Redpaper

Hank Sautter
Paul Spagnolo

IBM DS8000 Series Easy Tier Volume Extent Allocation and Chargeback Utility

IBM DS8000 Easy Tier® is an optional, no-charge feature of the IBM® System Storage® DS8700, DS8800, and DS8870.

Easy Tier can move logical volume extents between disk ranks within an extent pool to provide automated hot spot management. Extents of a logical volume that are considered "*hot*" can be promoted to a higher tier of disk. Extents that are considered "*cold*" and that have no or little I/O activity can be demoted to a lower tier of disk. On which disk tier, such as Solid-State Drive (SSD), Enterprise, or Nearline, a volume currently has extents might be of interest for various reasons, including possible chargeback or cost recovery for high performance disk tier use.

Because Easy Tier operates at a subvolume or sublogical unit number (LUN) level, a large volume can have many extents that are spread out over multiple ranks and multiple disk tiers simultaneously. To determine where the exact extents are at any certain time, the storage administrator queries the IBM DS8000® to get the current extent mapping. This can be done easily and automatically by using the examples that are provided in this paper.

A couple of commands is all that is needed. Using the provided examples generates the commands automatically and provides a pair of output files that are based on a query to the DS8000. After the query of the DS8000 extent information is completed, you can create a spreadsheet similar to the one that is shown in Figure 1 on page 2. After that data is captured in the spreadsheet, you can easily chart the information as shown in Figure 2 on page 2.

Setting up profiles that specify the DS8000 Storage Systems and saving passwords in the `security.dat` file makes the operation of DSCLI simpler. The batch file that is shown in Example 1 on page 5 is an example of how to automate DSCLI by using these profiles. Each DS8000 Storage Subsystem is identified by a profile. With the appropriate naming convention, multiple profiles can be run with a single invocation of the batch file.

This IBM Redpaper™ publication contains a sample script that provides an example of the coding technique for those individuals interested in implementing a chargeback capability.

VOLTYPE	NAME	ID	SAM	RANK	EXT	VOLSER	Disk Class
ckdVolume	Vol_8000	8000	Standard	R9	7	WA8000	ENT
ckdVolume	Vol_8000	8000	Standard	R17	2	WA8000	SSD
ckdVolume	Vol_8001	8001	Standard	R9	2	WA8001	ENT
ckdVolume	Vol_8001	8001	Standard	R17	5	WA8001	SSD
ckdVolume	Vol_8001	8001	Standard	R19	2	WA8001	ENT
ckdVolume	Vol_8002	8002	Standard	R17	5	WA8002	SSD
ckdVolume	Vol_8002	8002	Standard	R19	4	WA8002	ENT
ckdVolume	Vol_8003	8003	Standard	R17	6	WA8003	SSD
ckdVolume	Vol_8003	8003	Standard	R19	3	WA8003	ENT
ckdVolume	Vol_8004	8004	Standard	R9	3	WA8004	ENT
ckdVolume	Vol_8004	8004	Standard	R21	5	WA8004	NL
ckdVolume	Vol_8004	8004	Standard	R19	1	WA8004	ENT
ckdVolume	Vol_8005	8005	Standard	R17	5	WA8005	SSD
ckdVolume	Vol_8005	8005	Standard	R19	4	WA8005	ENT
ckdVolume	Vol_8006	8006	Standard	R9	1	WA8006	ENT
ckdVolume	Vol_8006	8006	Standard	R17	5	WA8006	SSD
ckdVolume	Vol_8006	8006	Standard	R19	3	WA8006	ENT
ckdVolume	Vol_8007	8007	Standard	R9	3	WA8007	ENT
ckdVolume	Vol_8007	8007	Standard	R17	5	WA8007	SSD
ckdVolume	Vol_8007	8007	Standard	R19	1	WA8007	ENT
ckdVolume	Vol_8008	8008	Standard	R17	4	WA8008	SSD

Figure 1 Completed query

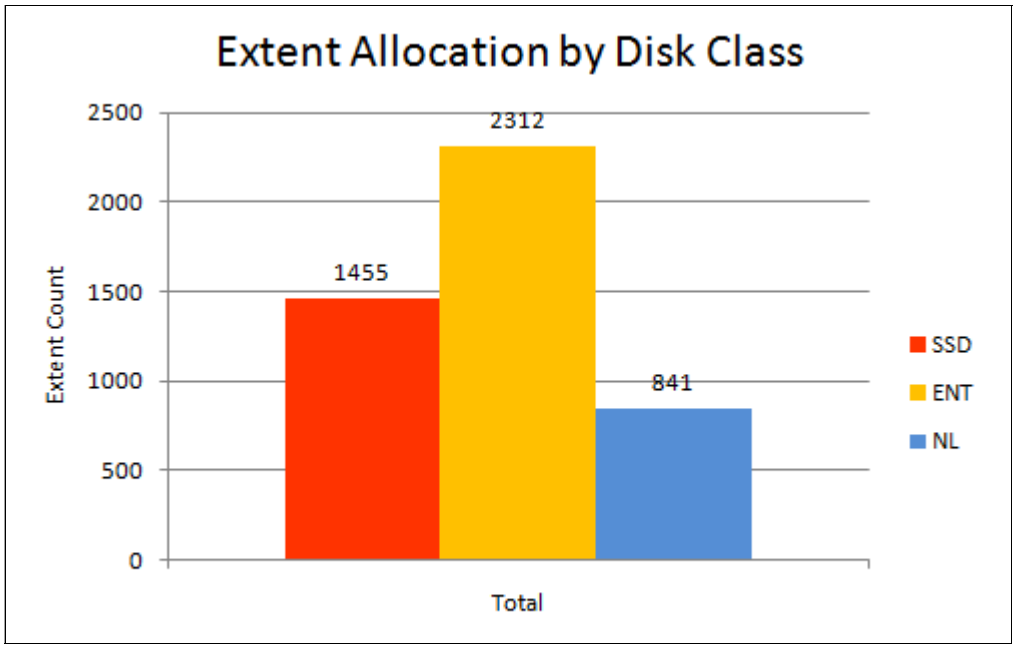


Figure 2 Extent allocation graph

DSCLI installation and user setup

The examples that are provided in this Redpaper depend on the installation of the DS8000 DSCLI tool and an optional profile. The profile is useful if you have multiple DS8000s from which you want to collect data. The profile is also useful if you routinely collect volume and extent information regularly or use some method of automated scheduling.

The following section describes the initial setup steps.

