HPCC Systems® Data Tutorial

Boca Raton Documentation Team

HPCC Systems® Data Tutorial

Boca Raton Documentation Team

We welcome your comments and feedback about this document via email to <docfeedback@hpccsystems.com>

Please include **Documentation Feedback** in the subject line and reference the document name, page numbers, and current Version Number in the text of the message.

2023 Version 9.0.12-1

Introduction

The ECL Development Process

This tutorial provides a walk-through of the development process, from beginning to end, and is designed to be an introduction to working with data on any HPCC Systems platform. HPCC[1]. We will write code in ECL[2]to process our data and query it.

This tutorial assumes:

- You have a running HPCC Systems platform. This can be a single or multinode HPCC Systems platform deployment.

You have the ECL IDE[3] installed and configured

In this tutorial, we will:

- Download a raw data file

  There are links to data file available at https://hpccsystems.com/training/documentation/learning-ecl

  The download is approximately 30 MB (compressed) and is available in either ZIP or .tar.gz format. Choose the appropriate link.

- Spray the file to a Data Refinery cluster HPCC Systems clusters "spray" data into file parts on each node.

  A *spray* or *import* is the relocation of a data file from one location to an HPCC Systems cluster. The term spray was adopted due to the nature of the file movement -- the file is partitioned across all nodes within a cluster.

- Examine the data and determine the pre-processing we need to perform
- Pre-process the data to produce a new data file
- Determine the types of queries we want
- Create the queries
- Test the queries
- Deploy them to a Rapid Data Delivery Engine (RDDE) cluster, also know as a Roxie cluster.

[1] **H**igh **P**erformance **C**omputing **C**luster (HPCC) Systems is a massively parallel processing computing platform that solves Big Data problems. See http://www.hpccsystems.com/Why-HPCC/How-it-works for more details.
[2] **E**nterprise **C**ontrol **L**anguage (ECL) is a declarative, data centric programming language used to manage all aspects of the massive data joins, sorts, and builds that truly differentiate HPCC Systems (High Performance Computing Cluster) from other technologies in its ability to provide flexible data analysis on a massive scale.
[3] The ECL IDE (Integrated Development Environment) is the tool used to create queries into your data and ECL files with which to build your queries.

Working with Data

The Original Data

In this scenario, we receive a structured data file containing records with people's names and addresses. The HPCC Systems platform also supports unstructured data, but this example is simpler. This file is documented in the following table:

Field Name	Type	Description
FirstName	15 Character String	First Name
LastName	25 Character String	Last name
MiddleName	15 Character String	Middle Name
Zip	5 Character String	ZIP Code
Street	42 Character String	Street Address
City	20 Character String	City
State	2 Character String	State

This gives us a record length of 124 (the total of all field lengths). You will need to know this length for the **File Spray** process.

Load the Incoming Data File to your Landing Zone

A Landing Zone (or Drop Zone) is a physical storage location defined in your HPCC's environment. A daemon (DaFileSrv) must be running on that server to enable file sprays and desprays.

For smaller data files, you can use the upload/download file utility in ECL Watch (a Web-based interface to your HPCC Systems platform). The sample data file is ~100 mb.

1. Download the sample data file from the HPCC Systems® portal.

   The data file is available from links found on https://hpccsystems.com/training/documentation/tutorials . The download is approximately 30 MB (compressed) and is available in either ZIP or tar.gz format (**OriginalPerson.tar.gz** or **OriginalPerson.zip**)

2. Extract it to a folder on your local machine.

3. In your browser, go to the **ECL Watch** URL. For example, http://nnn.nnn.nnn.nnn:8010, where nnn.nnn.nnn.nnn is your ESP[1] Server's IP address.

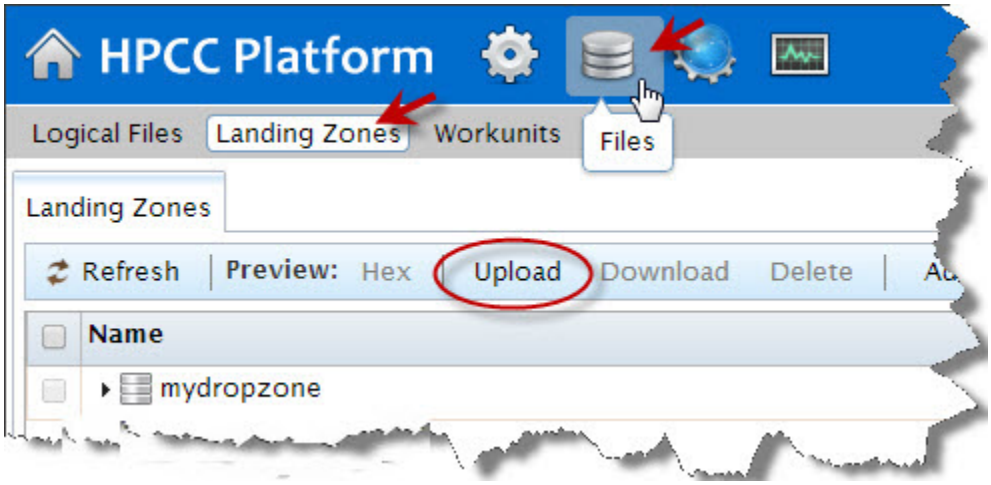
Your IP address could be different from the ones provided in the example images. Please use the IP address provided by **your** installation.

[1]The ESP (Enterprise Services Platform) Server is the communication layer server in you HPCC Systems environment.

4. From the ECL Watch home page, click on the **Files** icon, then click the **Landing Zones** link from the navigation sub-menu.

   Press on the **Upload** action button on the Landing Zones tab.

Figure 1. Upload/download

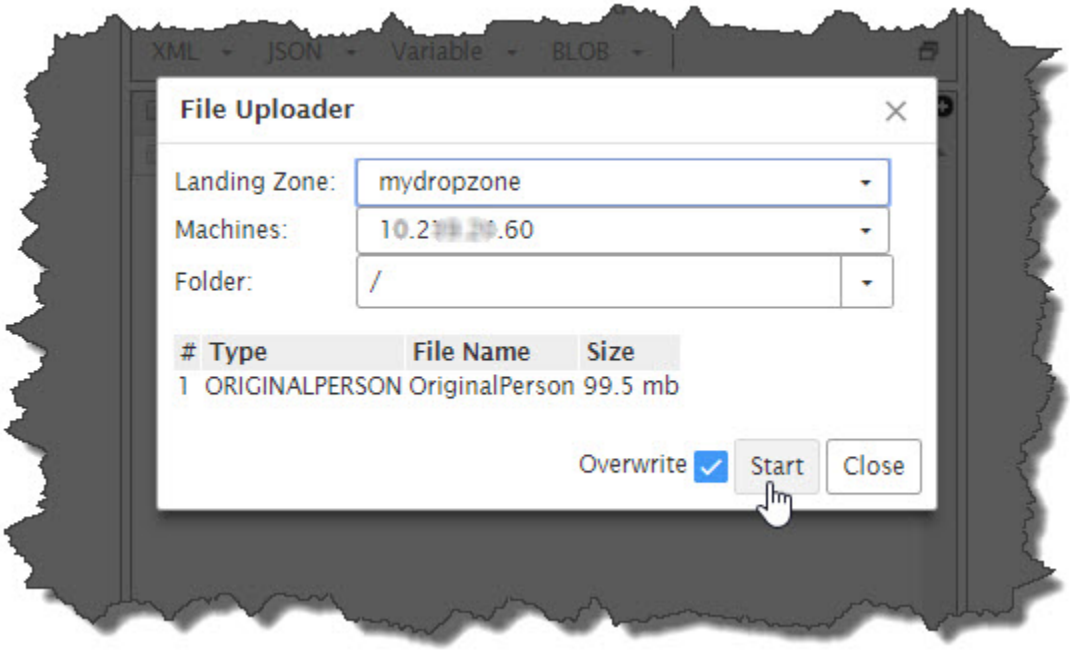


Once you press the Upload button, a dialog opens where you can choose a file to upload.

5. Browse the files on your local machine, select the file to upload, and then press the **Open** button.

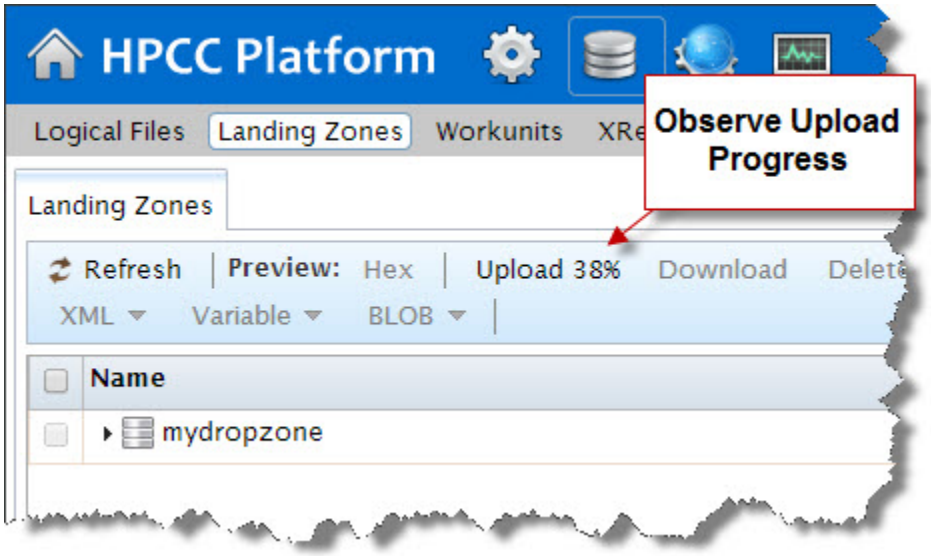
   The file you selected displays in the **File Uploader** dialog.

