Informatica

And

Query Banding

Version: 1.1

Date: 30 June 2014

Copyright © 2014 by Teradata. All Rights Reserved.

TERADATA THE BEST DECISION POSSIBLE

Table of Contents

Overview

The primary purpose of this document is to provide information on how to implement 'query banding' in informatica.

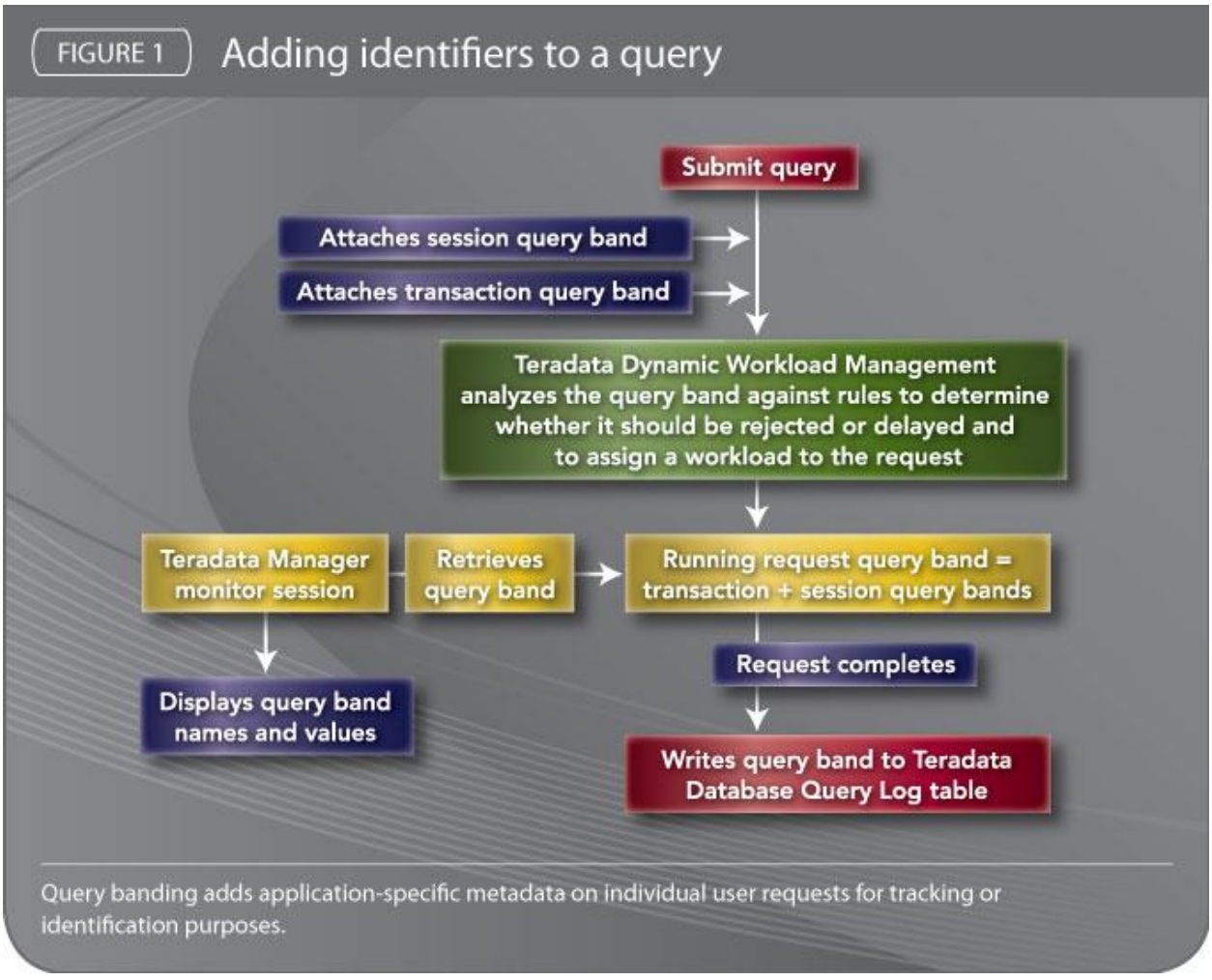
For a complete overview of the Query Band feature, please see Teradata User Documentation as well as existing Orange Books on this topic. This document will only discuss the opportunities to implement Query Bands within an Informatica environment.