Installing and setting up profiles and passwords

Follow these steps:

1. Install DSCLI by following the program's installation instructions. The current version of the DSCLI tool is available on an IBM FTP site:

   `ftp://ftp.software.ibm.com/storage/ds8000/updates/DS8K_Customer_Download_Files/`

2. Customize settings for your installation:

   a. Optional: Edit the properties of the DSCLI shortcut on your desktop to keep your user profiles and password files separate from the system defaults that are installed with DSCLI:

   ```
   Target: c:\Windows\System32\cmd.exe /k cd %USERPROFILE%\dscli
   ```

   b. Copy the default profile to your user directory:

   ```
   C:\Program Files\IBM\dscli\profile\dscli.profile
   Copy to: %USERPROFILE%\dscli\profile\
   ```

   c. Edit the profile and save it as a unique name for each DS8000. Use file extension `.ini` [`Example.ini`]. Do not put any blanks after the values.

   - Update these fields to match your configuration:

     ```
     hmc1:127.0.0.1
     hmc2:127.0.0.1
     devid:IBM.2107-AZ12341
     remotedevid:IBM.2107-AZ12341
     username: myuserid
     ```

   - Update other defaults as desired:

     ```
     Banner:off
     Fullid:on
     ```

   d. Load passwords. After you load passwords, passwords will be read from the `security.dat` file.

   Connect by using your profile and add the password:

   ```
   dscli -cfg Example.ini -user admin
   managepwfile -action add -name myuserid -pw mypassword
   ```

Gathering volume data

In our example, we used Notepad and created a Microsoft Windows command file (batch file) and saved it with the file name `Get_Vol_Data.bat`. Follow these steps:

1. Gather the necessary information to determine the number of extents that is allocated in each Data Class.

   Run **`Get_Vol_Data.bat`** and specify the profile for the DS8000 Storage System to be queried. This can be a wild card profile specification to process multiple DS8000 subsystems:

   ```
   Get_Vol_Data  Example*
   ```

2. Verify that two files were created with volume and rank data:
   - Volume List [`Example_vol.xml`]

     This `xml` file can be imported into Microsoft Excel and is displayed in a table.
   - Rank List [`Example_rank.csv`]

     The rank information can be added to Excel by using “Get External Data”.
3. Later in this document is an Excel spreadsheet example with the imported data and a pivot table that is created to help analyze the data. A lookup function is used to add the Data Class to each volume rank row.

Batch file execution details

The batch file has two execution methods:

- Use a DSCLI profile to identify the set of DS8000s:

  ```
  Get_Vol_Data Profile* or
  Get_Vol_Data Mydisk
  ```

  `Profile` specifies the profile sets to be used. The asterisk is used to indicate a mask and any `.INI` files in the directory are read as input and processed. So `Example*` reads `Example1.ini`, `Example2.ini`, `Example3.ini`, and so on.
- The other option that you can use is to specify certain DSCLI parameters that are required to access a single DS8000:

  ```
  Get_Vol_Data  nickname -hmc1 xx.xx.xx.xx -user uuuuu -passwd ppppp
  ```

  `Nickname` is used as the output file name and is a required positional parameter.

  The remaining parameters are the usual required DSCLI parameters where:
  - `-hmc1` *`xx.xx.xx.xx`* is the IP address or domain name server (DNS) name of the DS8000
  - `-user` *`uuuu`* is your user ID
  - `-passwd` *`ppppp`* is your password

  Using this second method of invoking the script causes the script to prompt you for the device name. In the DSCLI, this has the format of `IBM.2107-75AAAAA` and corresponds to the `-DEV` parameter that is associated with many of the DSCLI commands.

The output file names are based on the profile name or nickname that is specified:

```
>set file1=Example_vol.xml
>set file2=Example_rank.csv
```

The DSCLI commands that are executed gather volume data for fixed block (FB) and count key data (CKD) volumes:

1. **lsarray** is used to gather the disk class for each rank.
2. **lsfbvol** and **lsckdvol** get a list of the volumes that is used to build a DSCLI script, which uses the **showfbvol** and **showckdvol** commands to obtain the rank level details of the volume extent placement:

   ```
   showfbvol -rank "volid"
   showckdvol -rank "volid"
   ```

3. Results from the s**howfbvol** or **showckdvol** commands are used to create the volume xml file (see Example 1).

Example 1 DSCLI commands to gather volume data

```
>dscli -cfg profile\Example.ini lsarray -fmt delim -delim , -state assigned -l
>dscli -cfg profile\Example.ini lsfbvol   -hdr off -s -eam managed
>dscli -cfg profile\Example.ini -script tmp.lst
>dscli -cfg profile\Example.ini lsckdvol   -hdr off -s -eam managed
>dscli -cfg profile\Example.ini -script tmp.lst
```

The output files can be easily imported into Excel (Example 2). Follow these steps:

1. Open the volume xml file with Excel and an Excel table is created.
2. Import the rank CSV file by using data import and create a table.
3. Add a column to the volume table and use **vlookup** to specify the Data Class.
4. Create a PivotTable from the volume table, and the number of extents allocated to each volume in each rank or Data Class can be summarized. This is covered in detail in “Volume information processing” on page 12.

Example 2 Creating output files for import into Excel

```
>Get_Vol_Data.bat Example

>set file1=Example_vol.xml
>set file2=Example_rank.csv
>set parms=-cfg profile\Example.ini

>dscli -cfg profile\Example.ini lsarray -fmt delim -delim , -state assigned -l
1>tmp.txt
>dscli -cfg profile\Example.ini lsfbvol   -hdr off -s -eam managed  1>tmp.txt
>dscli -cfg profile\Example.ini -script tmp.lst   | findstr "^R... ID Name
volser sam"  1>tmp.txt
>dscli -cfg profile\Example.ini lsckdvol   -hdr off -s -eam managed  1>tmp.txt
>dscli -cfg profile\Example.ini -script tmp.lst   | findstr "^R... ID Name
volser sam"  1>tmp.txt
```

Batch file internals

The example batch file can be altered as required. This section explains how the batch file is structured and some details about general Microsoft Windows command functions. The Windows Help is useful for understanding the details of the commands that are used in these examples.

The intent of the process is to provide an Excel table with the pertinent information about the volumes. The output that is created is in XML format for the volume data. There are two advantages to this format. First, Excel opens an XML file and automatically creates a table, greatly simplifying the import process. The second reason for the XML format is that the display of the DSCLI commands is composed of individual lines. It is easier to handle each line separately as an XML statement and then let Excel build the table format during import.

Input parameters

The initial input for execution needs to identify the DS8000 to be queried. This can be done by providing a DSCLI profile name (see “Installing and setting up profiles and passwords” on page 3) or by specifying the required DSCLI invocation parameters. The top lines of the batch file handle the input parameters and provide a basic help function.

