Oracle® Database Mobile Server

Troubleshooting and Tuning Guide

Release 11.2

E35798-01

September 2012

ORACLE®

Oracle Database Mobile Server Troubleshooting and Tuning Guide Release 11.2

E35798-01

Contents

2   Troubleshooting

Preface

This preface introduces you to the *Oracle Database Mobile Server Troubleshooting and Tuning Guide*, discussing the intended audience, documentation accessibility, and structure of this document.

Audience

This manual is intended for application developers as the primary audience and for database administrators who are interested in application development as the secondary audience.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

Use the following manual as reference when installing and configuring a WebLogic application server:

- *Oracle® Fusion Middleware Installation Guide for Oracle WebLogic Server*

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.

Convention	Meaning
`...`	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.
`italic monospace`	Italic monospace type indicates a variable in a code example that you must replace. For example: `Driver=`*`install_dir`*`/lib/libtten.sl` Replace *`install_dir`* with the path of your TimesTen installation directory.
`< >`	Angle brackets enclose user-supplied names.
`[ ]`	Brackets enclose optional clauses from which you can choose one or none.

1

Improving Performance

Mobile devices do not have the processing power and memory that standard enterprise systems maintain. If the mobile applications and infrastructure are not tuned appropriately they really are of little benefit to the organization.

The most important performance concepts for a mobile infrastructure are as follows:

- The time it takes to enter and retrieve data.
- The time it takes to synchronize data with the enterprise data store.

The following sections describe the methods you can manage the performance of Oracle Database Mobile Server:

- Section 1.1, "Improving Connection Performance"
- Section 1.2, "Increasing Synchronization Performance"
- Section 1.3, "Integrating Oracle Database Mobile Server With the Oracle Real Application Clusters"
- Section 1.4, "Maximizing JVM Performance By Managing Java Memory"

1.1 Improving Connection Performance

The following methods enable you to streamline the connections between the client/server and the mobile server and back-end database:

- Section 1.1.1, "Using Connection Pooling for Applications"
- Section 1.1.2, "Limit Application Connection Requests to the Database"

1.1.1 Using Connection Pooling for Applications

Connection pooling enables you to eliminate the time delay in creating and destroying connections for incoming application requests. Instead, enable connection pooling, as shown in Section 3.4, "Managing Application and Connection Properties" in the *Oracle Database Mobile Server Administration and Deployment Guide*, so that each incoming connection request uses an existing connection from the pool.

1.1.2 Limit Application Connection Requests to the Database

You can limit the number of connections that access the database from each application, as shown in Section 3.4, "Managing Application and Connection Properties" in the *Oracle Database Mobile Server Administration and Deployment Guide*. Set the maximum database connection limit. Any request for a database connection beyond the limit is refused.

1.2 Increasing Synchronization Performance

The following sections describe how you can manipulate the synchronization performance:

- Section 1.2.1, "Analyzing Performance of Publications With the Consperf Utility"

- Section 1.2.2, "Monitoring Synchronization Using SQL Scripts"

- Section 1.2.3, "Create SQL Scripts With All Dependencies"

- Section 1.2.4, "Configuration Parameters in the MOBILE.ORA that Affect Synchronization Performance"

- Section 1.2.5, "Tuning Queries to Manage Synchronization Performance"

- Section 1.2.6, "Synchronization Tablespace Layout"

- Section 1.2.7, "Shared Maps"

- Section 1.2.8, "Use Map Table Partitions to Streamline Users Who Subscribe to a Large Amount of Data"

- Section 1.2.9, "Configuring Back-End Oracle Database to Enhance Synchronization Performance"

- Section 1.2.10, "Priority-Based Replication"

- Section 1.2.11, "Caching Publication Item Queries"

- Section 1.2.12, "Architecture Design of Mobile Server and Oracle Database for Synchronization Performance"

- Section 1.2.13, "Designing Application Tables and Indexes for Synchronization Performance"

1.2.1 Analyzing Performance of Publications With the Consperf Utility

The Consperf utility profiles your subscriptions and may modify how the publication item is executed if the utility determines that there is a more performant option. The Consperf tool evaluates how the SQL within the publication item interacts with our Data Synchronization query templates. The first synchronization is always a complete refresh, which is a direct invocation of the query. On subsequent synchronizations, the query templates determine incremental refreshes. This improves your performance from not having to perform a complete refresh each time you synchronize. However, the interaction of our query templates and your SQL may not be optimal, which is discovered by the Consperf tool. We either modify the query template or type of logical delete or insert for you or you can adjust your SQL to be more performant in regards to our templates.

In addition, application developers and administrators use this utility to analyze the performance of subscriptions and identify potential bottlenecks during synchronization.

This tool generates the following two primary analysis reports:

1. Timing statistics for publication items

2. Explain plans for publications

The Consperf tool automatically tunes subscription properties, if the default templates do not supply the highest performing option. You can select a client and choose the desired subscription for performance analysis. Users can change parameter values before analyzing performance. The analysis results, which are timing and execution

plan reports, are stored on the server and can be accessed by viewing the same user and subscription.

You can execute the Consperf utility through one of the following locations:

- Click the Users link under the Consperf section on the Performance tab.

- Click the Users link from the Repository screen.

Then, perform the following:

1. Select the User that you want to execute the Consperf tool against and click **Subscriptions**.

2. From the subscriptions screen, choose the publication and click **Consperf performance analysis**. This starts the Consperf analysis.

3. Click **Set consperf parameters and launch the consperf thread**, which brings you to a screen where you can configure parameters that effect how the performance analysis is executed. See Section 1.2.1.1, "Deciphering the Performance Evaluation Files" for more information on these parameters and how they effect the performance evaluation output.

4. Once you have set the configuration for how you want your performance analysis to occur, click **OK**. The Consperf tool executes and prepares the reports for you, based on your configuration. You are returned to the first Consperf page with the reports listed as hyperlinks under the **Last Consperf Run Results** section as **View Timing File** or **View Execution Plan File**.

1.2.1.1 Deciphering the Performance Evaluation Files

There are two performance evaluatons that come out of the Consperf utility:

- Section 1.2.1.1.1, "Timing File"

- Section 1.2.1.1.2, "Configuration for Data Synchronization"

- Section 1.2.1.1.3, "Execution Plan File"

1.2.1.1.1 Timing File The timing file contains the analysis of how the publication item performs with the data synchronization defaults against how it could perform if other options were chosen. The output of this file shows you the conclusions of the analysis and how the data synchronization defaults could be modified to perform better with your particular publication items.

The first section of the timing file provides you information on the configuration with which this analysis was executed. Thus, if you modify the configuration for other analysis, you can go back and compare each file to each other to easily see the differences in the output.

> **Note:** The results of this analysis may cause the data synchronization engine to modify the type of query template or logical delete/insert/update used with your publication item. To change it back to the defaults, you will have to rerun Consperf with `CLEARTUNE` set to `YES`. See Table 1–2 for a full description of parameter settings.

The following example shows the publication that is examined is the `T_SAMPLE11` publication. The version of the Oracle Database Mobile Server is 11.2.0.0.0. The user is `S11U1`. And the configuration is set to time out if the query takes longer that 1000

milliseconds and change the defaults if the difference between the default and the other templates are greater than 20 seconds (20000 milliseconds). The command that authorizes the changes is when `AUTOTUNE` is set to true. If set to false, the analysis is provided, but nothing is modified.

```
VERSION = 11.2.0.0.0
OPTMIZER_MODE = null
APPLICATION = null
PUBLICATION = T_SAMPLE11
CLIENTID = S11U1
TIMEOUT = 1000 ms
TOLERANCE = 20000 ms
ITERATIONS = 2
AUTOTUNE_SUPPORT = true
```

The next part of the Timing File lists the time in milliseconds each template type takes to complete with each publication item in the publication. There are three templates that data synchronization can use to "wrap" your SQL query. The default query template is `SYNC_1`. Since the tolerance is set to 20 seconds, then if either template `SYNC_2` or `SYNC_3` perform at least 20 seconds bettern than `SYNC_1`, then the template type will be modified for your publication item. You can set the `TOLERANCE` level to fewer seconds in the Consperf configuration. See Table 1–2 for a description of `TOLERANCE`.

```
Publication Item Name | NS | BS | SYNC_1 | SYNC_2 | SYNC_3 | AS | Total
 ----------------------------------------------------------------------
P_SAMPLE11-D          | <3> | <0> | <6>   | 10     | -1000  | <0> | 9
P_SAMPLE11-M          | <3> | <0> | <5>   | 8      | -1000  | <0> | 8
```

- There are two publication items in the subscription.

- NS stands for Null Sync. Your application may be issuing a null synchronization. If so, this shows the time in milliseconds that it took to complete. The null synchronization is a tool to see if it is the data that is causing the performance hit or the application itself.

- BS stands for Before Synchronization; AS stands for After Synchronization. You can provide callouts that are executed either before or after each synchronization for this application. This shows the time in milliseconds it takes to perform each task. In this example, there is no before or after synchronization callouts.

- `SYNC_1` is the default template. In combination with the publication items, it still is executing the fastest as compared to the other two options: `SYNC_2` and `SYNC_3` with 6 and 5 milliseconds for each publication item respectively. Thus, these publication items will continue to use `SYNC_1` template. Note that `SYNC_3` has -1000 as its time. That either means that the template was not appropriate to execute or that it timed out.

  - `SYNC_1` uses an outer-join for inserts, updates, and deletes

  - `SYNC_2` is a simple insert and update

  - `SYNC_3` uses the base view for insert and update. The base view is the first table in the select statement, as it is the primary key used to search for all records in the query.

- The total is the total number of milliseconds to execute the entire publication item.

The second section is how the MGP performs with the templates it uses for deletes and inserts. It evaluates the default against other options, as follows:

- Logical delete options:

- MGP template for logical deletes using `EXISTS`: default for logical delete.

- MGP template for logical deletes using correlated `IN`.

- MGP template for logical deletes using `HASH_AJ`.

- MGP template for logical deletes using `IN`.

- Logical insert options:

  - MGP template for logical inserts using `EXISTS`: default for logical insert.

  - MGP template for logical inserts using correlated `IN`.

  - MGP template for logical inserts using `IN`.

- Logical update options

  - MGP template for logical updates using correlated `IN`: default for logical updates.

