accesses the new HPE Software Integrations and Solutions Catalog website. This site enables you to explore HPE Product Solutions to meet your business needs, includes a full list of Integrations between HPE Products, as well as a listing of ITIL Processes. The URL for this website is https://softwaresupport.hpe.com/km/KM01702731.

Contents

Part I: Introducing the Database Environment

Chapter 1: Introduction to Preparing the Database Environment

This chapter includes:

Databases in Use Overview

To work with Universal CMDB, you must set up the CMDB database. The CMDB database is used for storage of configuration information that is gathered from the various Universal CMDB and third-party applications and tools. This information is used when building Universal CMDB views.

You can set up the Universal CMDB databases on a Microsoft SQL Server, on an Oracle Server, or on a PostgreSQL Server, depending on the type of database server used in your organization.

If you are working with a Microsoft SQL Server database, refer to "Deploying the Microsoft SQL Server Database" on page 17.

If you are working with an Oracle Server database, refer to "Deploying the Oracle Server Database" on page 37.

If you are working with a PostgreSQL Server database, refer to "Deploying the PostgreSQL Server Database" on page 64.

> **Note:**
>
> - It is strongly recommended to host database server (Oracle, Microsoft SQL, or PostgreSQL) on a **physical machine**, and it should be an independent server without other applications

(including the UCMDB server) running on it.

Also, if the database server machine is a virtual machine, the resource MUST be dedicated for the database server.

- Apart from the embedded PostgreSQL database server, installing UCMDB server and database server (Oracle, Microsoft SQL, or PostgreSQL) together on the same machine is not supported.
- Database servers must be set to the same time zone, daylight savings settings and time as the Universal CMDB servers.
- For details on working in a non-English language Universal CMDB environment, see the section on non-English locales in the interactive *HPE Universal CMDB Deployment Guide*.
- The UCMDB server(s) should be located in the same LAN with the database servers (without a proxy and firewalls between them). Otherwise, your system's performance may be impacted.

Universal CMDB Sizing

Universal CMDB database configuration requirements are dependent on the amount of data, as well as on the runtime load, generated by Universal CMDB. A small Universal CMDB deployment requires a database that can hold up to 1.5 million CMDB objects and links. A large Universal CMDB deployment requires a database containing 1.5 million or more CMDB objects and links.

For more details, see the *HPE Universal CMDB Sizing Guide*.

Hardware Requirements

The following table describes the minimum hardware (CPU and memory) requirements recommended for the Universal CMDB Oracle, Microsoft SQL, or PostgreSQL database server:

Deployment	CIs and Relationships	Number of Processors	Physical Memory
Small	< 2 million	Minimum 1 dual core	8 GB RAM
Standard	2 - 10 million	Minimum 2 dual core or 1 quad core	8 GB RAM

Deployment	CIs and Relationships	Number of Processors	Physical Memory
Large	10 - 40 million	Minimum 2 dual core or 1 quad core	16 GB RAM
	40 - 60 million	2 x 4 cores	32 GB RAM
	60 - 120 million	2 x 8 cores	64 GB RAM
	> 120 million	2 x 8 cores	128 GB RAM

Note: It is strongly recommended to host database server (Oracle, Microsoft SQL, or PostgreSQL) on a **physical machine**, and it should be an independent server without other applications running on it.

Also, if the database server machine is a virtual machine, the resource MUST be dedicated for the database server.

UCMDB Server Connection Retries to the Database

Configure UCMDB Server Connection Retries to the Database

By default UCMDB server tries to connect to the database for 10 times when there is a network problem.

If you need to increase the number of the retries, do the following:

1. Open the **<UCMDB_Server_Home>\bin\wapper.conf** file using a text editor.
2. Locate the following entry in the **wapper.conf** file and increase the number of startup retries to a desired value:

   ```
   # Number of startup retries before stopping the service
   wrapper.max_failed_invocations=10
   ```

3. Save the file.
4. Restart the UCMDB server for the change to take affect.

   In case of HA, restart the UCMDB server cluster.

Configure UCMDB Server Connection Pool Connections Validations

There are several best practices used regarding database connections. The most popular approach uses a connection pool to interact with the database as opposed to creating and closing the connection every time a request to UCMDB database is needed. Making a connection every time is very costly on the way UCDMB operates. Because of this reason UCMDB uses a connection pool, Apache DBCP Component, which is a third party library used. Once the pool is operating, every time a UCMDB component needs a connection to database it takes one from the pool that is already established. After using the connection it is returned to the pool.

One role of the connection pool is to ensure that the connections are opened and can be used at any time. There are several ways the Apache DBCP can achieve this. One is using validation queries which means a select statement is run against the database server. For example, “`select 1 from dual`" is the Oracle specific validation query.

The latest database drivers provides the feature of testing the database connection without running a validation query. UCMDB uses this mechanism by default.

In case the old mechanism is needed to be used, you can configure it as follows:

1. Open the **<UCMDB_Server>\conf\connection_pool.conf** file using a text editor.
2. Locate the **useValidationQuery** setting and set its value to **1**, as follows:

   ```
   useValidationQuery=1
   ```

   Any other value will use the default mechanism.
3. Save the **connection_pool.conf** file.
4. Restart the UCMDB server.

How to Modify Composite Indexes

A composite index contains multiple key columns. You can invoke the **modifyCompositeIndexes** JMX method to add or remove the **CMDB_ID** column as a key column in the indexes for the specified class. The method then modifies the indexes according to the specified parameters.

Note:

- This method only works for a Microsoft SQL or Oracle database.
- When you perform a fresh install of the UCMDB 10.30 (or later), by default there are no composite indexes with the **CMDB_ID** as a key column on Oracle database (**ROOT** tables),

and on Microsoft SQL databases (**ROOT** and **CDM** tables). To create the composite indexes with the **CMDB_ID** as a key column, follow the steps described in this procedure.

It is highly recommended that you follow these best practices for running the **modifyCompositeIndexes** method:

- Run the method when the probes are stopped and the database is not heavily used.
- Create a schema dump before running the method.
- For environments that execute heavy data-in operations, in order to speed up the INSERT statements, it is recommended to transform the database from composite index to non composite index by invoking the **modifyCompositeIndexes** method with **composite indexes** setting to **false**.

To modify composite indexes:

1. On the UCMDB server, launch the Web browser and enter the following address: **https://localhost:8443/jmx-console**

   You may have to log in with a user name and password.

2. Locate **UCMDB:service=Dal Services** and click the link to open the JMX MBean View, and then locate the **modifyCompositeIndexes** method.
3. Specify the following parameters of the method:
   - **customerId**: Specify the Customer ID (leave empty for the default customer)
   - **class name**: Specify the class name (allowed values are `root` and `all`)
   - **composite indexes**: Specify if the method rebuilds the indexes with **CMDB_ID** as a key column
     - `True`: **CMDB_ID** is a key column
     - `False`: **CMDB_ID** is not a key column
4. Click **Invoke**.

Note: If any problem occurs, the operation rolls back and causes no change in the database.

How to Reconfigure UCMDB to Use Another Schema/Database

To reconfigure UCMDB to use another schema or database,

1. Stop all the Data Flow Probes connected to the UCMDB Server.
2. Stop the UCMDB Server.
3. Make sure the existing schema/database server is reachable.
4. Run the Configuration Wizard.

   Windows: Start > All Programs > UCMDB > Start Universal CMDB Server Configuration Wizard

   Linux: Run the **<UCMDB_Server_InstallDir>/bin/configure.sh** script
5. Follow the wizard steps.
6. Start the UCMDB Server.
7. After the UCMDB Server has started, start all the Data Flow Probes.

Large Capacity Planning for UCMDB

This section includes:

- Large Capacity Planning Overview
- Configuring the UCMDB Server for Large Capacity
- Configuring the Oracle Database for Large Capacity
- Configuring the Microsoft SQL Database for Large Capacity
- System Capacity Test

Large Capacity Planning Overview

Using the default configuration, HPE Universal CMDB can work with a deployment of more than 25 million CIs and relationships. To work with a larger deployment, you must implement the following configuration:

Depending in the number of CIs and relationships, increase the CMDB heap as follows:

# CIs and Relationships	Heap Size
≤ 40 million	12 GB
40 million – 60 million	16 GB
60 million – 125 million	24 GB
> 125 million	55 GB

- For capacity planning requirements, see the *HPE Universal CMDB Support Matrix* document.
- For details about the changes you must make to the system configuration to support this capacity, see Configuring the UCMDB Server for Large Capacity below.
- For details on how to improve performance, see Configuring the Oracle Database for Large Capacity and Configuring the Microsoft SQL Database for Large Capacity below.
- For details about the setup used for capacity testing and performance results, see Testing System Capacity below.

Configuring the UCMDB Server for Large Capacity

For the system to support the desired number of CIs and relationships, update the following parameters on the UCMDB Server:

Parameter	Default	CIs and Relationships (million)				Location
		≤ 40	**40 – 60**	**60 – 125**	**> 125**	
wrapper.java. initmemory	1024	2048	8192			• Windows: **C:\hp\UCMDB\ UCMDBServer\bin\ wrapper-platform.conf**

Parameter	Default	CIs and Relationships (million)				Location
		≤ 40	40 – 60	60 – 125	> 125	
						• Linux: **/opt/hp/UCMDB/UCMDBServer/bin/wrapper-platform.conf**
wrapper.java.maxmemory	4096	8192	16384	24576	56320	
dal.object.condition.max.result.size	2000000	50000000	50000000			• Windows: **C:\hp\UCMDB\UCMDBServer\conf\settings.override.properties** • Linux: **/opt/hp/UCMDB/UCMDBServer/conf/settings.override.properties**
dal.use.memory.instead.temp.table.high.threshold.oracle	600000	6000000	6000000	10000000		
dal.joinf.max.result.size	400000	4000000	4000000			

Configuring the Oracle Database for Large Capacity

When working on a system containing more than 40 million objects and relationships, you can improve performance by increasing the Oracle SGA and PGA to the following suggested sizes:

CIs and Relationships	SGA	PGA
40 million – 60 million	22 GB	6 GB
60 million – 120 million	42 GB	14 GB
> 120 million	88 GB	24 GB

This improves the performance of both the TQL calculation for several types of TQL queries, as well as for data-in operations performed on the system.

Configuring the Microsoft SQL Database for Large Capacity

When working on a system containing more than 40 million objects and relationships, you can improve performance by increasing the Microsoft SQL Server Memory to the following suggested sizes:

CIs and Relationships	Microsoft SQL Server Memory
40 million – 60 million	28 GB
> 60 million	56 GB

System Capacity Test

Setup

The system capacity test is conducted for Microsoft SQL Server (with 125 million CIs and relationships) and Oracle Database (with 200 million CIs and relationships) separately, by using the following hardware configurations.