What is Query Banding?

Scientists will often band the legs of birds with devices to track their flight paths. Monitoring and analyzing the data retrieved via the bands provides critical information about the species.

The same process holds true for DBAs who need more information about a query than what is typically available. Metadata—such as the name of the requesting user, the work unit and the application name—is important for, among other purposes, workload management, tracking the use of the data warehouse and query troubleshooting.

The sort of details provided through metadata can be linked to the query using the query banding feature in the Teradata Database. A query band contains any number of name/value pairs that use reserved or custom-defined names to set, for instance, the initiating user's corporate ID, department name and location, the name and version of the application, and the time the initiating thread of execution started. These query identifiers can be included in workload management rules and in applications and then captured in the Database Query Log (DBQL), where they are used to analyze the work flowing through the system (See figure 1).



Tag queries

Query banding is especially valuable when applications use connection pools to send queries to the database. Because each query session in a connection pool is tied to the database with the same user ID, all standard metadata is identical. Adding a query band that stores the detailed metadata enables various departments and units to identify the unique attributes of each request. For instance, DBAs can track the user or application of each request for chargeback purposes and associate users with requests for security measures.

Teradata Dynamic Workload Management also uses query banding to identify on a more specific level the queries that should receive a higher or lower priority. Concurrency levels are adjusted by assigning different workload rules. System administrators use query banding for troubleshooting to identify queries that are running too long or those that have created a system backlog. The information typically available in queries does not offer this sort of refinement.

Determine usage needs

Before query banding can be applied, users must first strategize how it will be employed, then design the query band names and values. Query bands may be set with the specific values (indicated in parentheses) and used to:

- Identify the origin of application requests made via connection pools (Client User, Group, Application Name, Source and Action)
- Associate all requests in a job (JobID, JobLen and JobSeq)
- Adjust the priority of a job (Importance level)

The standard set of query band names is defined for use by Teradata enterprise applications, as well as by the applications of customers and partners. (See table.) The list of reserved and optional query band names will provide additional consistency.

Examples of query band name/value pairs are:

```
- ApplicationName=InventoryApp;

- Version=01.00.00.00;

- ClientUser=dg120444;

- JobID=998;
```

Banding

The query band is set by using the “SET QUERY_BAND” SQL statement:

```
SET QUERY_BAND = 'ApplicationName=InventoryApp; Version=01.00.00.00;' FOR
SESSION;
```

The session query band is stored in the session table and applied to each request. It remains set for the duration of the session or until the query band is replaced by another. In case of a system reset, the query band is recovered.

