Macrotone Consulting Ltd.

Oracle Transparent Gateway to SQL*Server

Author:	**G S Chapman**
Date:	**9th January 2006**
Version:	**2.0**
Location of Document:	

DOCUMENT HISTORY

Version	Date	Changed By:	Remarks
1.0	18/08/05	G S Chapman	Initial version
1.0A	25/08/05	G S Chapman	Update with basics of gateway package
1.0B	3/11/05	G S Chapman	Update with new details.
1.0C	7/11/05	G S Chapman	More details on views upon tables with NVARCHAR2 fields greater than 1000 characters.
1.0D	14/11/05	G S Chapman	Update with gateway padding information.
1.0E	18/11/05	G S Chapman	Add create_views_3 procedure details
1.0F	6/12/05	G S Chapman	Changes to reflect new data drop.
2.0	9/01/06	G S Chapman	Changes for UAT environment.

DOCUMENT DISTRIBUTION

Copy No	Name	Role	Organisation

DOCUMENT REFERENCES

Document Name	Originator	Part Number	Version	Date
Oracle Transparent Gateway for Microsoft SQL Server Administrator's Guide 10g Release 1 (10.1) for Microsoft Windows	Oracle Corp	B10544_01	10.1	
Oracle Transparent Gateway for Microsoft SQL Server Administrator's Guide Release 2 (9.2) for Microsoft Windows NT	Oracle Corp	A96552_01	9.2	
Oracle9i Messaging Gateway Supplement Release 1 (9.0.1)	Oracle Corp	A90837_01	9.0.1	
Oracle Transparent Gateway for Microsoft SQL Server Administrator's Guide 10g Release 2 (10.2) for Microsoft Windows (32-bit)	Oracle Corp	B14270-01	10.2	June 2005
Oracle Database Heterogeneous Connectivity Administrator's Guide 10g Release 2 (10.2)	Oracle Corp	B14232-01	10.2	June 2005

TABLE OF CONTENTS

1 Introduction 1
2 Installation of SQL*Server Transparent gateway 3
2.1 Release Version specifics 3
2.1.1 Oracle V9.2.0 3
2.1.2 Oracle TG4MSQL V10.1.0.2 3
2.1.3 Oracle TG4MSQL V10.2.1 3
2.2 Oracle Database dictionary table requirements 4
2.3 TNSNAMES entries 4
2.3.1 Common Errors 4
2.4 Listener.ora 4
2.4.1 Common Errors 5
2.5 Gateway Configuration 5
2.5.1 SQL*Server Oracle connection account 6
2.5.2 Gateway transaction modes 6
2.5.3 Configuring for Two-Phase Commit 6
2.5.3.1 Task 1: Create a Recovery Account and Password 6
2.5.3.2 Task 2: Create the Transaction Log Table 7
2.5.4 Specifying an Owner 7
2.6 Testing connectivity 7
2.6.1 Common Errors 8
2.7 Common Errors and Solutions 8
2.8 Errors encountered with UAT setup 9
2.9 Oracle Connectivity 10
3 Customer setup 13
3.1 Oracle databases 13
3.1.1 PODS9 13
3.1.2 PODS 13
3.2 UTILS schema 13
3.3 Views 13
3.3.1 View required in SQL*Server 14
3.4 Known Issues 16
3.4.1 NVARCHAR type longer than 1000 characters 16
3.4.2 NVARCHAR padding 16
4 Warehouse Requirements 17
4.1 Long table names and long column names. 17
4.2 SQL*Server data type conversion 17
4.2.1 Create Table As Select (CTAS) Syntax 18
4.3 Inline View Support 20
4.4 Data Type Conversion 20
4.4.1 Date variables 21
4.4.2 NCHAR variables 21
4.4.3 NVARCHAR padding 21
4.4.4 Microsoft SQL Server IMAGE, TEXT and NTEXT Data Types 21
5 Workarounds 22
5.1 Simple select using DBMS_HS_PASSTHROUGH 22
5.2 Select using cursor for loop 23

5.3 Creating a SQL*Server view 23
6 GATEWAY Package 25
6.1 Package error codes 28
7 Tuning Considerations 29

TABLE OF FIGURES

Figure 1 - Service Gateway 1
Figure 2 - Gateway Process Flow 1
Figure 3 - View Creation Script 27

TABLES

Table 1- Data type conversions 21
Table 2- GATEWAY package procedures 26

Appendices

A. Gateway Specifications
A.1 Supported Views and Tables
A.2 Data Dictionary Mapping
A.3 Useful SQL Server view

PURPOSE OF DOCUMENT

This document describes the installation, configuration and usage of the Oracle Transparent Gateway to SQL*Server installed at a customer site.

1 Introduction

The requirement is to describe the use of the Oracle Transparent Gateway for Microsoft SQL*Server (TG4MSQL) at a customer site to access data held with certain SQL*Server production databases from the data warehouse databases.

There are three areas of operation for effective interoperation: SQL translation, data dictionary translation and data type translation. To meet these requirements Oracle provides Generic Connectivity and Transparent Gateways. The combination is the Heterogeneous Services (HS) component, integrated in the Oracle server and agent, which provides information for and connectivity to non-Oracle systems.

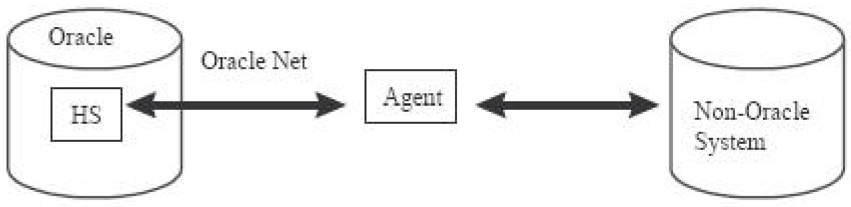

Figure 1 - Service Gateway

In addition there is the provision of Pass-through SQL, which enables the use of native SQL against a non-Oracle database system. This flexibility enables the execution of functions or procedures on non-Oracle systems that are not supported by the Generic Connectivity or the Transparent Gateway. The Pass-through SQL supports both result sets and bind variables, and can also be used to perform DDL on the non-Oracle system.

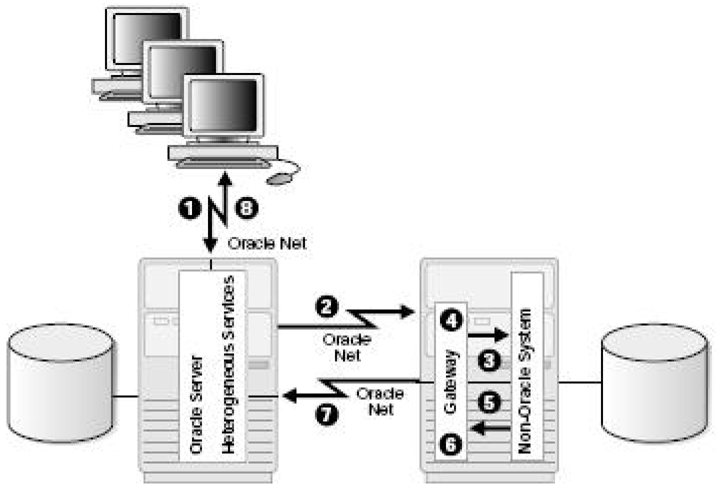

Figure 2 - Gateway Process Flow

1. The client application sends a query over Oracle Net to the Oracle database server.
2. Heterogeneous Services and the gateway converts the Oracle SQL statement into a SQL statement understood by the non-Oracle database system.
3. The Oracle database server sends the query over to the gateway using Oracle Net.
4. For the first transaction in a session, the gateway logs into non-Oracle database system using a username and password that is valid in the non-Oracle system.

5. The gateway retrieves data using non-Oracle database system SQL statements.
6. The gateway converts retrieved data into a format compatible with the Oracle database server.
7. The gateway returns query results to the Oracle database server, again using Oracle Net Services.
8. The Oracle database server passes the query results to the client application using Oracle Net. The database link remains open until the gateway session is finished or the database link is explicitly closed.

2 Installation of SQL*Server Transparent gateway

The following describes the setup and experiences in getting the SQL*server transparent gateway from Oracle to work. The initial version used for the setup was TG4MSQL (Transparent Gateway for MS SQL Server) for Windows based platform, release 10.1.0.4. Initially connections were made from an Oracle 9.2.0.6 and 10.1.0.4 server. Later connections were to Oracle 9.2.0.7 and 10.2.0.1 databases.

The configuration was later changed to use release 2 of the 10g gateway (10.2.0.1).

