Volume

1

Exodus

Map⏵Document⏵Migrate⏵JSON & Tables

User Guide

Exodus

User Guide

© IHS Markit

Developed By Christian Leigh

Table of Contents

[What can Exodus do? 1](#_Toc507665137)

[Requirements 2](#_Toc507665138)

[Terminology 3](#_Toc507665139)

[Application Anatomy 6](#_Toc507665140)

[Main Window 6](#_Toc507665141)

[Define Migration Window 7](#_Toc507665142)

[Manage JSON 8](#_Toc507665143)

[Getting Started 9](#_Toc507665144)

[SQL\*Loader 12](#_Toc507665145)

[EXAMPLE Loader control file 13](#_Toc507665146)

[Installation Instructions 14](#_Toc507665147)

[Production Systems 17](#_Toc507665150)

Chapter

1

What can Exodus do?

Learn what Exodus does and familiarize yourself with the basic terminology.

E

xodus is a mapping, documenting, and migration tool especially designed with Oracle PL/SQL developers in mind. If you are comfortable with Oracle PL/SQL then all your mapping and migration code can be constructed here without the need for non-Oracle external tools. There is no magic “black box” behavior. All of the auto-generated code (also in PL/SQL) is made available and viewable from within the tool, and because it’s in a language that seasoned Oracle developers are expert in, there will be no unexpected surprises with the technical approach.

It’s entirely possible to use Exodus to “*just”* document your mapping efforts (it will certainly be far less stressful and error prone than using a spreadsheet to capture mappings and relationships). With just a basic understanding of the tool tables you can write your own SQL reports to get information about how much you’ve mapped (in other words, report on SOURCE data that you have understood enough to express a connection to the target end state) in your migration.

Exodus can be used to do far more than documenting the migration. Exodus can form the core nucleus of your migration where all of the steps required to take data from a set of source tables to a set of target tables in multiple schemas can be defined and coded. Exodus can map table-to-table and table-to-JSON documents. You will be able to capture comments, questions, value translations, functions, and even SQL code to drive your migration.

There is an extensive run framework (including logging, introspection, and debugging instrumentation) to execute your migration to any level of concurrency that your hardware and licenses can support to maximize your compute resource.

# Requirements

🖳

Exodus is designed for Oracle systems. The mapping tool is developed using Oracle Forms 11g and above. You will need an Oracle Database (to contain your source data and to receive the target data (migrated data) and to store the metadata that the tool produces). You will also need one forms server running either locally on your PC or from a centralized server. The database that executes the migration can be hosted in the cloud or on premises. The cloud is an excellent way to boost migration performance because you can elastically “scale up” your compute requirement for the migration “heavy lifting”, and then “scale down” to the modest compute requirements required for day-to-day running.

Oracle corporation allow downloads of all the database tools, and middleware necessary to evaluate the software. Of course if you use this application for production systems you will need the appropriate licenses.

# Terminology

###### See Glossary…

###### The glossary towards the end of this manual will give a useful guide.

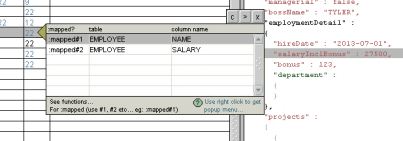
As with any system or methodology EXODUS has its own terminology. Although concepts like **SOURCE** and **TARGET** are universal, there are others that might not be. This section will discuss the key concepts to get you up to speed quickly.

Every migration has a **SOURCE** and a corresponding **TARGET**. In EXODUS the **SOURCE** is “staged data” that represents the starting data before any processing has been done. If your staged data comes from an external non-Oracle system, then the recommended way to get that data onto your system in the most cost effective (inexpensive) way is to use Oracle’s SQL\*Loader tool. There are two types of **TARGET.**  The storage **TARGET** is typically a table (after all, Oracle stores data in tables). **TARGETs** can also be JSON documents. A JSON document will ultimately become a value for a table column (sometimes described as a PAYLOAD) that holds the JSON document; often as a CLOB or VARCHAR2(32767). In EXODUS the **TARGETs** are always expressed as JSON (even tables – Tables appear as a JSON representation of a target table).

**SOURCE** is shown on the left hand side of the EXODUS tool window, and the **TARGET** on the right hand side of the main tool window {See: [Application Anatomy : Main Window](#_Main_Window)}.