The section of code that is shown in Example 3 determines whether the script was involved by passing it a list of `.ini` files (`Example.ini`) or whether a series of strings were passed containing the nickname, Hardware Management Console (HMC) IP address, user ID, and password. If neither of these two valid options exists, the script exits with a help line suggesting the correct parameters.

Example 3 Parsing the script input parameter

```
@setlocal
@set DEV=
@if "%1"=="" goto end
@if "%2"=="" ( for %%f in (profile\%1.ini) do @call :GetVol %%f -cfg %%f
) else ( @call :Getdev %* )
@goto :EOF
```

If there are no parameters that are provided, a syntax message is presented to the user (shown in the syntax error that is presented to the user in Example 4).

Example 4 Syntax error that is presented to the user

```
:end
@echo enter a profile name or a nickname and DSCLI parameters
@echo %0 nickname -hmc1 xxxxx -user uuuuu -passwd ppppp
```

The “`@`” symbol as a command prefix means “do not echo the command”, which keeps the console output a bit neater in that only the “*important*” commands are displayed.

`@setlocal` creates a set of environment variables that allows multiple instances of the batch file to be run. This allows parallel execution against multiple DS8000s if desired.

`DEV=` clears the device parameter and is needed as part of the DSCLI commands. This is separate from the other parameters that relate to the DSCLI invocation itself.

`:end` is a label that can be used to change the execution order, which in this case prints a message if there are no parameters entered.

`:EOF` is a special label that means go to the end of the file. This results in returning to the calling routine.

The **`for`** command calls the GetVol routine one time for each profile that is found. Profiles are in the `profile` subdirectory, and any valid wildcard characters (`*` and `?`) can be used to return a list of profiles. This allows a group of DS8000 Storage Systems to be queried with one command.

GetVol routine

If there are multiple parameters, it is assumed that these parameters are for a specific DS8000 and that the parameters are used to invoke DSCLI. In this case, the `DEV` parameter is needed and the user must provide that input. The `^` character is an escape character that allows the inclusion of the > symbol, which otherwise is considered a "*pipe*" for redirecting the output. See Example 5 for a sample of this code.

Example 5 Executing the GetVol routine

```
:Getdev
@set /p DEV= "Enter DevID: IBM.2107-75xxxxx" ^>
set DEV=-dev %DEV%
:GetVol
set file1=%~n1_vol.xml
set file2=%~n1_rank.csv
set parms=%2 %3 %4 %5 %6 %7 %8 %9
@set tmp1=temp%time:~6,2%%time:~9,2%.txt
@set tmp2=temp%time:~6,2%%time:~9,2%.lst

@call :array %parms%
@echo ^<IRETURNVALUE^> >%file1%
@call :cmds fb
@call :cmds ckd
@echo ^</IRETURNVALUE^> >>%file1%
@del %tmp1% %tmp2%
@goto :EOF
```

`@/set /p` prompts the user for the `DEV` parameter using the text provided. `%DEV%` will be set to the user's response and then "`-dev`" is prefixed to complete the correct DSCLI command syntax. Execution continues with the `:GetVol` routine.

`:GetVol` begins with formatting the two output file names. The input might be a path and file name, so `%~n1` extracts only the file name. This might also be only a nickname that is provided with the DSCLI parameter execution format. The remaining input parameters are saved in the *%parms%* variable for use with each DSCLI invocation.

There are two temporary files that are needed throughout the process. To keep these files unique, the names are derived from the current time using a special substring format (see Example 6). This example gets two characters from the time display starting at position 6 and two characters starting at position 9.

Example 6 Forming temporary file names

```
%time:~6,2%%time:~9,2%
```

The routines that perform the actual processing of the DSCLI commands are the `:array` and the `:cmds` subroutines:

- `:array` gets the list of configured ranks, which includes the drive type information and the resulting list is saved in the `rank.csv` file (see Example 7).

  The volume information needs to be saved in XML format so the initial XML statement is added to the output XML file.

- `:cmds` gets data for FB or CKD volumes.

  The closing XML statement is added to file1 and then the temp files are deleted. In this case, `goto :EOF` returns to the command prompt, ending the bat file execution.

Array routine

The drive type for each configured array is needed as a lookup table in the Excel Workbook. This data is saved in the `rank.csv` file as a comma-separated value (CSV) list.

Example 7 Array routine

```
:array
@if EXIST %file2% @del %file2%
dscli %parms% lsarray %DEV% -fmt delim -delim , -state assigned -l 1> %tmp1%
@for /F "eol=C tokens=6,7,8,9 delims=," %%R in (%tmp1%) do @echo %%R,%%U,%%S,%%T
>> %file2%
@goto :EOF
```

The DSCLI command **`lsarray`** provides the needed information and the results are saved in the `%tmp1%` file.

The next **`for /F`** command reads this file and parses each line. The word in position 6 is the rank, and the word in position 9 is the rank type. These values are printed (**`echo`**) to `%file2%`.

The word delimiter is a comma as specified in the DSCLI command (**`-fmt delim -delim ,`**) which results in a comma-delimited file.

`Eol=C` means that C is an end of line character. This skips the processing of any line that begins with C and is used to skip DSCLI messages such as messages that appear when there are no arrays defined (DSCLI messages begin with `CMUC`*`xxxxxx`*).

CMDS routine

There are two sets of DSCLI commands: one set for FB volumes and the other set for CKD volumes. The `:cmds` routine (see Example 8) uses `FB` or `CKD` as a parameter to select the correct command set. The DSCLI command **`lsfbvol`** or **`lsckdvol`** is used to obtain a list of the volume IDs. The DSCLI option `-s` provides this list as one volume ID per line, and the result is saved in the `%tmp1%` file.

Example 8 CMDS routine

```
:cmds
@if EXIST %tmp2% @del %tmp2%
dscli %parms% ls%1vol %DEV% -bnr off -hdr off -s -eam managed 1> %tmp1%
@for /F "eol=C" %%V in (%tmp1%) do @echo show%1vol %DEV% -rank %%V >>  %tmp2%
@if EXIST %tmp2% ( @call :list %1 ) else ( @type %tmp1% )
@goto :EOF
```

The ' **for /F "eol=C"** ' command parses this file and skips the DSCLI messages. The result is written to the %tmp2% file, which is a list of DSLCI **showfbvol** or **showckdvol** commands, issued against each volume ID.

If there are any DSCLI commands in the %tmp2% file, the :list routine processes them and :EOF returns to the :GetVol routine. If there are no commands to process, any DSCLI messages that were in the %tmp1% file are printed to the console.

List routine

The DSCLI commands **showfbvol** and **showckdvol** provide the number of extents in each rank for the volume when the -rank option is specified. This output has extra information so the result is filtered through the findstr application. (For more details, see the Windows Help). The :list routine (Example 9) searches for lines that begin with the strings "^R... ID Name volser sam " and the lines are written to the %tmp1% file.