Microsoft SQL Server

Role	CPU	Memory	OS + 3rd Party SW
CMDB	2 x 4-cores @ 2.67 GHz	32 GB	Microsoft Windows Server 2008 R2 Enterprise Edition x64 SP1
Database	2 x 8-cores @ 2.93 GHz	64 GB	• Microsoft Windows Server 2008 R2 Enterprise Edition x64 SP1 • Microsoft SQL Server 2014 - 12.0.2000.8 (x64)

Oracle Database

Role	CPU	Memory	OS + 3rd Party SW
CMDB	2 x E5-2630V3 @ 2.40GHz	32 GB	Red Hat Enterprise Linux 7.2
Database	2 x E5-2630V3 @ 2.40GHz	97 GB	• Oracle Linux 7.2 • Oracle Database 12c Enterprise Edition x64 - 12.1.0.2

The following business flows were tested as part of the system test:

- **TQL Calculation**

  TQLs were divided into sub groups according to the result size (<100, <1000, and <10000), according to the data set that the TQL retrieves, and according to the TQL configuration:

  - Like Condition
  - Like Ignore case
  - Different number of hierarchies in the TQL results (2-5)
  - Compound
  - Sub-graph
- **Data-in**

  The data-in scenario in the system test included insertion, updates, and deletion.
- **Enrichments**

  Enrichment scenarios included insert, update, and delete.

Results

Following the load test in the scenario that includes query execution (2 days), data-in (7 days for Oracle and 10 days for MS SQL), and enrichment execution, the following results were achieved:

- The system was stable throughout the run. No restarts, memory leaks, or any other degradation over time was observed.
- System performance was acceptable.

Part II: Deploying the Microsoft SQL Server Database

Chapter 2: Overview of Microsoft SQL Server Deployment

This chapter includes:

About Microsoft SQL Server Deployment

To deploy Microsoft SQL Server for use with Universal CMDB, you must perform the following procedures:

- **Install and configure Microsoft SQL Server.**

  For details on installing and configuring Microsoft SQL Server, see the Microsoft SQL Server documentation for your version of Microsoft SQL Server.

- **Create databases on Microsoft SQL Server.**

  You can create Universal CMDB databases manually or you can use the UCMDB Server Configuration Wizard to create the databases (**Start > All Programs > UCMDB > Start Universal CMDB Server Configuration Wizard**). For details on manually creating databases for Universal CMDB, see "Manually Creating the Microsoft SQL Server Databases" on page 21.

For details on creating a Microsoft SQL database, see "Creating and Configuring the Microsoft SQL Server Database" on page 23.

System Requirements

This section describes the system requirements for working with Microsoft SQL Server in conjunction with Universal CMDB.

This section includes the following topics:

"Hardware Requirements" below

"Software Requirements" below

Hardware Requirements

For Universal CMDB hardware sizing guidelines, see "Hardware Requirements" on page 8. For Microsoft SQL Server hardware requirements, refer to the installation guide for your Microsoft SQL Server release for your operating system.

Software Requirements

For details on supported versions of Microsoft SQL Server, see the Server Database Requirements section of the *HPE Universal CMDB Support Matrix*.

Installation Prerequisites

The following prerequisite must be met before installing Microsoft SQL Server:

- If you want to install Universal CMDB using the installation wizard, you must provide a user account that has database creator privileges. If you are going to create the database manually, supply Universal CMDB with a user account that is a part of **db_datareader**, **db_datawriter** and **db_ddladmin** in the Universal CMDB database.

> **Note:** When working with Microsoft SQL Server, the CMDB collation and the SQL Server collation must be the same.

Checklist for Support and Certification

Information is provided in this section for both supported and certified Microsoft SQL Server options.

The certified options are recommended for working with Universal CMDB. Certified options are rigorously tested by HPE quality assurance personnel. Supported options are those options for which HPE quality assurance personnel have successfully performed basic tests.

Subject	Microsoft SQL Server	
	Supported	**Recommended**
Instances	Default, Single	
Authentication Mode	Mixed, Windows	
Collation	Case-insensitive. Universal CMDB does not support binary sort order and case sensitivity. Only case-insensitive order with a combination of accent, kana, or width settings is supported .	Use the Collation Settings dialog box to select the collation. Do not select the **binary** check box. Accent, kana, and width sensitivity should be selected according to the relevant data language requirements. The selected language must be the same as the OS Windows regional settings language.
Network Libraries	**Server:** TCP/IP and Named Pipes **Client:** TCP/IP and Named Pipes	**Server:** TCP/IP **Client:** TCP/IP
Server Configuration Options	Defaults, unless instructed otherwise	
Data File Properties	Manual file growth, or FILEGROWTH less than or equal to 100 MB	FILEGROWTH: ~30-100 MB
Collation Database Property	Server default	
Database Options	Defaults, unless instructed otherwise	
Recovery Model	Any	Full

Note: For Turkish, the recommended collation is Turkish_100_CI_AS.

Chapter 3: Manually Creating the Microsoft SQL Server Databases

This chapter includes:

Overview of the Microsoft SQL Server Databases

Universal CMDB uses the configuration management database (CMDB) for its persistency.

During the Universal CMDB setup, the new database can be set up automatically by the setup procedure, or an existing database can be used. An existing database can either be created manually in advance (for example, due to organization security restrictions), or can be created by a previous installation of the same release of Universal CMDB.

For details about installing the Universal CMDB server, see the interactive *HPE Universal CMDB Deployment Guide*.

Manually Creating the Universal CMDB Microsoft SQL Server Database

Before creating the database, you must install the Microsoft SQL Database Server according to the Microsoft SQL Server documentation.

If the Universal CMDB database is manually created prior to running the HPE Universal CMDB setup, during the setup procedure, you select the option to use an existing database and enter the relevant data about the existing database for which you are prompted. The data you enter is written to the configuration files that are used by the JDBC driver to connect to the existing database.

Creating the Database

The database administrator should create a Microsoft SQL Server database for the CMDB data. No scripts are necessary to create the objects to populate this database.

To create a database, you must have CREATE DATABASE permissions. To connect to an existing database, the login account with which you are connecting must be mapped to dbo in the database.

> **Note:** Members of the sysadmin server role automatically have CREATE DATABASE permissions, and are also mapped to dbo in all databases. A database owner is automatically mapped to dbo in the database.

For details on creating the database manually, see "Creating and Configuring the Microsoft SQL Server Database" on page 23.

> **Note:** When the Universal CMDB setup procedure automatically creates the database, it uses two file groups for each database; one to hold the system tables and one for the application data.

Chapter 4: Creating and Configuring the Microsoft SQL Server Database

This chapter includes:

Creating the Database

This section includes the following topics:

"Database Permissions" below

"Database File Layout" on the next page

"System Databases" on page 27

Database Permissions

To create a database, you must have CREATE DATABASE permissions. To grant CREATE DATABASE permissions to a user, the user’s login must first be mapped to a database user in the master database.

Note: Universal CMDB login accounts should be mapped to dbo in the database. Members of the sysadmin server role automatically have CREATE DATABASE permissions, and are also mapped to dbo in all databases. A database owner is automatically mapped to dbo in the database.

To check whether a user has CREATE DATABASE permissions, log in to Management Studio with the login account of the user whose permissions you want to check, and run the following:

```
USE master
IF PERMISSIONS() & 1 = 1
PRINT ‘User has CREATE DATABASE permissions.’
```

To check whether a user has enough permissions in the database, log in to Management Studio with the login account of the user whose mapping you want to check. Change the database context to the required database, and run the following command:

```
select case when IS_MEMBER ('db_owner')=1
or IS_SRVROLEMEMBER ('sysadmin')=1
or (IS_MEMBER ('db_ddladmin') = 1 and
IS_MEMBER ('db_datareader')=1 and
IS_MEMBER ('db_datawriter')=1 and
IS_MEMBER ('db_denydatareader')=0 and
IS_MEMBER ('db_denydatawriter')=0 )
then 'User has enough permissions'
else 'User does not have enough permissions'
end
```

Database File Layout

When you create a database, it must consist of at least one data file (with an .mdf extension) and one transaction log file (with an .ldf extension). You can optionally create additional data files (.ndf), as well as additional log files (.ldf).

To enhance performance, you may want to create several data files. Microsoft SQL Server stripes the data among the data files, so that if you do not have RAID controllers that stripe your data, you can spread the data files over several regular physical disks and, in this way, have the data striped. The log, however, is read sequentially, so that there is no performance gain in adding more log files. An additional log file should be created on a different disk when your existing log is out of disk space.

Data and Log Placement

> **Caution:**
>
> - It is recommended not to place data or log files on the same disk that stores the page (swap) file.
> - It is recommended that you place the data and log files on separate disk subsystems.

- **Log files.** Changes are not flushed to the database until they are written to the log, and the log architecture dictates serial writes, so it is advisable that there be as little interference as possible with the log activity. It is usually sufficient to place the log on a RAID 1 system because of the serial writes to the log. If you have processes reading from the log (for example, triggers accessing the inserted and deleted views which are formed from the log records or transactional replication), or several log files for different databases, consider placing the log file(s) on a RAID 0+1 (striped mirror) system.

- **Data files.** Data files should be placed on a RAID 0+1 system for optimal performance.

File and Database Properties

When you create a database you can specify the following properties for each file (.mdf, .ndf, .ldf):

- **NAME.** The logical file name which you can use later when you want to alter one of the properties.
- **FILENAME.** The physical file path and name. Make sure the destination directory is not compressed (right-click the directory in Windows Explorer, select **Advanced**, and verify that the **Compression** check box is not selected).
- **SIZE.** The initial file size.
- **MAXSIZE.** Determines the maximum size to which the file can grow. If this argument is omitted, or if you specify **Unlimited**, the file can grow until the disk is full.
- **FILEGROWTH.** The automatic growth increment of the file. This argument can be specified as either a percentage of the existing file size, or as a fixed size.

  An autogrowth operation invoked by a modification sent by a client that timed out cannot be completed successfully. This means that the next time a client sends a modification, the autogrowth process starts at the beginning and may also time out.

  To avoid this problem, it is recommended that you either expand the files manually every time the database nearly reaches full capacity (for example, 20 percent free), or set the growth increment to a fixed size that takes less time to be allocated than the client's timeout setting. Using a small growth increment is not recommended because it increases file system fragmentation. On the other hand, if you use a very large increment, modifications sent by clients might incur connection timeouts while waiting for the automatic expansion to finish. For large databases, a percentage growth increment can lead to exponential growth of the database and should be avoided.

  For more information on this problem, refer to Microsoft Knowledge Base Article - 305635 (http://support.microsoft.com/kb/305635).