**Map/Rules Group**: With any target table, or JSON document there might be more than one use case for mapping to the same target. A good example of this would be a target JSON document that has exactly the same shape, but each use case sources its values from 2 completely different tables (for completely separate migration steps), for example Historical Data and Current Data. Current data could be derived from source table A, but historical data could derive from source table B. They might end up in the same target table (or different target tables), the JSON payloads would be constructed differently because they are mapped differently (In other words the data is sourced differently). The migration definition specifies what **Map/Rules Group** to use (see EXODUS – Define Migration).

**Map & Un-Map**: “To **Map**” means to create a connection between the source table column and a target JSON line. “To **Un-Map**” means to break that connection. A target JSON line can be mapped to multiple source columns (even from multiple source columns from multiple tables – when more than one source table/column is used then it is known as a multi-mapping).

**Multi-Mapping** means to create a connection to more than one source column. **Multi-Mapped** targets can ref. columns in comment code by using # (i.e**.:mapped#1** , **:mapped#2** etc.)

**Stored Document / Stored Fragment / Table Document**: These are all **TARGET** documents. A **Stored Document** is a completely formed JSON document that will be used as a template (or “document by example”) for a mapping target. A **Stored Fragment** is also a mapping target, and is a piece of a JSON document that can be included into a **Stored Document** based upon some Boolean rule (note this “Boolean” is expressed as a ‘TRUE’ ‘FALSE’ VARCHAR2). A **Table Document** is the target table merely expressed as a JSON document (you can only map to column names in this document).

**Related Document**: A related document/line is a way of sharing any mapping made across related JSON documents. Once related, any change to a mapping is made simultaneously to the related mappings. For example you might have a document that has some boilerplate attribute values that are arrived at in the same way from the same source tables and rules, or maybe they’re static values supplied by a fixed value. By relating these target attributes you can alter them all with one edit instead of repeating the change in all the targets.

**Function Comment**: There are 7 types of comment that can be set against a mapping. Three of these comment types can contain code. Array ([ ]), Lookup (LK), and Function (FN). You are not compelled to put code in the comments; you can put anything you like. However if the function begins with an equals sign then it is considered code, a lookup must start with a SELECT or a WITH clause, if it doesn’t then it will just be a comment, and it won’t be considered code.

**Manage JSON**: Used to Add, Update, or Delete Target JSON “documents by example”. If you want to change a JSON document that already has mappings then the tool allows you to re-map either semi manually or automatically so that you don’t lose any of your work made in those mappings. The automatic remapping cannot easily distinguish between attributes that have repeating identical attribute names, so semi-manual re-mapping may be required in some instances, although even that is preferable to mapping from scratch. Table JSONs can also be automatically added.

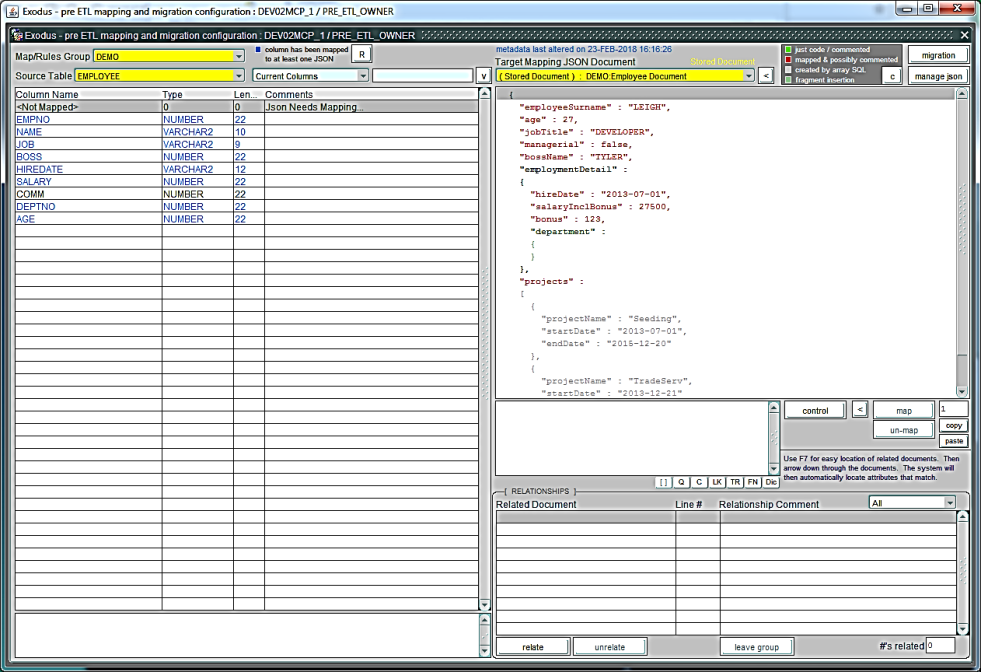
**Migration Grouping**: A way of grouping steps together for specific migrations. The launching of a migration requires that you specify which migration you mean to launch (in other words which Migration Group), and the processing of the migration will normally execute sequentially starting at step 1 of the **Migration Group** and continue to the specified end step. You can also configure **Migration Groups** to execute if certain error conditions arise (for example a **Migration Group** could be a set of steps necessary to return the target schema to a known state on the event of an error).