Example 9 List routine

```
:list
dscli %parms% -script %tmp2% | findstr "^R... ID Name volser sam " 1> %tmp1%
@set P=
@echo ^<INSTANCE VOLTYPE="%1Volume"^>  >>%file1%
@for /F "tokens=1,2" %%A in (%tmp1%) do @call :XML %%A %%B
@if defined P ( @echo ^</PROPERTY^> >>%file1% )
@echo ^</INSTANCE^> >>%file1%
@goto :EOF
```

The variable %P% is used to determine whether a </Property> XML statement is needed and is set when <Property> statements are written to the file. This detects the first time that the routine is called. If there are no volumes to process, the Property statement is not needed.

The <INSTANCE VOLTYPE="%1Volume"> XML statement provides separation between the FB volumes and the CKD volumes in the XML file.

The XML routine creates the XML statements for the volume properties.

The </INSTANCE> statement is written and :EOF returns to the :cmds routine.

XML routine

The function of the XML routine (see Example 10) is to handle the creation of the XML statements as needed. Each line is parsed and the routine that handles the needed XML statement is called.

Example 10 XML routine

```
:XML
@if %1==Name   ( @call :NAME %* ) ELSE (
@if %1==ID     ( @call :ID %*   ) ELSE (
@if %1==sam    ( @call :SAM %*  ) ELSE (
@if %1==volser ( @set volser=%2 ) ELSE ( @Call :RANK %* ) )))
@goto :EOF

:NAME
@if defined P ( @echo ^<VOLSER^>%volser%^</VOLSER^>^</PROPERTY^> >>%file1% ) else
set P=1
```

```
@set volser=
@echo ^<PROPERTY NAME="%2"^> >>%file1%
@goto :EOF

:SAM
@echo ^<SAM^>%2^</SAM^> >>%file1%
@goto :EOF

:ID
@echo ^<ID^>%2^</ID^> >>%file1%
@goto :EOF

:RANK
@echo ^<RANK EXT="%2"^>%1^</RANK^> >>%file1%
@goto :EOF
```

There is some variability in the data in that FB volumes do not have a volser. However, we need a column for that data and so a `<VOLSER>` statement is always needed. The *%P%* variable detects the first set of volume data and the volser information is written along with the ending `</PROPERTY>` XML statement. The *%volser%* variable is cleared and will be set later if a volser is available.

`:NAME` is the beginning of the volume data and the name of each volume is written in the name attribute of the `<PROPERTY name= >` statement.

`:SAM` provides information about the Storage Allocation Method for the volumes and has the values `"Standard, ESE, TSE"`

`:ID` is the volume ID.

`:RANK` is the default routine for collecting the rank information.

The `<Rank>` statement is a bit different because the rank name is used as the data, and the `EXT= attribute` is the number of extents that is used by the volume in that rank. This results in the Excel XML parser formatting the data correctly. The volume can be allocated across many ranks so there is a line for each rank in the final Excel table. The XML parsing automatically fills in the redundant data for the volume that is associated with that rank.

The complete batch file

Example 11 is the entire batch file. You can use the copy and paste function to create a working copy of this example if you want.

Example 11 The complete batch file

```
@setlocal
@set DEV=
REM a single parameter is considered the name of a profile file
@if "%1"=="" goto end
@if "%2"=="" ( for %%f in (profile\%1.ini) do @call :GetVol %%f -cfg %%f
) else ( @call :Getdev %* )
@goto :EOF

REM Ask for the -dev DSCLI command parameter
:Getdev
@set /p DEV= "Enter DevID: IBM.2107-75xxxxx" ^>
```

```
set DEV=-dev %DEV%
REM Run the data collection process
:GetVol
set file1=%~n1_vol.xml
set file2=%~n1_rank.csv
set parms=%2 %3 %4 %5 %6 %7 %8 %9
@set tmp1=temp%time:~6,2%%time:~9,2%.txt
@set tmp2=temp%time:~6,2%%time:~9,2%.lst
@call :array %parms%
@echo ^<IRETURNVALUE^> >%file1%
@call :cmds fb
@call :cmds ckd
@echo ^</IRETURNVALUE^> >>%file1%
@del %tmp1% %tmp2%
@goto :EOF

REM Either FB or CKD commands are created
:cmds
@if EXIST %tmp2% @del %tmp2%
dscli %parms% ls%1vol %DEV% -bnr off -hdr off -s -eam managed 1> %tmp1%
@for /F "eol=C" %%V in (%tmp1%) do @echo show%1vol %DEV% -rank %%V >>  %tmp2%
@if EXIST %tmp2% ( @call :list %1 ) else ( @type %tmp1% )
@goto :EOF

REM Use the list of volumes to create DSCLI commands
:list
dscli %parms% -script %tmp2% | findstr "^R... ID Name volser sam " 1> %tmp1%
@set P=
@echo ^<INSTANCE VOLTYPE="%1Volume"^>  >>%file1%
@for /F "tokens=1,2" %%A in (%tmp1%) do @call :XML %%A %%B
@if defined P ( @echo ^</PROPERTY^> >>%file1% )
@echo ^</INSTANCE^> >>%file1%
@goto :EOF

REM Parse the volume information and create XML statements
:XML
@if %1==Name   ( @call :NAME %* ) ELSE (
@if %1==ID     ( @call :ID %*   ) ELSE (
@if %1==sam    ( @call :SAM %*  ) ELSE (
@if %1==volser ( @set volser=%2 ) ELSE ( @Call :RANK %* ) )))
@goto :EOF
:NAME
@if defined P ( @echo ^<VOLSER^>%volser%^</VOLSER^>^</PROPERTY^> >>%file1% ) else
set P=1
@set volser=
@echo ^<PROPERTY NAME="%2"^> >>%file1%
@goto :EOF
:SAM
@echo ^<SAM^>%2^</SAM^> >>%file1%
@goto :EOF
:ID
@echo ^<ID^>%2^</ID^> >>%file1%
@goto :EOF
:RANK
@echo ^<RANK EXT="%2"^>%1^</RANK^> >>%file1%
```

```
@goto :EOF

REM Collect the Rank type information to be used as a lookup table
:array
@if EXIST %file2% @del %file2%
dscli %parms% lsarray %DEV% -fmt delim -delim , -state assigned -l 1> %tmp1%
@for /F "eol=C tokens=6,7,8,9 delims=," %%R in (%tmp1%) do @echo %%R,%%U,%%S,%%T
>> %file2%
@goto :EOF

REM no parameters will print a help message
:end
@echo enter a profile name or a nickname and DSCLI parameters
@echo %0 nickname -hmc1 xxxxx -user uuuuu -passwd ppppp
```

Volume information processing

The `Get_Vol_Data` batch file produces two files that contain all the data that is needed to determine the volume extent allocations on the configured ranks. The `vol.xml` file lists all the managed volumes that might have extents that are moved by Easy Tier. The `rank.csv` file lists the disk class for each rank. You can easily combine these two files to create an Excel table. The volume information can be filtered to produce various reports by using pivot tables.