- **ALLOW_SNAPSHOT_ISOLATION and READ_COMMITTED_SNAPSHOT.** Set the database properties **ON** for **ALLOW_SNAPSHOT_ISOLATION** and **READ_COMMITTED_SNAPSHOT** with the following script:

  ```
  ALTER DATABASE [${dbName}] SET ALLOW_SNAPSHOT_ISOLATION ON

  ALTER DATABASE [${dbName}] SET SINGLE_USER WITH ROLLBACK IMMEDIATE

  ALTER DATABASE [${dbName}] SET READ_COMMITTED_SNAPSHOT ON
  ```

```
ALTER DATABASE [${dbName}] SET MULTI_USER
```

> **Note:**
>
> - The above database properties MUST be set for UCMDB database even if the database is not on AlwaysOn.
> - If the database is set on AlwaysOn, the above database properties must be set before you add the database on AlwaysOn.

The tempdb Database Settings

The frequent expansion of the tempdb system database can affect the database's performance, especially in large Microsoft SQL Server installations. The size of the tempdb, therefore, should be large enough to avoid the need for early expansion. Its growth increment should be large enough to avoid fragmentation, yet not too large to expand in a reasonable amount of time. Create the tempdb with a minimum, initial size of 1 GB and with a growth increment of 50 MB. The tempdb database should be striped across several disks, ideally on a RAID 0+1 controller. It is recommended to move the tempdb database to its own set of disks.

To ensure that there is enough disk space for the tempdb to grow during times of heavy usage (for example, when aggregating or sorting data), it is recommended that you leave at least 20 GB free disk space on the drive where the tempdb is located.

File Groups

File groups are logical groupings of data files. Each of the following objects can be placed in its own file group unit:

- A table's data
- A table's large objects (text, ntext, image columns)
- An index

Data is inserted proportionally into all files belonging to the file group in which the object is stored, according to the amount of free space in each file. The **.mdf** file is placed in a file group called **PRIMARY**, which is marked as **Default** when the database is created (the default file group for objects when no file group is specified). If you do not place other data files (**.ndf** files) in their own file groups, they are also placed in the **PRIMARY** file group. Note that you can change the **Default** file group later on.

File groups can be used for performance tuning or maintenance. For details, see Microsoft SQL Server Books Online at http://www.microsoft.com/downloads.

Following is an example that demonstrates how to use file groups for maintenance:

- **Partial Restoring.** Microsoft SQL Server does not support the restoration of a single table. Even if you place a single table in a file group, you cannot restore a file group to a point in time earlier than the rest of the data. Instead, you must apply all log file backups in order to synchronize the file group with the rest of the data. Microsoft SQL Server supports partial restoration to a database with a different name. A partial restoration allows you to restore a single file group, and supports point-in-time restoration. However, you must restore the PRIMARY file group because it contains the SYSTEM tables.

  To be able to restore a single table to a point in time if a logical error occurs, you need to design the file groups in your database as follows:

  - Ensure that the **.mdf** file is the only file in the **PRIMARY** file group.
  - Place each large table in its own file group.
  - Place all small tables in a separate file group.

System Databases

The following system databases are especially important for the smooth performance of Microsoft SQL Server:

- **tempdb.** Numerous Microsoft SQL Server activities—such as creating local and global temporary tables, creating work tables behind the scenes to spool intermediate query execution results, and sorting—implicitly or explicitly use the tempdb system database.

  If your system is not configured properly, the tempdb database can become a performance bottleneck, so it is very important to determine the tempdb database's original size correctly.

  For more information on setting database sizes, see "The tempdb Database Settings" on the previous page.

  To move tempdb's files, use the ALTER DATABASE tempdb MODIFY FILE command, and restart Microsoft SQL Server.

- **master, msdb, model.** These databases, although crucial for the operation of Microsoft SQL Server, are smaller than tempdb because they store only meta data.

  It is strongly recommended to use a fault tolerant disk—ideally, RAID 1—for these databases.

> **Note:** For Universal CMDB certification, place system databases on fault tolerant disks. It is recommended to use RAID 1 disks.

To check the database's properties, run the following:

```
EXEC sp_helpdb <database name>
```

Configuring the Database

Once you have created the necessary databases, you can add new files to the databases, change some of the existing database file properties, and set the database configuration options appropriately.

This section includes the following topics:

Database File Configuration

You can change certain database file properties, as well as add or drop files using either of the following methods:

- use the Properties dialog box in Management Studio
- use the ALTER DATABASE command (for details, see Microsoft SQL Server Books Online at http://www.microsoft.com/downloads)

Adding Files

Data files can be added to an existing file group in a database, or to a new file group. There are no special restrictions or requirements.

Dropping Files

To drop a file, you must first empty it using the DBCC SHRINKFILE command’s EMPTYFILE option, which transmits the file data to all the other files in the file group. Once you empty the file, you can use the ALTER DATABASE <database name> DROP FILE command to drop it.

Changing File Properties

You can change the size-related properties for all databases, as well as the filename property for the tempdb database (this takes effect after you restart Microsoft SQL Server). The SIZE, MAXSIZE, and FILEGROWTH properties can be changed using the ALTER DATABASE tempdb MODIFY FILE command. Note that the SIZE property can only be enlarged.

To shrink the file, use the DBCC SHRINKFILE command. For details and recommendations concerning file properties, see "Creating the Database" on page 23.

Database Configuration Options

Each database contains a set of configurable options that determine its behavior. You can view or change the database options using any one of the following utilities:

- the Options tab in the Management Studio's Properties dialog box
- the EXEC sp_dboptions stored procedure
- the ALTER DATABASE <database name> SET command

Note: Not all of the database configuration options are available in this dialog box.

The following table lists, in alphabetical order, the default configuration options, as well as the configuration settings required for Universal CMDB certification:

Configuration Option	Description	Default	Universal CMDB Certification in Microsoft SQL Server
ANSI NULL default (see note below)	Specifies whether the database columns are defined as NULL or NOT NULL, by default	Not set	Not set
ANSI PADDING	Controls the way the column stores values shorter than the defined size of the column and the way the column stores values that have trailing blanks in char, varchar, binary, and varbinary data.	OFF	ON
Auto close	Specifies whether the database shuts down after its resources are freed and all users exit	Not set	Not set **Note:** If set, it may take a long time for the database to allocate

Configuration Option	Description	Default	Universal CMDB Certification in Microsoft SQL Server
			resources every time a user connects, after the database is closed.
Auto create statistics	Specifies whether missing statistics required by a query for optimization are built automatically during optimization	Set	Set
Auto shrink	Specifies whether the database is automatically shrunk every hour, leaving 25% of free space	Not set	Not set **Note:** If set, constant growth/ shrinkage may cause file system fragmenta tion.
Auto update statistics	Specifies whether out-of-date statistics required by a query for optimization are built automatically during optimization	Set	Set
Compatibility level	The version of Microsoft SQL Server that the database appears to be (for the application)	The same version as the release installed	The same version as the release installed
Read only	Database is read only	Not set (READ_ WRITE)	READ_ WRITE
Recovery	The database recovery model determines the recovery capabilities by controlling the	Full	Full (unless you are certain that the lower recovery

Configuration Option	Description	Default	Universal CMDB Certification in Microsoft SQL Server
	amount of bulk operation logging (such as Select into, Bulk, Insert, Create index, LOB manipulation). The higher the recovery model, the higher the recovery capabilities. However, the amount of logging also increases, which may affect performance.		capabilities are sufficient for your system)
Recursive triggers	Specifies whether recursive triggers are supported	Not set	Not set
Restrict access	Only single users or members of the db_owner, dbcreator, or sysadmin groups can access the database.	Not set (MULTI_ USER)	MULTI_USER
Torn page detection	Specifies whether incomplete pages can be detected	Set	Set
Truncate log on checkpoint	Automatically marks inactive portions of log for reuse on checkpoint	Not set	N/A
Use quoted identifiers	Specifies whether the Microsoft SQL Server enforces ANSI rules regarding quotation marks. Select this option to specify that double quotation marks be used only for identifiers, such as column and table names. Note that character strings must be enclosed in single quotation marks.	Not set	Not set

Note:

Not all ANSI options can be set using Management Studio. The ANSI database configuration options include: ANSI_NULLS, ANSI_NULL_DEFAULT, ANSI_PADDING, ANSI_WARNINGS, ARITHABORT, CONCAT_NULL_YIELDS_NULL, NUMERIC_ROUNDABORT, and QUOTED_ IDENTIFIER.

The options you set may not take effect, since these options can also be set at a higher level.

> For example, if the session option **QUOTED_IDENTIFIER** was turned **on**, the equivalent database configuration option is irrelevant. Some tools or database interfaces turn certain session options on or off, so that the database configuration options never take effect.

The following table summarizes the characteristics of each recovery model:

Model/ Support	Allows Log Backup	Allows Point-in-Time/Log Mark Restoration	Allows Backup Log when Data Crashes (Saves changes until the crash point)	Amount of Bulk Operation Logging (can affect the performance of bulk operations)
Simple	No	No	No	Minimal
Bulk Logged	Yes	No	No	Minimal
Full	Yes	Yes	Yes	Full

To check your database's properties, run the command:

```
EXEC sp_helpdb <database name>
```

For information on SQL databases, see Microsoft SQL Server Books Online at http://www.microsoft.com/downloads.

Composite Indexes vs. Non Composite Indexes

For environments that execute heavy data-in operations, in order to speed up the INSERT statements, it is recommended to transform the database from composite index to non composite index.

You can achieve this by invoking the **modifyCompositeIndexes** JMX method with **composite indexes** setting to **false**. For detailed instructions, see "How to Modify Composite Indexes".

Monitoring the Fragmentation Indexes

Due to the fact that UCMDB is an OLTP application and performs many insert, update, and delete operations each day, its indexes might become fragmented. The index fragmentation could be even higher if discovery jobs that run also modify the data.

To determine the indexes' fragmentation, run the following script every day. This script could be scheduled as part of the SQL job. The result of the script contains all the indexes having average fragmentation larger than 15 percent. When you have data from several days, you should compare the results and determine if the fragmentation index continue to increase day by day.

If the indexes continue to become more fragmented, you should rebuild these indexes each day. You must rebuild or reorganize the indexes when no discovery, enrichment, or other jobs that modify the data are running.