**Contexts**: There is ONLY one PRE\_ETL\_OWNER schema on an instance; this is where the metadata for all migrations are maintained, and defined. Your design may require more than one target schema for a given migration (A testing schema target, or a production target for example). One way to achieve this would be to have separate migration groups (and mapping rule/groups) for each use case. That would be unwieldy, error prone, and it would necessitate much copy pasting of behavior). Rather than “hard code” these **TARGET** values into your migration you can use ${} variables for substitutes. By doing this you can dynamically change these values for any migration at the point of execution by specifying the context in which you want the migration to run. One way to think of **Contexts** is as a grouping of substitution values.

# Application Anatomy

BELOW are the three main areas of the tool that allow the migration designers to configure the metadata.

## Main Window



**Map Rules Group**

**Search Option**

* All Tables
* Current Columns
* All Tables & Columns
* Targets/Comments/Migration

**Search Value**

Wild Card can be %

EG: TEST%A

**Target JSON**

Target can be JSON Doc. / Fragment or Table

**Colour Key**

JSON documents…

Mapped to Source: Red

**Define Migration Steps**

**Manage JSON**

Add, Modify, Delete

JSON & Remap

**Source Columns**

Double Click to view mapping usage. Blue lines are mapped lines.

**Source Table**

.

**JSON Document**

Attributes can be mapped, commented, or a combination of both.

**Relationship Management**

Relate Fragments

Relate Attributes

**Comment/Code**

Capture comments and *CODE*.

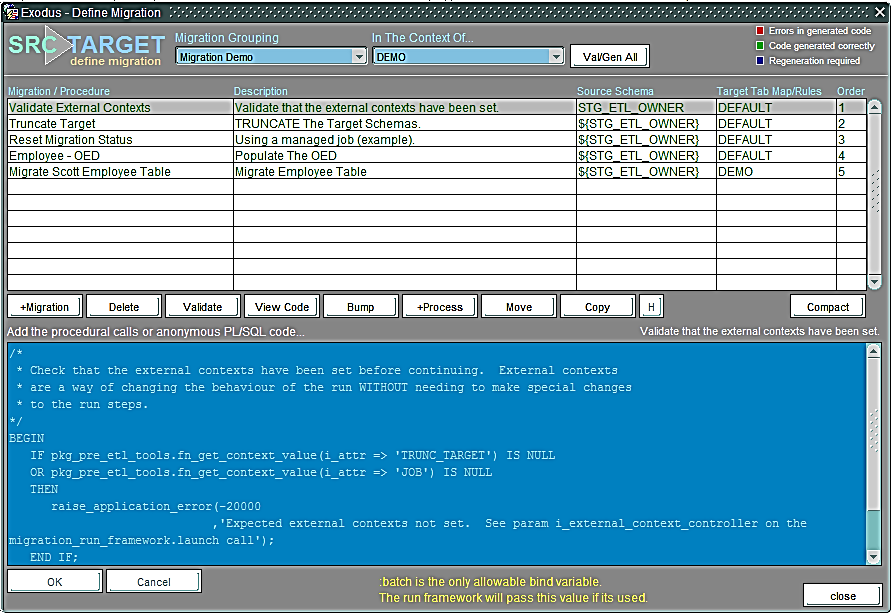
**Leave Relationship Group**

.