Importing the vol.xml file

The next several pages describe a way to use a spreadsheet program, such as Microsoft Excel, to process the XML table and the CSV file to create a single sheet with this information:

- Volume type
- Name
- Volume ID
- Allocation method
- Rank
- Extent count
- Volser
- Disk class

Follow these steps to import the file:

1. You simply open the file with Excel and accept the defaults for the import options. Figure 3 shows the file named `Example_vol.xml` in Windows Explorer.

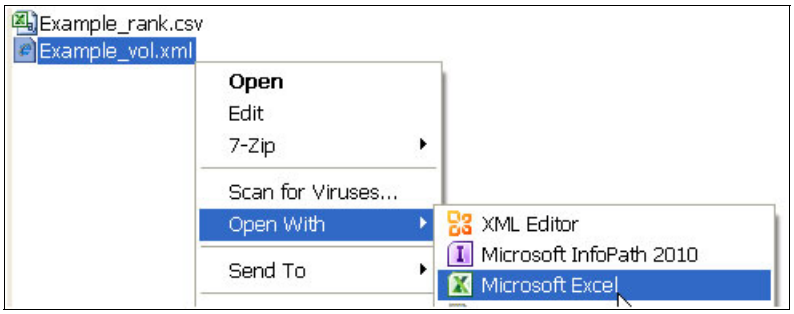


Figure 3 Windows Explorer file list open

2. Right-click the file name to display the context-sensitive menu bar that you use to open the XML file with Microsoft Excel, as shown in Figure 4. Confirm that you want to open the file as an XML table. Select the option and click **OK**.

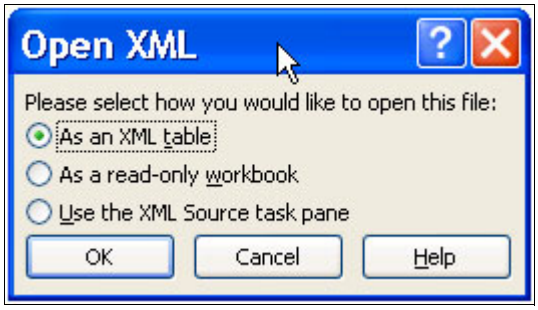


Figure 4 Opening as an XML table

3. There is no XML schema that is associated with this XML file. As seen in Figure 5, Excel prompts you to ensure that it can create a schema that is based on the source data. Click **OK** and continue.

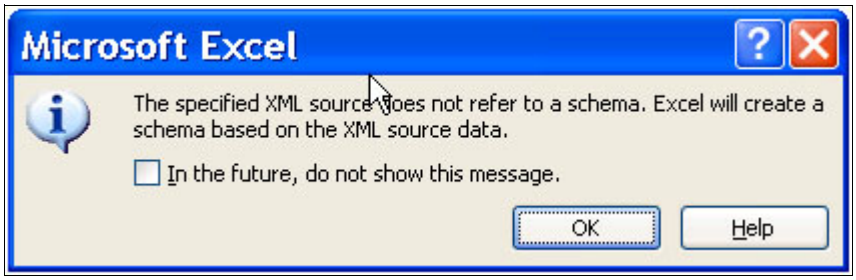


Figure 5 No XML schema is required

4. An Excel table is created with all of the volume information. Although the rank name is shown, the disk class of the rank is missing. This information is contained in the `rank.csv` file and can easily be added to the table. Select the **Data** tab → **Get External Data** → **From Text**. Then, select the `rank.csv` file as the file to import.

5. Figure 6 shows a sample of the menu bar option to import data from text.

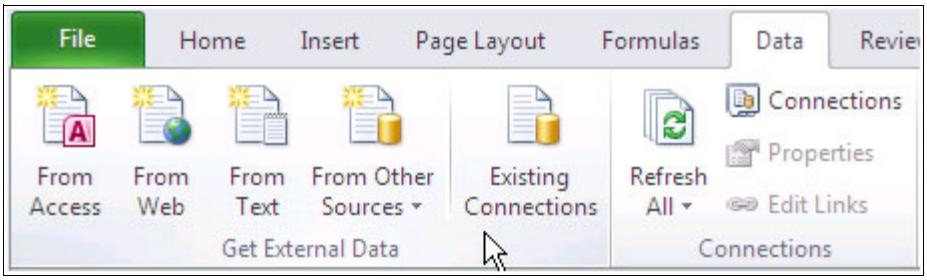


Figure 6 Excel import the rank CSV file

6. Select the associated file name as shown in the example in Figure 7.

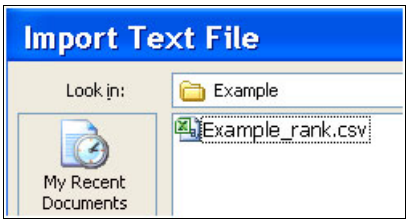


Figure 7 Selecting the CSV file

7. When importing text, Excel prompts whether the format is delimited or fixed width. In our case, we are using a comma-delimited file. The dialog that is shown in Figure 8 is where you select the option **Delimited**.

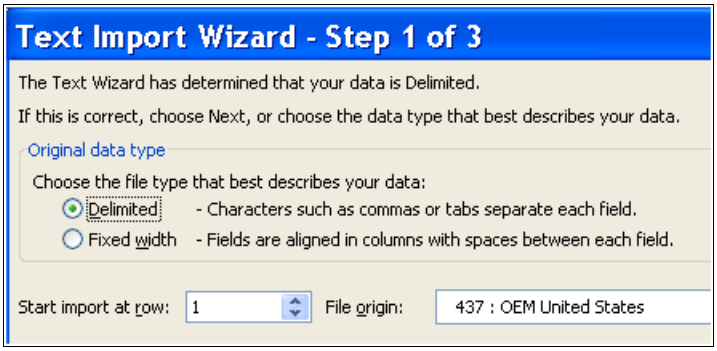


Figure 8 Text import options

8. In the next dialog, which is shown in Figure 9, you can select the delimiter character. We selected the comma in our script (you might choose something else). Select the check box next to **Comma** and click **OK**.