NOTE: Problems were experienced using the base release of 10g (10.1.0.1) for the gateway and the patch to version 10.1.0.4 was required to avoid the problems. Connections from a 9.2.0.5 database experienced problems assessing system tables.

2.1 Release Version specifics

2.1.1 Oracle V9.2.0

For V9.2.0.1.2 (9iR2 and above) gateways, the access method uses ODBC. So it is necessary to have the MS SQL Server ODBC driver installed on the gateway machine. If it is not available, download the latest MDAC Microsoft Data Access Components) from the Microsoft Web page and install it. This package contains a SQL Server ODBC driver. This was not necessary with the customer setup.

2.1.2 Oracle TG4MSQL V10.1.0.2

With the OUI (Oracle Universal Installer) install TG4MSQL. This product is part of the server installation CDs. (Start OUI, choose a separate. Oracle Home, select the Oracle database for installation. Then choose custom install. A product list pops up where you have to scroll down to ORACLE TRANSPARENT GATEWAY; click on the '+' in front and a product list opens. Choose the SQL*Server gateway to install)

Within the standard Oracle installed directories (within the Oracle Home default c:\oracle\product\10.1.0\) a directory called TG4MSQL will be created and a file called TG4MSQL.EXE is placed within the bin directory of the Oracle Home.

If not already done, upgrade the base release by applying the 10.1.0.4 patch set to the same Oracle Home directory that the gateway was installed within.

2.1.3 Oracle TG4MSQL V10.2.1

As with the earlier release, with the OUI (Oracle Universal Installer) install TG4MSQL. This product is part of the server installation CDs. (Start OUI, choose a separate. Oracle Home, select the Oracle database for installation. Then choose custom install. A product list pops up where you have to scroll down to ORACLE TRANSPARENT GATEWAY; click on the '+' in front and a product list opens. Choose the SQL*Server gateway to install)

Within the standard Oracle installed directories (within the Oracle Home default c:\oracle\product\10.2.0\tg_1), a directory called TG4MSQL will be created and a file called TG4MSQL.EXE is placed within the bin directory of the Oracle Home.

The release 2 installation prompts for the name of the SQL*Server instance to connect to and create the entries in the listener.ora file. Unfortunately there was still a need to edit the file and add the specifics for tg4msql.exe file to be executed when the listener is contacted. Once this change was made the listener performed without any further problems.

At the time of writing there are no patches released for this current version (10.2.0.1).

2.2 Oracle Database dictionary table requirements

TG4MSQL needs data dictionary tables in the Oracle database.

To check their existence, run a query on i.e. SYS.HS_FDS_CLASS.

If it fails, run the caths.sql script located in ORACLE_HOME\RDBMS\ADMIN\ as user sys or internal. In the case of the customer's installations this step was not necessary.

2.3 TNSNAMES entries

This file is located in ORACLE_HOME\NETWORK\ADMIN.

An entry such as the following is required:

```
tg4msql.machine3.customer.uk=
          (DESCRIPTION=
            (ADDRESS=(PROTOCOL=tcp)
              (HOST=rmachine3.customer.uk)
              (PORT=1521)
            )
            (CONNECT_DATA=(SID=tg4msql))
            (HS=OK)
          )
```

Entries of the above form will be required within every tnsnames.ora file upon servers that are required to connect to the SQL*Server gateway. In the current installation there are 2 SQL*Server databases that are being used. The two SIT instances were/are installed upon VMWare machines known as MACHINE3 and MACHINE4. The UAT instance is/was installed upon server 10.117.241.202 (MACHINE01).

2.3.1 Common Errors

Make sure, that there are 2 closing brackets after the SID; the HS keyword is outside of the Connect Data block.

Further ensure that only the TNS Alias is at the first position of the line; all other lines must start at least with one SPACE (blank); otherwise it is identified as an alias and the configuration is incorrect.

2.4 Listener.ora

This file is located in the ORACLE_HOME\NETWORK\ADMIN directory as well.

Add the entry for the gateway SID to the SID_List of the listener.ora and restart the listener afterwards. (After the restart a service handler for tg4msql should exist). The example listener.ora file is listed below for the 10G release 2 version installed..

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\product\10.2.0\tg_1)
      (PROGRAM = extproc)
    )
    (SID_DESC =
      (SID_NAME = tg4msql)
      (ORACLE_HOME = C:\oracle\product\10.2.0\tg_1)
      (PROGRAM = C:\oracle\product\10.2.0\tg_1\bin\tg4msql)
    )
  )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)
        (HOST = MACHINE3.customer.uk)(PORT = 1521))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
      )
    )
  )
```

Lines 8 to 12 were required to be added to the listener for the 10G release 2 installations.

If the listener service does not exist run the net configuration agent and define the listener service. Stop and restart the listener service and ensure that the gateway service exists.

2.4.1 Common Errors

After configuring the listener, restart it from the command line

```
(lsnrctl -> stop, start)
```

Then check the service summary. It has to contain a service handler for the TG4MSQL SID.

See section 2.8 for issues that may be encountered when trying to restart the listener.

2.5 Gateway Configuration

Adjust the configuration file of the gateway. It is located in ORACLE_HOME\TG4MSQL\ADMIN. The name depends on the SID used for the gateway. Multiple SIDs, for multiple connections to SQL*Server instances will require multiple initialisation files, one for each instance.

In our testing the listener SID used is tg4msql and so the configuration file must be named inittg4msql.ora. The 10G release 1 version of the file is shown below.

```
#
# HS init parameters
#
HS_DB_NAME=TG4MSQL
HS_DB_DOMAIN=MACHINE1.UK.COMPANY.INTRANET
HS_FDS_CONNECT_INFO=MACHINE1\\PCS.Northwind
HS_FDS_TRACE_LEVEL=OFF
HS_FDS_RECOVERY_ACCOUNT=oracle_con
HS_FDS_RECOVERY_PWD=password
```

The HS_DB_NAME and the HS_DB_DOMAIN parameters can be specified to enable a match to be made with the expected SID for the calling database. Otherwise the service name HO.WORLD is returned which will be an issue if global_names is set to TRUE.

The 10G release 2 installation uses the account name of ‘ODS_link’ for its connection. The account password and the parameters HS_DB_NAME and HS_DB_DOMAIN are changed as appropriate.

2.5.1 SQL*Server Oracle connection account

An account is required within the SQL*Server database to enable the connection to be made. This account can be any desired name and password and as a minimum requires 'select' privileges on he tables to which access is required. Other permissions may be granted as required.

At the current time only read access is provided to the SQL*Server tables for the created account 'ODS_link'.

2.5.2 Gateway transaction modes

The gateway supports the following transaction capabilities:

- ▪COMMIT_CONFIRM
- ▪READ_ONLY
- ▪SINGLE_SITE
- ▪TWO_PHASE_COMMIT

By default, the gateway runs in COMMIT_CONFIRM transaction mode. When the Microsoft SQL Server database is updated by a transaction, the gateway becomes the commit point site. The Oracle database server commits the unit of work in the Microsoft SQL Server database after verifying that all Oracle databases in the transaction have successfully prepared the transaction. Only one gateway can participate in an Oracle two-phase commit transaction as the commit point site.

Note: The current gateway is only configured in COMMIT_CONFIRM transaction mode by the customer. This is by intent as there is no requirement to run procedures upon the source SQL*Server databases.

2.5.3 Configuring for Two-Phase Commit

The following has not been configured upon the customer gateways but is included in this document for completeness, in case there is a need to configure it later.

To enable the TWO_PHASE_COMMIT transaction mode, create a recovery account and password and create a log table. The log table, called by default **HS_TRANSACTION_LOG**, is where two-phase commit transactions are recorded.

2.5.3.1 Task 1: Create a Recovery Account and Password

For the gateway to recover distributed transactions, a recovery account and password must be set up in the Microsoft SQL Server database. By default, both the user name of the account and the password are `RECOVER`. The name of the account can be changed with the gateway initialization parameter HS_FDS_RECOVERY_ACCOUNT. The account password can be changed with the gateway initialization parameter HS_FDS_RECOVERY_PWD.

Note: Oracle Corporation recommends that you do not use the default value RECOVER for the user name and password. Moreover, storing plain-text as user name and password in the initialization file is not a good security policy. There is now a utility called tg4pwd, that should be used for encryption. Refer to Chapter 4, 'Encrypting Initialization parameters' in the Heterogeneous Connectivity Administration Guide for further details.

1. Set up a user account in the Microsoft SQL Server database. Both the user name and password must be a valid Microsoft SQL Server user name and password.

2. In the initialization parameter file, set the following gateway initialization parameters:

HS_FDS_RECOVERY_ACCOUNT to the user name of the Microsoft SQL Server user account you set up for recovery.

HS_FDS_RECOVERY_PWD to the password of the Microsoft SQL Server user account you set up for recovery.

For information about HS_FDS_RECOVERY_ACCOUNT and HS_FDS_RECOVERY_PWD, see the relevant Oracle documentation.

2.5.3.2 Task 2: Create the Transaction Log Table

When configuring the gateway for two-phase commit, a table must be created in the Microsoft SQL Server database for logging transactions. The gateway uses the transaction log table to check the status of failed transactions that were started at the Microsoft SQL Server database by the gateway and registered in the table.

Note: Updates to the transaction log table cannot be part of an Oracle distributed transaction.

Note: The information in the transaction log table is required by the recovery process and must not be altered. The table must be used, accessed, or updated only by the gateway.

The table, called HS_TRANSACTION_LOG, consists of two columns, GLOBAL_TRAN_ID, data type CHAR(64) NOT NULL and TRAN_COMMENT, data type CHAR(255).

Any desired name can be used for the log table, other than HS_TRANSACTION_LOG, by specifying the other name using the HS_FDS_TRANSACTION_LOG initialization parameter.

Create the transaction log table in the user account you created in 2.5.3.1 "Task 1: Create a Recovery Account and Password". Because the transaction log table is used to record the status of a gateway transaction, the table must reside at the database where the Microsoft SQL Server update takes place. Also, the transaction log table must be created under the owner of the recovery account.

Note: To utilize the transaction log table, users of the gateway must be granted privileges on the table.

To create a transaction log table use the `tg4msql_tx.sql` script, located in the directory `ORACLE_HOME\tg4msql\admin` where ORACLE_HOME is the directory under which the gateway is installed. Use isql (or sqlplus) to execute the script at the MS-DOS prompt, as follows:

```
Isql: -Urecovery_account -Precovery_account_password [-Sserver] -itg4msql_tx.sql
```

2.5.4 Specifying an Owner

Instead of using the default owner name for the Microsoft SQL Server tables as defined in Microsoft SQL Server, or explicitly specifying a different owner in the SQL statements, you can specify a default owner that is used whenever a name is not explicitly specified in the SQL statements.

To specify the owner, set the gateway initialization parameter HS_FDS_DEFAULT_OWNER in the initialization parameter file.

2.6 Testing connectivity

Testing the connectivity between Oracle database and the SQL Server:

create a database link within the Oracle database to the SQL Server:

create database link tg4msql connect to "oracle_con"

```
                identified by "<password of oracle_con>" using 'tg4msql';
        select * from all_catalog@tg4msql;
```

[Alternatively the link could be specified completely without reference to the tnsnames.ora entry as follows:

```
                create database link tg4msql connect to "oracle_con" identified by
                "<password of oracle_con>"
                using '(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=hostname)
                (PORT=1521))(CONNECT_DATA=(SID=tg4msql))(HS=OK))';
```

]

Example selecting a demo table of the MS Northwind database:

```
        select "TerritoryID"  from "EmployeeTerritories"@<link name>;
```

2.6.1 Common Errors

The usernames, passwords, table names, views, columns .are case sensitive.

For creating the database link, make sure you surround the username, password by double quotes and write them as they are defined in the MS SQL Server.

Another common error is that the TG4MSQL does not allow OS Authentication (Windows Authentication) for the SQL Server. Only SQL Server authentication with a username AND a password is supported. Make sure, that the SQL Server Security option is checked to allow both authentication methods.

Table names and column names have to be less than 30 characters long. Set up a view to get around these issues on the SQL*Server database.

The session does not display the requested information and complains that the database link points to another service name. This is a global_names setting. Set the global_names parameter for the session to false and retry the query.

2.7 Common Errors and Solutions

```
/*****************************************************************/
ORA-28509: unable to establish a connection to non-Oracle system
ORA-02063: preceding line from TG4MSQL
/*****************************************************************/
```

Cause:

This indicates a problem with the Oracle configuration files.

Action:

Make sure the HOST parameter in the tnsnames.ora file is correct.

Make sure the PORT number is correct.

Make sure the SID name is correct in both the TNSNAMES.ORA and LISTENER.ORA

```
/*****************************************************************/
```

```
ORA-28500: connection from ORACLE to a non-Oracle system returned this message:
[Transparent gateway for MS SQL Server] The environment variable
<HS_FDS_CONNECT_INFO> is not set.
ORA-02063: preceding 2 lines from TG4MSQL
/*****************************************************************/
```

Cause:

Incorrect parameter settings in the HS init.ora file.

Action:

Check HS_FDS_CONNECT_INFO in the TG4MSQL init.ora file.

It might be missing or TG4MSQL is not able to find the correct initialisation file.

Make sure the HS init.ora file exists in the ORACLE_HOME\tg4msql\admin directory and has the same name as the SID in the LISTENER.ORA.

Example: If SID=mssql in the listener.ora file, then the init.ora file would be named ORACLE_HOME\hs\admin\initmssql.ora

```
/*****************************************************************/
ORA-00942: table or view does not exist
[Transparent gateway for ODBC]DRV_OpenTable: [Microsoft][ODBC SQL Server
Driver][SQL Server]Invalid object name '%table%'. (SQL State: S0002; SQL Code:208)
ORA-02063: preceding 2 lines from TG4MSQL
/*****************************************************************/
```

Cause:

The init.ora file specifies the wrong MS SQL Server database.

A second cause might be that MS SQL Server tables are case sensitive and thus should be surrounded by double quotes.

2.8 Errors encountered with UAT setup

There were a number of problems encountered with the installation of the gateway on the Windows UAT environment. These are described below along with the solution implemented.

Windows machine name for UAT environment = MACHINE01.customer.uk

AIX machine hosting ODS database = warehouse2.customer.uk

1. The installation of the Oracle software upon MACHINE01 went smoothly, although at the time the host name was not known and the IP address was used in the inittg4msql.ora file. This is suspected to be one possible cause of a problem relating to connecting to the SQL*Server database. This configuration file was modified to use the hostname, although this may not be the real cause of the UAT connection problem, which is complicated by other issues described below.

2. The AIX hosts were unable to ping the Windows server, although it worked from devwarehouse1. No routes were available from rgdsdw2.
3. After establishing a database link from devwarehouse1 a select from a table upon the SQL*Server database would just hang, with no response from the gateway.
4. Adding an entry for the warehouse2 server into the UAT DNS environment did not produce any improvement.
5. A sqlplus session from the Windows server to the warehouse2 ODS database worked only when the IP address was hard coded. Using the host name did not work.
6. Observation of the SQL*Server database showed that connected from oracle were establishing sessions within the database.
7. There were multiple invocations of the executable tg4msql.exe running upon the windows server. These relate directly one to one, with the sessions initiated from Oracle.
8. It was found that 'killing' the oracle sessions (within the ODS database) did not remove the sessions, which indicated they were marked for KILL but still remained. Killing the associated SQL*Server sessions also did not remove the oracle sessions within ODS. Only once the tg4msql.exe processes running on the Windows server were killed could did the Oracle sessions on PODS disappear.
9. It was discovered that ACL had been established upon the Windows UAT environment to prevent any communication between the machines within the UAT environment and any productions machines was the cause of the lack of communication between the AIX node and the Windows SQL*Server node. Once the ACL had been removed all communication sprang into life and worked without any further problems.
10. It will be necessary to re-establish the ACL settings in the near future now the connectivity has been proved.

NOTE: It is not possible to bounce the Windows gateway listener whilst any of the tg4msql.exe processes are running upon the machine. This is due to the tg4msql.exe processes using port 1521 (the listener default), and not releasing it, to enable the listener process to restart. This can be illustrated by running the command 'netstat –n' upon the windows machine and viewing the port allocation.

2.9 Oracle Connectivity

The following describes the method that Oracle uses when connecting to a database. This is included due to the problems encountered in setting up the UAT environment due to the ACL settings, but it is equally applicable to firewall setup generally.

When the Oracle client makes a connection to the database e.g. (sqlplus userid/password@alias), it compares the alias name supplied in the sqlplus line and looks for a match in the TNSNAMES.ORA file or Names server. Once it obtains the address for the database server, a connection attempt is made to the server from the client. The Listener is contacted on the database server and port redirection can take place depending on the platform, configuration of the INIT<SID>.ORA file and/or the Oracle product being used. The underlying network layer on the server will obtain a free port from the Operating System (OS) and send back to the client via the Listener the new port assignment. The client will then try to connect to the database on a new port. This is where connection failure normally occurs.

A remote Oracle client making a connection to an Oracle database can fail if there is a firewall installed between the client and the server and if port redirection is taking place. The firewall will block the connection to the new port when the Oracle client connects to the database - the client typically fails with Oracle errors ORA-12203 or ORA-12535. [In the case of the UAT setup no error message was seen, since the ACL prevented any response what so ever.] The client connection failure is due to port redirection from the Database Server's operating system. Port redirection requires the client to connect to the database using a different port than originally configured in the configuration files. Oracle Multi-Threaded Server (MTS) on Unix platforms, (without specifying the address with the ports in the INIT<SID>.ORA file), Oracle Secure Sockets Layer (SSL) and Windows NT/2000 platforms will cause port redirection.

A Net8 level 16 client trace file can verify if the problem is a firewall issue. In the SQLNET.ORA file on the client add the following lines:

```
  trace_level_client  = 16
  trace_file_client = client
  trace_directory_client = a valid directory and path
```

Save the changes to the SQLNET.ORA file and try connecting with SQL*Plus once to force the error. This will create a trace file. Here are several sample excerpts from a level 16-trace file to give an indication of what to look for.

The initial packets sent to the listener on port 1521 in trace file.