Figure 2. File Uploader



6. Press the **Start** button to complete the file upload.

Figure 3. Upload Progress



Spray the Data File to your Thor Cluster

To use the data file in our HPCC Systems cluster, we must first "spray" it to a Thor cluster. A *spray* or *import* is the relocation of a data file from one location to a Thor cluster. The term spray was adopted due to the nature of the file movement -- the file is partitioned across all nodes within a cluster.

In this example, the file is on your Landing Zone and is named **OriginalPerson.**

We are going to spray it to our Thor cluster and give it a logical name of **tutorial::YN::OriginalPerson** where **YN** are your initials. The Distributed File Utility maintains a list of logical files and their corresponding physical file locations.

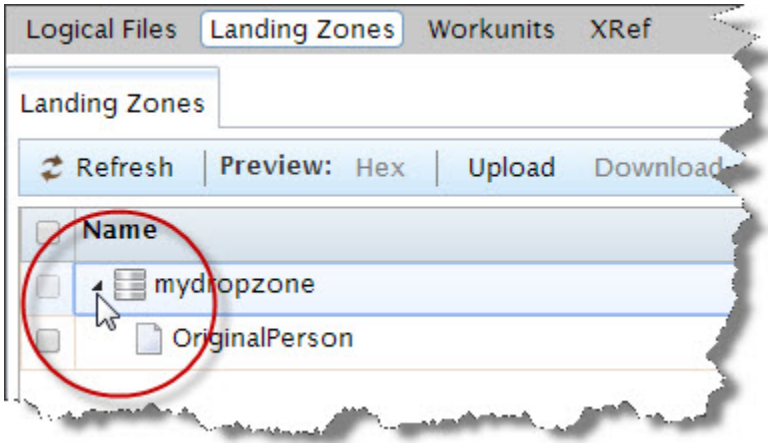
1. Open ECL Watch in your browser using the following URL:

   http://nnn.nnn.nnn.nnn:pppp (where nnn.nnn.nnn.nnn is your ESP Server's IP Address and pppp is the port. The default port is 8010)

2. From the ECL Watch home page, click on the **Files** icon, then click the **Landing Zones** link from the navigation sub-menu.

   On the Landing Zones tab, click on the arrow next to your mydropzone container to expand the list of uploaded files.

Figure 4. mydropzone

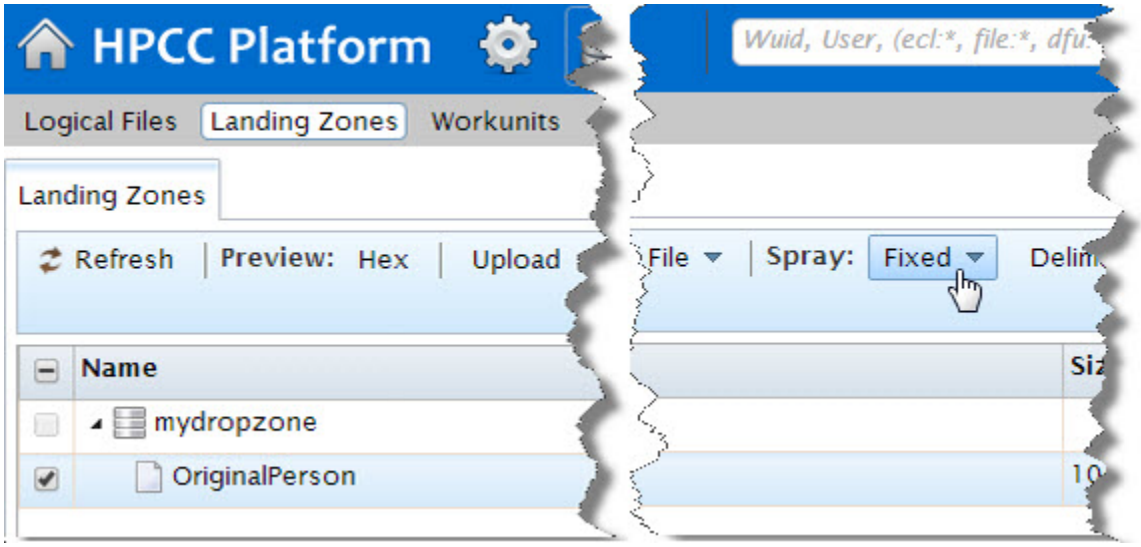


   Find the file you want to spray in the list (OriginalPerson), check the box next to that file name to select that file.

   Once you select the file from the list, the **Spray** action buttons become enabled.

3. Press the **Fixed** action button. This indicates that you are spraying a fixed width file.

Figure 5. Spray: Fixed action button



   The **Spray Fixed** dialog displays.

4. The Target name field is automatically filled in with the selected file.

Figure 6. Spray Fixed dialog

5. Choose the mythor cluster from the **Group** drop list.

6. If there are multiple queues, select one from the list.

7. Fill in the **Record Length** (124).

8. Fill in the **Target Scope** using the naming convention described earlier: **tutorial::YN** (remember, **YN** are your initials).

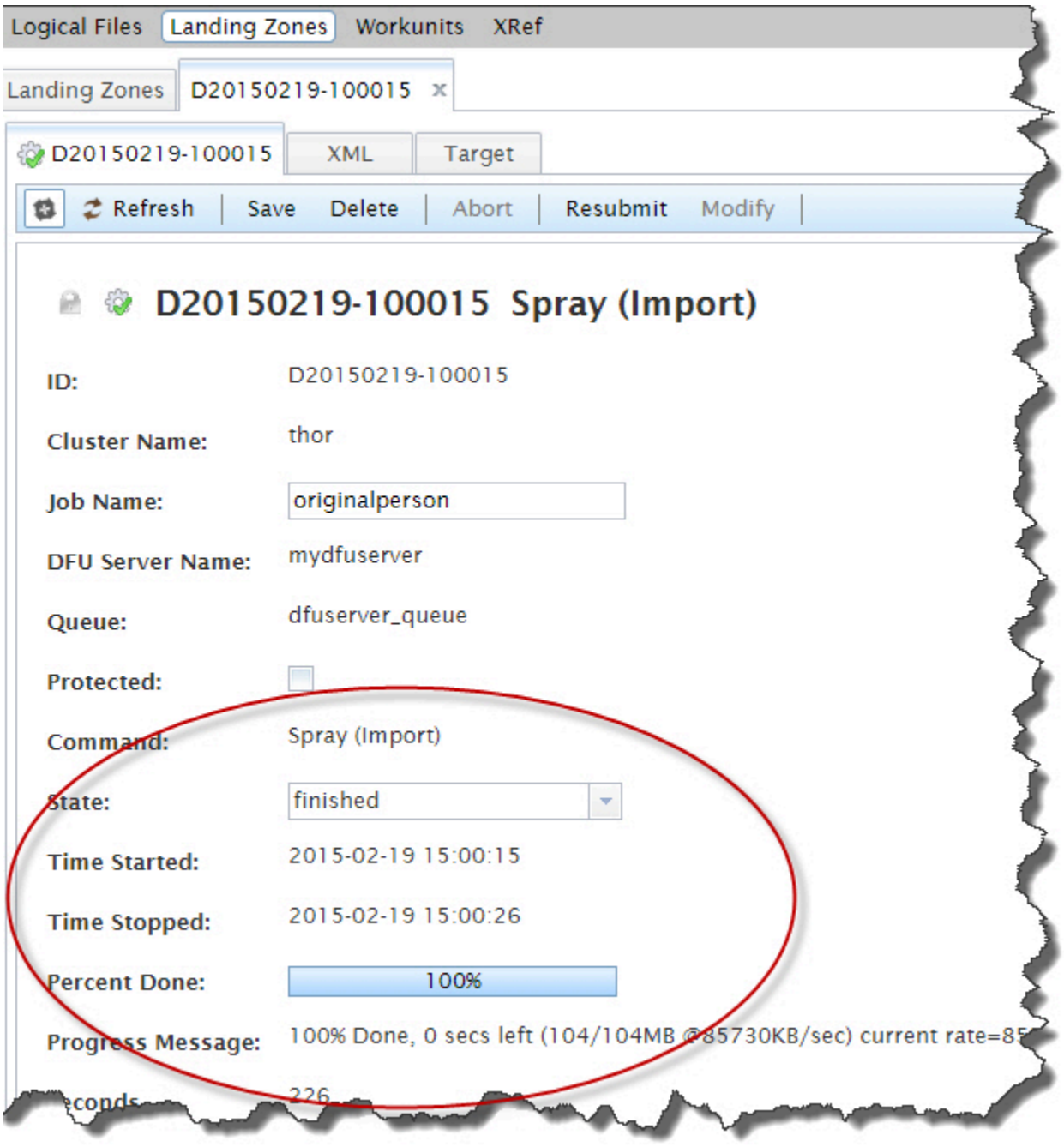
9. Make sure the **Replicate** box is checked.