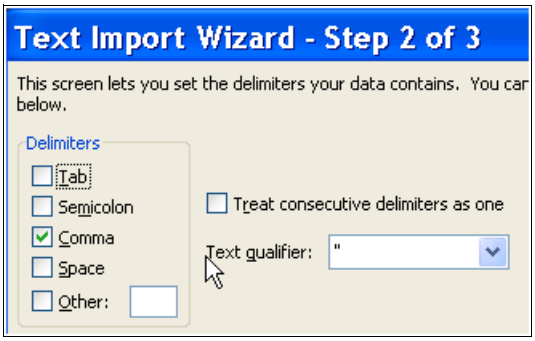


Figure 9 File that is created is comma delimited

9. When you import data, Excel provides the option to select where you want to place the imported data. In our example, we selected the existing worksheet and specific cell location. Figure 10 shows that option.

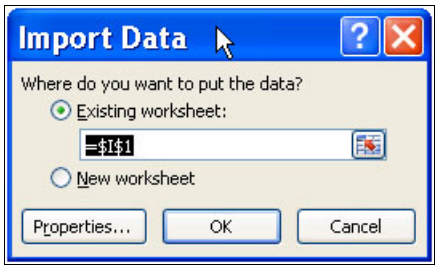
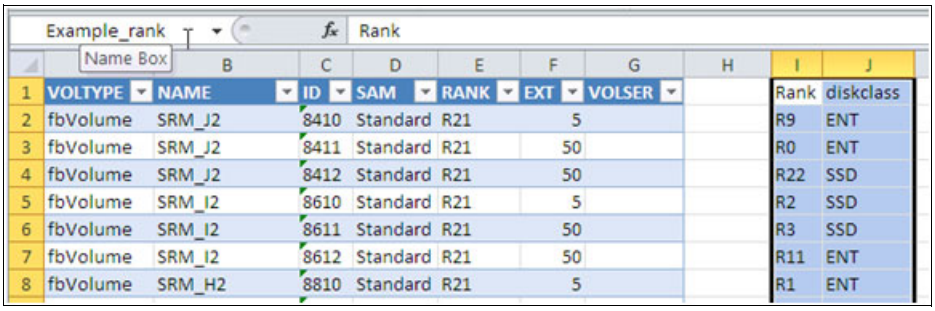


Figure 10 Cell location selection

10. The rank information is placed in the spreadsheet as a named range. This name can be used in a lookup formula to add the rank disk class to the volume table. Figure 11 shows the result from the import function.



	B	C	D	E	F	G	H	I	J
1	VOLTYPE	NAME	ID	SAM	RANK	EXT	VOLSER		Rank	diskclass
2	fbVolume	SRM_J2	8410	Standard	R21	5			R9	ENT
3	fbVolume	SRM_J2	8411	Standard	R21	50			R0	ENT
4	fbVolume	SRM_J2	8412	Standard	R21	50			R22	SSD
5	fbVolume	SRM_I2	8610	Standard	R21	5			R2	SSD
6	fbVolume	SRM_I2	8611	Standard	R21	50			R3	SSD
7	fbVolume	SRM_I2	8612	Standard	R21	50			R11	ENT
8	fbVolume	SRM_H2	8810	Standard	R21	5			R1	ENT

Figure 11 Adding the rank information to the Excel table

11. The next step is to add a column to the table (Figure 12).

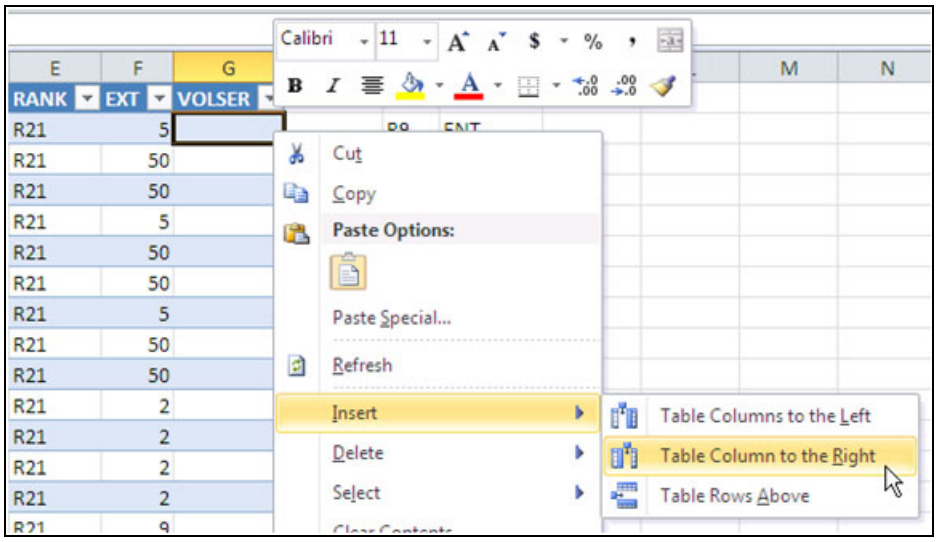


Figure 12 Adding a column to the table

12. Then, add a heading for the column and format the cells as **General** so that the formula can be evaluated (Figure 13).

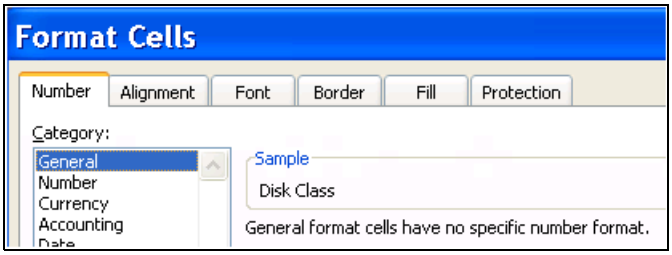


Figure 13 Formatting the new column as General

13. Insert a vlookup formula (Figure 14) that uses the rank name to refer to the rank data in the named range. Specify an exact match and return the data in column 2, which is the disk class.

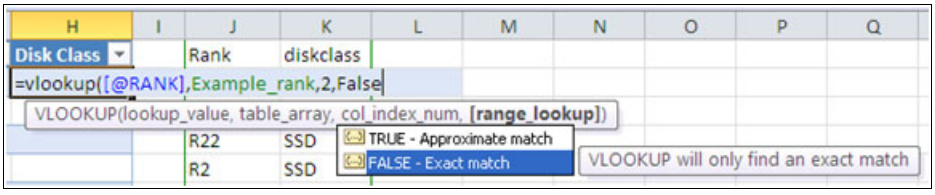


Figure 14 Adding the Excel =vlookup function

Create reports

The disk class is now available from the volume information table that is shown in Figure 15. You can sort and filter easily by using the table header drop-down menus. For example, you can select all ESE volumes and sort the list by rank name.