To determinate the fragmentation indexes

You must run this script in the context of the UCMDB database. In this example, replace CMDB_DATABASE with the actual name of the client's UCMDB database.

```
USE CMDB_DATABASE
select
       OBJECT_NAME(b.object_id) as TblName
       , CAST(a.object_id as varchar) as object_id
       , CAST(a.index_id as varchar) as index_id
       , Coalesce(b.name,'') as IndexName
       , CAST(Coalesce(a.avg_fragmentation_in_percent,'') as varchar) as
PercentFragment
       , CAST(CAST(Coalesce(a.fragment_count,'') as bigint) as varchar) as
TotalFrags
       , CAST(Coalesce(a.avg_fragment_size_in_pages,'') as varchar) as
PagesPerFrag
       , CAST(CAST(Coalesce(a.page_count,'') as bigint) as varchar) as NumPages
       , Coalesce(a.index_type_desc,'') as IndexType
       , CAST(Coalesce(a.index_depth,'') as varchar) as index_depth
       , CAST(Coalesce(a.index_level,'') as varchar) as index_level
       , CAST(Coalesce(a.avg_page_space_used_in_percent,'') as varchar) as avg_
page_space_used_in_percent
       , CAST(CAST(Coalesce(a.record_count,'') as bigint) as varchar) as record_
count
FROM sys.dm_db_index_physical_stats(DB_ID('CMDB_DATABASE'),NULL, NULL, NULL ,
'SAMPLED') AS a
JOIN sys.indexes AS b
   ON a.object_id = b.object_id AND a.index_id = b.index_id
WHERE a.avg_fragmentation_in_percent > 15 and record_count > 100000
ORDER BY OBJECT_NAME(b.object_id),a.index_id;
```

Chapter 5: Using Windows Authentication to Access the Microsoft SQL Server Database

Unless configured otherwise, Universal CMDB uses Microsoft SQL Server authentication to access the Microsoft SQL Server databases. However, Windows authentication can also be used.

This chapter describes how to enable Universal CMDB to use Windows authentication to access Microsoft SQL Server databases.

> **Note:** The authentication procedure described in this chapter is relevant for Configuration Manager as well as Universal CMDB.

This chapter includes:

Enabling Universal CMDB to Work with Windows Authentication

You can enable Universal CMDB to use Windows authentication instead of Microsoft SQL Server authentication to access the Universal CMDB database.

To enable Universal CMDB to use Windows authentication to access a Microsoft SQL database, you must:

- Configure the Microsoft SQL Server to use Windows authentication.
- Launch the Universal CMDB Server service on all the Universal CMDB servers with a Windows user that has the necessary permissions to access the Microsoft SQL database.
- Run the UCMDB Server Configuration utility to create or connect to a Microsoft SQL database and specify the use of Windows authentication.

This section includes the following topics:

"Configuring Microsoft SQL Server to Use Windows Authentication" on the next page

"Launching the UCMDB Server service with a Windows User " on the next page

"Creating or Connecting to a Microsoft SQL Database Using Windows Authentication" on page 36

Configuring Microsoft SQL Server to Use Windows Authentication

To configure Microsoft SQL Server to use Windows authentication, perform the following steps:

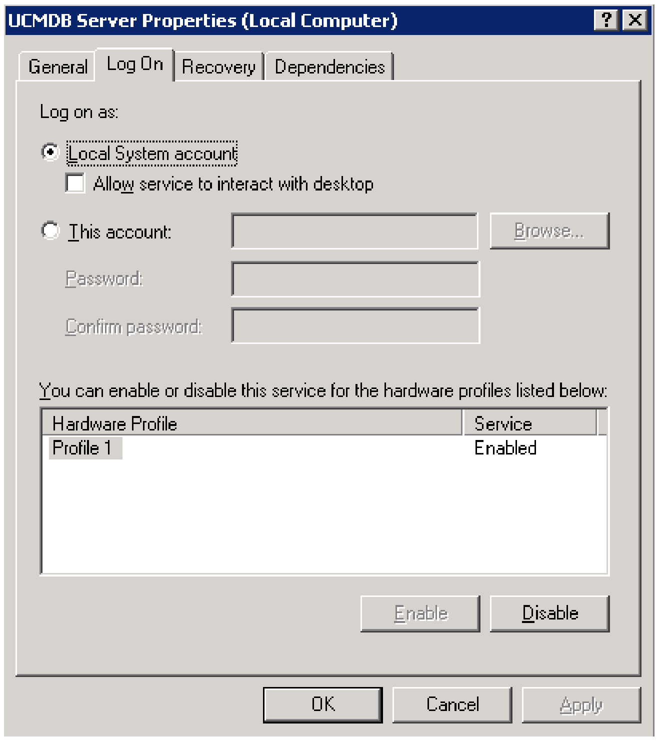
1. In the SQL Server Enterprise Manager, select **Security > Logins**, right-click and choose **New Login**.
2. Enter the desired domain account, including the domain name, in the following pattern: DOMAIN\USER (for example, MY_DOMAIN\MY_ACCOUNT).
3. In the **Server Roles** tab, select **System Administrators** and click **OK**.

Launching the UCMDB Server service with a Windows User

By default, the UCMDB Server service is run as a system service. If you have configured your Microsoft SQL Server to use Windows authentication, you must change the user running the UCMDB Server service to the same Windows user you defined for the Microsoft SQL Server in order to enable the service user to access the database.

To change the UCMDB Server service user:

1. Disable Universal CMDB (**Start > All Programs > UCMDB > Stop Universal CMDB Server**).
2. In Microsoft's Services window, double-click **UCMDB_Server**. The UCMDB Server Properties (Local Computer) dialog box opens.
3. Click the **Log On** tab.



Log on as:

(●) Local System account

[ ] Allow service to interact with desktop

( ) This account:

Password:

Confirm password:

You can enable or disable this service for the hardware profiles listed below:

Hardware Profile	Service
Profile 1	Enabled

4. Select **This account** and browse to choose the user you previously defined for your Microsoft SQL Server.

5. Enter the selected user's Windows password and confirm this password.

6. Click **Apply** to save your settings and **OK** to close the dialog box.

7. Enable Universal CMDB (**Start > All Programs > UCMDB > Start Universal CMDB Server**).

Creating or Connecting to a Microsoft SQL Database Using Windows Authentication

You create or connect to a database using the UCMDB Server Configuration utility. To create or connect to a Microsoft SQL database using Windows authentication, you must select this option within the UCMDB Server Configuration utility. For details on using the utility, see the section about accessing the servers in the interactive *HPE Universal CMDB Deployment Guide*.

Part III: Deploying the Oracle Server Database

Chapter 6: Overview of Oracle Server Deployment

This chapter includes:

About Oracle Server Deployment

To deploy Oracle Server for use with Universal CMDB, perform the following procedures:

> **Note:** Oracle 12C must be installed without container database (CDB). UCMDB does not support Oracle 12C with CDB.

- **Install Oracle Server.**

  For details on Oracle software installation, refer to the installation guide in the documentation for your specific Oracle platform. For software installation options, see "Oracle Server Configuration and Sizing Guidelines" on page 43.

- **Build a database on Oracle Server to store Universal CMDB data.**

  For instance configuration and sizing guidelines, see "Oracle Server Configuration and Sizing Guidelines" on page 43. For details on database instance installation, refer to the installation guide in the documentation for your specific Oracle platform.

- **Create one or more Oracle tablespaces to store Universal CMDB data.**

  For details, see "Oracle Tablespaces" on page 46.

- **Create an Oracle user schema for Universal CMDB schemas.**

  You can create Universal CMDB user schemas manually, or you can use the UCMDB Server Configuration Wizard to create the schemas for you (**Start > All Programs > UCMDB > Start Universal CMDB Server Configuration Wizard**). For details on creating an Oracle user schema for Universal CMDB, see "Manually Creating the Oracle Server Database Schemas" on page 40.

System Requirements

This section describes the system requirements for working with Oracle Server in conjunction with Universal CMDB.

This section includes the following topics:

"Hardware Requirements" below

"Software Requirements" below

Hardware Requirements

For Universal CMDB hardware sizing guidelines, see "Universal CMDB Sizing" on page 8.

For Oracle hardware requirements, refer to the installation guide for your specific Oracle platform. Additional information is also available in the Oracle software distribution media as well as the online Oracle documentation. For Oracle documentation, refer to: http://otn.oracle.com/documentation/index.html.

Software Requirements

For details on supported versions of Oracle Server, see the Server Database Requirements section of the *HPE Universal CMDB Support Matrix*.

Chapter 7: Manually Creating the Oracle Server Database Schemas

This chapter includes:

Overview of the Oracle Database Schema

Universal CMDB uses the configuration management database (CMDB) database schema for its persistency.

During the Universal CMDB setup, new schemas can be set up automatically by the setup procedure, or already existing schemas can be used. Existing schemas can either have been created manually in advance, or can have been created by a previous installation of the same release of Universal CMDB.

If you choose to have the schema set up automatically by the setup procedure, you must have administrator permissions. Usually, the System account credentials are used. If you are unable to submit database administrator connection parameters (for example, due to organization security restrictions), manually set up the schema in advance.

For details about installing the Universal CMDB server, see the interactive *HPE Universal CMDB Deployment Guide*.

Prerequisites

Before creating the database schema, the Oracle Database Server must be installed according to the Oracle documentation. For further details on installing and configuring the Oracle Database Server, see "Oracle Server Configuration and Sizing Guidelines" on page 43.

If the schema is manually created prior to running the Universal CMDB setup, during the setup procedure, select the option to use an existing schema and enter the relevant data about the existing schema for which you are prompted. The data you enter is written to the configuration files that are used by the JDBC driver to connect to the existing schema.

Creating Schema Default and Temporary Tablespaces

The database administrator should create at least one Temporary tablespace in the database instance hosting the Universal CMDB schema, and at least one tablespace to serve as the default tablespace for the Universal CMDB schema.

The tablespace specification should be made according to your storage policy, taking into account RAID configuration, available disks and storage devices.

For further details on tablespace file sizes and recommended settings, see "Oracle Tablespace Settings" on page 47.

Creating Schema User Permissions

The database administrator should create an Oracle schema user with the database permissions required by the Universal CMDB application.

The following database permissions must be granted to Universal CMDB Oracle schema users. These permissions are also used by the Universal CMDB installer to create a new Oracle user:

- Roles
  - Connect
- Privileges
  - CREATE TABLE
  - CREATE VIEW
  - CREATE SEQUENCE
  - CREATE TRIGGER
  - CREATE PROCEDURE

- UNLIMITED TABLESPACE
- ALTER USER ${user} DEFAULT ROLE ALL
- CREATE TYPE
- EXECUTE ON DBMS_LOB
- EXECUTE ON DBMS_STATS

Note:

- Universal CMDB supports any user with higher permissions. For Universal CMDB certification, use an Oracle user that has the exact Oracle permissions described above.
- The last two permissions (EXECUTE ON DBMS_LOB and EXECUTE ON DBMS_STATS) are granted by default.

Creating Schemas and Schema Objects

The database administrator should create a schema for CMDB persistency.