   Note: This option is only available on systems where replication has been enabled.

10. Press the **Spray** button.

11. The workunit details page displays. You can view the progress of the spray.

Figure 7. View Progress



Once the spray is complete, we can proceed.

Begin Coding

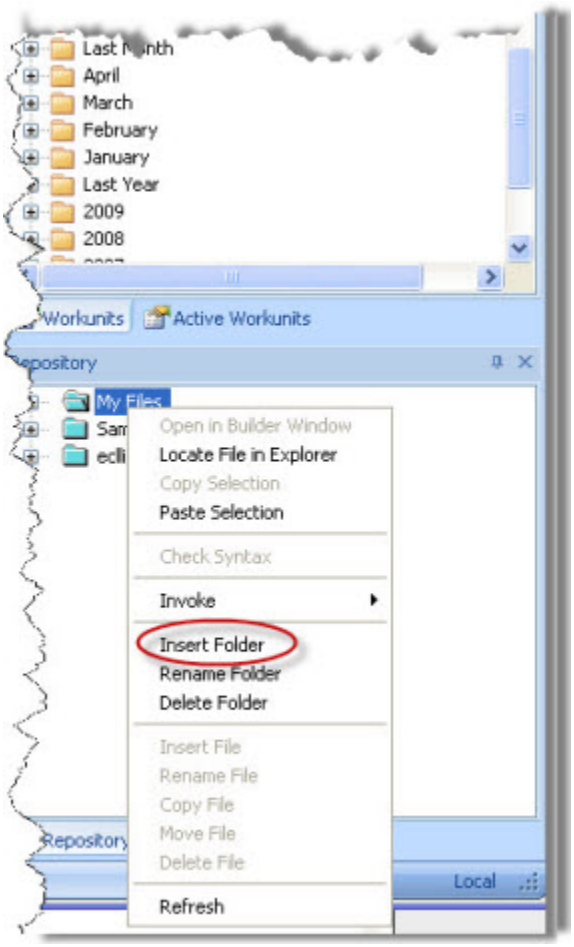
In this portion of the tutorial, we will write ECL code to define the data file and execute simple queries on it so we can evaluate it and determine any necessary pre-processing.

1. Start the ECL IDE (Start >> All Programs >> HPCC Systems >> ECL IDE)

2. Log in to your environment

3. Right-click on the **My Files** folder in the Repository window, and select **Insert Folder** from the pop-up menu.

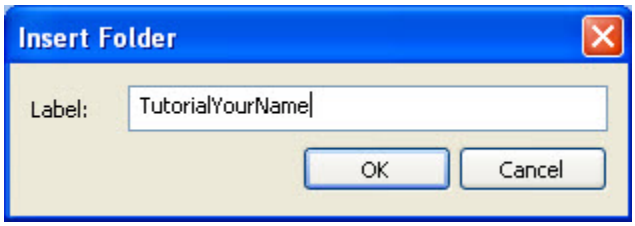
Figure 8. Insert Folder



For purposes of this tutorial, let's create a folder called **TutorialYourName** (where *YourName* is your name).

4. Enter **TutorialYourName**(where *YourName* is your name) for the label, then press the OK button.

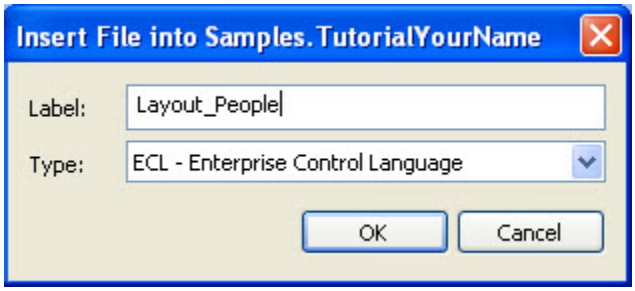
Figure 9. Enter Folder Label



5. Right-click on the **TutorialYourName**Folder, and select **Insert File** from the pop-up menu.

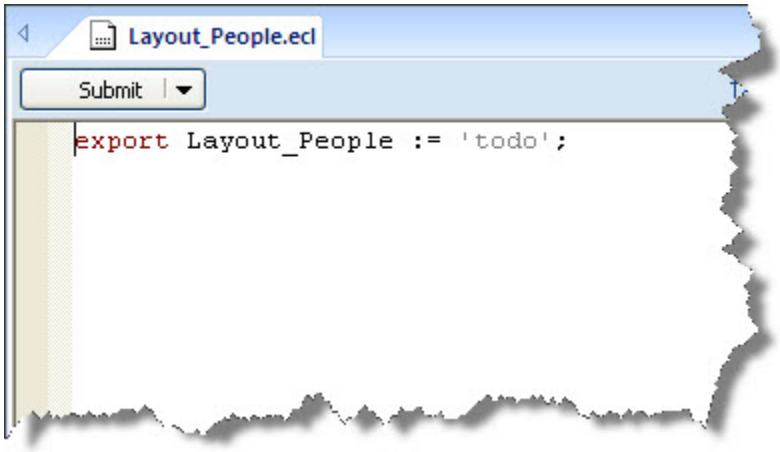
6. Enter **Layout_People** for the label, then press the OK button.

Figure 10. Insert File



A Builder Window opens.

Figure 11. Layout People in Builder

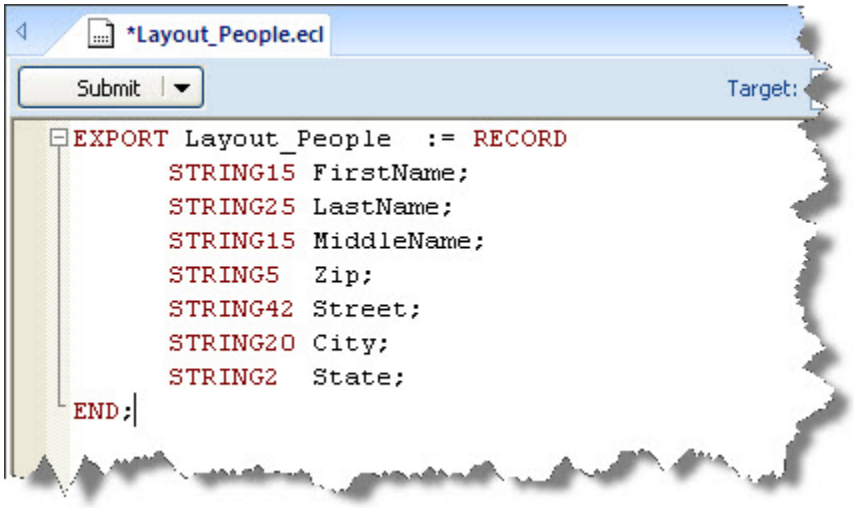


Notice that some text has been written for you in the window. This helps you to remember that the name of the file (Layout_People) *must always exactly match* the name of the single EXPORT definition (Layout_People) contained in that file. This is a requirement -- one EXPORT definition per file, and its name must match the filename.

7. Write the following code in the Builder workspace:

```
EXPORT Layout_People  := RECORD
      STRING15 FirstName;
      STRING25 LastName;
      STRING15 MiddleName;
      STRING5  Zip;
      STRING42 Street;
      STRING20 City;
      STRING2  State;
END;
```

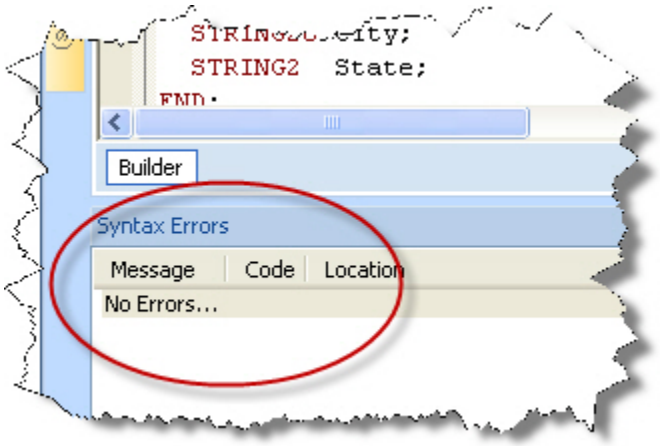
Figure 12. Code in Builder Window



8. Press the syntax check button on the main toolbar (or press F7).

   It is always a good idea to check syntax before submitting.