VOLTYPE	NAME	ID	SAM	RANK	EXT	VOLSER	Disk Class
fbVolume	TestESE_8E00	8E00	ESE	R21	9		NL
fbVolume	TestESE_8E01	8E01	ESE	R21	8		NL
fbVolume	TestESE_8E02	8E02	ESE	R21	8		NL
fbVolume	TestESE_8E03	8E03	ESE	R21	8		NL
fbVolume	TestESE_8E04	8E04	ESE	R21	8		NL
fbVolume	TestESE_8E05	8E05	ESE	R21	8		NL
fbVolume	TestESE_8E06	8E06	ESE	R21	8		NL
fbVolume	TestESE_8E07	8E07	ESE	R21	8		NL
fbVolume	TestESE_8E08	8E08	ESE	R21	3		NL
fbVolume	TestESE_8E09	8E09	ESE	R21	2		NL
fbVolume	TestESE_8E0A	8E0A	ESE	R21	2		NL
fbVolume	TestESE_8E0B	8E0B	ESE	R21	2		NL
fbVolume	TestESE_8E0C	8E0C	ESE	R21	2		NL
fbVolume	TestESE_8E0D	8E0D	ESE	R21	2		NL
fbVolume	TestESE_8E0E	8E0E	ESE	R21	2		NL
fbVolume	TestESE_8E0F	8E0F	ESE	R21	2		NL

Figure 15 The completed report for each volume

The number of extents that a volume allocated in each disk class can be reported with a pivot table. Excel performs the math.

You can also filter the data in various ways. Select a cell within the volume table and select the **Insert** tab. Click **PivotTable** to create a pivot table in a new sheet. The volume table is automatically selected as the input range. Figure 16 illustrates the steps to create a pivot table.

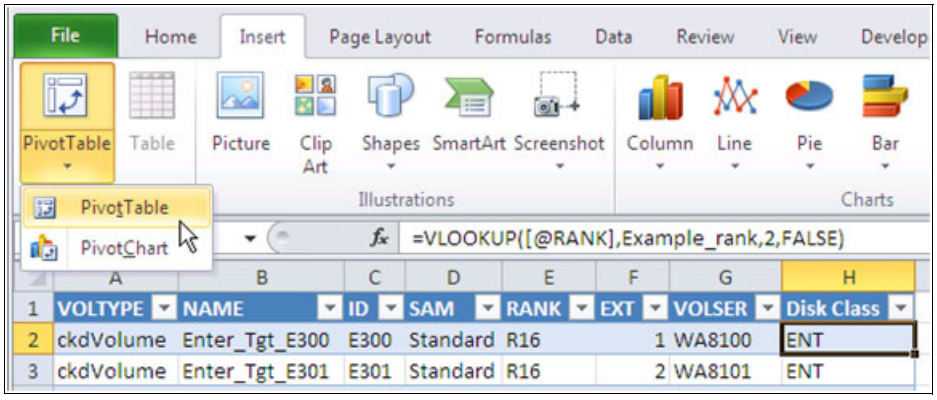


Figure 16 Creating an Excel pivot table

Follow these steps:

1. Inserting a new pivot table prompts you for the location of the table or range. Because we are using a table (due to the XML coding), you can select **Table1**, choose to place it in a **New Worksheet**, and click **OK**. Figure 17 shows an example of the Create PivotTable option panel.

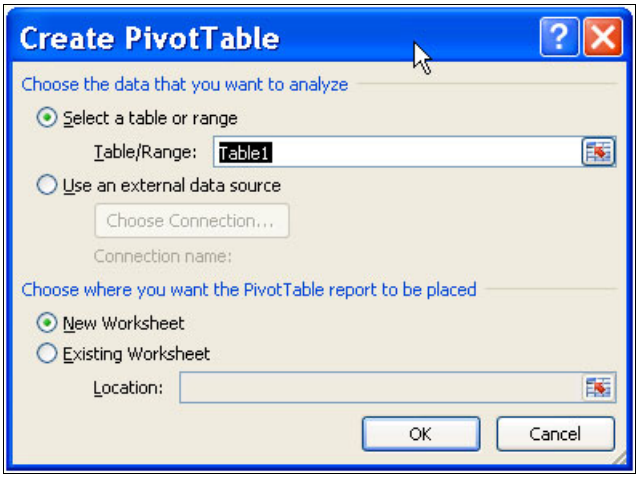


Figure 17 Using the table that you created

2. Add the **Name** field to the pivot table and display the **Sum of EXT**. Add **VOLTYPE** as a report filter and add **Disk Class** as a legend field (column label). Other fields can be added as needed, such as adding **SAM** (Storage Allocation Method) as a filter or a legend field (column label). The example pivot table that is shown in Figure 18 displays the allocation totals for the FB volumes by disk class and allocation method.

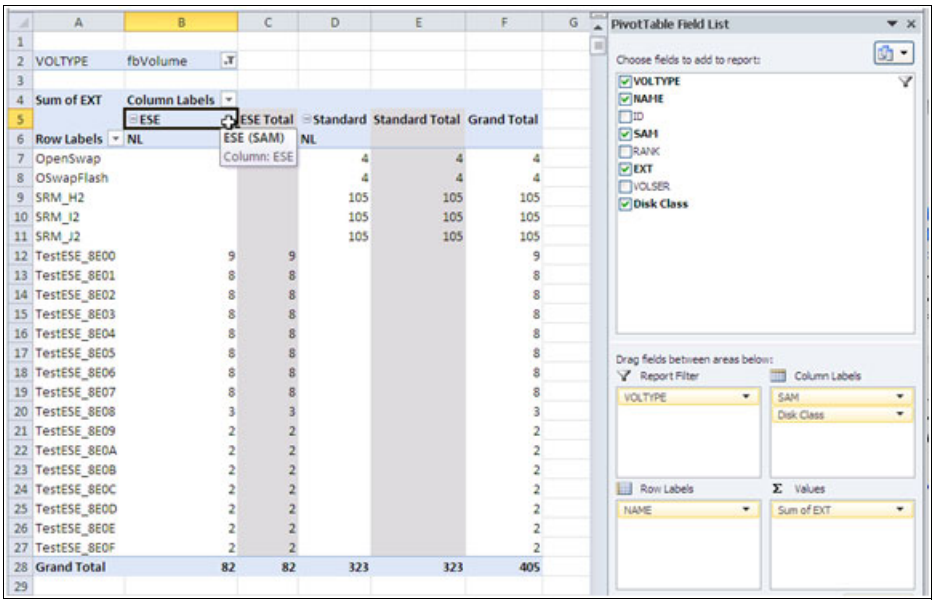


Figure 18 Sample data manipulation in Excel

3. You can use various label filters to select specific volumes, for example, all the volumes that begin with "SRM". These options are shown in Figure 19, Figure 20, and Figure 21 on page 20.