Create a user schema with the permissions described in "Creating Schema User Permissions" on the previous page. No scripts are necessary to create the objects to populate these schemas.

Chapter 8: Oracle Server Configuration and Sizing Guidelines

This chapter contains guidelines for the Oracle database configuration and storage settings that should be used when working with Oracle Server and Universal CMDB. Note that the recommended settings differ according to the size of your Universal CMDB deployment. For details, see "Universal CMDB Sizing" on page 8.

This chapter includes:

Oracle Parameter Settings

The following table describes the recommended values for a number of Oracle database initialization parameters, when working with the Universal CMDB database server:

Parameter Name	**Universal CMDB Deployment**		**Remarks**
	Small	**Large**	
DB_BLOCK_ SIZE	8K	8K-16K	Should be a multiple of the operating system block size.
DB_CACHE_ ADVICE	ON	ON	For gathering statistics when tuning is required .
SGA_TARGET	1 GB	4 GB and	See remarks below the table.

Parameter Name	Universal CMDB Deployment		Remarks
	Small	Large	
		higher	
MEMORY_ TARGET	2 GB	5 GB and higher	See remarks below the table.
LOG_BUFFER	1 MB	5 MB	
DB_FILE_ MULTIBLOCK_ READ_COUNT	Oracle default value	Oracle default value	
PROCESSES	200	400	Add an additional 100 as a safety net
SESSIONS	225	445	(1.1 * PROCESSES) + 5
OPTIMIZER_ INDEX_COST_ ADJ parameter value	100	100	Affects performance
TIMED_ STATISTICS	True	True	
LOG_ CHECKPOINT_ INTERVAL	0	0	
LOG_ CHECKPOINT_ TIMEOUT	0; or greater than or equal to 1800	0; or greater than or equal to 1800	
OPTIMIZER_ MODE	ALL_ ROWS	ALL_ ROWS	
CURSOR_ SHARING	Exact	Exact	
OPEN_ CURSORS	800	800	
COMPATIBLE	The same as the release installed	The same as the release installed	

Parameter Name	Universal CMDB Deployment		Remarks
	Small	Large	
SQL_TRACE	False, True	False	
UNDO_ MANAGEMENT	Auto	Auto	
UNDO_ RETENTION	Oracle default value	Oracle default value	
RECYCLEBIN	Off	Off	
NLS_LENGTH_ SEMANTICS	BYTE	BYTE	This parameter controls the length definition of character type columns.
NLS_COMP	BINARY	BINARY	Using a different value for this parameter may cause severe performance problems.
NLS_SORT	BINARY	BINARY	Using a different value for this parameter may cause severe performance problems.
WORKAREA_ SIZE_ POLICY	AUTO	AUTO	
PGA_ AGGREGATE_ TARGET	400 MB	1 GB and higher	
STATISTICS_ LEVEL	TYPICAL	TYPICAL	Enables tuning if required.
OPTIMIZER_ CAPTURE_ SQL_PLAN_ BASELINES	FALSE	FALSE	Controls Automatic Plan Capture as part of Oracle 11g SQL Management Base (SMB).
AUDIT_TRAIL	NONE	NONE	In Oracle 11g, the default is changed from **none** to **db**, which means that out-of-the-box database auditing is written to the SYS.AUD$ audit trail table. It is advisable to change this value to **none** to avoid growth of the system tablespace.
CURSOR_ SPACE_FOR_ TIME	False	False	
USE_ STORED_	False	False	Oracle Default

Parameter Name	Universal CMDB Deployment		Remarks
	Small	Large	
OUTLINES			
BLANK_ TRIMMING	False	False	
FIXED_DATE	Not set	Not set	Universal CMDB uses the SYSDATE function for generating system time as part of the application process.
PARTITION LARGE_ EXTENTS	FALSE	FALSE	Relevant for Oracle 11.2.0.2 only. When this hidden parameter is set to TRUE, it affects the size of partitions in native partitioned tables. The initial extent allocated for each partition is very large, thus causing unwanted growth of database data files.

Note the following:

- **SGA_TARGET.** Setting this parameter configures Oracle to automatically determine the size of the buffer cache (db_cache_size), shared pool (shared_pool_size), large pool (large_pool_size), java pool (java_pool_size), and streams pool (streams_pool_size).

  The value configured for SGA_TARGET sets the total size of the SGA components.

  When SGA_TARGET is set (that is, it's value is not 0), and one of the above pools is also set to a non-zero value, the pool value is used as the minimum value for that pool.

- **MEMORY_TARGET.** In Oracle 11g, Automatic Memory Management enables the entire instance memory to be automatically managed and tuned by the instance. The instance memory contains the System Global Area (SGA) and the Program Global Area (PGA). In Oracle 11g, MEMORY_ TARGET is the only required memory parameter to set, however it is recommended to set SGA_ TARGET or PGA_AGGREGATE_TARGET as well to avoid frequent resizing of the SGA and PGA components. The values entered for SGA_TARGET and PGA_AGGREGATE_TARGET serve as minimum values.

> **Note:** HPE does not recommend to collect statistics for any temporary tables.

Oracle Tablespaces

An Oracle tablespace is an Oracle object that is a logical container of database objects, for example, tables, indexes, and so forth. When working with Universal CMDB, you must create one or more dedicated default tablespaces for your Universal CMDB user schema. You may also want to create a dedicated temporary tablespace for Universal CMDB. To create a tablespace, you must provide specific operating system files that physically represent the tablespace, as well as extent parameters.

When mapping operating system files, there is an option to make the file auto-extendable. This feature is supported by Universal CMDB, but not certified for use with Universal CMDB, since it can cause the system to consume all available disk space.

Locally Managed Tablespaces

A locally managed tablespace is a feature introduced in Oracle8i. Prior to Oracle8i, all tablespaces were dictionary-managed tablespaces. A tablespace that manages its extents locally can have either uniform extent sizes, or variable extent sizes that are determined automatically by the system. When you create the tablespace, the **uniform** or **autoallocate** (system-managed) option specifies the type of allocation.

For system-managed extents, Oracle determines the optimal size of extents, with a minimum extent size of 64 KB. This is the default extent size for permanent tablespaces.

For uniform extents, you can specify an extent size, or use the default size, which is 1 MB. Temporary tablespaces that manage their extents locally can only use this type of allocation.

Note that the NEXT, PCTINCREASE, MINEXTENTS, MAXEXTENTS, and DEFAULT STORAGE storage parameters are not valid for extents that are managed locally.

All data and temporary tablespaces should be locally managed when working with Universal CMDB.

For information on locally managing temporary tablespace using TEMPFILE, see "Temporary Tablespace Settings" on page 49.

Oracle Tablespace Settings

This section describes the storage settings and file sizing guidelines for data tablespaces, temporary tablespaces, redo logs, and undo tablespaces.

This section includes the following topics:

"Data Tablespace Settings" on the next page

"System Tablespace Settings" on page 49

"Temporary Tablespace Settings" on page 49

"Redo Log Settings" on page 50

"Undo Segment Settings" on page 50

Data Tablespace Settings

The following table specifies the recommended sizes for Universal CMDB tablespaces:

	Universal CMDB Deployment		
Tablespace	**Small**	**Large**	**Remarks**
CMDB	5 GB	60 GB	The specified size is a minimum requirement.

Note: The data in the table is only relevant when the following parameters are assigned their default values:

- the Oracle parameter **deferred_segment_creation** is set to **true** (only relevant for Oracle 11g R2)
- the tablespace setting **INITIAL_EXTENT** is set to **65,536**

Data Tablespace Default Storage Settings

The storage settings for data tablespaces should be:

- Locally managed tablespace
- Automatic segment space management
- Automatic local extent management

The following command can be used to create a data tablespace:

```
CREATE TABLESPACE <tablespace name> DATAFILE ’<data file full path>’ REUSE SIZE
<file size>
EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;
```

For example:

```
CREATE TABLESPACE APPTBS DATAFILE
’/ORADATA/ORCL/APPTBS1.DBF’ REUSE SIZE 1024M
EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;
```

System Tablespace Settings

The following table specifies the recommended settings for system tablespaces:

	Universal CMDB Deployment	
Tablespace	**Small**	**Large**
SYSTEM	2 GB	5 GB
SYSAUX	2 GB	5 GB

The System Tablespace storage default settings should be:

- Locally managed tablespace
- Segment space management:
    - SYSAUX: Automatic
    - SYSTEM: Manual
- Automatic local extent management

Temporary Tablespace Settings

The following table specifies the recommended settings for temporary tablespaces:

	Universal CMDB Deployment		
Tablespace	**Small**	**Large**	**Remarks**
TEMP	1 GB	10 GB	Use multiple files with large tablespaces.
TEMP storage settings	Uniform allocation: 2 MB	Uniform allocation: 2 MB	• Should be locally managed (Uniform allocation). • Tablespaces should be of a temporary type (use of TEMPFILE). • Segment space management in temporary tablespaces is manual.

Redo Log Settings

The following table specifies the recommended settings for redo log files:

	Universal CMDB Deployment	
Setting	**Small**	**Large**
Redo log file size	100 MB	200 MB - 500 MB
Minimum number of groups	4	4
Minimum number of members per group	2	2

Undo Segment Settings

The following table specifies the recommended Undo settings:

	Universal CMDB Deployment		
Setting	**Small**	**Large**	**Remarks**
Undo tablespace size	1 GB	10 GB	The number of segments, the minimum number of extents, and the rollback segment size (initial, next) are all set automatically by Oracle.
UNDO_ MANAGEMENT parameter	AUTO		Oracle default value
UNDO_ RETENTION parameter	Oracle default value		

The Undo Tablespace storage default settings should be:

- Locally managed tablespace
- Automatic segment space management
- Segment space management in undo tablespaces is manual

Using RAID Configuration

The use of RAID is transparent to Oracle. All the features specific to RAID configurations are handled by the operating system and not by Oracle.

The use of RAID devices differs according to the Oracle file type. Data files and archive logs can be placed on RAID devices, since they are accessed randomly. Redo logs should not be put on RAID devices, since they are accessed sequentially and performance is enhanced by having the disk drive head near the last write location. However, mirroring of redo log files is strongly recommended by Oracle.

RAID is much easier to use than the Oracle techniques for data placement and striping.

Note the following RAID configuration recommendations:

- RAID usually impacts write operations more than read operations. This is especially true where parity needs to be calculated (RAID 3, RAID 5, and so forth).
- You can place online or archived redo log files on RAID 1 devices. Do not use RAID 5. In addition, place TEMP tablespace data files on RAID 1 devices, instead of RAID 5, because the streamed write performance of distributed parity (RAID 5) is not as good as that of simple mirroring (RAID 1).
- Swap space can be used on RAID devices without affecting Oracle.

The following table describes the RAID devices and types to be used with each Oracle file type:

RAID	**Type of RAID**	**Control File**	**Database File**	**Redo Log File/Temporary**	**Archive File**
0	Striping	Avoid	OK	Avoid	Avoid
1	Shadowing	OK	OK	Recommended	Recommended
0+1	Striping + Shadowing	OK	Recommended	Avoid	Avoid
3	Striping with static parity	OK	Avoid when this data file involves heavy write operation	Avoid	Avoid
5	Striping with rotating parity	OK	Avoid when this data file involves heavy write operation	Avoid	Avoid

> **Note:**
>
> - RAID 0 does not provide protection against failures. It requires a strong backup strategy.
> - RAID 0+1 is recommended for database files because it avoids hot spots and provides the best possible performance during a disk failure. The disadvantage of RAID 0+1 is its costly configuration.
> - Use the highest RPM disks for temporary/redo logs. Use as many controllers as you can in the array, and ensure that you place the redo log groups on different controllers.

Connecting to an Oracle Database Using Service Name

Starting with version 10.32, it is possible to connect UCMDB server to an Oracle database using Service Name.

> **Note:** This support only works with the Oracle native driver, and does not work with Data Direct drivers. By default it should be native driver. You can check the configuration in the **wrapper.conf** file, see the following line: **wrapper.java.additional.47=-Doracle.native.driver=true**.
>
> Starting with version 10.33, to use the Oracle native driver, you may need to perform additional steps. For details, see "Configuring Oracle JDBC Driver" on the next page.

You can enable this support using either of the following approaches:

- ### (Recommended) Provide a value for the Service Name field in the UCMDB Server Configuration Wizard

  When creating a new schema or connecting to an existing schema with the UCMDB Server Configuration Wizard, on the CMDB Schema Settings tab page, provide a value for the **Service Name** field.

  > **Note:**
  >
  > - The **Service Name** field is mutually exclusive with the **SID** field, you can only provide a value for either **Service Name** or **SID**.
  > - The **Service Name** field is only available when the Oracle native drivers are used.

- Modify the dal.datamodel.service.name setting in the cmdb.conf file

  a. On the UCMDB server machine, open the **<UCMDB_Server>\conf\cmdb.conf** file using a text editor.

  b. Locate the new database context configuration **dal.datamodel.service.name**.

  c. Provide the service name.

  d. Save the file.

  e. Restart the UCMDB Server.

  > **Note:** The **dal.datamodel.service.name** setting in the **cmdb.conf** file has precedence over the **dal.datamodel.sid** setting. That is to say, if the Service Name setting is set, it will be used for database schema connection regardless of whether the SID setting is also set or not, therefore these two settings cannot be used simultaneously.
  >
  > For example, if a wrong Service Name is provided, the connection will fail, even if a correct SID is provided. To use the **dal.datamodel.sid** setting for connection, simply leave the **dal.datamodel.service.name** setting blank.

Configuring Oracle JDBC Driver

Starting with version 10.33, to use Oracle native driver, you may need to perform additional steps to configure the Oracle JDBC driver.

To use Oracle JDBC driver, do the following:

1. Stop the UCMDB Server.
2. Download Oracle JDBC Driver v12.1.0.2 from here: http://www.oracle.com/technetwork/database/features/jdbc/default-2280470.html
3. Copy the downloaded jar file to the following location: **{UCMDB_INSTALLATION_DIR}/lib/**
4. Edit the **wrapper.conf** file from the **{UCMDB_INSTALLATION_DIR}/bin** directory to include the following line:

   ```
   wrapper.java.additional.47=-Doracle.native.driver=true
   ```

   The default value for `-Doracle.native.driver` is `false`.

5. Save the file.
6. Start the UCMDB Server.

Composite Indexes vs. Non Composite Indexes

For environments that execute heavy data-in operations, in order to speed up the INSERT statements, it is recommended to transform the database from composite index to non composite index.

You can achieve this by invoking the **modifyCompositeIndexes** JMX method with **composite indexes** setting to **false**. For detailed instructions, see "How to Modify Composite Indexes".

Checklist for Universal CMDB Support and Certification

Information is provided in this section for both supported and certified Oracle options.

The certified options are recommended for working with Universal CMDB. Certified options are rigorously tested by HPE quality assurance personnel. Supported options are those options for which HPE quality assurance personnel have successfully performed basic tests.

Option	Supported	Recommended	Remarks	For more information, see
Oracle edition	Standard, Enterprise	Enterprise		
Dedicated Universal CMDB server	Not necessary	Not necessary. It is recommended to dedicate an instance for Universal CMDB.		
Use of multiple Oracle instances	Yes	No	The configuration of all instances must match in a certified environment.	"System Requirements" on page 39
Use of non-default port	Yes	Yes		

Option	Supported	Recommended	Remarks	For more information, see
Undo management	Automatic; Manual	Automatic	Set UNDO_ MANAGEMENT parameter to AUTO in a certified environment	
Oracle Shared Servers connection method	Yes	No	Universal CMDB uses a connection pool architecture. Use the dedicated server connection method in a certified environment	
Oracle replication	No full support	No		
Operating System file compression	No	No	Not supported by Oracle; Causes abnormal behavior and affects performance	
Database control files required	Greater than or equal to 2		Preferable on different disks.	
Redo log groups	Greater than or equal to 3	4	Oracle enables the software mirroring of redo log files. This is achieved by creating at least two members of redo log in each group. Members of the same group should reside on different disks.	
Character set	UTF8,AL32UTF8	AL32UTF8		
OPEN_ CURSORS	800	800		
Working in archive log mode	True; False	True		

Option	Supported	Recommended	Remarks	For more information, see
Autoextend option in tablespace files	Yes	No		
Locally managed data tablespace	Yes	Yes		"Locally Managed Tablespaces" on page 47
Tablespace extent management	Local uniform for TEMP tablespace	Local uniform for TEMP tablespace		"Oracle Tablespace Settings" on page 47
Automatic Segment Space Management Tablespace (ASSM)	Yes	Yes		

How to Enable Support for Oracle Advanced Security (OAS) in Generic DB Adapter Based Integrations

To enable support for Oracle Advanced Security (OAS) in Generic DB Adapter based integrations, do the following

1. Open the **adapter.conf** file in the Generic DB Adapter.

   In UCMDB UI, go to **Data Flow Management > Adapter Management**. Under the Packages root folder in the Resource pane, go to **db-adapter > Configuration Files > GenericDBAdapter/META-INF/adapter.conf**

2. Locate the following line:

   ```
   dal.use.persistence.xml=false
   ```

3. Change the value to **true** as follows:

   ```
   dal.use.persistence.xml=true
   ```

4. Click **Save** Save.
5. Open the **persistence.xml** file by clicking **GenericDBAdapter/META-INF/persistence.xml**.
6. Locate the following line:

   ```
   <persistence-unit name="GenericDBAdapter"/>
   ```

7. Replace it with the following content:

   ```
   <persistence-unit name="GenericDBAdapter">
       <properties>
           <!-- added to fix:  org.hibernate.HibernateException:
   'hibernate.dialect' must be set when no connection available -->
           <property name="hibernate.dialect"
   value="org.hibernate.dialect.Oracle10gDialect"/>
           <property name="hibernate.hbm2ddl.auto" value="create-drop"/>
           <property name="hibernate.connection.driver_class"
   value="oracle.jdbc.OracleDriver"/>
           <property name="hibernate.connection.url"
   value="jdbc:oracle:thin:@<oracle-host-name>:<oracle-port>:<oracle-db-sid>"/
   >
           <property name="oracle.net.encryption_client" value="REQUIRED" />
           <property name="oracle.net.encryption_types_client" value="( DES40 )
   " />
           <property name="oracle.net.crypto_checksum_client" value="REQUESTED"
   />
           <property name="oracle.net.crypto_checksum_types_client" value="MD5"
   />
       </properties>
   </persistence-unit>
   ```

8. Update `connection.url` in the above code snippet with the proper hostname/port/SID, and also update the username and password.
9. Click **Save** Save.
10. Wait a few minutes for the probe to retrieve the latest settings.
11. Run test connection and check whether it works.

Chapter 9: Real Application Cluster Support

This chapter includes:

Note: The information in this chapter applies to advanced users only.

About Oracle Real Application Cluster (RAC)

A cluster is a collection of interconnected servers that appear as one server to the end user and to applications. Oracle Real Application Cluster (RAC) is Oracle's solution for high availability, scalability, and fault tolerance. It is based on clustered servers that share the same storage.

Oracle RAC is a single Oracle database installed on a cluster of hardware servers. Each server runs an instance of the database and all the instances share the same database files.

For more details about Oracle RAC, refer to the *Oracle Clusterware Guide* and the *Oracle Real Application Clusters Administration and Deployment Guide* in the Oracle documentation set of your release.

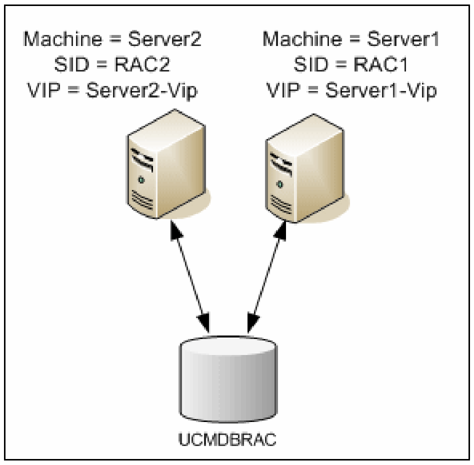
In this chapter, the following Oracle RAC example is used:

- Oracle RAC database name: UCMDBRAC
- Machine names: Server1, Server2
- On each machine there is an Oracle instance of UCMDBRAC:
  - SID on Server1: RAC1
  - SID on Server2: RAC2
- On each machine there is a virtual IP (Server1-Vip and Server2-Vip):

 - Server1-Vip is assigned to Server1
 - Server2-Vip is assigned to Server2

 The virtual IP is in addition to the static IP assigned to the machine.

- The listeners on both servers are listening on the default port 1521 and support the database service UCMDBRAC.



Single Client Access Name

In release 11g, Oracle introduced the Single Client Access Name (SCAN), as a preferred access method for clients connecting to the RAC. In this method clients are not required to configure individual nodes in the RAC, rather they use a single virtual IP known as the SCAN or the SCAN VIP.

The SCAN is a single network name defined for the cluster either in your organization's Domain Name Server (DNS) or in the Grid Naming Service (GNS) that rotates between several IP addresses reflecting multiple listeners in the cluster. The SCAN eliminates the need to change clients when nodes are added to or removed from the cluster.

The SCAN and its associated IP addresses provide a stable name for clients to use for connections, independent of the nodes that make up the cluster. Database server SCAN addresses, virtual IP addresses, and public IP addresses must all be on the same subnet.

In Universal CMDB, when using Oracle RAC, it is recommended to use the SCAN method.

Client Side Configuration for Oracle RAC

Universal CMDB uses DataDirect's JDBC driver to connect to regular Oracle databases and to Oracle RAC databases.

> **Note:** Starting with version 10.33, to use Oracle JDBC driver, you may need to perform additional steps. For details, see "Configuring Oracle JDBC Driver" on page 53.

When using a RAC database, install the servers without creating the relevant UCMDB database schemas (for a description of UCMDB schemas, see "Introduction to Preparing the Database Environment" on page 7). After completing the client and server side RAC configuration, create the schemas as described in "Create/Connect to a Universal CMDB Database" on page 62.

Make the following changes in Universal CMDB's configuration files before you create the management database or connect to an existing one on Oracle RAC:

1. On the UCMDB Server, create the file **<Universal CMDB root directory>\UCMDBServer\conf\ucmdb-tnsnames.ora**.

   The format of **ucmdb-tnsnames.ora** is the same as the Oracle **tnsnames.ora** network configuration file:

   ```
   <DB service name> =
   (DESCRIPTION =
    (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST = <first instance virtual ip> )
   (PORT = <first instance's listener port>))
    (ADDRESS = (PROTOCOL = TCP)(HOST = <second instance virtual ip> )
   (PORT = <second instance's listener port>))
    (... entry for each instance...)
    (LOAD_BALANCE = on)
    (FAILOVER = on)
   )
    (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = <DB service name>)
   )
   )
   ```

   where:

- `<DB service name>` is the name of a service the listeners support. It is the same one used in the CONNECT_DATA part.
- `ADDRESS_LIST` contains an address entry for each node in the RAC environment. In the case of Oracle 11g using SCAN, it contains only the SCAN virtual IP. The address contains all the details needed for connecting to the node:
  - `HOST` contains the virtual-IP for that instance. It is important to use the virtual IP and not the static IP of the node for faster failure detection.
  - `PORT` is the port on which the listener is configured to listen on that given node.
  - `FAILOVER` set to **on** allows the driver to try to connect to another node after failing to connect to one of the nodes. The connection attempts continue until a connection is successfully established to one of the nodes or until none of the nodes can be reached.

    An attempt to connect to another node takes place only if the connection to the current node fails. If the connection is refused by the node (for example, communication to the node was established, but the connection was rejected), no attempt is made to connect to another node.

    Caution: Failover is for connection attempts only. If a connection fails during a database transaction, there is no failover of the transaction to another machine to continue the transaction.
  - `LOAD_BALANCE` set to **on** instructs the driver to distribute connection requests between the nodes to prevent overloading any single node. The order in which the driver accesses the nodes is random.
  - `SERVER` is the connection mode you want to use. There are two possible connection modes: **dedicated** and **shared**. Configure this according the Server configuration you support.
  - `SERVICE_NAME` is the name of a service that the listeners support. It is the same one you gave in `<DB service name>`.

In the above example, **ucmdb-tnsnames.ora** would be configured as:

```
UCMDBRAC =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = Server1-Vip)(PORT = 1521))
(ADDRESS = (PROTOCOL = TCP)(HOST = Server2-Vip)(PORT = 1521))
(LOAD_BALANCE = on)
(FAILOVER = on)
)
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = UCMDBBSMRAC)
)
)
```

2. On the UCMDB Server, open the directory **<Universal CMDB root directory>\UCMDBServer\conf**. Locate the **jdbc.properties** file.

   a. Find the line starting with **cmdb.url**.

   b. Replace this with the following line:

   ```
   cmdb.url=
   jdbc:mercury:oracle:TNSNamesFile=<Universal CMDB root
   directory>\\UCMDBServer\\conf\\ucmdb-tnsnames.ora;TNSServerName=<SERVICE
   NAME>
   ```

   where `<SERVICE NAME>` is the entry in **ucmdb-tnsnames.ora**, equivalent to the RAC service name.

   Note that each back slash (`\`) in the path of `TNSNamesFile` is doubled.

   c. If the file does not exist, create an empty **jdbc.properties** file under the above folder and add the following entry:

   ```
   Oracle = ddoracle
   cmdb.url = jdbc:mercury:oracle:TNSNamesFile=<Universal CMDB root
   directory>\\UCMDBServer\\conf\\ucmdb-tnsnames.ora;TNSServerName=<SERVICE
   NAME>
   ```

   where `<SERVICE NAME>` is the entry in **ucmdb-tnsnames.ora**, equivalent to the RAC service name.

   d. If your server is running the UNIX or LINUX operating system, replace all the double back slashes with single backslashes.

Create/Connect to a Universal CMDB Database

When creating a new database schema or connecting to an existing schema in the UCMDB Server Configuration Wizard, fill all the parameters as follows:

- **Oracle Machine.** One of the Virtual IPs. In Oracle 11g, the SCAN virtual IP can be used.
- **Port.** The local listener port on the Oracle machine or the port of the SCAN listener.

- **SID.** The service name of the database.
- **Schema name and schema password.** The name and password of the existing user schema, or the name that you are giving the new user schema (for example, UCMDB_schema).

If you are creating a new user schema, you need the following additional information:

- **Admin user name and admin password (to connect as an administrator).** The name and password of a user with administrative permissions on Oracle Server (for example, a System user).
- **Default tablespace.** The name of the default tablespace you created for the user schema. For details on creating an Universal CMDB tablespace, see "Manually Creating the Oracle Server Database Schemas" on page 40".
- **Temporary tablespace.** The name of the temporary tablespace you assigned to the user schema. The default Oracle temporary tablespace is **temp**.

In the example, the parameters would be:

Parameter	Value
Oracle Machine	Server1-Vip
Port	1521
SID	UCMDBRAC

Part IV: Deploying the PostgreSQL Server Database

Chapter 10: Overview of PostgreSQL Server Deployment

This chapter includes:

About PostgreSQL Server Deployment

Caution: Embedded PostgreSQL is not supported on Enterprise deployments of UCMDB.

To deploy PostgreSQL Server for use with Universal CMDB, you must perform the following procedures:

- **Install and configure PostgreSQL Server.**

  For details on installing and configuring PostgreSQL Server, see the PostgreSQL Server documentation.

- **Create databases on PostgreSQL Server.**

  You create Universal CMDB databases using the UCMDB Server Configuration Wizard.

System Requirements

This section describes the system requirements for working with PostgreSQL Server in conjunction with Universal CMDB.

This section includes the following topics:

"Hardware Requirements" on the next page

"Software Requirements" on the next page

Hardware Requirements

For Universal CMDB hardware sizing guidelines, see "Hardware Requirements" on page 8. For PostgreSQL Server hardware requirements, refer to the installation guide for your PostgreSQL Server release for your operating system.

Software Requirements

For details on supported versions of PostgreSQL Server, see the Server Database Requirements section of the *HPE Universal CMDB Support Matrix*.

Overview of the PostgreSQL Server Databases

Universal CMDB uses the configuration management database (CMDB) for its persistency.

During the Universal CMDB setup, the new database can be set up automatically by the setup procedure, or an existing database can be used. An existing database can either be created manually in advance (for example, due to organization security restrictions), or can be created by a previous installation of the same release of Universal CMDB.

For details about installing the Universal CMDB server, see the interactive *HPE Universal CMDB Deployment Guide*.

PostgreSQL Deployment Options

There are two options for deploying PostgreSQL Server with Universal CMDB:

- **Embedded.** PostgreSQL is part of the Universal CMDB installation.
- **External.** PostgreSQL installed on a separate server.

The embedded deployment option is appropriate for small capacity UCMDB installations. In this option, the following configuration files are used:

- **postgresql.conf**
- **ucmdbpg.conf**

Both files are located in the **UCMDB\UCMDBServer\PostgreSQL\data\** directory. PostgreSQL parameters in the ucmdbpg.conf file overwrite parameters in the postgresql.conf file.

> **Note:** In the external deployment option, only the **postgresql.conf** file is used by default.

Initialize the PostgreSQL Server

If you choose to not configure the UCMDB Server with the PostgreSQL database at the time of UCMDB installation, you can initialize the PostgreSQL database and create the database service according the following instructions:

> **Note:** Initializing the PostgreSQL database and creating the database service is necessary only if you want to use the local PostgreSQL as an alternative to Oracle/MSSQL. It is not mandatory to use the local PostgreSQL; the UCMDB Server functions properly without it.

For Windows:

> **Note:** In the following instructions, replace **C:\hp\UCMDB\UCMDBServer\** with the appropriate path.

1. Download and install **vcredist_x64.exe** from the Microsoft website.
2. Run: **C:\hp\UCMDB\UCMDBServer\PostgreSQL\pgInitDb.bat username password port**

   The username, password, and port must be chosen by the user. Verify that the port is free.
3. Run:
   C:\hp\UCMDB\UCMDBServer\PostgreSQL\pgsql\bin\pg_ctl.exe register -N UCMDB_Server_DB -D "C:\hp\UCMDB\UCMDBServer\PostgreSQL\pgsql\data".
4. Run: **sc description UCMDB_Server_DB "HP UCMDB Database"**.
5. Run: **net start UCMDB_Local_DB**.
6. Run the configuration wizard, and provide the username, password and port from above.

For Linux:

> **Note:** In the following instructions, replace **/opt/hp/UCMDB/UCMDBServer/** with the appropriate path.

1. Run: **find /opt/hp/UCMDB/UCMDBServer/PostgreSQL -type f \(-name "*.sh" -or -name "*.sql" -or -name "*.conf" -or -name "*.cnf" \) -exec dos2unix {} \;**

2. Create the postgres_server group using: **/usr/sbin/groupadd -f postgres_server**
3. Create the postgres_server user using: **/usr/sbin/useradd -r -g postgres_server postgres_server**
4. Run: **rpm -qa | grep -ic postgres**

   > **Note:** If there are other versions of the Postgres RPM package installed on the system, it is recommended to remove them as they may cause conflicts.

5. Run:

   cd /opt/hp/UCMDB/UCMDBServer/PostgreSQL

   chmod -R 770 .

   chown -R postgres_server .

   chgrp -R postgres_server .

6. Run: **runuser -l postgres_server -c "/opt/hp/UCMDB/UCMDBServer/PostgreSQL/pgInitDB.sh username password port"**

   The username, password, and port must be chosen by the user. Verify that the port is free.

