This Isn’t Your Father’s Essbase Export

There are several ways to export data from Essbase on a large scale. Pulling it via Excel (Smart View or the Essbase Add-In) is not the best way to get large amounts of data when the goal is to move the data somewhere else, so this option will not be covered.

Database Export

The easiest method is to export all the data from a database by exporting the database. This can be done in EAS. This method is easy to automate with Maxl, but has little flexibility with formatting and the only option is to export all the data. It can be exported in column format so the data can easily be loaded into another data repository. If the data needs to be queried, or manipulated, this is a good option.

Essbase Report Script

The classic way, prior to version 9, was to write a report script. This provides a tremendous amount of flexibility in formatting, but most admins struggle with this method, as it requires an in-depth knowledge of how data is most effectively queried. If there isn’t a strong understanding, then extracting data is extremely slow. Report scripts also offer the ability to export specific data, which is often required.

Calculation Script or Business Rule

Now, there is an alternate method that somewhat combines the best of the previous 2 methods. A calculation script, or business rule, can now extract data in column format with a predefined column delimiter. It can filter the result to specific criteria, and doesn’t require the knowledge reports scripts does to make them efficient.

```
/* Export to text file */
DATAEXPORT "File" "delimiter" "fileName" "missingChar"

/* Export to relational database */
DATAEXPORT "DSN" "dsnName" "tableName" "userName" "password"
```

When DATAEXPORT is used, there are a series of settings that can be applied.

```
SET DATAEXPORTOPTIONS {
/* set the level of data to be extracted – same as a database export from EAS */
DataExportLevel ALL | LEVEL0 | INPUT;

/* Turning this on will export any member within the fix statement that is a dynamically calculated member that produces a value */
DataExportDynamicCalc ON | OFF;

/* A value between 0 and 16 – if no value is supplied, the number of decimals positions in the data, or the DataExportPrecision will be used */
DataExportDecimal n;

/* The number of positions numeric data is exported with, emphasizes data accuracy
NOTE: if DataExportDicimal is set, it will override this setting */
DataExportPrecision n;

/* When set to ON, every column will have a member name and they will be repeated from row to row */
DataExportColFormat ON | OFF;

/* Specifies the dense dimension used for columns – typically the months, or periods, are the most logical choice */
DataExportColHeader dimensionName;
/* When set to ON, the dimension names will preceed the data
as column headers */
DataExportDimHeader ON | OFF;

/* When set to ON, data will be in column format and all
members names will be repeated, and no column header will be
included.  Missing and invalid data will be skipped producing
successive dilimiters */
DataExportRelationalFile ON | OFF;

/* When set to ON, if the file specified already exists, it
will be overwritten */
DataExportOverwriteFile ON | OFF;

/* When set to ON, the file exported will include additional
details about the export
     - Summary of data export settings
    - Info, Warning, and Error messages
     - Exact number of blocks to be exported
     - Estimated time, excluding I/O time. */
DataExportDryRun ON | OFF;□};
```

Each method has its advantages. I see the DATAEXPORT function becoming the primary export method because of its flexibility and ease of use.

A Word of Caution

When exporting data, always remember the population that has access to the export. Exporting data is a very popular method for backing up data, but often times it includes total company financials, employee salaries, and other sensitive data.