```
  niotns: Calling address:
(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
  (HOST=server1)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=v920.world)
  (CID=(PROGRAM=D:\V920\BIN\SQLPLUSW.EXE)(HOST=server1)(USER=system))))
  nladget: entry
  nladget: exit
  nscall: entry
  nscall: connecting...
  nsc2addr: entry
  nttbnd2addr: entry
  nttbnd2addr: port resolved to 1521
```

The received packet from the listener telling the client to use 1729 port (search for "NSPTRD").

```
  nscon: recving a packet
  nsprecv: entry
  nsbal: entry
  nsbgetfl: entry
  nsbgetfl: normal exit
  nsmal: entry
  nsmal: 44 bytes at 0xb892d0
  nsmal: normal exit
  nsbal: normal exit
  nsprecv: reading from transport...
  nttrd: entry
  nttrd: socket 232 had bytes read=64
  nttrd: exit
  nsprecv: 64 bytes from transport
  nsprecv: tlen=64, plen=64, type=5
  nsprecv: packet dump
  nsprecv:00 40 00 00 05 00 00 00  |.@......|
  nsprecv:00 36 28 41 44 44 52 45  |.6(ADDRE|
  nsprecv:53 53 3D 28 50 52 4F 54  |SS=(PROT|
  nsprecv:4F 43 4F 4C 3D 74 63 70  |OCOL=tcp|
  nsprecv:29 28 48 4F 53 54 3D 31  |)(HOST=1|
  nsprecv:33 38 2E 32 2E 32 31 33  |38.2.213|
  nsprecv:2E 36 31 29 28 50 4F 52  |.61)(POR|
  nsprecv:54 3D 31 37 32 39 29 29  |T=1729))|    <- port change
  nsprecv: normal exit
  nscon: got NSPTRD packet
  nscon: got 54 bytes connect data
  nscon: exit (0)
```

The client resolving the connection to port 1729.

```
  nscall: connecting...
  nsc2addr: entry
  nttbnd2addr: entry
  nttbnd2addr: port resolved to 1729
  nttbnd2addr: using host IP address: 138.2.213.61
  nttbnd2addr: exit
  nsc2addr: normal exit
```

You can see the send packets sent from the client on port 1521 (or your port if different) to the Listener. There will be receive packets returned from the server to the client reflecting a new port assignment. Then the client will send packets again from the client only this time to a different port. The connection will then fail at this point in the trace file.

The port that is assigned to the client is randomly chosen by the operating system and cannot be modified. It can be any free port available (usually above port 1024) that the server determines is not is use by any other software or hardware device.

3 Customer setup

This section details the specific set for the customers project.

3.1 Oracle databases

Two databases existing upon the server rdsgdw2 named ODS and ODS9 have been set up to enable connection to the 2 SQL*server databases located upon MACHINE3 and MACHINE4.

3.1.1 PODS9

This database is a Release 9.2.0.7 database.

3.1.2 PODS

This database is a Release 10.2.0.1 database.

3.2 UTILS schema

The UTILS schema within these databases is used to own the GATEWAYS package. In addition this schema has been given the privilege to create and delete PUBLIC database links.

The following script was run to create the 2 database links for SIT within these databases.

```
drop public database link omega_sit1;
create public database link omega_sit1
connect to "ODS_link" identified by "xxxxx"
using
'(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=machine3.customer.uk)(PORT=152
1))(CONNECT_DATA=(SID=tg4msql))(HS=OK))';
drop public database link omega_sit2;
create public database link omega_sit2
connect to "ODS_link" identified by "xxxxx"
using
'(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=machine4.customer.uk)(PORT=152
1))(CONNECT_DATA=(SID=tg4msql))(HS=OK))';
```

The password has been changed in the script above to ensure security.

The database links for the UAT and PROD environment are similar, differing only in the host name, username/passwords and obviously the link name itself.

3.3 Views

Some views are required in the SQL*Server database to ease selection from the Oracle database. Views created will be named as the first 28 characters of the table name appended with the characters '_V'.

Six views are required and these are detailed below.

The views were created on the MACHINE4 database [Omega_sit2] and then copied over by the system administrator to MACHINE3 [Omega_sit1]. The code for the creation is part of the GATEWAY package for the OMEGA_SIT2 instance.

Views were also established on the UAT environment and will be required on the production environment when it goes live.

3.3.1 View required in SQL*Server

The following SQL*Server tables need views:

Table Name	View Name
CUSTOMERVERSIONREGULAROUTGOINGS	CUSTOMERVERSIONREGULAROUTGO_V

The following table column names need views creating.

Table Name	Column Name	View Name	View Column Name
APPLICATIONFACTFIND	INTRODUCERLEVEL1INDUCEMENTVALUE	APPLICATIONFACTFIND_V	INTRODUCERLEVEL1INDUCEMENTVALU
APPLICATIONFACTFIND	INTRODUCERLEVEL2INDUCEMENTVALUE	APPLICATIONFACTFIND_V	INTRODUCERLEVEL2INDUCEMENTVALU
APPLICATIONFACTFIND	INTRODUCERLEVEL3INDUCEMENTVALUE	APPLICATIONFACTFIND_V	INTRODUCERLEVEL3INDUCEMENTVALU
BXAPPLICATIONBUREAU	BX12_PCPERCENTHHSWITHCCJSLAST36M	BXAPPLICATIONBUREAU	BX12_PCPERHHSWITHCCJSLAST36M
HOMETRACKVALUATIONDETAILS	VALUATIONRESULTSCONFIDENCELEVEL	HOMETRACKVALUATIONDETAILS_V	VALUATIONRESULTSCONFIDENCELEVE
HOMETRACKVALUATIONDETAILS	VALUATIONRESULTSREFERENCENUMBER	HOMETRACKVALUATIONDETAIL_V	VALUATIONRESULTSREFERENCENO
HOMETRACKVALUATIONDETAILS	VALUATIONRESULTSVALUATIONAMOUNT	HOMETRACKVALUATIONDETAILS_V	VALUATIONRESULTSVALUATIONAMT
INCOMESUMMARY	TOTALLOANSLIABILITYANNUALOUTGOINGS	INCOMESUMMARY_V	TOTALLOANSLIABILITYANNUALOUT
INCOMESUMMARY	TOTALLOANSLIABILITYOUTSTANDINGBALANCE	INCOMESUMMARY_V	TOTALLOANSLIABILITYOUTSTANDBAL
INCOMESUMMARY	TOTALMORTGAGEOUTSTANDINGBALANCE	INCOMESUMMARY_V	TOTALMORTGAGEOUTSTANDINGBAL
INCOMESUMMARY	TOTALREGULAROUTGOINGSANNUALOUTGOINGS	INCOMESUMMARY_V	TOTALREGULAROUTGOINGSANNUALOUT
INCOMESUMMARY	UNDERWRITEROTHERINCOMEPERCENTAGE	INCOMESUMMARY_V	UNDERWRITEROTHERINCOMEPERC
INCOMESUMMARY	UNDERWRITEROVERRIDEINCLUDEOTHERINC	INCOMESUMMARY_V	UNDERWRITEROVERRIDEINCLUDEOTH
MORTGAGESUBQUOTE	CONFIRMEDCALCULATEDINCMULTIPLIERTYPE	MORTGAGESUBQUOTE_V	CONFIRMEDCALCULATEDINCMULTTYPE

The following SQL*Server tables need views due to issues with the NVARCHAR2 fields that are greater than 1000 characters.

Table Name	**View Name**
MEMOPAD	MEMOPAD_V
CASETASK	CASETASK_V
CASETASKARCHIVE	CASETASKARCHIVE_V
CREDITCHECKREASONCODE	CREDITCHECKREASONCODE_V
RISKASSESSMENTRULEOVERRIDE	RISKASSESSMENTRULEOVERRIDE_V

The following table column(s) within the tables are the problem cause.

Table Name	**Column Name**
MEMOPAD	MEMOENTRY
CASETASK	ONTEXT
CASETASKARCHIVE	CONTEXT
CREDITCHECKREASONCODE	SMOVERRIDEREASON
RISKASSESSMENTRULEOVERRIDE	RAOVERRIDEREASON

In each of the above tables multiple output columns have been generated, each appended with an identified P? where the ? represents the part number.

3.4 Known Issues

3.4.1 NVARCHAR type longer than 1000 characters

A problem was discovered that relates to the selection of fields from a SQL*Server table containing a column whose size is greater than 1000 characters. i.e. NVARCHAR(4000).

Four tables were found to have this particular problem, MEMOPAD, CASETASK, CASETASKARCHIVE and CREDITCHECKREASONCODE. Each of these four tables contained one fields which was split down into four separate sub string fields with a created view. The select upon the view returns all the appropriate details, and the 4 sub fields could then be re-combined upon the Oracle database prior to being stored in a table.

The four views created are named:

MEMOPAD_V, CASETASK_V, CASETASKARCHIVE_V and CREDITCHECKREASONCODE_V.

The four subfields are named after the original column name appended with p1, p2, p3 or p4, reflecting the specific part of the string data.

3.4.2 NVARCHAR padding

A feature has been discovered in that SQL*Server variables defined as NVARCHAR are being mapped to NCHAR variables within the gateway. The NVARCHAR variable is a variable width character type, whilst the NCHAR variable is a fixed width character type. The net effect of this feature is that selection of the column from the SQL*Server table results in the variable being padded to the field length with NULL characters. Oracle has accepted the problem and a BUG report has been raised upon the subject.

Whilst a fix is awaited from Oracle there are two possible work solutions to get around the feature.

1. Implement a RTRIM function around each of the fields that are defined as NVARCHAR upon SQL*Server for the tables of interest.
2. The second option would be to create a view upon each of the required tables upon SQL*Server. A example using the casestage table and the stageid field is as follows:

   create view gsc_v1 as select CAST(stageid AS VARCHAR) AS stageid from casestage;

   A select from this view would return the variable of the correct length. In actual fact the view would need to encompass all the desired columns within the table and not just one field but the procedure is exactly the same.

Which every solution is employed, the end effect is the same with a character type translation occurring within the gateway along with an implicit trim, or the trimming occurring upon the Oracle RDBMS server. The deciding factor is likely to be the performance of the system where the work is to be carried out, on the SQL*Server machine or the Oracle machine.

4 Warehouse Requirements

The exact schema that will be present upon the intended SQL*Server target database is not currently known, however some tests have been carried out on a number of likely scenarios and solutions are suggested below.

4.1 Long table names and long column names.

There is a restriction imposed by Oracle upon the length of table and column names to 30 characters. SQL*Server does not have this restriction.

- What does Oracle do when a SQL server table name is over 30 characters?

1. SELECT from a long table name.