7. Edit **/opt/hp/UCMDB/UCMDBServer/PostgreSQL/pgsql/postgresql.server** and replace the string **_install_dir_** with the path to the installation folder (for example, **/opt/hp/UCMDB/UCMDBServer/**).
8. Run:

   cd /opt/hp/UCMDB/UCMDBServer/PostgreSQL/pgsql

   chmod a+rx ./postgresql.server

9. Copy **/opt/hp/UCMDB/UCMDBServer/PostgreSQL/pgsql/postgresql.server** to **/etc/init.d/** and rename it to **postgresql_server**.
10. Run:

    chkconfig --add postgresql_server

    /sbin/service postgresql_server start

11. Run the configuration wizard, and provide the username, password and port from above.

Chapter 11: PostgreSQL Server Configuration

This chapter contains guidelines for the database configuration and storage settings that should be used when working with PostgreSQL Server and Universal CMDB.

This chapter includes:

PostgreSQL Parameter Settings

The following table describes the recommended (non-default) values for a number of PostgreSQL database initialization parameters, when working with the Universal CMDB database server:

Category	Parameter Name	Universal CMDB Deployment		Defaults and Remarks
		Small	Large	
Memory	shared_buffers	1024 MB	4096 MB	Default: 32M, shared resource setting
	work_mem	25 MB	50 MB	Default: 1M, setting per session
	maintenance_work_mem	256 MB	340 MB	Default: 16M, setting per session
Planner	effcetive_cache_size	4096 MB	8192 MB	Default: 128M, setting per session and based on total available RAM
Checkpoint (WAL)	checkpoint_segments	32	64	Default: 3 - maximum distance in log segments between WAL checkpoints
	checkpoint_timeout	15 minutes	20 minutes	Default: 300 sec - maximum time between WAL checkpoints
	checkpoint_completion_	0.9	0.9	Default: 0.5 - target of checkpoint completion, as a fraction of total time

Category	Parameter Name	Universal CMDB Deployment		Defaults and Remarks
		Small	Large	
	target			between checkpoints
Autovacuum	autovacuum_ vacuum_ threshold	5000	5000	Default: 50 - minimum number of tuple updates or deletes prior to vacuum
	autovacuum_ analyze_ threshold	5000	5000	Default: 50 - minimum number of tuple changes prior to analyze
	autovacuum_ analyze_scale_ factor	0.1	0.2	Default: 0.1 (10% of table size) – estimated percent of tuple changes prior to analyze
Logging	log_min_ messages	info	info	Default: **warning**
	log_min_ duration_ statement	1500	3000	**0** prints all queries; **1** turns the feature off.
	log_checkpoints	on	on	Default: **off** - logs each checkpoint
	log_statement	ddl	ddl	Default: **none** - sets the type of statements logged
	log_ autovacuum_ min_duration	0	0	Default: **1** - turns autovacuum logging off; **0** prints all actions

Critical PostgreSQL Server Files

Note the following types of files that are relevant when working with PostgreSQL Server with Universal CMDB. It is strongly recommended for these file types to be located on different disks from each other and from the main database files (the installation files):

- **Data (PGDATA)**

  All the data needed for a database is stored within the data directory, commonly referred to as PGDATA (after the name of the environment variable that can be used to define it). A common location for PGDATA is /var/lib/pgsql/data.

The PGDATA directory contains several subdirectories and control files. In addition to these required items, the cluster configuration files PostgreSQL.conf, pg_hba.conf, and pg_ident.conf are traditionally stored in PGDATA (although in PostgreSQL 8.0 and later, it is possible to keep them elsewhere).

During the PostgreSQL installation, select the option to store the PGDATA files in a designated path.

- **Write-Ahead Logging (WAL)**

  Write-Ahead Logging (WAL) is a standard method for ensuring data integrity. WAL's central concept is that changes to data files (where tables and indexes reside) must be written only after those changes have been logged, that is, after log records describing the changes have been flushed to permanent storage. This procedure eliminates the need to flush data pages to disk on every transaction commit, because in the event of a problem, any changes that have not been applied to the data pages can be redone from the log records. (This is roll-forward recovery, also known as REDO.)

  You can ensure that these files are located on different disks by moving the pg_xlog directory to another location. To do this, follow this procedure:

  a. Shut down the server.

  b. Create a symbolic link from the original location in the main data directory to the new location, using the following command:

     ```
     mklink /j "C:\Program Files\PostgreSQL\9.2\data\pg_xlog" "E:\pg_xlog".
     ```

     For details, see **http://www.postgresql.org/docs/9.2/static/wal-internals.html**.

How to Configure PostgreSQL on Probe to Accept Local Connections Only

By default the PostgreSQL instance running on the probe does not need to listen to requests from the network. However, you can perform this additional configuration to bind the PostgreSQL instance on the probe to localhost.

To do so,

1. On the probe, open the following file using a text editor:

   hp\UCMDB\DataFlowProbe\conf\postgresql.conf

2. Locate and modify the following file from:

   ```
   listen_addresses = '*'
   ```

   to:

   ```
   listen_addresses = 'localhost'
   ```

3. Save the file.
4. Restart the probe.

   To restart the probe properly,

   a. Open **services.msc**.
   b. Stop the **UCMDB Probe** service.
   c. Stop the **UCMDB_Probe_DB** service.
   d. Start the **UCMDB_Probe_DB** service.
   e. Start the **UCMDB Probe** service.

How to Validate If the PostgreSQL Instance Is Listening Only to Localhost?

Validate from a remote PostgreSQL server

To validate that the change to the **postgresql.conf** file is taking effect successfully, you can run the following PostgreSQL client command from a remote server with PostgreSQL installed:

> **Note:** Do not run the command below from the same machine where the probe is installed.

psql --username="mamuser" --host=<Target DataFlowProbe Host or IP>

When PostgreSQL is listening only to localhost, you will receive a "`Connection refused`" message.

Example outputs:

PostgreSQL listening to localhost:

```
C:\hp\UCMDB\DataFlowProbe\pgsql\bin>psql --username="mamuser" --
host=10.10.15.220

psql: could not connect to server: Connection refused (0x0000274D/10061)

Is the server running on host "10.10.15.220" and accepting

TCP/IP connections on port 5432?
```

PostgreSQL listening to any host (known as “*” in the **postgressql.conf** file):

```
C:\hp\UCMDB\DataFlowProbe\pgsql\bin>psql --username="mamuser" --
host=10.10.15.220

psql: FATAL:  no pg_hba.conf entry for host "10.10.20.254", user "mamuser",
database "mamuser", SSL off
```

The `FATAL` error indicates simply that the **pg_hba.conf** file is not properly set up, but for the purpose of this test it is sufficient to understand that the connection was successful from a remote host. This would indicate the change to **listen_addresses** in **postgresql.conf** was NOT applied properly per the instructions above. Please check those steps again.

Validate using Telnet

If you do not have another PostgresSQL system with **psql** installed, you can try to Telnet to the PostgresSQL port from a different computer to confirm if the change was applied correctly:

1. Install Telnet.
   a. Go to **Start > Control Panel > Turn Windows features on or off**.

      The Windows Features dialog opens.
   b. Select the check box for **Telnet Client**.
   c. Click **OK**.
2. Test connection.
   a. Open the Command Prompt window by going to **Start > Run > cmd** and clicking **OK**.
   b. From the Command Prompt window, run the following:

      telnet <target host/IP and port>

      For example, **telnet 10.10.15.220 5432**.

If you receive a message like: "`telnet: connect to address <address you specified>: Connection refused`", it indicates that PostgresSQL is properly configured to refuse remote connections.

How to Configure PostgreSQL Log Files Rotation by Size

The log rotation by size on PostgreSQL keeps all the log files, and then the disk space could become full.

To configure rotation of the PostgreSQL log files by size:

1. Locate and open the PostgreSQL configuration file using a text editor:
   - For embedded PostgreSQL: **<UCMDB_Server_Home>\PostgreSQL\ucmdbpg.conf**
   - For external PostgreSQL: **<UCMDB_Server_Home>\PostgreSQL\data\postgresql.conf**
2. Copy and paste the following configuration into the file, under the CUSTOMIZED OPTIONS section:

```
# These are only used if logging_collector is on:
log_directory = 'pg_log'           # directory where log files are written,
                                   # can be absolute or relative to PGDATA
log_filename = 'postgresql-%a.log' # log file name pattern,
                                   # can include strftime() escapes
log_file_mode = 0600               # creation mode for log files,
                                   # begin with 0 to use octal notation
log_truncate_on_rotation = on      # If on, an existing log file with the
                                   # same name as the new log file will be
                                   # truncated rather than appended to.
                                   # But such truncation only occurs on
                                   # time-driven rotation, not on restarts
                                   # or size-driven rotation.  Default is
                                   # off, meaning append to existing files
                                   # in all cases.
log_rotation_age = 1440            # Automatic rotation of logfiles will
                                   # happen after that time.  0 disables.
log_rotation_size = 0              # Automatic rotation of logfiles will
                                   # happen after that much log output.
                                   # 0 disables
```

3. Save the file.
4. Restart PostgreSQL for the changes to take effect.

Recommended usage of the options for the backup period:

- For backup period of only 24 hours, just use **%H**: **log_filename = 'postgresql-%H.log'**, set **log_rotation_age = 60**.

  One day could be too short for future investigation and troubleshooting, because the log files will be overwritten after one day.

- For backup period of one week, use **%a-%w**: **log_filename = 'postgresql-%a.log'**, and set **log_rotation_age = 1440**.

  The log files will be overwritten after one week. **%a** is the day of the week and **%w** is the week of the month.

- For backup period of one month, use **%d**: **log_filename = 'postgresql-%d.log'**, and set **log_rotation_age = 1440**.

  The log file will be overwritten after a month. **%d** is the day of the month.

  For old log files like “31”, you can right click **order by modified date** on Windows, or **ls -t** on Linux, then you can know this “31” is this month’s or last month’s.

  Note that this configuration could increase the space used by the PostgreSQL log files.

Always Exclude PostgreSQL Data Folder from Anti-Virus Scan

When a third-party Anti-Virus software scans the PostgreSQL server data folder, it could cause the PostgreSQL data tables to corrupt.

To resolve the issue, always make sure that the PostgreSQL install directory is added into the anti-virus software exclusion list. The exclusion of data files will not introduce any potential security risk.

If you need to run weekly-based scan, monitor the **probeerror.log** file, and if a database error shows up (for example, a database error related to the Discovery_result table), do the following:

1. Clean the probe log folder.
2. Run **Clear Probe Results Cache** from **UCMDB UI > Data Flow Management > Universal**

Discovery > Discovery Modules/Jobs to clean the problematic table.

This should resolve the issue.

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Database Guide (Universal CMDB 10.33)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to cms-doc@hpe.com.

We appreciate your feedback!