**Map / Un-Map**

## Define Migration Window

This is shown when the migration designer presses the migration button from the main window.



**Migration Steps**

All the steps for the chosen Migration Group.

**Migration Group**

Group your logically related migration steps together.

**Context**

The Context can affect the compilation status.

**Validate/Regen**

Compiles and validates the Migration Group.

**Status Key**

Red: Invalid

Green: Valid

Blue: Unknown

**Compact**

Compact the step order numbers to monotonically increasing values.

**Hint**

For auto generated migration steps. Useful for specifying optimiser hints. (Red if used)

**Copy**

Copy a migration step to within the same Migration Group, or to another Migration Group.

**Move**

Move a migration step to another group.

**Add a Process**

Add a manual PL/SQL step.

**Bump**

Increase the step number for the highlighted step and all those below it. Useful for making space to insert a new step.

**View Code**

View the code in a larger window. Useful to examine auto-generated code. The code view window allows convenient copying of the code.

**Validate**

Validates the highlighted step.

**\Delete**

Deletes the highlighted step.

**Add Migration Step**

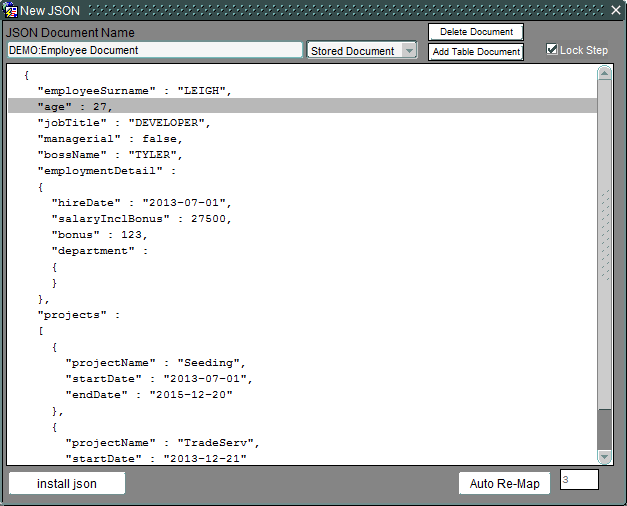
Adds a migration step.



## Manage JSON

This is shown when the migration designer presses the manage JON button.





**Install JSON**

Installs the JSON. Note that if you want to retain any mapping for modified JSON, then you will need to Re-Map BEFORE installing.

**Auto Re-Map**

Instead of manually arrowing down through the JSON, and pressing RETURN/ENTER to copy over the mappings from the target. Auto-Remap performs this task for you.

Use in conjunction with the Lock-Step.

**Add Table Document**

Add tables from schemas defined in the parameters table;

SELECT \*

FROM pre\_etl\_params a

WHERE a.param\_name = 'ACCESSIBLE\_SCHEMAS'

**Lock Step**

This needs to be set for Auto-Remap. When in Lock Step mode, navigation in this window is reflected in the target document too.

**Document Name**

The name of the document being added or amended.

**Type of document**

* Stored Document
* Table Document
* Stored Fragment

**Delete**

Delete the document.

**The JSON document**

Pressing RETURN or ENTER will cause the highlighted line to copy over the mappings and comments of the highlighted line in the target document.

Typically used with Lock-Step mode.

Chapter

2

# Getting Started

1. These getting started steps assume you have installed the EXODUS code for both the database and Oracle Forms tools. See installation instructions.
2. Load your source data into an Oracle database. See: [SQL\*Loader](#_SQL*Loader)

You need to get the SOURCE tables created in the STAGING schema.

You can name the STAGING schema whatever you like, although you may wish to use a naming convention. For example: PROD\_STAGING. This is so that you can also have a DEV\_STAGING and a QA\_STAGING for example. You can use Contextual Substitution Values in your migration code instead of real schema names. Using Substitution Values is a good way to avoid hard coding values that might need to change depending on the context under which you wish to run a migration.

1. Programmatically populate the EXODUS tool tables that store the Source Tables and Columns. You can do this either by writing directly to the tables (there are only 2 to consider; **pre\_etl\_db2\_columns** & **pre\_etl\_db2\_tables**), or you can use the utilities in PRE\_ETL\_OWNER). One advantage to populating the tables directly is that you can add table and column comments. The supplied procedures (shown below) don’t do that. Alternatively you can use the supplied procedures and then back-populate any comments using a script.