```
SQL> select * from aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa@test1msql.customer.uk;
select * from aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa@test1msql.customer.uk
          *
ERROR at line 1:
ORA-00972: identifier is too long
```

- What does Oracle do when a SQL server column name is over 30 characters?

2. SELECT from a view on a long table name.

```
select * from v_aaa@test1msql.customer.uk
          *
ERROR at line 1:
ORA-28500: connection from ORACLE to a non-Oracle system returned this message:
[Transparent gateway for MSSQL][H00C] Attempt to access a column
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA with unsupported name length (greater than 30
characters).
ORA-02063: preceding 2 lines from TEST1MSQL
ORA-00604: error occurred at recursive SQL level 1
ORA-01013: user requested cancel of current operation
```

The solution to the above is to create an alias for the table and/or the column via the use of a view located upon the SQL*Server database. Initially this has been done by Robin upon the test SQL*Server instance located upon MACHINE1, but it is suggested (see below) that this can be achieved via the gateway from the Oracle server using the DBMS_HS_PASSTHROUGH package.

4.2 SQL*Server data type conversion

SQL*Server data types are converted by the gateway into Oracle known databases. These are illustrated below in Table 1- Data type conversions. In addition SQL*Server allows for mixed (upper and lower) case characters for both table and column names. Whilst Oracle can also handle this situation, the SQL syntax is complicated by the need to surround each name by quote marks.

- What does Oracle convert the following SQL server data types to, and can data be copied.

SQL> desc v_aaa_v2@test1msql.customer.uk

```
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CustomerID                                NOT NULL NCHAR(5)
 AAA                                       NOT NULL NCHAR(40)
 ContactName                                        NCHAR(30)
 ContactTitle                                       NCHAR(30)
 Address                                            NCHAR(60)
 City                                               NCHAR(15)
 Region                                             NCHAR(15)
 PostalCode                                         NCHAR(10)
 Country                                            NCHAR(15)
 Phone                                              NCHAR(24)
 Fax                                                NCHAR(24)
 Test1                                              LONG RAW
 Test2                                              NUMBER(3)
 Test3                                              DATE
 Test4                                              FLOAT(49)
 Test5                                              NCHAR(50)
 Test6                                              RAW(6)
 Test7                                              LONG
 Test8                                              NUMBER(18)
 Test9                                              NUMBER(19,4)
 Test10                                             NUMBER(5)
```

The columns Test1 to test10 correspond to:

Image, tinyint, datetime, float, nvarchar, binary, text, numeric, money, smallint.

Using a mixed case example:

SQL> desc v_aaa_v3@test1msql.customer.uk

```
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CustomerID                                NOT NULL NCHAR(5)
 aAaAaAa                                   NOT NULL NCHAR(40)
 ContactName                                        NCHAR(30)
 ContactTitle                                       NCHAR(30)
 Address                                            NCHAR(60)
 City                                               NCHAR(15)
 Region                                             NCHAR(15)
 PostalCode                                         NCHAR(10)
 Country                                            NCHAR(15)
 Phone                                              NCHAR(24)
 Fax                                                NCHAR(24)
 Test1                                              LONG RAW
 Test2                                              NUMBER(3)
 Test3                                              DATE
 Test5                                              NCHAR(50)
 Test4                                              FLOAT(49)
 Test6                                              RAW(6)
 Test7                                              LONG
 Test8                                              NUMBER(18)
 Test9                                              NUMBER(19,4)
 Test10                                             NUMBER(5)
```

4.2.1 Create Table As Select (CTAS) Syntax

If one was to do a create table as… from sql server into Oracle. What does it do with column and table names that are held in mixed case in sql server?

Simple CTAS

SQL> create table ctas1 as select * from v_aaa_v2@test1msql.customer.uk;

```
create table ctas1 as select * from v_aaa_v2@test1msql.customer.uk
                             *
ERROR at line 1:
ORA-00997: illegal use of LONG datatype
```

If one tries the following, ignoring columns Test1 (LONG RAW) and Test7 (LONG)

```
create table ctas_v2 as
select
"CustomerID", AAA,
 "ContactName", "ContactTitle",
 "Address", "City", "Region", "PostalCode",
 "Country", "Phone", "Fax",
 "Test2", "Test3", "Test4", "Test5", "Test6", "Test8", "Test9", "Test10"
from v_aaa_v2@test1msql.customer.uk;
```

The table gets created successfully. The column names in the new table (CTAS_V2) are in mixed case as a consequence of the 'select'.

Oracle documentation states (Note:1022030.6):

```
LONG columns cannot be referenced when creating a table with query (CREATE
TABLE...AS SELECT..) or when inserting into a table(or view) with a query
(INSERT INTO ... SELECT...)
```

In addition one cannot use a 'to_lob()' function for the remote select.

A number of other options were tried including the SQLPLUS COPY command, and hit a number of problems, including ORA-12154, ORA-01041 but the final one that prevented any further wok in this area is ORA-28547 which seems to imply that the Transparent Gateway will not work with the COPY command. Consequently at the current time there is not a simple way to copy over the LONG & LONG RAW column data. Suggested alternative is to use a procedure within PL/SQL.

IF there is a view that does not contain the LONG or LONG RAW columns then a simple create table as select works fine.