  - MGP template for logical updates using `EXISTS`.

  - MGP template for logical updates using `IN`.

- MGP template for logical updates with multiple table dependencies.

For example, the following evaluates how each publication item performs with its logical deletes:

```
MGP Output...
Pub Item Name | LDEL_1 | LDEL_2 | LDEL_3 | LDEL_4
P_SAMPLE11-D  | <5>    | 3      | 3      | 3
P_SAMPLE11-M  | <5>    | 3      | 5      | 4
```

The `LDEL_1` is the default and even though `LDEL_2` , 3 and 4 are faster, they are not 20 seconds faster, which is the tolerance level. So, the default for deletes is kept the same. If the difference in speed had been greater than the tolerance level, the Consperf utility would have modified the logical delete method in the repository for the publication item in future—if the autotune parameter was set to yes.

The last section, Subscription Properties, describes the following:

- Profiled: Has autotune been turned on and Consperf executed previously on this subscription?

- Base View: True if this publication item uses more than one table.

- How many records are in the subscription.

- How many records are dirty?

- How many records have been flagged as dirty to simulate an actual run? Up to the number of records in the subscription or `MAXLOG` will be flagged as dirty, whichever is least.

1.2.1.1.2 Configuration for Data Synchronization

Table 1–1   Consperf Parameters for Both Synchronization and MGP Processing

Parameter	Default Value	Allowed Values	Description
PUBITEMLIST	*<ALL>*	*Pub1*, *Pub2*, and so on.	Specifies comma-separated list of publication items to process. The default is all publication items in the publication.

Table 1–1   (Cont.) Consperf Parameters for Both Synchronization and MGP Processing

Parameter	Default Value	Allowed Values	Description
SKIPPUBITEMLIST	*<NONE>*	*Pub1*, *Pub2*, and so on.	Specifies comma-separated list of publication items to skip.
OPTIMIZER	*<DB>*	Can set to RULE or CHOOSE; otherwise sets to what database is set to.	Specifies the optimizer mode to use within Oracle. The default is the current database setting.
ORDERBYPUBITEM	NO	Yes or No	Orders all output by publication item name.

Table 1–2   Consperf Parameters for Synchronization Timing Performance

Parameter	Default Value	Allowed Values	Description
TIMEOUT	10 seconds	Integer for seconds	Specifies the query timeout value in seconds. This is the amount of time Consperf will wait before it cancels a query.
UPDATECOUNT	5	Integer for number of records	Specifies the number of records to mark as dirty during synchronization.
MAXLOG	5000	Integer for number of records	Specifies the number of records to put in the log table. Simulates the transaction log
AUTOTUNE	NO	Yes or No	Enables auto-tune.
CLEARTUNE	NO	Yes or No	Clears existing auto-tune results.
TOLERANCE	20 seconds	Integer for seconds	A template must be faster by this number of seconds before it replaces the default template.

1.2.1.1.3 Execution Plan File To improve performance when accessing data on the local client database, view the execution plan file, which shows the performance of your SQL query execution on the client database. To execute a SQL statement, Oracle might need to perform several steps. Each of these steps either physically retrieves rows of data from the database or prepares them in some way for the user issuing the statement. The combination of the steps Oracle uses to execute a statement is called an execution plan, which includes an access path for each table that the statement accesses and an ordering of the tables (the join order) with the appropriate join method. The execution plan shows you exactly how Oracle Database Mobile Server executes your SQL statement.

The components of an execution plan include the following:

- An ordering of the tables referenced by the statement.

- An access method for each table mentioned in the statement.

- A join method for tables affected by join operations in the statement.

The execution plan file shows how your publication items interact with the different logical delete, insert, and update templates. From this report, you can evaluate your SQL to see if you want to modify it in any way to speed up your query. Set the optimizer parameter to designate how the database is organized. If you set this parameter to a setting that the database is not set to, it still acts as if the database is set

to this way to show you how it would execute. See Table 1–3 for all configuration parameters that relate to this search.

Table 1–3   Consperf Parameters for Execution Performance Plan

Parameter	Default Value	Allowed Values	Description
GATHERSTATS	NO	Yes or No	Gathers optimizer statistics on all mobile server objects. MGP compose must be disabled while Consperf analyzes objects. Consperf blocks this automatically, but the safest approach is to manually stop the MGP process before running Consperf with the GATHERSTATS option. If Consperf fails while gathering statistics, users must re-run CLEARSTATS before starting the MGP process again.
CLEARSTATS	NO	Yes or No	Removes optimizer statistics on mobile server objects.
SQLTRACE	NO	Yes or No	Enables Oracle SQL trace. TKPROF can be used to analyze the resulting trace file.

1.2.2 Monitoring Synchronization Using SQL Scripts

If, instead of viewing MGP statistics within the Mobile Manager, you would rather execute SQL scripts to monitor mobile application status during synchronization, you may use any of the following SQL scripts to retrieve the desired information.

- Section 1.2.2.1, "Synchronization Times for All Clients"

- Section 1.2.2.2, "Failed Transactions for all Clients"

- Section 1.2.2.3, "Completely Refreshed Publication Items for all Clients"

- Section 1.2.2.4, "Publications Flagged for Complete Refresh for All Clients"

- Section 1.2.2.5, "Clients and Publication where Subscription Parameters are Not Set"

- Section 1.2.2.6, "Record Counts for Map-based Publication Item by Client"

- Section 1.2.2.7, "Record Count for Map-based Publication Items by Store"

- Section 1.2.2.8, "All Client Sequence Partitions and Sequence Values"

- Section 1.2.2.9, "All Publication Item Indexes"

1.2.2.1 Synchronization Times for All Clients

Using the following script, you can check the latest successful synchronization times for all clients by retrieving such information from the `all_clients` table.

```
select client, lastrefresh_starttime, lastrefresh_endtime
from cv$all_clients
order by client
/
```

1.2.2.2 Failed Transactions for all Clients

Using the following script, you can retrieve a list of failed transactions for all clients from the `all_errors` table.

```
select client, transaction_id, item_name, message_text
from cv$all_errors
```

```
where message_text is not null
order by client,transaction_id
/
```

1.2.2.3 Completely Refreshed Publication Items for all Clients

Using the following SQL script, you can retrieve a list of all publication items for all clients which were completely refreshed during the last synchronization process.

```
select clientid, publication_item
from c$complete_refresh_log
order by clientid, publication_item
/
```

1.2.2.4 Publications Flagged for Complete Refresh for All Clients

Using the following SQL script, you can retrieve a list of publications for all clients that are flagged for a complete refresh during the next synchronization process.

```
select clientid, template as publication
from c$all_subscriptions
where crr = 'Y'
/
```

1.2.2.5 Clients and Publication where Subscription Parameters are Not Set

Using the following SQL script, you can retrieve a list of clients and their publications where the subscription parameters have not been set.

```
select client, name as publication, param_name, param_value
from cv$all_subscription_params
where param_value is null
order by client, name
/
```

1.2.2.6 Record Counts for Map-based Publication Item by Client

Using the following script, you can retrieve record counts for all clients in queues for map-based publication items, that are grouped by clients.

```
select clid$$cs as client, count(*) as "RECORD COUNT"
from c$in_messages
group by clid$$cs
/
```

1.2.2.7 Record Count for Map-based Publication Items by Store

Using the following SQL script, you can retrieve record counts for all client in-queues for map-based publication items, that are grouped by store.

```
select clid$$cs as client, tranid$$ as transaction_id, store as item_name,
count(*) as "RECORD COUNT"
from c$in_messages
group by clid$$cs, tranid$$, store
/
```

1.2.2.8 All Client Sequence Partitions and Sequence Values

Using the following SQL script, you can retrieve a list of all client sequence partitions and current sequence values.

```
select clientid, name, curr_val, incr
from c$all_sequence_partitions
```

```
order by clientid, name
/
```

1.2.2.9 All Publication Item Indexes

Using the following SQL script, you can retrieve a list of all publication item indexes.

```
select publication as NAME, publication_item, conflict_rule as "INDEX_TYPE",
columns
from c$all_indexes
order by publication, publication_item
/
```

1.2.3 Create SQL Scripts With All Dependencies

When you create a SQL script in MDW or with the Consolidator APIs, you should include all dependent DDL statements in the same script in the order necessary. If you separate dependent DDL statements into separate scripts, Oracle Database Mobile Server may be executing them randomly, causing dependency errors and re-execution of each script. See Section 4.7 "Create and Load a Script into the Project" in the *Oracle Database Mobile Server Developer's Guide* for more information.

1.2.4 Configuration Parameters in the MOBILE.ORA that Affect Synchronization Performance

The following parameters in the `[CONSOLIDATOR]` section of the `mobile.ora` file are used for tuning synchronization:

- `MAX_THREADS`

  The `MAX_THREADS` parameter is used by the MGP and controls the number of concurrent threads. As a rule, do not set this higher than 1.5 times the number of CPUs on the database machine. For example, if your system has four CPUs, theb you should not set it higher than six.

- `RESUME_MAXACTIVE` and `RESUME_MAX_WAIT`

  You can configure for maximum concurrent clients with the `RESUME_MAXACTIVE` and `RESUME_MAX_WAIT` parameters. This limits the maximum number of concurrently synchronizing clients to `RESUME_MAXACTIVE`; additional incoming clients wait `RESUME_MAX_WAIT` in minutes before timing out with an error. Maximum concurrent clients are configured without the resume feature with these parameters if you set `RESUME_TIMEOUT=0`.

  For more details on the resume feature, see Section 4.7, "Resuming an Interrupted Synchronization" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

- `COMPOSE_TIMEOUT`

  The `COMPOSE_TIMEOUT` parameter specifies in seconds the MGP timeout for the compose phase for each user.

- `CONNECTION_POOL`

  The `CONNECTION_POOL` parameter enables pooling of database connections.

- `MAX_THREADS`

  The `MAX_THREADS` parameter sets the maximum number of threads spawned within the MGP process.

For full details on these and more parameters, see Section A.1.5, "CONSOLIDATOR" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

Each synchronization request requires a number of system resources, such as creating a database connection, using memory, and so on. If you have too many requests competing for the same resources, then the overall performance can be poor. Limiting the number of parallel requests with the MAX_THREADS and MAX_CONCURRENCY parameters improve the average response time.