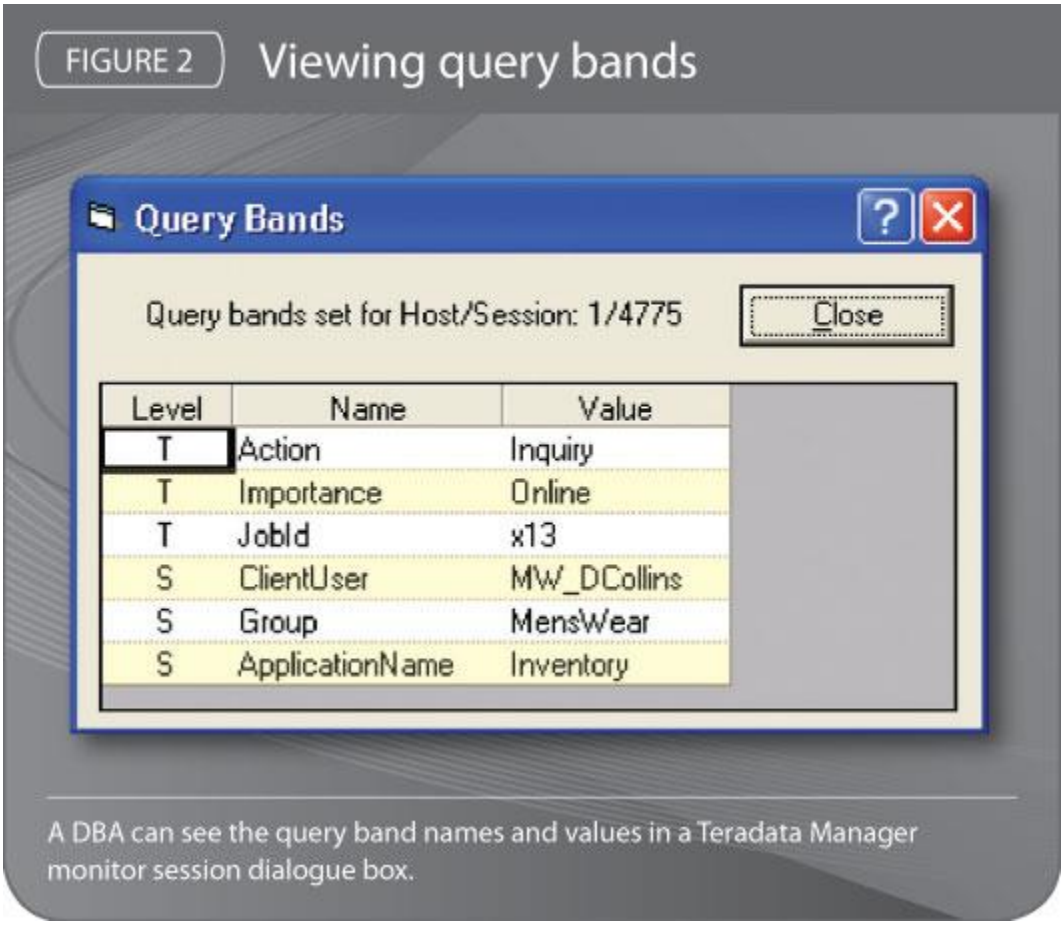
A query band can also be applied to all of the requests within a transaction. When the application requests a connection from the pool to perform a service for a client user—for instance, saving data to the database—it sets the transaction query band with attributes specific to that service request, as follows:

```
SET QUERY_BAND = 'ClientUser=dg120444; Group=Sales; JobID=998;' FOR
TRANSACTION;
```

When the transaction completes, the transaction query band is automatically discarded. Therefore, no cleanup is required before reusing the connection for another service request.

Landing

DBQL is used to record query processing activity. When Query Logging is enabled, the query band for each request is written to the DBQlogTbl table.



For chargeback, accounting and other types of system management, administrators find it useful to have the additional data that is provided with the query band, such as corporate user ID, department and report name. If an application has set the query band with ClientUser name, for example, users can extract resource usage reports from DBQL based on this information:

```
SEL SUM(AMPCPUTIME), SUM(t1.ParserCPUTime)from dbc.qrylog t1 where

t1.QueryBand is NOT NULL AND GetQueryBandValue(t1.queryband, 0,
```

```
'ClientUser') = 'MW_DCollins';

SUM(AMPCPUTime)  SUM(ParserCPUTime)

--------------  --------------------

         0.016                0.062
```

Informatica and Query Banding

Currently, QB implementation can be implemented in the following Informatica Connections:

A) TPTAPI (i.e., Power Exchange for Teradata Parallel Transporter API)
B) Relational (i.e., use of ODBC DSN from a Relational Connection)

Currently, use of Query Bands within a stand-Alone Utilities Job (FastLoad, MultiLoad, TPump, and FastExport) is not supported.

Note the syntactical differences between inputting Query Band name-value pair between TPTAPI and non-TPTAPI scenarios.

- TPTAPI QB input field requires only the name-value pairs separated by a semicolon (;).
- The Relational-Teradata-ODBC input field requires the standard SET QUERY_BAND = 'a=1; ' FOR SESSION; syntax.

Many Built-in variables can be utilized for dynamic value portion of the Query Band. Refer to Transformation Language Reference documentation for updated list. See the example for Relational/ODBC for details.