Figure 13. Check Syntax



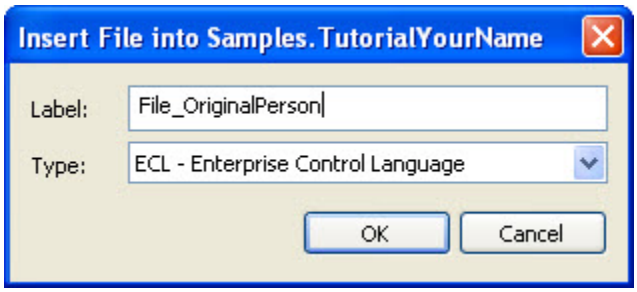
This file defines the record structure for the data file. Next, we will examine the data.

Examine the Data

In this section, we will look at the data and determine if there is any pre-processing we want to perform on the data. This is the step in the development process where we convert the raw data into a form we can use.

1. Right-click on the **TutorialYourName** Folder, and select **Insert File** from the pop-up menu.
2. Enter **File_OriginalPerson** for the label, then press the OK button.

Figure 14. Insert File



A Builder Window opens.

3. Write the following code (remember to replace *YN* with your initials):

```
IMPORT TutorialYourName;
EXPORT File_OriginalPerson :=
DATASET('~tutorial::YN::OriginalPerson',TutorialYourName.Layout_People,THOR);
```

Figure 15. File_OriginalPerson.ecl

4. Press the syntax check button on the main toolbar (or press F7) to check the syntax.

   This defines the Dataset. Next, we will examine the data.

5. Open a new Builder Window (CTRL+N) and write the following code (remember to replace *YourName* with your name):

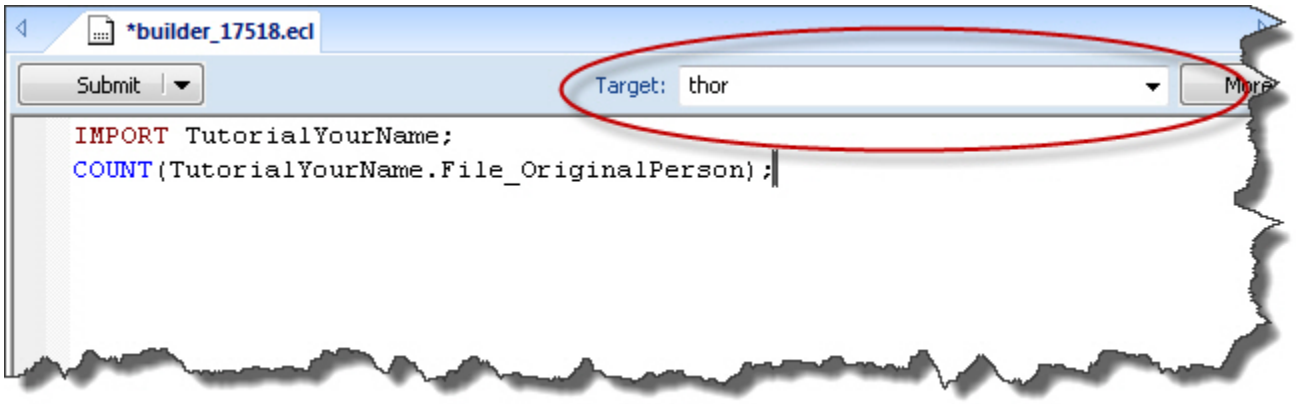
```
IMPORT TutorialYourName;
```

```
COUNT(TutorialYourName.File_OriginalPerson);
```

6. Press the syntax check button on the main toolbar (or press F7) to check the syntax.

7. Make sure the selected cluster is your Thor cluster, then press the **Submit** button. Note that your target cluster might have a different name.

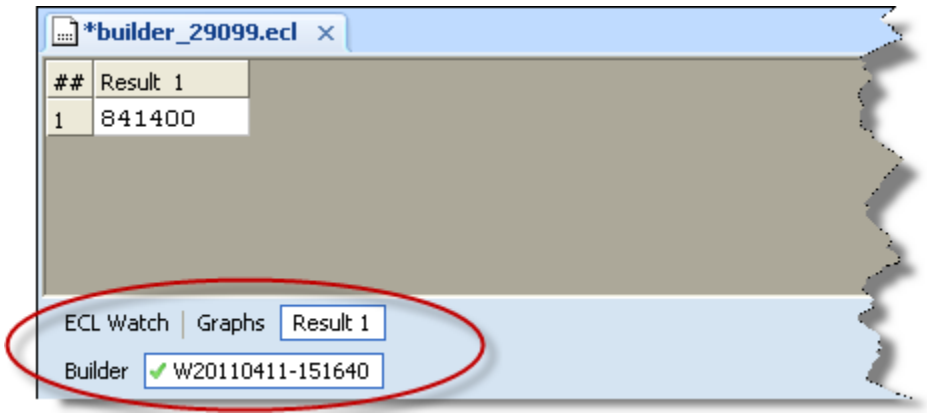
Figure 16. Target Thor



8. When the Workunit completes, it displays a green checkmark ✓.

9. Select the Workunit tab (the one with the number next to the checkmark) and select the **Result 1** tab (it may already be selected).

Figure 17. Result tab



This shows us that there are 841,400 records in the data file.

10. Select the Builder tab and change COUNT to OUTPUT, as shown below:

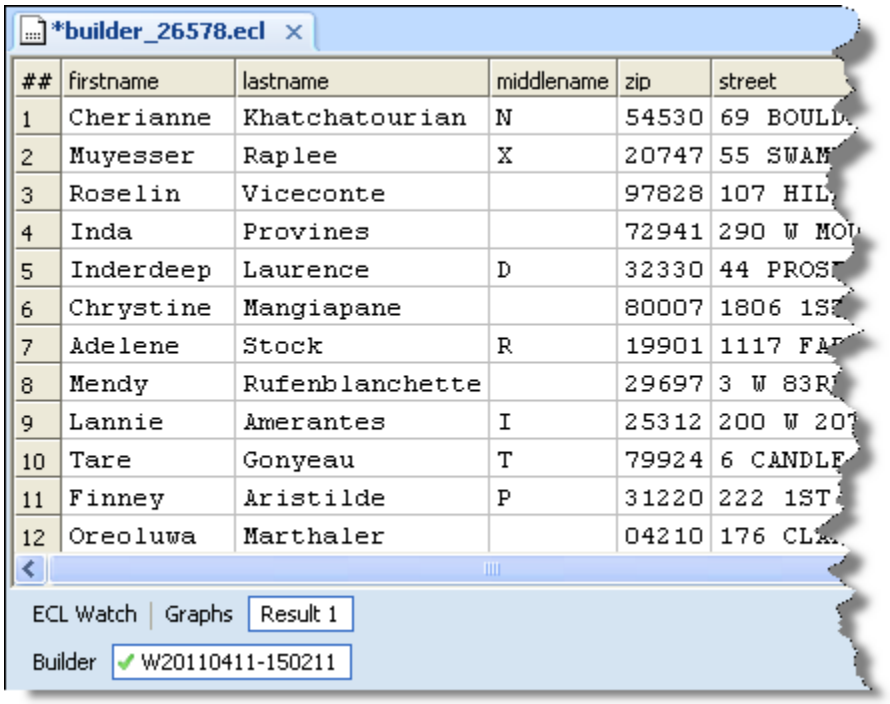
```
IMPORT TutorialYourName;
OUTPUT(TutorialYourName.File_OriginalPerson);
```

Note: The modified portion is shown in **bold**.

11. Check the syntax, if no errors, press the **Submit** button.

12.When it completes, select the Workunit tab, then select the **Result 1** tab.

Figure 18. Output Results



Notice the names are in mixed case.

For our purposes, it will be easier to have all the names in all uppercase. This demonstrates one of the steps in the basic process of preparing data (Extract, Transform, and Load--ETL) using ECL.

13.Close the Builder Window.

Process the Data

In this section, we will write code to convert the original data so that all names are in uppercase. We will then write this new file to our Thor cluster.