Set the MAX_THREADS and MAX_CONCURRENCY parameters if you notice that the synchronization performance is not linear. For example, if twice the number of parallel requests results in a synchronization time that is five times longer for each client, then you probably have resource contention. The value depends on your environment and should be determined on a trial and error basis.

1.2.5 Tuning Queries to Manage Synchronization Performance

You can increase synchronization performance by monitoring the performance of the SQL queries in your applications. The following sections provide details on how to tune your queries:

- Section 1.2.5.1, "Avoid Using Non-Mergable Views"
- Section 1.2.5.2, "Tune Queries With Consperf Utility"
- Section 1.2.5.3, "Manage the Query Optimizer"

1.2.5.1 Avoid Using Non-Mergable Views

You should avoid using database query constructs that prevent a view from being mergable, as publication item queries that use non-mergable views do not perform well. Examples of such constructs are union, minus, and connect by. For more information on mergable views, see the Oracle database documentation.

1.2.5.2 Tune Queries With Consperf Utility

Once you have defined your application, use the consperf utility to profile the performance of the publication item queries. The mobile server does not execute your publication item queries directly; instead the query is wrapped into a template query, which is executed by the mobile server. The template query may have an unexpected query execution plan, resulting in poor performance. The consperf utility generates an EXPLAIN PLAN execution plan for those template queries, allowing you to tune your publication item query for best performance. In addition, consperf generates timing information for the execution of all template queries, so that you can identify bottleneck queries. For more information on the consperf utility, see Section 1.2.1, "Analyzing Performance of Publications With the Consperf Utility".

1.2.5.3 Manage the Query Optimizer

You must make sure that the optimizer picks the correct execution path when you either are using the cost-based optimizer or you have set the optimizer settings to choose. The optimizer can pick the correct execution path only when all of the tables are properly analyzed and statistics have been gathered for these tables.

The mobile server uses temporary tables during synchronization. Once a number of users have been created, and they have synchronized with the mobile server, run consperf with the gatherstats option to generate the statistics information for the temporary tables. For more information on the consperf utility, see Section 1.2.1, "Analyzing Performance of Publications With the Consperf Utility".

1.2.6 Synchronization Tablespace Layout

Tablespace layout across multiple disks can improve the performance of the mobile server data synchronization, as it reduces movement of the disk heads and improves I/O response time.

By default, the synchronization tablespace is SYNCSERVER, and is stored in the mobile*xx*.dbf file in the default location for the database under *ORACLE_HOME*, where *xx* is a number between 1 and 25. The tablespace name, filename, and file location for the tablespace is defined in the $ORACLE_HOME/Mobile/Server/admin/consolidator_o8a.sql script file, which is executed during the mobile server installation process. So, if you want to modify the tablespace, perform the following BEFORE you install the mobile server; otherwise, the default tablespace is created.

If you want to customize the SYNCSERVER tablespace, for example, by using multiple data files spread across several disks, or by using specific storage parameters, then you can precreate the SYNCSERVER tablespace with the required settings. The installation process automatically detects that the tablespace exists and uses it. Refer to the Oracle Database documentation for full details on how to create a tablespace.

1.2.7 Shared Maps

It is very common for publications to contain publication items that are used specifically for lookup purposes. That is, a publication item that creates a read-only snapshot. The server may change these snapshots, but the client would never update them directly. Furthermore, many users often share the data in this type of snapshot. For example, there could be a publication item called zip_codes, which is subscribed to by all mobile users.

The main function of Shared Maps is to improve scalability for this type of publication item by allowing users to share record state information and reduce the size of the resulting replication map tables. By default, if you have a non-updatable publication item, it defaults to using shared maps.

> **Note:** Shared Maps can also be used with updatable snapshots if the developer is willing to implement their own conflict detection and resolution logic; however, normally shared maps are only for non-updatable snapshots.

Shared maps shrink the size of map tables for large lookup publication items and reduce the MGP compose time. Lookup publication items contain read-only data that is not updatable on the clients and that is shared by multiple subscribed clients. When multiple users share the same data, their query subsetting parameters are usually identical.

For example, a query could be the following:

```
SELECT * FROM  WHERE EMP WHERE DEPTNO = :dept_id
```

In the preceding example, all users that share data from the same department have the same value for dept_id. The default sharing method is based on subscription parameter values.

In the following example, the query is:

```
SELECT * FROM WHERE EMP WHERE DEPTNO = ( SELECT DEPTNO FROM
        EMP WHERE EMPNO = :emp_id )
```

In this example, users from the same departments still share data. Their subsetting parameters are not equal, because each user has a unique emp_id. To support the sharing of data for these types of queries (as illustrated by the example), a grouping function can be specified. The grouping function returns a unique group id based on the client id.

There is also another possible use for shared maps. It is possible to use shared maps for shared updatable publication items. However, this type of usage requires implementation of a custom **dml** procedure that handles conflict resolution.

1.2.7.1 Performance Attributes

The performance of the MGP compose cycle is directly proportional to the following:

```
NC * NPI
```

where:

- NC = number of clients

- NPI = number of publication items that must be composed

With shared maps, the length of the MGP cycle is proportional to the following:

```
NC*(NPI - NSPI)  + NG*NSPI
```

where:

- NSPI = number of shared publication items

- NG = number of groups

> **Note:** If NG = NC, then the MGP performance is similar in both cases. However, with fewer groups and more shared publication items, the MGP compose cycle becomes faster. In addition, map storage requirements are governed by these same factors.

1.2.7.2 Shared Map Usage

To set up a publication item to be shared, use the AddPublicationItem API and enable the shared flag. It is also possible to toggle the shared property of a publication item once it is added to the publication with the SetPublicationItemMetadata API. Both the AddPublicationItem API and the SetPublicationItemMetadata API allow users to specify a PL/SQL grouping function. The function signature must be as follows:

```
(
CLIENT in VARCHAR2,
PUBLICATION in VARCHAR2,
ITEM in VARCHAR2
)return VARCHAR2.
```

The returned value must uniquely identify the client's group. For example, if client **A** belongs to the group **GroupA** and client **B** belongs to the group **GroupB**, the group function **F** could return:

```
F ('A','SUBSCRIPTION','PI_NAME') = 'GroupA'
F ('B','SUBSCRIPTION','PI_NAME') = 'GroupB'
```

The implicit assumption of the grouping function is that all the members of the **GroupA** group share the same data, and that all the members of the **GroupB** group share the same data.. The group function uniquely identifies a group of users with the same data for a particular **PUBLICATION ITEM**.

For the query example in Section 1.2.7, "Shared Maps", the grouping function could be:

```
Function get_emp_group_id (
  clientid in varchar2,
  publication in varchar2,
  item in varchar2
) return varchar2 is
  group_val_id varchar2(30);
begin
  select DEPTNO into group_val_id
  from EMP where EMPNO = clientid ;
  return group_val_id;
end;
```

> **Note:** This function assumes that `EMPNO` is the Consolidator Manager client id. If the `group_fnc` is not specified, the default grouping is based on subscription parameters.

1.2.7.3 Compatibility and Migration for Shared Maps

If you have been using a version prior to Oracle Database Mobile Server 11*g*, then you must migrate your existing mobile server schema with shared maps, as follows:

1. Run one cycle of MGP.

2. The clients must sync with the server to get the latest changes prepared by the MGP.

3. Stop the Web server and MGP to migrate the server to 11*g*. This automatically sets all the nonupdatable publication items to shared items. If any shared publication items need to use grouping functions or any publication items need to change their sharing attribute, execute custom code that calls the appropriate Consolidator Manager API. See the `SetPublicationItemMetadata` API in Section 1.2.7.2, "Shared Map Usage".

4. The `ShrinkSharedMaps` Consolidator Manager API must be called to set the clients to use shared map data and remove old redundant data from the maps.

5. Start the Web server and MGP.

1.2.8 Use Map Table Partitions to Streamline Users Who Subscribe to a Large Amount of Data

Sync Server database objects called map tables are used to maintain the state for each mobile client. If there are a large number of clients, and each client subscribes to a large amount of data, the map tables can become very large creating scalability issues. Using the following APIs, map tables can be partitioned by client id, making them more manageable.

The API allows you to create a map table partition, add additional partitions, drop one or all partitions, and merge map table partitions. Map table partitions can be monitored using the `ALL_PARTITIONS` database catalog view.

> **Note:** This form of partitioning is not related to the partition functionality provided by Oracle Server, and is used exclusively by Oracle Database Mobile Server.

1.2.8.1 Create a Map Table Partition

Creates a partition for the referenced publication item map table. If there is data in the map table, it is transferred to the partition being created. After the partition has been successfully created, the map table can be truncated to remove redundant data using the SQL command TRUNCATE TABLE.

> **Note:** Records removed from the server through a `truncate` command will not be removed from the client unless a complete refresh is triggered. The truncate command is considered a DDL operation. Consequently, the necessary DML triggers do not fire and therefore the operations are not logged for fast refresh.

Syntax

```
public static void partitionMap
   (String pub_item,
    int num_parts,
    String storage,
    String ind_storage) throws Throwable
```

The parameters of `partitionMap` are listed in Table 1–4.

Table 1–4   The partitionMap Parameters

Parameter	Definition
pub_item	The publication item whose map table is being partitioned.
num_parts	The number of partitions.
storage	A string specifying the storage parameters. This parameter requires the same syntax as the SQL command CREATE TABLE. See the *Oracle SQL Reference* for more information.
ind_storage	A string specifying the storage parameters for indexes on the partition. This parameter requires the same syntax as the SQL command CREATE INDEX. See the *Oracle SQL Reference* for more information.

Example

```
consolidatorManager.partitionMap("P_SAMPLE1", 5, "tablespace mobileadmin",
 "initrans 10 pctfree 70");
```

1.2.8.2 Add Map Table Partitions

Adds a partition for the referenced publication item's map table. If there is data in the map table, it is transferred to the partition being created. After the partition has been successfully created, the map table can be truncated to remove redundant data using the SQL command TRUNCATE TABLE.

> **Note:** Records removed from the server through a `truncate` command will not be removed from the client unless a complete refresh is triggered. The truncate command is considered a DDL operation. Consequently, the necessary DML triggers do not fire and therefore the operations are not logged for fast refresh.