BEGIN

pkg\_pre\_etl\_tools.pr\_add\_schema\_to\_mapping\_tool

(i\_schema\_name => '<YOUR SOURCE SCHEMA>'

,i\_tableset\_name => '<A TAG TO NAME THE SET>'

,i\_delete\_existing\_in\_set\_bool => TRUE);

END;

/

Or

BEGIN

pkg\_pre\_etl\_tools.pr\_add\_table\_to\_mapping\_tool

(i\_owner\_name => '<THE SOURCE SCHEMA>'

,i\_table\_name => '<A TABLE NAME>'

,i\_tableset\_name => '<A TAG TO NAME THE SET>'

,i\_delete\_existing\_in\_set\_bool => TRUE);

END;

/

1. Start the EXODUS Mapping Tool.

An example URL might look like this:

<http://mypc:7001/forms/frmservlet?form=C:\forms\PRE_ETL_MAPPING.fmx&userid=pre_etl_owner/pre_etl_owner@migration_db&quiet=YES&separateFrame=True&background=NO&width=1265&height=1000&allowAlertClipboard=false&Logo=%22no%22>

Oracle Forms starts up a Java Applet that runs the tool.

1. If this is the first time you have used the tool then there will be no JSON targets. You will need to add a JSON target. Click the manage JSON button. 

If you are adding a JSON payload target (In other words a DOCUMENT that you want to migrate the SOURCE data to, then you need to paste (or type) in the document. These are often referred to as a “document by example”. For example from the DEMO:

{

"employeeSurname" : "LEIGH",

"age" : 27,

"jobTitle" : "DEVELOPER",

"managerial" : false,

"bossName" : "TYLER",

"employmentDetail" :

{

"hireDate" : "2013-07-01",

"salaryInclBonus" : 27500,

"bonus" : 123,

"department" :

{

}

},

"projects" :

[

{

"projectName" : "Seeding",

"startDate" : "2013-07-01",

"endDate" : "2015-12-20"

},

{

"projectName" : "TradeServ",

"startDate" : "2013-12-21"

}

]

}

You will need to give a name for the document. It’s suggested (but not necessary), that you define a naming standard for your documents. A good way would be to prefix the project name. For example “DEMO: Employee Document”. Set the document type as a **Stored Document**. Then press the load JSON button.



If you want to add a TARGET table (the table will be rendered as JSON), then you should use the Add Table Document button. You can define what schemas make their table definitions available in the PRE\_ETL\_PARAMS table (ACCESSIBLE\_SCHEMAS).

1. You will then be ready to either accept the document by pressing the **install JSON** button, or you can (if you have another document already loaded) re-map or copy the mappings from another document.
2. Once you have loaded your document you can map to it. Select a line in your JSON document, and then choose a SOURCE table/column. Press the map button. If you map to more than one source you will be given an option to start a multi-mapping, or overwrite the existing mapping (if any).
3. It is strongly recommend that you install the DEMO migration. The DEMO migration takes the classic SCOTT/TIGER database, and migrates it to a Target database in which the contents of the source database are represented as JSON documents.

Chapter

3

# SQL\*Loader

We recommend that you export data from any non-Oracle system in a text based flat file using extremely unique multi-character separators to delimit each field (see below). The staged data must have at least one extra column field; “MIGRATION\_BATCH\_ID” defined as a NUMBER. It’s further recommended (but not necessary) that the staged data have an additional column field of “MIGRATION\_STATUS”, defined as a VARCHAR2 (1), although this isn’t necessary for EXODUS to function for the purposes of the migration (but it will come in handy later when you being diagnosing any migration issues). This User Guide cannot be a substitute to familiarizing yourself with the SQL\*Loader documentation.

For example:

@BOR@1@DELIM@Apollo11@DELIM@21.5@DELIM@21/07/1969@DELIM@@EOR@

@BOR@2@DELIM@Apollo12@DELIM@31@DELIM@19/11/1969@DELIM@@EOR@

@BOR@3@DELIM@Apollo13@DELIM@@DELIM@15/04/1970@DELIM@@@EOR@

KEY

|  |  |
| --- | --- |
| @BOR@ | Denotes Beginning Of Record |
| @DELIM@ | Denotes Column Delimiter |
| @EOR@ | Denotes End Of Record |

Once processed through SQL\*Loader the staged table rows for the above might look something like this (notice the additional fields of MIGRATION\_BATCH\_ID, and MIGRATION\_STATUS) (You need to create your own stage tables – this can be easily scripted; see PRE\_ETL\_OWNER.PRE\_ETL\_DB2\_COLUMNS):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Surrogate\_ID** | **Mission** | **Hrs On Surface** | **Date** | **MIGRATION\_BATCH\_ID** | **MIGRATION\_STATUS** |
| 1 | Apollo11 | 21.5 | 21/07/1969 | 1 |  |
| 2 | Apollo12 | 31 | 19-11-1969 | 1 |  |
| 3 | Apollo13 |  | 15/04/1970 | 1 |  |

## EXAMPLE Loader control file

OPTIONS

(

DIRECT=TRUE

)

LOAD DATA

INFILE '/data/bulk/source\_file.txt' "str '@EOR@\r\n'"

APPEND

INTO TABLE STAGED\_APOLLO\_DATA

fields terminated by '@DELIM@' TRAILING NULLCOLS

(

BOR FILLER POSITION (1:5)

, SURROGATE\_ID "replace(:SURROGATE\_ID,chr(0),null) "

, MISSION char(8)

, SURFACE\_HOURS "replace(:SURFACE\_HOURS,chr(0),null) "

, EXPECTED\_DATE to\_date(replace(:EXPECTED\_DATE,chr(0),null) , CASE WHEN INSTR(:EXPECTED\_DATE,'/') > 0 THEN 'MM/DD/YYYY' WHEN INSTR(:EXPECTED\_DATE,'-') > 0 then 'YYYY-MM-DD' end )"

, MIGRATION\_BATCH\_ID "1"

, MIGRATION\_STATUS "null"

)

Chapter

4

# Installation Instructions

ORACLE Forms can be fiddly to install, but the payoff is a fantastic tool to perform migration, mapping and other documentation. I won’t describe the steps necessary to install an Oracle Database; the chances are that if you are reading this document then you already are adept at that, or have an Oracle Database already available to host the EXODUS tools. However Oracle Forms can present its own challenges.

These instructions will assume you will be deploying Oracle Forms 11g on Windows10 (64bit).

On your PC create a directory called: C:\InstallFormsAndWeblogic

Note that this directory has no special characters or spaces, and it’s located just off of root on the C: drive. Don’t deviate from this suggestion. You may be lucky in your installation, you might not be. It’s best to avoid the frustrations of doing what might appear at face value to be the “right thing”, but as a result being unable to install the software correctly. These steps also assume a system where no Java has previously been installed. If you have other Java Homes you might need to uninstall them, and then re-install the later.

These steps assume that you are installing on a PC for evaluation purposes, your organization will likely have different standards for installation. It is suggested that you try the deployment in a virtual machine with a virgin installation of Windows 10 (64bit).

Pre Installation Steps

Locate these files (from Oracle) and download them into that directory:-

* wls1036\_generic.jar
* ofm\_frmrpts\_win\_11.1.2.2.0\_64\_disk1\_1of2.zip
* ofm\_frmrpts\_win\_11.1.2.2.0\_64\_disk1\_2of2.zip
* jdk-7u80-windows-x64.exe

Unpack the zips into the same directory (thereby creating 5 folders):-

* C:\InstallFormsAndWeblogic\Disk1
* C:\InstallFormsAndWeblogic\Disk2
* C:\InstallFormsAndWeblogic\Disk3
* C:\InstallFormsAndWeblogic\Disk4
* C:\InstallFormsAndWeblogic\Disk5

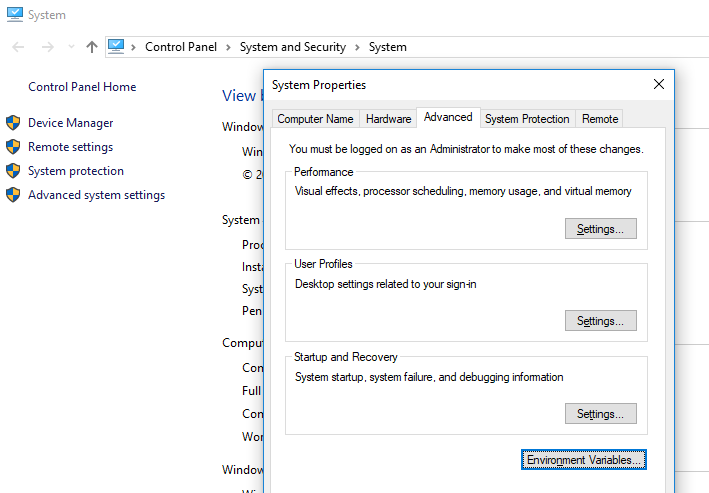
You are now ready to begin the installation.

1. Change these 2 environment variables to have the value: **C:\temp**

**TEMP=C:\temp**

**TMP=C:\temp**

You can find these values in here…



1. Now install the JDK:

Simply double click the file you download during the Pre Installation steps above:-

jdk-7u80-windows-x64.exe

1. Add a new environment variable (located in the same place as described above). (Browse to the location in case its different to the one shown).

**java\_home=C:\Program Files\Java\jdk1.7.0\_80**

1. Modify your path to put the Java bin directory at the very beginning of your path. I.e. your PATH environment variable MUST start with the location of the bin directory of your JAVA installation for example:: **C:\Program Files\Java\jdk1.7.0\_80\bin** (and then a semi-colon separator and the rest of your previous PATH variable).
2. Now open a command prompt.

**cd** to the location you have the installation files.

i.e. C:\InstallFormsAndWeblogic

1. Type : java -jar wls1036\_generic.jar

(press return).

1. Follow the prompts for the installation. When it arrives at the step where it asks about: “I wish to receive security updates via My Oracle Support”. Leave the checkbox ticked. But don’t supply an email address. Then press **NEXT**, and select **YES** in the following 2 dialog boxes.
2. Another window will appear (with a heading of Connection Failed).

Check the box next to the statement: “I wish to remain uninformed of security issues in my configuration or this machine has no Internet access”. Press **CONTINUE**.

1. Leave the options as default. Typical installation. All the way through until the window shows as Installation Complete.

Uncheck the “Run Quickstart” option, and press **DONE**.

1. You may see a warning in the command window about “Could not open/create prefs…”. You can ignore this.
2. Using windows Explorer navigate to the:

C:\InstallFormsAndWeblogic\Disk1

1. Right click on the Setup.exe file and select “**RUN AS ADMINISTRATOR”.**
2. When the install wizard appears; hit **NEXT** and choose the option to Skip Software Updates.
3. Continue through – choosing the default options (until you get to the section for Pre-Requisite checks).
4. When the section for Pre-Requisite checks appears IGNORE the error about system certification (if it appears). Press **CONTINUE**.
5. Chose **Configure for Development** (this is not the DEFAULT) so you must consciously choose that option.
6. In the Create Domain section: Choose a password and make a note of it, as you will need this later.
7. Uncheck the box for “I wish to receive security updates via My Oracle Support”. Press **NEXT** and acknowledge the message about not providing an email address.
8. In the Configure Environment option select ALL the checkboxes.
9. Now chose the defaults all the way through and press INSTALL when the button appears.
10. Now the installation will take place, and it can take a long time, at least 30 Minutes. Make a cup of tea.
11. If the Windows firewall dialog appears: Check the box for “Private Networks, such as my home or work network”. Leave the other checkboxes in that window as they are (probably checked). Press “**ALLOW ACCESS**”
12. You should have had no errors during the installation. If you do something wasn’t configured correctly. Please re-read these instructions.

## Production Systems

For your production system you only need to deploy the PRE\_ETL\_OWNER schema. You don’t need the migration definition tool to actually run a migration, *but* ***you do need*** *the metadata it creates*. You can copy metadata from one PRE\_ETL\_OWNER schema to another. There is a script provided to do this. The script uses the EXP/IMP tools Oracle already provides to achieve this:

export\_metadata\_from\_dev.sh

The export script creates a file called: dev\_meta\_export.dmp

You can import the “.dmp” into another installation of the EXODUS (PRE\_ETL\_OWNER) schema by using:

import\_metadata\_into\_target\_deploy\_db.sh

The import script performs any necessary bootstrapping. For example you might find that you need to run some initializers to create local tables that are required for the migration. Without the import running these “special” steps the migration steps you have defined might not be able to compile. Initializers are actually just other migration groupings that create empty work tables.