Oracle® Enterprise Data Quality for Product Data

.NET API Interface Guide

Release 5.6.2

E48206-01

July 2013

This document provides information about the Enterprise DQ for Product (EDQP) .NET application programming interface (API). The EDQP .NET API provides a communication interface to the Oracle DataLens Servers and includes the following:

1 Introducing the EDQP .NET API

The EDQP .NET API is developed using the Microsoft .NET Framework 2 (Microsoft .NET Framework 4 is also supported). It is a language-neutral platform for you to use to integrate the .NET API into your applications.

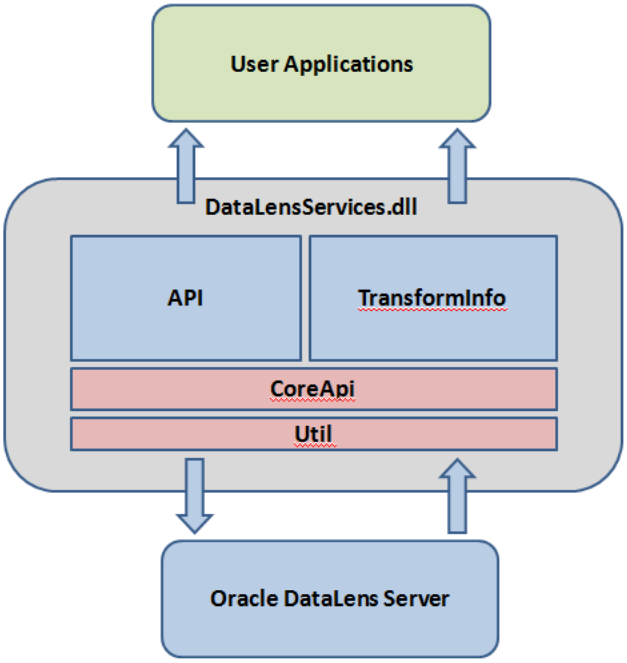
The EDQP .NET API uses only the data structures defined in the common type system of Microsoft .NET. In addition, it conforms to the common language infrastructure, which means that the EDQP .NET API object instances and data types may be exchanged between any applications (or libraries) written in any Microsoft .NET language. The EDQP .NET API can be integrated into the following:

- Microsoft .NET and other .NET languages such as C#, C++, F#
- Microsoft Office applications, such as Excel and Access
- Microsoft Active Server Pages (ASP) .NET Applications
- HTML pages and Internet Information Services (IIS) servers

This section introduces the EDQP .NET API components that work together to provide Microsoft .NET data access to Oracle DataLens Servers and describes how they relate to each other.

1.1 Overview

The EDQP .NET API package contains the `DataLensServices.dll` and that is comprised of API classes, objects, and functions for you to use in your applications to communicate with your Oracle DataLens Servers. It can be used on systems running the Microsoft Windows operating system.



1.2 EDQP .NET API Assembly

The `DataLensServices.dll` assembly contains the following:

1.2.1 EDQP .NET API Component

The EDQP .NET API contains all the classes that are used to interact with the Oracle DataLens Server to allow you to do the following:

- authenticate with the Oracle DataLens Server
- connect to the entire Oracle DataLens Server topology
- obtain access to individual Oracle DataLens Servers
- obtain Data Service Application (DSA) information
- run DSA jobs

1.2.2 TransformInfo Component

The TransformInfo component contains metadata classes that contain the objects with information from EDQP .NET API requests that include:

- DSA information
- data lens server group information
- user information
- metadata about transformed data records and fields from DSA jobs

1.2.3 CoreApi and Util Component

The CoreApi and Util component cannot be directly accessed because they handle the connections to the server via HTTP SOAP requests and are the utility packages used for network trafficking, error handling and XML DOM interaction for the various parts of EDQP.

2 Installing the EDQP .NET API

This section describes installation and configuration requirements for EDQP .NET API.

> **Note:** If you have installed the EDQP Services for Excel application on your Windows system, the EDQP .NET API is already installed and no further installation is necessary.