1. Right-click on the **TutorialYourName** Folder, and select Insert File from the pop-up menu.

2. Name this one **BWR_ProcessRawData** and write the following code (changing YN and YourName as before):

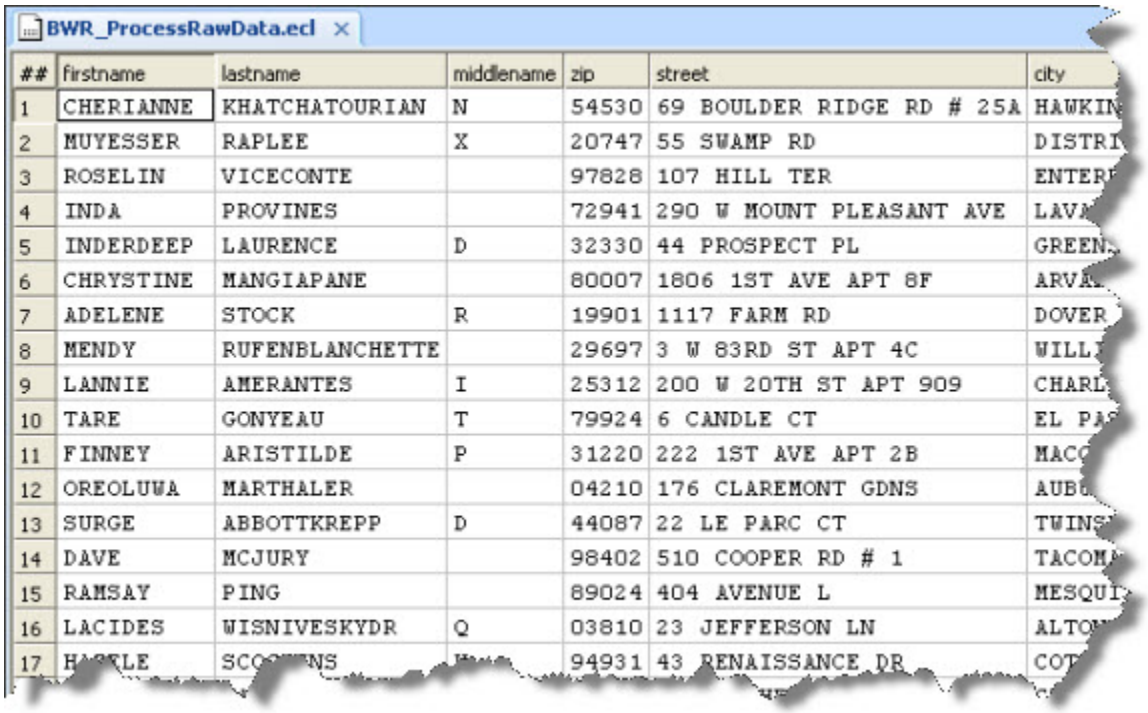
```
IMPORT TutorialYourName, Std;
    TutorialYourName.Layout_People toUpperPlease(TutorialYourName.Layout_People pInput)
      := TRANSFORM
     SELF.FirstName := Std.Str.ToUpperCase(pInput.FirstName);
     SELF.LastName := Std.Str.ToUpperCase(pInput.LastName);
     SELF.MiddleName := Std.Str.ToUpperCase(pInput.MiddleName);
     SELF.Zip := pInput.Zip;
     SELF.Street := pInput.Street;
     SELF.City := pInput.City;
     SELF.State := pInput.State;
    END ;

OrigDataset := TutorialYourName.File_OriginalPerson;
UpperedDataset := PROJECT(OrigDataset,toUpperPlease(LEFT));
OUTPUT(UpperedDataset,,'~tutorial::YN::TutorialPerson',OVERWRITE);
```

3. Check the syntax, if no errors press the **Submit** button.

4. When it completes, select the Workunit tab, then select the Result 1 tab.

Figure 19. Process Result



##	firstname	lastname	middlename	zip	street	city
1	CHERIANNE	KHATCHATOURIAN	N	54530	69 BOULDER RIDGE RD # 25A	HAWKIN
2	MUYESSER	RAPLEE	X	20747	55 SWAMP RD	DISTRI
3	ROSELIN	VICECONTE		97828	107 HILL TER	ENTERP
4	INDA	PROVINES		72941	290 W MOUNT PLEASANT AVE	LAVA
5	INDERDEEP	LAURENCE	D	32330	44 PROSPECT PL	GREEN
6	CHRYSTINE	MANGIAPANE		80007	1806 1ST AVE APT 8F	ARVA
7	ADELENE	STOCK	R	19901	1117 FARM RD	DOVER
8	MENDY	RUFENBLANCHETTE		29697	3 W 83RD ST APT 4C	WILLI
9	LANNIE	AMERANTES	I	25312	200 W 20TH ST APT 909	CHARL
10	TARE	GONYEAU	T	79924	6 CANDLE CT	EL PA
11	FINNEY	ARISTILDE	P	31220	222 1ST AVE APT 2B	MACO
12	OREOLUWA	MARTHALER		04210	176 CLAREMONT GDNS	AUBU
13	SURGE	ABBOTTKREPP	D	44087	22 LE PARC CT	TWINS
14	DAVE	MCJURY		98402	510 COOPER RD # 1	TACOM
15	RAMSAY	PING		89024	404 AVENUE L	MESQUI
16	LACIDES	WISNIVESKYDR	Q	03810	23 JEFFERSON LN	ALTO
17	HA LE	SCO NS		94931	43 RENAISSANCE DR	COT

The results show that the process has successfully converted the name fields to uppercase.

5. After you examine the results, close the Builder window.

Using our New Data

Now that we have our data in a useful format and the file is in place, we can write more code to use the new data file. We will determine the indexes we will need and create them. For this tutorial, let's assume the field we need to index is the Zip code field.

In the DATASET definition, we will add a virtual field to the RECORD structure for the fileposition. This is required for indexes.

1. Insert a File into the **TutorialYourName** Folder. Name it **File_TutorialPerson** and write this code (changing *YN* to your initials):

```
IMPORT TutorialYourName;
EXPORT File_TutorialPerson :=
DATASET('~tutorial::YN::TutorialPerson',
       {TutorialYourName.Layout_People,
        UNSIGNED8 fpos {virtual(fileposition)}},THOR);
```

2. Check the syntax, if no errors press the **Submit** button.

3. When it completes, it displays a green checkmark ✔.

Index the Data

Next, we will define the INDEX.

1. Insert a File into your Tutorial Folder. Name it **IDX_PeopleByZip** and write this code (changing *YN* and *YourName* as before):

```
IMPORT TutorialYourName;
EXPORT IDX_PeopleByZIP :=
INDEX(TutorialYourName.File_TutorialPerson,{zip,fpos},'~tutorial::YN::PeopleByZipINDEX');
```

2. Check the syntax.

   Next, we will build the index file.

3. Insert a File into the **TutorialYourName** Folder and name it **BWR_BuildPeopleByZip** and write this code (replacing *YourName* with your name):

```
IMPORT TutorialYourName;
BUILDINDEX(TutorialYourName.IDX_PeopleByZIP,OVERWRITE);
```

4. Check the syntax and if there are no errors, press the **Submit** button.

5. Wait for the Workunit to complete, then close the Builder Window.

Build a Query

Now that we have an index file, we will write a query that uses it.

1. Insert a File into your Tutorial Folder. Name it **BWR_FetchPeopleByZip** and write this code (changing *YourName* as before):

```
IMPORT TutorialYourName;
ZipFilter :='33024';
FetchPeopleByZip :=
FETCH(TutorialYourName.File_TutorialPerson,
      TutorialYourName.IDX_PeopleByZIP(zip=ZipFilter),
      RIGHT.fpos);
OUTPUT(FetchPeopleByZip);
```

2. Check the syntax and if there are no errors, press the **Submit** button.

3. When it completes, select the Workunit tab, then select the **Result** tab.

4. Examine the result, then close the Builder window and resubmit the code.

   Note: You can change the value of the **ZipValue** field to get results from different Zip codes.

Publishing your Thor Query

Now that we have created an indexed query, the next step is to enable access to it through a Web interface.

Our STORED variables provide a means to pass values as query parameters. In this example, the user can supply the ZIP code so the results are people from that ZIP code.

1. Insert a File into the **TutorialYourName** Folder and name it **FetchPeopleByZipService**

2. Write this code (changing *YourName* as before):

```
IMPORT TutorialYourName;
STRING10 ZipFilter := '' :STORED('ZIPValue');
resultSet :=
            FETCH(TutorialYourName.File_TutorialPerson,
            TutorialYourName.IDX_PeopleByZIP(zip=ZipFilter),
            RIGHT.fpos);
OUTPUT(resultset);
```

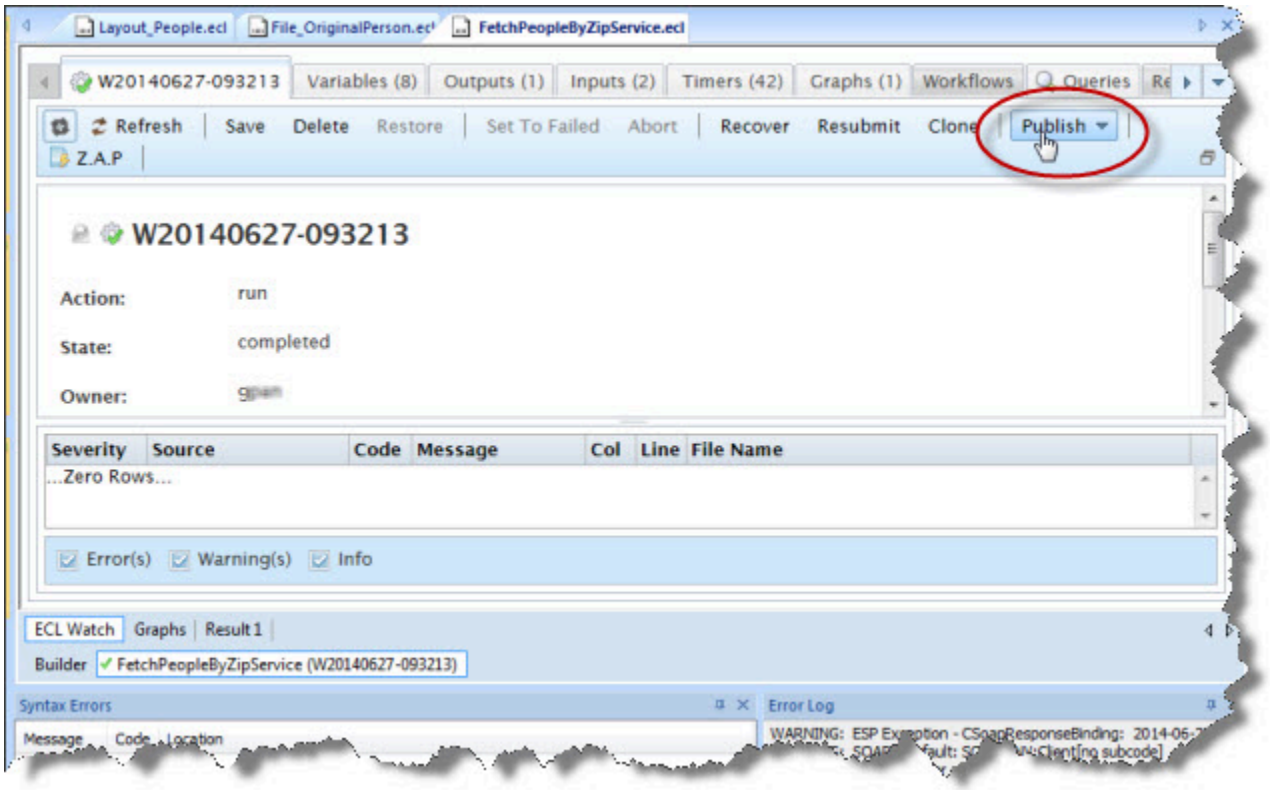
3. Check the syntax, and save the file.