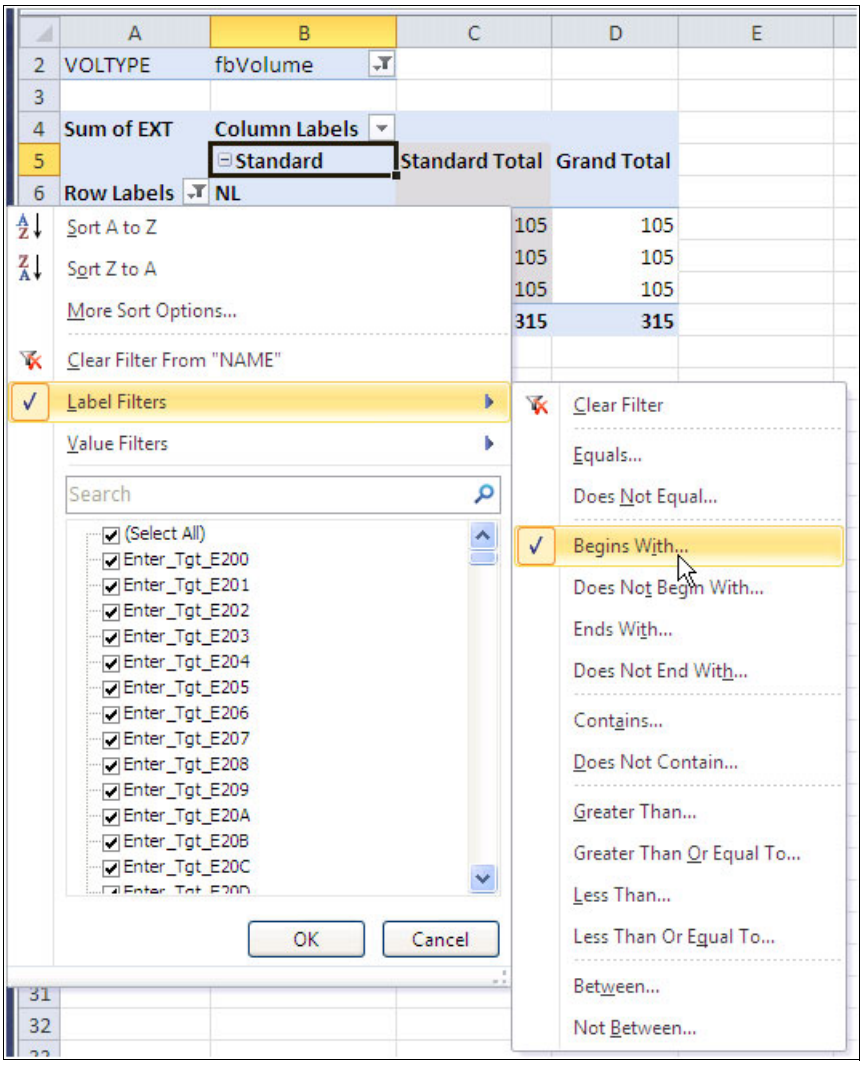


Figure 19 Optional filters

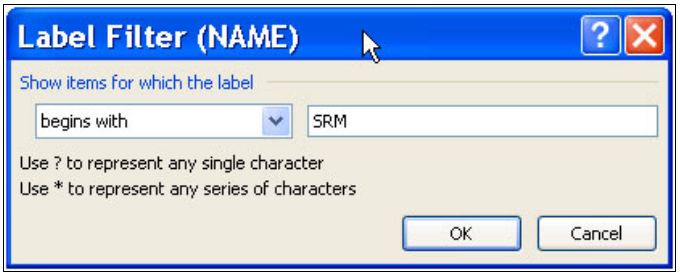


Figure 20 Filter with names that begin with SRM

	A	B	C	D
2	VOLTYPE	fbVolume		
3				
4	Sum of EXT	Column Labels		
5		Standard	Standard Total	Grand Total
6	Row Labels	NL		
7	SRM_H2	105	105	105
8	SRM_I2	105	105	105
9	SRM_J2	105	105	105
10	Grand Total	315	315	315
11				

Figure 21 Filtering results

By using pivot tables, you can also *group* various rows to view the data differently with summary totals by group (Figure 22).

Sum of EXT	Column Labels				
	ESE	ESE Total	Standard	Standard Total	Grand Total
Row Labels	NL		NL		
Group1					
OpenSwap			4	4	4
OSwapFlash			4	4	4
Group2					
SRM_H2			105	105	105
SRM_I2			105	105	105
SRM_J2			105	105	105
Group3	82	82			82
Grand Total	82	82	323	323	405

Figure 22 More options to filter and summarize

Select the row values for each group, right-click, and select **Group**, as shown in Figure 23. Groups can be easily expanded or collapsed to hide the details and make it easier to read.

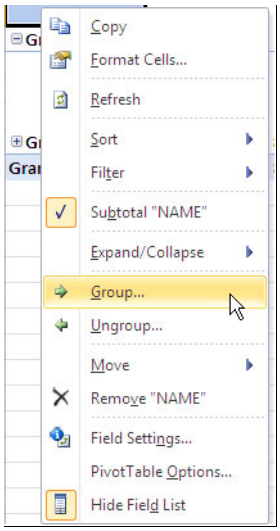


Figure 23 Example of grouping

The team who wrote this paper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Hank Sautter is a Consulting I/T Specialist with IBM Americas Advanced Technical Skills. He has 20 years experience with IBM System 390 and IBM disk storage hardware and Advanced Copy Services functions while working in Tucson, Arizona. His previous years of experience include IBM Processor microcode development and S/390® system testing while working in Poughkeepsie, NY. He has worked at IBM for 33 years. Hank's areas of expertise include enterprise storage performance, disaster recovery implementation for large systems, and open systems. He writes and presents on these topics. He holds a BS degree in Physics.

Paul Spagnolo is an Executive I/T Specialist with IBM Americas Advanced Technical Skills. He has 13 years experience working with IBM storage products, most recently focusing on IBM enterprise disk storage products. Before joining the storage team, Paul was a Systems Programmer who supported large systems and networking. Paul has worked for IBM for 23 years and his areas of expertise include storage performance, business continuity, and replication technologies. Paul teaches and writes on these and other topics. He holds a BS in Computer Science and a Masters of Business Administration.

Thanks to the following people for their contributions to this project:

Bertrand Dufrasne
International Technical Support Organization, San Jose Center

Dale Anderson
IBM Tucson

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author — all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:

http://www.linkedin.com/groups?home=&gid=2130806

- Explore new IBM Redbooks® publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4916-00 was created or updated on December 6, 2012.

Send us your comments in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks