Bob Foreman/Hugo Watanuki
Senior Software Engineers
LexisNexis RISK Solutions

ECL Alive! - a Deep-Dive into Definitive HPCC Systems
Hour 3– The ECL Cookbook: Tips and Tricks for Files, Strings, Dates, and more!

HPCC SYSTEMS

Introduction

This workshop is based on the new book by Richard Taylor:

Definitive HPCC Systems

Volume II: Data Transformation and Delivery

Hour 1: Chapters 2 and 3

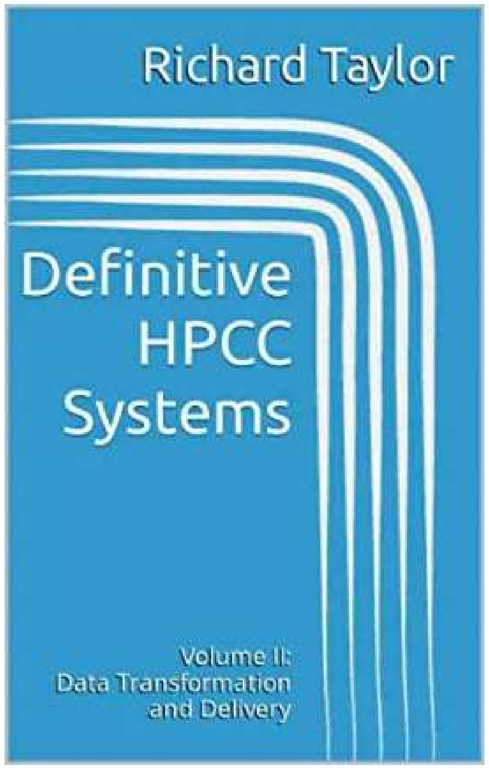
Hour 2: Chapter 4

Hour 3: Chapter 5 (ECL Cookbook)

Volume I and II are currently available on Amazon.

https://www.amazon.com/Definitive-HPCC-Systems-Overview-Platform-ebook/dp/B087Y1FMDH

https://www.amazon.com/Definitive-HPCC-Systems-Transformation-Delivery-ebook/dp/B0BCMZCXDD



Introduction: The ECL Cookbook

- In our first two hours, we have examined the complete product life cycle, from data ingest, through data transformation, and product delivery -- the standard process that every data product must go through no matter what platform is used -- and been introduced to how that process is done on the HPCC Systems platform.
- This workshop examines many more ECL tools and demonstrates aspects of ECL that have not yet been covered.
- You may use these tools, or modify them, or extrapolate from their specific use case to something similar but not quite the same in your projects.

Today’s Chefs:

File Tips

- The HPCC Systems platform is all about working with Big Data, and the choice was made early on to do that using standard file system datasets. This section is about doing things with files that are not simply "straight out of the box" tasks.
- XML and JSON file formats have been industry standards for many years. Advantages are their built-in metadata information which makes them easy to read and interpret.
- Disadvantages are their file sizes, since field information is stored and duplicated in every record.
- Let’s look at the ECL way to generate these formats:

File Tips – Generating:

Creating XML and JSON Files

XMLandJSONFiles.ecl

- Simple XML and JSON
- XML Attributes
- XML row and file tags
- JSON row and file tags

BWR_CustomXMLandJSON.ecl

- Generating custom XML and JSON rows

File Tips – Automatic File Cleanup

- As you work with your HPCC Systems platform you will invariably create a lot of files.
- And it's typical that, as you migrate from project to project, some of those files will no longer be relevant to your production work.
- That means you will periodically need to eliminate files that are no longer needed, and one major determinant in whether or not a file is still needed is found by asking the question, "How long ago was this file last accessed?"
- If the answer to that question is "too long" then it's a candidate for removal. And the code in this section covers how you can do that automatically.
- The *AutoFileCleanup* FUNCTION is designed to clean up logical files by either setting their EXPIRE option or simply deleting them.

AutoFileCleanup.ecl

File Tips – Automatic File Cleanup: Usage

- The *AutoFileCleanup* function takes three parameters. The first is the *WhichMethod* parameter that specifies which of the four supported methods the caller is choosing to use:

  1 – Set EXPIRE for all matching files matching *FilePattern* parameter
  2 – Same as 1, but only delete files that haven't been accessed (*Expiry* parameter)
  3 – Immediate delete of all files matching *FilePattern* parameter
  4 - Delete of all files matching *FilePattern* parameter that haven't been accessed (*Expiry* parameter)

BWR_DeleteXMLJSONSamples.ecl

File Tips – Incremental File Updates

- Data is continually being updated. That means you need to have a plan in place for updating your files.
- This tip demonstrates one way of doing that when you already have a base dataset in place and your added, changed, and deleted records come to you in the form of a single "incremental update" dataset with the same record structure as the base dataset.

IncrementalAddChgRecs.ecl
BWR_IncrementalFileTest.ecl

- This example relies on a unique identifier and implements a FUNCTIONMACRO with clever use of template language.

File Tips – Incremental File Updates

- The previous macro relies on a unique identifier in your dataset.
- But what if you don’t have a single Unique Identifier field?
- Some files have records that are only uniquely identified by a combination of two or more field values.
- The following macro was written to handle this situation.

IncrementalAddChgMulti.ecl
BWR_IncrementalFileTestFieldList.ecl

File Tips – Quoted Strings in CSV Files

- If a CSV file contains string fields that might have the field delimiter as part of the data, then you must enclose that string data in double quotes (“”) so the field delimiter character(s) in the data are not misinterpreted as an actual field delimiter.
- This is easily done in the following example, but how do you enclose all data in the quotes and not just strings?
- Here is a FUNCTIONMACRO that demonstrates this technique:

fnMAC_AddQuotes.ecl
BWR_QuotedCSVStrings.ecl

File Tips – More!

NULL Handling – BWR_NULLinCSVfile.ecl

- Add new fields to original dataset that marks string and integer “nulls”
- Filtering techniques, output to SQL format

File Format Comparison – fnMAC_CompFormats.ecl, FileFormatComparison.ecl

- Use the Template Language #DECLARE and #EXPORT statements to produce a string containing the DFU metadata of the dataset’s RECORD structure.
- Use the PARSE() function to extract the contents of the *name=attribute*.
- Use the *Std.Str.FindReplace()* function to strip out the *name=attribute*’s content.
- Use the ASSERT action to FAIL a workunit when the file formats do not match.

String Tricks – DATA and STRING

- ECL has a Standard Library which already contains many string-handling functions. Therefore, this section is about doing things with strings that are not simply tasks that could be easily accomplished with “straight out of the box” string functions.
- Example 1: Data VS STRING
  Storage, Type Casting, Type Transfer
- Let’s assume you want to validate a username/password combination, and you do NOT want to actually transmit the username and password due to security concerns. One solution would be to use the HASHMD5() function to create the actual transmission value and then compare that to a stored HASHMD5 value.

DATAvsSTRING.ecl

String Tricks – Strings to Codes

- One very common data programming requirement is to map commonly occurring string data to a numeric code. Once you’ve done that, you can either permanently replace the strings with those codes (by creating “lookup” tables) and using that for third normal form data normalization, or temporarily replace them for use in Machine Learning algorithms. This section demonstrates a simple way to accomplish both.

`fnMAC_Txt2Code.ecl – DCTCodeMapping.ecl`

- Create a utility FUNCTIONMACRO to produce string-to-code mapping
- Create and use a DICTIONARY for indexed “lookup” value retrieval

String Tricks – String Differences

- The ECL Standard Library has many string handling functions, but the only two that do string comparisons are the *EditDistance()* and *EditDistanceWithinRadius()* functions, neither of which tells you what is different about the two strings. This section demonstrates a simple function that provides that missing information.

StringDiff.ecl – StringDifferenceTest.ecl

Up to four parameters can be passed to the *StringDiff()* function:

- The first two (S1 and S2) are the strings to compare. These are not omittable.
- The *CS* parameter flags whether to perform a case-sensitive or case- insensitive compare. If omitted, the default is case-sensitive.
- The *JustDiffs* parameter flags whether to return a 1-record record set with only the differences or a 3-record record set with the two input strings and the differences. If omitted, the default is the 3- record return.

Set Tricks – Position in a Set

- Much of the ECL language is set-based, so learning to work with sets is very important.
- One function not directly in the ECL language is the ability to discover which element in the set a specific value is.
- Since this is something that you may want to accomplish in any project you work on, it makes sense to create a utility function to do it.

PosInSet.ecl – GetSetElementNumber.ecl

Math Tricks – Correlation Matrix

- A table of the Pearson's Product Moment Correlation Coefficient between variables.
- Used to summarize potential relationships between data or as input for advanced analysis, such as Machine Learning algorithms.
- A MACRO can be used to generate the ECL code to produce a Correlation Matrix.

MAC_CorrelationMatrix.ecl - BWR_CorrelationTest.ecl

Date Tricks – Date Parsing

- A common requirement on working with dates and times is standardization, because in the real world they can be represented in many different ways. And because of those myriad formats, parsing the information so it can be standardized is the real problem to solve. This section demonstrates two ways to accomplish the task.

BWR_DateParsing.ecl – fnMAC_ParseDates.ecl

- Use the *Std.Date.MatchDateString()* function to parse consistent date strings into a standard YYYYMMDD UNSIGNED4 date.
- Construct parsing patterns that build on each other to create the complex rules that allow any of the most common forms of date string to be parsed into a standard YYYYMMDD UNSIGNED4 date.
- Use the PARSE() function with a TRANSFORM to parse dates and times.

And Even More!

- LastOfMonth FUNCTION (**LastDayOfMonth.ecl**)
- Converting Epoch Date/Time (**BWR_UnixEpochDateTime.ecl**)
- Distance between Two Lat/Long Positions
  - Conversion from degrees-minutes-seconds to digital degrees values (**LatLongDMS2DD.ecl**)
  - Haversine formula implementation (**DistLatLong.ecl**)
- Text similarity
  - Cosine similarity comparison on text documents (**CosineSimilarity.ecl**)
  - Scoring the degree of similarity between documents (**BWR_TextSimilarity.ecl**)

HPCC SYSTEMS

End of Hour 3 Workshop:

This concludes this workshop!

Thanks for attending!

- ✓**Download it all at: https://github.com/hpcc-systems/Community-Workshops**
- ✓**Contact us:** **robert.foreman@lexisnexisrisk.com**
  hugo.watanuki@lexisnexisrisk.com