```
SQL> desc v_aaa_v4@test1msql
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CustomerID                                NOT NULL NCHAR(5)
 aAaAaAa                                   NOT NULL NCHAR(40)
 ContactName                                        NCHAR(30)
 ContactTitle                                       NCHAR(30)
 Address                                            NCHAR(60)
 City                                               NCHAR(15)
 Region                                             NCHAR(15)
 PostalCode                                         NCHAR(10)
 Country                                            NCHAR(15)
 Phone                                              NCHAR(24)
 Fax                                                NCHAR(24)
 Test2                                              NUMBER(3)
 Test3                                              DATE
 Test5                                              NCHAR(50)
 Test4                                              FLOAT(49)
 Test6                                              RAW(6)
 Test8                                              NUMBER(18)
 Test9                                              NUMBER(19,4)
 Test10                                             NUMBER(5)
```

SQL> create table ctas_v2a as select * from v_aaa_v4@test1msql;

Table created.

SQL> desc ctas_v2a

```
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CustomerID                                NOT NULL NCHAR(5)
 aAaAaAa                                   NOT NULL NCHAR(40)
 ContactName                                        NCHAR(30)
 ContactTitle                                       NCHAR(30)
 Address                                            NCHAR(60)
 City                                               NCHAR(15)
 Region                                             NCHAR(15)
 PostalCode                                         NCHAR(10)
 Country                                            NCHAR(15)
 Phone                                              NCHAR(24)
 Fax                                                NCHAR(24)
 Test2                                              NUMBER(3)
 Test3                                              DATE
 Test5                                              NCHAR(50)
 Test4                                              FLOAT(49)
 Test6                                              RAW(6)
 Test8                                              NUMBER(18)
```

```
Test9                                                 NUMBER(19,4)
Test10                                                NUMBER(5)
```

4.3 Inline View Support

There does not appear to be any problems using inline views providing they are syntactically correct for the SQL*Server gateway.

4.4 Data Type Conversion

The gateway converts Microsoft SQL Server data types to Oracle data types as follows:

Microsoft SQL Server	**Oracle**	**Comment**
BINARY	RAW	-
BIT	NUMBER(3)	-
CHAR	CHAR	-
DATETIME	DATE	Fractional parts of a second are truncated
DECIMAL	NUMBER(p[,s])	-
FLOAT	FLOAT(49)	-
IMAGE	LONG RAW	-
INTEGER	NUMBER(10)	NUMBER range is -2,147,483,647 to 2,147,483,647
MONEY	NUMBER(19,4)	-
NCHAR	NCHAR	-
NTEXT	LONG	-
NVARCHAR	NCHAR	-
NUMERIC	NUMBER(p[,s])	-
REAL	FLOAT(23)	-
SMALL DATETIME	DATE	The value for seconds is returned as 0
SMALL MONEY	NUMBER(10,4)	-
SMALLINT	NUMBER(5)	NUMBER range is -32,767 to 32,767
TEXT	LONG	-
TIMESTAMP	RAW	-
TINYINT	NUMBER(3)	-
VARBINARY	RAW	-
VARCHAR	VARCHAR2	-

Table 1- Data type conversions

4.4.1 Date variables

Microsoft SQL Server does not support implicit date conversions. Such conversions must be explicit.

For example, the gateway issues an error for the following SELECT statement:

```
SELECT DATE_COL FROM TEST@MSQL

WHERE DATE_COL = "1-JAN-2004";
```

To avoid problems with implicit conversions, add explicit conversions, as in the following:

```
SELECT DATE_COL FROM TEST@MSQL

WHERE DATE_COL = TO_DATE("1-JAN-2004")
```

4.4.2 NCHAR variables

One issue discovered during testing involves the access of NCHAR columns within some of the SQL*Server tables with a length greater than 1000 characters. This can be considered as expected behaviour, in is described in Oracle Metalink Note 144808.1. This has nothing to do with the Gateway, but as the Gateway will "dynamically" build Oracle data types depending of the SQL Server data type, this will be made in evidence after the table creation step. Depending on the database national character set, the number of BYTES allowed in a NCHAR or NVARCHAR will change. Also, the Gateway mapping between SQL Server NCHAR/NVARCHAR and Oracle NCHAR/NVARCHAR2 will require more space then the original definition indicates. For example, a SQL Server nvarchar(1000) is mapped to a Oracle nvarchar2(2000). As, in AL16UTF16, the maximum for nvarchar2 is 2000, it will not be possible to access a SQL Server nchar > 1000. As can be guessed the NLS_NCHAR_CHARACTERSET for the PODS (and PODS9) database is AL16UTF16, the default national character set.

The solution is to split the field into multiple parts accessed via a view, and to concatenate the parts on the Oracle store, or access.

4.4.3 NVARCHAR padding

See section 3.4.2 for the possible solutions and issues relating to NVARCHAR padding within the gateway.

4.4.4 Microsoft SQL Server IMAGE, TEXT and NTEXT Data Types

The Oracle Manual states:

The following restrictions apply when using IMAGE, TEXT and NTEXT data types:

- An unsupported SQL function cannot be used in a SQL statement that accesses a column defined as Microsoft SQL Server data type IMAGE, TEXT or NTEXT.
- You cannot use SQL*Plus to select data from a column defined as Microsoft SQL Server data type IMAGE, TEXT or NTEXT when the data is greater than 80 characters in length. Oracle recommends using Pro*C or Oracle Call Interface to access such data in a Microsoft SQL Server database.
- IMAGE, TEXT and NTEXT data types must be NULLABLE for INSERT or UPDATE to work.
- A table including an IMAGE, TEXT or NTEXT column must have a unique index defined on the table or the table must have a separate column that serves as a primary key.

- IMAGE, TEXT and NTEXT data cannot be read through pass-through queries.
- If a SQL statement is accessing a table including an IMAGE, TEXT or NTEXT column, the statement will be sent to Microsoft SQL Server as two separate statements. One statement to access the IMAGE, TEXT or NTEXT column, and a second statement for the other columns in the original statement. This will result in two connections to Microsoft SQL Server due to a limitation in the Microsoft ODBC driver which only allows one statement for each connection, which can cause a hang depending on the sequence of SQL statements. If this happens, try issuing a commit and separating the statements in different transactions.

The only apparent solution to the handling of these LONG types seems to be to write a procedure in Pro*C or to use an OCI connection.

From the 10.2 Heterogeneous Connectivity Administration Guide:

Piecewise LONG Data Type

Earlier versions of gateways had limited support for the `LONG` data type. `LONG` is an Oracle data type that can be used to store up to 2 gigabytes (GB) of character data or raw data (`LONG RAW`). These earlier versions restricted the amount of `LONG` data to 4 MB because they treated `LONG` data as a single piece. This led to restrictions of memory and network bandwidth on the size of the data that could be handled.

Current gateways have extended the functionality to support the full 2 GB of heterogeneous `LONG` data. They handle the data piecewise between the agent and the Oracle server, thereby doing away with the large memory and network bandwidth requirements.

The `HS_LONG_PIECE_TRANSFER_SIZE` Heterogeneous Services initialization parameter can be used to set the size of the transferred pieces. For example, consider fetching 2 GB of `LONG` data from a heterogeneous source. A smaller piece size means less memory requirement, but more round-trips to fetch all the data. A larger piece size means fewer round-trips, but more of a memory requirement to store the intermediate pieces internally. Thus the initialization parameter can be used to tune a system for the best performance, that is, for the best trade off between round-trips and memory requirements. If the initialization parameter is not set, the system defaults to a piece size of 64 KB.

Note: Do not confuse this feature with piecewise operations on `LONG` data on the client side. Piecewise fetch and insert operations on the client side did work with the earlier versions of the gateways, and continue to do so. The only difference on the client side is that, where earlier versions of the gateways were able to fetch a maximum of 4 megabytes (MB) of `LONG` data, now they can fetch the entire 2 GB of `LONG` data. This is a significant improvement, considering that 4 MB is only 0.2% of the data type's full capacity.

5 Workarounds

Most of the examples make reference to the link test2msql. This link is created as described above, and is just another database link using the gateway.

5.1 Simple select using DBMS_HS_PASSTHROUGH

The following shows the use of DBMS_HS_PASSTHROUGH to obtain values back from the gateway. The variable type fetched into needs to be the same type as returned by the gateway. i.e. If the gateway returns the field NCHAR2 then the local variable should be NVARCHAR2).

```
declare
   val  VARCHAR2(100);
   nval nvarchar2(100);
   id   integer;
   c    INTEGER;
   nr   INTEGER;
BEGIN
   c:=  DBMS_HS_PASSTHROUGH.OPEN_CURSOR@test2msql;
   DBMS_HS_PASSTHROUGH.PARSE@test2msql(c,
           'select name, id from sysobjects');
   LOOP
     nr :=  DBMS_HS_PASSTHROUGH.FETCH_ROW@test2msql(c);
     DBMS_HS_PASSTHROUGH.GET_VALUE@test2msql(c,1,nval);
     DBMS_HS_PASSTHROUGH.GET_VALUE@test2msql(c,2,id);
     DBMS_OUTPUT.PUT_LINE(nval||' '||to_char(id));
   END LOOP;
   EXCEPTION
   WHEN NO_DATA_FOUND
   THEN  DBMS_HS_PASSTHROUGH.CLOSE_CURSOR@test2msql(c);
END;
/
```

5.2 Select using cursor for loop

The variable "Test7" is a LONG variable as determined by the MSQL gateway. The following illustrates the saving of the LONG variable into a table. [The table is a simple single LONG column.]

```
declare
   cursor c1 is
   select "Test7" test7 from v_aaa_v2@test2msql;

begin
   for r1 in c1 loop
      dbms_output.put_line('Yes '||substr(to_char(r1.test7),1,10));

      insert into gsc1 (l1)
        values (r1.test7);
      commit;
   end loop;
   dbms_output.put_line('Finished');
exception
   when no_data_found then
      dbms_output.put_line('No data');
end;
/
```

5.3 Creating a SQL*Server view

The following script was used to create a view upon the SQL*Server database. The connected user (oracle_con) had to have been granted the create view privilege upon the database before this would work successfully.

```
declare
   c    INTEGER;
BEGIN
   c:=  DBMS_HS_PASSTHROUGH.execute_immediate@test2msql
     ('create view gsc_test (ALIASCOL) as
      select AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA from
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa');
END;
```

```
/
```

The alternative syntax for creating a view with aliased names should also work:

```
CREATE VIEW GSC_TEST AS
SELECT AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AS ALIASCOL
FROM AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

6 GATEWAY Package

The GATEWAY package has been written to simplify the creation of views upon SQL*Server which are required to enable Oracle to access tables, or tables with column names that exceed 30 characters.

The package makes use of the DBMS_HS_PASSTHROUGH package that conceptually resides on the Microsoft SQL*Server gateway. In addition it utilises the views upon the SQL*Server objects.

Procedure Name	Parameter Name	Parameter Type	Comments
Debugging and Versioning Routines			
PROCEDURE PACKAGE_VERSION	MAJOR_VERSION	OUT NUMBER	
	MINOR_VERSION	OUT NUMBER	
PROCEDURE set_dbg			
PROCEDURE set_nodbg			
FUNCTION debugging			RETURN BOOLEAN
Internal Routines			
PROCEDURE run_on_sqlserver	sql_text	IN VARCHAR2	Externally exposed. See Note 1 below.
	server	IN VARCHAR2 DEFAULT 'OMEGA_SIT1'	
PROCEDURE run_on_sqlserver_sit1	sql_text	IN VARCHAR2	Calls main run_on_sqlserver procedure. Internal only
PROCEDURE run_on_sqlserver_sit2	sql_text	IN VARCHAR2	Calls main run_on_sqlserver procedure. Internal only.
PROCEDURE run_on_sqlserver_uat	sql_text	IN VARCHAR2	Calls main run_on_sqlserver procedure. Internal only.
PROCEDURE run_on_sqlserver_prod	sql_text	IN VARCHAR2	Calls main run_on_sqlserver procedure. Internal only.
PROCEDURE drop_view	p_viewname	IN VARCHAR2	Drops specified views. Internal only. See Note 1 below.
	server	IN VARCHAR2 DEFAULT 'OMEGA_SIT1'	
Main Routines			
PROCEDURE create_sit_views	N/A	N/A	Creates views on both OMEGA_SIT1 and OMEGA_SIT2 instance. Calls create_views1, 2 3 etc.
PROCEDURE create_views_sit1	N/A	N/A	Creates views on OMEGA_SIT1 instance. Calls

Procedure Name	**Parameter Name**	**Parameter Type**	**Comments**
			create_views1, 2 3 etc.
PROCEDURE create_views_sit2	N/A	N/A	Creates views on OMEGA_SIT2 instance. Calls create_views1, 2 3 etc.
PROCEDURE create_views_uat	N/A	N/A	Creates views on OMEGA_UAT instance. Calls create_views1, 2 3 etc.
PROCEDURE create_views_prod	N/A	N/A	Creates views on OMEGA_PROD instance. Calls create_views1, 2 3 etc. **NOTE: This is not active as of 9th Jan 2006.**
PROCEDURE create_views1	sit	IN VARCHAR2	Creates views on specified SQL Server instance. These views are upon the tables that contain NVARCHAR2 columns greater than 1000. See Note 1 below.
PROCEDURE create_views2	sit	IN VARCHAR2	Creates views on specified SQL Server instance. These views are upon the tables that contain NVARCHAR2 columns greater than 1000. See Note 1 below.
PROCEDURE create_views3	sit	IN VARCHAR2	Creates views on specified SQL Server instance. Currently only one view is involved. See Note 1 below.
PROCEDURE set_tg_mode	mode_in	IN VARCHAR2 [Must be one of COMMIT_CONFIRM, READ_ONLY, SINGLE_SITE or TWO_PHASE_COMMIT.]	Sets transparent gateway mode. [Effectively on effects COMMIT_CONFIRM mode which is default anyway.]

Table 2- GATEWAY package procedures

Note:

1 The server (or sit) parameter when specified is usually the name of the Oracle database link to the SQL Server instance. Currently four instances are known, with three being active, OMEGA_SIT1, OMEGA_SIT2, and OMEGA_UAT are live, and OMEGA_PROD is known. As a convenience the abbreviations SIT1, SIT2, UAT or PROD are acceptable.

It was originally intended to implement a generic naming of the created views, but upon inspection of the schema a more specific creation was decided upon. For that reason the procedures create_views, create_views1, create_views2 and create_views3 were written and comprise the actual code required to generate the specific views required. A simple SQL script was used to generate the basic code required, which was then implemented within the procedure.

```
set verify off
set pagesize 0
set trimspool on
set heading off
set feedback off
accept tab prompt 'Table name: '
spool &spoolname
select 'create view '||table_name||'_V ('||
      column_name||','
from dba_tab_columns@omega_sit2
where table_name = upper('&tab')
and column_id = 1;
select column_name||','
from dba_tab_columns@omega_sit2
where table_name = upper('&&tab')
and column_id > 1
order by column_id;
select ') as select '||column_name||','
from dba_tab_columns@omega_sit2
where table_name = upper('&&tab')
and column_id = 1;
select column_name||','
from dba_tab_columns@omega_sit2
where table_name = upper('&&tab')
and column_id > 1
order by column_id;
select 'from '||table_name
from dba_tables@omega_sit2
where table_name = upper('&&tab');
spool off
set verify on
set feedback on
set heading on
set pagesize 10
```

Figure 3 - View Creation Script

NOTES:

- A problem was encountered when writing the GATEWAY package such that it was not possible to use a cursor with an input variable accessing tables upon the SQL*Server node. This problem is suspected to be caused by something in the package, as a simple stand alone procedure works successfully. To get around the problem, the package makes use of DBMS_SQL to parse and execute code to access tables upon the SQL*Server node.
- At the current time, there does not appear to be a way in which the name of the database link (pointing to the SQL*Server gateway) can be set as a variable in one location. This also implies the passing of the link name into a procedure. The complication is involved with the DBMS_HS_PASSTHROUGH call.
- No attempt has been made to use an EXECUTE IMMEDIATE statement form, which is expected to resolve the need for multiple ‘duplicate’ procedures for multiple links.
- The entry point create_views_prod is coded but will generate an error until such time as the database links are established and the code is modified to remove the comment markers in two specific places in the code, and remove the error alert code. Around lines 164 and 292 in the package body.

6.1 Package error codes

There are a few error codes generated by the package which are as follows:

Error Code	Description
20001	Unexpected exception raised. This is a generic error message, which will be displayed along with the corresponding Oracle error message generated by the system. This problem is likely to be caused by an underlying database, or link problem rather than the Gateway package itself.
20002	Unknown gateway mode specified. This error is generated if an invalid mode is specified for the Transparent gateway. Acceptable values are as follows: COMMIT_CONFIRM, READ_ONLY, SINGLE_SITE or TWO_PHASE_COMMIT. See section 2.5.2 for more details.
20003	Unused.
20004	Invalid (or unknown) server (link) name specified. The specified database link name is not known, or is invalid. Currently there are four known links established, which are OMEGA_SIT1, OMEGA_SIT2, OMEGA_UAT and OMEGA_PROD although aliases SIT1, SIT2, UAT and PROD are acceptable as special cases. See note re OMEGA_PROD above.
20005	This error is raised when code that is not yet implemented is invoked through the entry points.

7 Tuning Considerations

There are no real parameters to tune generic connectivity or the open gateways. Everything that follows is to demonstrate the behaviour of heterogeneous services, to explain the working mechanism and what can be done to improve the performance.

Background information: HS connects from an Oracle database to a foreign data source and fetches records. The amount of the fetched records reflects the time needed to transfer them. The simplest way to improve performance is to reduce the amount of fetched data.

RULE 1:

> AVOID select * from remote table

Fetching all records from the remote database can sometimes occur without wanting to fetch all the records. The cause is called POST PROCESSING.

The Transparent Gateways tell the Oracle database during connect time which functions and operators they support. If the remote database does not support this function or operator, then the Oracle database must execute the function or operand. But to do this ALL records from the remote database must be fetched and processed locally in the Oracle database.

RULE 2:

> Pay attention to which functions/operands are supported by HS.
>
> Unsupported functions are post processed!

Some of the gateways such as the IBM gateways support explain plan. With this feature you're able to see the statements passed and processed. But this feature is not yet implemented in the generic connectivity or open gateways or the Microsoft SQL*Server gateway. So the only way to figure out what statements are passed between the Oracle database and the remote database is to verify the HS traces.

One option to solve this issue is to CREATE VIEWS.

With access to the remote database:

> It is possible to create a view at the remote database that pre selects the records. HS now performs its own query only to this view and processes only the pre selected records.

With no access to the remote database:

> The whole operation could be used in conjunction with DBMS_HS_PASSTHROUGH packages. DBMS_HS_PASSTHROUGH allows the creation of views at the remote database as well as the sending of select statements as they are to the foreign database without being pre processed.

Another option is to use the COST-BASED optimizer.

The cost-based optimizer uses indexes on remote tables and considers more execution plans then the rule based optimizer.

The real performance tuning is to reduce the amount of data fetched by the HS agent.

However there are a few parameters that adapt the interface and might improve performance as well.

By default, an agent fetches data from the non-Oracle system until it has enough data retrieved to send back to the HS related part in the Oracle database. The agent reblocks the data between the agent and the Oracle database server in sizes defined by the value of HS_RPC_FETCH_SIZE.

The other part is the transfer of the foreign database related interface like odbc and the agent. This transfer can be manipulated by setting the parameter

HS_FDS_FETCH_ROWS.

The default value is 20, but it makes sense to increase this value by adapting it to the fetched rows of the remote database.

While some ODBC drivers initialize the SQL_FETCH_ROWS to 100 the HS_FDS_FETCH_ROWS should be set to 100 or even to n*SQL_FETCH_ROWS.

For example, assume that you set HS_RPC_FETCH_SIZE to 64K and HS_FDS_FETCH_ROWS to 100 rows. Assume that each row is approximately 600 bytes in size, so that the 100 rows are approximately 60K.

The agent starts fetching 100 rows from the non-Oracle system.

Because there is only 60K bytes of data in the agent, the agent does not send the data to the Oracle database server. Instead, the agent fetches the next 100 rows from the non-Oracle system.

Now the agent is filled with 120K of data and the first 64K can be sent to the Oracle database server. There is 56K of data left in the agent. The agent fetches another 100 rows from the non-Oracle system before sending the next 64K of data to the Oracle database server.

From the description above it looks like the agent is a bottle neck. It only sends data after a buffer is filled. So a better idea would be to STREAM the data.

If the HS agent supports array fetching (please check out the documentation for a specific type of agent) the blocking can be switched of. Set the initialization parameter HS_RPC_FETCH_REBLOCKING to OFF.

According to the sample from above, this means that the first 100 rows are immediately sent to the Oracle server.

So in theory a performance improvement will take place by setting HS_RPC_FETCH_REBLOCKING to OFF and HS_FDS_FETCH_ROWS to a value of n*SQL_FETCH_ROWS.

KEEP IN MIND:

All these parameters manipulate the DATA TRANSFER. While each fetched record depends on networking capacity, on CPU usage, the best tuning mechanism is still to reduce the amount of transferred data.

And this can be achieved by CREATING VIEWS to pre-select the fetched records as described earlier.

A. Gateway Specifications

A.1 Supported Views and Tables

The gateway supports the following views and tables:

ALL_CATALOG	ALL_COL_COMMENTS
ALL_CONS_COLUMNS	ALL_CONSTRAINTS
ALL_IND_COLUMNS	ALL_INDEXES
ALL_OBJECTS	ALL_TAB_COLUMNS
ALL_TAB_COMMENTS	ALL_TABLES
ALL_USERS	ALL_VIEWS
DBA_CATALOG	DBA_COL_COMMENTS
DBA_OBJECTS	DBA_TAB_COLUMNS
DBA_TAB_COMMENTS	DBA_TABLES
DICT_COLUMNS	DICTIONARY
DUAL	TABLE_PRIVILEGES
USER_CATALOG	USER_COL_COMMENTS
USER_CONS_COLUMNS	USER_CONSTRAINTS
USER_IND_COLUMNS	USER_INDEXES
USER_OBJECTS	USER_TAB_COLUMNS
USER_TAB_COMMENTS	USER_TABLES
USER_USER	USER_VIEWS

A.2 Data Dictionary Mapping

The tables in this section list Oracle data dictionary view names and the equivalent Microsoft SQL Server system tables used. A plus sign (+) indicates that a join operation is involved.

View Name	**Microsoft SQL Server System Table Name**
ALL_CATALOG	sysusers + sysobjects
ALL_COL_COMMENTS	sysusers+sysobjects+syscolumns
ALL_CONS_COLUMNS	sp_pkeys + sp_fkeys
ALL_CONSTRAINTS	sysusers + sysobjects + sysindexes + sysconstraints + sysreferences
ALL_IND_COLUMNS	sysusers + sysindexes + syscolumns

ALL_INDEXES	sysusers + sysindexes + sysobjects
ALL_OBJECTS	sysusers + sysobjects + sysindexes
ALL_TAB_COLUMNS	sysusers + sysobjects + syscolumns
ALL_TAB_COMMENTS	sysusers + sysobjects
ALL_TABLES	sysusers + sysobjects
ALL_USERS	sysusers
ALL_VIEWS	sysusers + sysobjects + syscomments
DBA_CATALOG	sysusers + sysobjects
DBA_COL_COMMENTS	sysusers + sysobjects + syscolumns
DBA_OBJECTS	sysusers + sysobjects + sysindexes
DBA_TABLES	sysusers + sysobjects
DBA_TAB_COLUMNS	sysusers + sysobjects + syscolumns
DBA_TAB_COMMENTS	sysusers + sysobjects
DICT_COLUMNS	sysobjects + syscolumns
DICTIONARY	sysobjects
DUAL	sysusers
TABLE_PRIVILEGES	sysprotects + sysusers + sysobjects
USER_CATALOG	sysusers + sysobjects
USER_COL_COMMENTS	sysusers + sysobjects + syscolumns
USER_CONS_COLUMNS	sp_pkeys + sp_fkeys
USER_CONSTRAINTS	sysusers + sysobjects + sysindexes + sysconstraints + sysreferences
USER_IND_COLUMNS	sysusers + sysindexes + syscolumns
USER_INDEXES	sysusers + sysindexes + sysobjects
USER_OBJECTS	sysusers + sysobjects + sysindexes
USER_TAB_COLUMNS	sysusers + sysobjects + syscolumns
USER_TAB_COMMENTS	sysusers + sysobjects
USER_TABLES	sysusers + sysobjects
USER_USERS	sysusers
USER_VIEWS	sysusers + sysobjects + syscomments

A.3 Useful SQL Server view

One useful view has been created upon the Oracle database to view the ‘true’ table definitions upon the SQL Server database. Currently only created on the SIT1 instance the syntax for the view is as shown below:

```
CREATE OR REPLACE VIEW OMEGA_TAB_COLUMNS
(TABLE_NAME, COLUMN_NAME, DATATYPE, COLUMN_ORDER, NULLABLE,
 "prec", "scale")
AS
SELECT syso."name" table_name
      ,sysc."name" column_name
      ,syst."name" datatype
   --,syst2."name" datatype2
      ,sysc."colid" column_order
   ,decode(sysc."isnullable",0,'NOT NULL','NULL') nullable
   ,sysc."prec"
   ,sysc."scale"
FROM   "syscolumns"@omega_sit1 sysc
      ,"sysobjects"@omega_sit1 syso
      ,"systypes"@omega_sit1 syst
   --,"systypes"@omega_sit1 syst2
WHERE  syso."type" = 'U'                AND
       sysc."id" = syso."id"            AND
       syst."xusertype" = sysc."xusertype"
```