TPT API Connections

Minimum Requirements:

- Power Exchange for Teradata Parallel Transporter 8.6.1.0.3

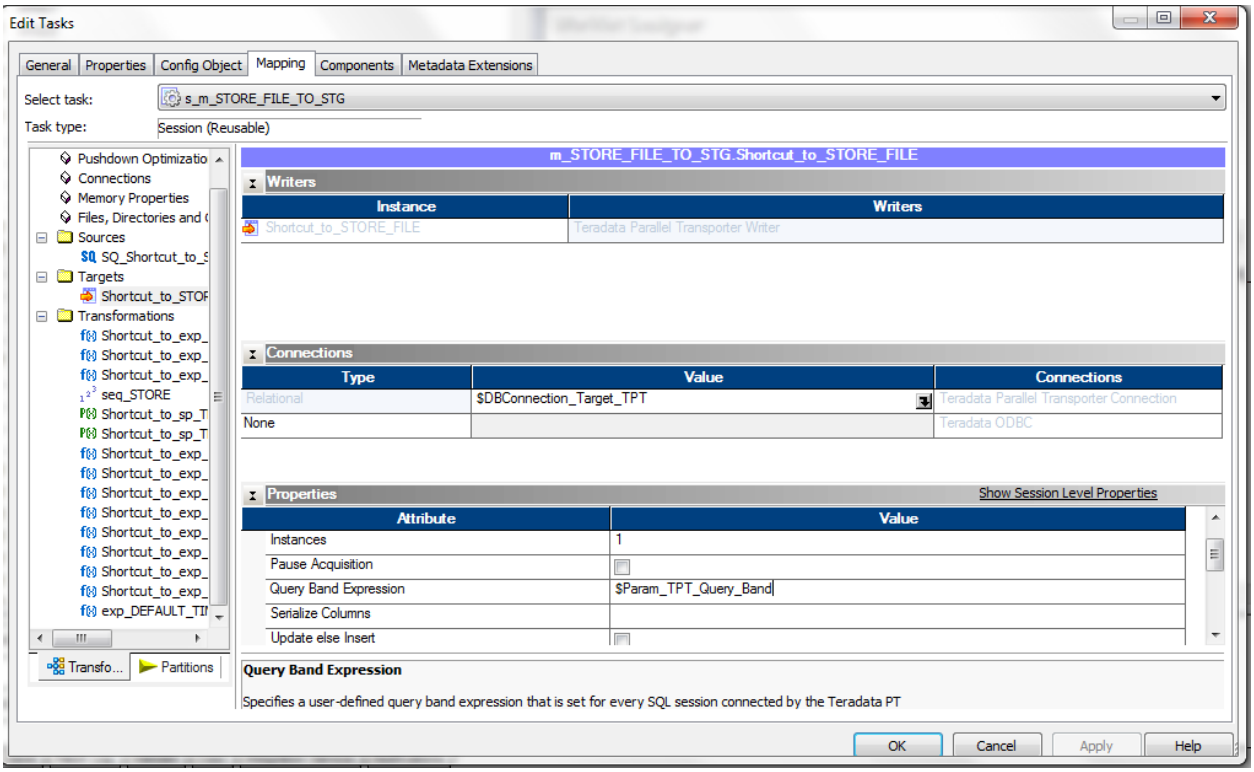
The query band can be set in the informatica session where a Teradata Parallel Transport Reader or Writer is used.

The attribute 'Query Band' can be set with name-value pairs, with each name-value pair terminated with a semi-colon.

The internal informatica pre-defined parameters can be used. Those are then 'parsed' and set in the query band during initialization. (see appendix for a list of parameters and what they do).

The query band string can be set from a parameter whom's value resides in the parameter file.

See example below:



If the parameter was set to be

$Param_TPT_Query_Band=APP=Infa;UserName=$PMRepositoryUserName;StartTime=$$$SessStartTime
;

The output in the DBQL table would be

DBQL result: =S> APP=Infa;UserName=Administrator;StartTime=03/12/2012 11:38:44;

Sample:

$Param_TPT_Query_Band=APP=INFA;LOADER=TPT;PMFolderName=$PMFolderName;PMWorkflowRunId=$PMWorkflowRunId;PMWorkflowName=$PMWorkflowName;PMSessionName=$PMSessionName;PMMappingName=$PMMappingName;PMRepositoryUserName=$PMRepositoryUserName;StartTime=$$$SessStartTime; Table_Name=$Param_Target_Table_Name;

