

Oracle® Product Data Quality

COM Interface Guide

Version 5.5

June 2010

Copyright © 2001, 2010, Oracle and/or its affiliates. All rights reserved.

Primary Author: Lorna Vallad

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	v
Audience	v
Related Documents	v
Conventions	vi
1 Overview	
Overview of the Oracle DataLens Server APIs	1-1
APIs	1-1
Platforms	1-2
Pre-Installation Requirements.....	1-2
Installation of the Oracle DataLens Server COM Components	1-3
COM High-Level Overview	1-3
High-Level DSA Interface.....	1-3
2 System and Server	
Transform System Object	2-1
Getting the List of Available Servers.....	2-1
Manually Creating a Server Object.....	2-1
Checking the Version of the COM Object	2-2
Transform Server Object	2-2
Getting Server Information.....	2-2
3 DSA Jobs	
Options on a Job Object	3-1
Filter Data	3-1
Separator Character	3-1
Email	3-1
FTP.....	3-2
Starting a Job	3-2
Collection Input.....	3-2
Array Input	3-3
Database Parameter Input	3-3
Getting the Result Data from the Job	3-3
Simple DSA Job	3-4
Synchronous Example.....	3-4

Asynchronous Example	3-4
Multiple Output DSA Job	3-4
Synchronous Example.....	3-4
Getting Result Data from the Individual Job Steps	3-5
Extracting Data from the Transformed Records	3-5
Pull the Data from the Transformed Records	3-5
Pull the Data from the Transformed Records as Tab-Separated Output.....	3-5
Optional - Waiting for the Job to Complete	3-6
Syncing Returned Results with the Original Input Data.....	3-6

4 Server Information

Server Repository Information	4-1
DSAs.....	4-1
DSA Details	4-1
Transform Maps	4-2
Transform Map Details	4-2
Data Lenses	4-3
Data Lens Details.....	4-3
Server DSA Job Information.....	4-3
Options on Listing DSA Jobs on the Server	4-3

5 Error Handling

Client-Side Exceptions	5-1
Server-Side Exceptions	5-1
Server-Side Log Messages.....	5-1

A Server Information API

Server Information API to the Oracle DataLens Server	A-1
Getting Transform Map and Data Lens Information.....	A-1
Instantiate the InfoClient	A-1
Get Data Lens Information	A-1
Initialize the COM Object with the Following Parameters.....	A-1
Get the Data Lens Information from the Oracle DataLens Server.....	A-1
Get Transform Map Information	A-2
Initialize the COM Object with the Following Parameters.....	A-2
Get the Transform Map Information from the Oracle DataLens Server.....	A-2
Check the State of the Client-Side Objects	A-3
Check the Status of Initialization and Session	A-3
Check the Status of Instantiation	A-3

B System Configuration File

C Installing the Client Software

Preface

This guide is intended to explain the basic capabilities of the Oracle Product Data Quality Server COM Interface.

To understand all of the features presented, you must use this guide in conjunction with the Oracle Product Data Quality documents listed in "[Related Documents](#)" on page -v.

You must have the Oracle Product Data Quality client software installed on your computer.

Audience

A thorough understanding of the material in this guide is required for the following customer personnel:

- Subject Matter Experts (SMEs)
- IT Administrators

Related Documents

For more information, see the following documents in the documentation set:

- The *Oracle Product Data Quality Oracle DataLens Server Installation Guide* provides detailed Oracle Product Data Quality Oracle DataLens Server installation instructions.
- The *Oracle Product Data Quality Oracle DataLens Server Administration Guide* provides information about installing and managing an Oracle DataLens Server.
- The *Oracle Product Data Quality Java Interface Guide* provides information about installing and using the Oracle DataLens Server Java APIs.
- The *Oracle Product Data Quality Application Studio Reference Guide* provides information about creating and maintaining Data Service Applications (DSAs).
- The *Oracle Product Data Quality AutoBuild Reference Guide* provides information about creating initial data lens based on existing product information and data lens knowledge.
- The *Oracle Product Data Quality Knowledge Studio Reference Guide* provides information about creating and maintaining data lenses.
- The *Oracle Product Data Quality Glossary* provides definitions to commonly used Oracle Product Data Quality technology terms.

- The *Oracle Product Data Quality Services for Excel Reference Guide* provides information about creating a DSA based on data contained in a Microsoft Excel spreadsheet.
- The *Oracle Product Data Quality Task Manager Reference Guide* provides information about managing tasks created with the Task Manager or Governance Studio applications.