4. Press the **Submit** button.

5. When the workunit completes, select the Workunit tab, then select the ECL Watch tab.

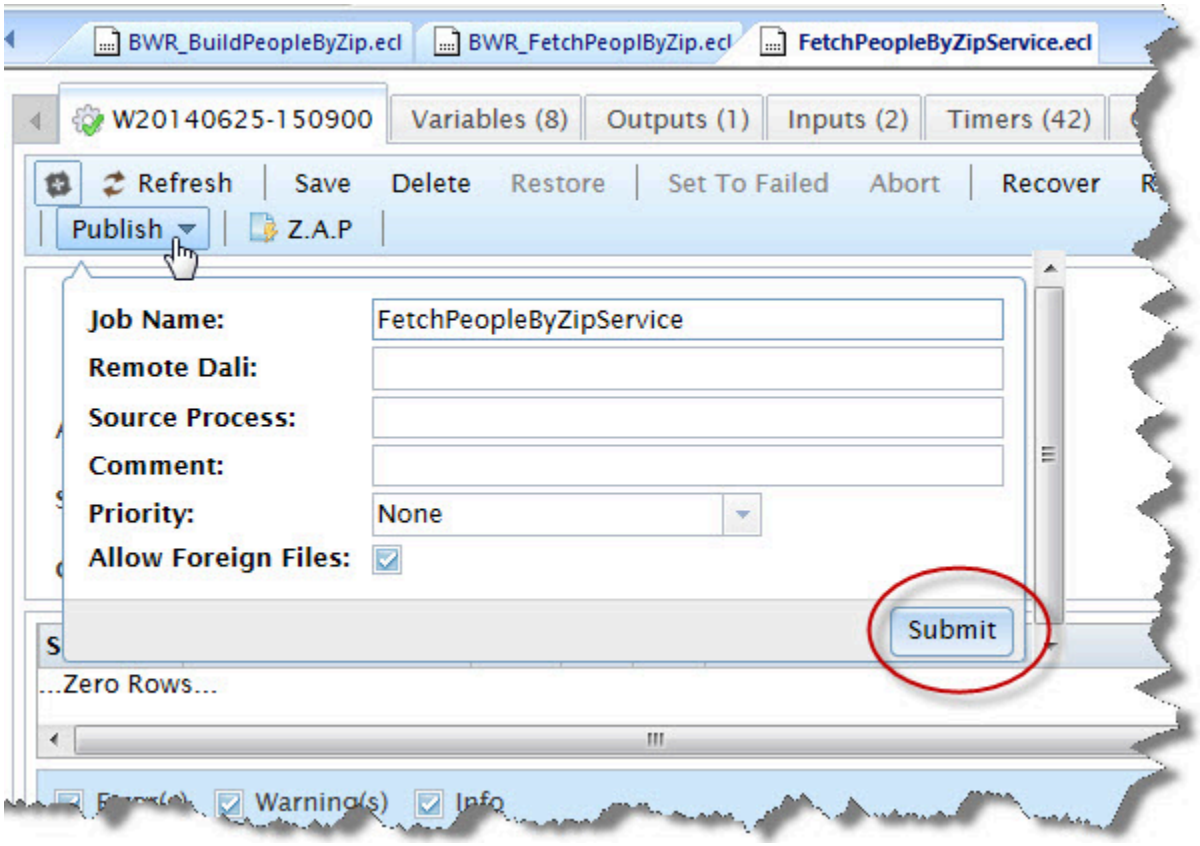
6. Press the **Publish** button, on the ECL Watch tab.

Figure 20. Publish Workunit



The Publish dialog displays, with the Job Name field automatically filled in. You can add a comment in the Comment field if you wish, then press Submit.

Figure 21. Publish Dialog



7. If there are no error messages, the workunit is published. Leave the builder window open, you will need it again later.

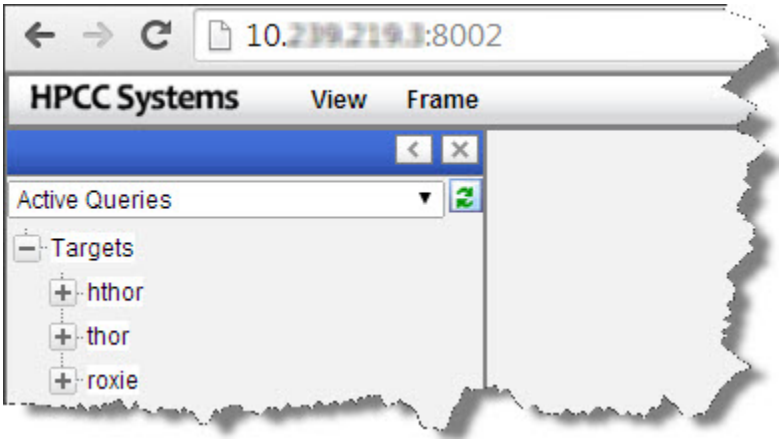
Execute using WsECL

Now that the query is published, we can run it using the WsECL Web service. WsECL provides a Web-based interface to your published query. It also automatically creates an entry form to execute the query.

Using the following URL:

http://nnn.nnn.nnn.nnn:pppp (where nnn.nnn.nnn.nnn is your ESP Server's IP address and pppp is the port. Default port is 8002)

Figure 22. WsECL

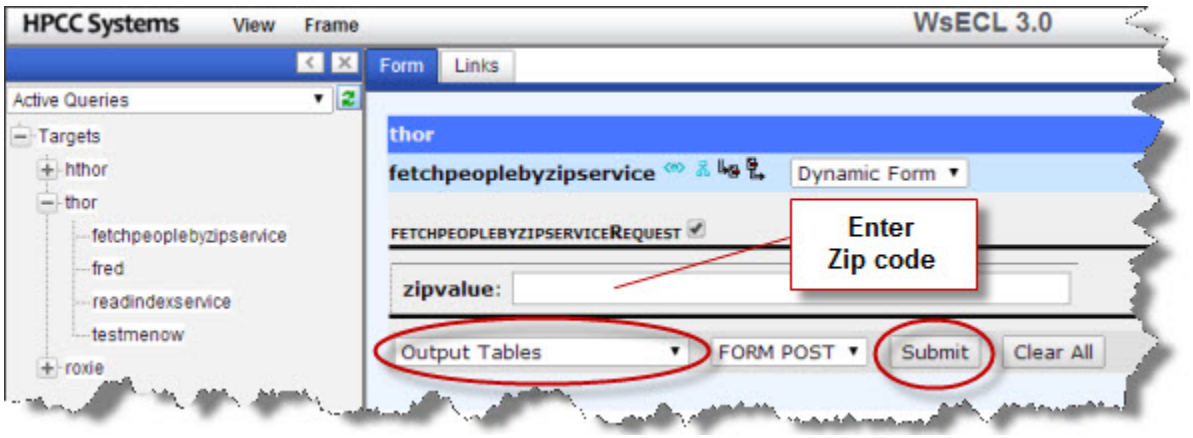


1. Click on the + sign next to **thor** to expand the tree.

2. Click on the **fetchpeoplebyzipservice** hyperlink.

   The form for the service displays.

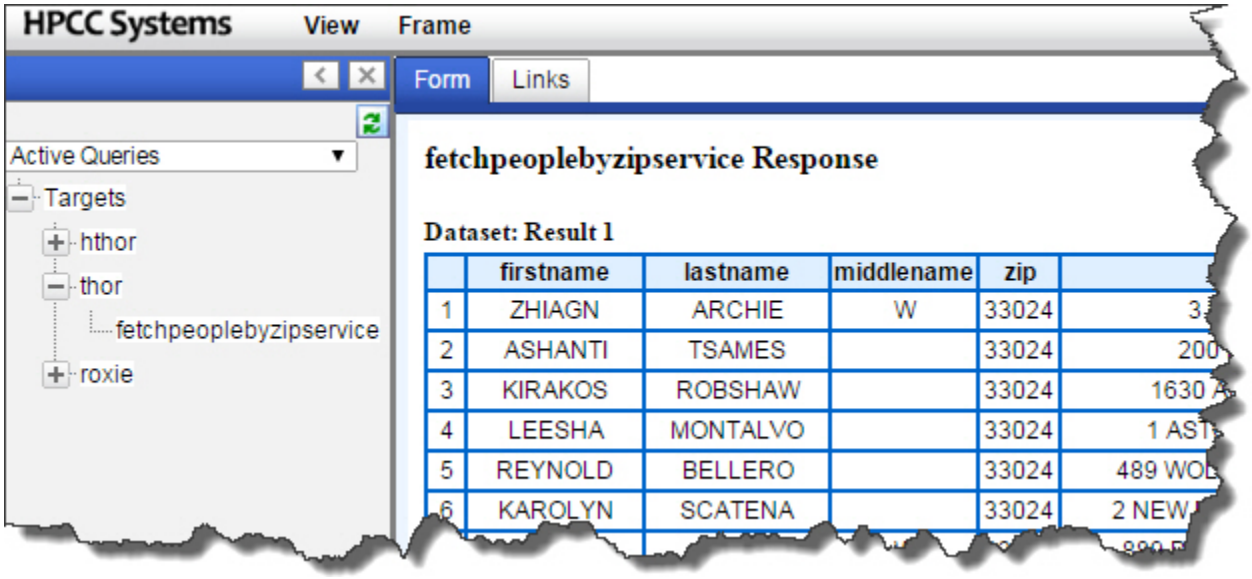
Figure 23. Service Form



3. Provide a zip code (e.g., 33024) in the **zipvalue** field. Select **Output Tables** from the drop list, then press the **Submit** button.

   The results display.

Figure 24. Results



Compile and Publish the Roxie Query

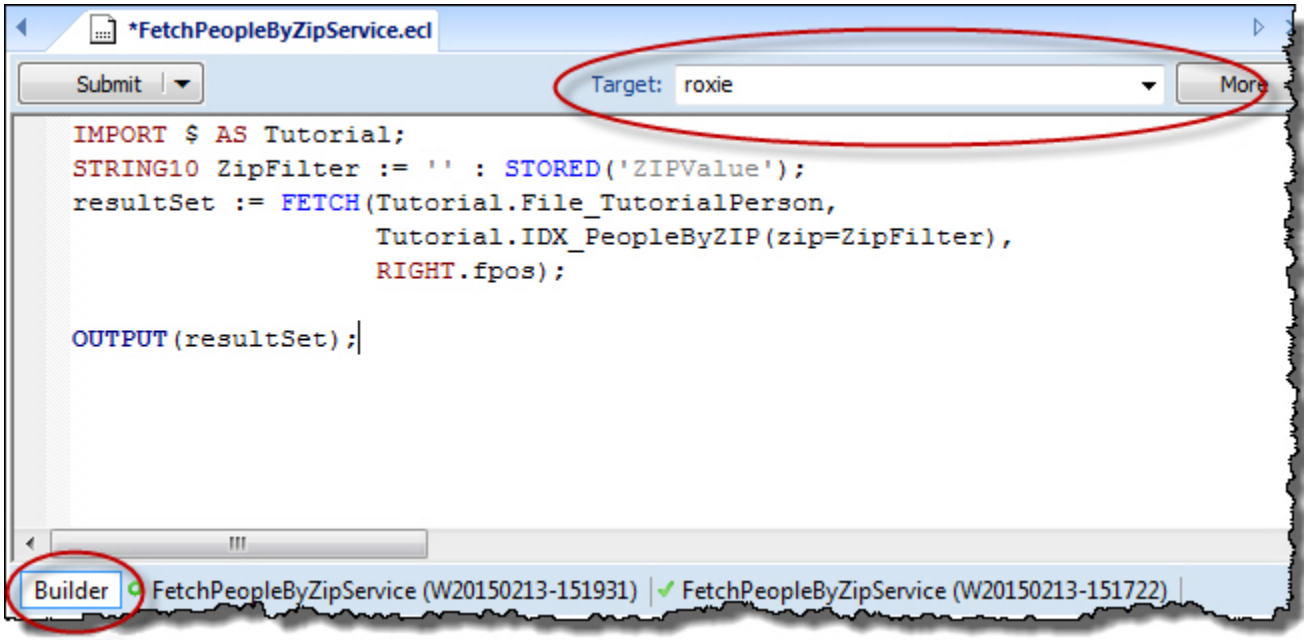
The final step in this process is to publish the indexed query to a Rapid Data Delivery Engine (Roxie) Cluster.

We will recompile the code with Roxie as the target cluster, then publish it to a Roxie cluster.

1. In the ECL IDE, select the Builder tab on the FetchPeopleByZipService file builder window.

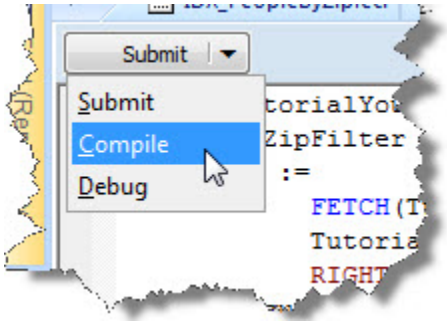
2. Using the **Target** drop list, select Roxie as the Target cluster.

Figure 25. Target Roxie



3. In the Builder window, in the upper left corner the **Submit** button has a drop down arrow next to it. Select the arrow to expose the **Compile** option.

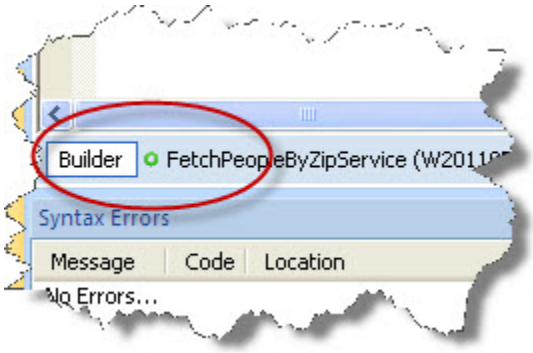
Figure 26. Compile



4. Select **Compile**

5. When the workunit finishes, it will display a green circle indicating it has compiled.

Figure 27. Compiled



Publish the Roxie query

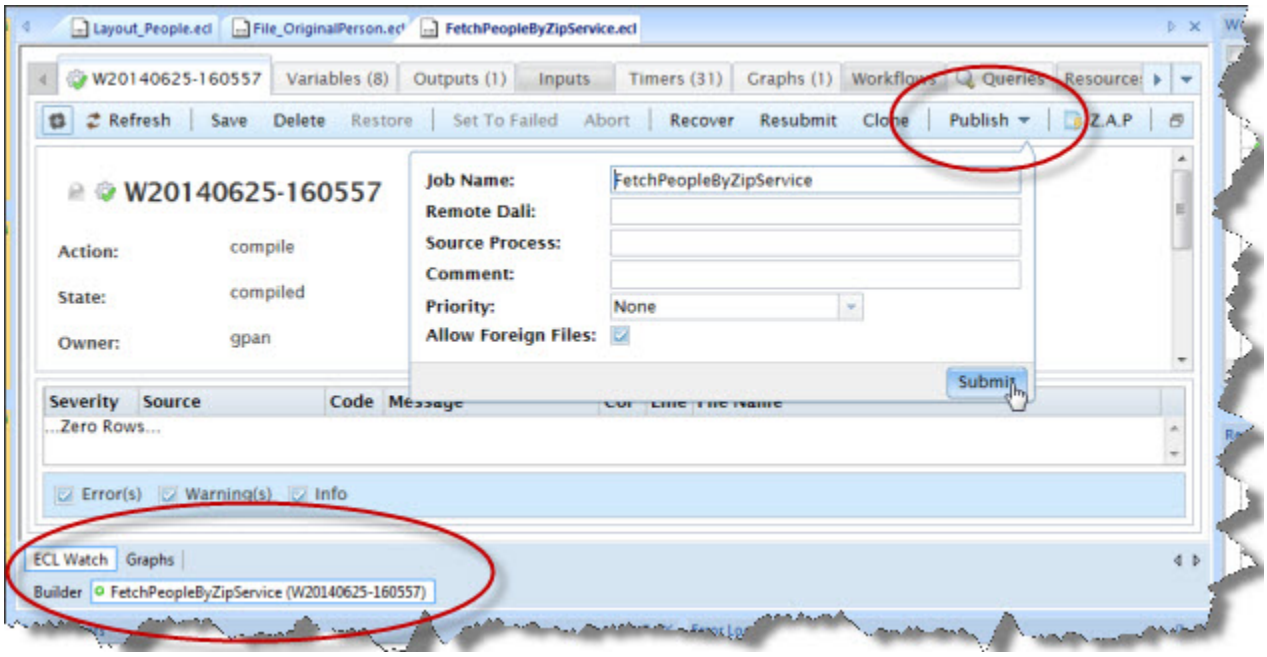
Next we will publish the query to a Roxie Cluster.

1. Select the workunit tab for the FetchPeopleByZipService that you just compiled.

   This opens the workunit in an ECL Watch tab.

2. Press the **Publish** action button, then verify the information in the dialog and press **Submit**.

Figure 28. Publish Query



This publishes the query.

Run the Roxie Query in WsECL

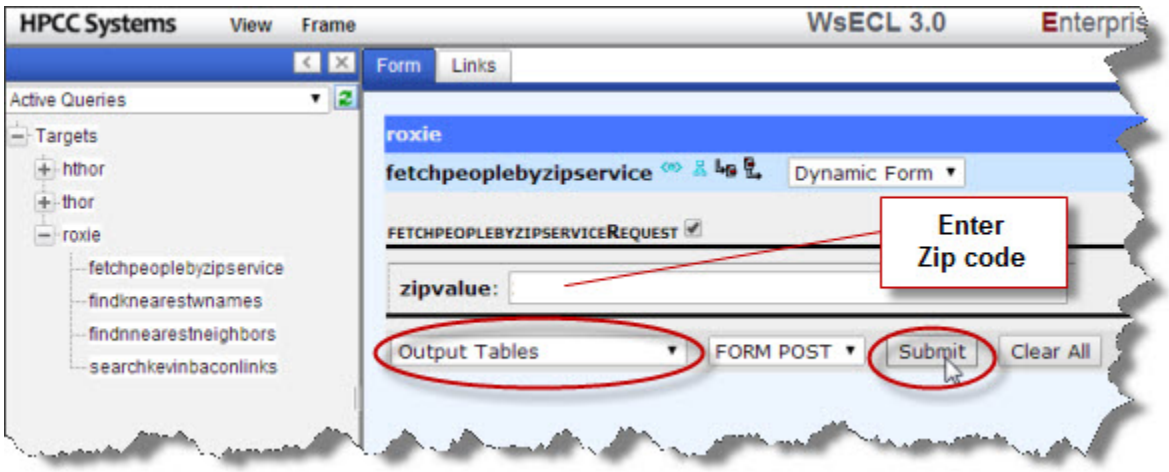
Now that the query is deployed to a Roxie cluster, we can run it using the WS-ECL service Using the following URL:

http://nnn.nnn.nnn.nnn:pppp (where nnn.nnn.nnn.nnn is your ESP Server's IP address and pppp is the port. The default port is 8002)

1. Click on the + sign next to **myroxie** to expand the tree.
2. Click on the **fetchpeoplebyzipservice** hyperlink.

   The form for the service displays.

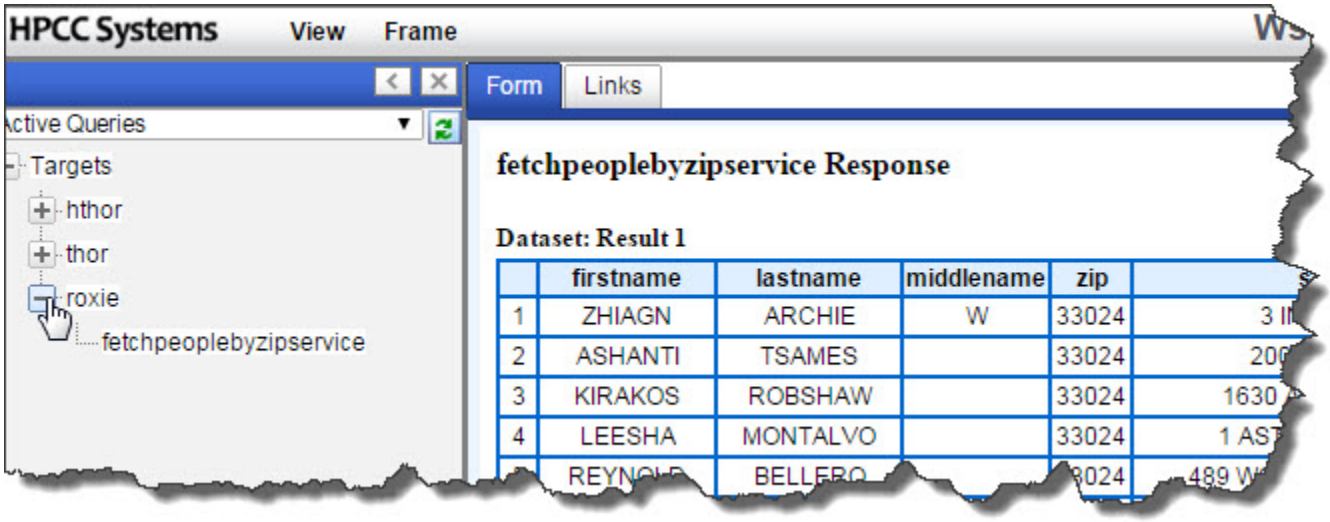
Figure 29. RoxieECL



3. Provide a zip code (e.g., 33024), select **Output Tables** from the drop list, and press the Submit button.

   The results display.

Figure 30. RoxieResults



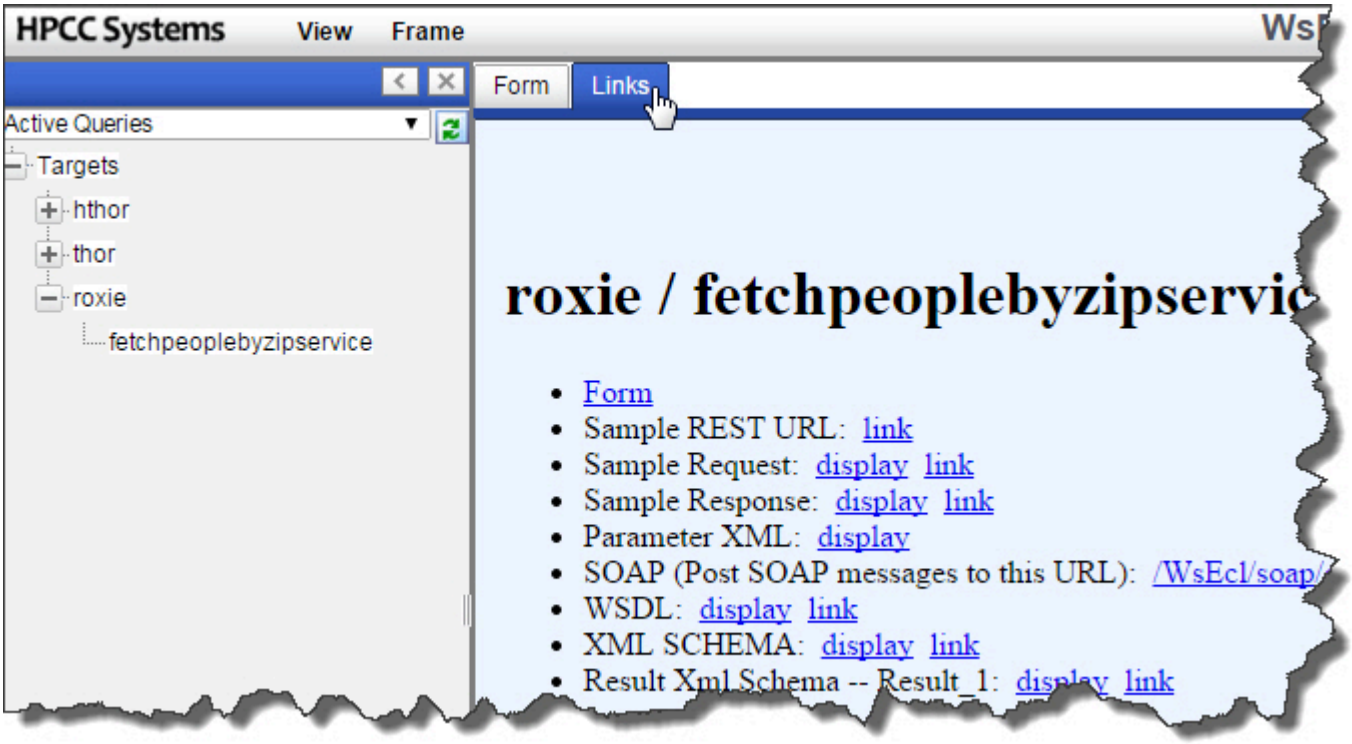
Summary

Now that you have successfully processed raw data, sprayed it onto a cluster, and deployed it to a RDDE cluster, what's next?

Here is a short list of suggestions on the path you might take from here:

- Create indexes on other fields and create queries using them.
- Write client applications to access your queries using JSON or SOAP interfaces.
- Looks at the resources available on the Links tab

Figure 31. Links



The Links tab provides easy access to a form, a Sample Request, a Sample Response, the WSDL, the XML Schema (XSD) and more...

- Follow the procedures in this tutorial using your own data!