ODBC Connections

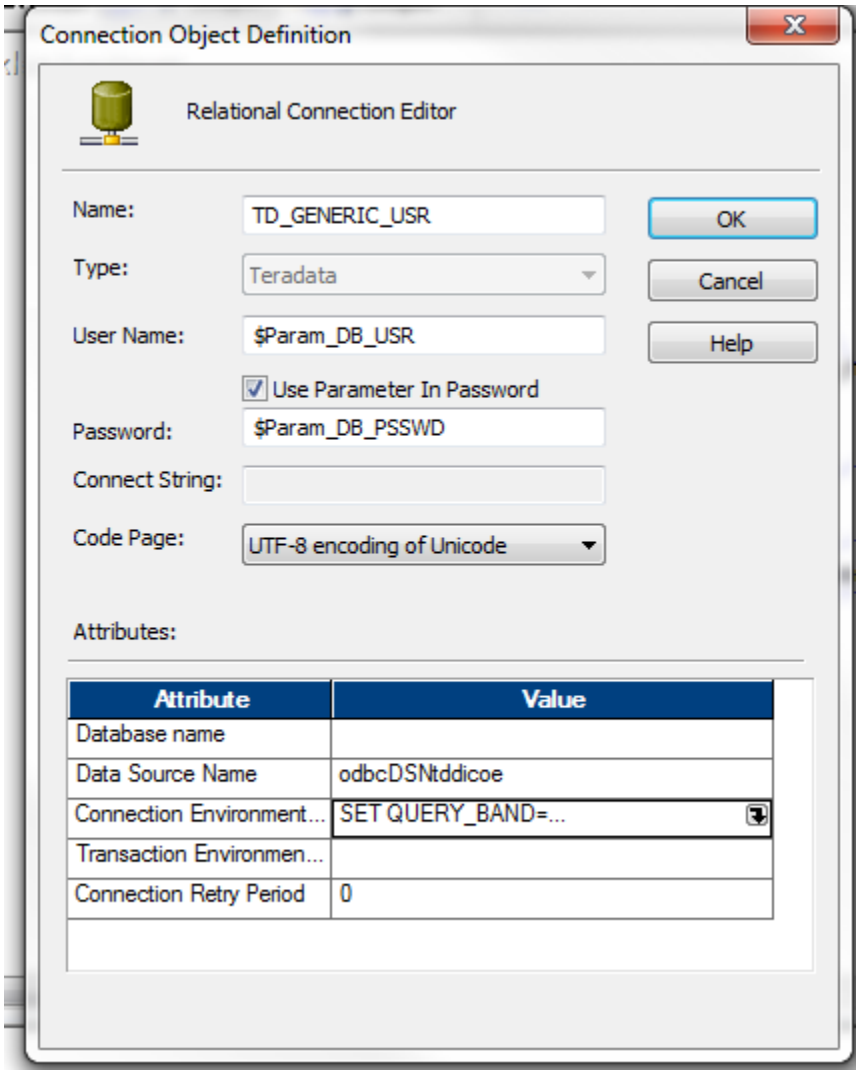
The Query Band can be set in the 'Connection' section of the odbc connection. Every time this connection is used, the query band will be set.

Best practice is to use the internal informatica pre-defined parameters that provide information about what workflow, session, mapping, user etc is been used.

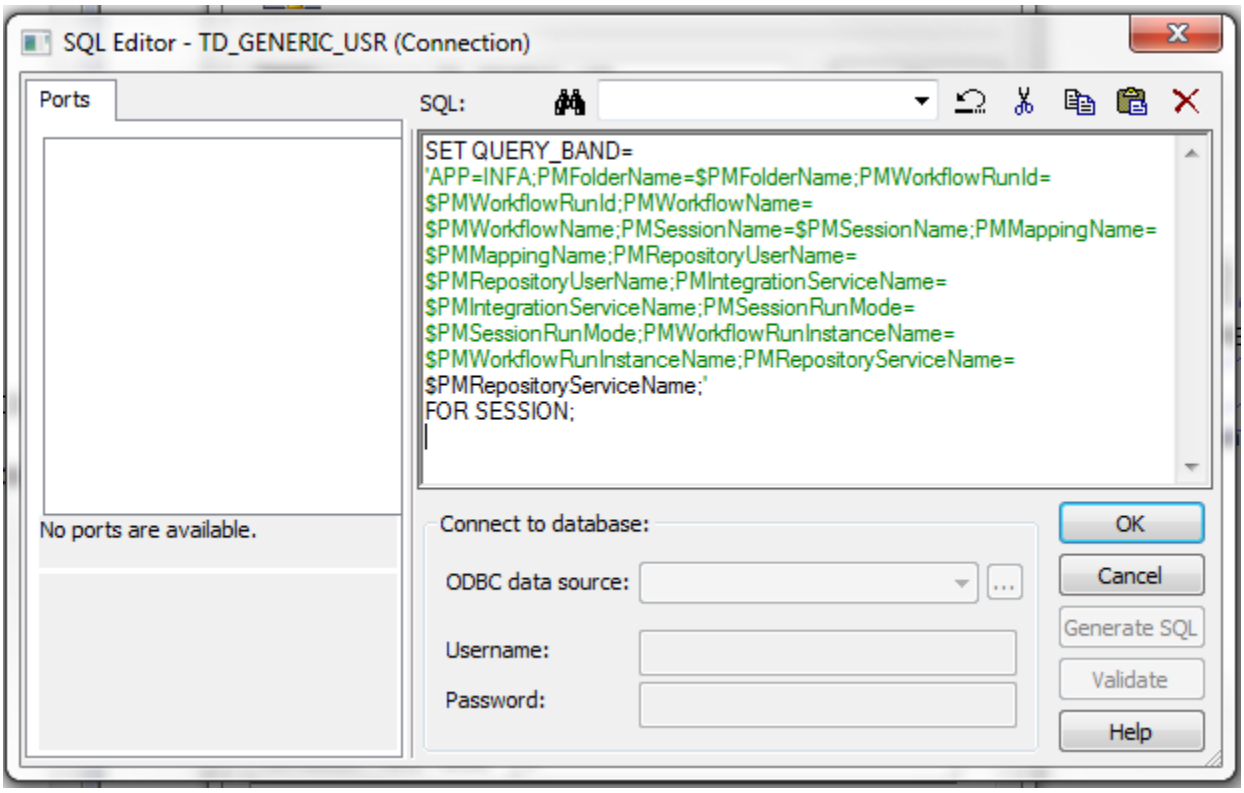
The syntax for the connection string is :

```
SET QUERY_BAND =
'App=InventoryApp;UserName=$PMRepositoryUserName;StartTime=$$$SessStartTime;' FOR
SESSION;
```

Ie ODBC Connection information



Ie Connection Attribute:



The output in the DBQL table would be

DBQL result: =S>
APP=INFA;PMFolderName=SAMPLES_CORE;PMWorkflowRunId=2839;PMWorkflowName=wf_STORE_STG_TO_STORE_CORE;PMSessionName=s_m_STORE_INS;PMMappingName=m_STORE_INS;PMRepositoryUserName=Administrator;PMIntegrationServiceName=DICOEis_USA;PMSessionRunMode=Normal;PMWorkflowRunInstanceName=;PMRepositoryServiceName=DICOErep_USA;

Sample

```
SET QUERY_BAND=
'APP=INFA;PMFolderName=$PMFolderName;PMWorkflowRunId=$PMWorkflowRunId;PMWorkflowName=$PMWorkflowName;PMSessionName=$PMSessionName;PMMappingName=$PMMappingName;PMRepositoryUserName=$PMRepositoryUserName;PMIntegrationServiceName=$PMIntegrationServiceName;PMSessionRunMode=$PMSessionRunMode;PMWorkflowRunInstanceName=$PMWorkflowRunInstanceName;PMRepositoryServiceName=$PMRepositoryServiceName;StartTime=$$$SessStartTime;'
FOR SESSION;
```

Appendix

Informatica Pre-Defined Parameters

Informatica Pre-defined Parameters are as follows:

Parameter Name	Description
$PMFolderName	Returns the folder name.
$PMWorkflowName	Returns the workflow name.
$PMWorkflowRunId	Returns the workflow run ID.
$PMWorkflowRunInstanceName	Returns the workflow run instance name.
$PMIntegrationServiceName	Returns the Integration Service name.
$PMMappingName	Returns the mapping name.
$PMRepositoryServiceName	Returns the Repository Service name.
$PMRepositoryUserName	Returns the repository user name.
$PMSessionName	Returns the session name.
$PMSessionRunMode	Returns the session run mode (normal or recovery).
$PMSourceName@TableName	Returns the table name for the named source instance.
$PMTargetName@TableName	Returns the table name for the named target instance.
$$$SessStartTime	Returns the date time the session executes