2.1 Before You Install

It is important to note the following about the EDQP .NET API:

- It was developed and compiled with the Microsoft Visual Studio 2010 release.
- All libraries and software necessary to this API are included as part of the Microsoft .NET system libraries; there are no third-party libraries used by the API so it can be used:
  - to access the Oracle DataLens Server from a Microsoft ASP or .NET application.
  - to access the Oracle DataLens Server from a Web Browser using client-side scripting such as VBScript, Jscript, or JavaScript.
  - to be embedded in applications, such as Microsoft Excel and Access using .NET integration or VBA scripting.
  - from any application that supports the Microsoft .NET interface.

2.2 Installation Prerequisites

You must ensure that the Microsoft .NET Framework 2 is installed on your Windows-based system. You can obtain the installation instructions and the download file from the Microsoft website at

`http://msdn.microsoft.com/netframework`

2.3 Installing the EDQP .NET API

The EDQP .NET API is provided as part of your Oracle DataLens Server installation, though it is not automatically loaded onto your system for you. This means that you need to register it on your system for use by your Microsoft ASP or Office applications.

To register the appropriate dynamic link libraries (DLLs) on your Windows system:

1. Open a Windows Command Prompt (`cmd.exe`) in the directory you installed the Microsoft .NET Framework 2 as an administrator user.
2. Register the .NET assembly by entering:

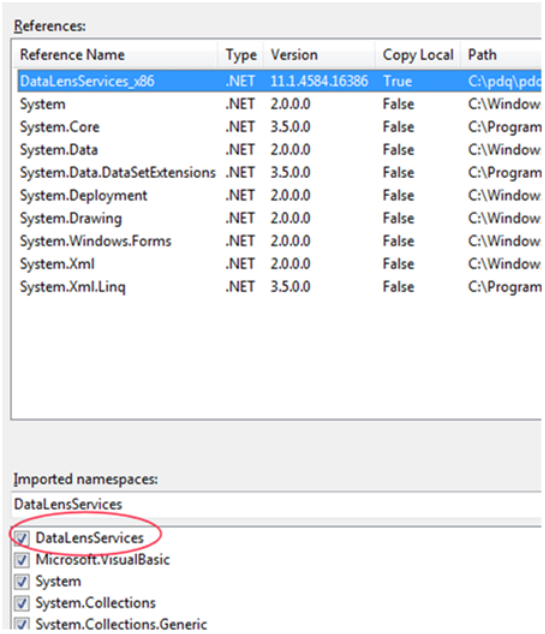
   ```
   regasm DataLensServices.dll
   ```

   A confirmation message is displayed that indicates the success of the DLL registration.
3. Add the `DataLensServices.dll` to Global Assembly Cache by entering:

   ```
   gacutil /i DataLensServices.dll
   ```
4. Close the Command Prompt window.

2.4 Using the EDQP .NET API with Microsoft Visual Studio

If you want to use the EDQP .NET API with Microsoft Visual Studio 2008 or 2010, you must add a reference to it in your project. In the following example, the'DataLensServices_x86' reference added is for use with 32-bit applications:



If you want to interface with 64-bit applications, then you must add a reference to 'DataLensServices_x64' in the same manner.

Either reference sets the `DataLensServices` namespace so that all the objects that are created in your source do not need to be prefixed with a `DataLensServices`. The examples in this document are formatted in this manner.

3 EDQP .NET API Objects

The DSA interface and the associated job interface in the EDQP .NET API provide the highest level of abstraction from the low-level processing, minimizing the amount of code that must change when modifying the DSAs and data lenses.

The EDQP .NET main objects that you can use in the API are:

- **`ApiError`**

  This error object is inherited from all API objects. The status of any object can be checked by accessing the members of this class.

- **`Topology`**

  This top level object contains information on all the server groups in the topology. It contains a collection of `TopologySession` and `TopologyServer` objects based on the currently active Server Group as follows:

  - `TopologyServer`—-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 `JobStep` objects:

TransformedRecord	A collection of the result data from the DSA job.
TransformedField	A collection of the individual fields in each record.

4 Developing with the EDQP .NET API

This section describes how to use the EDQP .NET API when developing your applications.

4.1 Using the Topology Object

The Topology object has information on the entire EDQP Topology, which includes all server groups (pods) and all Oracle DataLens Servers contained within each pod. When a pod has been selected for processing within a topology, then a TopologyServer object can be obtained from that particular pod. This server object can be used directly to run DSA jobs.

There are three steps to activating a TopologySystem object before it is ready for production use:

1. Log on to the Oracle DataLens Administration Server in your topology.
2. Activate the Server Group that you want to use for DSA job processing.
3. Get an active server to use for processing a job.