See the latest version of this and all documents listed at the Oracle Product Data Quality Documentation Web site at:

http://download.oracle.com/docs/cd/E17135_01/index.htm

Conventions

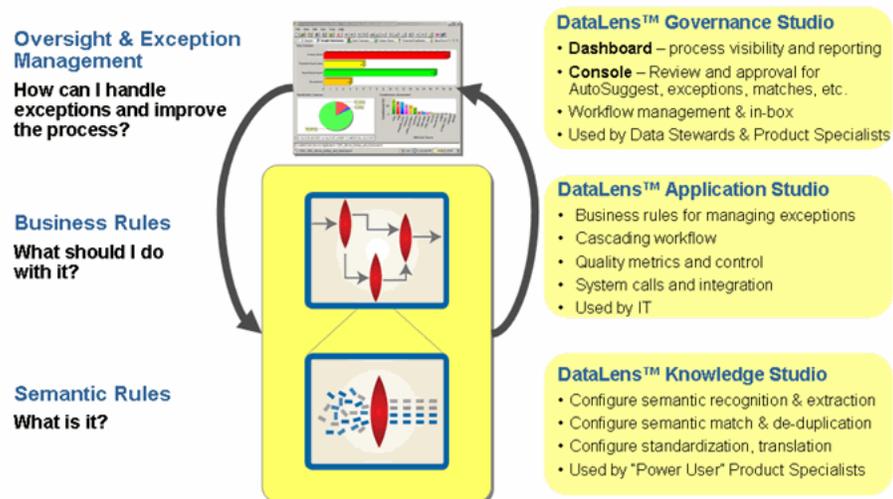
The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	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, text that you enter, or a file, directory, or path name.
monospace	Boldface, monospace type indicates commands or text that you enter.

Overview

Oracle DataLens Server is built on industry-leading DataLens™ Technology to standardize, match, enrich, and correct product data from different sources and systems. The core DataLens Technology uses patented semantic technology designed from the ground up to tackle the extreme variability typical of product data.

Oracle Product Data Quality uses three core DataLens Technology modules: Governance Studio, Knowledge Studio, and Application Studio. The following figure illustrates the process flow of these modules.



Overview of the Oracle DataLens Server APIs

APIs

The Application Studio and Governance Studio are the primary ways to access the Oracle DataLens Server platform. These are the recommended interfaces for application developers.

DSA Client

The Application Studio is used for direct access to the DSAs loaded on the Oracle DataLens Servers for processing application/enterprise data. The DSA can process

lists of tab-separated data. This is primarily exposed with the job and job step objects. The DSA Client can process the following:

- Tab-separated (or user-defined separator character) input data
- Input data from a database query
- XML input data using an Xpath query
- XML document as input; the updated XML document is returned.

Transform Server Client

The Governance Studio allows access to the repository and DSA jobs on the Oracle DataLens Server.

There are two low-level Application Programming Interfaces to the Oracle DataLens Server platform. These interfaces are not recommended for application developers. If these interfaces are needed, then Oracle Product Data Quality Professional Services should be contacted to get assistance on the best practices and use of these interfaces to the Oracle DataLens Servers.

Oracle DataLens Client

The Oracle Server Real-Time Client interface. This is used for direct access to the data lens loaded on the Oracle DataLens Servers for processing application data. The RT Client can process the following:

- Single line of data
- Array of data
- List of data

Information Client

The Oracle DataLens Server Information Client interface. This is used for access to information about the data lens Transform Maps that are loaded on the Oracle DataLens Servers. This is useful to check what KBs and Maps are available from a client application

Platforms

The Microsoft COM DLL can be used for integrating to the following

- Visual Basic
- Microsoft applications such as Excel and Access
- VBA applications such IIS Web pages
- HTML pages
- Available only on Microsoft Windows operating systems

Pre-Installation Requirements

Visual Basic 6.0 - The COM dll was compiled and built with this release.

This document assumes that the reader is an experienced Microsoft Visual Basic or VBA software developer.

All libraries and software needed for use by this API are loaded as part of the Knowledge Studio and Oracle DataLens Server installations.

- To access the Oracle DataLens Server from a Microsoft ASP (Active Server Pages) or VB application, the Oracle DataLens Server COM component can be used.
- The COM object can also be used to access the Oracle DataLens Server from a Web Browser using client-side scripting such as VBScript, Jscript, or JavaScript.
- The COM object can also be embedded in applications such as Microsoft Excel using VBA scripting.
- The COM component can be used from any application that supports the Microsoft COM interface.

Installation of the Oracle DataLens Server COM Components

The Oracle DataLens Server COM component to the Server is provided as part of the normal server installation, although it is not automatically loaded onto your system for you.

Copy the COM library file from the Oracle Product Data Quality Installation disk from the `/Interfaces` directory on the CD-ROM.

The file is called `STARServices.dll`.

Copy this file to the following directory:

```
\Datalens\Applications\lib
```

This COM component now needs to be registered on your system for use by your Microsoft ASP (or Microsoft VB) applications. Enter the following on the command-line of your machine:

```
regsvr32 c:\Datalens\Applications\lib\ SCSServices.dll
Set TransformSystem = New SCSServices.TransformSystem
```

There is a separate installation for the COM objects that will install the COM components into the `C:\Datalens\Applications\lib` directory and register them automatically.

COM High-Level Overview

The DSA APIs are the only interfaces where the submitted jobs can be tracked and manipulated on the server. The DSAs provide the highest level of abstraction from the low-level processing, minimizing the amount of code that needs to change when modifying the Transform Maps and data lenses.

High-Level DSA Interface

This is the preferred interface to access the DSAs.

It is setup with the following VB Objects:

System

Top level object, contains information on all the servers in the topology. Contains a collection of server objects

Server

A particular server in the topology (from the system object) to which job requests are made. The server has Jobs.

Job

This is a particular DSA job. Monitors the progress of that job. Retrieves the results of specific steps from the job. Contains a collection of Job Step objects

Job Step

A particular step can have the result data return to the application program. Contains a collection of transformed record objects.

Transformed records

A collection of the result data from the job.

Transformed fields

A collection of the individual fields in each record.

System and Server

The Oracle DataLens Server object has information on all the servers in the Oracle DataLens Server group. The system object returns Oracle DataLens Server objects. The Oracle DataLens Server objects are used to process and get information from a particular Oracle DataLens Server group.

Transform System Object

```
Public oTransformSystem As SCSServices.TransformSystem
```

Getting the List of Available Servers

When this object is instantiated, the servers that are configured are made available.

```
' Create the System object
Set oTransformSystem = New SCSServices.TransformSystem
If oTransformSystem.IsError Then
    m_sStatusTest = "Error creating SCSServices.TransformSystem" & vbCrLf
End If

' Load the configuration information from the system.cfg file
oTransformSystem.Load ("admin")

'Get the list of available servers
Dim arrServerNames
arrServerNames = oTransformSystem.names
Dim i As Integer
For i = 0 To UBound(arrServerNames)
    Combo_SCS_servers.AddItem (arrServerNames(i))
Next
```

Manually Creating a Server Object

Sometimes it is not convenient to load a configuration file from a known location. In that case, the name of the server can be passed into the creation function.

```
Set oTransformServer = oTransformSystem.createDataServer(Text_url.Text, _
    Text_port.Text, _
    Text_code.Text, _
    Text_application.Text)
If oTransformServer.IsError Then
    RichTextBox_1.Text = "Error creatinSCSServices.TransformServer" & vbCrLf
End If
```

Checking the Version of the COM Object

The version of the COM DLL can be checked to verify that an older or newer version is not being used.

```
Dim sDLLVersion As String
sDLLVersion = g_oTransformSystem.getVersion()
RichTextBox1.Text = "Initialized using SCS DLL Version " + sDLLVersion
```

Transform Server Object

```
Public TransformServer As SCSServices.TransformServer
```

When a job is run against a particular Oracle DataLens Server, the DSA is always run on that particular server. The processing of the Transform Maps and data lenses may be done on any server in the same server group that the DSA is a member of. The Server decides which server to use based on the load on each server and automatically provides load balancing. For information on configuring each server to run particular types of maps or data lenses, see *Oracle Product Data Quality Oracle DataLens Server Administration Guide*.

Getting Server Information

Additional information can be obtained from the server configuration file such as the following.

Information	Function Call
Server Name	<code>oTransformServer.Name</code>
Server Port	<code>oTransformServer.portNumber</code>
Server URL	<code>oTransformServer.serverURL</code>
Application Title	<code>oTransformServer.appTitle</code>
User	<code>oTransformServer.user</code>
Name Filter	<code>oTransformServer.nameFilter</code>
Map Filter	<code>oTransformServer.caMapFilter</code>
CFB Client code	

The Job object is used to start jobs, check on the job status, and retrieve the results from finished jobs. The Transform Server object is used to define the server that the job will be run against.

There is one option that is required to be set after creating the job object

· **Transform Server** - The server object of the server on which to run the DSA.

```
' Create the Job object
  Dim oJob As Job
  Set oJob = New Job
  oJob.DataServer = oTransformServer
```

Options on a Job Object

The following options can be set on a job object

Filter Data

This will filter out invalid control characters from the input data that prevent the HTTP transfer of data. This is ignored if the input data is from a database query.

```
oJob.filterData = False
Dim lJobId As Long
```

Separator Character

To define a user-specified separator character, other than the default tab character. This character is used to delineate the input data fields and result data fields

```
oJob.separatorChar = "|"
```

To toggle this option back off and revert to the default Tab separator character, just set separatorChar to an empty string.

Email

Setting this will email the result of the job to the specified email address. Note that this will use the email server configured in the Oracle DataLens Server Administration Web Pages. If this is specified, then the job results will not be returned to the application program, but will be emailed instead.

```
oJob.Email = yogi@bear.com
```

To toggle this option back off, just set Email to an empty string.

FTP

Setting this will FTP the results of the job to the named FTP configuration. The FTP configuration is setup in the Oracle DataLens Server Administration Web Pages. If this is specified, then the job results will not be returned to the application program, but will be FTPd instead.

```
oJob.ftp = "CompanyCFG"
```

To toggle this option back off, just set FTP to an empty string.

Starting a Job

A job can be started in asynchronous mode. The job is started and a job Id is returned that is used to check on the status of the job and to get the result data when the job has finished. There are three different types of input that can be used to start a job. They are as follows:

1. Collection Input
2. Array Input
3. Db Parameter input

Collection Input

A collection of String input-fields of data is passed to the start method and a single job ID is returned. The start Method is called just a single time with the collection.

Actual parameters:

Job Priority

This is the priority the job will be given on the server for processing. Large batch overnight jobs should be given a priority of low. Small jobs with few input records, or requests that need a quick response, such as users waiting for a response should get a priority of High. All other jobs should use a priority of medium. Note that the number of concurrent jobs that can be run on the server is also controlled by the priority of the job (See the Server configuration pages of the Oracle DataLens Administration Web pages for more information). There are the priority values that can be used.

- "Low"
- "High"
- "Medium"

Description

A description of this particular job.

DSA Name

The name of the DSA to run on the Oracle DataLens Server

Data Collection

A collection of collections of input data fields as Strings.

Run-time Locale

This locale to use for output for this job.

Return data:

Job ID

The DSA Job ID obtained from the Start call.

```
' Rem Start the WFG Job with an input data Collection
  lJobId = oJob.Start(sPriority, sDescription, _
                    sProcessMap, dataCollection, sRuntimeLocale)
If oJob.IsError Then
  Call reportError(oJob.errorMsg)
  Exit Function
End If
```

Array Input

An array of tab-separated data is passed to the start method and a single job ID is returned. The start Method is called just a single time with a single one-dimensional array of data.

Actual parameters are the same as for the collection-based start method. The difference is that the fourth parameter is an array of tab-separated data:

Array

The Array of tab-separated input data to be processed.

```
' Rem Start the WFG Job with an input data array
Dim lJobId As Long
  lJobId = oJob.StartArr(sPriority, sDescription, _
                       sProcessMap, dataArray, sRuntimeLocale)
If oJob.IsError Then
  Call reportError(oJob.errorMsg)
  Exit Function
End If
```

Database Parameter Input

Here is the code to start a job with a database query as input. The dataArray is replaced with the arrQueryParameterList:

```
Dim arrQueryParameterList
arrQueryParameterList = Split("p1|p2|p3, "|")

' Run a job with database input and an array of Db parameters
lJobId = oJob.startDb(sPriority, sDescription, _
                    sProcessMap, arrQueryParameterList, sRuntimeLocale)
```

Getting the Result Data from the Job

All of these results can be obtained synchronously or asynchronously. The last parameter in the getResultData and getResultStep data calls is the flag:

WaitForResults

This flag is true for synchronous and false for asynchronous

The advantage of synchronous calls is that the code is very simple and straight-forward. The call will not return until the data has finished processing.

The advantage of asynchronous calls is the application code can continue processing while the job finishes processing until the result data is ready. This is useful for doing additional processing for large DSA jobs and for keeping a user-interface responding to user-input, rather than hanging until the result data is returned.

Simple DSA Job

A simple DSA job is defined as a job with no approvals or reviews and only a single output step. This is also a very common scenario for DSA jobs called from the API.

Synchronous Example

```
' Synchronously get the results;
' The function call will not return till the data is ready
Set TransformedRecords = oJob.getResultData(lJobId, True)
If TransformedRecords.IsError Then
    MsgBox (TransformedRecords.errorMessage)
    GoTo exception
End If
```

Asynchronous Example

```
' Asynchronously get the results
Do
    Set TransformedRecords = oJob.getResultData(lJobId, False)
    If TransformedRecords.IsError Then
        If TransformedRecords.errorCode = oJob.JOB_NOT_COMPLETED Then
            Sleep (5 * 1000)
        Else
            MsgBox (TransformedRecords.errorMessage)
            GoTo exception
        End If
    End If
Loop While TransformedRecords.errorCode = oJob.JOB_NOT_COMPLETED
```

Multiple Output DSA Job

The transformed result data can also be retrieved directly from the job object by using named output steps. This is useful for integrating a DSA process into an application where there are multiple outputs that need to be used for different purposes within the application.

Synchronous Example

```
' Synchronously get the results;
Set TransformedRecords = oJob.getResultStepData(lJobId, sStepName, True)
If TransformedRecords.IsError Then
    MsgBox (TransformedRecords.errorMessage)
    GoTo exception
End If
```

Getting Result Data from the Individual Job Steps

The transformed result data can also be obtained directly from "output" Job Steps as follows. This gives the application programmer control of the job down to the individual DSA step level. This can be useful for an application where the actual names of the output steps are not known at run-time.

```
'Retrieve result sets for all output steps
Dim JobStep As JobStep
Dim TransformedRecords As TransformedRecords
For Each JobStep In oJob.steps
  If JobStep.isOutputStep Then
    If oJobStep.isCompleted Then
      Set TransformedRecords = JobStep.getResults(TransformServer, True)
      If TransformedRecords.IsError Then
        Call reportError(TransformedRecords.errorMsg)
        GoTo exception
      End If
    End If
  End If
End If
Next
```

Extracting Data from the Transformed Records

Pull the Data from the Transformed Records

The individual fields can be extracted from each transformed record.

```
'Iterate through the collection of transformed records
'and the collection of transformed fields
Dim TransformedRecord As TransformedRecord
Dim sField As String
For Each TransformedRecord In TransformedRecords.Items
  Dim TransformedField As TransformedField
  For Each TransformedField In TransformedRecord.Items
    sField = TransformedField.Value
  Next
Next
```

Pull the Data from the Transformed Records as Tab-Separated Output

This is useful if there is only a single field of data returned. This can also be a quick check for debugging to determine the correctness of the records being returned.

By default, the data is tab-separated, but this can be changed during the job.start call by defining a different separator character.

```
'Iterate through the collection of transformed records
Dim TransformedRecord As TransformedRecord
For Each TransformedRecord In TransformedRecords.Items
  tab_separated_data = TransformedRecord.tabSepResult
Next
```

Optional - Waiting for the Job to Complete

The job is running on the server when the Job Id is returned from the job.start function call. The job can be checked for completion before the results can be obtained as demonstrated in the following:

Note: Each individual job step can be checked for completion, rather than checking the entire job as is done in the following code example.

Note: Job data can also be retrieved synchronously, eliminating the need to check the status of the job.

```
'Get the job from the server
Dim oJob As Job
Set oJob = g_TransformServer.getJob(jobId)
If oJob.IsError Then
    Call reportError(oJob.errorMessage)
    GoTo exception
End If

'Results can only be retrieved once
If oJob.isResultsRetrieved Then
    GoTo exception
End If

'Wait for the job to finish
Do While Not oJob.IsDone
    Sleep (5 * 1000)
    Set oJob = g_TransformServer.getJob(jobId)
    If oJob.IsError Then
        Call reportError(oJob.errorMessage)
        GoTo exception
    End If
Loop
```

The results of the job are in the job object once the job has finished.

Syncing Returned Results with the Original Input Data

The DSA Maps are not guaranteed to be in records input and in records returned. They are also not guaranteed that the output records will be in the same order as the input records. For this reason, the Id values need to be matched up with the input records if the input records are to be updated with the results from the DSA call.

A simple example follows where we have input data in a RecordSet called RS and we are simply updating the description with the matching ID values:

```
RS.MoveFirst
For lRsCounter = 0 To RS.RecordCount - 1
    ' Get the returned record based on the ID
    Dim TransformedRecord As TransformedRecord
    sID = UCase(Trim(RS("ID")))

    On Error Resume Next
    Set TransformedRecord = TransformedRecords.Item(sID)
    ' Error will be thrown if we attempt to access an item that does not
```

```
' exist in the returned data; in that case we skip to the next record
If Err Or (TransformedRecord Is Nothing) Then
    Err.Clear
    GoTo nextRecord
End If
On Error GoTo 0

iFieldCount = TransformedRecord.Count()
' Pull out the SCS Returned data Fields from the matching records
Dim TransformedField As TransformedField
For Each TransformedField In TransformedRecord.Items
    sField = TransformedField.Value
    If (Count = 0 And iFieldCount > 0) Then
        sScsId = sField          ' 1st field has the Id
    ElseIf (Count = 1 And iFieldCount > 1) Then
        sScsDesc = sField       ' Next field has the description
    ElseIf (Count = 2 And iFieldCount > 2) Then
        sScsCategory = sField   ' Next field has the category
    End If
    Count = Count + 1
Next

RS.Edit
RS("Description") = Trim(sScsDesc)
If sScsCategory <> "" Then
    RS("Category") = Trim(sScsCategory)
End If
RS.Update
nextRecord:
    RS.MoveNext
Next lRsCounter
```

Server Information

The Transform Server object can also be used to retrieve information about the server. These types of information can be useful for applications where a selection list of maps and data lenses needs to be presented to the user.

Typically, processing is done with known Maps that are hard-coded into your application. In this case, this server information would not be needed.

Server Repository Information

DSAs

A list of all the DSAs that are available to the server can be obtained.

```
Dim oProcessMaps As ProcessMaps
Dim oProcessMap As ProcessMap
Set oProcessMaps = oTransformServer.getProcessMaps()
If oProcessMaps.IsError Then
    MsgBox ("Error Getting TMap information: " + oProcessMaps.errorMessage)
    Exit Sub
End If
For Each oProcessMap In oProcessMaps.Items
    m_tmpStatus = m_tmpStatus + oProcessMap.Name + vbCrLf
Next
```

DSA Details

Details on individual DSAs can be obtained from the server. This includes:

- Name
- Description
- Input Steps
- Output Steps
- Database Connections

```
Dim oDSA As ProcessMap
Set oDSA = g_oTransformServer.getProcessMap(sPMapName)
If oDSA.IsError Then
    MsgBox ("Error Getting DSA information: " + oPMap.errorMessage)
    Exit Sub
End If
```

- `oPMap.Name` - This will return the DSA Name as a String
- `oPMap.Description` - This will return the description of the DSA
- `oPMap.inputSteps` - This returns a collection of jobStep objects
- `oPMap.outputSteps` - This returns a collection of jobStep objects.
- `oPMap.dbConnections` - This returns a collection of Strings with the connection name
- `oPMap.transformationMaps` - This returns a collection of Strings with the Transformation Map name

Transform Maps

The server object can be used to return the list of Transform Maps used by the server. For Example:

```
Dim oTMaps As TransformMaps
Dim oTMap As TransformMap
Set oTMaps = g_oTransformServer.getTransformMaps()
If oTMaps.IsError Then
    MsgBox ("Error Getting TMap information: " + oTMaps.errorMessage)
    Exit Sub
End If
For Each TMap In oTMaps.Items
    m_tmpStatus = m_tmpStatus + oTMap.Name + vbCrLf
Next
```

Transform Map Details

Details on the individual Transform Maps can be obtained from the Transform Server object. This includes:

- Name
- Description
- Input Columns
- Output Columns
- Database Connections
- Data lenses Used
- Output Maps

```
Dim oTMap As TransformMap
Set oTMap = g_oTransformServer.getTransformMap(sTMapName)
If oTMap.IsError Then
    MsgBox ("Error Getting TMap information: " + oTMap.errorMessage)
    Exit Sub
End If
```

- `oTMap.Name` Name - This will return the Transform Map Name as a String
- `oTMap.Description` - This will return the description of the DSA
- `oTMap.isDecisionMap` - Boolean flag on the type of Transform Map

- `oTMap.hasDbDataSource` - Boolean flag on the data input. There will be no input Columns if the Db Data Source is true.
- `oTMap.inputColumns` - Returns a collection of column names as strings
- `oTMap.outputColumns` - Returns a collection of column names as strings
- `oTMap.dbConnections` - Returns a collection of database connections as strings
- `oTMap.kbNames` - Returns a collection of data lens names as strings
- `oTMap.outputMaps` - Returns a collection of output Transform Maps as strings

Data Lenses

The Transform Server object can be used to return a list of data lenses that are in the server Repository.

```
Dim oDLs As KnowledgeBases
Dim oDL As KnowledgeBase
Set oDLs = g_oTransformServer.getKnowledgeBases
If oDLs.IsError Then
    MsgBox ("Error Getting TMap information: " + oDLs.errorMessage)
    Exit Sub
End If
For Each oDL In oDLs.Items
    m_tmpStatus = m_tmpStatus + oDL.Name + vbCrLf
Next
```

Data Lens Details

The only details that can be obtained on a data lens is the description.

```
Dim oDL As KnowledgeBase
Set oDL = g_oTransformServer.getKnowledgeBase(Text_DLName)
If oDL.IsError Then
    MsgBox ("Error Getting DSA information: " + oDL.errorMessage)
    Exit Sub
End If
```

- `oDL.Name` - This will return the data lens name as a String.
- `oDL.Description` - This will return the description of the data lens.

Server DSA Job Information

Information on the DSA jobs that have been submitted to the server can be obtained from the Transform Server object.

Options on Listing DSA Jobs on the Server

- All Jobs run on a particular server can be listed.
- All Jobs run on all servers in the server group can be listed.
- Jobs can be listed by date range from the current time.
- Jobs can be filtered by the user that submitted the job.
- Jobs can be filtered to just the jobs that were started on this particular Server.
- Jobs can be filtered by the job status

- Currently running jobs
- Jobs awaiting user input
- Completed, failed or cancelled jobs
- Pending jobs

There are two calls that are used to get all this job information:

```
Dim oJobs As Jobs
Set oJobs = g_oTransformServer.getJobsBySubmitter(user, bListAllServers, _
                                                sinceSecs, bRefresh)
Set oJobs = g_oTransformServer.getJobs(bListAllServers, sinceSecs, bRefresh)
```

These are the parameters used in the preceding call:

- User - The user that submitted the job
- bListAllServers - List all jobs on all servers, not just this particular server
- sinceSecs - Number of seconds to go back looking at the job history
- bRefresh - Refresh the job list since this Transform Server object was created

Following is a code example that shows the job information for the individual jobs.

```
Dim oJobs As Jobs
For Each oJob In oJobs.Jobs
    Call displaySingleJob(oJob)
Next

Private Sub displaySingleJob(oJob As Job)
    m_tmpStatus = CStr(oJob.jobId) + vbTab
    m_tmpStatus = m_tmpStatus + CStr(oJob.priority) + vbTab
    m_tmpStatus = m_tmpStatus + oJob.getStatusDesc(oJob.Status) + vbTab
    m_tmpStatus = m_tmpStatus + oJob.runtimeLocale + vbTab
    m_tmpStatus = m_tmpStatus + oJob.createdBy + vbTab
    m_tmpStatus = m_tmpStatus + oJob.Description + vbTab
    m_tmpStatus = m_tmpStatus + oJob.definition + vbTab
    m_tmpStatus = m_tmpStatus + oJob.startTime + vbTab
    m_tmpStatus = m_tmpStatus + oJob.server + vbTab
End Sub
```

Client-Side Exceptions

All of the Oracle DataLens objects keep track of the error state. This includes the API objects and the Oracle DataLens objects that return collections of data. Simply add the following check:

```
If oSilverCreekObject.IsError Then
' Process the error in
' oSilverCreekObject.errorMsg
End If
```

Server-Side Exceptions

Most of the server-side errors are propagated back to the client where they can be checked for and reported on.

Server-Side Log Messages

Go to the Oracle DataLens Server Administration Web Pages and examine the log file from the home page. This will have a listing of any errors that were encountered in the server-side processing of your request.

DSAs can be checked with the Oracle DataLens Server Administration Web Pages for status. If there is a problem, then the Job ID, DSA name and Map Step where the error occurred is listed, with the complete error message.

Server Information API

This API has been deprecated. The TransformServer Object should be used instead.

Server Information API to the Oracle DataLens Server

Getting Transform Map and Data Lens Information

The Oracle DataLens Server provides a COM component called Client Info, which is part of the `SCSServices.dll` library.

This is used as an interface to the Oracle DataLens Server. This COM component provides functions to get information on the data lenses and the Transform Maps.

Instantiate the InfoClient

```
Dim oInfoClient as SCSServices.ClientInfo
Set oInfoClient = new SCSServices.ClientInfo
```

Get Data Lens Information

Initialize the COM Object with the Following Parameters.

- Server name or address
- Server port number
- Oracle DataLens Server-supplied application API access code
- Application name/identifier used for tracking statistics/usage

```
REM Initialize the COM Object with the server information
call oInfoClient.initProjectClient(serverName, serverPort, clientCode, userName)
```

Get the Data Lens Information from the Oracle DataLens Server

```
REM Do the Transform Map Transformation
Dim dataResultPrj
dataResultPrj = oInfoClient.transformData()
```

`dataResultPrj` is an array of data lens information with the following output fields:

- Data lens Name
- Source locale

```
FOR i=1 to UBound(dataResultPrj, 1)
Val = "DataLensName" & dataResultPrj(i,1)
    Val2 = "Source Locale" & dataResultPrj(i,2)
Next
```

Get Transform Map Information

There are both array and collection interfaces to this information. This document describes the collection interface.

Initialize the COM Object with the Following Parameters. ■ Server name or address

- Server port number
- Oracle DataLens Server-supplied application API access code
- Application name/identifier used for tracking statistics/usage

```
REM Initialize the COM Object with the server information
Call oInfoClient.initMapClient(serverName, serverPort, clientCode, userName)
```

Get the Transform Map Information from the Oracle DataLens Server

```
Rem Do the Transform Map Transformation and get a ProjectClient.STARMapInfo object
Set oStarMapInfo = oInfoClient.transformData2()
```

StarMapInfo is an Object that can be accessed as follows:

```
Dim oStarMap As Object
Dim cStarMaps As Collection
Set cStarMaps = oStarMapInfo.getStarMaps()
For Each oStarMap In cStarMaps
    Dim sMapName As String
    Dim bIsDecision, bHasDbSource As Boolean
    Dim lInputColumnsCount, lOutputColumnsCount As Long

    Rem Get the name and properties
    sMapName = oStarMap.mapName
    bIsDecision = oStarMap.IsDecisionMap
    bHasDbSource = oStarMap.hasDbSource
    resultStr = resultStr + sMapName + vbCrLf
    resultStr = resultStr + vbTab + "Is Decision: " + CStr(bIsDecision) +
vbCrLf
    resultStr = resultStr + vbTab + "Has Db Src: " + CStr(bHasDbSource) +
vbCrLf

    Rem Get the Input columns
    lInputColumnsCount = oStarMap.InputColumnsCount
    resultStr = resultStr + vbTab + CStr(lInputColumnsCount) + " Input
Columns." + vbCrLf
    Dim cInputColumns As Collection
    Set cInputColumns = oStarMap.InputColumns
    Dim oInputColName As Variant
    For Each oInputColName In cInputColumns
        resultStr = resultStr + vbTab + vbTab + oInputColName + vbCrLf
    Next
```

```

        Rem get the Output columns
        lOutputColumnsCount = oStarMap.OutputColumnsCount
        resultStr = resultStr + vbTab + CStr(lOutputColumnsCount) + " Output
Columns." + vbCrLf
        Dim cOutputColumns As Collection
        Set cOutputColumns = oStarMap.OutputColumns
        Dim oOutputColName As Variant
        For Each oOutputColName In cOutputColumns
            resultStr = resultStr + vbTab + vbTab + oOutputColName + vbCrLf
        Next
        resultStr = resultStr + vbCrLf
    Next

```

Check the State of the Client-Side Objects

Check the Status of Initialization and Session

This function can be used to output informational messages while dumping out the state of the ClientProcessMap object. This function must be called after the object has been initialized and a session has been started.

```

checkStr = ObjReference.CheckInitAndStart("Processing was done with the following
parameters.")

```

The output will be similar to the following:

```

Processing was done with the following parameters.; Run-Time Server URL:
http://127.0.0.1:8080/datalens/XfmRt; Oracle Product Data Quality Cleansing and
Matching Server Map name: sampleMap; Oracle Product Data Quality Cleansing and
Matching Server User name: testApplication

```

Check the Status of Instantiation

Another function that can be used to check that the COM object is being accessed correctly from the application program is just called CheckCOM. This function just returns the string that is passed to it, verifying that the COM object has been instantiated correctly and that it is working properly. The Oracle DataLens COM object does not need any special initialization to call this particular function.

```

dim checkComStr
checkComStr = ObjReference.CheckCOM("Test the COM object for a response.")

```

System Configuration File

This file is read by the `TransformSystem` object. The `TransformSystem` object reads the list of Oracle DataLens Servers and associated connection parameters from this configuration file, if available. A collection of `TransformServer` objects is built from this file.

`system.cfg` is a simple tab separated file with a header row. The file is located in the `/datalens/applications/config` directory.

Server Configuration File Format:

SERVER	PORT	CLIENTCODE	USER	PWD	NAMEFILTER	CAMAPFILTER
my-staging	8080	myAssignedCode	ksmith	xky3kx	Main	Classif
my-prod	8090	anotherCode	ksmith	xky3kx	Prod	Classif

The user name and password are currently not used. The calling application is responsible for supplying the user name.

Only the first `NAMEFILTER` is used in the file.

The `CAMAPFILTER` is not used.



Installing the Client Software

Oracle Product Data Quality uses a concept called Java Web Start to initially install and maintain the current version of the software on your client desktop. The process requires you to access the Oracle DataLens Server to initiate the connection and download the software.

You download and install the Oracle Product Data Quality client applications using Java Web Start by browsing to the installation page for your Oracle DataLens Server as follows:

1. Using Microsoft Internet Explorer, browse to one of the following URLs as appropriate for your server:

Note: If you setup a different port number for your application server other than 2229, you must use that port number in the following URL when browsing to the Oracle DataLens Server to download the client applications.

32-bit

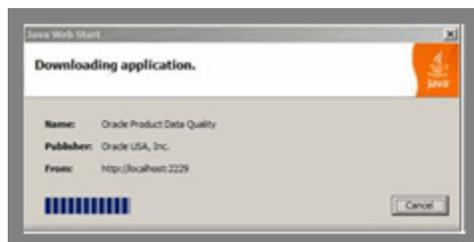
<http://<server>:2229/datalens/datalens.html>

64-bit

<http://<server>:2229/datalens/datalens64.html>

Where *<server>* is the hostname of the Oracle DataLens Server

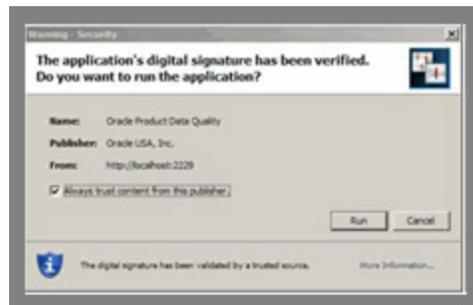
The application download and installation begins. If you do not have a supported Java environment on the target installation machine the Java Web Start program automatically redirects you to a Java download site and begins a Java Runtime installation.



2. If the preceding Java Web Start message is not displayed, you must initiate a connection and download the software by browsing to:

<http://<server>:2229/datalens/datalens.jnlp>

Oracle Product Data Quality files are digitally signed by a trusted source so the following security warning is displayed.



3. To avoid the security dialogue in the future you can select the **Always trust content from this publisher** check box.
4. Click **Run** to continue and complete the installation.
The Oracle Product Data Quality log on dialog is displayed.