Syntax

```
public static void addMapPartitions
   ( String pub_item,
    int num_parts,
    String storage,
    String ind_storage) throws Throwable
```

The parameters of `addMapPartitions` are listed in Table 1–5:

Table 1–5   The addMapPartitions Parameters

Parameter	Definition
pub_item	The publication item whose map table is being partitioned.
num_parts	The number of partitions.
storage	A string specifying the storage parameters. This parameter requires the same syntax as the SQL command `CREATE TABLE`.
ind_storage	A string specifying the storage parameters for indexes on the partition. This parameter requires the same syntax as the SQL command `CREATE INDEX`.

Example

```
consolidatorManager.addMapPartitions("P_SAMEPLE1",5,"tablespace
 mobileadmin","initrans 10 pctfree 40");
```

> **Note:** Map Partitions are created only for existing users. New users are placed in the original map table.

1.2.8.3 Drop a Map Table Partition

Drops the named partition. In the following example, the `partition` parameter is the name of the partition. Partition names must be retrieved by querying the `ALL_PARTITIONS` table view `CV$ALL_PARTITIONS` since partitions are named by Data Synchronization.

Syntax

```
public static void dropMapPartition( String partition) throws Throwable
```

Example

```
consolidatorManager.dropMapPartition("MAP101_1");
```

1.2.8.4 Drop All Map Table Partitions

Drops all partitions of the map table for the named publication item.

Syntax

```
public static void dropAllMapPartitions( String pub_item) throws Throwable
```

Example

```
consolidatorManager.dropAllMapPartitions("P_SAMPLE1");
```

1.2.8.5 Merge Map Table Partitions

Merges the data from one partition into another. Partition names must be retrieved by querying the ALL_PARTITIONS table view CV$ALL_PARTITIONS, since partitions are named by Data Synchronization.

Syntax

```
public static void mergeMapPartitions
   ( String from_partition,
    String to_partiton) throws Throwable
```

Example

```
consolidatorManager.mergeMapPartition(""MAP101_1", "MAP101_2");
```

1.2.9 Configuring Back-End Oracle Database to Enhance Synchronization Performance

You can configure the Oracle Database in such a way as to enchance your mobile server synchronization performance, as follows:

- Section 1.2.9.1, "Physically Separate Map Tables and Map Indexes"
- Section 1.2.9.2, "Database Parameter Tuning"

1.2.9.1 Physically Separate Map Tables and Map Indexes

During synchronization, map tables are used extensively. Map tables are internal tables, and have table names using the following pattern: CMP$pub_item_name. Each map table has four separate indexes. By default, both map table and indexes are created in the default tablespace SYNCSERVER.

You can improve performance if you move the map table indexes to a different disk than the map table itself. Create a separate tablespace (for example: MAPINDEXES) on a different disk and manually move all indexes. Because the process of moving the indexes requires you to drop and re-create the indexes, you should move the index before many users have synchronized. Otherwise recreating the indexes on the map tables may be very time consuming, as map tables grow with the number of users who have synchronized.

To move the indexes on a map table, do the following:

1. Identify all indexes on the map table (CMP$pub_item_name). There are three or four indexes. Move all of them.

2. For each index, record the type of index and column lists.

3. If the index is a primary key index, then remove the primary key constraint on the map table.

4. Drop the index.

5. Recreate the index using the same name, type and column list. Use the storage clause in the create index statement to specify the new tablespace. You may also specify different storage parameters. Refer to the Oracle database documentation for more information on how to create indexes and storage clause parameters.

> **Note:** Repeat step 3 through 5 for all other indexes on the map table.

1.2.9.2 Database Parameter Tuning

Tuning the database for the mobile server is similar to any Oracle database tuning that is required for query intensive applications. Configure the SGA to be as large as possible on your system to maximize the caching capabilities and avoid I/O wherever possible.

Tune your Oracle database with the following database parameters:

- `db_block_buffers`
- `sort_area_size`
- `log_buffers`

Refer to the Oracle database tuning guide for more information on database tuning.

1.2.10 Priority-Based Replication

With priority-based replication, you can specify what rows in the snapshot are synchronized by setting the priority.

There are two steps to enable this feature. These are described in the following sections:

1. Section 1.2.10.1, "Create Restricting Predicate in Publication Item"
2. Section 1.2.10.2, "Set Priority Flag in Mobile Sync API Before Initiating Synchronization"

1.2.10.1 Create Restricting Predicate in Publication Item

Create a Restricting Predicate expression in the publication item that is to be restricted when priority is requested. A Restricting Predicate is a conditional expression added to the SQL statement in the snapshot. It limits what records are sent to the client. Create the restricting predicate with either the `addPublicationItem` method or the MDW equivalent.

> **Note:** You can only use fast refresh with a high priority restricting predicate. If you use any other type of refresh, the high priority restricting predicate is ignored.
>
> High-priority replication combined with the selective synchronization feature provides a mechanism for minimizing the synchronization payload to the most relevant data.

For example, if you created a snapshot with the following statement:

```
select * from projects where urgency_level in (1,2,3,4)
```

The developer can set the Restricting Predicate with the `addPublicationItem` method . For example, you could create a Restricting Predicate with the `urgency_level` as follows:

```
consolidatorManager.addPublicationItem("T_SAMPLE11", "P_SAMPLE11-M",
        null, null, "S", "urgency_level = 1", null);
```

This specifies what the priority synchronization should do when requested. To request the priority synchronization, set the high priority flag in the Mobile Sync APIs, as described in Section 1.2.10.2, "Set Priority Flag in Mobile Sync API Before Initiating Synchronization".

1.2.10.2 Set Priority Flag in Mobile Sync API Before Initiating Synchronization

Once the publication item has the Restricting Predicate set, then when you want the priority restriction to occur for a synchronization, then you set the priority flag to 1 in the Mobile Sync API.

In each of the Mobile Sync APIs, there is a priority flag in either the environment or options. Setting this flag to 1 means that only high priority tables are synchronized. The default for the priority flag is 0, which specifies that all tables synchronize.

> **Note:** The Mobile Sync API is documented in Chapter 3, "Managing Synchronization on the Mobile Client" in the *Oracle Database Mobile Server Developer's Guide*.

In our previous example, when you set the priority flag to 0 (the default), all projects with urgency_level 1,2,3,4 are replicated. However, if you set the priority flag to 1, then the Restricting Predicate is enabled for the synchronization.

When the high priority flag is set, MGP appends (AND) the Restricting Predicate to the snapshot definitions when composing data for the client. In this example, the high priority statement would be transformed as follows:

```
SELECT * FROM projects where urgency_level in (1,2,3,4) AND urgency_level = 1;
```

In this case, only projects with urgency_level =1 are replicated to the client.

1.2.11 Caching Publication Item Queries

This feature allows complex publication item queries to be cached. This applies to queries that cannot be optimized by the Oracle query engine. By caching the query in a temporary table, the Sync Server template can join to the snapshot more efficiently.

Storing the data in a temporary table does result in additional overhead to MGP operation, and the decision to use it should only be made after first attempting to optimize the publication item query to perform well inside the Sync Server template. If the query cannot be optimized in this way, the caching method should be used.

The following example is a template used by the MGP during the compose phase to identify client records that are no longer valid, and should be deleted from the client:

```
UPDATE pub_item_map map
SET delete = true
WHERE client = <clientid>
AND NOT EXISTS (SELECT 'EXISTS' FROM
    (<publication item query>) snapshot
     WHERE map.pk = snapshot.pk);
```

In this example, when <publication item query> becomes too complex, because it contains multiple nested subqueries, unions, virtual columns, connect by clauses, and other complex functions, the query optimizer is unable to determine an acceptable plan. This can have a significant impact on performance during the MGP compose phase. Storing the publication item query in a temporary table, using the publication

item query caching feature, flattens the query structure and enables the template to effectively join to it.

1.2.11.1 Enabling Publication Item Query Caching

The following API enables publication item query caching.

Syntax

```
public static void enablePublItemQueryCache(String name)
      throws Throwable
```

The parameters for `enablePublItemQueryCache` are listed in Table 1–6:

Table 1–6   The enablePubItemQueryCache Parameters

Parameters	Description
name	A string specifying the name of the publication item.

Example

```
consolidatorManager.enablePubItemQueryCache("P_SAMPLE1");
```

If you are using an input string from the input parameter `argv` array, cast it to a `String`, as follows:

```
consolidatorManager.enablePubItemQueryCache( (String) argv[0]);
```

1.2.11.2 Disabling Publication Item Query Caching

The following API disables publication item query caching.

Syntax

```
public static void disablePubItemQueryCache(String name)
      throws Throwable
```

The name parameter for `disablePubItemQueryCache` is listed in Table 1–7:

Table 1–7   The disablePubItemQueryCache Parameters

Parameters	Description
name	A string specifying the name of the publication item.

Example

```
consolidatorManager.disablePubItemQueryCache("P_SAMPLE1");
```

1.2.12 Architecture Design of Mobile Server and Oracle Database for Synchronization Performance

It is recommended that you execute the mobile server and the Oracle database on separate machines. If possible, use multi-CPU machines for both the mobile server and the Oracle database.

1.2.13 Designing Application Tables and Indexes for Synchronization Performance

Your clients may perform a large number of insert and delete operations on snapshots, and then synchronize their data changes with the mobile server. If this is the case, then

consider placing the application tables and the indexes on those tables on separate disks.

1.3 Integrating Oracle Database Mobile Server With the Oracle Real Application Clusters

Oracle Real Application Clusters (Oracle RAC) enables a single database to run across multiple clustered nodes in a grid, pooling the processing resources of several machines. Oracle RAC enables you to cluster Oracle databases where the combined processing power of the multiple servers provides greater throughput and scalability than is available from a single server.

Oracle RAC is a unique technology that provides high availability and scalability for all application types. The Oracle RAC infrastructure is also a key component for implementing the Oracle enterprise grid computing architecture. Having multiple instances access a single database prevents the server from being a single point of failure. Oracle RAC enables you to combine smaller commodity servers into a cluster to create scalable environments that support mission critical business applications.

> **Note:** For full details on Oracle RAC capabilities, limitations, installation and configuration requirements, see the *Oracle Database* documentation.

You can use Oracle Database Mobile Server with a back-end Oracle RAC database. The mobile server repository is installed in the Oracle RAC database. Then, you can install and configure multiple mobile servers to access the repository in the Oracle RAC database to increase performance and availability.

> **Note:** Section 4.3.3, "Providing High Availability with a Farm of Mobile Servers" in the *Oracle Database Mobile Server Installation Guide* describes how you can install multiple mobile servers interacting with a single Oracle RAC database with a loadbalancer managing the incoming load.

Applications deployed on Oracle Database Mobile Server in an Oracle RAC configuration operate without any code modifications. The only impact is that during installation and configuration of Oracle Database Mobile Server and some of its tools, you need to provide an Oracle RAC URL instead of the regular JDBC URL.

The JDBC URL for an Oracle RAC database can have more than one address in it for multiple Oracle databases in the cluster and follows this URL structure:

```
jdbc:oracle:thin:@(DESCRIPTION=
 (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=TCP)(HOST=PRIMARY_NODE_HOSTNAME)(PORT=1521))
   (ADDRESS=(PROTOCOL=TCP)(HOST=SECONDARY_NODE_HOSTNAME)(PORT=1521))
 )
 (CONNECT_DATA=(SERVICE_NAME=DATABASE_SERVICENAME)))
```

If you are using an Oracle RAC database, you will be asked to supply the JDBC URL for the Oracle RAC database anytime you need to connect to the back-end Oracle database. This includes the following:

- During the mobile server repository installation—which uses the Repository Wizard tool.

- In the Mobile Database Workbench tool when connecting to the database.

- In the Packaging Wizard tool when publishing an application to the database.

- When providing the JDBC URL for the Mobile Manager, MSRDT and WSH tools.

1.4 Maximizing JVM Performance By Managing Java Memory

You can maximize your JVM performance by modifying the amount of memory used by the three areas of Java memory. This is fully described in Section 2.4.1, "Troubleshooting An Out of Memory Error".

2

Troubleshooting

The following sections describe how to troubleshoot the mobile server:

- Section 2.1, "Troubleshooting Synchronization"
- Section 2.2, "Troubleshooting the Mobile Server"
- Section 2.3, "Troubleshooting the Mobile Server Repository"
- Section 2.4, "Troubleshooting JVM Errors"
- Section 2.5, "Troubleshooting Security"

2.1 Troubleshooting Synchronization

The following sections describe how to troubleshoot the synchronization process or what to do in the event of certain synchronization scenarios:

- Section 2.1.1, "Synchronization Errors and Conflicts"
- Section 2.1.2, "Situations Where the Client is Out of Sync that Triggers a Complete Refresh"
- Section 2.1.3, "The "Inconsistent Datatypes" SQLException Received If Order is Not Correct in Query"
- Section 2.1.4, "MGP Compose Postponed Due to Transaction in the In-Queue"
- Section 2.1.5, "Avoiding the Server Busy Warning"

2.1.1 Synchronization Errors and Conflicts

Consult the following sections for details on how to resolve any synchronization errors or conflicts:

- Section 2.1.1.1, "General Synchronization Errors and Conflicts"
- Section 2.1.1.2, "Synchronization Error if Client Device Clock is Inaccurate"
- Section 2.1.1.3, "Synchronization Error After Modifying Client Password"

2.1.1.1 General Synchronization Errors and Conflicts

With the mobile server, you can have the following errors when synchronizing: nullity violations, foreign key constraint violations, or the client updates a row at the same time that the server deletes it.

The mobile server does not automatically resolve synchronization errors. Instead, the mobile server rolls back the corresponding transactions, and moves the transaction operations into the error queue. It is up to the administrator to view the error queue

and determine if the correct action occurred. If not, the administrator must correct and re-execute the transaction. If it did execute correctly, then purge the transaction from the error queue.

A mobile server synchronization conflict occurs if:

- Nullity violations.

- Foreign key constraint violations.

- The client and the server update the same row.

- The client and server create rows with the same primary key values.

- The client deletes the same row that the server updates.

- The client updates a row at the same time that the server deletes it.

See Section 2.10, "Resolving Conflict Resolution with Winning Rules" in the *Oracle Database Mobile Server Developer's Guide* for more information on conflict resolution techniques.

2.1.1.2 Synchronization Error if Client Device Clock is Inaccurate

The client device clock must be accurate within the timezone set on the device before attempting to synchronize. An inaccurate time may result in the following exception during synchronization: CNS: 9026 "Wrong user name or password. Please enter correct value and reSync."

2.1.1.3 Synchronization Error After Modifying Client Password

If you have an active client and change its password on the server, then the client cannot synchronize. Return the password back to its original value on the server and retry the synchronization.

2.1.2 Situations Where the Client is Out of Sync that Triggers a Complete Refresh

When a client is out of sync with the server, any outstanding uploaded transaction from the client is placed in the error queue and a complete refresh is triggered to re-initialize the client data with what is currently on the server.

The following are a list of the situations—ordered from most to least likely—that can trigger a complete refresh for the client:

- Dropping and then republishing the application.

- Synchronizing by the same mobile user from multiple devices on the same platform or from different platforms when the publications are not platform-specific.

- Receiving unexpected server apply phase conditions—such as constraint violations, unresolved conflicts, other database exceptions.

- Modifying the application—such as changing subsetting parameters, or adding or altering publication items.

- Requesting a force refresh from either the server admin or client.

- Two separate applications using the same backend store.

- Unexpected client apply conditions—such as deleting, moving or restoring the client database, database corruption, memory corruption, and other general system failures.

- Loss of transaction integrity between server and client. The server fails post processing after completing a download and disconnecting from the client.

- Data transport corruptions.

2.1.3 The "Inconsistent Datatypes" SQLException Received If Order is Not Correct in Query

If you are creating a fast refresh publication item on a table with a composite primary key, the snapshot query should list the primary key columns in the order that they are present in the table definition. This automatically happens during the column selection when MDW is used or when a SELECT * query is used. Note that the order of the primary key columns in the table definition may be different from those in the primary key constraint definition.

The following example demonstrates what is valid or invalid given the table definition for TAB1:

```
CREATE TABLE TAB1(
 ID1 NUMBER(10) NOT NULL,
 ID2 NUMBER NOT NULL,
 COL1 VARCHAR2(200),
 COL2 VARCHAR2(200),
 ID3 NUMBER(4) NOT NULL);
ALTER TABLE TAB1 ADD CONSTRAINT TAB1_PK PRIMARY KEY (ID3, ID2, ID1);
```

The following are valid snapshot queries:

```
SELECT * FROM TAB1
SELECT ID1,ID2,ID3,COL1,COL2 FROM TAB1
SELECT ID1,ID2,COL1,COL2,ID3 FROM TAB1
```

The following are invalid snapshot queries:

```
SELECT ID3,ID2,ID1,COL1,COL2 FROM TAB1
SELECT ID3,ID2,COL1,COL2 ID1 FROM TAB1
```

Define the table columns where the primary key columns appear before other columns. The order of the primary key columns in the table definition order must match the constraint definition in the snapshot query.

2.1.4 MGP Compose Postponed Due to Transaction in the In-Queue

If the user synchronized and uploaded some more changes after the last apply cycle for a particular user; by default, the MGP must first apply thse changes before it can compose. If this keeps happening, the compose could be postponed beyond what you would like. By default, the MGP tries to avoid a compose phase postponed due to a transaction in the in-queue by performing an apply for any unprocessed in-queue data before doing a new compose. However, if the MGP compose is postponed due to a transaction in the in-queue, you can modify the following parameters to avoid the error:

- SKIP_INQ_CHK_BFR_COMPOSE: By default, this parameter is set to NO. Setting this parameter to YES, then a compose is performed for a client even if there is unprocessed data in the in-queue.

- DO_APPLY_BFR_COMPOSE: By default, this parameter is set to YES. If set to YES, the unprocessed data in the in-queue is applied before a client compose. This parameter takes effect only if SKIP_INQ_CHK_BFR_COMPOSE is set to NO.

For most situations, preserving the default values for these two parameters avoids the occurrence of the MGP Compose postponed error.

2.1.5 Avoiding the Server Busy Warning

The Server Busy warning can be thrown for one of the following reasons:

- When the MGP is processing apply/compose for that user.—To avoid MGP contention with sychronization, MGP should be scheduled to run when few clients are synchronizing. Alternatively, you could use queue-based synchronization, which does not use the MGP at all; thus, avoiding MGP contention with synchronizaiton.

- If a previous synchronization was interrupted for that user and Oracle Database Mobile Server rolls back the transaction—If the Server Busy warning is a result of a long rollback, then Oracle Database recommended tuning steps for rollback operation may reduce the Server Busy state for the client.

2.2 Troubleshooting the Mobile Server

The following sections detail how to troubleshoot the mobile server and its repository:

- Section 2.2.1, "Running the Mobile Server With Tracing Enabled"

2.2.1 Running the Mobile Server With Tracing Enabled

If you experience any difficulty with the mobile server, you can enable tracing in the mobile server.

To enable tracing in the mobile server, set up your environment as described in Chapter 3, "Tracing and Logging".

2.3 Troubleshooting the Mobile Server Repository

The following sections describe how to evaluate, validate and recover the mobile server repository:

- Section 2.3.1, "Troubleshooting the Mobile Server Repository with the Mobile Server Repository Diagnostic Tool"

- Section 2.3.2, "Inspecting Files in the Mobile Repository With the WSH Tool"

- Section 2.3.3, "Modifying IP Address of Machine Where Mobile Repository Exists"

2.3.1 Troubleshooting the Mobile Server Repository with the Mobile Server Repository Diagnostic Tool

Customers may modify the mobile server repository in the back-end database and, without realizing it, violate some of the rules. The Mobile Server Repository Diagnostic Tool (MSRDT) provides a mechanism for the customer to analyze, validate, and debug the mobile server repository.

The Mobile Server Repository Diagnosis Tool will automatically correct errors that it can and prints out all results of what was modified incorrectly or missing.

> **Note:** The output generated is best viewed if you set your column width to 80 or if you pipe it into a file, to view it with Word or WordPad.

- Section 2.3.1.1, "Use the Mobile Server Repository Diagnostic Tool to Validate Your Environment and the Repository"
- Section 2.3.1.2, "Execute the Repository Diagnostics Tool"

2.3.1.1 Use the Mobile Server Repository Diagnostic Tool to Validate Your Environment and the Repository

You can use the Mobile Server Repository and Diagnostic Tool (MSRDT) to validate your environment and what is in the back-end mobile server repository. The following sections describe the validation that occurs:

- Section 2.3.1.1.1, "Validate the Environment for the Mobile Server"
- Section 2.3.1.1.2, "Validate Integrity of Mobile Server Tables and Data"
- Section 2.3.1.1.3, "Validate the Structure and Contents of the Repository"
- Section 2.3.1.1.4, "Validate Application Databases"

2.3.1.1.1 Validate the Environment for the Mobile Server The MSRDT tool displays the system configuration environment for the host where the mobile server resides, as follows:

- Consolidator version
- Back-end Oracle database version
- Definition of the Java library path
- Definition of the Java `CLASSPATH`
- Operating system architecture, type and version
- Java VM vendor, version, name, home, and info (mode)
- File encoding used
- Path separator and file separator used
- Country, time zone, and user language
- Lists the mobile server administrators
- Lists the contents/current configuration for the `mobile.ora` file

2.3.1.1.2 Validate Integrity of Mobile Server Tables and Data When the mobile server and the repository are installed, certain tables, constraints, objects, and so on cannot be modified. The MSRDT diagnostic tool checks that these requirements are consistent as with what was installed. This includes the following:

- Required tables exist
- All columns within required tables exist
- Required table attributes exist
- Required constraints exist or have not been modified
- Required sequences exist

- Extra tables have not been added to mobile server schemas

- Application files integrity check: If you have published an application to the server, then check if the folder exists and is not empty. Also, if sharing a repository among multiple mobile servers, ensures that the application is published on this mobile server, where this tool is executed.

2.3.1.1.3 Validate the Structure and Contents of the Repository If you are experiencing trouble with your repository, execute the MSRDT tool to determine if any of the following have occurred:

- The mapping between primary keys for the map table and corresponding base table must be consistent.

- Every map table must have a corresponding base table.

- Identify the user with the most records in the map table.

- Invalid indexes

- Consistency of the In Queue schema with the (CPV) view schema.

- Check if the Error Queue and the In Queue have the same records; that is, both queues should not contain records with the same primary key, which consists of the CLIENTID, TRANID$$, and SEQNO$$.

- The records in the In Queue master table have corresponding records in the In Queue detail table.

- Any DML lock is held by any table in the C$ALL_LOGGED_TABLES.

- Invalid triggers.

- Every publication item is included in C$PUBITEM_PROPS.

- Validate that all of the users subscribed to a publication also have all of the corresponding sequences assigned to the same publication.

- Invalid sequence values.

- Validate that every record in C$ALL_SEQUENCES corresponding C$WS_<ID> window sequence object, where <ID> is the values of ID column in C$ALL_SEQUENCES.

- Validate that every window sequence has the same id as a record in the C$ALL_SEQUENCES.

- Look for any orphaned objects in the repository.

- Verify that the mobile server repository owner is granted with sufficient privileges.

- Validate the MGP properties: If MGP_SUSPEND(apply suspend) is false, then MGP_RUN must be true.

- Reports on the most recent jobs and their status.

- Check for any automatic synchronization notification sent to a non-existent user in C$DATA_NOTIFICATION.

- Check for any automatic synchronization notification that has not been sent by the Device Manager for more than two days.

2.3.1.1.4 Validate Application Databases You can specify that the application uses a database other than the one in which the mobile server repository resides for all application schema data. If any application databases are defined, then the MSRDT tool checks for the following in all of the defined application databases:

- Check for consistency for the primary keys for both the map table and corresponding base table

- Check for any records that are in the In Queue master table, but not in the corresponding In-Queue detail table

- Check for any tables in the `C$ALL_LOGGED_TABLES` that hold any DML locks

- Check for duplicate records in the Error Queue and the In Queue

- Identify the user with the most records in the map table

- Verify that the In-Queue schema is consistent with the CPV view schema

- Check for any invalid triggers and recompile any invalid triggers

- Check for any invalid application database connections

- Check for any application databases that are registered more than once. This shows up as duplicate records in the `C$DB_INST` table.

- If the publication is dropped from the main database, then the base tables for the publication must also be dropped in the application database. The MSRDT tool checks if the publication was dropped in both the main database and the application database. If the publication tables still exist in the application database, then the MSRDT tool removes the publication entries in `C$ALL_PK_HINTS` and `C$EQ` in the application database.

- Check for any records in the `C$IN_MESSAGES` and `C$INQ` tables in the application database for any dropped clients in the main database. In this case, the MSRDT tool reports the records and moves the `C$IN_MESSAGES` records to the `C$EQ` table, and purges the `C$IN_MESSAGES` table.

2.3.1.2 Execute the Repository Diagnostics Tool

You can use the Mobile Server Repository Diagnostics Tool (`msrdt`) to validate the repository and provide error reporting. It also performs some error recovery.

The following is the usage and syntax for the `msrdt` tool:

```
msrdt -v <username>/<password>@<jdbc_url>
```

Where:

- `<username>/<password>`: The mobile server repository administrator user name and password.

- `<jdbc_url>`: You can specify the JDBC URL of a single Oracle database or an Oracle RAC database, as follows:

  - The URL for a single Oracle database has the following structure: `<host>:<port>:<SID>`

  - The JDBC URL for an Oracle RAC database can have more than one address in it for multiple Oracle databases in the cluster and follows this URL structure:

    ```
    jdbc:oracle:thin:@(DESCRIPTION=
     (ADDRESS_LIST=
        (ADDRESS=(PROTOCOL=TCP)(HOST=PRIMARY_NODE_HOSTNAME)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=SECONDARY_NODE_HOSTNAME)(PORT=1521))
     )
     (CONNECT_DATA=(SERVICE_NAME=DATABASE_SERVICENAME)))
    ```

> **Note:** if you supply a Oracle RAC URL as the JDBC URL, then
> enclose it within two double-quotes as the operating system treats the
> equal sign (=) as a delimiter, which truncates the Oracle RAC URL and
> throws the `syntax error: unexpected token '('.` error

2.3.2 Inspecting Files in the Mobile Repository With the WSH Tool

You can use the mobile server shell utility (wsh) to inspect and modify the mobile
server repository interactively. For full details, see Appendix B.2, "Running a Script
File with the WSH Tool" in the *Oracle Database Mobile Server Administration and
Deployment Guide*.

2.3.3 Modifying IP Address of Machine Where Mobile Repository Exists

During the installation, the machine name or IP address is provided by the user where
the repository is created. If the IP address of the machine changes, then perform one of
the two options:

- If the user provided the machine name; then even after the IP change, the machine
  name will still work.

- If user provided the IP address—instead of machine name—then after changing
  the IP address of the repository machine, the user must change the `ADMIN_JDBC_
  URL` and `THIN_JDBC_URL` parameters in the `mobile.ora` file on the mobile
  server.

2.4 Troubleshooting JVM Errors

This section focuses on how to debug the following Java error:

- Section 2.4.1, "Troubleshooting An Out of Memory Error"

2.4.1 Troubleshooting An Out of Memory Error

When you are experiencing the `OutOfMemory` error, then you should have an
understanding of JVM memory architecture when tuning mobile server performance.

The following may cause an `OutOfMemory` error:

- A memory leak in the mobile server.

- Not enough physical memory to handle your application.

- In-appropriate allocation of the three memory areas that used by the JVM. See
  Section 2.4.1.1, "JVM Memory Settings" for a full description.

- Memory being held by the mobile server. See Section 2.4.1.2, "Why is Memory Not
  Released?" for more information.

- Understanding how threads are consuming your memory. See Section 2.4.1.3,
  "Thread Memory Consumption and Concurrency" for full details.

2.4.1.1 JVM Memory Settings

JVMs may have different implementations of memory management and garbage
collection schemes. But at a higher level, they all arrange the memory in the following
three areas:

- Section 2.4.1.1.1, "Java Heap"—The Java heap is where the Java objects live. It is normally the largest of the three.

- Section 2.4.1.1.2, "Permanent Generation"—The memory where the classes are loaded.

- Section 2.4.1.1.3, "Native Space"—The memory used by native code, which includes JVM native code and application JNI calls.

- Section 2.4.1.1.4, "Setting Java Options for Java Memory"—Setting options depending on your environment.

This section describes how to modify the allocation of memory to the JVM memory areas.

> **Note:** Memory should be allocated properly for the three areas. Otherwise, different kinds of `OutOfMemory` error may surface.

2.4.1.1.1 Java Heap The Java heap is where Java objects live. It consists of both the young and tenured generations. The amount of Java heap memory that the JVM starts with is designated by the initial heap size option (`-Xms`) and the maximum heap size option (`-Xmx`).

If you see from the stack trace that a Java method throws an `OutOfMemory` error, then you have exhausted your Java heap space.

For example:

```
java.lang.OutOfMemoryError: Java heap space
```

The default settings for the Java heap for a Oracle UNIX JVM is as follows: `Xmx:64M Xms:4M`. However, the default for the mobile server—if you start up the mobile server with the `runmobileserver.bat` executable—is set to `Xmx:256M Xms:512M`.

The size of the space reserved can be specified with the `-Xmx` option. The `-Xms` specifies the space that is immediately committed to the virtual machine. We recommend to allocate ¼ to ½ of the available physical memory to Java Heap. If you set the maximum Java Heap size to be large—such as, 512M—and you still receive this error, then there may be a leak in the Java code.

The amounts specified should be based on the available resources. At the minimum, you should set both values to at least 256 MB. Of course, the amount of memory you allocate depends on what you have available.

> **Note:** Set the Java heap memory size before starting the mobile server with the `runmobileserver` executable.

2.4.1.1.2 Permanent Generation The permanent generation holds data needed by the JVM that describes objects which do not have an equivalence at the Java language level. For example, permanent generation is where the classes are loaded. It holds objects that describe classes and methods.

If a classloader method or a String intern method throws an `OutOfMemory` error in the stack trace, then you have run out of permanent generation space.

For example:

```
java.lang.OutOfMemoryError: PermGen space at
java.lang.ClassLoader.defineClass1(Native Method) at
```

```
java.lang.ClassLoader.defineClass(ClassLoader.java:620)
```

The default for the permanent generation for a Oracle UNIX JVM is 64MB. To set a new initial size for the Oracle JVM, use the `-XX:PermSize` option when starting the virtual machine. To set the maximum permanent generation size use the `-XX:MaxPermSize` option.

2.4.1.1.3 Native Space The native space is the memory used by native code, which includes JVM native code and application JNI calls. If a native method throws an `OutOfMemory` error or the JVM crashes with such an error, then you run out of native space.

For example:

```
java.lang.OutOfMemoryError: requested 14892 bytes. Out of swap space?
java.lang.OutOfMemoryError: unable to create new native thread
```

The native space is the (Available physical memory) – (Java heap + Permanent generation). There is no way to set the native space, except to decrease the Java heap or permanent space. If you allocate too much memory for the Java heap, then the native code is left with not enough memory and may run out. If you have to increase the native memory, then decrease the `-Xmx` parameter to a reasonable value to leave enough memory for the native space. If you still get this error, the native code may have a memory leak.

2.4.1.1.4 Setting Java Options for Java Memory Set the Java options when you start the Java servlet container. The following example sets the initial Java heap to 256M, the maximum Java heap to 512M, and the permanent generation memory to 64M:

```
-Xms=256m -Xmx=512m -XX:MaxPermSize=64m
```

For the mobile server, all modifications for the Java options must be specified on the command-line or in the `runmobileserver.bat` file.

By default, the MGP executes as a job in the Job Scheduler in the mobile server. Thus, the MGP and other mobile server components, such as the Sync Server, share the same memory space. This provides efficiency and manageability; however, if the MGP has a memory leak, then the mobile server is affected. In this case, perform the following:

1. Disable the MGP job.

2. Restart the mobile server.

3. Restart the MGP in a separate JVM with either the `mgp.bat` or, if using UNIX, the `mgp` shell script. This JVM is restarted periodically and may hide the memory leak issue.

> **Note:** Set the Java memory options for the MGP in the `mgp.bat` file.

With a larger Java heap size, the garbage collector collects less often and consumes less CPU time. Therefore, a larger heap size is desired for better performance. For the mobile server, most of the code is Java code; for the JDBC connection and Java mSync client, most of the code is in native code. So, setting the Java heap size larger, helps the efficiency and performance of the mobile server; however, if it is set too high, the JDBC connection and mSync client may have memory issues.

2.4.1.2 Why is Memory Not Released?

You may expect the mobile server to release the free memory back to operating system after it has finished its work. However, the mobile server holds a large amount of memory even when it is idle. This may not be an indication of memory leak; instead, it may be for one or more of the following reasons:

■   If you set the `-Xms` option to a large number—such as, 1024 MB—then you should expect the mobile server process to use at least 1024 MB until the process is killed.

■   For performance reasons, the mobile server caches metadata in Java heap memory.

■   The garbage collector may not collect the objects right way when they are no longer referenced. In addition, the garbage collector keeps a large amount of free memory in the Java heap for future allocations, instead of returning them to the operating system. You can use Java options to adjust the free memory size; instead, view the mobile server total runtime Java heap size and free heap size in the Mobile Manager at Mobile Manager->Data Synchronization->Host.

2.4.1.3 Thread Memory Consumption and Concurrency

The Java heap and permanent generation together are called managed heap, since the garbage collector manages them. The native space can be divided into native heap and thread stack space. Each thread consumes memory, as follows:

■   Each thread created consumes about 1MB stack space, although it is JVM dependent. Take this memory into consideration if you execute multiple threads. For example, on a 32-bit x86 system, the (managed heap + native heap + thread stack size * number of threads) could not exceed 2 GB. On any system, ensure that the total JVM memory is less than the available physical memory size.

■   Each thread allocates additional Java and native heap memory as it executes.

■   There is an overhead associated with multi-threading. Therefore, be careful when executing too many concurrent threads. If concurrency is set to larger than 20, then you are more likely decreasing the mobile server throughput—instead of increasing it.

You can configure for concurrency with the parameters described in Section 1.2.4, "Configuration Parameters in the MOBILE.ORA that Affect Synchronization Performance".

2.5 Troubleshooting Security

The following section describes how to troubleshoot security issues:

■   Section 2.5.1, "SSL Certificate Rejection for Client Authentication"

2.5.1 SSL Certificate Rejection for Client Authentication

If you are using a reverse proxy and have configured SSL between the client and the reverse proxy, you may receive the following error:

```
A certificate is required to complete client authentication.
```

For all clients you can only use SSL authentication with a signed certificate. If you use a self-signed certificate, you must turn off SSL authentication by adding the following to the `NETWORK` section in the client `devmgr.ini` file:

```
DISABLE_SSL_CHECK=YES
```

This parameter tells the reverse proxy firewall to use SSL encryption for the communication from the client, but not to perform SSL authentication.

3

Tracing and Logging

You can enable tracing for the mobile server. In addition, you can view the log files from the underlying application server. For the mobile server, there are two main sections for tracing: the general tracing for mobile server components and specific tracing for data synchronization components. How to enable tracing for each part of the mobile server is described in the following sections:

- Section 3.1, "General Tracing for the Mobile Server"
- Section 3.2, "Data Synchronization Tracing"

3.1 General Tracing for the Mobile Server

To set general tracing for the mobile server, perform the following steps.

1. From the mobile server page, select **Administration**.

2. Select **Trace Setting**. This brings up the Trace Settings page, as shown in Figure 3–1, where you can choose to generate trace output, specify the trace output destination to the local console, file, or remote console (viewed by `WSH`).. The Trace Settings page provides system filters to generate trace output to the required system level.

3. Configure the type of tracing you want and click **Apply**.

Figure 3–1 General Trace Settings for Mobile Server

Trace Settings

Trace Properties

Trace Output	NO ▾

Destination

◉ Console
○ File
　Trace Base File name　　C:\Oracle\Middleware\wls
　Trace File Size (in mb)　10
　Trace File pool Size　　5
○ Remote
　Trace monitor host
　Trace monitor port　　0

System Filter

☐ HTTP Request
☐ SQL Statements
☐ Java Methods

Table 3–1 Trace Settings Page Description

Field	Description
Trace Output	To generate trace output, select **Yes**.
Console	You can print the messages to a console. If you are in an WebLogic environment, select File or Remote.
File	You can direct all messages to a local file. If you selected a file for trace output, then enter the name (including path), the maximum size of the file in MB, and the number of files allowed (pool size). For example, if you set the pool size to 10, then when a trace file hits the maximum size in MB, then a new file is opened and the trace output is written to the new file. This continues until all 10 files of the maximum size exist. At this point, the first file is deleted and a new file is started to contain the trace output. This enables you to manage the amount of disk space that the trace files can use.
	To create a trace file for every user, select **Yes** for the Create Trace File for Every User box.
System Filter	▪ HTTP Request—To generate HTTP output information as trace output, select this option. This includes general system information.
	▪ SQL Statements—To generate SQL queries as trace output, select this option.
	▪ Java Methods—To generate all `system.out` output from the mobile server and Data Synchronization Java methods, select this option.
	Note: The mobile server automatically filters exceptions and errors as trace output at the Mandatory level.

3.2 Data Synchronization Tracing

The administrator can turn on tracing for components involved in the synchronization phase, including MGP functions.

1. From either the home page or the Administration page for the mobile server, select **Data Synchronization** in the Components section, as shown in Figure 3–2.

Figure 3–2   Mobile Server Job Scheduler and Data Synchronization Components

2. Select **Administration** off of the Data Synchronization home page.

3. Select **Trace Settings**, which displays all five components for which you can enable tracing, as shown in Figure 3–3. For a description of each component, see Section 3.2.1, "Description of the Five Data Synchronization Components".

Figure 3–3   The Trace Components for the Data Synchronization

4. Select the component for which you want to enable tracing, which brings up the trace configuration screen, as shown in Figure 3–4.

Figure 3–4   Data Synchronization Component Trace Configuration

■ In the Filter section, select the required **Level** and **Type**. To specify a trace filter for users, enter comma separated user names in the **Users** field.

Table 3–2   Data Synchronization Component Trace Level and Type

Filter	Description
trace level, where each level includes the previous levels as well.	`OFF`: no tracing enabled.
	`MANDATORY`: Mandatory messages only, such as program exceptions.
	`WARNING`: Warning messages.
	`NORMAL`: Normal messages of which the user must be informed.
	`INFO`: Informational messages, such as synchronization timing, MGP apply, MGP compose, and MGP status.
	`CONFIG`: Configuration messages, such as JDBC driver version.
	`FINEST`: Developer level of tracing.
	`ALL`: Logs messages for all trace levels.
trace type	`SQL`: SQL-related messages only, such as SQL statements.
	`TIMING`: Timing data only. **Note**: This option is trace level sensitive. For MGP Cycle time and Synchronization time, use the Trace Level INFO option with the TIMING option on the MGP and SYNC components respectively.
	`DATA`: Data only.
	`RESUME`: Logs messages with Reliable Transport.
	`FUNCTION`: Displays the program flow by logging methods such as Entry, Exit or Invoke. For Long methods, this option logs the method entry or exit; which is a simple invoke log.
	`GENERAL`: Logs messages that do not belong to any of the above listed trace types. **Note**: This type is trace level sensitive.
	`ALL`: This option generates logs of all trace types.

> **Note:**   You can set these parameters within the `mobile.ora` file in the `CONSOLIDATOR` section. See Section A.1.5, "[CONSOLIDATOR]" in the *Oracle Database Mobile Server Administration and Deployment Guide* for more details on these parameters.

- In the Destination section, select **Local Console** to receive the trace file to the same console as the General tracing is using. If the console is not open, then these messages are sent to the same place that the General tracing is directed. See what the Destination is configured to in Figure 3–1 to determine where these messages are directed.

  To send trace information to a file, select the **File** option. The file name is generated based upon the session id. You can configure the file size in MB and the files allowed (pool number). For example, if you set the pool size to 10, then when a trace file hits the maximum size in MB, then a new file is opened and the trace output is written to the new file. This continues until all 10 files of the maximum size exist. At this point, the first file is deleted and a new file is started to contain the trace output. This enables you to manage the amount of disk space that the trace files can use.

5. To implement the modified values, click **OK**. To retain existing values, click **Cancel**.

To view trace files, navigate to the Data Synchronization page. Select **Administration**. Select **Trace Files** and the Trace Files screen appears, as shown in Figure 3–5.

Figure 3–5   Viewing Data Synchronization Trace Files

- To view a trace file, select the trace file name or click the Select button next to the trace file name and click **View**.

  > **Note:**   When you view the trace file online, it truncates the file to 10,000 lines. To view the whole trace file, download the file and view it using any text editor.

- To download or delete a trace file, click the Select button next to the trace file name and click either **Download** or **Delete**.

- If there are too many files to view on a page, you can search by entering the name of the trace file in the Search field and clicking **Go**.

3.2.1 Description of the Five Data Synchronization Components

There are five components that you can turn on to describe what is happening in the synchronization process, as described in the following sections:

- Section 3.2.1.1, "MGP"

- Section 3.2.1.2, "MGPAPPLY"

- Section 3.2.1.3, "MGPCOMPOSE"

- Section 3.2.1.4, "SYNC"

- Section 3.2.1.5, "GLOBAL"

3.2.1.1 MGP

You can trace the MGP process. However, if an `MGP Cycle ID` is not yet available, then tracing is enabled by the configuration of the `GLOBAL` component. If the trace destination is to be written to a file, then all of the generated logs are recorded in a log file named `MGP_<cycle_id>.log`.

3.2.1.2 MGPAPPLY

This refers to the `APPLY` phase in the MGP process. However, between the beginning of the `APPLY` phase till the availability of the MGP Client ID, tracing is enabled by the configuration of the component MGP. If tracing is sent to a file, then all messages are written to a file named `MGPAPPLY_<client_id>_<cycle_id>.log`.

3.2.1.3 MGPCOMPOSE

This refers to the `COMPOSE` phase in the MGP process. Similar to the `MGPAPPLY` phase where the Client ID is not yet available, tracing is enabled by the configuration of the component MGP. If tracing is sent to a file, then all messages are written to a file named `MGPCOMPOSE_<client_id>_<cycle_id>.log`.

3.2.1.4 SYNC

This refers to the server-side synchronization process. When a `Sync session ID` is not yet available, tracing is enabled by the configuration of the `GLOBAL` component. If the trace destination is set to file, then the messages are written to a file named `SYNC_<cycle_id>.log`. When the Client ID becomes available, the file is renamed to `SYNC_<client_id>_<cycle_id>.log`.

3.2.1.5 GLOBAL

This component logs tracing messages that are not specific to any of the above listed components. This component also includes logs that are generated during the execution of the `ConsolidatorManager` APIs. If the trace destination is set to file, then the messages are written to a file named `GLOBAL_<file_number>.log`.

4

Backup and Recovery

Performing backup and recovery for Oracle Database Mobile Server is the same as what you would normally do for Oracle database applications. The following sections help you understand how to use the Oracle database backup and recovery methods for preserving your mobile server and mobile applications:

- Section 4.1, "How Does Oracle Database Mobile Server Store its Information?"
- Section 4.2, "Backing Up Oracle Database Mobile Server"
- Section 4.3, "Oracle Database Mobile Server Backup Coordination Between Client and Server"
- Section 4.4, "Oracle Database Mobile Server Recovery Issues"

4.1 How Does Oracle Database Mobile Server Store its Information?

Oracle Database Mobile Server uses the Oracle database to store information, as follows:

- The mobile server itself is installed and configured as a database application. Thus, the mobile server stores its metadata and client state information within a database schema.
- For each mobile application, the mobile server installs triggers and stores transaction data in a schema for that application.

4.2 Backing Up Oracle Database Mobile Server

Since all of the data needed for a backup and recovery strategy exists in the database, you should use the Oracle database backup and recovery strategies discussed in the following books:

- *Oracle Backup Installation Guide*
- *Oracle Database Recovery Manager Quick Start Guide*
- *Oracle Database Backup and Recovery Basics*
- *Oracle Backup Administrator's Guide*
- *Oracle Database Backup and Recovery Advanced User's Guide*

> **Note:** In the past, we recommended that you use export/import to perform a backup. This is not a recommended option anymore. Use the normal online Oracle database backup procedure.

However, the following sections describe what to keep in mind when coming up with a backup and recovery strategy for your Oracle Database Mobile Server environment: Section 4.3, "Oracle Database Mobile Server Backup Coordination Between Client and Server" and Section 4.4, "Oracle Database Mobile Server Recovery Issues".

4.3 Oracle Database Mobile Server Backup Coordination Between Client and Server

When a client and a server synchronize with each other, the mobile server assigns the same "magic" number to both sides to indicate that the data is in-sync. If this number is different on both sides, the mobile server knows that the data is out of sync and is in an error condition. The following example details how this could effect your attempts for a clean recovery.

When you perform a backup, you may lose client data unless you plan accordingly. Figure 4–1 demonstrates the following scenario:

1. The client inserts record 1 and synchronizes the data to the server. The mobile server assigns the same magic number to both the client and the server to denote that they are in sync. In this example, the magic number on both the client and the server is 10.

2. The client inserts record 2. No synchronization is performed.

3. The backup is performed. Client record 1 is saved to the backup. The latest magic number on both the client and the server is 10.

4. The client inserts record 3 and synchronizes the data to the server. The mobile server assigns the same magic number to both the client and the server to denote that they are in sync. In this example, the magic number on both the client and the server is 20.

5. The client inserts record 4. No synchronization is performed.

6. A failure occurs and the last backup is used to recover the mobile server, the mobile applications and the application data. In this scenario, only record 1 is in the backup, so it will exist in the restored database.

7. The client synchronizes. Records 2 and 3 can be lost, because they are not in the backup. The msync client does not send them to the server, since they were already sent in step 4. However, the msync client does send record 4 to the server, since it is a new record that has never been synchronized with the database. After the synchronization, record 4 is stored in the error queue, not in the application tables.

Figure 4–1   Lost Data With Backup and Recovery Strategy

client inserts record 1, syncs, magic number = 10

client inserts record 2

perform backup

client inserts record 3, syncs, magic number = 20

client inserts record 4

perform recovery

client syncs

TOTAL ELAPSED TIME
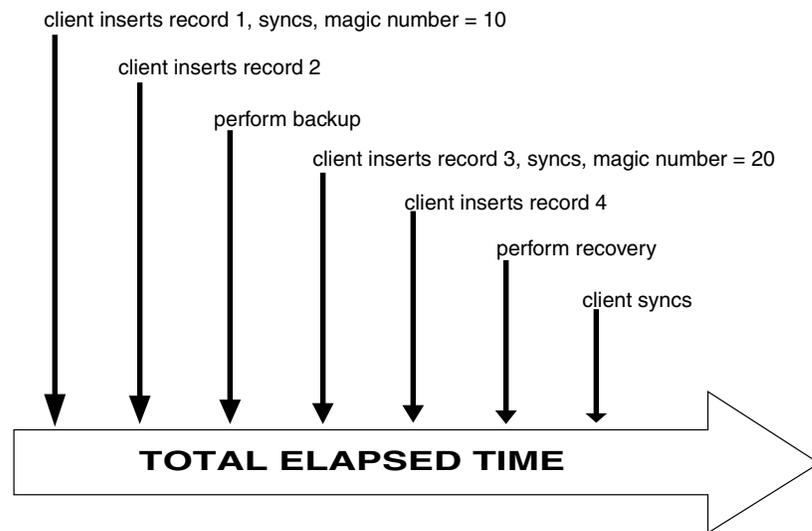
The mobile server checks the magic numbers on both the client and the server. It verifies the state of the data on the client to determine what action to take. When the client performs the next synchronization, if the magic numbers are not the same, then the following occurs:

1. The client checks if there are any new records—whether newly inserted, modified, or deleted—on the client. If so, then these records are sent to the server, which saves these records in the error queue.

2. A full refresh of all of the subscribed data is sent to the client.

In our example, if you did nothing, the client would send record 4 to the server, which would end up in the error queue, and records 2 and 3 would be lost. To save records 2 and 3, do the following:.

1. On the server, retrieve and restore the last backup.

2. On the mobile client that is out of sync, update any record that has been modified since the last synchronization. In our example, you would do any sort of update that makes the record seem to contain new information in records 2 and 3. For example, you could update the VARCHAR field with the same content.

3. Initiate a synchronization on the client. The Oracle Database Mobile Server software detects that the client database is out of sync and that some of the records have been modified. Thus, the following occurs:

   a. The modified records are updated in the restored database on the server and saved in the error queue.

   b. The server pushes a full refresh down to the client.

4. In order for you to reapply the modified records to the applications table, you must first modify the DML operation from Error to Update. The DBA must modify the record in the error queue for the base table, named CEQ$<base_table_name>, changing the DML operation from Error (E) to Update (U) or Insert (I).

5. Once updated to Update, re-execute the command. Navigate to the Error Queue screen in the Mobile Manager. Click on the modified record. Click Execute.

> **Note:** For more information on the error queue and how to reapply the records, see Section 2.10.1 "Resolving Errors and Conflicts on the Mobile Server Using the Error Queue" in the *Oracle Database Mobile Server Developer's Guide*.

6. The next time that the MGP runs, the update is applied to the application table.

Thus, all information contained within records 2 and 3 will be restored from the device.

4.4 Oracle Database Mobile Server Recovery Issues

When you perform a recovery, the state of both the mobile server and the mobile application schemas must be in-sync. If they are out-of-sync, severe problems may occur. Therefore, when you perform a backup and restoration for the mobile server and the mobile application schemas, each must be recovered to the same point in time. Use the Oracle database Point-in-Time Recovery strategy to ensure that both the mobile server schema and mobile application schemas are recovered to the same point in time.

The mobile application schemas usually reside on the same Oracle database as the mobile server. However, if you have used a database link to store the mobile application schemas on a separate Oracle database, then you must use a backup and restore strategy for distributed database systems.

Index

repository,   2-4
truncate command,   1-14, 1-15

U

URL
   using Oracle RAC,   1-20
user
   large amounts of data,   1-13
using Oracle RAC,   1-20

V

validation
   repository,   2-4
views
   non-mergable,   1-10