The following sections provide examples of how to use the Topology object.

4.1.1 TopologySession Object

The TopologySession object can be used as follows:

```
Public g_oTopologySession As TopologySession = New TopologySession()
```

The following sections describe an example method of creating a Topology Server object.

4.1.1.1 Logging On to the Topology

Use the server name and port with a valid user name and password to log on to your Oracle DataLens Server Topology as in the following example:

```
        bLoggedIn = g_oTopologySession.SessionLogon(sServerName, sServerPort,
sUsername, sPassword)
        If bLoggedIn = True Then
            ' Success!
        Else
        ' Failure
            MsgBox(g_oTopologySession.errorMsgTerse)
        End If
```

This example uses the error message from the TopologySession object. All the objects in the EDQP .NET API are derived from the ApiError class and use these error status fields.

4.1.1.2 Determining Available Server Groups

In a production environment, the server pod types can be checked and you can automate the activation of a server group based on the type. So if you had 4 server groups, then you would select the server group that is a production group and set that

as the active pod. This avoids the need for the name of the server group to be included in the software code in case it should change in the future. For example:

```
Dim oInfoPods As DataLensServices.InfoPods = g_oTopologySession.getPods()
        If oInfoPods.isDlsError Then
            MsgBox(oInfoPods.errorMsgTerse)
            Exit Sub
        End If
        Dim colPods As List(Of InfoPod) = oInfoPods.getPods()
        Dim pod As DataLensServices.InfoPod
        For Each pod In colPods
            currPodId = CStr(pod.id)
            currPodName = pod.Name
            currPodType = pod.podAreaType
        Next
```

4.1.1.3 Activating a Server Group

The topology may have only an Oracle DataLens Administration Server, or it may have any combination of Production, QA or Development Server Groups. A server group must be activated before any DSA jobs can be run. In a production environment, the server group can be a parameter in the end-user application, rather than retrieving a list and selecting the group of interest. For example:

```
        If g_oTopologySession.setActivePod(currPodId) Then
            'successfully set
        Else
            MsgBox("Error setting " + currPodName + " as current Server Group"
        End If
```

4.1.1.4 Checking User Permissions

Your client applications must ensure that the user you used to log in to the topology has the correct permissions to run DSAs in a particular server group. The `TopologySession` object is used to do this by beginning with an activated `TopologySession` object as in the following example:

```
 ' Check if the user has rights to run this job.
    If g_oTopologySession.canRunDSA Then
        RichTextBox_job.AppendText("Starting User Test at " + CStr(DateTime.Now) +
vbCrLf)
    Else
        MsgBox("User " + g_oTopologySession.InfoUser.name + " Does not have
capability to run in this Server Group")
        Return
    End If
```

For more information about users, see *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Administration Guide.*

4.1.1.5 Obtaining a Server for Processing Jobs

All the servers in the currently active server group are stored as `TopologyServer` objects in the `TopologySession` object. If there multiple servers in the server group, then each call to the `getServer` function returns a different server in the server group using a round-robin approach. This approach helps distribute the processing load across all the servers in a server group. These server objects are cached in the `TopologySession` object so that there is no performance penalty with distributing the processing load among all the servers in the pod.

Use the following standard method of the TopologySession object to get a topology server:

```
g_oTopologyServer = g_oTopologySession.getServer()
If g_oTopologyServer.isDlsError Then
    MsgBox("Error: " & g_oTopologyServer.errorMsg & vbCrLf)
    Exit Sub
End If
```

4.2 Using the TopologyServers Object to Run DSA Jobs

The TopologyServers object is used to perform DSA jobs and other actions. Use of this object is dependent on the creation of a TopologySession object as described in Section 4.1, "Using the Topology Object".

```
Public g_oTopologyServer As TopologyServer = New TopologyServer()
```

When a DSA job is run against a particular Oracle DataLens Server, the DSA is then always run on that particular server. The processing of the Transform Maps and data lenses can run 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, which automatically provides load balancing. For information about configuring each server to run particular types of maps or data lenses, see *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Administration Guide*.

4.2.1 Running a DSA Job

When you have a topology server obtained using the TopologySession object, you can use this in conjunction with a job object to run a DSA job. The following example creates a job object and assigns it to the obtained server in your TopologySession object:

```
' Create the Job object
Dim oJob As Job = New Job
oJob.topologyServer = g_oTopologyServer
```

Next, you set the basic options for the DSA job as in this example:

```
' Set all the basic job data
Dim sPriority As String = "High"
Dim sDescription As String = "Process Acme Product Data"
Dim sWorkflow As String = "myDSA"
Dim sRuntimeLocale As String = "en_US"
```

The input data is a collection of input lines, each with a collection of input data fields that must be provided to the DSA job as in this example:

```
  Dim rec1 As System.Collections.ArrayList = New System.Collections.ArrayList
  Dim m_cInputCol As System.Collections.ArrayList = New
System.Collections.ArrayList
  Rec1.add() = "17
  Rec1.add() = "resistor, 10 ohm, 20w, 25% 5 v"
   m_cInputCol.Add(rec1)
```

Once you have created the job object, set the job options, and provided the input data, you run the job as in this example:

```
'Run the Job
Dim lJobId As Long
' Use the Collections Interface
```

```
      lJobId = oJob.start(sPriority, sDescription, sWorkflow, m_cInputCol,
sRuntimeLocale)
```

You must continue to check the status of the job until it has completed so that you can retrieve the results if the job successfully runs as in this example:

```
' Synchronously get the results;
' The function call will not return till the data is ready
Dim oTransformedRecords As TransformedRecords
oTransformedRecords = oJob.getResultData(lJobId, True)
If oTransformedRecords.isDlsError Then
     MsgBox(oTransformedRecords.errorMsgTerse)
End If
```

Finally, you iterate through the collection of returned data from your DSA job as in this example:

```
'Iterate through the collection of transformed records
' and the collection of transformed fields
Dim sField As String
For Each oTransformedRecord In oTransformedRecords.Items
    Dim oTransformedField As TransformedField
    For Each oTransformedField In oTransformedRecord.Items
        sField = oTransformedField.value
        textBox.AppendText(sField + vbTab)
    Next
    textBox.AppendText(vbCrLf)
Next
```

4.2.2 Manually Creating a Server Object

You can manually create a server object as a shortcut to creating a `TopologyServer` object; this does *not* require an activated `TopologySession` object to create the `TopologyServer` object. Additionally, this method bypasses the topology login, server group selection, and the round-robin accessing of the topology servers.

This is useful in an environment where you want to re-instantiate these classes each time there is a call, rather than keeping a `TopologySession` object in memory.

An example of this manual creation is as follows:

```
g_oTopologyServer = g_oTopologySession.createDataServer(serverName.Text,
                                                         serverPort.Text,
                                                         serverUser.Text)
If g_oTopologyServer.isDlsError Then
      'oops
End If
```

When manually creating a server object, there are no checks to ensure that the server name and port are correct or that the user is a valid user in the topology.

This manually created `TopologyServer` object can be used to run jobs the same as a `TopologyServer` object created with the `TopologySession` standard method.

4.2.3 Using Optional Job Parameters

You can set the following optional parameters for use when running a job:

- Email output from a job.
- FTP output from a job.

- Filtering out control characters from the input data prior to submitting the job.
- Changing the separator character. This is used internally between DSA job steps during processing and is used to separate output data fields for file, email, or FTP output. The default is the TAB character.

Examples of these settings are as follows:

```
If outputEmail.Checked Then
    oJob.email = outputEmail.Text
ElseIf outputFTP.Checked Then
    oJob.ftp = outputFTP.Text
End If
oJob.filterData = True
oJob.separatorChar = "|"
```

You can set the job priority using the following valid values:

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

4.2.4 Running Jobs Using Data from a File

This is a similar call to the `ojob.start()` object previously described. This call, `oJob.startFile`, uses 3 additional parameters as follows:

- `filePath` - The path to the input text file.
- `RTOutput` flag - if the results of the job need to be retrieved.
- The final parameter (`"\tmp"` in the example) sets the location of where output data files are written. This is only used if the `RTOutput` flag is set to `True`.

An example of how `oJob.startFile` is as follows:

```
lJobId = oJob.startFile(sPriority, sDescription, sWorkflow, filePath,
sRuntimeLocale, isRTOutput, "\tmp")
```

4.2.5 Retrieving Job Results Asynchronously

This is useful if you want to process a large amount of data with the .NET API and do not want your program to hang while waiting for the job results to be ready.

Use the same call as in Section 4.2.1, "Running a DSA Job" (`oJob.getResultData(jobId, syncFlag)`) though you set the Synchronous flag to `True`.

An example of a simple way to check when the job has completed is as follows:

```
' Asynchronously get the results
Do
  oTransformedRecords = oJob.getResultData(lJobId, False)
  If oTransformedRecords.isDlsError Then
      If oTransformedRecords.errorCode = oJob.JOB_NOT_COMPLETED Then
          System.Threading.Thread.Sleep(5 * 1000) ' Sleep for 5 seconds
      Else
          MsgBox(oTransformedRecords.errorMsgTerse)
          Exit Sub
      End If
  End If
```

```
Loop While oTransformedRecords.errorCode = oJob.JOB_NOT_COMPLETED
```

4.2.6 Debugging Jobs

There is no client-side debugging toggle to review the actual HTTP request and response SOAP calls. However, there is a better way to look at this data in the development stage of a project. You turn on Server Tracing on your Oracle DataLens Development Server, then the output of this is written into the log file. The log file will contain the actual requests from the .NET API with the data and the DSA job processing server response from the Oracle DataLens Server.

For information about toggling on server-side tracing, see *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Administration Guide*.

4.2.7 Running DSA Jobs Using a Database

You can use a database query as input to your DSA jobs by replacing the `System.Collections.ArrayList` parameter with the `arrQueryParameterList` parameter.

To set the parameters on the job object prior to this call in the same manner as described Section 4.1, "Using the Topology Object", the following example could be used:

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

' Create the Job object
 Dim oJob As Job = New Job
 oJob.topologyServer = g_oTopologyServer

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

You can run the job in real-time (synchronously) as in this example:

```
' Synchronously get the results;
' The function call will not return till the data is ready
Dim oTransformedRecords As TransformedRecords

oTransformedRecords = oJob.getResultData(lJobId, True)
If oTransformedRecords.IsError Then
    MsgBox (oTransformedRecords.errorMsg)
    GoTo exception
End If
```

Alternatively, you can run the job in the background (asynchronously) as in this example:

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

```
      End If
   Loop While oTransformedRecords.errorCode = oJob.JOB_NOT_COMPLETED
```

4.2.8 Retrieving Multiple DSA Output Step Results

The transformed DSA job results can be retrieved directly from the job object by using named DSA output steps. This is useful for integrating a Transform Map process into an application where there are multiple outputs to be used for different purposes within the application. For example:

```
   ' Synchronously get the results (Note: if the flag is False, then the call is
asynchronous)
   oTransformedRecords = oJob.getResultStepData(lJobId, sStepName, True)
   If oTransformedRecords.IsError Then
       MsgBox (oTransformedRecords.errorMsg)
       GoTo exception
   End If
```

4.2.9 Retrieving Individual DSA Output Step Results

The transformed DSA job results can be retrieved directly from the individual DSA output steps using `JobStep`. This gives you control of the job at the individual DSA output step level. This is useful for an application where the actual names of the output steps are not known at run-time. For example:

```
'Retrieve result sets for all output steps
Dim oJobStep As JobStep
Dim oTransformedRecords As TransformedRecords
For Each oJobStep In oJob.steps
    If oJobStep.isOutputStep Then
        RichTextBox.AppendText(vbCrLf + "Output Step: " + oJobStep.name + vbCrLf)
        If oJobStep.isCompleted Then
            oTransformedRecords = oJobStep.getResults(g_oTopologyServer, True)
            If oTransformedRecords.isDlsError Then
                MsgBox(oTransformedRecords.errorMsgTerse)
                Exit Sub
            End If
            Dim oTransformedRecord As TransformedRecord
            For Each oTransformedRecord In oTransformedRecords.Items
                Dim colFields As System.Collections.ArrayList =
oTransformedRecord.Items
                Dim sFields As String = ""
                Dim oField As Object
                For Each oField In oTransformedRecord.Items
                    sFields = sFields + oField.Value + vbTab
                Next
                RichTextBox.AppendText(sFields + vbCrLf)
           Next
        End If
    Else
        ' No need to output the non-output step names
        'RichTextBox_jobStep.AppendText("Process Step: " + oJobStep.name + vbCrLf)
    End If
Next
```

4.3 Retrieving Server Information

This section describes some examples of how to retrieve information about your Oracle DataLens Server and the DSAs on the server.

4.3.1 Listing Server Job Information

The following example lists all the jobs on a particular server group:

```
Dim oJob As Job
Dim oJobs As Jobs
Dim bListAllServers As Boolean
Dim sinceSecs As Long
' use 0 to list all the jobs. This will list all jobs in the last hour
sinceSecs = 3600
' The TopologyServer object will not refresh the job list once it has been created
for performance.
' In order to get a fresh list, we must reset the Topology Server object.
g_oTopologyServer.resetJobs()
Dim bRefresh As Boolean = False
If CheckBox_topSrv_userJobs.Checked Then
    Dim user As String = g_oTopologyServer.user
    oJobs = g_oTopologyServer.getJobsBySubmitter(user, bListAllServers, sinceSecs,
bRefresh)
Else
    oJobs = g_oTopologyServer.getJobs(bListAllServers, sinceSecs, bRefresh)
End If

If oJobs.isDlsError Then
    RichTextBox.AppendText("Error loading TransformSystem... " & oJobs.errorMsg +
vbCrLf)
    Exit Sub
End If
Dim sUserName As String = g_oTopologyServer.user
Dim sJobInfo As String = ""
For Each oJob In oJobs.Jobs
    sJobInfo = sJobInfo + displaySingleJob(oJob)
Next
```

You can list the individual job information as in this example:

```
Private Function displaySingleJob(ByVal oJob As Job) As String
   Dim m_tmpStatus As String
   m_tmpStatus = vbCrLf + "Job Id: " & CStr(oJob.jobId) + vbTab
   m_tmpStatus += CStr(oJob.priority) + vbTab
   m_tmpStatus += oJob.getStatusDesc(oJob.status) + vbTab
   m_tmpStatus += oJob.runtimeLocale + vbTab
   m_tmpStatus += oJob.createdBy + vbTab
   m_tmpStatus += oJob.description + vbTab
   m_tmpStatus += oJob.definition + vbTab
   m_tmpStatus += oJob.startTime + vbTab
   m_tmpStatus += oJob.endTime + vbTab
   m_tmpStatus += oJob.server + vbTab
   m_tmpStatus += " Completed: " + CStr(oJob.isCompleted) + vbTab
   m_tmpStatus += " Cancelled: " + CStr(oJob.isCanceled) + vbTab
   m_tmpStatus += " Results retrieved: " + CStr(oJob.isResultsRetrieved) + vbTab
   m_tmpStatus += " Error Msg: " + oJob.errorMsg + vbTab
   displaySingleJob = m_tmpStatus
End Function
```

4.3.2 Listing Information About All DSAs on the Server

You can list information about the DSA jobs that have been submitted to an Oracle DataLens Server using the `ProcessMaps` object.

The following example lists all of the DSAs that are available to run in the current server group:

```
Dim oProcessMaps As Dsas = g_oTopologyServer.getDsas
If oProcessMaps.isDlsError Then
    MsgBox(oProcessMaps.errorMsgTerse)
    Exit Sub
End If
Dim oProcessMap As Dsa
For Each oProcessMap In oProcessMaps.Items
    RichTextBox.AppendText(oProcessMap.name + vbCrLf)
Next
```

4.3.3 Listing Information for One DSA

You can list the information about a particular DSA on your server as in the following example:

```
Dim oPMap As Dsa = g_oTopologyServer.getDsa(SvrDsa.Text)
If oPMap.isDlsError Then
    MsgBox(oPMap.errorMsgTerse)
    Exit Sub
End If
RichTextBox.AppendText("Name: " + oPMap.name + vbCrLf)
RichTextBox.AppendText("Desc: " + oPMap.description + vbCrLf)
RichTextBox.AppendText("Is Db Input: " + CStr(oPMap.hasDbDataSource) + vbCrLf)
RichTextBox.AppendText("Num Db Parms: " + CStr(oPMap.dbParameterCount) + vbCrLf)
'Input Steps
RichTextBox.AppendText("Input Steps: " + vbCrLf)
Dim cInput As System.Collections.ArrayList = oPMap.inputSteps
Dim step1 As DsaStep
' the step is a jobStep object
For Each step1 In cInput
    RichTextBox.AppendText(vbTab + step1.name + vbCrLf)
    RichTextBox.AppendText(vbTab + vbTab + "Description: " + step1.description +
vbCrLf)
    RichTextBox.AppendText(vbTab + vbTab + "Data Fields:" + vbCrLf)
    Dim cInputFields As List(Of String) = step1.inputFields
    Dim sField As Object
    For Each sField In cInputFields
        RichTextBox.AppendText(vbTab + vbTab + vbTab + sField + vbCrLf)
    Next
 Next
 'Output Steps
 RichTextBox.AppendText("Output Steps: " + vbCrLf)
 Dim cOutput As System.Collections.ArrayList = oPMap.outputSteps
' the step is a jobStep object
For Each step1 In cOutput
    RichTextBox.AppendText(vbTab + step1.name + vbCrLf)
    RichTextBox.AppendText(vbTab + vbTab + "Description: " + step1.description +
vbCrLf)
    RichTextBox.AppendText(vbTab + vbTab + "Dont Return Results: " +
                                                CStr(step1.dontReturnResults) +
 vbCrLf)
    RichTextBox.AppendText(vbTab + vbTab + "Data Fields:" + vbCrLf)
    Dim cOutputFields As List(Of String) = step1.outputFields
    Dim sOutField As Object
    For Each sOutField In cOutputFields
        RichTextBox.AppendText(vbTab + vbTab + vbTab + sOutField + vbCrLf)
```

```
    Next
Next
'Db Connections
RichTextBox.AppendText("Db Connections: " + vbCrLf)
Dim cConnections As System.Collections.ArrayList = oPMap.dbConnections
Dim connection As Object
For Each connection In cConnections
    RichTextBox.AppendText(vbTab + connection + vbCrLf)
Next
```

4.4 Handling Errors

You use the `ApiError` object to monitor error status and display any errors that occur.

The fields that are available to check the error status in all of the EDQP .NET API objects are:

- `isDlsError`—A flag that checks the error status of any .EDQP .NET API objects.
- `errorMsg`—A full error message, often with SOAP fragments and stack trace.
- `errorMsgTerse`—A shorter error message, suitable for display to users.
- `errorCode`—An integer error code that can be use to check the fault state.

In the example in Section 4.1.1.1, "Logging On to the Topology", if an error occurs then a standard Microsoft Windows message box would be displayed because the following line of code was included:

```
MsgBox("User " + g_oTopologySession.InfoUser.name + " Does not have capability to
run in this Server Group")
```

4.4.1 Checking Client-Side Exceptions

All of the EDQP .NET API objects keep track of the error state. This includes the API objects and the objects that return collections of data. You can add the following check to retrieve the client-side exceptions:

```
If oDlsObject.IsError Then
' oSilverCreekObject.errorMsg
End If
```

4.4.2 Checking Server-Side Exceptions

Server-side errors are propagated back to the client where they can be monitored and reported on.

4.4.3 Checking Server-Side Log Messages

All server-side error messages encountered during all processing are displayed in the Oracle DataLens Administration page. For detailed information, see *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Administration Guide.*

5 Audience

This document is intended for programmers who are developing applications to access an Oracle DataLens Server using the EDQP .NET API.

To use this document, you must be familiar with Microsoft .NET Framework classes and ADO.NET and have a working knowledge of application programming using Microsoft C#, Visual Basic .NET, or another .NET language.

Although the examples in the documentation and the samples in the sample directory are written in Microsoft .NET, developers can use these examples as models for writing code in other .NET languages.

6 Related Documents

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

- The *Oracle Enterprise Data Quality for Product Data Getting Started Guide* provides information about how to get started with EDQP.
- The *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Installation Guide* provides detailed Oracle DataLens Server installation instructions.
- The *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Administration Guide* provides information about managing an Oracle DataLens Server including users and user roles.

See the latest version of this and all documents in the Oracle Enterprise Data Quality for Product Data Documentation Web site at

http://docs.oracle.com/cd/E35636_01/index.htm

For additional information about Microsoft .NET, see:

http://msdn.microsoft.com/netframework

7 Documentation Accessibility

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

Access to Oracle Support

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

Oracle Enterprise Data Quality for Product Data .NET API Interface Guide, Release 5.6.2
E48206-01