CoolSpools

By Ariadne Software

DATABASE USER GUIDE

V7R1

Version 2.0
April 2024

Table of Contents

Introduction

CoolSpools Database (formerly called "**Slipstream**") enables companies and organizations that use the IBM system i platform deliver high-quality information to their customers and users more quickly and more simply than was possible previously.

CoolSpools Database does this by providing tools for exporting system i database tables as PC-format stream files, such as Excel spreadsheets, XML documents, CSV files, HTML web pages and fixed-format flat text files. As of V7R1M0, it also provides a way of extracting information from Excel-format spreadsheets into system i database format.

CoolSpools Database is option 4 of the CoolSpools suite of modules. In order to use CoolSpools Database, you need only to install the CoolSpools Base module and CoolSpools Database, but if you install other CoolSpools options, CoolSpools Database will work with them:

- if you have CoolSpools Spool Conversion module (option 1) installed, you can output to PDF format from CoolSpools Database (e.g. run a piece of SQL and save the output as a PDF)
- if you have CoolSpools Email module (option 2) installed, you will be able to distribute the files you create (e.g. Excel and XML files) as email attachments after you create them too
- if you have the free CoolSpools NetServer module (option 5) installed, you can easily create and manage file shares to allow PC access to files created by CoolSpools Database and stored on the system i

The system i database (UDB for system i) is an advanced, function-rich, easily managed relational database allowing large quantities of data to be stored securely and retrieved and updated rapidly and efficiently. However, today, many of our customers and users want to be able to access and manipulate the data held in that database using tools such as spreadsheets, PC databases and Business Intelligence tools. Enabling their customers and users to access data held on the system i in the form in which they need it is a challenge that many system i sites face on daily basis.

One frequently used approach to this problem is to allow users to import data from the system i database into PC application using tools such as iSeries Access file transfers and ODBC queries.

However, these solutions have a number of drawbacks, mainly related to the fact that they are "pull" technologies. In other words, an iSeries Access file transfer or an ODBC query has to be initiated from the PC end and cannot easily be driven from the system i. This has a number of unfortunate consequences:

- Your users have to be in the office to press the buttons. This means that these types of operations typically have to be run during the day. This imposes an unnecessary load on your system i and your network when they are at their busiest. It also means that your users waste time preparing data for use. Wouldn't it be better if the data was ready waiting for them when they arrive in the office?
- Your users have to push the buttons at the right time. What happens if the job that creates or updates the data they want hasn't finished running yet? Do they get yesterday's data instead? Or might data get lost, corrupted or duplicated? Wouldn't it

be better if you could integrate and schedule the preparation of PC data with your batch processes so that the PC data isn't created until the system i is ready?

- Your users have to make sure to push the right buttons! Running regular file transfers and queries can be a repetitive, tedious, error-prone chore. Human beings are fallible and mistakes will inevitably occur from time to time. Wouldn't it be better if these processes were automated?

CoolSpools Database by contrast is a "push" technology. It addresses these problems by providing a set of integrated tools to enable PC-format data to be extracted from the system i database entirely on the system i, without the need for a PC or any user intervention. This means that the entire process can be fully automated and controlled programmatically, and can be easily integrated with your existing applications and batch schedules.

CoolSpools Database can process any of the following as input sources:

- system i physical files
- system i logical files
- system i DDM files
- ad hoc SQL query statements
- SQL query statements stored in a source file member
- Query/400 queries
- Query Management (QM/400) queries
- Excel Spreadsheet

CoolSpools Database can output the results or contents of these input sources in the form of stream files in any of the following file formats:

- **Excel**. As of V7R1M0, the flavors of Excel supported are:
  - BIFF 8 (Excel 97-2003 .xls format)
- Office Open XML (Excel 2007 .xlsx)
- **XML**
- **HTML**
- **Delimited ASCII text**, including:
  - Comma Separated Variable (CSV)
  - Tab Separated Variable (TSV)
  - other delimited text formats (e.g. semicolon)
- **Fixed-width text**

This means that you can now execute a piece of SQL and save the output directly as an Excel spreadsheet, simply by running one command on your system i!

Similarly, you could run a query and save the output as an XML document or ASCII CSV file, ready to be loaded into a Business Intelligence tool. Or if you want to display the latest sales figures on your Intranet, rather than printing a paper report as you might have done

before, save the output from you sales reporting program as a temporary database file, and use ***CoolSpools Database*** to convert it to HTML format.

Stream files created by ***CoolSpools Database*** can be saved anywhere in your system i's IFS (Integrated File System). This means they can either be held on the system i itself (using the root or QDLS file systems), or transferred seamlessly to a Windows server (using the QNTC file system) or to another system i (use the QFileSvr.400 file system). See the section entitled "Using the IFS" for details of how to transfer data directly to an NT server when running ***CoolSpools Database***. ***CoolSpools Database*** can also send its output to another system (e.g. a UNIX machine) using FTP.

When processing an existing system i physical or logical file as input, ***CoolSpools Database*** provides a number of additional functions which make delivering high-quality, easily accessible data to your users even easier:

- **Record sorting** – records can be sorted by multiple key fields in ascending or descending sequence
- **Record selection** – records can be selected or omitted based on criteria that you define in a familiar format
- **Field inclusion/exclusion** – you can include all fields in the file being processed, specify a list of fields to include or a list of fields to exclude
- **Field sorting** – you can specify the order in which fields are to be presented by listing them in the order you want them
- **Formatting** – date, time and number formats can be managed and edit codes and edit words are reflected in Excel formatting.
- **User-defined named styles** allowing you to specify formatting of Excel columns and HTML table data such as colors, fonts etc
- Conditional formatting allowing you to specify different styles for Excel cells based on rules you define (e.g. your data could appear green if the value is below $1000, orange if between $1000 and $5000, red if over $5000 Applications

CoolSpools Database comprises a series of related commands which can take any system i database file (i.e. DB2 UDB for system i file or table) and convert it to formats that can be used by popular PC applications such as spreadsheets, databases and browsers. Because ***CoolSpools Database*** is command driven, it is easy to integrate it with your batch applications: all you need to do is to insert a call to the relevant command into a CL program or via QCMDEXC in an RPG or COBOL program.

Here are a few possible ways ***CoolSpools Database*** can help you deliver data to your users more efficiently and reliably.

Automated distribution of reports and documents

Still sending reports to your users on paper? Still distributing data to customers via snail mail?

Wouldn't your users rather have the same information in electronic form so they can load it into their PC spreadsheets without having to re-key it? ***CoolSpools Database*** can help you modify your applications so you can deliver data to your users in the form they need. All

that is needed is that you change your programs to output to a database file rather than to a spooled file. You can then use ***CoolSpools Database*** to create an Excel spreadsheet, CSV or HTML or fixed text file from your database file. Or if the application uses a Query/400 or QM query, all you need to do is specify that query as the input to a ***CoolSpools Database*** command.

Files created by ***CoolSpools Database*** can be e-mailed to users, colleagues and customers, rather than sent out on paper, making their lives easier whilst saving you money and simplifying your business processes.

Integration with your normal batch jobs

Imagine you have Excel power-users in your finance department who need to analyze your company's sales figures from the previous week each Monday morning. To get the information from the sales database into Excel, your users could perform an iSeries Access file transfer or run an ODBC query. "Isn't that the easiest way?" you might ask. However, ***CoolSpools Database*** offers some significant advantages over these methods:

- Data is ready for your users as soon as they want it. Run your conversions overnight and in the morning your users simply need to connect to the system i and open a file from the IFS or a PC server. They don't need to waste time running file transfers or ODBC queries.
- Data can be prepared when convenient to you. Conversions can be easily integrated with your normal overnight or weekend batch jobs, and therefore run out of hours when the load on your system and network is lowest. There's no need for a PC at all and no need for complicated, error-prone system i-to-PC communications to trigger file transfers or ODBC queries.
- Conversions don't take place until the data is ready. Run the conversion step immediately after the step that creates the data to ensure that the two occur in the right sequence.
- Since the conversion parameters are built into your applications, you can be sure that the conversion will be done correctly every time, automatically. You don't have to rely on your users to run the transfer or the query correctly each day or each week, and they have to remember what buttons to press and which selections to make each time.

For example, rather than having your users run ODBC queries against the sales database on a Monday morning, you could run a job over the weekend to extract information about sales made the previous week. This could be as simple as running a Query/400 query. With ***CoolSpools Database*** you can run the query and save the output as an Excel, XML, CSV file, HTML or ASCII text file in your IFS. Your users only need a network drive assigned to the right system i and the necessary permissions to access the data. They can simply load their spreadsheet application on Monday morning, open the file previously created using CoolSpools Database, and away they go.

The beauty of this approach is that the system i controls, schedules, and manages the creation of the PC-format data entirely. It saves your users time and effort, saves you worries about whether the users will get their file transfer right or run their ODBC query at the wrong time, and saves the system i from unnecessary workload during office hours.

Data Sharing

Often, many of your users will want the same data. This can often mean that different users will be running the same or very similar queries or file transfers against your database all in the same day. This is very inefficient. Why not run that query for them once, overnight, and save the results as an Excel, CSV or HTML file to a central server so the information can be shared by everyone?

Alternatively, you could publish these files on your website for customers worldwide to see, or on a secure Intranet or Extranet.

Stream files created by ***CoolSpools Database*** can be saved anywhere in your system i's IFS (Integrated File System).

File Formats

Let's have a closer look at how ***CoolSpools Database*** achieves all this.

CoolSpools Database does this through a set of simple commands such as CVTDBFXL (Convert Database File to Excel) and CVTDBFXML (Convert Database File to XML), CVTDBFXLSX (Convert Database File to Excel 2007-2013). These commands take a system i database file (physical or logical) and convert it to a stream file in a particular PC file format (e.g. Excel for CVTDBFXL or CVTDBFXLSX and XML for CVTDBFXML). The stream file can be saved anywhere in the ***IFS (Integrated File System). See the section*** entitled "Using the IFS" below for details.

Please note that it is not necessary for the stream file to be stored either temporarily or permanently on your system i disk: the IFS provides the functionality to write the stream file directly to an NT or UNIX server, if you prefer to store your files there for your users to access. Alternatively, store the stream files locally in the root file system of your system i, and anyone with the right authorities and a network drive attached can read them

You can choose several different formats for your files, depending on your particular requirements.

Excel Format

If you use the ***CVTDBFXL*** command, you can create an Excel (™) spreadsheet from your database file. The default is to create a BIFF 8 file (compatible with Excel 97, Excel 2000, Excel XP and above). Alternatively, you can choose to create an Excel 2007 .xlsx (Office Open XML) file.

If you use the ***CVTDBFXLSX*** command, you can only create an Excel (™) 2007 or later .xlsx (Office Open XML) file.

Files created by ***CVTDBFXL*** in Excel format can be opened directly by any application with supports the Excel file format, including Microsoft Excel but many others besides, such as Lotus 123 and MS Works spreadsheet.

Files created by ***CVTDBFXLSX*** in Excel Office Open XML) format can be opened directly by any application which supports the Excel file format, including Microsoft Excel but many others besides

XML Format

The ***CVTDBFXML*** command converts database files to XML format. You can select, for any field in the database file, whether it should be converted to an attribute or an element in the XML file.

The ***IMPXMLDBF*** command imports data from an XML document to one or more database files. Mapping of XML elements and attributes to database files and fields is defined using the commands ***BLDXMLDBF*** and ***WRKXMLDBF***.

Delimited ASCII Text

The ***CVTDBFCSV*** command creates delimited ASCII text files, typically, but not limited to, CSV (comma-separated variable file). This format is ideal for loading reports containing columns of numbers into a spreadsheet, Business Intelligence tool or other application for further manipulation. ***CVTDBFCSV*** can use any field delimiter you like (by default a comma, but also tabs, semicolons, blanks etc.) and any string separator you specify (by default a double quote "), allowing you to generate files in the precise format required by your PC application or national environment.

HTML (Hypertext Mark-up Language)

If your want to view your data in a browser, such as Microsoft ® Internet Explorer or Mozilla Firefox, ***CVTDBFHTML*** is the command you need.

HTML is the language in which web pages are normally written. ***CVTDBFHTML*** can convert your database files to a formatted HTML table, which provides a neat way of displaying your data on the Internet, or on your company Intranet or Extranet.

Fixed ASCII Text

CVTDBFTXT will convert your system i database files to a simple ASCII text file with fixed-width data columns without delimiters or separators. This format can be readily processed by user-written applications so long as the file layout is known in advance, and can be useful for exchanging data with business partners in EDI operations.

PDF

If you have the CoolSpools Spool Conversion option installed (product option 1), ***CVTDBFPDF*** will run the query you specify or an auto-generated query against the database file you specify and convert the output to a PDF file.

What's new in CoolSpools Database V7?

Highlights include:

- **New format-specific command**
  CVTDBFXLSX
  - Excel 2007 and above (.xlsx)
  - DFNSTYLES parameter removed. The command interface has been simplified by removing the complex DFNSTYLES parameter. The same results can be achieved, more flexibly and simply, using styles created with CRTSTLDFN, or temporarily defined or overridden using OVRSTLDFN.
  - Heading lines defined on CVTDBFXLSX can now be 256 characters – previously 128
  - On the FromFile parameter the spool files displayed may be reduced by using F14=Subset. A user can save a subset selection for use with CVTSPL* commands
  - Far more flexibility on the Excel options, now including

- **Import Excel to database format**
  IMPXLDBF
  - The existing command (CVTXLDBF) writes an iSeries database record for each cell in the workbook identified by the user. This command writes a database record for each row in a workbook. Only Excel 2007 and above (.xlsx) input is supported, not .xls.

- **Addrows parameter on Stream File Option**
  - Append the converted data to an existing worksheet

- **Ongoing enhancements**
  - New functionality is being developed for CoolSpools V7 on an ongoing basis and delivered within fix packs. This includes new commands (WRKXMLDBF, IMPXMLDBF) to import data from an XML document to one or more IBM i database files, and new commands (IMPCSVDBF, IMPFXDDBF) to import from delimited and fixed position text files to IBM i database files.

System Requirements

- An IBM POWER server running IBM i (OS/400) V6R1M0 or above.
- 20 Mb of IBM i disk space.
- **No** PC is required.

Upgrade Notes

Please read the following notes carefully before upgrading to Version 7 from a version of CoolSpools Database prior to Version V6R1. We advise if you have not previously installed CoolSpools V6R1 to read the Version 6 Memo to Users and take into consideration the changes that took place in Version 6.

Why V7R1M0?

CoolSpools Database was previously known as “Slipstream” and was available as both a standalone module and as part of the complete CoolSpools PLUS suite, the earlier version of which was V5R1M0. Now, CoolSpools Database is packaged as a product option (option 4) of the CoolSpools package, which has been assigned the overall version number V7R1M0. **Ariadne Software Limited continue to develop and publish new functionality on an ongoing basis, but new functionality is only being added to V7R1M0, and not to older versions.**

Licence Keys

You are entitled to upgrade to V7R1M0 of CoolSpools Database free of charge if:

- the machine on which you wish to run CoolSpools Database V7R1M0 has a valid licence for an earlier version of Slipstream or CoolSpools Database

and

- you are either in your first 12 months’ maintenance period after purchase or have paid your latest annual maintenance invoice.

If you wish to upgrade, you can simply download the software from **www.coolspools.com** and install it according to the instructions contained in the “Installation” section of this User Guide. However, if you licensed an earlier version of CoolSpools Database, you will need to request a licence key for the new version. Simply e-mail ***support@ariadnesoftware.co.uk*** and ask for your key for V7R1M0 of CoolSpools Database. Please quote your system serial number(s) and LPAR (s) in your e-mail. These are shown at the top of the WRKLICINF screen or if at CoolSpools V6 run DSPPRDINF.

Without a licence key, CoolSpools Database V7R1M0 will allow you a 30-day grace period and will then no longer run.

If you need longer than 30 days to test CoolSpools V7R1M0, we will, on request, send you a temporary licence key to extend the trial period.

If you require additional temporary licence keys to assist with testing CoolSpools V7R1M0, please contact us at ***support@ariadnesoftware.co.uk*** or if you run into any problems during your testing, please log a ticket on our Help Desk ***http://www.ariadnesoftware.co.uk/support/*** with detailed information.

Warning/Disclaimer

We recommend strongly that all production applications are re-tested thoroughly using the new version in your development environment before you switch over to running the new version in your production environment.

All CoolSpools versions are packaged as separate licensed programs and install into different libraries. This means that all versions of CoolSpools can coexist and run alongside one another on the same machine. You can switch an application from using one version to using another simply by changing the library list of the job to include the appropriate version library or by specifying a different library name when you run the command. Hence it is quite a simple matter to test your applications using the new version while continuing to run the older version in production.

Please note that while Ariadne makes every effort to ensure that CoolSpools functions in the same way with the same parameters from one version to the next, it is not possible to guarantee this. This is why you should re-test your applications against a new version before going live with it as it is possible that in some cases different parameter settings will be necessary to obtain the same results as before.
Ariadne software accepts no responsibility for any damage, expense or loss of income incurred as a result of unforeseen and unwanted effects resulting from installing new versions of its software or applying PTFs.

Minimum OS/400 Release Level

The minimum OS/400 release level required to run V7R1M0 of CoolSpools Database is OS/400 **V6R1M0.**
If you are running V5R1M0 or an earlier version of OS400, you will not be able to install V7R1M0 of CoolSpools Database.

Product Library

All product options of CoolSpools V7R1M0 install into the single product library COOLSPV7R1. This means that you no longer have to manage multiple product libraries for the separate modules that made up CoolSpools PLUS V5R1M0 (Slipstream, Communiqué, CoolSpools etc.)
You may need to change library lists in job descriptions and other system objects in order to pick up the new version of the code rather than the old.
This change of library name has the advantage that it allows you to run both V7R1M0 and the earlier versions on the same machine. You are therefore able to test V7R1M0 before swapping your production applications over to the new version, as we strongly advise you to do.

Memo to Users

Memo to Users

Please refer to the **Memo to Users** for important information about changes you need to take account of before migrating to CoolSpools Database from an earlier version of Slipstream or CoolSpools PLUS. Please note that CoolSpools V7 includes all the changes that were made in CoolSpools V6. If you are migrating from a version prior to CoolSpools V6 you need to take into consideration the changes that were made in that release. Please refer to **Memo to Users V6**

Installation

Refer to the ***Installation Guide*** for instructions.

Maintenance

Refer to the ***Maintenance Guide*** for instructions.

Where Did My Output Go?

Each of CoolSpools Database's CVTDBFxxxx commands converts an iSeries database file to a stream file in a format such as Excel, XML or HTML. Where the output is created depends on what you specify on the TOSTMF parameter of the CVTDBFxxxx command that you ran. You have a number of options which we will discuss shortly.

Normally you will want to access these stream files from a PC application such as a spreadsheet application or a browser. How you access CoolSpools Database output from your PC depends on a number of factors which we will also consider now.

The TOSTMF parameter

When you run one of the CVTDBFxxxx commands, you specify where you want the output to go and what you want it to be called on the **TOSTMF (To Stream File)** parameter.

There are 3 basic options:

- **IFS path name**

You can define an absolute or relative IFS path specifying the name of the file to be created and the directory in which it will be placed.

The IFS is a collection of file systems provided by your iSeries. Depending on which file system you select, your output may be stored locally on your iSeries' disks or remotely on another system on your network, which could be a PC, another iSeries a UNIX server etc.

Use of the IFS is explained more fully below.

The special value ***FROMFILE** (the parameter default value) tells CoolSpools Database to create a file name from the name of the database file and an appropriate extension based on the format of the file being created (e.g. .xml for a XML file, .xls for an Excel file etc.) and place it in the current directory of the job.

- ***FTP**

This tells CoolSpools Database to send the output using FTP (File Transfer Protocol) to another system running an FTP server process. This could be another iSeries, a PC server, a UNIX machine etc.

Understanding IFS path names

The IFS (Integrated File System) is a collection of file systems that your iSeries can use to store and retrieve information. Depending on which file system you choose to use, the data may be stored locally (on your iSeries' own disks) or remotely (on another system in your network).

When you enter a path name on the TOSTMF parameter, you are telling CoolSpools Database the name of the file you wish to create. You will also be telling it, explicitly or implicitly, in which file system and directory to save that file.

The path consists of four elements:

- **The Extension**

If you type a name that ends with a period (.) and then a sequence of characters, you have specified an **extension**.

For example: **.xml, .xls, .htm**

Windows and other operating systems may use this extension to determine what type of file you have created. For example, if you double-click in Windows on a file name ending in .xls, it is likely that Windows will start or switch to Microsoft Excel and open the file.

This makes it very important that you should choose an extension which is appropriate to the type of file you are creating.

For example, if you are using CVTDBFXLSX to create an Excel file, specify a file name ending in **.xlsx** so Windows recognizes that the file should be opened with Excel, but if you are using CVTDBFHTML to create an HTML file, choose a file name ending in **.htm** to ensure that Windows recognizes the file as HTML and will open it in your browser.

- **The File Name**

The part of the path name that precedes the extension is the name of the file itself. CoolSpools Database does not impose any restrictions other than the limit of 1,024 bytes for the entire path name.

Please note, however, that the syntax and rules that apply to the name will be dependent on the file system you choose. For example, the QDLS file system ("shared folders") does not allow the file name to be longer than 8 characters with an optional extension of 1-3 characters (old DOS-style 8.3 naming). Also note file names in some file systems are case-insensitive (e.g. root file system) while file names in other file systems are case-sensitive (e.g. QOpenSys).

- **The Directory Path**

You can optionally specify a directory or list of sub-directories in which the file is to be saved.

For example, if you have a directory called sales with subdirectories for each region, and then subdirectories for each year and month, you may need to specify a path such as:

sales/north/2009/nov

to indicate that the directory in which you wish to save your file is the November subdirectory within the 2009 subdirectory of the north region's subdirectory within **sales**.

- **The File System**

You can optionally specify a file system name at the beginning of the path to indicate to which file system the path refers.

Here is a list of commonly used file system names that can be used at the beginning of a path name. Note that each begins with a / (forward slash) and that the root file system is indicated by a single forward slash alone:

/	The "root" file system. This is the "default" iSeries hierarchical file system.
/QDLS	Document Library Services ("shared folders")
/QNTC	Windows NT Server file system. This file system provides access to data and objects that are stored on a Windows server

	This file system can be used to directly read data from and write data to a separate Windows server on your network.
/QOpenSys	A hierarchical file system compatible with UNIX and POSIX. Uses case-sensitive names.
/QSYS.LIB	The iSeries database. Although it is possible to save CoolSpools Database output in a database file member, this is not recommended as the data is unlikely to be easily accessed there.

You should also understand the difference between an **absolute** path name and a **relative** path name.

An absolute path name is one which explicitly defines the full location at which a file is to be saved.

For example, the path name

/sales/north/2009/nov/new_business.xml

is an absolute path name which specifies the full location of a file to be created and breaks down as follows:

/	The initial / indicates the root file system
sales	The name of the directory in the root file system
north	The name of a subdirectory within **/sales**
2009	The name of a subdirectory within **/sales/north**
nov	The name of a subdirectory within **/sales/north/2009**
new_business	The name of the file to be created
.xml	The file extension, indicating an XML file

However, if you do not enter a forward slash (/) at the beginning of a path name, your iSeries will interpret this as a **relative** path name. Relative path names are interpreted relative to the current directory of the job (similar to the current directory in Windows or DOS).

For example, if your current directory is already set to **/sales**, the path

north/2009/nov/new_business.xml

(note there is no leading /) would be interpreted relative to **/sales** and would refer to exactly the same location as the absolute path

/sales/north/2009/nov/new_business.xml

The current directory of your job can be set with the **CHGCURDIR** or **CD** commands. Often, the current directory will be set automatically for you when you sign on to the iSeries based upon the HOMEDIR (home directory) attribute of your user profile.

Assume that your user profile has HOMEDIR = **/home/john**, indicating that when you sign on the current directory should be set to the **john** subdirectory within the **home** directory of the root file system. Unless you have changed this with CHGCURDIR or CD, if you specify a relative path name, the path will be interpreted relative to your current directory **/home/john.**

For example, the relative path

reports/sales.xml

would be interpreted as referring to a file called **sales.xml** in a subdirectory called reports within **/home/john**.

You will need to enclose path names in single quotes (‘) on the TOSTMF parameter if they contain forward slashes or other special characters.

For example:

TOSTMF(new_business.xml)

is acceptable to OS/400 without single quotes, but your iSeries will insist that:

TOSTMF(‘/sales/north/2009/nov/new_business.xml’)

is entered with single quotes around the path name. When prompting the command with F4, the iSeries will enclose the path name in quotes for you if you have not already done it.

Further information on the IFS can be found at:

http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm?info/ifs/rzaaxmst02.htm

Choosing where to store your output

When it comes to deciding where to save your CoolSpools Database output, a number of factors need to be considered, for example:

- **Simplicity**

How easy is it to save files to and retrieve files from a particular IFS file system? Are the naming rules for the file system complex or restrictive?

- **Performance**

How well does that file system perform? Is saving and retrieving data from that file system quick and efficient or slow and laborious?

- **Reliability**

Will the file system always be available or is there a chance that it might be unavailable for some reason at the time when you try to save data to it or retrieve data from it?

- **Access**

What choices do you have with regards to accessing the data? How easy is it to retrieve data from the file system you choose to use using an appropriate application? For example, how easy is it to open an Excel file from a PC?

- **Management**

How easy is it to perform management functions on the files in the file system, such as backup, archiving and purging of old documents?

- **Security**

Can you ensure that only the right people have access to the documents?

- **Scalability**

Will problems occur when volumes increase?

We will now consider the various IFS file systems you are most likely to want to use according to these criteria.

Root File System

The “root” file system is in many ways the “default” IFS file system and is probably where most CoolSpools Database users choose to store their output.

You save a CoolSpools Database file in the root file system if you enter a path name on the TOSTMF parameter which does not explicitly and implicitly refer to any other file system.

Users can access files created on your iSeries in the “root” file system using network drives. For example, if your users have their I: drive assigned to the iSeries root file system, they could open a file called sales_report.xls saved in a directory called sales by opening i:/sales/sales_report.xls in Excel.

Simplicity	Excellent. The simplest and easiest to use. Long file names are supported. Not case-sensitive.
Performance	Good. Writing data locally will keep down the time taken to create the files. Speed of retrieval from a PC will depend on your network and other factors such as the power and loading of your iSeries.
Reliability	Excellent. Writing data locally means that file creation is not dependent on the availability of the network or another system.
Access	Good. Easy to access from Windows using network drives.
Management	Good. Can be backed up with the iSeries. Can be managed from the iSeries command line or from Windows using a network drive.
Security	Excellent. iSeries security applies.
Scalability	Moderate. High cost of iSeries disks a possible issue.
Comments	Recommended unless other factors dictate otherwise

QDLS File System

The QDLS or “shared folders” file system implements a DOS-style method of saving PC files and other documents on the iSeries’ own disks. It is really a legacy file system providing backwards compatibility for older applications written for the S/38 or versions of OS/400 that pre-date the availability of the IFS (OS/400 V3R1M0).

You save a CoolSpools Database file in the QDLS file system if you enter a path name on the TOSTMF parameter which starts /QDLS or if you use a relative path name and your current directory path starts /QDLS.

Users can access files created on your iSeries in the QNTC file system using network drives. For example, if you users have their I: drive assigned to the iSeries root file system, they could open a file called REPORT.xml saved in a shared folder called SALES by opening i:/QDLS/SALES/REPORT.xml in Adobe Acrobat.

Simplicity	Good. Familiar to long-standing users of S/38 and AS/400 applications. Not case-sensitive. Naming limited to DOS-style 8.3 conventions so long file names will cause errors.
Performance	Poor compared to the “root” file system.
Reliability	Excellent. Writing data locally means that file creation is not dependent on the availability of the network or another system.

Access	Good. Easy to access from Windows using Network drives
Management	Good. Can be backed up with the iSeries. Can be managed from the iSeries command line or from Windows using a ISeries Access network drive.
Security	Excellent. iSeries security applies.
Scalability	Moderate. High cost of iSeries disks a possible issue.
Comments	Use the “root” file system instead.

QNTC File System

The QNTC file system is the iSeries implementation of Windows network neighborhood. It allows you to write to and read from files stored on a Windows server.

You save a CoolSpools Database file in the QNTC file system if you enter a path name on the TOSTMF parameter which starts /QNTC or if you use a relative path name and your current directory path starts /QNTC. The file system name /QNTC should be followed by the name of the server, then the name of the shared resource on that server (e.g. the shared directory name) and then the path within that shared directory.

Imagine you have a Windows server which is known to the network as **server1**. On that server there is a directory called **sales** which is shared under the name **sales**. Within that shared directory there is a subdirectory called **2009**. If you have QNTC configured and your security settings allow it, you can save a file called november.xml in that subdirectory from the iSeries by specifying the path name:

/QNTC/server1/sales/2009/november.xml

The QNTC file system can be quite difficult to configure and manage, but once you have it running it can provide a very effective means of creating CoolSpools Database output directly on a Windows server in your network.

Please note in particular that the iSeries user profile of the job which accesses QNTC must be the same name and have the same password as a user id that Windows networking recognizes.

Further information on QNTC is at:
http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm?info/ifs/rzaaxmstqntcfs.htm
http://www-1.ibm.com/support/docview.wss?uid=nas1aea450153eebf8ff8625670f0072550f&rs=110
http://www.itjungle.com/fhg/fhg031704-story04.html
http://www.itjungle.com/mgo/mgo111903-story02.html

Once you have saved your files on a Windows server in your network, users can then access files created with CoolSpools Database on that Windows server using Windows networking. For example, if they have their F: drive assigned to a directory called sales on that server, they could access a file called sales_report.xml in that directory simply by opening file F:/sales_report.xml.

Simplicity	Can be difficult to set up and manage. Once files are saved on the Windows server, access should be very simple.
Performance	Creating files across the network on the PC server may be slow. Retrieval of files once created should be very fast but will depend on the server and network loading.

Reliability	Creating files across the network on the PC server requires both the server and the network to be available at the time.
Access	Easy to access from Windows using Windows networking.
Management	Good. Will need to be backed up with your Windows server.
Security	Good. Windows security applies.
Scalability	Good. Low-cost PC disks can be used.
Comments	If you prefer to store your files on a Windows PC server rather than on the iSeries, this is an ideal solution if the initial setup issues can be overcome and you can ensure that the PC server will be available to the iSeries when it needs to create the files.

Typical Solutions

When implementing CoolSpools Database, it is important to make the right choices about where you will save the files you create and how you will access them.

Here are a few typical approaches that users have successfully implemented in the past.

- **Save the files in the iSeries "root"**

This is a really simple, easy and reliable method.

To save a file in the "root" file system, you just specify a path name starting with a forward slash /.

You can open files saved in the root file system from your PC applications (Acrobat, Excel, Word etc.) by using ISeries Access network drives to open the file just as you would a file saved locally on your PC or on a Windows or UNIX server.

The only real downside of this approach is that the files occupy space on your iSeries disks, which can be expensive compared to PC disks.

For further information on configuring the iSeries so your users can connect from their PC using iSeries Access network drives, see
publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/info/rzaij/rzaijconnetas.htm

For further information on configuring your users' PC so they can connect to the iSeries using iSeries Access network drives, see
http://publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/info/rzaij/rzaijnetserverpc.htm

- **Save the files directly to a Windows server using QNTC**

As explained above, the QNTC file system allows you to write directly to a Windows server from your iSeries.

Once QNTC is configured, you can use CoolSpools Database to create your files on a suitable Windows server by specifying a path name starting /QNTC on the TOSTMF parameter of the CoolSpools Database command you are running.

Once your files are saved on your Windows server, they can be accessed by any authorized user who can connect to that server.

- **Save the files directly to a Windows server using FTP**

As an alternative to using the QNTC file system, if your Windows server is running the FTP service, you can use the CoolSpools Database TOSTMF(*FTP) option to send the output to that server via FTP.

Once your files are saved on your Windows server, they can be accessed by any authorized user who can connect to that server.

- **Email**

In the past you may have run queries and produced a large number of iSeries spooled files which were printed then distributed on paper through your internal or external mail.

This process can be transformed into an automated, low-cost electronic service by using CoolSpools Database to save those queries as XML documents, RTF files or Excel spreadsheets.

If you have installed the CoolSpools Email option (product option 2) or if you have some other method of sending email from your iSeries, you can distribute them electronically by email. The stream files could then be deleted once they had been emailed if they were no longer required.

CoolSpools Database Variables

Certain parameters listed below support the use of CoolSpools Database variables.

CoolSpools Database variables consist of a pre-defined variable name from the list below enclosed in<: … :> (start of variable marker = left-hand angle bracket followed by a colon, end of variable marker = colon followed by a right-hand angle bracket).

You can define a different pair of markers from <: and :> by adding/changing the environment variables **CS_VAR_LEFT_MARKER** and **CS_VAR_RIGHT_MARKER.** For example, if you have CS_VAR_LEFT_MARKER set to $% and CS_VAR_RIGHT_MARKER set to %$, you would use $%PAGSETNBR%$ rather than <:PAGSETNBR:> etc.

These variable names, including the markers, are replaced at run time by the corresponding data value. Variable names are not case-sensitive.

Variable name	Description
<:CURJOB:>	Current job name
<:CURUSER:>	Current user id
<:CURJOBNBR:>	Current job number
<:FROMFILE:>	Name of the database file specified on FROMFILE
<:FROMLIB:>	Library specified on FROMFILE
<:FROMMBR:>	Member name specified on FROMFILE
<:RCDFMT:>	Name of the record format in the file specified on FROMFILE. Invalid if more than one record format is selected for processing.
<:FILETEXT:>	File-level text of the file specified on FROMFILE
<:MBRTEXT:>	Member-level text of the member specified on FROMFILE
<:RCDTEXT:>	Record format-level text of the record format in the file specified on FROMFILE. Invalid if more than one record format is selected for processing.
<:QRYDFN:>	Name of the Query/400 query object specified on the QRYDFN parameter
<:QRYDFNLIB:>	Library name specified on the QRYDFN parameter
<:QMQRY:>	Name of the QM Query object specified on the QMQRY parameter
<:QMQRYLIB:>	Library name specified on the QMQRY parameter
<:STMFEXT:>	Default file extension corresponding to the format being output (e.g. '.CSV' when CSV being generated or '.XLS' when an Excel file is being created.
<:TOFMT:>	To-format. The format of the data being generated

	(corresponding to the TOFMT parameter of the CVTDBFSTMF command), .e.g. '*CSV', '*XLS'
<:CURDATE:>	The current date in the format of the current job (DATFMT attribute).
<:CURDATE*xxx:>	The current date in the format indicated by *xxx, where *xxx is any one of: *YMD, *MDY, *DMY, *YYMD, *MDYY, *DMYY, *CYMD, *CMDY, *CDMY, *ISO, *EUR, *JIS, *JUL, *LONGJUL, *JOB or *SYSVAL.
<:CURDAY:>	The current day of the month as a number 01-31.
<:CURMONTH:>	The current month as a number 01-12.
<:CURYEAR:>	The current year as a number 0001-9999
<:CURYEAR4:>	The current year as a number 0001-9999
<:CURYEAR3:>	The current year as a number c01-c99 where c is 0 for the 20th century and 1 for the 21st.
<:CURYEAR2:>	The current year as a number 01-99.
<:CURTIME:>	The current time in hhmmss format.
<:CURUSEREMAIL:>	The email address of current user. The email address is the SMTP email address of the user from the system directory.
<:CURUSERNAME:>	The name of the current user. The name is derived from the information held for the user in the system directory.
<:CURUSERHOME:>	The home directory of the current user. The home directory is taken from the HOMEDIR attribute of the user profile.

Example:

```
CVTDBFXL
      FROMFILE(QADSPOBJ)
      TOSTMF('<:fromlib:>_<:fromfile:>_<:frommbr:>.xls')
      EMAIL(*YES)
      EMAILTO((<:CURUSEREMAIL:>))
```

Here the CVTDBFXL command is being applied to a database file called QADSPOBJ. The name of the stream file to be generated will be derived from the library name, file name and member name, e.g.:

QGPL_QADSPOBJ_QADSPOBJ.XLS

CoolSpools Database Functions

When using CoolSpools Database variables, you can also use a number of CoolSpools Database functions to adjust the data substituted at run time for each variable. These functions can often be helpful in converting the data returned by a variable to a consistent, standard format. For example, you might want to use CoolSpools Database variables to build the names of the Excel files you're creating from data held inside the spooled file. CoolSpools Database functions can help with this, for example by allowing you to:

- remove any leading or trailing spaces

- pad a numeric value to a constant fixed length with leading zeros
- translate certain characters which would be invalid in a file name (such as /) to an alternative acceptable character (such as -)

By default, CoolSpools Database functions consist of a pre-defined function name from the list below preceded by the marker **$$** but you can define a different marker from $$ by adding/changing the environment variable **SL_FCN_MARKER.** For example, if you have SL_FCN_MARKER set to %%, you would use %%TRIM, %%PADL etc. rather than $$TRIM, $$PADL etc.

Function parameters are enclosed in parentheses () and separated by commas. Character values used as parameters are case-sensitive and can be either enclosed in single quotes ' ' (doubled up where required by OS/400), double quotes " ", or not enclosed by anything.

Function names are not case-sensitive.

Function name	**$$FIXNAME**
Description	Creates a valid name usable in a path name. Converts characters that are invalid or unwise in a path name to acceptable characters. This can be useful, for example, if you are creating a directory or file name from some piece of information extracted from the report (e.g. a date or a name) and that data might contain characters that are invalid in a path name (such as a date separator / or an apostrophe in the name). Spaces are converted to underscores. Other invalid characters are converted to hyphens. The CCSID of the job is assumed.
Parameters	
1	String to convert to a valid name
Examples	
$$FIXNAME(<:EXITPGMPOS1:>)	If the <:EXITPGMPOS1:> variable returns the value "John O'Brien" $$FIXNAME(<:EXITPGMPOS1:>) converts this to "John_O-Brien".

Function name	**$$FIXSHEET**
Description	Creates a valid Excel sheet name. Converts characters that are invalid in an Excel sheet name to blanks. This can be useful, for example, if you are creating a sheet name from some piece of information extracted from the report

	(e.g. a name) and that data might contain characters that are invalid in an Excel sheet name (such as / \ [ ] * : and ?). The length of the name is truncated to the maximum length of 31 characters if longer than 31 characters. The CCSID of the job is assumed.
Parameters	
1	String to convert to a valid Excel sheet name
Examples	
$$FIXSHEET(<:CUSTOMER_NAME:>)	If the <:CUSTOMER_NAME:> variable returns the value **SMITH/JONES TRADING [EUROPE] CORPORATION** $$FIXSHEET(<:CUSTOMER_NAME:>) converts this to **SMITH JONES TRADING EUROPE CO**

Function name	**$$TRIM**
Description	Trim characters from the left and right sides of the data. Similar to the ILE RPG %trim() builtin function.
Parameters	
1	Data to trim (typically a CoolSpools Database variable).
2	Characters to remove (optional, default = blank).
Examples	
$$TRIM(<:FILETEXT:>)	Trims blanks from the start of the value returned by CoolSpools Database variable <:FILETEXT:>. For example, the value “ 000123.45- “ becomes “ 000123.45-“
$$TRIM(<:FILETEXT:>,’0’)	Trims zeros from the start of the value returned by CoolSpools Database variable <:FILETEXT:>. For example, the value “000123.4500 “ becomes “123.45 “.

Function name	**$$TRIML**

Description	Trim characters from the left (start) of the data. Similar to the ILE RPG %triml() builtin function.
Parameters	
1	Data to trim (typically a CoolSpools Database variable).
2	Characters to remove (optional, default = blank).
Examples	
$$TRIML(<:FILETEXT:>)	Trims blanks from the start of the value returned by CoolSpools Database variable <:FILETEXT:>. For example, the value “ 000123.45- “ becomes “000123.45- “
$$TRIML(<:FILETEXT:>,’0’)	Trims zeros from the start of the value returned by CoolSpools Database variable <:FILETEXT:>. For example, the value “000123.4500 “ becomes “000123.45 “.

Function name	**$$TRIMR**
Description	Trim characters from the right (end) of the data. Similar to the ILE RPG %trimr() builtin function.
Parameters	
1	Data to trim (typically a CoolSpools Database variable).
2	Characters to remove (optional, default = blank).
Examples	
$$TRIMR(<:FILETEXT:>)	Trims blanks from the end of the value returned by CoolSpools Database variable <:FILETEXT:>. For example, the value “ 000123.45- “ becomes “ 000123.45-“
$$TRIMR(<:FILETEXT:>,’0’)	Trims zeros from the end of the value returned by CoolSpools Database variable <:FILETEXT:>. For example, the value “000123.4500 “ becomes “000123.45 “.

Function name	$$PADL	
Description	Pad a string to a given length by adding a specified character at the start.	
Parameters		
1	Data to pad (typically a CoolSpools Database variable).	
2	Length to pad to	
3	Characters to pad with (optional, default = blank).	
Examples		
$$PADL(<:FILETEXT:>,10)	Pads the value returned by CoolSpools Database variable <:FILETEXT:> to a length of 10 characters by adding blanks at the start. For example, the value “123.45- “ becomes “ 123.45-“	
$$PADL(<:FILETEXT:>,10,’0’)	Pads the value returned by CoolSpools Database variable <:FILETEXT:> to a length of 10 characters by adding zeros at the start. For example, the value “123.45- “ becomes “0000123.45-“	

Function name	$$PADR	
Description	Pad a string to a given length by adding a specified character at the end.	
Parameters		
1	Data to pad (typically a CoolSpools Database variable).	
2	Length to pad to	
3	Characters to pad with (optional, default = blank).	
Examples		
$$PADL(<:FILETEXT:>,10)	Pads the value returned by CoolSpools Database variable <:FILETEXT:> to a length of 10 characters by adding blanks at the end. For example, the value “123.45-“ becomes	

	“123.45- “
$$PADL(<:FILETEXT:>,10,’0’)	Pads the value returned by CoolSpools Database variable <:FILETEXT:> to a length of 10 characters by adding zeros at the end. For example, the value “123.45“ becomes “123.450000 “

Function name	**$$SUBST**
Description	Returns a substring. Similar to ILE RPG’s %subst.
Parameters	
1	Data to substring (typically a CoolSpools Database variable).
2	Start position
3	Length (optional, default = to end of string).
Examples	
$$SUBST(<:FILETEXT:>,5)	Returns the substring of the value returned by CoolSpools Database variable <:FILETEXT:> starting at character position 5 and extending to the end of the string. For example, the value “0000123456“ becomes “123456 “.
$$PADL(<:FILETEXT:>,5,3)	Returns the substring of the value returned by CoolSpools Database variable <:FILETEXT:> starting at character position 5 and extending for 3 characters. For example, the value “0000123456“ becomes “123 “.

Function name	**$$UPPER**
Description	Converts a string to upper case, assuming the CCSID of the job.
Parameters	
1	Data to convert (typically a CoolSpools Database variable).
Examples	

$$UPPER(<:FILETEXT:>)	Converts the value returned by CoolSpools Database variable <:FILETEXT:> to upper case. For example, the value “John Smith“ becomes “JOHN SMITH“.

Function name	**$$XLATE**
Description	Translates characters in the data. Similar to the ILE RPG %xlate function.
Parameters	
1	List of characters to translate from
2	List of characters to translate to
3	Data to translate (typically a CoolSpools Database variable).
4	Start position (optional, default = first)
Examples	
$$XLATE(“ “,”_”,<:FILETEXT:>,1)	Translates spaces in the value returned by CoolSpools Database variable <:FILETEXT:> to underscores, starting at the first character. For example, the value “John Smith“ becomes “John_Smith“.

Function name	**$$XLDATE**
Description	Converts a date to an Excel date (day count since 1st Jan 1900). This function can be useful when specifying conditional formatting rules that require a date constant to be specified, as Excel requires these to be defined in Excel date format.
Parameters	
1	The date to convert, specified as date string e.g. 07/04/2011
2	The format in which the first parameter is specified. If this parameter is omitted, the date format and separator implied by job attributes DATFMT and DATSEP are assumed.

	Other valid formats are similar to those supported by ILE RPG, e.g.: *ISO, *USA, *EUR, *JIS, *YMD, *MDY, *DMY. Optionally, a separator character may be appended to the format code, e.g. *MDY/, *DMY, *YMD0 etc.
Examples	
$$XLDATE(07/04/2011)	This would return the date (either 7th April or July 4th, depending on whether the DATFMT attribute is *DMY or *MDY) as a day count since 1900. This expression could be used as part of a conditional formatting rule, for example to highlight in red any dates equal to or after 1st January 2011: CNDFMTRULE((1 1 *FLDNAM date_fld *GE '$$XLDATE(01/01/2011)' *NONE red)
$$XLDATE(07/04/2011,*MDY/)	Same as the above, but with the date format explicitly stated to be MDY with a separator of /

Excel Placeholders

When specifying header and footer text to be printed within an Excel file (XLSPRINT parameter), you can use Excel placeholders. Excel will substitute a data value for the placeholder before printing the file.

Note that while CoolSpools Database variables are defined by CoolSpools Database and a data value is substituted for them by CoolSpools Database as it creates the file, Excel placeholders are substituted by Excel as it prints the file. So, if you were to include the current date in an Excel footer text using a CoolSpools Database variable such as <:CURDATE:>, that value would be replaced by the current date at the time the file was created and would not change when the file is printed, whereas if you use an Excel placeholder such as &D to insert the current date, that date will be substituted by Excel with the current date every time the file is printed.

Excel placeholders take the form of a single letter preceded by an ampersand (&), with the sole exception of the codes indicating a change of font name and/or size (see below).

&P	Current page number
&N	Page count
&D	Current date

&T	Current time
&A	Sheet name
&F	File name without path
&Z	File path without file name (BIFF8)
&U	Underlining on/off
&E	Double underlining on/off
&S	Strikeout on/off
&X	Superscript on/off
&Y	Subscript on/off
&"<fontname>"	Set new font <fontname>
&"<fontname>,<fontstyle>"	Set new font with specified style <font style>. The style <fontstyle> is in most cases one of "Regular", "Bold", "Italic", or "Bold Italic". But this setting is dependent on the font, it may differ (localised style names, or "Standard", "Oblique", ...)
&<fontheight>	Set font height in points (<fontheight> is a decimal value). If this command is followed by a plain number to be printed in the header, it must be separated from the font height with a space.

Example: **&"Arial,Bold Italic"&14Page &P of &N**

On Page 2 of a 3-page spreadsheet, this header/footer string prints the text "Page 2 of 3" in Arial Bold Italic 14-point font.

Using CoolSpools Database

Getting Started

The product library for all options of the CoolSpools V7R1M0 product is **COOLSPV7R1.**

You run the main CoolSpools Database functions by executing one of the format-specific commands on your system i.

Either:

(a) prompt one of the following command strings with F4:

To create an excel file:	**CVTDBFXL**
To create an XML file:	**CVTDBFXML**
To create a delimited ASCII file (e.g. CSV):	**CVTDBFCSV**
To create an HTML file:	**CVTDBFHTML**
To create a fixed-width text file:	**CVTDBFTXT**
To extract cells contents from Excel files:	**CVTXLDBF**

Or

(b) display the CoolSpools Database menu by entering:

GO COOLSPV7R1/ DATABASE

Using styles

You can define styles that will be applied to your output when you are converting to Excel, HTML or XML formats. These styles specify the appearance of data on screen when the spreadsheet is opened (in MS Excel or another spreadsheet application) or when the HTML or XML document is opened (in your browser).

The command CVTDBFXLSX does not include the DFNSTYLES parameter.

There are two ways to define styles:

- Permanently, by means of the WRKSTLDFN (Work with Style Definitions) and CRTSTLDFN (Create Style Definition) commands. Styles defined in this way are stored for future reference by name on APYSTYLES parameter of CVTDBFXL, CVTDBFHTML and CVTDBFXML, as well as the CNDFMTRULE parameter of CVTDBFXL.
- Temporarily, using the DFNSTYLES parameter of the CVTDBFXL, CVTDBFXML and CVTDBFHTML commands. Styles defined in this way exist only for the duration of this running of the command and become undefined after the command completes. If the name of a style defined on the DFNSTYLES parameter is the same as that of an existing style definition created using CRTSTLDFN, the attributes defined on the DFNSTYLES parameter override those of the permanent style definition for the duration of the current running of the command.

There are five predefined styles which define the default appearance of different types of information in Excel and HTML:

Style Name	Description
*DATA	Data rows
*HEADER	Column heading rows generated based on the setting of the HEADER parameter
*TITLE	Additional heading rows (additional headings defined on HEADER plus HTML caption)
*SUBTOTAL	Subtotals in Query/400 output using the *COMBINED output type
*TOTAL	Final totals in Query/400 output using the *COMBINED output type

plus two more that are relevant only to XML output:

Style Name	Description
*ROOT	The root element of the document
*ROW	The row element of the document. The row element is that corresponding to records in the input file.

If you do not specify these styles on the DFNSTYLES parameter, they will assume certain default values (see the table in the section on the DFNSTYLES parameter below).

You can modify the defaults for these predefined styles by creating a style definition with the appropriate name (i.e. *DATA, *HEADER etc.) using CRTSTLDFN.

On the other hand, if you do define one or more of these styles on the DFNSTYLES parameter, the values you enter for that parameter override the defaults for the type of information associated with that style.

In addition, you can define your owned named styles on the DFNSTYLES parameter.

There are two ways in which to associate a style with a piece of data in your output file (e.g. a cell in an Excel spreadsheet or an element of an XML document):

1. **Implicitly**

   By defining the style name the same as the name of a row group in your Database-to-Excel map or an element in your Database-to-XML map, you implicitly apply that style to the data in question. For example, a style called REPORT_HEADING will be implicitly and automatically applied to an Excel row group called REPORT_HEADING.

   Note that style names are case-sensitive because XML element names need to be case-sensitive and for this association of names to work, the names must match exactly in terms of case.

2. **Explicitly**

   Alternatively, use the APYSTYLES parameter to define the styles you wish to apply to different parts of the file you create.

   Example:
   CVTDBFXL
   ...

```
DFNSTYLES((HIGHLIGHT *YES *NO *GENERAL *NONE *BOTTOM *NO *NO
*AUTOFIT *ARIAL 12 *YES *NO *NO *YELLOW *BLUE *AUTO *NONE *THIN))
APYSTYLES((*FLDNAM BALANCE HIGHLIGHT))
```

This code defines a new style called “highlight” that uses Arial bold 12-point yellow on blue and applies that style to the field called “BALANCE”.

Using conditional formatting

Styles are also used when you want to apply conditional formatting rules to Excel spreadsheets that CoolSpools Database generates. Conditional formatting lets you modify the appearance of cells in the spreadsheet depending on whether certain rules you define are met or not. For example, if your spreadsheet contains data from customer accounts, you might color those rows that relate to accounts with a negative balance red to highlight them, while those with a credit balance over $1,000 might be colored green.

Use the CNDFMTGRP (Conditional Formatting Groups) parameter to define the range of columns to which a group of related rules should be applied.

Use the CNDFMTRULE (Conditional Formatting Rules) parameter to define the rules to be applied and the style (as defined on DFNSTYLES) that will be used to format cells where those rules evaluate to true.

Using encrypted passwords

In the past, if you specified a password on a command such as CVTDBFSTMF and embedded that command in your CL source code, you would need to store that password in plain text form. This was clearly a security exposure.

Now, CoolSpools Database gives you the opportunity to use encrypted passwords on all command parameters that accept a password string. An encrypted password is a scrambled version of your password which is returned to you when you supply the actual password to the DSPENCPWD (Display Encrypted Password) command. You can then code the scrambled password in your source code and specify *YES for the associated **“Encrypted password supplied”** element to indicate to CoolSpools Database that it needs to decrypt the password before use.

For example, if you supply the password “test” to DSPENCPWD, thus:

```
DSPENCPWD PWD('test')
```

it will send you the completion message:

```
Encrypted password is X'178D2D35E0EBFF508A63252433D6C4E0'.
```

You can then use this encrypted password on commands that require a password, e.g.:

```
ZIPDTA ... PWD(X'178D2D35E0EBFF508A63252433D6C4E0' *YES)
```

The password of the zipped file(s) will be “test”.

CVTDBFxxxx Command Parameters

The sections below look at each of the parameters to the various ***CVTDBFxxxx***commands (CVTDBFXLSX, CVTDBFXL, CVTDBFXML, CVTDBFHTML, CVTDBFPDF, CVTDBFCSV, CVTDBFTXT, plus the now deprecated CVTDBFSTMF) in turn and explain how they should be used to get the most out of this invaluable utility.

In the examples, an ellipsis (…) indicates that a number of required parameters have been omitted for the sake of clarity.

FROMFILE – From database file

Parameter	**FROMFILE**
Description	**Specifies the database file to be converted or indicates the source of the data to be converted (query, SQL)**
Applies to commands:	**CVTDBFXLSX, CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **FROMFILE** (From File) parameter specifies the name and library of the database file that contains the records to be converted. Alternatively, one of the special values explained below can be used to run a query and save the output to a stream file as a single operation.

A database file can be a physical file or a logical file.

There are three elements to this parameter.

From File

The first element of the FROMFILE parameter is the qualified name of the file which contains the data to be converted or one of the special values listed below.

Unless one of the special values listed below is input, the file name must be specified as:

file-name Specify the name of the database file that contains the records to be converted.

The possible library values are:

***LIBL** All libraries in the user and system portions of the job's library list are searched until the first match is found.

***CURLIB** The current library for the job is used to locate the database file. If no library is specified as the current library for the job, the QGPL library is used.

library-name Specify the name of the library to be searched.

Special values for the FROMFILE parameter are as follows:

***SQL** Run a SQL statement. The **SQL** parameter will be used to define the SQL statement to be run and the naming convention to be applied. The SQL statement will be validated, and, if valid, it will be executed and the output saved in a temporary result table, which will then be converted to the stream file format defined on the **TOFMT** parameter.

SQLSRC** Run a SQL statement held in a source file. The **SQLSRC** parameter will be used to specify the name of the source file and member where the SQL statement is held. The SQL statement will not be validated until it is executed. ***CoolSpools Database will create a temporary Query Management (QM) query object from the contents of the source file. This temporary QM query will be executed and the output saved in a temporary result table, which will then be converted to the stream file format defined on the **TOFMT** parameter.

QMQRY** Run a Query Management query. The **QMQRY** parameter will be used to specify the name of the QM query object to be run. ***CoolSpools Database will execute the QM query and save the output in a temporary result table, which will then be converted to the stream file format defined on the **TOFMT** parameter.

QRYDFN** Run a Query/400 query. The **QRYDFN** parameter will be used to specify the name of the Query/400 query object to be run, and optionally, the file to be queried. ***CoolSpools Database will execute the Query/400 query and save the output in a temporary result table, which will then be converted to the stream file format defined on the **TOFMT** parameter.

***MAP** (Excel and XML only). Indicates that a database map will be used to control the structure of the output file and that the input file or query object was specified when the map was created. The database map name must be specified on the MAPNAME parameter.

Select records and fields

The second element of the FROMFILE parameter allows you to control the prompting of record format names and field names. This prompting cannot be performed if one of the special values listed above has been specified for the file name parameter.

***NO** (Default). Record format and field names will not be prompted.

***YES** The record format will be selected and all of the fields from the database file, will be included in the selection.

	The RCDFMT and INCLFLD parameters will be populated, with the record format and all the fields from the database file**.**
***PROMPT**	You will be prompted with a list of record formats and field names from the file specified on the FROMFILE parameter, allowing you to indicate which fields and record formats should be included in the stream file, and in what order. The values selected here, will populate the RCDFMT, INCLFLD, RPTBKRS, RPTSMRY and SORT parameters. Please note, if you select a sort sequence, then go to the next screen, then F12 to go back, you will lose the sort sequences entered.

To prompt the command, type the command name (CVTDBFXL etc.) on a command line and press F4.

From member

The third element of the FROMFILE parameter allows you to specify the name of the member in the file from which input will be taken.

This element must be ***FIRST** if one of the special values is specified for the file name.

***FIRST**	(Default). The first member is used.
Member_name	Specify the member to be used.

Example: CVTDBFXL FROMFILE(CUSTLIB/CUSTFILE *YES)…

Here the CVTDBFXL command is being applied to a database file called **CUSTFILE** which is located in library **CUSTLIB**. If you prompt the command with F4, you will be shown a list of fields and record formats from the file on the INCLFLD (Include Fields) and RCDFMT (Record Formats) parameters respectively.

Example: CVTDBFXL FROMFILE(*SQL) …

Here the CVTDBFSTMF command is being used to run a piece of SQL. The SQL will be syntax checked, and, if valid, executed. The results will be saved in a temporary result table then converted to a stream file in the format specified on the **TOFMT** parameter.

MAPNAME – Database map name

Parameter	**MAPNAME**
Description	**Specifies the names of the database map to be used when FROMFILE(*MAP) was specified**
Applies to commands:	**CVTDBFXLSX** **CVTDBFXL** **CVTDBFXML**
Dependent on:	**FROMFILE(*MAP)**
Supports CoolSpools Database variables	**No**

If FROMFILE(*MAP) is specified on the CVTDBFXL or CVTDBFXML commands, indicating that the structure of the output from and the source of the input to those commands is to be determined by a database map, the name of the database map to use must be specified on this parameter.

Options are:

***NONE** No map is to be used. Invalid if FROMFILE(*MAP) was specified.

dataasbe_map_name Specify the name of the database-to-Excel or database-to-XML map to be used. Ignored unless FROMFILE(*MAP) was also specified.

TOSTMF – To stream file

Parameter	**TOSTMF**
Description	**Specifies the path name of the stream file to create or update or the special value *FTP indicating FTP output**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**Yes**

The **TOSTMF** (To Stream File) parameter specifies the path name of the stream file into which converted data is placed. All directories in the path name must exist.

New directories are not created. If the stream file does not exist, it is created.

The name you enter here may be up to 1,024 characters long. However, the name that you choose must be a valid name for the IFS file system into which the stream file is to be created. For example, the shared folders (QDLS) file system only supports file names in the 8.3 format, i.e. a file name up to 8 characters long followed by an optional extension of up to 3 characters. If you choose an invalid file name, an error will occur and the file will not be saved.

You should choose a file name which is suitable for the type of file being created. For example, Excel files should be given the extension **.xls** so that they are recognized as Excel files by applications such as Excel and Lotus 123. HTML files are normally given as extension of **.htm** or **.html**. ASCII fixed text files often have an extension such as **.txt**, **.asc**, **.prn** or **.dat**.

Options are:

***FROMFILE** (Default). A stream file name is generated based on the name specified on the FROMFILE parameter. The stream file will be created in the current directory. The stream file name will be the name of the file specified on the FROMFILE parameter followed by a period (.) and an appropriate extension indicating the type of file being created.

***FTP** The output will be sent by FTP to a remote server running an FTP service (e.g. another system i, a UNIX machine or a Windows server). The output will not be stored permanently on the local system i. You will be required to

	define the details of the FTP server to receive the output on the FTP parameter.
to-stmf	Specify the full relative or absolute path name of the stream file to be created or updated.

For more information on specifying path names, refer to the IFS Introduction manual in the system i Information Center at http://publib.boulder.ibm.com/system i/V5R2/ic2924/info/ifs/rzaaxmstpath.htm#HDRPATH

Example: CVTDBFXL FROMFILE(CUSTFILE)
TOSTMF(*FROMFILE)

Here the CVTDBFXL command is being applied to a database file called **CUSTFILE**. The data is to be converted to Excel format and, since **TOSTMF(*FROMFILE)** is specified, the stream file will be created in the current directory and will be called **CUSTFILE.XLS**.

Example: CVTDBFXL FROMFILE(CUSTFILE)
TOSTMF('/customer_data/customers.xls')

Here again the CVTDBFXL command is being applied to a database file called **CUSTFILE.** This time the stream file to be created is called **customers.xls** and will be created in directory **customer_data** in the root file system.

TOFMT – To format

Parameter	**TOFMT**
Description	**Specifies the format of the output**
Applies to commands:	**CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **TOFMT** (To Format) parameter allows you to select the format to which the database file should be converted.

Starting from this release (V7R1M0), CoolSpools Database supplies a series of format-specific commands such as CVTDBFXL (Convert Database File to Excel) and CVTDBFHTML (Convert Database File to HTML) where the format of the output is implicit in the command itself. This parameter therefore relates only to the CVTDBFSTMF command which can output several different formats. Use of the CVTDBFSTMF command is now deprecated. CVTDBFSTMF will not be enhanced in future releases and may be withdrawn in a future release.

The options are:

***XLS** (Default). Excel © (BIFF) format. The contents of the database file are converted to columns and rows in an Excel spreadsheet. Column widths can be managed and formatting is implemented to reflect edit codes and edit words associated with the fields in the database file.

***CSV** Delimited ASCII text (e.g. Comma Separated Variable). The database file contents are converted to delimited ASCII text records. The delimiters used to separate fields and enclose strings are defined on the **CSV** parameter. This option enables you to create files in CSV (comma-separated variable) format and other similar formats (e.g. tab separated or blank separated) for loading into spreadsheets and other PC applications.

***HTML** HTML. The contents and attributes of the database file are converted to HTML format, suitable for viewing in a browser such as Internet Explorer or Netscape Navigator.

***FIXED** Fixed-format ASCII text. The contents of the database file are converted to ASCII text records. Each field is

assigned a fixed width so that data for each field always starts and ends in the same position. Each column is separated from the next by a single blank.

Example: CVTDBFSTMF FROMFILE(ORDERS)…
TOFMT(*XLS)

Here the CVTDBFSTMF command is being applied to a database file called **ORDERS** in order to create a stream file in Excel format.

STMFOPT – Stream file option

Parameter	**STMFOPT**
Description	**Indicates what action should be taken if the output file already exists.**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **STMFOPT** (Stream File Option) parameter allows you to select the action the command should take if the stream file you have specified on the **TOSTMF** parameter already exists. This parameter is ignored if the stream file does not already exist.

On the format-specific commands (e.g. CVTDBFXLSX) there are three elements to this parameter. On CVTDBFSTMF there are only 2 elements.

Option

The first element specifies the option to be taken if the file exists.

The options are:

***NONE** The command reports an error and the existing file is not changed. For safety's sake, this is the default value.

***REPLACE** The existing file is replaced.

***ADD** The contents of the database file are appended to the end of the existing file. In relation to Excel output, this results in a new worksheet being added.

***UNIQUE** CoolSpools Database generates a unique file name for the output file by appending a numeric suffix to the name specified on the TOSTMF parameter (before any extension). The numeric suffix will be one higher than the highest suffix associated with any existing file of this name in the directory. If a value other than *NONE is specified on the ***UNIQUE separator character** element below, that character is inserted between the name and the suffix.

***ADDROWS** Will append the converted data to an existing worksheet. The worksheet to be updated should be specified in the Worksheet name element of the EXCEL parameter. Headers and/or additional text can also be included with the appended rows by means of the HEADER parameter.

***RPLXLSSHT** When outputting to Excel format, this option allows you to specify that the new worksheet(s) created this time will replace one or more existing worksheets in the file. The worksheets to be replaced are specified on the RPLXLSSHT parameter.

Create Empty File

The second element specifies the option to be taken if there is no data to be output.

The options are:

***NO** CoolSpools Database produces no output if there is no data to convert.

***YES** CoolSpools Database produces an empty file if there is no data to convert (e.g. an Excel file with zero data rows).

*UNIQUE suffix separator

On the CVTDBFSTMF command, this element does not exist on STMFOPT and the information is supplied instead on the separate **SEPCHAR** (Separator Character) parameter. It enables you to specify the separator character that CoolSpools Database will use when generating a unique stream file name in conjunction with STMFOPT(*UNIQUE).

When STMFOPT(*UNIQUE) is specified, this character is inserted between the file name you specify on the **TOSTMF** parameter (minus the extension) and the numeric suffix which CoolSpools Database appends to that name (minus the extension) to create a unique file name.

On the format-specific commands CVTDBFXL etc., this parameter is now part of the STMFOPT parameter.

Options are:

***NONE** (The default). No separator is used.
***UNDERSCORE** An underscore character (_).
Any other character Any other character allowed in a file name.

Example: CVTDBFXL FROMFILE(INVOICES)
TOSTMF(invoices.xls)
STMFOPT(*ADD)

Here the CVTDBFXL command is being applied to a database file called **INVOICES** and converted to a stream file called **invoices.xls**. A new worksheet will be added to the existing file.

Example: CVTDBFXLSX FROMFILE(INVOICES)
TOSTMF(invoices.xls)
STMFOPT(*ADDROWS)
HEADER(*AVAIL *HEADER *NO 'Created Date<:CURDATE:>')
EXCEL(*XLSX *MM *HMS NewInvoices)

Here the CVTDBFXLSX command is being applied to a database file called **INVOICES** and converted to a stream file called **invoices.xlsx**. The rows will be appended to the NewInvoices worksheet. The text 'Created Date' followed by the date will be included before the appended data.

RPLXLSSHT – Replace Excel worksheet names

Parameter	**RPLXLSSHT**
Description	**Specifies the names of Excel worksheets to be replaced when STMFOPT(*RPLXLSSHT) is specified**
Applies to commands:	**CVTDBFXLSX** **CVTDBFXL** **CVTDBFSTMF (deprecated)**
Dependent on:	**STMFOPT(*RPLXLSSHT)**
Supports CoolSpools Database variables	**No**

When outputting to Excel format, specifying STMFOPT(*RPLXLSSHT) allows you to indicate that the new worksheet(s) created this time will replace one or more existing worksheets in the file. The worksheet(s) to be replaced is/are specified on this RPLXLSSHT parameter.

There is a single value:

***NO** (Default) No worksheet(s) will be replaced.

Alternatively, specify between 1 and 10 names of existing worksheets in the file specified on the TOSTMF parameter that will be replaced by the new worksheet(s) created this time the command is run. The new worksheet(s) will be inserted into the file after the worksheet preceding the first worksheet listed on this parameter or at the beginning of the file if the first worksheet in the file is listed here. All worksheets listed here will be dropped from the file.

Options are:

***FIRST**	The first worksheet
***LAST**	The last worksheet
worksheet_name	Specify the name of the worksheet.

FORMAT – Format specification

Parameter	**FORMAT**
Description	**Indicates the format of the data to convert where the DDS of the file is not the best guide**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **FORMAT** (Format specification) parameter lets you specify the name of a second file which will determine the structure of the data in the file being converted. This option is similar to the FORMAT parameter of the OPNQRYF command and can be useful where the data is held in a flat file with no DDS. Define the structure of the data in DDS, create a format file and specify that file on the FORMAT parameter.

Options are:

Element 1:

***FROMFILE** (The default). The DDS (metadata) of the file specified on the FROMFILE parameter determines the structure of the data being converted.

Qualified_file_name Specify the name of the file that will define the structure of the data being converted.

Element 2:

***FILE or *FMTSET** Specifies the type of object being referenced - a file or a record format set.

FROMRCD – From record number

Parameter	**FROMRCD**
Description	**The first relative record number to convert**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **FROMRCD** (From record) parameter identifies the relative record number of the first record to be converted.

Options are:

***START**	(The default). Conversion begins with the first record in the file.
Record_ number	The relative record number of the first record to be converted.

TORCD – To record number

Parameter	**TORCD**
Description	**The last relative record number to convert**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **TORCD** (To record) parameter identifies the relative record number of the last record to be converted.

Options are:

***END**	(The default). Conversion ends with the last record in the file.
Record_ number	The relative record number of the last record to be converted.

SEPCHAR – Separator character

Parameter	**SEPCHAR**
Description	**Separator character to use when building file names for STMFOPT(*UNIQUE)**
Applies to commands:	**CVTDBFSTMF (deprecated)**
Dependent on:	**STMFOPT(*UNIQUE)**
Supports CoolSpools Database variables	**No**
Migration notes	**This parameter has been deleted and incorporated into the STMFOPT parameter on the new format-specific commands.**

The **SEPCHAR** (Separator Character) parameter enables you to specify that separator character that CoolSpools Database will use when generating a unique stream file name in conjunction with STMFOPT(*UNIQUE).

When STMFOPT(*UNIQUE) is specified, this character is inserted between the file name you specify on the **TOSTMF** parameter (minus the extension) and the numeric suffix which CoolSpools Database appends to that name (minus the extension) to create a unique file name.

On the format-specific commands CVTDBFXL etc., this parameter is now part of the STMFOPT parameter.

Options are:

***NONE** (The default). No separator is used.
***UNDERSCORE** An underscore character (_).
Any other character Any other character allowed in a file name.

SQL – SQL statement options

Parameter	**SQL**
Description	**The SQL statement to run and SQL statement options**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**FROMFILE(*SQL)**
Supports CoolSpools Database variables	**Yes**

The **SQL** (SQL Statement Options) parameter only appears if **FROMFILE(*SQL)** is specified, indicating that the input is to be the result set of an SQL statement. This parameter allows you to input the SQL statement to be run and define the naming convention to be used.

There are 2 elements to the parameter.

SQL Statement

Input the SQL statement you wish to execute. This must be a SELECT statement, otherwise an error will occur. The statement will be syntax checked prior to being executed.

Naming

Specifies the naming convention used for objects in SQL statements.

***SYS**	(Default). The system naming convention (library-name/file-name) is used.
***SQL**	The SQL naming convention (collection-name.table-name) is used.

Example: CVTDBFXL FROMFILE(*SQL) …
SQL('select * from custfile where state = ''NY''')

This example shows an SQL statement being run to provide the input to the CVTDBFXL command. The output from the SQL query will be saved as an Excel file.

See the QRYSLT parameter for details of how that parameter relates to record selection performed with the SQL.

See the SORT parameter for details of how that parameter relates to record ordering performed with the SQL.

SQLSRC – SQL source options

Parameter	**SQLSRC**
Description	**The source member containing SQL to run and related options**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**FROMFILE(*SQLSRC)**
Supports CoolSpools Database variables	**No**

The **SQLSRC** (SQL Source Options) parameter only appears if **FROMFILE(*SQLSRC)** is specified, indicating that the input is to be the result set created by running an SQL statement held in a source file. This parameter allows you to indicate the source file and source member where the SQL statement is stored.

Please note that:

- the file specified must exist
- the member specified must exist in the file
- the file must be a source physical file
- the source member must not be empty

There are 2 elements to the parameter.

SQL Source file

The first element of the **SQLSRC** parameter is the qualified name of the file which contains the SQL statement to be executed.

file-name Specify the name of the source physical file that contains the SQL statement to be executed.

The possible library values are:

***LIBL** All libraries in the user and system portions of the job's library list are searched until the first match is found.

***CURLIB** The current library for the job is used to locate the database file. If no library is specified as the current library for the job, the QGPL library is used.

library-name Specify the name of the library to be searched.

SQL Source member

The second element of the **SQLSRC** parameter is the name of the source member which contains the SQL statement to be executed.

member-name	Specify the name of the source member that contains the SQL statement to be executed.

Example: CVTDBFXL FROMFILE(*SQLSRC) …
SQLSRC(SQLSRC CUSTSQL)

This example shows an SQL statement stored in a source file being run to provide the input to the CVTDBFXL command. The output from the SQL query will be saved as an Excel file. The SQL statement is in source member CUSTSQL in source file SQLSRC.

See the QRYSLT parameter for details of how that parameter relates to record selection performed with the SQL.

See the SORT parameter for details of how that parameter relates to record ordering performed with the SQL.

QRYDFN – Query/400 options

Parameter	**QRYDFN**
Description	**Query/400 options when FROMFILE(*QRYDFN) used**
Applies to commands:	**CVTDBFXLSX ,CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**FROMFILE(*QRYDFN)**
Supports CoolSpools Database variables	**No**
Migration notes	**The Query File and Query File Member elements of this parameter have been deleted from the new format-specific commands. If you have code which references this parameter, it will need to be modified when moving away from CVTDBFSTMF to the new format-specific commands if that code references these deleted elements or any later elements.** **The Query File and Query File Member elements are now replaced by the QRYFILE parameter (which previously provided an alternative means of specifying a query file and was the only way of specifying multiple query files).**

The **QRYDFN** (Query/400 Options) parameter only appears if **FROMFILE(*QRYDFN)** is specified, indicating that the input is to be the output created by running a Query/400 query. This parameter allows you to indicate the name of the Query/400 query object to be run.

There are five elements to this parameter (seven in the case of CVTDBFSTMF):

Query/400 object

Specifies the qualified name of the Query/400 query object.

query-name Specify the name of the Query/400 query object (object type *QRYDFN) which should be run.

The possible library values are:

***LIBL** All libraries in the user and system portions of the job's library list are searched until the first match is found.

***CURLIB** The current library for the job is used to locate the database file. If no library is specified as the current library for the job, the QGPL library is used.

library-name Specify the name of the library to be searched.

Record selection

Allows you to control whether you will be prompted with the Query/400 Record Selection screen. This is only possible if the command is being run interactively and if Query/400 is licenced.

See the QRYSLT parameter for details of how that parameter relates to record selection performed with the SQL.

See the SORT parameter for details of how that parameter relates to record ordering performed with the SQL.

Options are:

***NO**	(Default). The Query/400 record selection screen is not displayed.
***YES**	The Query/400 record selection screen is displayed, if the command is run interactively.

Query file

This element has been deleted from the format-specific commands: use the QRYFILE parameter instead.

Specifies the qualified name of the file the Query/400 query should process as its input file.

***QRYDFN**	(The default). The input file specified in the Query/400 object is used.
file_name	The name of the file to be queried. This overrides the file name specified in the Query/400 object.

The possible library values are:

***LIBL**	All libraries in the user and system portions of the job's library list are searched until the first match is found.
***CURLIB**	The current library for the job is used to locate the database file. If no library is specified as the current library for the job, the QGPL library is used.
library-name	Specify the name of the library to be searched.

Query file member

This element has been deleted from the format-specific commands: use the QRYFILE parameter instead.

Specifies member in the file which the Query/400 query should process as its input file.

***FIRST**	(The default). The first member is used.
member_name	The name of the member to be used.

Output form

Specifies the form of output produced by the query. If no value was specified in the query and no value was entered on the command, or if a query name is not specified, *DETAIL is assumed.

The possible values are:

***RUNOPT**	The output form (detail or summary) specified in the query definition is used. If the query contains both detail records and summary records, both are output, as if *COMBINED has been specified.
***DETAIL**	The output form produced by the query is a report containing detail records only.
***SUMMARY**	The output form produced by the query is a report containing summary records only.
***COMBINED**	Where a query contains both detail fields and summary functions, CoolSpools Database will create a file containing both detail lines and summaries.
***MIXED**	Where a query contains both detail fields and summary functions, CoolSpools Database will create a file containing both detail lines and summaries. This differs from *COMBINED in the way in which summary rows are presented. Summary rows are output with the summary-level data values rather than corresponding text labels.

Summary line style

This element controls how CoolSpools database formats summary lines in the output.

The possible values are:

***PRINTER**	The style of the output is similar to that used when Query/400 output is directed to a spooled file (RUNQRY ... OUTPUT(*PRINTER) option).
***QRYDFNTXT**	The value of the break level field (a number representing the break level) is replaced by the corresponding level break text defined for the appropriate summary level in the Query/400 query.
***LEVELBRK**	The break level field is replaced by a piece of text retrieved from message id SLP5001 in message file CP_MSGF, combined with the value of the break level field. For level 0 (grand totals) the text is derived from message SLP5002. You may modify this text if you wish but please note that these modifications will need to be repeated every time a CoolSpools Database PTF or a new release is installed.
***SUBTOTAL**	The break level field is replaced by a piece of text retrieved from message id SLP5004 in message file CP_MSGF. This is useful if you just want the same word to appear each time, e.g. "Sub-total'. For level 0 (grand totals) the text is derived from message SLP5002. You may modify this text if you wish but please note that these modifications will need to be repeated every time a

CoolSpools Database PTF or a new release is installed. Note that the title of the column representing the break level is derived from message id SLP5003 in message file CP_MSGF. You may modify this text if you wish but please note that these modifications will need to be repeated every time a CoolSpools Database PTF or a new release is installed.

***INCLUDE** The value of the break level field is included as a column in the output. The column contains a number denoting the summary level to which the other columns relate.

***EXCLUDE** The break level field is dropped from the output completely.

Include overflow indicator?

When Query/400 produces summary output and that output is directed to an output file, the file created includes two additional columns: the break level, specifying the summary level to which the columns relate, and an overflow indicator, which flags field overflow if it occurs.

This element determines how CoolSpools Database handles the second of those two additional fields (overflow indicator).

The possible values are:

***EXCLUDE** The overflow indicator field is dropped from the output completely.

***INCLUDE** The value of the overflow indicator is included as a column in the output.

Example: CVTDBFXL FROMFILE(*QRYDFN) …
QRYDFN(CUSTQRY *YES)

This example shows a Query/400 query call CUSTQRY being run to provide the input to the CVTDBFXL command. If this command is run interactively, before the query is executed, you will be prompted with the Query/400 Record Selection screen to enable you to enter record selection criteria. The output from the query will be saved as an Excel file.

See the QRYSLT parameter for details of how that parameter relates to record selection performed with the query.

See the SORT parameter for details of how that parameter relates to record ordering performed with the query.

QRYFILE – Query file

Parameter	**QRYFILE**
Description	**The input file(s) to use with a Query/400 query**
Applies to commands:	**CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**FROMFILE(*QRYDFN)**
Supports CoolSpools Database variables	**No**
Migration notes	**The Query File and Query File Member elements of the QRYDFN parameter have been deleted from the new format-specific commands. If you have code which references this parameter, it will need to be modified when moving away from CVTDBFSTMF to the new format-specific commands if that code references these deleted elements or any later elements.** **The Query File and Query File Member elements are now replaced by the QRYFILE parameter (which previously provided an alternative means of specifying a query file and was the only way of specifying multiple query files).**

The **QRYFILE** (Query file) parameter only appears if **FROMFILE(*QRYDFN)** is specified, indicating that the input is to be the output created by running a Query/400 query. This parameter allows you to override the default names of the files used by your Query/400 query.

The default is the single value ***QRYDFN** which indicates that the default file names, library names and member names held in the Query/400 query object specified on the QRYDFN parameter will be used.

Alternatively you can specify the file, library and member name(s) of up to 32 files to be used instead of the files defined in the Query/400 object.

QMQRY – QM Query options

Parameter	**QMQRY**
Description	**QM query options when FROMFILE(*QMQRY) used**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**FROMFILE(*QMQRY)**
Supports CoolSpools Database variables	**No**

The **QMQRY** (Query Management Query Options) parameter only appears if **FROMFILE(*QMQRY)** is specified, indicating that the input is to be the output created by running a Query Management (QM) query. This parameter allows you to indicate the name of the QM Query object to be run.

See the QRYSLT parameter for details of how that parameter relates to record selection performed with the query.

See the SORT parameter for details of how that parameter relates to record ordering performed with the query.

There are four elements to this parameter.

QM query object

The first element is the qualified name of the QM Query object.

query-name Specify the name of the QM Query object (object type *QMQRY) which should be run.

The possible library values are:

***LIBL** All libraries in the user and system portions of the job's library list are searched until the first match is found.

***CURLIB** The current library for the job is used to locate the database file. If no library is specified as the current library for the job, the QGPL library is used.

library-name Specify the name of the library to be searched.

Query Management report form

The next element is the qualified name of the QM Report Form object. ***CoolSpools Database*** will retrieve column headings from the QM Report Form object you specify here. If you do not specify a QM Report Form object, headings will be taken from the output file created by running the QM Query.

***NONE**	(Default). No QM Form is specified and column headings are taken from the output file created when the QM Query is run.
***QMQRY**	The value specified on the Query management query prompt (QMQRY parameter) is used to locate the report form.
QM-form-name	Specify the name of the QM Form object (object type *QMFORM) from which headings will be retrieved.

The possible library values are:

***LIBL**	All libraries in the user and system portions of the job's library list are searched until the first match is found.
***CURLIB**	The current library for the job is used to locate the database file. If no library is specified as the current library for the job, the QGPL library is used.
library-name	Specify the name of the library to be searched.

Allow information from QRYDFN

This element specifies whether a query definition (*QRYDFN) object is used when no query management query (*QMQRY) object can be found using the specified object name.

The possible options are:

***NO**	A *QRYDFN object will not be used.
***YES**	A *QRYDFN object will be used if no *QMQRY object of the specified name is found.
***ONLY**	A *QRYDFN object will be used whether a *QMQRY object with the specified name exists or not.

Output form

Not available on CVTDBFSTMF.

Specifies the form of output produced by the query.

The possible values are:

***QMQRY**	The output form is determined by the type of QM Query: If a QMFORM is specified, and that QM form includes summary functions (e.g. SUM, COUNT, AVG), both are output, as if *COMBINED has been specified, otherwise, just detail-level information is output, as if *DETAIL had been specified.
***DETAIL**	The output form produced by the query is a report containing detail records only. Summary functions in the QM form are ignored.

***COMBINED**	If a QMFORM is specified, and that QM form includes summary functions (e.g. SUM, COUNT, AVG), both are output.
***MIXED**	If a QMFORM is specified, and that QM form includes summary functions (e.g. SUM, COUNT, AVG), both are output. This differs from *COMBINED in the way in which summary rows are presented. Summary rows are output with the summary-level data values rather than corresponding text labels.

Set variables

This element is a list of up to 50 variables referenced in the QM query and the value to be assigned to those variables when the query is run.

Values in this list appear in pairs of variable names and associated values. The variable name can be from 1 to 30 characters and the value from 1 to 55 characters in length. Enclose alphanumeric variable values in single quotes (‘) but leave numeric values without quotes.

FTP – FTP parameters

Parameter	**FTP**
Description	**FTP options to be used when TOSTMF(*FTP) specified**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**TOSTMF(*FTP)**
Supports CoolSpools Database variables	**Yes**
Migration notes	**Three new parameter elements have been added in the middle of this parameter for the new format-specific commands. If you have code which references this parameter, it will need to be modified when moving away from CVTDBFSTMF to the new format-specific commands if that code references parameter elements after the insertion point for the new elements.**

The FTP parameter allows you to define parameters needed to transfer the output to an FTP server when TOSTMF(*FTP) is specified.

Secure FTP options have been added in this release and are not available on CVTDBFSTMF.

There are 10 elements to this parameter (only 7 for CVTDBFSTMF):

Remote system name/IP address

Remote file path

Port number

Secure connection (not present on CVTDFBSTMF)

Data protection (not present on CVTDFBSTMF)

Remote user id

Remote password

Encrypted password supplied (not present on CVTDFBSTMF)

Logging

CCSID for log messages

The default is the single value:

***NONE** Indicates that you do not intend to use FTP. Invalid if TOSTMF(*FTP) specified.

Remote system name/IP address

Specify the name of IP address of the system to which the data should be transmitted by FTP.

If you specify a name, the system i must be able to resolve that name to an IP address either by means of a DNS (Domain Name Server) or by looking up the name in the system Host Table.

Remote file path

Specify the full path where the output should be saved on the server. This should include both the name of the file to be created and the directory tree in which it should be saved.

Note that names on the server may be case-sensitive, especially if it is a UNIX system or similar, and may need to be enclosed in single quotes.

Port number

The port number to use,

Options are:

***FTP**	The default port for FTP (21) will be used.
***SECURE**	The default port for secure FTP (990) will be used.
Port_number	A valid port number between 1 and 65535.

Secure connection

The element does not exist for CVTDBFSTMF, which does not support FTP over SSL.

Specifies the type of security mechanism to be used for protecting information transferred on the FTP control connection (which includes the password used to authenticate the session with the FTP server). Transport Layer Security (TLS) and Secure Sockets Layer (SSL) are compatible protocols which use encryption to protect data from being viewed during transmission and verify that data loss or corruption does not occur.

Options are:

***NONE**	CoolSpools Database client does not use encryption when connecting to the specified FTP server.
***IMPLICIT**	CoolSpools Database immediately attempts to use TLS/SSL when connecting to the specified FTP server (without sending an AUTH subcommand to the server). If the server does not support implicit TLS/SSL on the specified port, or the TLS/SSL negotiation fails for any reason, the connection is closed.
***SSL**	After connecting to the specified FTP server, CoolSpools Database sends an AUTH (authorization) subcommand requesting an SSL protected session. If the server does not support SSL, the connection is closed.
***TLS**	After connecting to the specified FTP server, CoolSpools Database sends an AUTH (authorization) subcommand requesting a TLS protected session. If the server does not support TLS, the connection is closed.

Data protection

The element does not exist for CVTDBFSTMF, which does not support FTP over SSL.

Specifies the type of data protection to be used for information transferred on the FTP data connection. This connection is used to transfer file data and directory listings. The FTP protocol does not allow protection of the data connection if the control connection is not protected.

Note: The Data Protection option controls the use of the PROT (protection) FTP server subcommand.

Options are:

***DFT**	If the Secure Connection option specifies a protected control connection, *PRIVATE is used; otherwise, *CLEAR is used.
***PRIVATE**	Information sent on the FTP data connection is encrypted. If the Secure Connection option specifies that the FTP control connection is not encrypted, *PRIVATE cannot be specified.
***CLEAR**	Information sent on the FTP data connection is not encrypted.

Remote user id

The user id to use when logging on. Names may be case sensitive and may need to be enclosed in single quotes.

Remote password

The password to use when logging on. Passwords may be case sensitive and may need to be enclosed in single quotes.

See the next element for details of how to supply this password in a scrambled form to avoid having to hold passwords in plain text form in source code.

When prompting the command, if you need to enlarge the size of this parameter element to allow specification of a hex string, enter an ampersand (&) then press return and OS/400 will increase the size the field.

If you need to enter a hex string, use the form X'0123456789ABCDEF' etc.

Encrypted password supplied

The element does not exist for CVTDBFSTMF.

Whether or not the password supplied on the previous element is supplied in the encrypted form returned by CoolSpools' DSPENCPWD (Display Encrypted Password) command.

DSPENCPWD applies an encryption algorithm to a password and returns a scrambled version of that password to you. If you specify the scrambled password on the previous element, and specify *YES here, CoolSpools Database will unscramble the password for you before using it. The main purpose of this facility is to avoid the need to hold passwords in plain text form in source code.

Options are:

***NO**	The password supplied on the previous element is in plain text format and not scrambled.
***YES**	The password supplied on the previous is in the scrambled form returned by DSPENCPWD. It will be automatically unscrambled before being used.

Logging

Whether to log messages relating to the FTP file transfer.

Options are:

***JOBLOG**	Messages will be logged to the joblog.
***NONE**	No messages will be logged.
Log_file_name	The name of the file to which messages should be logged. If a database file is to be used, specify the name in IFS format (e.g. '/QSYS.LIB/library_name.LIB/file_name.FILE/member_name.MBR')

CCSID for log messages

The CCSID in which messages should be written to the file.

Options are:

***CALC**	A CCSID will be calculated as follows. If the log file is a database file, the CCSID of the job will be used. Otherwise, the ISO ASCII equivalent of the CCSID of the job will be used.
CCSID	Specify the CCSID to use.

Example:

```
CVTDBFXL   FROMFILE(SALES)…
        TOSTMF(*FTP)
        FTP(SalesSvr '/Sales/Sales.xls' *FTP 'BILL' 'soccer')
```

The sales file is converted to FTP and the output is sent directly to a server known to the system i as “SalesSvr” by FTP. The file will be saved in the “Sales” directory as “Sales.xls”. The port number will be 21. The connection will be established by logging on as BILL with the password “soccer”.

EMAIL – Email the output?

Parameter	**EMAIL**
Description	**Whether the file should be emailed after it is created/updated**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**None. Requires CoolSpools Email (product option 2) to be installed.**
Supports CoolSpools Database variables	**No**

The **EMAIL** (Email the output) parameter enables you to specify that the stream file created by CoolSpools Database should be emailed as an attachment.

Please note that this facility is only available if ***CoolSpools Email*** (CoolSpools product option 2) is installed and licenced or on trial.

Options are:

***NO** (The default). The output is not emailed automatically as part of running this command. You are still able to email the output separately, e.g. by running a subsequent email command such as SNDDST or CoolSpools Email's SNDCMNMSG.
***YES** The output from this command will be emailed as an attachment according to the information you specify on the other email-related parameters.

EMAILOPT – Email options

Parameter	**EMAILOPT**
Description	**Email-related options**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**EMAIL(*YES).** **Requires CoolSpools Email (product option 2) to be installed.**
Supports CoolSpools Database variables	**Yes**

The **EMAILOPT** (Email options) parameter allows you to specify various options relating to the sending of the output from the command as an email attachment.

These options apply only to the sending of the output using ***CoolSpools Email*** and have no effect on the sending of the output using subsequent calls to SNDDST and other email facilities.

The parameter consists of the following elements:

- Delete after sending?
- Subject
- Attach or embed? (present only on CVTDBFSTMF, CVTDBFTXT and CVTDBFHTML)
- Priority
- Confirm delivery
- Send multiple messages?
- Attachment name
- Zip attachment
- Zip file password
- Encrypted password supplied? (not available on CVTDBFSTMF)
- Save message to allow resend (not available on CVTDBFSTMF)
- Retain for how many days (not available on CVTDBFSTMF)
- Encryption method (not available on CVTDBFSTMF)
- MSF or SMTP (not available on CVTDBFSTMF)
- SMTP port number (not available on CVTDBFSTMF)
- SMTP timeout (seconds) (not available on CVTDBFSTMF)

Delete after sending

This option allows you to indicate whether the output from the command should be deleted as soon as it has been sent as an attachment.

Use this option with caution: if the email fails to arrive for whatever reason, you may lose your data.

Possible values are:

***NO** (The default). The output is not deleted.
***YES** Once the email has been created, and the stream file attached to it, the stream file is deleted. Please note that CoolSpools Database can only tell if the email has been successfully created. It cannot tell if the email was successfully sent or delivered to its recipient.

Subject

This element allows you to define a subject line for the message. You can enter up to 50 characters of free-format text. When the email message is received, the text that you enter on this parameter element will appear in the subject line of the email.

Attach or embed?

This is where you specify the method by which the file is sent.

This option is only available on those commands which can generate HTML or text output (CVTDBFSTMF, CVTDBFHTML and CVTDBFTXT). The other commands generate binary output where the *EMBED option would be inappropriate.

Options are:

***ATTACH**	(Default) The file is sent as an attachment. It will appear as an attached file separate from the text of the email.
***EMBED**	The contents of the file are embedded in the text of the email and will follow the text of any message entered on the **EMAILMSG** parameter. Please note that your client email software is likely only to support the embedding of certain types of file, e.g. text and HTML.

Priority

The priority option controls whether the email message is flagged as a high-priority or low-priority in your email client software.

Values are:

***NORMAL**	(Default) The message is sent specifying normal priority. When the message arrives, the client email software will not mark it as high or low priority.
***HIGH**	High priority. When the message arrives, the client email software will mark it as high priority.
***LOW**	Low priority. When the message arrives, the client email software will mark it as low priority.

Confirm Delivery

This option controls whether confirmation of delivery is requested from the receiver of the email.

Values are:

***NO**	(Default) No confirmation of delivery if requested.
***YES**	The message is sent with an indication that you have requested that the recipient return confirmation of

delivery. When the message is opened, if you have not switched off this feature, the client software will either send a confirmation message back to the sender of the email or ask you whether you wish to send such a confirmation.

Send Multiple Messages

This option determines whether, when multiple recipients are specified on the EMAILTO parameter, a single message is sent listing all of those recipients, or whether separate messages are sent to each recipient. This controls whether or not each recipient is aware of how else has received the message.

Values are:

***NO**	A single message is sent to a list of recipients.
***YES**	Multiple messages are sent, one to each recipient.

Attachment name

The name to be given to the attachment in the email. The default is for the name to be the same as the name of the stream file being created, but can be overridden to something else if preferred.

Values are:

***TOSTMF**	The attachment name is the same as the name of the stream file being created.
attachment_name	Specify the name to be given to the attachment.

Zip attachment

Whether the attachment is sent inside a zip file or not.

Values are:

***NO**	The attachment is not zipped prior to being attached.
***YES**	The attachment is sent inside a zip file.

Zip file password

Where the attachment is sent inside a zip file, the optional password to encrypt that zip file.

Values are:

***NONE**	No zip file is used or the zip file is not encrypted.
zip_password	Specify the case-sensitive password for the zip file.

When prompting the command, if you need to enlarge the size of this parameter element to allow specification of a hex string, enter an ampersand (&) then press return and OS/400 will increase the size the field.

If you need to enter a hex string, use the form X'0123456789ABCDEF' etc.

Encrypted password supplied

The element does not exist for CVTDBFSTMF.

Whether or not the password supplied on the previous element is supplied in the encrypted form returned by CoolSpools' DSPENCPWD (Display Encrypted Password) command.

DSPENCPWD applies an encryption algorithm to a password and returns a scrambled version of that password to you. If you specify the scrambled password on the previous element, and specify *YES here, CoolSpools Database will unscramble the password for you before using it. The main purpose of this facility is to avoid the need to hold passwords in plain text form in source code.

Options are:

<u>*NO</u>	The password supplied on the previous element is in plain text format and not scrambled.
***YES**	The password supplied on the previous is in the scrambled form returned by DSPENCPWD. It will be automatically unscrambled before being used.

Save message to allow resend

Whether or not CoolSpools will keep a copy of the email on the system after it has been sent in order to allow you to resend it using CoolSpools Email's RSNCMNMSG command.

Options are:

<u>*NO</u>	The email is not saved.
***YES**	The email is saved.

Retain for how many days

The retention period, in days, to assign to the saved email.

Note that saved emails are only deleted when you run the DLTCMNMSG (Delete Email Messages) command, at which time using the DLTSAVMSG(*MSG) option will delete saved emails that have gone past the end of the retention period.

Options are:

<u>*NOMAX</u>	No retention period is assigned.
nbr_of_days	Specify the number of days.

Encryption method

If the zip file is to be encrypted, and a password has been supplied, this element determines the encryption method.

Options are:

<u>*ENVVAR</u>	The value of environment variable CS_DFT_ZIP_ENCRYPTION sets the encryption method. If this environment variable exists, and is set to one of the other values permitted for this element (*ZIP, *AES128 or *AES256), that value is used, otherwise *ZIP is used. This provides a simple means of setting the default value for this parameter element.
***ZIP**	The original zip encryption method. This method is now considered weak and AES is recommended if strong encryption is required. However, this encryption method is likely to be more widely supported than AES, which is

	recognized by WinZip and most major zip utilities, but not all zip software.
***AES128**	128-bit AES encryption.
***AES256**	256-bit AES encryption.

EMAILFROM – Email sender information

Parameter	**EMAILFROM**
Description	**The email address and name from which the email will be sent**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**EMAIL(*YES).** **Requires CoolSpools Email (product option 2) to be installed.**
Supports CoolSpools Database variables	**Yes**

The **EMAILFROM** (Email sender information) parameter allows you to specify the sender of the email and the email address to which a response should be sent.

The default value is ***CURRENT**, which means that ***CoolSpools Email*** will try to retrieve the email address of the user sending the email from the System Distribution Directory. If no email address is defined for the user in the System Distribution Directory, you will need to enter the values you wish to use manually.

There are 2 elements to this parameter: **Email address** and **Name**.

Email address

This is where you enter the email address of the sender.

Note that while ***CoolSpools Email*** will check that the email address that you enter conforms to the rules for valid email addresses, it is not possible to validate that the email address that you enter is correct or that any reply sent to the message will be deliverable.

For example, sales.ariadnesoftware.co.uk is not a valid email address (since it does not contain an @ sign), and ***CoolSpools Email*** will reject it. However, ***sales@ariadnesoftware.org.uk*** is a valid email address and ***CoolSpools Email*** will allow it, but it is not ariadne's correct email address (it should be ***sales@ariadnesoftware.co.uk***) and any reply sent to this email address will not be received.

Name

If you would like your email message to display a sender's name rather than the sender email address when it is delivered, enter the name here.

The default value is ***NONE**, i.e. no name is provided and the email address will appear as the sender instead.

For example, if you specify:

EMAILFROM((*Sales@ariadnesoftware.co.uk* *NONE))

when the message is received, the **From:** attribute will be shown as:

From: ***Sales@ariadnesoftware.co.uk***

However, if you specify:

EMAILFROM((*Sales@ariadnesoftware.co.uk* 'ariadne Sales'))

when the message is received, the **From:** attribute will be shown as:

From: ***ariadne Sales***

EMAILTO – Email recipient(s)

Parameter	**EMAILTO**
Description	**The email address(es) and name(s) of the person(s) to whom the email will be sent**
Applies to commands:	**CVTDBFXLSX ,CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**EMAIL(*YES).** **Requires CoolSpools Email (product option 2) to be installed.**
Supports CoolSpools Database variables	**Yes**

The **EMAILTO** (Recipients) parameter allows you to specify the email addresses to which the email message should be sent.

You can define up to 32 recipients for the message on this command parameter. If you need to send the same email address to more than 32 recipients simultaneously, you can do this by defining an address list and specifying the address list name on this parameter.

The default is the single value ***SELECT:** CoolSpools Email will prompt you to enter one or more email addresses to which the message should be sent, or you can select email addresses from email address directories. Please note that this feature is not available unless you have applied CoolSpools Email PTF 1CM0053 or later.

There are 3 elements to this parameter: **Email address**, **Name** and **Type**.

Email address

This is where you enter the email address to which the message is to be sent.

Note that while ***CoolSpools Email*** will check that the email address that you enter conforms to the rules for valid email addresses, it is not possible to validate that the email address that you enter is correct or that the message will be deliverable.

For example, sales.ariadnesoftware.co.uk is not a valid email address (since it does not contain an @ sign), and ***CoolSpools Email*** will reject it. However, ***sales@ariadnesoftware.org.uk*** is a valid email address and ***CoolSpools Email*** will allow it, but it is not ariadne's correct email address (it should be ***sales@ariadnesoftware.co.uk***) and the message will not be received.

Name

If you would like your email message to display the recipient's name rather than the email address when it is delivered, enter the name here.

The default value is ***NONE**, i.e. no name is provided and the email address will appear as the recipient instead.

For example, if you specify:

EMAILTO((*Sales@ariadnesoftware.co.uk* *NONE))

when the message is received, the **To:** attribute will be shown as:

To: ***Sales@ariadnesoftware.co.uk***

However, if you specify:

EMAILTO((*Sales@ariadnesoftware.co.uk* 'ariadne Sales'))

when the message is received, the **To:** attribute will be shown as:

To: ***ariadne Sales***

Type

Specify the type of recipient here.

Options are:

***PRI**	(Default) Primary recipient.
***CC**	Carbon copy recipient. A *CC recipient receives a copy of the message, and is identified to the primary recipient, but is not the primary recipient.
***BCC**	Blind carbon copy recipient. A *BCC recipient receives a copy of the message, but is not identified to the primary recipient or *CC recipients.
***ADRL**	***CoolSpools Email*** Address list. If you wish to send to an address list, this is the value that must be entered. Refer to the CoolSpools Email manual for details of how to create, manage and use email address lists.

Example:

Sending to ariadne Sales as a primary recipient with a copy to ariadne Marketing:

```
CVTDBFXL
     EMAIL(*YES)
     EMAILTO(    (Sales@ariadnesoftware.co.uk 'Sales' *PRI)
                 (Marketing@ariadnesoftware.co.uk 'Marketing' *CC))
```

Example:

Sending to an email address list called "Sales":

```
     EMAILTO((Sales *ADRL *ADRL))
```

EMAILMSG – Email message

Parameter	**EMAILMSG**
Description	**Defines an email message to be sent with the output file**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**EMAIL(*YES).** **Requires CoolSpools Email (product option 2) to be installed.**
Supports CoolSpools Database variables	**Yes**

The **EMAILMSG** (Message) parameter allows you to enter the text of an email message directly on the command line.

Up to 512 characters of free-format text can be entered here.

The message can be sent in either plain text, HTML or alternative plain text/HTML formats.

There are 3 elements to this parameter:

- Message text
- Message format.
- Text or path name specified?

Message text

This element plays two roles, depending on the value of the third element (Text or path name specified?):

- If “Text or path name specified?” is *MSG, this element holds a free-form text message
- If “Text or path name specified?” is *STMF, this element holds the path to a file containing the text of a message.
- If “Text or path name specified?” is *MSG, you can enter the text of a short message to accompany the email and its attachment here. Up to 512 characters can be entered. When received, the message will be displayed exactly as it is entered, with the following exceptions:
- If you want to force a line break, enter
. Even if the message is sent in plain text format, this HTML control will be interpreted and converted to a hard line break (carriage return-line feed sequence).
- Other HTML controls may be entered, but will only be interpreted as HTML controls if the message is sent and delivered in HTML format.

Message format

This is where you specify the format in which the message is sent.

Options are:

***BOTH**	(Default) The message is sent in alternative plain text/HTML format. This means that two copies of the message text will be sent: a plain text copy and an HTML copy. If the email client software used to receive the message can handle HTML messages, the HTML copy will be used, otherwise the plain text copy will be used.
***TEXT**	The message is sent in plain text format. The only HTML control which is interpreted is , which ***CoolSpools Email*** will convert to a hard line break.
***HTML**	The message is sent in HTML format. You can include HTML formatting (e.g. or <u> </u> to control bold text and underlining). ***CoolSpools Email*** will take the text that you enter and wrap it with some basic HTML header and footer controls (<HTML> <HEAD> <BODY>). These controls should not therefore be included in the text of the message.

Text or path name specified?

Controls the interpretation of the first element of this parameter. See above.

Options are:

***MSG**	The first element specified is a free-format message text.
***STMF**	The first element specified is the path name of a stream file containing the text of the message to be sent.

Example:

```
CVTDBFXL…
        EMAIL(*YES)
        EMAILMSG('Here"s a message <br>with<br>line <br>breaks.' *TEXT)
```

When this message is received, it will show as:

Here's a message
with
line
breaks.

Example:

```
CVTDBFXL…
        EMAIL(*YES)
        EMAILMSG(  'Here"s a message with HTML controls.<br>
                   <b>This line is in bold, </b><br>
                   <u>While this line is underlined.</u>')
```

When this message is received, it will show as:

Here's a message with HTML controls.
This line is in bold,
While this line is underlined.

RCDFMT – Record format

Parameter	**RCDFMT**
Description	**Defines the record format(s) to be converted**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **RCDFMT** (Record Formats) parameter enables you to select which record formats from the input file are included in the conversion process.

There is one single option:

***ONLY** (Default) The input file contains only a single record format. Use this option for all database files other than logical files which contain more than one record format.

Alternatively, specify a list of from one to 20 record format names from the input file that should be included in the conversion process.

If you prompt the **CVTDBFXL** command, and specify *YES for the “Select records and fields” component of the **FROMFILE** parameter, ***CoolSpools Database*** will display a list of up to 20 record format names from the input file for you to select from.

INCLFLD – Include fields

Parameter	**INCLFLD**
Description	**Lists fields to be included in the output**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **INCLFLD** (Include Fields) parameter enables you to select which fields from the input file will be included in the output stream file, and the sequence in which they will be presented.

There is one single option:

***ALL** (Default) All fields from the input file are included and they occur in the stream in the order in which they occur in the input file. However, any fields specified on the **EXCLFLD** (Exclude Fields) parameter (see below) will be omitted.

Alternatively, specify a list of from one to 300 qualified field names from the input file that should be included in the conversion process. These fields will be presented in the stream file in the order in which they are listed on this parameter.

If you prompt the **CVTDBFXL** command, and specify *YES for the “Select records and fields” component of the **FROMFILE** parameter, ***CoolSpools Database*** will display a list of up to 300 qualified fields names from the input file for you to select from.

Each qualified field name consists of the field name and a qualifying record format name. This allows ***CoolSpools Database*** to distinguish between fields of the same name in different record formats. If the input file contains only a single record format name, the special value ***ONLY** (the default) can be specified for the record format name, indicating that the field is to be taken from the single record format in the file.

EXCLFLD – Exclude fields

Parameter	**EXCLFLD**
Description	**Lists fields to be excluded from the output**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFTXT, CVTDBFXML** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **EXCLFLD** (Exclude Fields) parameter enables you to specify fields which should not be included in the output stream file. If your input file contains a large number of fields, and you wish to exclude only a small number of fields, it may be more convenient to specify the few fields to be excluded by name rather than list the large number of fields to be included.

There is one single option:

***NONE** (Default) No fields from the input file are excluded. However, if a value other than ***ALL** is input for the **INCLFLD** (Include Fields) parameter, only the fields listed there will be included in the output stream file.

Alternatively, specify a list of from one to 300 qualified field names from the input file that should be excluded in the conversion process. These fields will be omitted from the data in the stream file.

Each qualified field name consists of the field name and a qualifying record format name. This allows ***CoolSpools Database*** to distinguish between fields of the same name in different record formats. If the input file contains only a single record format name, the special value ***ONLY** (the default) can be specified for the record format name, indicating that the field is to be taken from the single record format in the file.

EXCEL – Excel options

Parameter	**EXCEL**
Description	**Options specific to Excel output**
Applies to commands:	**CVTDBFXLSX** **CVTDBFXL** **CVTDBFSTMF (deprecated)**
Dependent on:	**TOFMT(*XLS) (CVTDBFSTMF only)**
Supports CoolSpools Database variables	**Yes (all elements that accept a free-form text value)**
Migration notes	**The EXCEL parameter of CVTDBFXL has been considerably simplified compared with the equivalent parameter of CVTDBFSTMF through the creation of a separate XLSPRPRTY parameter where Excel file properties are now defined.** **By contrast, some new elements are only available on CVTDBFXLSX and CVTDBFXL while others which have been superseded are available only on CVTDBFSTMF. See below for details.**

When using CVTDBFSTMF, the **EXCEL** (Excel options) parameter only appears if **TOFMT(*XLS)** is selected. This parameter allows you to control many aspects of the process of creating an Excel format file from your database file.

Excel file format version

This option allows you to select the version of the Excel (BIFF) file format that will be implemented when the Excel file is created.

***XLS** Excel 97-2003 workbook (.xls file).
This is a binary file format compatible with versions of Excel from Excel 97 onwards.
Please note that certain options, in particular the use of conditional formatting to change number formats or the font name, are not supported by versions of Excel prior to Excel 2007. In order to use those features, you will need to select either the *XLS07 or *XLSX file format.

***XLSX** Excel 2007 Open Office XML format (.xlsx file).
The new XML-based file format introduced with Excel 2007 and compatible with Excel 2007 and 2010.

***XLS07** Excel 2007 workbook format (.xls file).
This is an adaptation of the Excel 97-2003 format with extensions to support certain new features such as the ability to modify number formatting or the font name using conditional formatting. If you wish to use features introduced with Excel 2007 and do not wish to use *XLSX format, you must use this format, but please note that the new features will not be available if the file is opened in Excel 97, Excel 2000 or Excel 2003.

Please also note that (confusingly) this is not the same as the Excel 2007 Excel binary workbook format (file extension .xlsb), which is not supported.

***BIFF8** The same as *XLS, provided for backwards-compatibility.

Note that support for BIFF5 format (Excel 95) is now withdrawn.

Note the following limits imposed by Excel (not CoolSpools Database):

Attribute	BIFF 8 Maximum	Open Office XML Maximum
Maximum rows in a worksheet	65,536	1,048,576
Maximum columns in a worksheet	256	16,384

If the number of records in the input file exceeds the maximum number of rows per worksheet for the format being implemented, ***CoolSpools Database*** will create additional worksheets for the overflow.
If the number of fields in the record being converted exceeds the maximum number of columns per worksheet for the format being implemented, ***CoolSpools Database*** will drop fields beyond the maximum.

Default edit code

Not available on CVTDBFXL and CVTDBFXLSX. The new Define Styles (DFNSTYLES) and Apply Styles (APYSTYLES) parameters, available on the CVTDBFXL command, provide greater control over the formatting of data and use of this feature is now deprecated.

This option controls the format in which numeric information in the input file is presented in the Excel file.

Where numeric fields in the input file have DDS edit codes or edit words defined for them, ***CoolSpools Database*** will convert the edit code or edit word to an Excel custom format for the corresponding column in the spreadsheet. This means that the data in the column will be displayed in a format which reflects the edit code or edit word of the original database field.

Where numeric fields in the input file do not have a DDS edit code or edit word associated with them, ***CoolSpools Database*** will format the data in the spreadsheet according to the value you enter on this parameter instead.

The value you enter must be ***NONE** (the default) to indicate that you do not wish to have numeric data edited in this way, or a valid system i edit code from the following table.

Edit Codes	Commas[1] Displayed	Decimal Points[1] Displayed	Sign Displayed When Negative Value	Blank Value of QDECFMT System Value	I Value of QDECFMT System Value	J Value of QDECFMT System Value	Leading Zero Suppressed
1	Yes	Yes	No sign	.00 or 0	,00 or 0	0,00 or 0	Yes
2	Yes	Yes	No sign	Blanks	Blanks	Blanks	Yes

3		Yes	No sign	.00 or 0	,00 or 0	0,00 or 0	Yes
4		Yes	No sign	Blanks	Blanks	Blanks	Yes
A	Yes	Yes	CR	.00 or 0	,00 or 0	0,00 or 0	Yes
B	Yes	Yes	CR	Blanks	Blanks	Blanks	Yes
C		Yes	CR	.00 or 0	,00 or 0	0,00 or 0	Yes
D		Yes	CR	Blanks	Blanks	Blanks	Yes
J	Yes	Yes	-(Minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
K	Yes	Yes	-(Minus)	Blanks	Blanks	Blanks	Yes
L		Yes	-(Minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
M		Yes	-(Minus)	Blanks	Blanks	Blanks	Yes
N	Yes	Yes	-(Minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
O	Yes	Yes	-(Minus)	Blanks	Blanks	Blanks	Yes
P		Yes	-(Minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
Q		Yes	-(Minus)	Blanks	Blanks	Blanks	Yes
W[2]							Yes
Y[3]							Yes
Z[4]							Yes

Notes:

1. The QDECFMT system value determines the decimal point character (period in U.S. usage), the character used to separate groups of three digits (comma in U.S. usage), and the type of zero suppression (depending on comma and period placement).
2. The W edit code suppresses the farthest left zero of a date field that is five digits long. It also suppresses the three farthest left zeros of a field that is six to eight digits long. The W edit code also inserts slashes (/) between the month, day, and year according to the following pattern:

   nn/nnn

   nnnn/nn

   nnnn/nnn

   nnnn/nn/nn

3. The Y edit code suppresses the farthest left zero of a date field that is three to six digits long or eight digits long. It also suppresses the two farthest left zeros of a field that is seven positions long. The Y edit code also inserts slashes (/) between the month, day, and year according to the following pattern:

   nn/n

   nn/nn

   nn/nn/n

   nn/nn/nn

   nnn/nn/nn

   nn/nn/nnnn

   If the DATE keyword is specified with EDTCDE(Y), the separator character used is the job attribute, DATSEP at run

time. The slash (/) is the default DATSEP.

4. The Z edit code removes the sign (plus and minus) from a numeric field. The sign of the units position is changed to a hexadecimal F before the field is written.

Floating currency symbol

Not available on CVTDBFXL and CVTDBFXLSX. Use the new User-defined Styles (DFNSTYLES) and APYSTYLES (Field Styles) parameters instead as these provide greater control over the formatting of data and use of this feature is now deprecated.

This option controls whether, when a default edit code has been specified on the previous option, a currency symbol (corresponding to the system value QCURSYM) is displayed immediately to the left of the left-most significant digit.

***NO**	(Default). No floating currency symbol appears.
***YES**	Numeric data edited using the default edit code (described above) is preceded by the floating currency symbol. The currency symbol is taken from the QCURSYM system value.

Date format

This option controls the default format in which dates from the input file are presented in the Excel file.

Excel represents dates as a count of days. ***CoolSpools Database*** will convert date fields in the input database file to an integer cell containing the appropriate day count and will apply formatting to display the value as a date. This makes it easy for you to apply formulas to the date subsequently.

In the main, the format in which date information is displayed in the Excel spreadsheet is determined by the regional settings on your PC. This parameter provides some options to vary that formatting slightly.

You can specify one of the following single values for this option:

***MM**	(Default) Months are displayed as 2-digit numeric values, e.g. 01=January.
***MMM**	Months are displayed as 3-character values, e.g. JAN=January.
***LABEL**	Dates are treated in the same way as character fields and output as an alphanumeric label cell rather than as an integer cell with date formatting.

Time format

This option controls the format in which time data from the input file is presented in the Excel file. Please note that this only applies to actual time fields in the database file: numeric and alphanumeric fields which contain times cannot be identified as times by ***CoolSpools Database***.

Excel represents times as a fraction of a day, e.g. 12 noon = 0.5. ***CoolSpools Database*** will convert time fields in the input database file to a cell containing a floating point value representing a day fraction of this type and will apply formatting to display the value as a time. This parameter controls the type of formatting that is applied.

You can specify one of the following single values for this option:

***HMS**	(Default) Times are displayed using the 24-hour clock including seconds, e.g. 3:30 in the afternoon is displayed something like 15:30:00.
***HM**	Times are displayed using the 24-hour clock excluding seconds, e.g. 3:30 in the afternoon is displayed something like 15:30.
***HMSAM**	Times are displayed using the AM/PM format including seconds, e.g. 3:30 in the afternoon is displayed something like 3:30:00 PM.
***HMAM**	Times are displayed using the AM/PM format excluding seconds, e.g. 3:30 in the afternoon is displayed something like 3:30 PM.

The actual format of the time displayed may differ from these examples (in particular the separator character may vary) as the formatting is also influenced by the regional options on the PC.

Worksheet name

This option allows you to specify the name given to worksheets in the Excel file.

Options are:

***DFT**	(Default) The default worksheet names are used. The default worksheet name is stored in the message text of message SLP5000 in the CoolSpools Database message file CP_MSGF. It is supplied as “Sheet1” (the standard Excel English-language worksheet name) when CoolSpools Database is first installed. However, you can change the text of this message if you wish to modify the default worksheet name. Note that you will need to repeat this change each time a PTF or new version of CoolSpools Database is installed.
‘Sheet name’	Enter a valid Excel worksheet name. Excel worksheets names are from 1-31 characters in length and can contain any characters except: colon (:), backslash (\), forward slash (/), question mark (?), asterisk (*), left-hand square bracket ([] and right-hand square bracket (]).

If the data will not fit into a single worksheet, ***CoolSpools Database*** will create additional worksheets up to a maximum of 36. The names of the second worksheet and subsequent worksheets are based on the value specified on this parameter according to the following rules:

1. If the name specified on this parameter, or the text retrieved from message id SLP5000 in message file CP_MSGF if *DFT is specified, ends in a '1' (e.g. English "Sheet1", or German "Blatt1" or Dutch "Blad1"), this suffix is replaced by 2, 3, 4… (e.g. "Sheet2","Sheet3"..., "Blatt2", Blatt3"..., "Blad2", Blad3"...).
2. If the name does not end in a 1, a numeric suffix is appended to the name. For example, if the name is "Invoices", subsequent worksheets will be called "Invoices2", "Invoices3"...

Column width option

Determines the way in which CoolSpools Database sets the column width.

***ENVVAR** Calculate the width of the column based on the setting of the CS_XLS_COLUMN_WIDTH environment variable. If this environment variable does not exist, or if it is not set to one of the following values, the value *AUTOFIT is assumed:

- *AUTOFIT
- *FIELDSIZE
- *FITTEXT
- *FIELDSIZE

Base the column width on the size of the field, according to its DDS definition. For example, an alphanumeric field that has a size of 50 characters will have the corresponding column width set to 50 characters too.

***AUTOFIT** Base the column width on the length of the largest data value, measured in characters.

For example, an alphanumeric field that has a size of 50 characters but where the longest value is 40 characters will have the corresponding column width set to 40 characters.

***FITTEXT** Base the column width on the character width of the largest data value. The character width here is the width of the text as displayed in the selected font, measured in points.

This option may give the best results in terms of fitting the column width closely to the width of the displayed text, but please note the following points:

- Use of this option can cause a noticeable degradation in performance, because of the need to calculate the displayed width of each text item

- The displayed width of text items can only be calculated accurately where one of the "well known" fonts (Arial, Courier or Times) is used.
- Excel can use different column width settings when printing data as opposed to when data is displayed on screen. If columns are fitted too closely to the width of the text when displayed on screen, the data may appear as ##### when printed (i.e. might not fit in the column width).

Note:

The way in which Excel calculates column widths is complex. More specifically, it is dependent on the font metrics of the default font for the workbook, which in itself can be specified to be any named font. Since CoolSpools Database runs on the iSeries and cannot therefore readily access those font metrics, when you specify a font other than one of CoolSpools Database's "well known" fonts (Courier New, Arial, Times Roman), or where you are adding to an existing spreadsheet that uses a font other than the "well known" fonts, CoolSpools Database cannot guarantee to calculate columns widths precisely.

Maximum rows per worksheet

The maximum number of rows that will be written to a worksheet before starting a new worksheet.

Options are:

***XLSVER**	(Default) The maximum is dictated by the version of Excel being output (65536 for *XLS and 1048576 for *XLSX)
1-65536	Specify the maximum. When this number of rows has been written to a worksheet, Excel will start a new overflow worksheet.

Hide unused columns

Not available from CVTDBFSTMF.

Whether unused columns are hidden or not.

Options are:

***NO**	(Default) Unused columns are not hidden and will appear as empty columns to the right of the last used column of data.
***YES**	Empty columns will be hidden and the last used column will be the last visible column in the worksheet.

Hide unused rows

Not available from CVTDBFSTMF.

Whether unused rows are hidden or not.

Options are:

***NO** (Default) Unused rows are not hidden and will appear as empty rows to the right of the last used row of data.

***YES** Empty rows will be hidden and the last used row will be the last visible row in the worksheet.

Set Column formatting

Specifies whether column-level formatting should be applied.

***YES**

Formatting is applied at column level. This means that unused cells in the column will share attributes such as number format with used cells in the same column.

***NO**

Formatting is not applied at column level. This means that unused in the column will inherit attributes from the default style.

Number of columns to freeze

Specifies the number of columns to freeze.

Autofilter on?

Specifies whether filters are to be applied or not.

***YES**

Specifies if the Filter option within excel, is set on for all the columns**.**

***NO**

Specifies that the Filter option in excel is not used.

Title

See XLSPRPRTY parameter.

Subject

See XLSPRPRTY parameter.

Author

See XLSPRPRTY parameter.

Manager

See XLSPRPRTY parameter.

Company

See XLSPRPRTY parameter.

Category

See XLSPRPRTY parameter.

Keywords

See XLSPRPRTY parameter.

Comments

See XLSPRPRTY parameter.

Document content status

See XLSPRPRTY parameter.

Font Name

CVTDBFSTMF only. Use of this option is now deprecated. Use the DFNSTYLES and APYSTYLES parameters to define named styles instead.

Specifies the font to be used in the Excel file.

***ARIAL**	(Default) The Arial font is used.
***COURIER**	Courier New.
***TIMES**	Times New Roman
***CALIBRI**	Calibri
Font_name	Specify the name of the font to be used. This must match a font installed on the PC which opens the file, otherwise Excel will substitute a different font.

Font Size

CVTDBFSTMF only. Use of this option is now deprecated. Use the DFNSTYLES and APYSTYLES parameters to define named styles instead.

Specify the size of the font to be used in the Excel file, measured in points.

10	(Default) A 10-point font is used.
Font_size	Specify the size of the font in points.

RPTBRKS – Report Breaks

Parameter	**RPTBRKS**
Description	**Options specific to Report Breaks**
Applies to commands:	**CVTDBFXLSX**
Dependent on:	
Supports CoolSpools Database variables	**No**

Specifies options available for report breaking.

RPTBRKS – Report breaks

Identify by

***FLDNAM** The following parameter element contains a field name identifying the field in the input file to which the styling relates.

***FLDNBR** The following parameter element contains a field number identifying the field in the input file to which the styling relates. The field number denotes the sequential position of the field in the file, e.g. 1=1st field, 2=2nd field. This can be useful where the field names cannot be easily identified, for example when the input to the command is a piece of SQL including functions such as SUM, AVG etc.

***XLCOLID** The following parameter element contains a column letter or letters indicating the column in the worksheet to which the styling relates.

You can also specify a range of column identifiers, e.g. A-C (columns A to C), although OS/400 will insist you enclose this in quotes: 'A-C'.

Fld name, nbr, col id, map ref

Identifies the field or column this parameter relates to. Styling defined here will be applied to this field.

name or number

If *FLDNAM was specified for the previous element, this value is the name of a field in the input file to which the styling relates.

If no record format name is specified, it is assumed that the field comes from the first record format. If that is not the case, use the form: 'record_format_name/field_name'

If *FLDNBR was specified for the previous element, this value is the sequential number of a field in the input file (e.g. 1=1st field, 2=2nd field etc.) to which the styling relates.

If no record format name is specified, it is assumed that the field comes from the first record format. If that is not the case, use the form:

If a qualified field name is required, use the form: 'record_format_name/field_number'

If *XLCOLID was specified for the previous element, this value is the column letter (A-Z, AA-ZZ etc.) of the column the styling relates to. You can also specify a range of column identifiers, e.g. A-C (columns A to C), although OS/400 will insist you enclose this in quotes: 'A-C'.

Apply style name

Which style is applied to data rows for this field.

***DATA**

The predefined *DATA style is applied to header rows.

***HEADER**

The predefined *HEADER style is applied to header rows.

***TITLE**

The predefined *TITLE style is applied to header rows.

***SUBTOTAL**

The predefined *SUBTOTAL style is applied to header rows.

***TOTAL**

The predefined *TOTAL style is applied to header rows.

***HYPERLINK**

Converts the field to a hyperlink.

***FLWDLINK**

Converts the field to give the appearance of an hyperlink already visited.

***NAME**

Specify the name of the user-defined style to apply to header rows.

RPTSMRY– Report Summary

Parameter	**RPTBRKS**
Description	**Options specific to Report Breaks**
Applies to commands:	**CVTDBFXLSX**
Dependent on:	**RPTBRKS being specified previously**
Supports CoolSpools Database variables	**No**

Specifies options available when report summaries are required.

Please note that when summarising, the print area in the excel spreadsheet being created, may not extend fully, to cover the totals calculated. The print area may have to be adjusted.

RPTSMRY – Report summary functions

Identify by

***FLDNAM**	By field name
***FLDNBR**	by field number
***XLCOLID**	by Excel column letter

Fld name, nbr, col id, map ref

Summary function

***AVG**	Average
***COUNT**	Count numbers
***COUNTA**	Count all
***MAX**	Maximum
***MIN**	Minimum
***PRODUCT**	Product
***STDEV**	Standard deviation
***STDEVP**	Population std. dev.
***SUM**	Sum
***VAR**	Variance
***VARP**	Population variance

EXAMPLE

```
CVTDBFXLSX FROMFILE(QCUSTCDT)
        TOSTMF('CustomerDue.xlsx')
        STMFOPT(*REPLACE)
        RPTBRKS((*FLDNAM STATE))
        RPTSMRY((*FLDNAM BALDUE *SUM *SUBTOTAL))
        SORT((STATE *ASCEND *NO))
```

Here the CVTDBFXLSX command is being applied to a database file called QCUSTCDT and converted to a stream file called CustomerDue.xlsx. The rows will be in STATE order with a break on change of STATE. The Balance Due (BALDUE) for customers within each state will be summed and included on the break line.

XLSPRPRTY – Document properties

Parameter	**XLSPRPRTY**
Description	**Specifies document properties for Excel files.**
Applies to commands:	**CVTDBFXLSX** **CVTDBFXL** **CVTDBFSTMF (deprecated)**
Dependent on:	**TOFMT(*XLS) (CVTDBFSTMF only)**
Supports CoolSpools Database variables	**Yes**
Migration notes	**The EXCEL parameter of CVTDBFXL has been considerably simplified compared with the equivalent parameter of CVTDBFSTMF through the creation of this separate XLSPRPRTY parameter where Excel file properties are now defined.**

This parameter allows you to define file properties for documentation and audit purposes.

The information defined here appears in Excel 2007 when you select:

Office button -> Prepare -> Properties

Title

***NONE** (Default) The file will have no title.
Title_text Up to 32 characters of title text.

Subject

***NONE** (Default) The file will have no subject.
Subject_text Up to 32 characters of subject text.

Author

A number of special values are available to help you use this field to document the origin of the file.

***NONE** (Default) The file will have no title.
***USRPRF** The user id of the user that created the file, e.g. SALESUSER.
***JOB** The name of the job that created the file, e.g. SALESJOB,
***QUALJOB** The qualified name of the job that created the file, e.g. 123456/SALESUSER/SALESJOB.
Author_text Up to 32 characters of author text.

Manager

***NONE** (Default) The file will have no manager.

Manager_text Up to 32 characters of manager text.

Company

***NONE** (Default) The file will have no company.
Company_text Up to 32 characters of company text.

Category

***NONE** (Default) The file will have no category.
Category_text Up to 32 characters of category text.

Keywords

***NONE** (Default) The file will have no keywords.
Keywords_text Up to 128 characters of keywords text.

Comments

***NONE** (Default) The file will have no comments.
Comments_text Up to 256 characters of comments text.

Document content status

***NONE** (Default) The file will have no document content status
Status_text Up to 32 characters of text describing the status of the document content (e.g. “Draft”, “Final”, “Approved” etc.)

XLSPROTECT – Excel worksheet protection

Parameter	**XLSPROTECT**
Description	**Excel worksheet protection**
Applies to commands:	**CVTDBFXLSX** **CVTDBFXL** **CVTDBFSTMF (deprecated)**
Dependent on:	**TOFMT(*XLS) (CVTDBFSTMF only)**
Supports CoolSpools Database variables	**Yes**
Migration notes	**Note that CVTDBFXL has an extra element compared with CVTDBFSTMF. Note also that columns can now be optionally left unlocked when a worksheet is protected by using a named style with the locking attribute set appropriately.**

This parameter allows you to specify options related to worksheet protection.

The default is the single option ***NO**, which indicates that no protection will be applied to the worksheet.

Note that if using STMFOPT(*ADD), this parameter only affects new worksheets added to the file and does not modify the protection status of existing worksheets in the file.

Elements are as follows.

Protect worksheet

Specify ***YES** to activate worksheet protection. When the file is opened, the worksheet will be protected. Cells to which a style has been supplied that indicates that cells should be locked will not be modifiable, except as permitted by the options specified below.

Worksheet protection password

Specifies the password that must be entered to unprotect the worksheet.

***NONE** (Default) No password is required. The worksheet can be unprotected simply by taking the appropriate menu option.

Password Specify the password that must be entered in order to unprotect the worksheet. This is case-sensitive and a maximum of 32 characters in length.

Encrypted password supplied

The element does not exist for CVTDBFSTMF.

Whether or not the password supplied on the previous element is supplied in the encrypted form returned by CoolSpools's DSPENCPWD (Display Encrypted Password) command.

DSPENCPWD applies an encryption algorithm to a password and returns a scrambled version of that password to you. If you specify the scrambled password on the previous element, and specify *YES here, CoolSpools Database will unscramble the password for you before using it. The main purpose of this facility is to avoid the need to hold passwords in plain text form in source code.

Options are:

***NO**	The password supplied on the previous element is in plain text format and not scrambled.
***YES**	The password supplied on the previous is in the scrambled form returned by DSPENCPWD. It will be automatically unscrambled before being used.

Allow actions

Defines the actions that can be applied to locked items on a protected worksheet.

Single options are:

***DFT**	(Default) The actions allowed by Excel by default when a worksheet is protected are permitted. Both locked and unlocked cells may be selected, and objects and scenarios may be edited.
***NONE**	No actions are permitted on locked cells.

Alternatively, specify the actions to be permitted from the following list:

***DLTCOLS**	Deletion of columns
***DLTROWS**	Deletion of rows
***AUTOFILTER**	Applying autofilters
***EDTOBJ**	Editing objects
***EDTSCN**	Editing scenarios
***FMTCELLS**	Changing the formatting of cells
***FMTCOLS**	Changing the formatting of columns
***FMTROWS**	Changing the formatting of rows
***INSCOLS**	Inserting columns
***INSROWS**	Inserting rows
***INSLINKS**	Inserting hyperlinks
***PIVOTTABLE**	Applying pivot tables
***SLTUNLOCKED**	Selecting unlocked cells
***SLTLOCKED**	Selecting locked cells
***SORT**	Sorting rows

XLSPRINT – Excel print setup

Parameter	**XLSPRINT**
Description	**Specifies Excel print options**
Applies to commands:	**CVTDBFXLSX** **CVTDBFXL** **CVTDBFSTMF (deprecated)**
Dependent on:	**TOFMT(*XLS) (CVTDBFSTMF only)**
Supports CoolSpools Database variables	**Yes.** **Excel placeholders can also be specified on the header and footer text options.**

Specifies print options for Excel spreadsheets.

Single values:

***DFT** — No print options are specified in the Excel file. Excel defaults will be used.

Other values:

Scaling

How the data is enlarged or reduced when you print so that it fits the required number of pages.

Specify *FIT and a number of pages wide and tall on the XLSFITPAGE parameter to fit the data to the required number of pages.

Specify *ADJUST and a percentage on the XLSADJUST parameter to scale the data by that percentage.

Options are:

***FIT** — Fit the data to a number of pages wide and a number of pages tall. The number of pages wide and tall are specified on the dependent parameter XLSFITPAGE.

***ADJUST** — Adjust the data by applying a percentage scaling. The percentage by which the data is scaled is specified on the dependent parameter XLSADJUST.

Page size

The paper size.

Options are:

***CNTRYID** — The paper size is determined by the country id of the job. If this is US (USA) or CA (Canada), letter paper is selected, otherwise A4 paper is selected.

***A3**	A3 (420 x 297 mm).
***A4**	A4 (297 x 210 mm).
***A5**	A5 (210 x 148 mm).
***B4**	B4 (364 x 257 mm).
***B5**	B5 (257 x 182 mm).
***LETTER**	Letter (11.5 x 8 inches).
***LEGAL**	Legal (14 x 8.5 inches).
***EXEC**	Executive (10.5 x 7.25 inches).
***LEDGER**	Ledger (17 x 11 inches).

Orientation

The page orientation.

Options are:

***LANDSCAPE**	Landscape mode.
***PORTRAIT**	Portrait mode.

Print gridlines

Whether gridlines should be printed or not.

Options are:

***NO**	Gridlines are not printed.
***YES**	Gridlines are printed.

Rows to repeat at topWhether any header row should be printed on each page.

Options are:

***NO**	The header row, if requested on the HEADER parameter, is printed only on the first page.
***YES**	The header row, if requested on the HEADER parameter, is printed on each page.

Unit of measure

The unit of measure in which margins are defined (see below)

Options are:

***INCH**	Inches
***MM**	Millimeters
***CM**	Centimeters

Left margin

The left page margin measured in the units specified (see Unit of Measure above).

Right margin

The right page margin measured in the units specified (see Unit of Measure above).

Top margin

The top page margin measured in the units specified (see Unit of Measure above).

Bottom margin

The bottom page margin measured in the units specified (see Unit of Measure above).

Page header left section

The text to appear in the left section of the page header.

CoolSpools Database variables and Excel placeholders are supported on this parameter.

Page header center section

The text to appear in the center section of the page header.

CoolSpools Database variables and Excel placeholders are supported on this parameter.

Page header right section

The text to appear in the right section of the page header.

CoolSpools Database variables and Excel placeholders are supported on this parameter.

Page footer left section

The text to appear in the left section of the page footer.

CoolSpools Database variables and Excel placeholders are supported on this parameter.

Page footer center section

The text to appear in the center section of the page footer.

CoolSpools Database variables and Excel placeholders are supported on this parameter.

Page footer right section

The text to appear in the right section of the page footer.

CoolSpools Database variables and Excel placeholders are supported on this parameter.

Header Margin

The header margin measured in the units specified. This is the space available for the print headings defined above.

CoolSpools Database variables and Excel placeholders are supported on this parameter.

Footer Margin

The footer margin measured in the units specified. This is the space available for the print footings defined above.

CoolSpools Database variables and Excel placeholders are supported on this parameter.

Print Quality

The print resolution in dots per inch (DPI).

Number of Copies

The number of copies to print.

XLSADJUST – Adjust pages to

Parameter	**XLSADJUST**
Description	**Specifies the percentage scaling when XLSPRINT(*ADJUST...) is requested.**
Applies to commands:	**CVTDBFXLSX** **CVTDBFXL** **CVTDBFSTMF (deprecated)**
Dependent on:	**XLSPRINT(*ADJUST)** **TOFMT(*XLS) (CVTDBFSTMF only)**
Supports CoolSpools Database variables	**No**

Specifies the percentage scaling when XLSPRINT(*ADJUST...) is used.

Options are:

100	Scale by 100% (no change).
0-400	Specify the percentage scaling.

XLSFITPAGE – Fit pages to

Parameter	**XLSFITPAGE**
Description	**Specifies the number of pages to fit the output to when XLSPRINT(*FITPAGE...) is requested.**
Applies to commands:	**CVTDBFXLSX** **CVTDBFXL** **CVTDBFSTMF (deprecated)**
Dependent on:	**XLSPRINT(*FIT)** **TOFMT(*XLS) (CVTDBFSTMF only)**
Supports CoolSpools Database variables	**No**

Specifies the number of pages to which the output is fitted when XLSPRINT(*FIT...) is used.

There are two elements:

The number of pages wide (horizontal).

Options are:

***AUTO**	Excel will calculate the number of pages required automatically.
0-65535	Specify the number of pages to which the data should be fitted horizontally.

The number of pages tall (vertical).

Options are:

***AUTO**	Excel will calculate the number of pages required automatically.
0-65535	Specify the number of pages to which the data should be fitted vertically.

CSV – CSV options

Parameter	**CSV**
Description	**Specifies options for delimited ASCII text output (typically comma-separated variable but also tab-separated etc.)**
Applies to commands:	**CVTDBFCSV** **CVTDBFSTMF (deprecated)**
Dependent on:	**TOFMT(*CSV) (CVTDBFSTMF only)**
Supports CoolSpools Database variables	**No**
Migration notes	**The format of this parameter has been altered somewhat between the CVTDBFSTMF and CVTDBFCSV commands. Specifically, CVTDBFSTMF allows the use of a single value *DFT which is not available on CVTDBFCSV. Also, on the CSV parameter of the CVTDBFSTMF command, the date format and date separator formed a related pair for which a single value could be specified, as did the time format and time separator. This tended to make use of this parameter excessively complicated. The CSV parameter of the CVTDBFCSV command has been re-designed to improve ease of use, but code changes may be necessary when migrating from CVTDBFSTMF ...TOFMT(*CSV) to use of the CVTDBFCSV command.**

The **CSV** (CSV options) parameter only appears if **TOFMT(*CSV)** is selected. This parameter allows you to control several aspects of the process of creating a delimited ASCII file from your database file.

Field delimiter

This option allows you to define the character that separates fields in the delimited ASCII file that is to be created.

Either type the character to be used, or select one of the special values:

***COMMA** (Default) A comma (,) is used
***TAB** A tab (x'09') is used
PIPE** A pipe character (| ASCII x'7C')BLANK** A blank (x'20') is used
***SEMICOLON** A semicolon (;) is used.

Example:

CVTDBFCSV FROMFILE(SALESSTATS)…

CSV(*COMMA *DBLQUOTE *CRLF)

The Sales Stats file is converted to a delimited file in CSV (Comma-separated variable format). Fields are separated by commas. Alphanumeric data is enclosed in double quotes. Records are terminated by a carriage return/line feed pair.

String delimiter

This element allows you to define the character that encloses string (alphanumeric) data in the delimited ASCII file that is to be created.

Either type the character to be used, or select one of the special values:

***DBLQUOTE**	A double quote (“) is used
***SGLQUOTE**	A single quote (‘) is used
***NONE**	No delimiter is used. Alphanumeric data is not enclosed by any special character.

Record delimiter

This element allows you to specify the characters to be used to indicate the end of a record in the CSV file.

Options are:

***CRLF**	Carriage return and line feed. Both a carriage return (x’0D’) and a line feed (x’0A’) character are used.
***CR**	Just a carriage return (x’0D’) is used.
***LF**	Just a line feed (x’0A’) is used.

Date format

This option controls the format in which date information in the database file is presented in the stream file. Please note that this only applies to actual date fields in the database file: numeric and alphanumeric fields which contain date information cannot be identified as dates by ***CoolSpools Database***.

***DBF**	(Default). The format of the date is determined by the date format specified for the date field in the database file (DDS DATFMT and DATSEP keywords).
***EXCEL**	The date is converted to Excel format, i.e. a numeric value representing a day count. This format is ideal if you are going to load the CSV file into Excel or a similar application. Excel will recognize the data as a date and allow date functions and editing to be applied to it.
***ISO**	ISO format (YYYY-MM-DD) is used.
***EUR**	European format (DD.MM.YYYY) is used.
***USA**	USA format (MM/DD/YYYY) is used.
***JIS**	Japanese Industrial Standard (YYYY-MM-DD) is used.
***SYSVAL**	The date format defined in system value QDATFMT is used.
***JOB**	The date format defined in the job attributes is used.
***DMY**	DDMMYY format.
***DMYY**	DDMMYYYY format.
***MDYY**	MMDDYYYY format.

***YYMD** YYYYMMDD format.

***CDMY** CDDMMYY format. The C indicates the century (0 = 20th, 1=21st)

***CMDY** CMMDDYY format. The C indicates the century (0 = 20th, 1=21st)

***CYMD** CYYMMDD format. The C indicates the century (0 = 20th, 1=21st)

***JUL** YYDDD format

***LONGJUL** YYYYDDD format

Date separator

***JOB** The date separator character defined in the job attributes is used.

***NONE** No date separator character is used.

***SLASH** A forward slash (oblique or solidus) / is used.

***HYPHEN** A hyphen (dash) – is used

***PERIOD** A period (full stop) . is used.

***COMMA** A comma , is used.

***COLON** A colon : is used.

***BLANK** A blank is used.

separator_char Type the separator character to be used to separate the day, month and year portions of the date.

Time format

This option controls the format in which time information in the database file is presented in the stream file. Please note that this only applies to actual time fields in the database file: numeric and alphanumeric fields which contain times cannot be identified as times by ***CoolSpools Database***.

***DBF** (Default) The format of the time is determined by the time format specified for the time field in the database file (DDS TIMFMT and TIMSEP keywords).

***EXCEL** The time is converted to Excel format, i.e. a numeric value representing a number of seconds. This format is ideal if you are going to load the CSV file into Excel or a similar application. Excel will recognize the data as a time and allow time functions and editing to be applied to it.

***ISO** ISO format (HH.MM.SS) is used.

***EUR** European format (HH.MM.SS) is used.

***USA** USA format (HH:MM:SS) is used.

***JIS** Japanese Industrial Standard (HH:MM:SS) is used.

***HMS** HHMMSS format.

Time Separator

***SYSVAL** The time separator defined by system value QTIMFMT is used.

***JOB** The time separator character defined in the job attributes is used.

***NONE**	No time separator character is used.
***PERIOD**	A period (full stop) . is used.
***COMMA**	A comma , is used.
***COLON**	A colon : is used.
***BLANK**	A blank is used.
separator_char	Type the separator character to be used to separate the day, month and year portions of the date.

Decimal point character

This element allows you to specify the characters to be used to denote a decimal point when representing fields with one or more decimal places.

Options are:

***SYSVAL**	The decimal format defined by system value QDECFMT is used.
***JOB**	The decimal format defined in the job attributes is used.
***PERIOD**	A period (full stop). is used.
***COMMA**	A comma , is used.

Apply edit codes and words

Whether CoolSpools Database edits numeric values with their associated edit code or edit word before outputting the data.

For example, if a packed decimal field contains the value -123456.78 and has an associated edit code of $M, and *YES is specified for this element, the field value will be output as **$123,456.78-** rather than 123456.78.

Options are:

***NO**	Edit codes and edit words are ignored
***YES**	Numeric fields with associated edit codes or edit words (including editing defined in a Query/400 query or QM/Query form) will be edited before being output.

Trim blank from char fields

Whether CoolSpools Database trims leading and/or trailing blanks when outputting character values.

Options are:

***BOTH**	Leading and trailing blanks are trimmed from character fields before they are output.
***LEADING**	Only leading blanks are trimmed
***TRAILING**	Only trailing blanks are trimmed
***NONE**	No blanks are trimmed.

Excel import with leading zero

(CVTDBFCSV only)

Whether CoolSpools Database adds an equals sign (=) prior to the string delimiter of character fields in order to force Excel to import the field with leading zeros preserved. Without this, Excel removes leading zeros from fields in CSV files even where the field is denoted as a text field by a string delimiter such a double quotes.

Options are:

***NO**	No equals sign is added at the front of character fields with leading zeros.
***YES**	An equals sign is added at the front of character fields with leading zeros.

FIXED – Fixed text options

Parameter	**FIXED (CVTDBFSTMF)** **TEXT (CVTDBFTXT)**
Description	**Specifies options for fixed-width ASCII text output**
Applies to commands:	**CVTDBFTXT** **CVTDBFSTMF (deprecated)**
Dependent on:	**TOFMT(*FIXED) (CVTDBFSTMF only)**
Supports CoolSpools Database variables	**No**
Migration notes	**The parameter of CVTDBFTXT corresponding to the FIXED parameter of the CVTDBFSTMF command is called TEXT for consistency with the name of the command and the TEXT parameter of CoolSpools's CVTSPLTXT command etc. Its format has also been modified somewhat.** **Specifically, the CVTDBFSTMF FIXED parameter allows the use of a single value *DFT which is not available on the CVTDBFTXT TEXT parameter. Also, on the FIXED parameter of the CVTDBFSTMF command, the date format and date separator formed a related pair for which a single value could be specified, as did the time format and time separator. This tended to make use of this parameter excessively complicated. The TEXT parameter of the CVTDBFTXT command has been re-designed to improve ease of use, but code changes may be necessary when migrating from CVTDBFSTMF ...TOFMT(*FIXED) to use of the CVTDBFTXT command.**

Specifies options related to fixed-width text output.

Record delimiter

This element allows you to specify the characters to be used to indicate the end of a record in the text file.

Options are:

***CRLF**	Carriage return and line feed. Both a carriage return (x'0D') and a line feed (x'0A') character are used.
***CR**	Just a carriage return (x'0D') is used.
***LF**	Just a line feed (x'0A') is used.

Date format

This option controls the format in which date information in the database file is presented in the stream file. Please note that this only applies to actual date fields in the database file: numeric and alphanumeric fields which contain date information cannot be identified as dates by ***CoolSpools Database***.

***DBF**	(Default) The format of the date is determined by the date format specified for the date field in the database file (DDS DATFMT and DATSEP keywords).
***EXCEL**	The date is converted to Excel format, i.e. a numeric value representing a day count. This format is ideal if you are going to load the CSV file into Excel or a similar application. Excel will recognize the data as a date and allow date functions and editing to be applied to it.
***ISO**	ISO format (YYYY-MM-DD) is used.
***EUR**	European format (DD.MM.YYYY) is used.
***USA**	USA format (MM/DD/YYYY) is used.
***JIS**	Japanese Industrial Standard (YYYY-MM-DD) is used.
***SYSVAL**	The date format defined in system value QDATFMT is used.
***JOB**	The date format defined in the job attributes is used.
***DMY**	DDMMYY format.
***DMYY**	DDMMYYYY format.
***MDYY**	MMDDYYYY format.
***YYMD**	YYYYMMDD format.
***CDMY**	CDDMMYY format. The C indicates the century (0 = 20th, 1=21st)
***CMDY**	CMMDDYY format. The C indicates the century (0 = 20th, 1=21st)
***CYMD**	CYYMMDD format. The C indicates the century (0 = 20th, 1=21st)
***JUL**	YYDDD format
***LONGJUL**	YYYYDDD format

Date Separator

The character used to separate the different part of a date value.

***JOB**	The date separator character defined in the job attributes is used.
***NONE**	No date separator character is used.
***SLASH**	A forward slash (oblique or solidus) / is used.
***HYPHEN**	A hyphen (dash) – is used
***PERIOD**	A period (full stop) . is used.
***COMMA**	A comma , is used.
***COLON**	A colon : is used.
***BLANK**	A blank is used.

separator_char Type the separator character to be used to separate the day, month and year portions of the date.

Time format

This option controls the format in which time information in the database file is presented in the stream file. Please note that this only applies to actual time fields in the database file: numeric and alphanumeric fields which contain times cannot be identified as times by ***CoolSpools Database***.

You can specify one of the following single values for this option:

***DBF** (Default). The format of the time is determined by the time format specified for the time field in the database file (DDS TIMFMT and TIMSEP keywords).

***EXCEL** The time is converted to Excel format, i.e. a numeric value representing a number of seconds. This format is ideal if you are going to load the CSV file into Excel or a similar application. Excel will recognize the data as a time and allow time functions and editing to be applied to it.

***ISO** ISO format (HH.MM.SS) is used.

***EUR** European format (HH.MM.SS) is used.

***USA** USA format (HH:MM:SS) is used.

***JIS** Japanese Industrial Standard (HH:MM:SS) is used.

***HMS** HHMMSS format.

Time Separator

The character used to separate the different part of a time value.

Options are:

***SYSVAL** The time separator defined by system value QTIMFMT is used.

***JOB** The time separator character defined in the job attributes is used.

***NONE** No time separator character is used.

***PERIOD** A period (full stop) . is used.

***COMMA** A comma , is used.

***COLON** A colon : is used.

***BLANK** A blank is used.

separator_char Type the separator character to be used to separate the day, month and year portions of the date.

Decimal point character

This element allows you to specify the characters to be used to denote a decimal point when representing fields with one or more decimal places.

Options are:

***SYSVAL** The decimal format defined by system value QDECFMT is used.

***JOB**	The decimal format defined in the job attributes is used.
***PERIOD**	A period (full stop). is used.
***COMMA**	A comma , is used.

Field delimiter

The character value (if any) output between fields. Typically, when outputting a fixed-width ASCII text file, no field delimiter is required as each field always occupies the same positions in the record, but you can output a separator value to make the file easier to read if you wish.

Options are:

***NONE**	No delimiter is output between fields.
***SPACE**	A space character (ASCII x'20')
***TAB**	A tab character (ASCII x'09')
***PIPE**	A pipe character (| ASCII x'7C')
***SEMICOLON**	A semicolon (; ASCII x'3B')
***COMMA**	A comma (, ASCII x'2C')
sep_character	Specify the character to use as the separator

Suppress leading zeros

Whether CoolSpools Database replaces leading zeros with blanks before outputting the data.

Numeric fields are always output at their full possible width, so that the values for a field in different records will line up on their decimal points, with a sign character at the beginning.

Options are:

***NO**	Leading zeros are retained
***YES**	Leading zeros are replaced by blanks.

Apply edit codes and words

Whether CoolSpools Database edits numeric values with their associated edit code or edit word before outputting the data.

For example, if a packed decimal field contains the value -123456.78 and has an associated edit code of $M, and *YES is specified for this element, the field value will be output as **$123,456.78-** rather than 123456.78.

Options are:

***NO**	Edit codes and edit words are ignored
***YES**	Numeric fields with associated edit codes or edit words (including editing defined in a Query/400 query or QM/Query form) will be edited before being output.

Positive sign

The character value used in the sign position for positive fields.

Numeric fields are always output at their full possible width, so that the values for a field in different records will line up on their decimal points, with a sign character at the beginning.

When the value is negative, a minus sign (ASCII x'2D') will appear in the sign character position. This option determines the character that appears in that position for a positive value.

Options are:

***SPACE**	A space character (ASCII x'20')
***PLUS**	A plus sign (+ ASCII x'2B')
pos_sign	Specify the character to use as the positive sign

HTML – HTML options

Parameter	**HTML**
Description	**Specifies options for HTML output**
Applies to commands:	**CVTDBFHTML** **CVTDBFSTMF (deprecated)**
Dependent on:	**TOFMT(*HTML) (CVTDBFSTMF only)**
Supports CoolSpools Database variables	**Yes (free-format text elements)**
Migration notes	**The format of the HTML parameter of the CVTDBFHTML command has been modified slightly from that of the CVTDBFSTMF command. Specifically, the CVTDBFSTMF HTML parameter allows the use of a single value *DFT which is not available on the CVTDBFHTML HTML parameter. Also, on the HTML parameter of the CVTDBFSTMF command, the date format and date separator formed a related pair for which a single value could be specified, as did the time format and time separator. This tended to make use of this parameter excessively complicated. The HTML parameter of the CVTDBFHTML command has been re-designed to improve ease of use, but code changes may be necessary when migrating from CVTDBFSTMF ...TOFMT(*HTML) to use of the CVTDBFHTML command.**

Specifies options related to HTML output.

Date format

This option controls the format in which date information in the database file is presented in the HTML file. Please note that this only applies to actual date fields in the database file: numeric and alphanumeric fields which contain date information cannot be identified as dates by ***CoolSpools Database***.

Options are:

***DBF** (Default). The format of the date is determined by the date format specified for the date field in the database file (DDS DATFMT and DATSEP keywords).

***EXCEL** The date is converted to Excel format, i.e. a numeric value representing a day count. This format is ideal if you are going to load the CSV file into Excel or a similar application. Excel will recognize the data as a date and allow date functions and editing to be applied to it.

***ISO** ISO format (YYYY-MM-DD) is used.

***EUR** European format (DD.MM.YYYY) is used.

***USA** USA format (MM/DD/YYYY) is used.

***JIS** Japanese Industrial Standard (YYYY-MM-DD) is used.

***SYSVAL** The date format defined in system value QDATFMT is used.

***JOB** The date format defined in the job attributes is used.

***DMY** DDMMYY format.

***DMYY** DDMMYYYY format.

***MDYY** MMDDYYYY format.

***YYMD** YYYYMMDD format.

***CDMY** CDDMMYY format. The C indicates the century (0 = 20th, 1=21st)

***CMDY** CMMDDYY format. The C indicates the century (0 = 20th, 1=21st)

***CYMD** CYYMMDD format. The C indicates the century (0 = 20th, 1=21st)

***JUL** YYDDD format

***LONGJUL** YYYYDDD format

Date separator

***JOB** The date separator character defined in the job attributes is used.

***NONE** No date separator character is used.

***SLASH** A forward slash (oblique or solidus) / is used.

***HYPHEN** A hyphen (dash) – is used

***PERIOD** A period (full stop) . is used.

***COMMA** A comma , is used.

***COLON** A colon : is used.

***BLANK** A blank is used.

separator_char Type the separator character to be used to separate the day, month and year portions of the date.

Time format

This option controls the format in which time information in the database file is presented in the HTML file. Please note that this only applies to actual time fields in the database file: numeric and alphanumeric fields which contain times cannot be identified as times by ***CoolSpools Database***.

You can specify one of the following single values for this option:

***DBF** (Default) The format of the time is determined by the time format specified for the time field in the database file (DDS TIMFMT and TIMSEP keywords).

***EXCEL** The time is converted to Excel format, i.e. a numeric value representing a number of seconds. This format is ideal if you are going to load the CSV file into Excel or a similar application. Excel will recognize the

data as a time and allow time functions and editing to be applied to it.

***ISO** ISO format (HH.MM.SS) is used.
***EUR** European format (HH.MM.SS) is used.
***USA** USA format (HH:MM:SS) is used.
***JIS** Japanese Industrial Standard (HH:MM:SS) is used.
***HMS** HHMMSS format.

Time separator

***SYSVAL** The time separator defined by system value QTIMFMT is used.
***JOB** The time separator character defined in the job attributes is used.
***NONE** No time separator character is used.
***PERIOD** A period (full stop) . is used.
***COMMA** A comma , is used.
***COLON** A colon : is used.
***BLANK** A blank is used.
separator_char Type the separator character to be used to separate the day, month and year portions of the date.

Decimal point character

This element allows you to specify the characters to be used to denote a decimal point when representing fields with one or more decimal places.

Options are:

***SYSVAL** The decimal format defined by system value QDECFMT is used.
***JOB** The decimal format defined in the job attributes is used.
***PERIOD** A period (full stop). is used.
***COMMA** A comma , is used.

Table cell spacing

This attribute specifies how much space (measured in pixels) the browser should leave between the left side of the table and the left-hand side of the leftmost column, the top of the table and the top side of the topmost row, and so on for the right and bottom of the table. The attribute also specifies the amount of space to leave between cells.

Apply edit codes and words

Whether CoolSpools Database edits numeric values with their associated edit code or edit word before outputting the data.

For example, if a packed decimal field contains the value -123456.78 and has an associated edit code of $M, and *YES is specified for this element, the field value will be output as **$123,456.78-** rather than 123456.78.

Options are:

***NO** Edit codes and edit words are ignored

***YES**	Numeric fields with associated edit codes or edit words (including editing defined in a Query/400 query or QM/Query form) will be edited before being output.

Convert null fields to

How CoolSpools Database outputs null fields.

Options are:

***NBSP**	The HTML table cell corresponding to a null field value will contain a single non-break space
***EMPTY**	The HTML table cell corresponding to a null field value will be empty. This can look strange with some browsers.

HTML title

Text input here will appear in the title bar of your browser when the HTML file is displayed.

***NONE**	(Default) The file will have no title.
Title_text	Up to 128 characters of title text.

Table caption

Text input here will appear as the HTML table caption. The HTML table caption is a piece of explanatory text displayed above a data table.

***NONE**	(Default) The table will have no caption.
Caption_text	Up to 128 characters of caption text.

XML – XML options

Parameter	**XML**
Description	**Specifies options for XML output**
Applies to commands:	**CVTDBFXML**
Dependent on:	
Supports CoolSpools Database variables	**Yes (free-format text elements)**

Specifies options related to XML output.

Root element

Determines the name given to the root element of the XML document that CoolSpools Database will generate.

Options are:

***FROMFILE** (Default) The name is generated from the name of the file being converted.

***QUALFILE** The name is generated from the qualified name (file name and library name) of the file being converted.

root_element Specify the name you want CoolSpools Database to use for the root element.

Row element name

Determines the name given to the row elements of the XML document that CoolSpools Database will generate. One row element is generated for each record selected from the input file.

Options are:

***RCDMT** (Default) The name is generated from the name of the record format being converted.

***QUALRCDFMT** The name is generated from the qualified record format name (file name and record format name) of the record format being converted.

row_element Specify the name you want CoolSpools Database to use for the row element.

Cols as elements or attributes

Controls the way in which the fields of a record are converted.

The value specified here controls the default action for the file. You can override that action by using the APYSTYLES parameter to specify a different action for a particular field or fields.

Options are:

***ELEMENT**	(Default) Each field will be converted as a sub-element of the row element.
***ATTRIBUTE**	Each field will be converted to an attribute of the row element.

Generate elem/attr names from

Specifies how CoolSpools Database should name elements and attributes generated from fields in the input file.

The value specified here controls the default naming for the file. You can override this naming by using the APYSTYLES parameter to specify the naming for a particular field or fields.

Options are:

***FLDNAM**	(Default) The element or attribute generated for each file will be named based on the field name.
***QUALFLD**	The element or attribute generated for each file will be named based on the qualified field name (field name, record format name, file name).
***ALIAS**	The element or attribute generated for each file will be named based on the field alias name.
***QUALALIAS**	The element or attribute generated for each file will be named based on the qualified alias name (alias name, record format name, file name).
***TEXT**	A header row is created from the field text descriptors (DDS TEXT keyword).

End of line separator

The character to use as an end-of-line marker.

Options are:

***NONE**	(Default). No end-of-line marker will be used.
***CRLF**	Lines will end with a carriage return-linefeed pair.
***LF**	Lines will end with just a linefeed.
***CR**	Lines will end with just a carriage return.

Apply edit codes and words

Whether numeric fields that have editing associated with them (edit code or edit words in DSS, or editing specified in Query/400 or a QM form), are edited before being output.

Options are:

***NO**	(Default). The editing is ignored and the field is written in its “raw” numeric form.
***YES**	Editing is applied and the field is written in edited form.

Null fields

How null fields are handled.

Options are:

***OMIT** (Default) Null fields are not output. Where a field is null for a particular row, there will be no sub-element or attribute corresponding to that field in the row element.

***NIL** Where a field is being converted as a sub-element, and that field is null for a particular row, the sub-element will be output with the attribute **xsi:nil="true"**. Where a field is being output as an attribute of the row element, the attribute will be omitted from the row element (same as *OMIT).

Convert blank char fields as

How character fields that are all blanks handled.

Options are:

***EMPTY** Blank fields are output as an empty element or attribute.

***KEEP** Blank fields are output as an element or attribute whose value is one or more space characters.

***OMIT** Blank fields are not output. Where a field contains all blanks for a particular row, there will be no sub-element or attribute corresponding to that field in the row element.

***NIL** Where a field is being converted as a sub-element, and that field contains all blanks for a particular row, the sub-element will be output with the attribute **xsi:nil="true"**. Where a field is being output as an attribute of the row element, the attribute will be omitted from the row element (same as *OMIT).

Trim blanks from char fields

Whether blanks are trimmed from character fields before the value is output.

Options are:

***BOTH** (Default) Leading and trailing blanks are trimmed.

***LEADING** Just leading blanks are trimmed.

***TRAILING** Just trailing blanks are trimmed.

***NONE** No blanks are trimmed.

XML Header

Options are :

***ENVAR** Take value from environment variable (SL_XML_HEADER). *XML assumed if envvar does not exist.

***XML** *XML Use the standard XML header (as currently).

***NONE** No XML header at all.

XML Footer

Options are :

***ENVAR**	Take value from environment variable (SL_XML_FOOTER). *NONE assumed if envvar does not exist.
***NONE**	No XML footer.

XMLNAMESPC – XML namespace options

Parameter	**XMLNAMESPC**
Description	**Specifies namespace-related options for XML output**
Applies to commands:	**CVTDBFXML**
Dependent on:	
Supports CoolSpools Database variables	**No**

Specifies options related to namespaces for XML output.

Namespace URI

If a namespace URI is specified here, CoolSpools Database will define that as the namespace for the document being created.

Options are:

***NONE**	No namespace will be specified.
namespace_URI	Specify the namespace URI to declare

Namespace prefix

Specifies the prefix which CoolSpools Database should apply to all names in the document.

Options are:

***NONE**	No prefix is applied.
namespace_prefix	Specify the prefix you want CoolSpools Database to apply to all names.

XMLSCHEMA – XML schema options

Parameter	**XMLSCHEMA**
Description	**Specifies schema-related options for XML output**
Applies to commands:	**CVTDBFXML**
Dependent on:	
Supports CoolSpools Database variables	**No**

Specifies options related to styling for XML output.

There is a one single value:

***NONE** (Default) No schema will be associated with the document.

Alternatively, specify options as listed below.

Schema type

Specifies how styling should be applied to the XML document.

Options are:

***XSD** (Default). CoolSpools Database will associate a XSD (XML Schemas Definition) schema with the document.

***DTD** CoolSpools Database will associate a DTD (Document Type Definitions) schema with the document.

Generate schema

Whether CoolSpools Database should generate a simple XSD or DTD schema itself, or whether you will specify the name of an existing schema to use.

Options are:

***YES** (Default) CoolSpools Database will generate a simple schema itself.

***NO** CoolSpools Database will use an existing schema.

Replace existing file

When CoolSpools Database is to generate a simple XSD or DTD schema itself, whether to replace any schema file that already exists.

Options are:

***STMFOPT** (Default) If the schema file already exists, it will be replaced if the value of the STMFOPT parameter is not

*NONE. If the value of the STMFOPT parameter is *NONE, it will not be replaced and an error will occur.

***NO** If the schema file already exists, it will not be replaced and an error will occur.

***YES** CoolSpools Database will replace any existing schema file.

Schema

The path name of the existing schema file that will be used, or the path name of the schema file that CoolSpools Database will generate.

Note that a relative path must be specified here, i.e. one that does not start with a /, and that the path will be interpreted as being relative to the directory path specified on the TOSTMF parameter, or to the path on the FTP server specified on the FTP parameter, if TOSTMF(*FTP) is specified.

For example, if

TOSTMF('/dir/subdir1/subdir2/filename.XML')

is specified and the path specified on this parameter element is

XMLSCHEMA(... 'subdir3/subdir4/schema.xsd')

the actual schema file used or created will be:

/dir/subdir1/subdir2/subdir3/subdir4/schema.xsd

The reference to the schema in the XML file will be to

subdir3/subdir4/schema.xsd

and this will similarly be interpreted by applications that consume the XML as being relative to the directory in which the XML document resides.

Options are:

***TOSTMF** (Default) The path name is the same as that specified on or derived from the TOSTMF parameter, with the extension changed appropriately (.XSD or .DTD).

schema_path Specify the path name of the existing schema file to use or the schema file to be generated.

XMLSTYLING – XML styling options

Parameter	**XMLSTYLING**
Description	**Specifies styling-related options for XML output**
Applies to commands:	**CVTDBFXML**
Dependent on:	
Supports CoolSpools Database variables	**No**

Specifies options related to styling for XML output.

There is a one single value:

***NONE** No styling will be applied to the document.

Alternatively, specify options as listed below.

Styling method

Specifies how styling should be applied to the XML document.

Options are:

***XSLT** (Default) CoolSpools Database will use XSLT (Extensible Stylesheet Language Transformations) to apply styling to the XML document.

***CSS** CoolSpools Database will use CSS (Cascading Stylesheets) to apply styling to the XML document.

Generate stylesheet

Whether CoolSpools Database should generate a simple XSLT or CSS stylesheet itself, or whether you will specify the name of an existing stylesheet document to use.

Options are:

***YES** (Default) CoolSpools Database will generate a simple stylesheet itself.

***NO** CoolSpools Database will use an existing stylesheet

Replace existing file

When CoolSpools Database is to generate a simple XSLT or CSS stylesheet itself, whether to replace any stylesheet that already exists.

Options are:

***STMFOPT** (Default) If the stylesheet file already exists, it will be replaced if the value of the STMFOPT parameter is not

*NONE. If the value of the STMFOPT parameter is *NONE, it will not be replaced and an error will occur.

***NO** If the stylesheet file already exists, it will not be replaced and an error will occur.

***YES** CoolSpools Database will replace any existing stylesheet.

Stylesheet

The path name of the existing stylesheet that will be used, or the path name of the stylesheet that CoolSpools Database will generate.

Note that a relative path must be specified here, i.e. one that does not start with a /, and that the path will be interpreted as being relative to the directory path specified on the TOSTMF parameter, or to the path on the FTP server specified on the FTP parameter, if TOSTMF(*FTP) is specified.

For example, if

TOSTMF('/dir/subdir1/subdir2/filename.XML')

is specified and the path specified on this parameter element is

XMLSTYLING(... 'subdir3/subdir4/stylesheet.xslt')

the actual stylesheet file used or created will be:

/dir/subdir1/subdir2/subdir3/subdir4/stylesheet.xslt

The reference to the stylesheet in the XML file will be to

subdir3/subdir4/stylesheet.xslt

and this will similarly be interpreted by applications that consume the XML as being relative to the directory in which the XML document resides.

Options are:

***TOSTMF** (Default) The path name is the same as that specified on or derived from the TOSTMF parameter, with the extension changed appropriately (.XSLT or .CSS).

stylesheet_path Specify the path name of the existing stylesheet to use or the stylesheet to be generated.

CSSSTYLING – CSS stylesheet options

Parameter	**CSSSTYLING**
Description	**Specifies styling-related options for CSS styling**
Applies to commands:	**CVTDBFXML**
Dependent on:	**XMLSTYLING(*CSS)**
Supports CoolSpools Database variables	**No**

Specifies options related to CSS styling for XML output.

There is a one single value:

***NONE** No CSS styling options are defined.

Alternatively, specify options as listed below.

Generate :before selector

Specifies if and how a :before selector is generated.

Note that not all browsers currently support :before selectors (notably MS Internet Explorer does not).

Options are:

***AVAIL** (Default). The :before selector is generated from the best available source, either the field text, alias name, column headings or field name.

***TEXT** The :before selector is generated from the field text attribute (DDS TEXT).

***ALIAS** The :before selector is generated from the field alias name (DDS ALIAS).

***COLHDG** The :before selector is generated from the field column headings (DDS COLHDG).

***FLDNAM** The :before selector is generated from the field name.

Padding width

Specifies the number of characters to which the :before text is padded. Note that this will only cause columns to line up if a monospace font (e.g. Courier New) is selected.

Options are:

50 (Default). 50 characters.

padding_width Specify the width to which the :before text is padded.

Padding type

Specifies the how the :before text is padded.

Options are:

***DOTS**	(Default) The :before text is padded with dot leaders (. . .)
***SPACES**	The :before text is padded with spaces.
***NONE**	:before text is not padded to a fixed width

:before selector style

Free-form unvalidated text specifying CSS styling options to be applied to :before text.

Options are:

***NONE**	(Default) No styling options are applied.
CSS_styling	Specify a valid CSS style declaration for the :before text.

HEADER – Header row

Parameter	**HEADER**
Description	**Specifies options for generating a header row**
Applies to commands:	**CVTDBFXLSX , CVTDBFXL, CVTDBFCSV, CVTDBFHTML** **CVTDBFSTMF (deprecated)**
Dependent on:	**TOFMT(*XLS|*CSV|*HTML) (CVTDBFSTMF only)**
Supports CoolSpools Database variables	**Yes (free-format text elements)**
Migration notes	**CVTDBFXL includes some options on this parameter which are not available on CVTDBFCSV or CVTDBFHTML.**

Allows you to specify whether a head row should be created in the stream file and, if so, how.

Please note that a header row cannot be created for fixed-length ASCII files created with CVTDBFTXT or CVTDBFSTMF TOFMT(*FIXED) as the fixed-length nature of the columns in these types of file make the creation of header text impossible.

However, for CSV, Excel and HTML files it is common practice to include a single row at the beginning of the data which provides column labels for the data.

Source of header information

What attributes of the input file are used to generate the headings.

Options are:

AVAIL** (Default) ***CoolSpools Database will select the best available labels from the input file to create the header row. This selection of label text is performed according to the following criteria. If the fields in the input file have Column Headings (DDS COLHDG keyword), they will be used to generate the column headings. Otherwise, if the fields in the input file have field aliases (DDS ALIAS keyword), those will be used instead. Otherwise, if the fields in the input file have text descriptors (DDS TEXT keyword), those will be used. Otherwise the field names will be used. For fixed-length ASCII text files, no header row will be created.

***NONE** No header row is created.

***COLHDG** For CSV, Excel and HTML files, a header row is created from the field column headings (DDS COLHDG keyword).

This option is invalid if TOFMT(*FIXED) is specified.

***ALIAS** For CSV, Excel and HTML files, a header row is created from the field aliases (DDS ALIAS keyword).
This option is invalid if TOFMT(*FIXED) is specified.

***TEXT** For CSV, Excel and HTML files, a header row is created from the field text descriptors (DDS TEXT keyword).
This option is invalid if TOFMT(*FIXED) is specified.

***FLDNAM** For CSV, Excel and HTML files, a header row is created from the field names. This option is invalid if TOFMT(*FIXED) is specified.

***COLHDG1** The first column heading element only.

***COLHDG2** The second column heading element only.

***COLHDG3** The third column heading element only.

***COLHDG12** Column heading elements 1 and 2 only.

***COLHDG13** Column heading elements 1 and 3 only.

***COLHDG23** Column heading elements 2 and 3 only.

Number of lines to freeze

Available on **CVTDBFXLSX** , CVTDBFXL and CVTDBFSTMF only.

Determines whether any rows at the top of the screen are frozen or not (i.e. whether those rows scroll with the rest of the data or remain fixed when remaining rows scroll).

Options are:

***HEADER** (Default). Any header row generated based on the setting of the preceding element is frozen along with any additional header rows (see below).

***NONE** No rows are frozen.

***YES** Identical to *HEADER and made available for reasons of backwards compatibility.

***NO** Identical to *NONE and made available for reasons of backwards compatibility.

rows_to_freeze Specify the number of rows to freeze, starting with and including the first row.

Line breaks in Excel headers?

Available on **CVTDBFXLSX** , CVTDBFXL and CVTDBFSTMF only.

Whether, when the header line is being constructed from DDS column heading values, line breaks are inserted between the different parts of the column headings so that the column headings wrap to multiple lines.

Options are:

***YES** (Default) Line breaks are inserted between the different parts of the DDS column headings so that they appear one per line at the top of the Excel worksheet.

***NO** A space is inserted between the different parts of the DDS column headings and Excel will control the wrapping of header text.

Additional header line 1

Additional header line 2

Additional header line 3

Three lines of free-format text which will appear at the top of the page above the column headings.

Headings on overflow sheets

Available on **CVTDBFXLSX** , CVTDBFXL and CVTDBFSTMF only.

Whether column headings are repeated at the top of any overflow worksheets that are necessary because the number of rows of data exceeds the maximum number of rows in an Excel worksheet.

Options are:

***YES**	(Default) Column headings appear at the top of each worksheet that is created.
***NO**	Column headings appear only at the top of the first worksheet that is created.

DFNSTYLES – Define styles

Parameter	**DFNSTYLES**
Description	**Defines styles which control the appearance of data on screen**
Applies to commands:	**CVTDBFXL, CVTDBFHTML, CVTDBFXML**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

Allows you to define the attributes of the default styles for different types of information or to created user-defined named styles. Styles control the appearance of data on screen. For example, they govern items such as:

- font attributes
- color
- numeric formatting

There are two ways in which to associate a style with a piece of data in your output file (e.g. a cell in an Excel spreadsheet or an element of an XML document):

1. **Implicitly**

   By defining the style name the same as the name of a row group in your Database-to-Excel map or an element in your Database-to-XML map, you implicitly apply that style to the data in question. For example, a style called REPORT_HEADING will be implicitly and automatically applied to an Excel row group called REPORT_HEADING.

   Note that style names are case-sensitive because XML element names need to be case-sensitive and for this association of names to work, the names must match exactly in terms of case.

2. **Explicitly**

   Alternatively, use the APYSTYLES parameter to define the styles you wish to apply to different parts of the file you create.

The precise set of attributes that can be controlled varies depending on the format of the data being created as some attributes are not relevant to certain output formats.

CoolSpools Database styles defined will translate into Excel user-defined styles if converting to Excel format and CSS styles if converting to HTML/XML.

Style name

Each style is identified by means of a style name, which can be up to 50 characters in length and is case-sensitive (in order to allow matching to XML elements, the names of which conform to the rules for XML names).

You can define your own named styles by choosing a name that is helpful to you, but there are also 7 pre-defined style names which have special meanings:

***DATA** The default style name for data rows.

If you specify *DATA for the name of the style, the attributes you specify will become the default attributes for data rows.

***HEADER** The default style name for header rows.

If you specify *HEADER for the name of the style, the attributes you specify will become the default attributes for header rows (rows generated as a result of the HEADER parameter settings).

***TITLE** The default style name for title rows.

If you specify *TITLE for the name of the style, the attributes you specify will become the default attributes for title rows. Title rows are those generated from the additional heading lines elements of the HEADER parameter and the caption text of the HTML parameter.

***SUBTOTAL** The default style name for subtotal rows.

If you specify *SUBTOTAL for the name of the style, the attributes you specify will become the default attributes for subtotal rows. Subtotal rows are those that result from subtotals and group-by fields in Query/400 queries when the *COMBINED output form is selected.

***TOTAL** The default style name for total rows.

If you specify *TOTAL for the name of the style, the attributes you specify will become the default attributes for total rows. Total rows are those that result from subtotals and group-by fields in Query/400 queries when the *COMBINED output form is selected.

***HYPERLINK** If you specify *HYPERLINK for the name of the style, the field will become a clickable hyperlink.

***FLWDLINK** If you specify *FLWDLINK for the name of the style, the field will become a clickable hyperlink, with the appearance of a link already visited.

***ROOT** **XML Only.** The default style for the root element.

***ROW** **XML Only.** The default style for the row element, i.e. the element corresponding to a record in the input file.

If these styles are not defined, the default attributes assigned are as shown in the table below:

Attribute	*DATA	*HEADER	*TITLE	*SUBTOTAL	*TOTAL	*ROOT	*ROW
Locked (Excel only)	Yes	Yes	Yes	Yes	Yes	N/A	N/A
Hidden (Excel only)	No	No	No	No	No	N/A	N/A
Horizontal alignment	General	General	General	General	General	General	General
Indent	0	0	0	0	0	N/A	N/A
Vertical alignment	Top	Bottom	Top	Top	Top	Top	Top
Wrap text	No	Yes	No	No	No	No	No
Shrink to fit (Excel only)	No	Yes	No	No	No	N/A	N/A
Vertical alignment	Top	Top	Top	Top	Top	Top	Top
Row height	*AUTOFIT	*AUTOFIT	*AUTOFIT	*AUTOFIT	*AUTOFIT	*AUTOFIT	*AUTOFIT
Font name (Excel)	Arial	Arial	Arial	Arial	Arial	N/A	N/A
Font name (HTML & XML)	sans-serif	sans-serif	sans-serif	sans-serif	sans-serif	sans-serif	sans-serif
Font size in point (Excel)	10	10	10	10	10	N/A	N/A
Font size in point (HTML & XML)	12	12	12	12	12	12	12
Bold	No	Yes	Yes	Yes	Yes	Yes	Yes
Italic	No	No	No	No	No	No	No
Underlined	No	No	No	No	No	No	No
Text color	Black	Black	Black	Black	Black	Black	Black
Background color	White	White	White	White	White	White	White
Pattern color (Excel only)	*AUTO	*AUTO	*AUTO	*AUTO	*AUTO	*AUTO	*AUTO
Pattern style (Excel only)	*NONE	*NONE	*NONE	*NONE	*NONE	*NONE	*NONE
Border styles (Excel)	*NONE	*NONE	*NONE	*NONE	*NONE	N/A	N/A

Border styles (HTML)	*INSET	*INSET	*INSET	*INSET	*INSET	N/A	N/A
Border widths (HTML only)	1	1	1	1	1	1	1
Border colors	*AUTO	*AUTO	*AUTO	*AUTO	*AUTO	*AUTO	*AUTO
Number format type (Excel only)	*DFT	*DFT	*DFT	*DFT	*DFT	N/A	N/A
Decimal places (Excel only)	*FIELD	*FIELD	*FIELD	*FIELD	*FIELD	N/A	N/A
Thousands separator (Excel only)	*FMT	*FMT	*FMT	*FMT	*FMT	N/A	N/A
Currency symbol (Excel only)	*FMT	*FMT	*FMT	*FMT	*FMT	N/A	N/A
Negative numbers (Excel only)	*FMT	*FMT	*FMT	*FMT	*FMT	N/A	N/A
Custom number format (Excel only)	*NONE	*NONE	*NONE	*NONE	*NONE	N/A	N/A
Cell padding (HTML only)	1	1	1	1	1	1	1
Additional style declaration (HTML only)	*NONE	*NONE	*NONE	*NONE	*NONE	*NONE	*NONE
Display option (XML only)	*BLOCK	*BLOCK	*BLOCK	*BLOCK	*BLOCK	*BLOCK	*BLOCK

Example:
CVTDBFXL

...
DFNSTYLES((HIGHLIGHT *YES *NO *GENERAL *NONE *BOTTOM *NO *NO *AUTOFIT
ARIAL 12 *YES *NO *NO *YELLOW *BLUE *AUTO *NONE *THIN))
APYSTYLES((BALANCE *FLDNAM HIGHLIGHT))

This code defines a new style called “highlight” that uses Arial bold 12-point yellow on blue and applies that style to the field called “BALANCE”.

Example:
CVTDBFXL

...
DFNSTYLES((*DATA *YES *NO *GENERAL *NONE *BOTTOM *NO *NO *AUTOFIT
COURIER 12 *NO *NO *NO *RED *SILVER *AUTO *NONE *THIN) (*HEADER *YES
NO *GENERAL *NONE *BOTTOM *YES *NO *AUTOFIT *COURIER 14 *YES *NO *NO
***RED *SILVER *AUTO *NONE *THIN))**

This code redefines the predefined *DATA and *HEADER styles and so modifies the default attributes for data rows to use Courier 12-point red on silver and the default attributes for column headings to use Courier 14-point bold red on silver.

Locked

(Excel only)

Whether cells to which this style is applied are locked when worksheet protection is in effect.

Options are:

***YES**	(Default) When the worksheet is protected, the cell will be locked (protected).
***NO**	When the worksheet is protected, the cell will remain unlocked.

Example:
CVTDBFXL

...
DFNSTYLES((*DATA *NO) (*HEADER *YES *NO *GENERAL *NO *BOTTOM *ARIAL 10 *NO *NO *NO *WHITE *BLUE) (LOCKED *YES *NO *GENERAL *NO * BOTTOM *ARIAL 10 *NO *NO *NO *WHITE *BLUE)) APYSTYLES((CUSTNO *FLDNAM LOCKED)) XLSPROTECT(*YES)

This code redefines the *DATA style such that, by default, data cells are not locked when worksheet protection is in effect. It then redefines the *HEADER style so that headers are locked and appear white on blue. Finally, it defines a new style called “LOCKED” such that cells to which this style is applied are locked and also appear white on blue. The style LOCKED is applied to the single column “CUSTNO” and worksheet protection is switched on.

The overall effect is to create a worksheet where the user can change the data apart from the headings and the customer number column, which appear white on blue rather than black on white to emphasize the fact they are different.

Hidden

(Excel only)

Allows you to indicate that a column should be hidden. This might be useful if you do not wish the column to appear but want it to be available for calculations.

Options are:

***NO**	(Default) The column is not hidden.
***YES**	The columns will be hidden

Horizontal alignment

Controls the horizontal alignment of data in a cell.

Options are:

***GENERAL**	(Default) Character data is left-aligned. Numeric data and dates are right-aligned. In relation to header text, the alignment is dictated by the nature of the data in the column, not the header text, i.e. headings for columns of

character data will align to the left and headings for numeric columns and date columns will align to the right.

***LEFT** Left-aligned.

***RIGHT** Right-aligned.

***CENTER** Center-aligned

***FILL** (Excel only) Fill. Repeats the data in the cell across the entire width of the column.

***JUSTIFY** Forces data to fill the entire width of the column, wrapping text to additional lines, if necessary.

***DISTRIBUTED** (Excel only) Distributed. Available only in Excel 2002 and above. It results in the cell contents being distributed across the width of the cell, to line up with both the left and right side.

Indent

Sets the text indent level. The effects of this are somewhat different between Excel and HTML/XML.

Options are:

***NONE** (Default) No indent is applied.

0-15 (Excel) Sets the indentation level. Each indentation level is equivalent to 3 spaces. All text affected is indented to the same extent, i.e. where text wraps to more than one line, it is all indented to the same point.

0-99 (HTML/XML). Sets the text-indent property in ems (the width of an em is equivalent to the point size of the font). The first line of the text only is indented.

Vertical alignment

Controls the vertical alignment of data in a cell.

Options are:

***BOTTOM** (Default). Information is aligned at the top of the cell.

***TOP** Information is aligned at the bottom of the cell.

***CENTER** information is aligned in the center of the cell.

***JUSTIFY** Text is spread evenly vertically across the height of the cell.

***DISTRIBUTED** (Excel only) Text is spread evenly between the top of the cell and the bottom. Effectively, blank space is placed between each line so that the complete cell is filled.

Wrap text

Controls whether text wraps in cells.

Options are:

***NO** (Default). Text does not wrap in the cell. If the text does not fit in the column width, it is truncated.

***YES** Text wraps in the cell. If the text foes not fit in the column width, it will flow on to multiple lines.

Shrink to fit

(Excel only)

Determines whether the cell contents are shrunk to fit the available column width by reducing the font size.

Options are:

***NO** (Default) Text is not shrunk to fit.
***YES** Text is fitted to the available column width by reducing the font size, as required.

Row height

Sets the height of rows.

Note that this attribute is only effective if set on one of the predefined styles:

- *DATA (controlling the height of data rows)
- *HEADER (controlling the height of the column headings row)
- *TITLE (controlling the height of title rows)
- *SUBTOTAL (controlling the height of subtotal rows)
- *TOTAL (Controlling the height of total rows)

Even if you associated every column with a style other than these, the row height set for that row will not be effective as the row height is always set from the appropriate predefined style from the list above.

Options are:

***AUTOFIT** (Default). The height of rows is automatically set by Excel or your browser (HTML/XML) based on the font size.
0-409 (Excel) Specify the row height in points (72 points = 1 inch)
0-32767 (HTML/XML) Specify the row height in points (72 points = 1 inch)

Font name

Specifies the name of the font to be used.

Note that CoolSpools Database cannot validate whether the font name you have specified is valid or whether it will be available when the file is opened. If the font name is typed incorrectly or if the font is not available when the file is opened, Excel or your browser will substitute a different font.

Note also that when the font you use in Excel is not one of the “well known” fonts (Arial, Courier New or Times New Roman), CoolSpools Database may not be able to calculate column widths correctly because it has no access to the font metrics on which those calculations depend.

Excel options are:

***ARIAL**	(Default) Arial
***COURIER**	Courier New
***TIMES**	Times New Roman
font_name	Specify the name of the font to use

HTML/XML options are:

***SANS**	(Default) Sans-serif font family.
***SERIF**	Serif font family
***MONO**	Monospaced font family.
***ARIAL**	(Default). Arial
***COURIER**	Courier New
***TIMES**	Times New Roman
font_name	Specify the name of the font to use

Font size in points

The point size of the font to use. The default is 10 for Excel and 12 for HTML/XML.

Bold

Whether the font is bold or not. Note that setting this attribute will only result in a bold font if a suitable bold version of the font is available or if the normal font can be adapted.

Options are:

***NO**	(Default). Normal font
***YES**	Bold font.

Italic

Whether the font is italic or not. Note that setting this attribute will only result in an italic font if a suitable italic version of the font is available or if the normal font can be adapted.

Options are:

***NO**	(Default). Normal font
***YES**	Italic font.

Underlined

Whether the font is underlined or not and, if it is, the style of underlining.

Excel options are:

***NO**	(Default). No underlining
***SINGLE**	Single underlining
***DOUBLE**	Double underlining
***SGLACC**	Single accounting underlining
***DBLACC**	Double accounting underlining

HTML/XML options are:

***NO**	(Default). No underlining
***YES**	Single underlining

Text color

Determines the color of text in cells.

The Excel default is:

***AUTO**	The Excel default text color (usually black)

Alternatively, you can use one of the built-in Excel colors listed below with their RGB coding.

*BLACK #000000	*WHITE #FFFFFF	*RED #FF0000	*BRIGHTGREEN #00FF00
*BLUE #0000FF	*YELLOW #FFFF00	*PINK #FF00FF	*TURQUOISE #00FFFF
*DARKRED #800000	*GREEN #008000	*DARKBLUE #000080	*DARKYELLOW #808000
*VIOLET #800080	*TEAL #800080	*GRAY25 #800080	*GRAY50 #800080
*MAUVE #9999FF	*PLUM #993366	*YELLOWWHITE #FFFFCC	*LIGHTTURQUOISE #CCFFFF
*DARKPINK #660066	*BLUSH #FF8080	*MEDIUMBLUE #0066CC	*PALEMAUVE #CCCCFF
*SKYBLUE #00CCFF	*LIGHTGREEN #CCFFCC	*LIGHTYELLOW #FFFF99	*PALEBLUE #99CCFF
*ROSE #FF99CC	*LAVENDER #CC99FF	*TAN #FFCC99	*LIGHTBLUE #3366FF
*AQUA #33CCCC	*LIME #99CC00	*GOLD #FFCC00	*LIGHTORANGE #FF9900
*ORANGE #FF6600	*BLUEGRAY #666699	*GRAY40 #969696	*DARKTEAL #003366
*SEAGREEN #339966	*DARKGREEN #003300	*OLIVEGREEN #333300	*BROWN #993300
*INDIGO #333399	*GRAY80 #333333		

When converting to *XLSX format, you can also optionally specify your own RGB color code in the form of six hexadecimal digits (similar to the codes shown in the table above). Please note that this option is not supported when converting to BIFF8 format.

The HTML/XML default is:

***BLACK**	Black

Alternatively, you can use one of the HTML colors listed below with their RGB coding.

*ALICEBLUE #F0F8FF	*ANTIQUEWHITE #FAEBD7	*AQUA #FF0000	*AQUAMARINE #00FF00
*AZURE #0000FF	*BEIGE #F5F5DC	*BISQUE #FFE4C4	*BLACK #000000
*BLANCHEDALMOND #FFEBCD	*BLUE #0000FF	*BLUEVIOLET #8A2BE2	*BROWN #A52A2A
*BURLYWOOD #DEB887	*CADETBLUE #5F9EA0	*CHARTREUSE #7FFF00	*CHOCOLATE #D2691E
*CORAL #FF7F50	*CORNFLOWERBLUE #6495ED	*CORNSILK #FFF8DC	*CRIMSON #DC143C
*CYAN #00FFFF	*DARKBLUE #00008B	*DARKCYAN #008B8B	*DARKGOLDENROD #B8860B
*DARKGRAY #A9A9A9	*DARKGREY #A9A9A9	*DARKGREEN #006400	*DARKKHAKI #BDB76B
*DARKMAGENTA #8B008B	*DARKOLIVEGREEN #556B2F	*DARKORANGE #FF8C00	*DARKORCHID #9932CC
*DARKRED #8B0000	*DARKSALMON #E9967A	*DARKSEAGREEN #8FBC8F	*DARKSLATEBLUE #483D8B
*DARKSLATEGRAY #2F4F4F	*DARKSLATEGREY #2F4F4F	*DARKTURQUOISE #00CED1	*DARKVIOLET #9400D3
*DEEPPINK #FF1493	*DEEPSKYBLUE #00BFFF	*DIMGRAY #696969	*DIMGREY #696969
*DODGERBLUE #1E90FF	*FELDSPAR #D19275	*FIREBRICK #B22222	*FLORALWHITE #FFFAF0
*FORESTGREEN #228B22	*FUCHSIA #FF00FF	*GAINSBORO #DCDCDC	*GHOSTWHITE #F8F8FF
*GOLD #FFD700	*GOLDENROD #DAA520	*GRAY #808080	*GREY #808080
*GREEN #008000	*GREENYELLOW #ADFF2F	*HONEYDEW #F0FFF0	*HOTPINK #FF69B4
*INDIANRED #CD5C5C	*INDIGO #4B0082	*IVORY #FFFFF0	*KHAKI #F0E68C
*LAVENDER #E6E6FA	*LAVENDERBLUSH #FFF0F5	*LAWNGREEN #7CFC00	*LEMONCHIFFON #FFFACD
*LIGHTBLUE #ADD8E6	*LIGHTCORAL #F08080	*LIGHTCYAN #E0FFFF	*LIGHTGOLDENROD #FAFAD2
*LIGHTGRAY #D3D3D3	*LIGHTGREY #D3D3D3	*LIGHTGREEN #90EE90	*LIGHTPINK #FFB6C1
*LIGHTSALMON #FFA07A	*LIGHTSEAGREEN #20B2AA	*LIGHTSKYBLUE #87CEFA	*LIGHTSLATEBLUE #8470FF
*LIGHTSLATEGRAY #778899	*LIGHTSLATEGREY #778899	*LIGHTSTEELBLUE #B0C4DE	*LIGHTYELLOW #FFFFE0
*LIME	*LIMEGREEN	*LINEN	*MAGENTA

#00FF00	#32CD32	#FAF0E6	#FF00FF
*MAROON #800000	*MEDIUMAQUAMARINE #66CDAA	*MEDIUMBLUE #0000CD	*MEDIUMORCHID #BA55D3
*MEDIUMPURPLE #9370D8	*MEDIUMSEAGREEN #3CB371	*MEDIUMSLATEBLUE #7B68EE	*MEDIUMSPRINGGREEN #00FA9A
*MEDIUMTURQUOISE #48D1CC	*MEDIUMVIOLETRED #C71585	*MIDNIGHTBLUE #191970	*MINTCREAM #F5FFFA
*MISTYROSE #FFE4E1	*MOCCASIN #FFE4B5	*NAVAJOWHITE #FFDEAD	*NAVY #000080
*OLDLACE #FDF5E6	*OLIVE #808000	*OLIVEDRAB #6B8E23	*ORANGE #FFA500
*ORANGERED #FF4500	*ORCHID #DA70D6	*PALEBLUE #ADD8E6	*PALEBROWN #CD853F
*PALECYAN #E0FFFF	*PALEGOLDENROD #EEE8AA	*PALEGRAY #D3D3D3	*PALEGREY #D3D3D3
*PALEGREEN #98FB98	*PALEMAG Pale Magenta #DDA0DD	*PALETURQUOISE #AFEEEE	*PALEVIOLETRED #D87093
*PALEYLW Pale Yellow #FFFFE0	*PAPAYAWHIP #FFEFD5	*PEACHPUFF #FFDAB9	*PERU #CD853F
*PINK #FFC0CB	*PLUM #DDA0DD	*POWDERBLUE #B0E0E6	*PURPLE #800080
*RED #FF0000	*ROSYBROWN #BC8F8F	*ROYALBLUE #041690	*SADDLEBROWN #8B4513
*SALMON #FA8072	*SANDYBROWN #F4A460	*SEAGREEN #2E8B57	*SEASHELL #FFF5EE
*SIENNA #A0522D	*SILVER #C0C0C0	*SKYBLUE #87CEEB	*SLATEBLUE #6A5ACD
*SLATEGRAY #708090	*SLATEGREY #708090	*SNOW #FFFAFA	*SPRINGGREEN #00FF7F
*TOMATO #FF6347	*TURQUOISE #40E0D0	*VIOLET #EE82EE	*VIOLETRED #D02090
*WHEAT #F5DEB3	*WHITE #FFFFFF	*WHITESMOKE #F5F5F5	*YELLOW #FFFF00
*YELLOWGREEN #9ACD32			

You can also optionally specify your own RGB color code in the form of six hexadecimal digits (similar to the codes shown in the table above).

Background color

Determines the color of the background of a cell.

The Excel default is:

***AUTO** The Excel default background color (usually white)

Alternatively, you can use the same Excel options as listed for text color above.

The HTML/XML default is:

***WHITE** White

Alternatively, you can use the HTML color options as listed for text color above.

Pattern color

(Excel only)

Determines the color of the any pattern applied to a cell.

The Excel default is:

***AUTO** The Excel default pattern color (usually black)

Alternatively, you can use the same Excel options as listed for text color above.

Pattern style

(Excel only)

Determines the style of the any pattern applied to a cell.

The default is:

***NONE** No pattern

The available pattern options are the following names, which correspond to Excel's builtin patterns:

***SOLID**
***GRAY75**
***GRAY50**
***GRAY25**
***GRAY12.5**
***GRAY6.25**
***HRZSTRIPE**
***VRTSTRIPE**
***REVERSEDIAGSTRIPE**
***DIAGSTRIPE**
***DIAGCROSSHATCH**
***THICKDIAGCROSSHATCH**
***THINHRZSTRIPE**
***THINVRTSTRIPE**

***THINREVERSEDIAGSTRIPE**
***THINDIAGSTRIPE**
***THINHRZCROSSHATCH**
***THINDIAGCROSSHATCH**

Border style (Excel)

Determines the style of the border around a cell.

Note that the DFNSTYLES parameter does not support the setting of different attributes for the top, bottom, left and right borders. The attributes defined here apply to all 4 borders. If you want to define different attributes for the different borders, you must use a user-defined named style (see Base Option manual, CRTSTLDFN Create Style Definition command)

The Excel default is:

***NONE**	No border

Other Excel options are the following list of names corresponding to Excel's builtin border styles:

***THIN**
***MEDIUM**
***DASHED**
***DOTTED**
***THICK**
***DOUBLE**
***HAIR**

The HTML/XML options correspond to the CSS border style options:

***INSET**	(Default) CSS inset border style
***DASHED**	
***DOTTED**	
***DOUBLE**	
***GROOVE**	
***HIDDEN**	
***OUTSET**	
***RIDGE**	
***SOLID**	

Border width

(HTML only)

The width of the cell border in pixels.

Note that the DFNSTYLES parameter does not support the setting of different attributes for the top, bottom, left and right borders. The attributes defined here apply to all 4 borders. If you want to define different attributes for the different borders, you must use a user-defined named style (see Base Option manual, CRTSTLDFN Create Style Definition command)

Border color

The color of the border. Options are the same as for text color above.

Note that the DFNSTYLES parameter does not support the setting of different attributes for the top, bottom, left and right borders. The attributes defined here apply to all 4 borders. If you want to define different attributes for the different borders, you must use a user-defined named style (see Base Option manual, CRTSTLDFN Create Style Definition command)

Number format type

(Excel only)

Sets the category of number formatting applied to numbers in cells to which this style relates. The following options allow you to modify or override aspects of the default formatting determined by your choice for this parameter element.

Options are:

***DFT**	CoolSpools Database will decide the formatting to apply based on the editing associated with the field in question, i.e. any edit code or edit word defined in the field's DDS or any editing defined in a Query/400 query or QM form for the field.
***GENERAL**	Ignore any editing associated with the field and format numeric data with general numbers in them.
***FIXED**	Ignore any editing associated with the field and format numeric data with a fixed number of decimal places.
***CURRENCY**	Ignore any editing associated with the field and format numeric data as a currency amount.
***ACCOUNTING**	Ignore any editing associated with the field and format numeric data as an accounting value. The Accounting category is the same as the Currency category, except it will align currency symbols and decimal points.
***DATE**	Ignore any formatting associated with the field and format it as a date. If the field does not contain a valid date, it will be formatted according to any editing associated with the field.
***TIME**	Ignore any formatting associated with the field and format it as a time or date/time. If the field does not contain a valid time or timestamp, it will be formatted according to any editing associated with the field.
***PERCENT**	Ignore any editing associated with the field, multiply the value by 100 and format numeric data as a percentage.

***SCIENTIFIC**	Ignore any editing associated with the field, and format numeric data in scientific notation.
***TEXT**	Ignore any editing associated with the field, and format numeric data as text.
***CUSTOM**	Apply a custom number format specified on the custom number format element below.

Decimal places

(Excel only)

Where a numeric format (other than *DFT) that can include decimal places was specified on the number format type parameter, this parameter element determines the number of decimal places displayed.

Options are:

***FIELD**	The number of decimal places defined for the field in its DDS.
dec_places	Specify the number of decimal places

Thousands separator

(Excel only)

Where a numeric format (other than *DFT) that can include thousands separators was specified on the number format type parameter, this parameter element determines whether thousands separators actually appear.

Options are:

***FMT**	Whether thousands separators appear depends on the number format type selected. Accounting and currency formatting will include thousands separators but other types will not.
***YES**	Include thousands separators in the number format irrespective of the fact that the number format type specified does not normally include them. For example you can format percentage values with thousands separators using this option.
***NO**	Do not **i**nclude thousands separators in the number format irrespective of the fact that the number format type specified does normally include them. For example, you can format currency values without thousands separators using this option.

Currency symbol

(Excel only)

Where a numeric format (other than *DFT) that can include a currency symbol was specified on the number format type parameter, this parameter element determines whether a currency symbol actually appears and what that symbol should be.

Options are:

***FMT**	Whether a currency symbol appears depends on the number format type selected. Accounting and currency formatting will include a currency symbol but other types will not. The currency symbol will be derived from the system value QCURSYM.
***SYSVAL**	Include a currency symbol in the number format irrespective of the fact that the number format type specified does not normally include one. The currency symbol will be derived from the system value QCURSYM.
***NO**	Do not **i**nclude a currency symbol in the number format irrespective of the fact that the number format type specified does normally include one. You can use this option to display a currency value with no currency symbol.
currency_symbol	Include a currency symbol in all numbers. The currency symbol will be the one specified here.

Negative numbers

(Excel only)

Overrides the way in which negative numbers are displayed.

Options are:

***FMT**	The format of negative numbers is determined by the option specified for the number format type.
***LEADING**	A leading minus sign is displayed.
***TRAILING**	A trailing minus sign is displayed.
***PARENTHESES**	Negative numbers appear in parentheses.
***RED**	Negative numbers appear in red.
***REDL**	Negative numbers appear in red with a leading minus sign.
***REDT**	Negative numbers appear in red with a trailing minus sign.
***REDP**	Negative numbers appear in red and in parentheses.

Custom number format

(Excel only)

Specify a custom number format. *CUSTOM must be specified for the number format type element above.

Options are:

***NONE**	No custom number format is defined.
number_format	Specify the custom Excel number format to use.

Cell padding

(HTML only)

The padding to apply to the cell, in pixels.

Additional style declaration

(HTML only)

A free-format, unvalidated string of text which will be appended to the style declaration generated by the previous elements. This option enables you to specify additional CSS formatting not available from this parameter. However, you must ensure that the text you enter is a valid portion of a CSS style declaration.

For example, specifying '**font-variant: small-caps**' would cause the text to appear in small capitals.

Display option (XML only)

Sets the CSS display style.

Options are:

***BLOCK**	(Default). Takes up the full width available, with a new line before and after.
***INLINE**	Takes up only as much width as it needs, and does not force new lines

CNDFMTGRP – Conditional formatting groups

Parameter	**CNDFMTGRP**
Description	**Defines groups of conditional formatting rules and the range of cells to which they apply**
Applies to commands:	**CVTDBFXL**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

Specifies conditional formatting rule groups. A conditional formatting rule group defines a group of related rules which will be applied, in a given priority sequence, to a range of cells in order to determine the appearance of those cells. For example, you might define a rule that tests the value of a customer account balance field and make rows where the balance is negative red and those where it is above a certain level green etc.

See the CNDFMTRULE parameter below for examples of how to define conditional formatting.

Rule group number

Specifies an arbitrary, non-zero, positive integer which identifies the rule group. You can choose any number you like to identify the group, but it must be unique for all rule groups defined on the command.

The rule group number is used to match rules defined on the CNDFMTRULE parameter against rule groups defined on the CNDFMTGRP parameter. The CNDFMTGRP parameter defines group-level attributes such as the range of cells to which the rules should be applied, whereas the CNDFMTRULE parameter defines the individual rules in the group that will be tested, one after another, in the priority sequence you specify, against those cells.

Rule group name

Specifies an optional, arbitrary name which identifies the rule group. You can choose any name you like to identify the group. The name has no function other than to help you document and remember the purpose of a given rule group.

Apply to rows

Which rows in the worksheet the rules should be applied to.

Options are:

***ALL** (Default). The rules are applied to all rows in the worksheet, including those not populated by data. If new data is entered after the last row of data, the rules will apply to those new rows too.

***USED** The rules apply only to the rows populated with data.

Identify columns by

How the columns the rules should apply to are identified on the following element:

Options are:

***FLDNAM** (Default). The following parameter element will contain one or more field names from the input file that identify the columns in the worksheet to which the rules apply. The rules will apply to the columns corresponding to those fields.

***FLDNBR** The following parameter element will contain one or more field numbers from the input file that identify the columns in the worksheet to which the rules apply. The rules will apply to the columns corresponding to those fields. The field numbers refer to the relative position of the fields in the database file record (1=first field, 2 = second field etc.)

***XLCOLID** The following parameter element will contain one or more column identifiers (A-Z, AA-ZZ etc.) identifying the columns in the worksheet to which the rules apply.

***MAPREF** (CVTDBFXL only) A database map is being used to control the structure of the Excel output file being created and the styling is being applied to an area of the Excel file. The next element contains a map reference that identifies the area of the Excel file to apply the styling to.

Apply to fields/columns

Identifies the columns in the worksheet to which the rules apply.

If **Identify columns by** above is *FLDNAM, names entered on this parameter element are interpreted as field names from the input file. The rules will apply to the corresponding columns in the worksheet.

If **Identify columns by** above is *XLCOLID, names entered on this parameter element are interpreted as column letters A-Z, AA-ZZ etc.) The rules will apply to the columns thus identified in the worksheet.

You may enter up to 50 field names or column letters. The default is the single value ***ALL** indicating that the rules apply to all columns in the worksheet.

CNDFMTRULE – Conditional formatting rules

Parameter	**CNDFMTRULE**
Description	**Defines individual conditional formatting rules**
Applies to commands:	**CVTDBFXL**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

Specifies conditional formatting rules.

A conditional formatting rule group defines a group of related rules which will be applied, in a given priority sequence, to a range of cells in order to determine the appearance of those cells. For example, you might define a rule that tests the value of a customer account balance field and make rows where the balance is negative red and those where it is above a certain level green etc.

The CNDFMTGRP parameter defines conditional formatting rule groups and group-level attributes such and the range of cells to which the rules in the group will be applied.

The CNDFMTRULE parameter defines the individual rules within those groups which are tested in turn.

Please note that Excel does not allow all of the attributes that can be specified for a style on the DFNSTYLE parameter to be modified when using conditional formatting. In addition, the number of attributes that can be set when defining conditional formatting is even more limited when you are outputting to *XLS format. In particular, number formats, font names and font sizes cannot be changed when outputting to *XLS format.

Also, there are restrictions on the types of test that can be used when outputting to *XLS format. Versions of Excel prior to Excel 2007 do not support many of the more complex test types shown below.

Rule group number

Specifies an arbitrary, non-zero, positive integer which identifies the rule group. You can choose any number you like to identify the group, but it must be unique for all rule groups defined on the command.

The rule group number you specify here must correspond to the rule group number of a rule group defined on the CNDFMTGRP parameter. The CNDFMTGRP parameter defines group-level attributes such as the range of cells to which the rules should be applied, whereas the CNDFMTRULE parameter defines the individual rules in the group that will be tested, one after another, in the priority sequence you specify, against those cells.

Rule priority

The priority of this conditional formatting rule. Where the group contains several rules, this value is used to determine which rule takes precedence and there which style is applied.

Lower numeric values are higher priority than higher numeric values, where 1 is the highest priority.

Identify columns by

How the columns the rules should apply to are identified on the following element:

Options are:

***FLDNAM** (Default). The following parameter element will contain one or more field names from the input file that identify the columns in the worksheet to which the rules apply. The rules will apply to the columns corresponding to those fields.

***FLDNBR** The following parameter element will contain one or more field numbers from the input file that identify the columns in the worksheet to which the rules apply. The rules will apply to the columns corresponding to those fields. The field numbers refer to the relative position of the fields in the database file record (1=first field, 2 = second field etc.)

***XLCOLID** The following parameter element will contain one or more column identifiers (A-Z, AA-ZZ etc.) identifying the columns in the worksheet to which the rules apply.

***MAPREF** (CVTDBFXL only) A database map is being used to control the structure of the Excel output file being created and the styling is being applied to an area of the Excel file. The next element contains a map reference that identifies the area of the Excel file to apply the styling to.

Apply to fields/columns

Identifies the columns in the worksheet to which the rules apply.

If **Identify columns by** above is *FLDNAM, names entered on this parameter element are interpreted as field names from the input file. The rules will apply to the corresponding columns in the worksheet.

If **Identify columns by** above is *XLCOLID, names entered on this parameter element are interpreted as column letters A-Z, AA-ZZ etc.) The rules will apply to the columns thus identified in the worksheet.

You may enter up to 50 field names or column letters. The default is the single value ***ALL** indicating that the rules apply to all columns in the worksheet.

Field to test

Specifies the field in the input file which is tested in order to determine if the rule should evaluate to true or false.

Options are:

***CELLIS** (Default). The logic test is carried out on each individual cell within the range of cells to which this rule is applied, not to any particular column.

For example, if you were using conditional formatting rules to apply different colors, and you specify a *CELLIS rule, each separate cell in the range of cells to which the rules apply will be colored differently depending on the value of those individual cells.

***FORMULA** You will specify a formula on the **Parameter value 1** element below. That formula will determine whether the rule evaluates to true or false and therefore what styling is applied.

field_name Specify a field name from the input file. CoolSpools Database will generate a formula which carries out the required logic test against the value of this particular field. For each row, the value of this field in that row will determine the formatting of cells to which this rule applies.

For example, if your file contains customer account details, and you wish to color the rows based on the value of the customer's account balance, you might specify the BALANCE field here as that is the field which determines how the rows should be formatted.

Test to apply

Specifies the logic test which is carried out to determine if rule evaluates to true or false and therefore what styling to apply.

Note that

Options are:

***NONE** (Default) None. Only valid if *FORMULA is specified or the previous element, i.e. you will specify your own formula to apply on the **Parameter value 1** element below.

The following tests compare the value of field identified by the previous parameter element, or each individual cell (if *CELLIS was specified), against the parameter value or values specified on the **Parameter value 1** and **Parameter value 2** elements below.

***EQ** Equal.

A single value must be specified on **Parameter value 1** below. The rule is true if the field or cell value is equal to this value.

***GT** Greater than.

A single value must be specified on **Parameter value 1** below. The rule is true if the field or cell value is greater than this value.

***LT** Less than.

A single value must be specified on **Parameter value 1** below. The rule is true if the field or cell value is less than this value.

***GE** Greater than or equal to.

A single value must be specified on **Parameter value 1** below. The rule is true if the field or cell value is greater than or equal to this value.

***LE** Less than or equal to.

A single value must be specified on **Parameter value 1** below. The rule is true if the field or cell value is less than or equal to this value.

***NE** Not equal to.

A single value must be specified on **Parameter value 1** below. The rule is true if the field or cell value is not equal to this value.

***BETWEEN** Between.

Two values must be specified on **Parameter value1** and **Parameter value 2** below. The rule is true if the field or cell value is greater than or equal to the first value and less than or equal to the second value.

***NOTBETWEEN** Not between.

Two values must be specified on **Parameter value1** and **Parameter value 2** below. The rule is true if the field or cell value is less than the first value or greater than the second value.

***CT** Contains.

A single value must be specified on **Parameter value 1** below and it is interpreted as a text string. The

rule is true if the field or cell value contains the text string specified.

Not supported when outputting to *XLS format.

***CONTAINS** Same as *CT.

***NC** Not contains.
A single value must be specified on **Parameter value 1** below and it is interpreted as a text string. The rule is true if the field or cell value does not contain the text string specified.

Not supported when outputting to *XLS format.

***NOTCONTAINS** Same as *NC.

***BEGINSWITH** Begins with.

A single value must be specified on **Parameter value 1** below and it is interpreted as a text string. The rule is true if the field or cell value begins with the text string specified.

Not supported when outputting to *XLS format.

***ENDSWITH** Ends with.

A single value must be specified on **Parameter value 1** below and it is interpreted as a text string. The rule is true if the field or cell value ends with the text string specified

Not supported when outputting to *XLS format.

***BLANKS** Contains blanks.

The value of the **Parameter value 1** below is irrelevant and is ignored. The rule is true if the field or cell value is blank (is empty or contains only spaces).

Not supported when outputting to *XLS format.

***NOTBLANKS** Does not contain blanks.

The value of the **Parameter value 1** below is irrelevant and is ignored. The rule is true if the field or cell value is not blank (is not empty or does not contain only spaces).

Not supported when outputting to *XLS format.

***TIMEPERIOD** Time period.

The value of the **Parameter value 1** below must be one of the special time-period values listed below (*LASTMONTH etc.). The rule is true if the field or cell value is number which Excel can interpret as a date and that date matches the time period specified.

Not supported when outputting to *XLS format.

The following tests are only supported if *CELLIS was specified for the **Field to test** element. Each value of cell in the range covered by the rule group is tested individually.

***TOPN** Top n values.

The value of the **Parameter value 1** below must be a number indicating the value of n. The rule is true if the field or cell value in the top n values.

Not supported when outputting to *XLS format.

***BOTTOMN** Bottom n values.

The value of the **Parameter value 1** below must be a number indicating the value of n. The rule is true if the field or cell value in the bottom n values.

Not supported when outputting to *XLS format.

***TOPNPC** Top n percent.

The value of the **Parameter value 1** below must be a number indicating the value of n. The rule is true if the field or cell value in the top n percent of values.

Not supported when outputting to *XLS format.

***BOTTOMNPC** Bottom n percent.

The value of the **Parameter value 1** below must be a number indicating the value of n. The rule is true if the field or cell value in the bottom n percent of values.

Not supported when outputting to *XLS format.

***DUPLICATE** Duplicate values

The value of the **Parameter value 1** below is irrelevant and is ignored. The rule is true if the field or cell value is not unique in the range.

Not supported when outputting to *XLS format.

***UNIQUE** Duplicate values

The value of the **Parameter value 1** below is irrelevant and is ignored. The rule is true if the field or cell value is not unique in the range.

Not supported when outputting to *XLS format.

Parameter value 1

The first parameter value required for the test defined above.

The interpretation of the parameter element is dependent on the value of the **Test to apply** element:

Value of **Test to apply**	Interpretation of **Parameter value 1**
*EQ, *LT, *LE, *GT, *GE, *NE	A value representing a number or string, e.g. 1000 New York
*BETWEEN, *NOTBETWEEN	The first of a pair of values representing numbers or strings. The second value in the pair must be specified on Parameter value 2. 1000 A
*CT, *NC, *CONTAINS, *NOTCONTAINS, *BEGINSWITH, *ENDSWITH	A value representing a text string, e.g. New York
*TIMEPERIOD	Must be one of the special time-period values specified below, e.g. *LASTMONTH.
*TOPN, *BOTTOMN	The ranking value, e.g. 10 = “Top 10”
*TOPNPC, *BOTTOMNPC	The percentage value, e.g. 10 = “Top 10%”

When **Test to apply** is *TIMEPERIOD, the value must be one of the following special time periods:

***THIS MONTH**	This month. The date falls in the current calendar month.
***LASTMONTH**	Last month. The date falls in the previous calendar month.
***NEXTMONTH**	Next month. The date falls in the following calendar month.
***THISWEEK**	This week. The date falls in the current week.
***LASTWEEK**	Last week. The date falls in the previous week.
***NEXTWEEK**	Next week. The date falls in the next week.
***LAST7DAYS**	Last 7 days. The date falls in the last seven days.
***TODAY**	Today. The date is the current date
***YESTERDAY**	Yesterday. The date is one day prior to the current date.
***TOMORROW**	Tomorrow. The date is one day after the current date.

When **Field to test** is *FORMULA, you must specify a formula of your own on this parameter element. If the result of the formula is true, the style associated with this rule will be applied. When specifying cell references in your formula, the row number should correspond to the data row in the worksheet, taking account of column headings and additional heading rows. Use a relative column reference to test each cell in the range separately or an absolute column reference to test the value of a specific column. Do NOT precede the formula by an equals sign = as you might do in a cell.

There is one other special value: ***AVG.** This allows you test against the average value for the selected range. This is only permitted where:

- **Field to test** is *CELLIS
- **Test to apply** is *EQ, *GT, *LT, *LE or *GE

Parameter value 2

The second parameter value required for the test defined above.

The default is ***NONE**.

A value other than ***NONE** must be specified if the test is *BETWEEN or *NOTBETWEEN. The value specified here must be greater than or equal to the value specified on **Parameter value 1**.

A value other than ***NONE** should not be specified for any other test.

Apply style name

The name of the style to apply if the rule evaluates to true.

The style name must match the name of a style defined with WRKSTLDFN or CRTSTLDFN or specified on the DFNSTYLES parameter.

Note that Excel does not allow all of the attributes that can be defined on the DFNSTYLES parameter to be controlled by conditional formatting. For example, while you can change the text color or make the text bold or italic, you cannot change the font name or font size. If you attempt to modify these using conditional formatting, Excel will ignore that change.

Stop if true

Determines whether Excel stops evaluating rules in the group as soon as one has evaluated to true or whether it carries on and checks the next rule.

If this flag is '1',

.Apply to rows

Which rows in the worksheet the rules should be applied to.

Options are:

***YES**	(Default). No rules with lower priority may be applied over this rule, when this rule evaluates to true
***NO**	Other rules with a lower priority will also be evaluated and may override aspects of the formatting.

Examples

The following examples assume that the DFNSTYLES parameter (not shown here for the sake of clarity) has been used to define styles called RED, ORANGE and YELLOW (which might set the cell colors to have a red, orange or yellow background, for example).

Example 1:

```
CVTDBFXL
        FROMFILE(CUSTACCT)
        ...
        CNDFMTGRP(      (1 BALANCES *USED *FLDNAM *ALL)
                        (2 DUESOON *ALL *FLDNAM (DUEDATE)))
        CNDFMTRULE(     (1 1 BALANCE *LT 0 RED *YES)
                         (1 2 BALANCE *BETWEEN 0 100 ORANGE)
                         (2 1 DUEDATE *TIMEPERIOD *NEXTMONTH *NONE YELLOW))
```

Here the customer accounts file is being converted to an Excel spreadsheet.

Two groups of conditional formatting rules are defined:

- Group 1 (named “BALANCES”) which is applied to all columns

  This has two rules:

  - If the value of the BALANCE field is zero, the entire row will have the RED style applied to it
  - If the value of the BALANCE field is between 0 and 100, the entire row will have the ORANGE style applied to it.

- Group 2 (named “DUESOON”) which is applied just to the DUEDATE field

  This has a single rule:

  - If the DUEDATE field contains a date in the following calendar month, the YELLOW style is applied to the DUEDATE cell in the row.

Example 2:

```
CVTDBFXL
        ...
        CNDFMTGRP(      (1 ABOVE_AVERAGE  *USED *FLDNAM BALANCE))
        CNDFMTRULE(     (1 1 BALANCE *GT *AVG *NONE RED *YES))
```

A single conditional formatting group is defined:

- Group 1 (named “ANOVE_AVERAGE”) which is applied just to the column corresponding to the field called BALANCE.

  This consists of a single rule:

  - If the value of the BALANCE field in the current row is greater than the average value of the BALANCE field in all rows, the balance field will have the RED style applied to it

Example 3:
CVTDBFXL

```
...
CNDFMTGRP((1 TOP_10_PERCENT  *USED *FLDNAM BALANCE))
CNDFMTRULE((1 1 *CELLIS *TOPNPC 10 *NONE RED *YES))
```

A single conditional formatting group is defined:

- Group 1 (named “TOP_10_PERCENT”) which is applied just to the column corresponding to the field called BALANCE.

  This consists of a single rule:

  - If the value of the BALANCE field in the current row is in the top 10% of values in the BALANCE field for all rows, the balance field will have the RED style applied to it

Example 4:
CVTDBFXL

```
...
CNDFMTGRP((1 ALERT_FLAG  *USED *FLDNAM *ALL))
CNDFMTRULE((1 1 ALERT *NOTBLANKS *NONE *NONE RED *YES))
```

A single conditional formatting group is defined:

- Group 1 (named “ALERT_FLAG”) which is applied to all columns.

  This consists of a single rule:

  - If the value of the ALERT field in the current row is not blanks, the entire row have the RED style applied to it

Example 5:
CVTDBFXL

```
...
CNDFMTGRP( (1 NORTH_AMERICA *USED *FLDNAM *ALL))
CNDFMTRULE((1 1 *FORMULA *NONE 'OR($J2="US",$J2="CA")' *NONE RED *YES))
```

A single conditional formatting group is defined:

- Group 1 (named “NORTH_AMERICA”) which is applied to all columns.

  This consists of a single rule:

  - A user-defined formula is applied. The formula checks the value of column J (assumed to contain a country code) and applies the RED style if the value is US or CA. Note that an absolute column reference ($J) is used and that the row reference is 2 (assuming a single row of column headings at the start of the sheet).

INCLFILE – Include files

Parameter	**INCLFILE**
Description	**Specifies image files to be included in Excel output**
Applies to commands:	**CVTDBFXL**
Dependent on:	**None**
Supports CoolSpools Database variables	**Yes**

Specifies one or more image files to be included in the Excel file.

Only support for Excel 2007 format, i.e. when EXCEL(*XLSX) is specified.

Image types currently supported are:

- JPEGs
- GIFs
- PNGs

Single values

***NONE** No images will be included in the Excel output.

Other values (up to 100 repetitions)

Image source type

How CoolSpools should locate the image file to include.

***PATH** The next element specifies the path to an image file to be included.

***FIELD** The next element specifies the name of a column or field from the input data which contains the path to an image file to be included.

***EXITPGM** The next field specifies the name of an exit program that will be called to provide the path to an image file to be included.

See member SL_IMGEXTR in source file COOLSPV7R1/SL_SRCFILE for sample source of an exit program that returns the path to an image file to be included.

Image path, column or pgm name

If the previous element is ***PATH**, specify the path name of the image file to include.

If the previous element is ***FIELD**, specify the name of the field or column from the input data.

If the previous element is ***EXITPGM**, specify the name of the exit program to call.

Anchor point specified by

Specifies how the position of the image in the Excel file is determined.

Please note that is you specify an anchor point, lower than the last row in the spreadsheet being created, the image may not be included in the print area.

***CELLREF**	The next two elements specify an Excel column letter and a row number where the top-left corner of the image file should be located.
***INCH**	The next two elements specify absolute coordinates on screen in inches.
***CM**	The next two elements specify absolute coordinates on screen in centimeters.
***MM**	The next two elements specify absolute coordinates on screen in millimeters.
***PTS**	The next two elements specify absolute coordinates on screen in points (72 points per inch).

Anchor point column/X

Specify one of the following.

A-XFD	Excel column reference. If "Anchor point specified by" is *CELLREF, specify the Excel column letter where the top left corner of the image should be located.
.00-99999.99	X coordinate. If "Anchor point specified by" is not *CELLREF, specify the absolute horizontal coordinate on screen measured in the unit given on "Anchor point specified by".

Anchor point row/Y

Specify one of the following.

***ALL** The image will be included at the specified horizontal location for each row.

row_number If "Anchor point specified by" is *CELLREF, specify the row number where the top left corner of the image will be located.

.00-99999.99 Y coordinate.

If "Anchor point specified by" is not *CELLREF, specify absolute vertical coordinate on screen measured in the unit given on "Anchor point specified by".

Size specified by

Determines the size of the image.

***IMAGE** The size of the image is calculated from the image properties (width and height in pixels and resolution).

***SCALE** The next two elements specify scaling factors by which the image size implied by the image properties will be multiplied.

***CELLREF** The next two elements specify an Excel column letter and a row number where the bottom right corner of the image will be located.

***INCH** The next two elements specify absolute coordinates on screen where the bottom right corner of the image will be located, measured in inches.

***CM** The next two elements specify absolute coordinates on screen where the bottom right corner of the image will be located, measured in centimeters.

***MM** The next two elements specify absolute coordinates on screen where the bottom right corner of the image will be located, measured in millimeters.

***PTS** The next two elements specify absolute coordinates on screen where the bottom right corner of the image will be located, measured in points.

Horizontal size/scaling

***IMAGE** If "Size specified by" is *IMAGE, this value is mandatory as this element is then unused.

A-XFD Excel column reference.

If "Size specified by" is *CELLREF, specify an Excel column letter where the bottom right corner of the image will be located.

scaling_factor If "Size specified by" is *SCALE, specify the horizontal scaling factor (1.00 = no scaling, 0.5 = half size etc.)

X coordinate If "Size specified by" is not *CELLREF or *SCALE, specify the horizontal coordinate of the bottom right corner of the image, measured in the unit defined on "Size specified by".

Vertical size/scaling

***IMAGE** If "Size specified by" is *IMAGE, this value is mandatory as this element is then unused.

***SAME** The bottom right corner of the image appears in the same row as the top left corner, i.e. the height of the image is one row.

absolute_row If "Size specified by" is *CELLREF, and only one copy of the image is being output ("Anchor point row/Y" is not *ALL), specify the absolute row number in the Excel file at which the bottom right corner of the image will be located.

relative_row If "Size specified by" is *CELLREF, and the image is being output for each record ("Anchor point row/Y" is *ALL), specify the row number relative to the top left corner at which the bottom right corner should be located, e.g. a value of 2 would place the bottom right corner 2 rows below the top left corner and the image would occupy 3 rows.

scaling_factor If "Size specified by" is *SCALE, specify the vertical scaling factor (1.00 = no scaling, 0.5 = half size etc.)

Y coordinate If "Size specified by" is not *CELLREF or *SCALE, specify the vertical coordinate of the bottom right corner of the image, measured in the unit defined on "Size specified by"

Rotation angle

0 The image is not rotated.

integer Specify the angle through which the image is rotated, in degrees.

Example:
CVTDBFXL

```
...
INCLFILE((*PATH logo.jpg *CELLREF A 1 *IMAGE))
EXCEL(*XLSX)
```

The JPEG image called **logo.jpg** in the current directory is output in cell A1.

Example:
CVTDBFXL

```
...
INCLFILE((*PATH logo.jpg *CELLREF A*ALL *IMAGE))
EXCEL(*XLSX)
```

Logo.jpg is output in column A of each row.

Example:
CVTDBFXL

...
INCLFILE((*PATH '/images/'<:LOGO:>png' *CELLREF A*ALL *IMAGE))
EXCEL(*XLSX)

The value of field LOGO from the input field is retrieved for each record and used to build a path to the image to be included. The path is **/images/xxxxxx.png** where xxxxxx is the value of the field LOGO. The image is output in cell A of each row.

Example:
CVTDBFXL

...
INCLFILE((*FIELD LOGO D *ALL *IMAGE))
EXCEL(*XLSX)

The path to the image file to output is retrieved from the field called LOGO in the input source and the corresponding image is output in column D of each row.

Example:
CVTDBFXL

...
INCLFILE((*EXITPGM IMGEXIT D *ALL *IMAGE))
EXCEL(*XLSX)

The path to the image file to output is retrieved by calling exit program IMGEXIT and the corresponding image is output in column D of each row.

APYSTYLES – Field styles and attributes

Parameter	**APYSTYLES**
Description	**Lets you override the style of individual fields, cells or sections of the output file and set field-level attributes.**
Applies to commands:	**CVTDBFXL, CVTDBFXLSX, CVTDBFHTML, CVTDBFXML, CVTDBFCSV, CVTDBFTXT**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The APYSTYLES parameter allows you to specify the styling to be applied to parts of the output file or to specify field-level properties such as date formatting

The single value *DFT indicates that CoolSpools Database will take all defaults. No field-level overrides will occur.

There are two ways in which to associate a style with a piece of data in your output file (e.g. a cell in an Excel spreadsheet or an element of an XML document):

1. **Implicitly**

   By defining the style name the same as the name of a row group in your Database-to-Excel map or an element in your Database-to-XML map, you implicitly apply that style to the data in question. For example, a style called REPORT_HEADING will be implicitly and automatically applied to an Excel row group called REPORT_HEADING.

   Note that style names are case-sensitive because XML element names need to be case-sensitive and for this association of names to work, the names must match exactly in terms of case.

2. **Explicitly**

   Alternatively, use the APYSTYLES parameter to define the styles you wish to apply to different parts of the file you create.

Identify by

Specifies how the field or column or area of the output file being referred to should be identified.

Options are:

***FLDNAM** (Default). This parameter relates to a single field in the input file and the field is to be identified by means of its field or ALIAS name (and optionally also record format name), specified on the next element.

***FLDNBR** This parameter relates to a single field in the input file and the field is identified by means of its field number, specified on the next element.

***XLCOLID** (CVTDBFXL only) This parameter relates to a single column in the Excel file being created and that column is identified by means of its column identifier (letter or letters), specified on the next element.

***MAPREF** (CVTDBFXL only) A database map is being used to control the structure of the Excel output file being created and the styling is being applied to an area of the Excel file. The next element contains a map reference that identifies the area of the Excel file to apply the styling to.

***ELEMENT** (CVTDBFXML only) This parameter relates to a single element in the Excel file being created and that element is identified by means of its element name, specified on the next element.

Fld name, nbr, col id, map ref (CVTDBFXL)

Field name or number (CVTDBFHTML, CVTDBFCSV, CVTDBFTXT)

Field name, number or element (CVTDBFXML)

The interpretation of this parameter element is dependent on the value of the preceding element “Identify by”.

Field_name Where ***FLDNAM** was specified for previous element, specify the name of a field from the input source. The styling and attributes relate to the field so identified.

Where a field name is entered here, the record format name defaults to *ONLY. If more than one record format has been selected on the RCDFMT parameter, *ONLY is invalid and you will need also to specify the record format in which the field can be found, in the format record_name/field_name.

You may specify either a short (10-character) database field name or a long (30-character) column name (DDS ALIAS).

Example:

```
CVTDBFXL
    FROMFILE(QADSPOBJ)
    ...
    DFNSTYLES((library...))
    APYSTYLES( (*FLDNAM ODLBNM library))
```

Here a style called “library” is being applied to the ODLBNM field.

Field_number Where *FLDNBR was specified for previous element, specify the field number of a field from the input source on this element.

Field numbers refer to the relative position of the field in the input source, counting from 1 for the first field. While this will be the same as the column number in an SQL statement, it is not necessarily the same as the column number in the output file (Excel, HTML etc.), as columns may have been excluded using the EXCLFLD parameter, or their sequence changed on the INCLFLD parameter.

Example:

```
CVTDBFXL
     FROMFILE(*SQL)
     ...
     SQL('select odlbnm,sum(odobsz) from qadspobj group by odlbnm order by 2 desc')
     ...
     APYSTYLES( (*FLDNBR 2 *DATA *HEADER 'Total of Object Sizes'))
```

Here, an SQL statement is being run against the DSPOBJD outfile to provide input to the conversion. The APYSTYLES parameter is used with the *FLDNBR option to apply a column heading to the second field returned by the SQL statement, namely SUM(ODOBSZ).

Column_ID (CVTDBFXL only) Where *XLCOLID was specified for previous element, specify the column identifier (letter or letters) on this element, for example A for the first column in a worksheet.

Example:

```
CVTDBFXL
     FROMFILE(*SQL)
     ...
     SQL('select odlbnm,sum(odobsz) from qadspobj group by odlbnm order by 2 desc')
     ...
     APYSTYLES( (*XLCOLID B*DATA *HEADER 'Total of Object Sizes'))
```

This is the same as the previous example, except that the styling is applied by specifying the column letter of the column in the Excel worksheet.

Map_Ref (CVTDBFXL only) Where *MAPREF was specified for previous element, specify the map reference determining the area of the Excel file to which the styling should be applied.

A map reference takes the following format:

row_group_name

or

row_group_name(cell_reference)

Where **row_group_name** is the name of a row group defined in the controlling database-to-Excel map and **cell_reference** (optional) is an Excel-style cell reference of the form ColumnID or ColumnID-RowNumber

(e.g. A for column A or A1 for column A row 1). Note that the row number here refers to the row number within the row group, not the row within the Excel worksheet.

Example:

```
CVTDBFXL
     FROMFILE(*MAP)
     MAPNAME(ORDERS)
     ...
     DFNSTYLES((Turquoise ...)
     APYSTYLES((*MAPREF CUSTOMER_LINE Turquoise))
```

The style called “Turquoise” is applied to all cells in the output Excel file that are generated based on the row group called CUSTOMER_LINE in controlling database-to-Excel map ORDERS.

Example:

```
CVTDBFXL
     FROMFILE(*MAP)
     MAPNAME(ORDERS)
     ...
     DFNSTYLES((Turquoise ...))
     APYSTYLES((*MAPREF ‘CUSTOMER_LINE(A1)’ Turquoise))
```

As before, except that this time the style called “Turquoise” is applied only to column A of the first row of cells in the output Excel file that are generated based on a row group called CUSTOMER_LINE in controlling database-to-Excel map ORDERS.

Element (CVTDBFXML only) Where *ELEMENT was specified for previous element, specify the element name. The styling is applied to that element in the stylesheet that is generated for the XML document.

Note that element names are case-sensitive.

Example:

```
CVTDBFXML
     FROMFILE(*MAP)
     MAPNAME(ORDERS)
     XMLSTYLING(*XSLT *YES)
     DFNSTYLES((Turquoise...)
     APYSTYLES((*ELEMENT 'customer' *DFT *DFT Turquoise))
```

A database-to-XML map is used to generate an XML document and associated XSLT stylesheet. The style called “Turquoise” is applied to nodes in the XML file that are generated based on the element called “customer” in controlling database-to-Excel map ORDERS.

Element or attribute

(XML only)

Whether the field should be output as an XML element or attribute.

This option is relevant only where “Identify by” is *FLDNAM or *FLDNBR and this parameter relates to a single field in the input source. It is not relevant and is ignored where “Identify by” is *ELEMENT.

Options are:

***DFT**	(Default). The field will be converted to a sub-element within the row element or an attribute of the row element depending on the value of the **Cols as elements or attributes** option of the XML parameter.
***ELEMENT**	The field will be converted to a sub-element of the row element.
***ATTRIBUTE**	The field will be converted to an attribute of the row element.

Element or attribute name

(XML only)

The name of the sub-element or attribute this field is converted to.

This option is relevant only where “Identify by” is *FLDNAM or *FLDNBR and this parameter relates to a single field in the input source. It is not relevant and is ignored where “Identify by” is *ELEMENT.

Options are:

***DFT**	(Default). The name of the sub-element or attribute this field is converted to will be generated based on the value of the **Generate elem/attr names from** option of the XML parameter.
***FLDNAM**	The name is generated from the field name
***QUALFLD**	The name is generated from the qualified field name (field name and record name).
***ALIAS**	The name is generated from the field alias.
***QUALALIAS**	The name is generated from the qualified field alias (field alias and record name).
name	Specify the name to use

Data style name

(Excel, HTML and XML only)

The name of the style to be applied to this field in data rows. Use this option to apply a different named style to the areas of output file identified by the first two parts of this parameter.

Options are:

***DATA**	(Default). The predefined *DATA style is applied.
***HEADER**	The default style name for header rows. If you specify *HEADER for the name of the style, the attributes you specify will become the default attributes for header rows (rows generated as a result of the HEADER parameter settings).

***TITLE** The default style name for title rows.

If you specify *TITLE for the name of the style, the attributes you specify will become the default attributes for title rows. Title rows are those generated from the additional heading lines elements of the HEADER parameter and the caption text of the HTML parameter.

***SUBTOTAL** The default style name for subtotal rows.

If you specify *SUBTOTAL for the name of the style, the attributes you specify will become the default attributes for subtotal rows. Subtotal rows are those that result from subtotals and group-by fields in Query/400 queries when the *COMBINED output form is selected.

***TOTAL** The default style name for total rows.

If you specify *TOTAL for the name of the style, the attributes you specify will become the default attributes for total rows. Total rows are those that result from subtotals and group-by fields in Query/400 queries when the *COMBINED output form is selected.

***HYPERLINK** If you specify *HYPERLINK for the name of the style, the field will become a clickable hyperlink.

***FLWDLINK** If you specify *FLWDLINK for the name of the style, the field will become a clickable hyperlink, with the appearance of a link already visited.

***ROOT** **XML Only.** The default style for the root element.

***ROW** **XML Only.** The default style for the row element, i.e. the element corresponding to a record in the input file.style_name Specify the name of a predefined or user-defined style to be applied to his field in data rows.

Header style name

(Excel, HTML and XML only)

The name of the style to be applied to this field in header rows. Use this option to apply a different named style to the header rows corresponding to fields identified by the first two parts of this parameter.

Note that this parameter element is ignored where a map is being used as headers are controlled not by the HEADER parameter but by the database map itself.

Options are:

***HEADER** (Default). The predefined *HEADER style is applied.

style_name — Specify the name of a predefined or user-defined style to be applied to his field in header rows.

Column heading

The column heading text for this field.

Note that this parameter element is ignored where a map is being used as column headings are controlled by the database map itself.

Options are:

***HEADER** — (Default). The values defined on the HEADER parameter will dictate the heading text.

***AVAIL** — CoolSpools Database will select the best available labels from the input file to create the header row. This selection of label text is performed according to the following criteria. If the fields in the input file have Column Headings (DDS COLHDG keyword), they will be used to generate the column headings. Otherwise, if the fields in the input file have field aliases (DDS ALIAS keyword), those will be used instead. Otherwise, if the fields in the input file have text descriptors (DDS TEXT keyword), those will be used. Otherwise the field names will be used.

For fixed-length ASCII text files, no header row will be created.

***NONE** — No header row is created.

***COLHDG** — For CSV, Excel and HTML files, a header row is created from the field column headings (DDS COLHDG keyword).
This option is invalid if TOFMT(*FIXED) is specified.

***ALIAS** — For CSV, Excel and HTML files, a header row is created from the field aliases (DDS ALIAS keyword).
This option is invalid if TOFMT(*FIXED) is specified.

***TEXT** — For CSV, Excel and HTML files, a header row is created from the field text descriptors (DDS TEXT keyword).
This option is invalid if TOFMT(*FIXED) is specified.

***FLDNAM** — For CSV, Excel and HTML files, a header row is created from the field names. This option is invalid if TOFMT(*FIXED) is specified.

***COLHDG1**	The first column heading element only.
***COLHDG2**	The second column heading element only.
***COLHDG3**	The third column heading element only.
***COLHDG12**	Column heading elements 1 and 2 only.
***COLHDG13**	Column heading elements 1 and 3 only.
***COLHDG23**	Column heading elements 2 and 3 only
heading_text	Specify the heading text required.

Convert to data type

Allows you to force conversion to a data type other than that to which the field or column would normally be converted. For example, you may wish to indicate that an alphanumeric field contains numeric data and force conversion to numeric output or that a numeric field contains a date and force conversion to date output.

***DATATYPE**	The field is converted to the format indicated by the data type of the field.
***NUMERIC**	Force conversion to numeric output even where the field data type is non-numeric.
***DATE**	Force conversion to date output even where the field data type is not a date and the editing does not suggest a date held in a numeric field.
***TIME**	Force conversion to time output even where the field data type is not a time.
***HYPERLINK**	The value of the field is converted to a hyperlink. No validation is performed by CoolSpools Database on the hyperlink specified. You should ensure that it is formatted correctly and includes the prefix http://, ftp:// etc. as appropriate.
***FORMULA**	Check for a formula in the field or column and create an Excel formula if one is found. An Excel formula must be indicated to CoolSpools Database by enclosing it in the characters: ==XL( )

For example, to generate a formula that calculates the sum of the range of cells A1 to A10, you would specify:

==XL(SUM(A1::A10))

No validation is performed by CoolSpools Database on the Excel formula specified and a file open error may occur if the formula is specified incorrectly.

N.B.

This option is only supported when creating an Office Open XML format file (i.e. when EXCEL(*XLSX) is specified). Note also that some performance degradation might be noticeable due to the extra overhead associated with checking for formulas.

***IMAGE**

The value of the field is interpreted as the path to an image file which will be included at the appropriate position for each record.

This option can be used on any of the following data types:

- character
- variable-length character
- graphic
- variable-length graphic
- CLOB
- DBCLOB

This option can also be used on a BLOB field, in which case the BLOB value will be interpreted as the image file itself rather than a path to an image file.

N.B.

This option is only supported when converting to Office Open XML format (*XLSX).

***LABEL**

Specify the fixed text associated with with a variable.

Format of num/char date/time

Indicates that the field in question contains a date or time and specified the format of this date or time.

If you have a file that contains dates or times in character fields (for example the outfile created by the DSPOBJD command), or times held in numeric fields, the only way to tell

CoolSpools Database that these fields should be treated as dates or times is to specify those fields on this parameter and define their format on this element.

If you have a file that contains dates in numeric fields, and CoolSpools Database does not recognize those fields as dates, (for example, because they have an unusual size or number of decimals or no date edit code/word), you can define those fields and their date formats here.

Similarly, if your file contains some numeric dates in one format (e.g. *MDY) and some in another (e.g. *YMD), you can specify the most commonly used date format on the DBFDATFMT parameter and list the fields that use a different format here.

Options are:

***NONE**	The field is not a date.
***YMD**	The field is a date in YYMMDD or YYYYMMDD format
***DMY**	The field is a date in DDMMYY or DDMMYYYY format
***MDY**	The field is a date in MMDDYY or MMDDYYYY format
***CYMD**	The field is a date in CYYMMDD format
***CDMY**	The field is a date in CDDMMYY format
***CMDY**	The field is a date in CMMDDYY format
***HMS**	The field is a time in HHMMSS format.

Example:

```
CVTDBFXL FROMFILE(QADSPOBJ)
    ...
    APYSTYLES((*FLDNAM  ODCDAT *DATA *HEADER *HEADER *MDY *NONE)
              (*FLDNAM  ODSRCD *DATA *HEADER *HEADER *YMD *NONE)
              (*FLDNAM  ODCTIM *DATA *HEADER *HEADER *HMS *NONE))
```

Here we are converting the outfile from the DSPOBJD command. The creation date field ODCDAT is flagged to CoolSpools Database as a date in *MDY format inside a character field. Similarly, ODSRCD (source change date) is also a date inside a character field, but is in *YMD format. Finally, ODCTIM (the creation time) is identified as a time value inside a character field.

Separator of char date/time

Where the field is a date or time inside a character field, this element defines the separator used to format the date or time.

Options are:

***NONE**	(Default). The character date does not contain a separator.
***SYSVAL**	The separator is the character defined by the QDATSEP system value
***JOB**	The separator is the character defined by the DATSEP job attribute
***SLASH**	/
***HYPHEN**	-
***PERIOD**	.

***COMMA**	,
***COLON**	:
***BLANK**	Space
sep_char	Specify the separator character

COLWIDTHS – Column widths

Parameter	**COLWIDTHS**
Description	**Let you specify the width of columns**
Applies to commands:	**CVTDBFXL**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The COLWIDTHS parameter allows you to define the widths of individual columns in an Excel workbook.

The single value *DFT indicates that CoolSpools Database will use the default method of specifying column widths, taken from the EXCEL parameter.

This attribute has not been implemented for HTML and XML because browser behavior and support in this area is just too variable and unreliable.

There is a single value:

***DFT** (Default). Calculate the width of the column in the default manner, i.e. using the method specified on the **Column width option** element of the EXCEL parameter.

Column reference

The column identifier (letter or letters) identifying the column whose width is being defined.

Column width

Set the width of the column for this field.

Options are:

***ENVVAR** Calculate the width of the column based on the setting of the CS_XLS_COLUMN_WIDTH environment variable. If this environment variable does not exist, or if it is not set to one of the following values, the value *AUTOFIT is assumed:

*AUTOFIT
*FIELDSIZE
*FITTEXT

***FIELDSIZE** Base the column width on the size of the field, according to its DDS definition. For example, an alphanumeric field that has a size of 50 characters will have the corresponding column width set to 50 characters too.

***AUTOFIT** Base the column width on the length of the largest data value, measured in characters. For example, an alphanumeric field that has a size of 50 characters but where the longest value is 40 characters will have the corresponding column width set to 40 characters.

***FITTEXT** Base the column width on the character width of the largest data value. The character width here is the width of the text as displayed in the selected font, measured in points. This option may give the best results in terms of fitting the column width closely to the width of the displayed text, but please note the following points:

- Use of this option can cause a noticeable degradation in performance, because of the need to calculate the displayed width of each text item
- The displayed width of text items can only be calculated accurately where one of the "well known" fonts (Arial, Courier or Times) is used.
- Excel can use different column width settings when printing data as opposed to when data is displayed on screen. If columns are fitted too closely to the width of the text when displayed on screen, the data may appear as ##### when printed (i.e. might not fit in the column width).

column_width Specify the column width in characters.

For Excel, this can be difficult for CoolSpools Database to calculate as it is dependent on the metrics of the fonts that are being used, which may not be available at the time the Excel workbook is created.

SORT – Sort specifications

Parameter	**SORT**
Description	**Defines how the data should be sorted**
Applies to commands:	**CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFXML, CVTDBFTXT** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **SORT** (Sort Specifications) parameter allows you change the order in which records retrieved from the input source are presented in the output stream file by sorting those records by one or more fields.

Up to 50 sort keys may be specified in the format explained below, or one of these two single values can be selected:

***FILE** (Default). The file keys are used to sequence the records. If the file has no keys, arrival sequence is used.

***NONE** Arrival sequence is used. The records will be presented in the sequence in which they appear physically in the file.

Each sort key has two elements:

Key field name

Field_name Specify the name of the field to use as a sort key.

Key field order

***ASCEND** (Default). Sort in ascending sequence.
***DESCEND** Sort in descending sequence.

Suppress Repetitions

Specified whether repeated values should be suppressed, i.e. output as empty cells.

***NO** Repetitions are not suppressed. If a key value is repeated from one record to the next, each value is output in full.

***YES** Repetitions are suppressed. If a key value is repeated from one record to the next, duplicate values after the first are output as empty cells.

QRYSLT - Query selection expression

Parameter	**QRYSLT**
Description	**Defines selection criteria**
Applies to commands:	**CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFXML, CVTDBFTXT** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**Yes**

The **QRYSLT** (Query Selection Expression) parameter allows you select and exclude records retrieved from the input source by applying criteria that you define here.

***ALL** (Default). All records in the input source are selected.

'Query_selection' Specify an expression of up to 512 characters (contained in apostrophes) that describes the values used to determine which records are selected. To expand the field beyond 512 characters to a maximum of 5000 characters, you must specify the parameter on the command entry display. You can specify any logical expression formed from relationships (such as *EQ and *NE) of field and constant values or functions of field and constant values. At least one field name is specified in each relationship.

The syntax of this query selection string is identical to that used by the OPNQRYF command QRYSLT parameter. See ***http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm?info/dbp/rbafomst207.htm*** for further details of how to use this parameter.

Please note that where the input to CVTDBFXL is one of the special values *SQL, *SQLSRC, *QMQRY or QRYDFN, any record selections made in the SQL or query will reduce the size of the temporary results file which CVTDBFXL will process as its input. The ***CoolSpools Database*** QRYSLT parameter then allows you to apply record selection to the temporary results file if that is appropriate.

For example, you could use the SQL to query your order file and calculate order totals by customer by means of the GROUP BY function. You could use the SQL WHERE clauses to select only customers in New York State. The temporary results file would contain totals for each customer. You could then use the ***CoolSpools Database*** QRYSLT parameter to select only those customers whose totals exceed a given threshold value.

OPTIONS – Miscellaneous command options

Parameter	**OPTIONS**
Description	**Specifies miscellaneous command options**
Applies to commands:	**CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFXML, CVTDBFTXT**
Dependent on:	**None**

The OPTIONS parameter is intended to provide a convenient place where miscellaneous command options can be added to commands without unduly cluttering the command with new parameters.

There are few options available at present but it is anticipated that new options will be added over time as the need arises.

Each command option consists of a key identifying the type of option being specified and a value element which enables the corresponding option value to be defined.

Option key

Options are:

***DIAGNOSTIC** Whether the command should be run in diagnostic mode or not. You should only use this option if instructed to do so by ariadne as in diagnostic mode the command may generate large volumes of logging data, produce inefficient output (e.g. uncompressed PDFs) and take an extended time to run.
***CRTDIR** Whether any directories in the path specified on the TOSTMF parameter will be created if they do not already exist.
***FCNMKR** Override marker for CoolSpools functions to the specified string.
***LINKLABEL** Identifies the field supplying the link label for a hyperlink field (Excel only).
<:var_name:> Defines a variable called **var_name** the value of which is specified on the next element.

***NEWSHEET** Specifies conditions for starting a new worksheet when converting Query/400 query to Excel and the *ENHANCED output style is selected.
Option value

A value corresponding to the key above.

The list of possible valid values is dependent on the corresponding key.

Key	Valid values	Description	Notes
***DIAGNOSTIC**	*NO	Do not run in	Diagnostic mode

		diagnostic mode (default)	should only be used when instructed by ariadne. Performance degradation and the creation of abnormally large output files could result.
	*YES	Run in diagnostic mode	
***CRTDIR**	*NO	Directories in the directory path specified on the TOSTMF parameter must already exist. If they do not, an error will occur (default);	You can set the system default action by creating an environment variable called SL_CRT_DIR_PATH set to either *YES or *NO.
	*YES	Directories in the directory path specified on the TOSTMF parameter will be created if they do not already exist.	
***FCNMKR**	$$	Any string of 2-10 characters	This option should only be used if instructed to do so by ariadne.
	SL_FCN_MARKER	Value of SL_FCN_MARKER environment variable	
***LINKLABEL**	Identifies the field supplying the link label for a hyperlink field (Excel only). The value must be in the format: li**nk_field_name:label_field_name** where link_field_name is the name of the field supplying thehyperlink URL (which must also be specified as data type *HYPERLINK on APYSTYLES) and label_field_name is the name of the field supplying the associated label.		
<:var_name:>	Defines a variable called **var_name** the value of which is specified on the next element. The variable can be specified (in the form <:var_name:>) wherever CoolSpools Database values are supported, e.g. on Database-to-XML element and attribute constant text. This enables you to supply different values for different runs of the		

	command, e.g. run date, year, period, company name etc.		
***NEWSHEET**	*FNLBRKLVL	Start a new sheet before final break summaries.	Specifies conditions for starting a new worksheet when converting Query/400 query to Excel and the *ENHANCED output style is selected.
	QRYBRKLVL1	Start a new sheet after level 1 break summaries.	
***NEWSHEET**	QRYBRKLVL2	Start a new sheet after level 2 break summaries.	
***NEWSHEET**	QRYBRKLVL3	Start a new sheet after level 3 break summaries.	
***NEWSHEET**	QRYBRKLVL4	Start a new sheet after level 4 break summaries.	
***NEWSHEET**	QRYBRKLVL5	Start a new sheet after level 5 break summaries.	

***NEWSHEET**	QRYBRKLVL6	Start a new sheet after level 6 break summaries.	

Example:

```
CVTDBFXL
...
TOSTMF('/dir1/subdir1/subdir2/subdir3/subdir4/subdir5/output.xls')
OPTIONS((*CRTDIR *YES))
```

Directories in the path specified on TOSTMF will be created if they do not already exist.

DBFCCSID – Database file CCSID

Parameter	**DBFCCSID**
Description	**Defines the encoding of data in the input file**
Applies to commands:	**CVTDBFXL, CVTDBFCSV, CVTDBFHTML, CVTDBFPDF, CVTDBFXML, CVTDBFTXT** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The DBFCCSID (Database File CCSID) parameter enables you to specify the CCSID (Coded Character Set Identifier) which best describes the encoding of data in the database file when this information is not otherwise available to CoolSpools Database.

There are two elements to this parameter.

CCSID

The first element indicates the CCSID to be used where the encoding of the data cannot be determined from the metadata associated with the database file.

Options are:

***DBF** The CCSID is determined from the information available in the database file metadata (DDS).
***JOB** The CCSID of the current job is used.
***SYSVAL** The value of the QCCSID system value is used.
***USER** The CCSID attribute of the user profile of the user running the command is used.
CCSID_value Specify a CCSID to use.

Override file CCSID?

The second element indicates whether this CCSID should be used only where no CCSID information is available, or whether it should be used in all cases and override any CCSID information available from the metadata (DDS) associated with the database file.

Options are:

***NO** Use the CCSID only where no CCSID information is available in the database file metadata (DDS).
***YES** Use the CCSID for all data, disregarding any CCSID information available from the metadata. This option can be useful where the encoding of the file is different from that suggested by the CCSID information associated with it.

STMFCODPAG – Stream file code page

Parameter	**STMFCODPAG**
Description	**Defines the encoding of data in the output file**
Applies to commands:	**CVTDBFCSV, CVTDBFHTML, CVTDBFXML, CVTDBFTXT** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The STMFCODPAG (Stream File Code Page) parameter controls the way system i EBCDIC character data is converted to ASCII for inclusion in the stream file.

Options are:

***CALC** CoolSpools Database calculates the most appropriate code page for the stream file based on the CCSID of the data in the database file and the format to which the data is being converted.
***PCASCII** EBCDIC data is converted to a Windows ASCII codepage corresponding to the EBCDIC CCSID of the data in the input file.
***WINDOWS** The same as *PCASCII.
***STDASCII** EBCDIC data is converted to an ASCII codepage corresponding to the EBCDIC CCSID of the data in the input file.
***IBMASCII** The same as *STDASCII.
***ISOASCII** EBCDIC data is converted to an ISO ASCII codepage corresponding to the EBCDIC CCSID of the data in the input file.
UNICODE EBCDIC data is converted to a Unicode codepage corresponding to the EBCDIC CCSID of the data in the input file. UCS2 will be used.
***UCS2** Universal Character Set encoded in 2 octets (a form of Unicode that consistently uses 2 bytes to represent characters).
***UTF8** UCS Transformation Format 8, a form of Unicode which uses 1, 2 or 4 bytes to represent characters.
***UTF16** UCS Transformation Format 16.
***NOCONV** Data is not converted and is left in its original format. This may give the best results with some languages such as Arabic.

CCSID Specify a CCSID between 1 and 65535 representing the encoding scheme to which data is converted.

Note that when converting to a format such as XML, CoolSpools Database does not force you to select an encoding which is likely to be immediately usable. For example, there is nothing to stop you from specifying CVTDBFXML ... STMFCODPAG(37) to create an XML file encoded in US EBCDIC, but it is unlikely that XML consumers will be able to process the file in that encoding.

UNICODE

Parameter	**UNICODE**
Description	**Unicode-related options**
Applies to commands:	**CVTDBFCSV, CVTDBFHTML, CVTDBFXML, CVTDBFTXT** **CVTDBFSTMF (deprecated)**
Dependent on:	**STMFCODPAG(*UNICODE)** **STMFCODPAG(*UTF8)** **STMFCODPAG(*UTF16)** **STMFCODPAG(*UCS2)**
Supports CoolSpools Database variables	**No**

The UNICODE (Unicode Options) parameter allows you to define various Unicode-related options when you specify a Unicode encoding scheme on the STMFCODPAG parameter.

There are two elements to this parameter.

Bigendian or littleendian

In relation to UCS2 encoding, which uses two bytes to represent each character, this option determines the order in which the bytes are represented.

Options are:

***BIG** Bigendian representation, with the most significant byte first (system i norm).
***LITTLE** Littleendian representation, with the least significant byte first (PC norm)

Include Unicode marker?

This option determines whether CoolSpools Database outputs a marker at the start of a text file which indicates to a reader application whether the byte order is bigendian or littleendian.

Some applications such as Windows NotePad check for a marker at the start of the file (hex x'FEFF', x'FFFE') and use this to identify whether UCS2 or UTF-16 Unicode data is encoded in bigendian or littleendian format. The marker x'EFBBB' denotes UTF-8 encoding.

Options are:

***TOFMT** (CVTDBFSTMF only) A marker is output if the TOFMT parameter if *FIXED or *HTML, but not for *CSV.
***YES** A marker is output
***NO** No marker is output

DBFDATFMT – Database date format

Parameter	**DBFDATFMT**
Description	**Specifies the format in which dates are held in numeric fields in the input data**
Applies to commands:	**CVTDBFCSV, CVTDBFHTML, CVTDBFXML, CVTDBFTXT, CVTDBFPDF, CVTDBFXL** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The DBFDATFMT (Database Date Format) parameter allows you to define the format in which dates are held in numeric fields in the database file you are converting.

If a field in the database file being converted is:

- Packed decimal or zoned decimal
- 6, 7 or 8 digits in length
- zero decimal places
- edited with an edit code of 'Y'

CoolSpools Database will attempt to interpret the field as a date and convert it to a date in the output file. In doing this, CoolSpools Database needs to know the date format in which the data is held in the field.

Note that if you have dates inside character fields (e.g. field ODCDAT in DSPOBJD outfile QADSPOBJ) or if you have some dates in numeric fields which are in one format and some in another (e.g. some are held *MDY and some *YMD), you will need to specify the format of those dates individually on the APYSTYLES parameter. This parameter defines the default assumed format for dates in numeric fields only.

Options are:

***ENVVAR** CoolSpools Database will use the date format defined in the environment variable SL_DBFDATFMT. If that environment variable exists, and is set to one of the values below, that value is used. If the environment variable does not exist, or if it contains anything other than one of the values listed below, *YMD is assumed.
***YMD** CoolSpools Database will assume the date is in YYMMDD format if the field is 6-7 digits long, and in YYYYMMDD format if the field is 8 digits long.

***NONE** CoolSpools Database will not attempt to convert numeric fields that appear to be dates to date. They will be converted as numbers.
***DMY** CoolSpools Database will assume the date is in DDMMYY format if the field is 6-7 digits long, and in DDMMYYYY format if the field is 8 digits long.
***MDY** CoolSpools Database will assume the date is in MMDDYY format if the field is 6-7 digits long, and in MMDDYYYY format if the field is 8 digits long.
***CYMD** CYYMMDD format, where C is 0 for the 20th century and 1 for the 21st
***CDMY** CDDMMYY format, where C is 0 for the 20th century and 1 for the 21st
***CMDY** CMMDDYY format, where C is 0 for the 20th century and 1 for the 21st

AUT

Parameter	**AUT**
Description	**Indicates the public authority to be given to the new file.**
Applies to commands:	**CVTDBFCSV, CVTDBFHTML, CVTDBFXML, CVTDBFTXT, CVTDBFPDF, CVTDBFXL** **CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The **AUT** (Public Authority) parameter allows you to control the public authority given to the stream file when it is created.

Options are:

***R** (Default). Read only
***W** Write only
***X** Execute only
***RW** Read and write
***RX** Read and execute
***WX** Write and execute
***RWX** Read, write and execute (all)
***NONE** No authority
autl_name Specify the name of an authorization list that will control public authority to the file

Note that in previous releases CoolSpools Database set the owner's authority to *RWX by default and set the public authority based on this parameter, but did not set the group authority. This has been corrected in V7R1M0 and the group authority is set to *RWX in all cases.

INHERITAUT

Parameter	**INHERITAUT**
Description	**Indicates whether authorities are inherited from the parent directory in which the stream file is created.**
Applies to commands:	**CVTDBFSTMF (deprecated)**
Dependent on:	**None**
Supports CoolSpools Database variables	**No**

The INHERITAUT (Inherit Authority) parameter allows you to control whether object authorities are inherited from the parent directory in which the stream file is created.

This parameter exists only on CVTDBFSTMF. The format-specific commands always inherit authorities from the parent directory (equivalent to CVTDBFSTMF ... INHERITAUT(*YES)).

Options are:

***NO** Authorities are not inherited from the directory
***YES** Authorities are inherited from the directory

When you specify INHERITAUT(*NO), the object authorities (*OBJEXIST, *OBJMGT, *OBJALTER, and *OBJREF) assigned to the owner, primary group, and *PUBLIC in respect of the stream file being created are copied from the owner, primary group, and public object authorities of the parent directory in which the stream file is created. This occurs even when the new file has a different owner from the parent directory. The new file does not have any private authorities or authorization list. It only has authorities for the owner, primary group, and public. The owner is assigned full data authorities and *PUBLIC is assigned the data authorities specified on the AUT parameter.

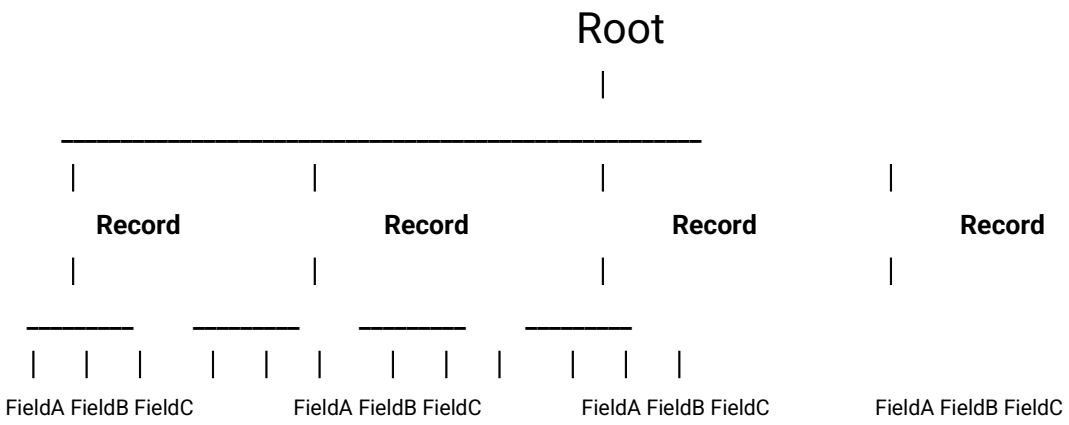
When you specify INHERITAUT(*YES), the object authorities (*OBJEXIST, *OBJMGT, *OBJALTER, and *OBJREF) assigned to the owner, primary group, and *PUBLIC in respect of the stream file being created are copied from the owner, primary group, and public object authorities of the parent directory in which the stream file is created. However, the private authorities (if any) and authorization list (if any) are also copied from the parent directory. If the new file has a different owner than the parent directory and the new file's owner has a private authority in the parent directory, that private authority is not copied from the parent directory. The authority for the owner of the new file is copied from the owner of the parent directory. The owner is assigned full data authorities and *PUBLIC is assigned the data authorities specified on the AUT parameter.

Using Database Maps

Database maps are a feature of CoolSpools Database that gives you the ability to control the structure of Excel spreadsheets and XML documents that it creates.

By default, Excel files created by CoolSpools database consist of rows of data where each row corresponds to a record from the input source (database file or query), preceded, optionally, by a single row of column headings. By using a database map, you can create more complex Excel spreadsheets consisting of multiple row types, such as headings and sub-headings, multiple levels of detail line, sub-totals and totals.

Similarly, by default, XML documents created by CoolSpools database consist of a root element which acts as a container for the document and includes nodes for each record and each field in each record, something like this:



Using a database-to-XML map, you can define more complex XML structures, including nested elements.

A database map specifies the **input** to the conversion, which can be either a database file, a QM Query, a Query/400 query, an SQL statement stored in a source member or an SQL statement associated with the map itself.

A database also defines the structure of the **output** from the conversion, either in terms of Excel rows types and cells, or in terms of XML elements and attributes.

Database-to-Excel maps

To create a database-to-Excel map, use the **CRTDBFXL (Create Database-to-Excel Map)** command to define the map, and its basic properties such as its name, input source and text.

Then use **ADDDBFXLR (Add Database-to-Excel Map Row Group)** command to define row groups in that map. Row groups are sets of related rows that are output to an Excel worksheet as a group, for example a set of heading or total rows.

For each row group, you are required to specify the circumstances in which a new row group is output, for example when a field or database column in the input source changes (e.g. when a new order or customer is encountered).

Finally, use the **ADDDBFXLC (Add Database-to-Excel Map Cell)** command to define the content of cells in those rows. Cell content can either be variable values derived from a field

or database column in the input source or fixed constant text (such as a label or heading), or the cell can be defined as empty.

Database-to-XML maps

To create a database-to-XML map, use the **CRTDBFXML (Create Database-to-XML Map)** command to define the map, and its basic properties such as its name, input source and text.

Then use **ADDDBFXMLE (Add Database-to-XML Map Element)** command to define XML elements in that map. Elements can be nested inside one another to produce a complex nested XML documents and can contain text nodes derived from the input source.

For each element, you are required to specify the circumstances in which a new element is output, for example when a field or database column in the input source changes (e.g. when a new order or customer is encountered).

Finally, use the **ADDDBFXMLA (Add Database-to-XML Map Attribute)** command to define the attributes of elements. The value of attributes will be derived from a field or database column in the input source.

CRTDBFXL and CRTDBFXML commands

To create a database map, use the CRTDBFXL (Create Database-to-Excel Map) or CRTDBFXML (Create Database-to-XML map) command, or press F6 from the WRKDBFXL (Work with Database-To-Excel maps) or WRKDBFXML (Work with Database-To-XML maps) display. These latter two commands are available as options 11 and 12 from the CoolSpools Database menu, respectively.

MAPNAME –Database-to-Excel/Database-to-XML map name

Specify the name of the map you wish to create. Map names can be up to 20 characters long but must otherwise conform to the naming standards for system i objects.

INPUT– Input source

Specifies the source of input to be used. This needs to be specified at this stage so that subsequent operations on the database map, for example adding Excel cells and XML attributes, can be properly validated against the source.

Options are:

***FILE**	When using this map, the input to the CVTDBFXL or CVTDBFXML command will be a database file.
***QRYDFN**	When using this map, the input to the CVTDBFXL or CVTDBFXML command will be a Query/400 query.
***QMQRY**	When using this map, the input to the CVTDBFXL or CVTDBFXML command will be a QM Query object.
***SQLSRC**	When using this map, the input to the CVTDBFXL or CVTDBFXML command will be an SQL statement held in a source file member.
***SQL**	When using this map, the input to the CVTDBFXL or CVTDBFXML command will be an SQL statement

specified on the SQL parameter of the command being run.

FILE– Database file name

When INPUT(*FILE) is specified, the name of the database file (physical, logical or DDM) from which the input, to the conversion will be taken. The file must exist.

Options for the library name are:

***LIBL**	CoolSpools Database will use the library list to locate the file.
***CURLIB**	CoolSpools Database will look for the file in the current library.
library_name	Specify the library in which the file is located.

MBR – Member name

The name of the member in the database file from which input will be taken. The member name is not validated until the database map is used.

Options for the member name are:

***FIRST**	The data will be read from the first member in the file.
***ALL**	The data will be read from all members in the file.
member_name	Specify the name of the member from which data will be read.

FMTSETNAME –Record Format set name

Specifies the name of the record format set.

Record format set names conform to the normal rules for OS/400

objects, except that they may be up to 20 characters in length.

Options are:

***NONE**	No Record Format set name is specified.
Record_Format_set_name	Specify the Record Format set name to be used.

SQL – SQL statement options

SQL statement

The SQL statement to be run and from which the input to the conversion will be taken, where INPUT(*SQL) was specified. The maximum length of the SQL statement is 5000 characters. When the command prompter is used, the prompter imposes a limit of 512 characters. If the SQL statement needs to be longer than 5000 characters, use INPUT(*SQLSRC) and store the SQL statement in a source member, or create a QM query.

Options are:

***NONE**	No SQL statement is supplied. Invalid if INPUT(*SQL) was specified.
SQL_Statement	The SQL statement that forms the input to the conversion. Must be a valid SQL statement at the time the database map is defined. Ignored unless INPUT(*SQL) was specified.

Naming

The naming standard used by the SQL statement.

Options are:

***SYS**	System naming (library/file) is used.
***SQL**	SQL naming (table.collection) is used.

SQLSRC - SQL source options

There is a single value:

***NONE**	No file is specified. Invalid if INPUT(*SQLSRC) was specified.

SQL source file

Specifies the name of the source file from which the SQL source is read when INPUT(*SQLSRC) is specified.

Options are:

file_name	Specify the name of a valid source file. Ignored unless INPUT(*SQLSRC) was specified.

Options for the library name are:

***LIBL**	CoolSpools Database will use the library list to locate the file.
***CURLIB**	CoolSpools Database will look for the file in the current library.
library_name	Specify the library in which the file is located.

SQL source member

Specifies the name of the member in the source file from which the SQL source is read when INPUT(*SQLSRC) is specified.

Options are:

member_name	Specify the name of a valid member in the source file.

Naming

The naming standard used by the SQL source option.

Options are:

***SYS**	System naming (library/file) is used.
***SQL**	SQL naming (table.collection) is used.

QRYDFN - Query/400 object

Specifies the name of a Query/400 query object from which the input is taken when INPUT(*QRYDFN) is specified.

Options are:

***NONE**	No Query/400 query object is specified. Invalid if INPUT(*QRYDFN) was specified.
query_name	Specify a valid Query/400 query name. Ignored unless INPUT(*QRYDFN) was specified.

Options for the library name are:

***LIBL**	CoolSpools Database will use the library list to locate the object.
***CURLIB**	CoolSpools Database will look for the object in the current library.
library_name	Specify the library in which the object is located.

QMQRY – QM Query options

Specifies the details relating to QM query objects when INPUT(*QMQRY) is specified.

There is a single option:

***NONE**	No QM query details are specified. Invalid if INPUT(*QRYDFN) was specified.

QM Query object

Options are:

query_name	Specify a valid QM query name. Ignored unless INPUT(*QRYDFN) was specified.

Options for the library name are:

***LIBL**	CoolSpools Database will use the library list to locate the object.
***CURLIB**	CoolSpools Database will look for the object in the current library.
library_name	Specify the library in which the fie is located.

Query management report form

Options are:

***NONE**	No QM form is used.
***QMQRY**	The QM form to use has the same name as the QM query object specified above.
form_name	Specify a valid QM query report form. Ignored unless INPUT(*QMQRY) was specified.

Options for the library name are:

***LIBL**	CoolSpools Database will use the library list to locate the object.
***CURLIB**	CoolSpools Database will look for the object in the current library.
library_name	Specify the library in which the object is located.

Naming

The naming standard used by the SQL statement in the QM Query.

Options are:

***SYS**	System naming (library/file) is used.
***SQL**	SQL naming (table.collection) is used.

DFTUSEAUT - Default use authority

The default authority to use this database map.

Individual user authorities to the map can be managed by means of the IBM CHGFCNUSG command or CoolSpools's WRKREGFNC. The function controlling authority to use a database map is

ARIADNE_XLS_MAP_nnnnnnnnnn_USE for a database-to-Excel map

and

ARIADNE_XML_MAP_nnnnnnnnnn_USE for a database-to-XML map

where nnnnnnnnn is the internal map identifier, which is displayed by DSPDBFXL and DSPDBFXML.

Options are:

***ALLOWED**	By default, users other than the user creating the map are permitted to use it when converting data to Excel or XML.
***DENIED**	By default, users other than the user creating the map are not permitted to use it when converting data to Excel or XML.

DFTCHGAUT - Default change authority

The default authority to change or delete this database map.

Individual user authorities to the map can be managed by means of the IBM CHGFCNUSG command or CoolSpools' WRKREGFNC. The function controlling authority to use a database map is

ARIADNE_XLS_MAP_nnnnnnnnnn_CHG for a database-to-Excel map

and

ARIADNE_XML_MAP_nnnnnnnnnn_CHG for a database-to-XML map

where nnnnnnnnn is the internal map identifier, which is displayed by DSPDBFXL and DSPDBFXML.

Options are:

***DENIED**	By default, users other than the user creating the map are not permitted to change, delete or manage it.
***ALLOWED**	By default, users other than the user creating the map are permitted to change, delete or manage it.

TEXT 'description'

Specify up to 50 characters of free-format descriptive text to help you identify the database map.

Options are:

***BLANK**	No text is specified.
Text	Specify the text 'description'.

GRPSEQOPT –Row group sequence option

Determines the order in which row groups are output.

Options are:

***MAP**	The order of rows in the Excel file is determined by the way row groups are organized in the database-to-Excel map. Rows will be written to the Excel worksheet based on the hierarchy of row groups in the report-to-Excel map, taking account of parent-child relationships between row groups and the sequence number of child row groups within the parent row group.
***INPUT**	The order of rows in the Excel file is determined by the order of data in the data base file.

CHGDBFXL and CHGDBFXML commands

These commands modify an existing Database-to-Excel or Database-to-XML map respectively.

Command parameters are the same as for CRTDBFXL and CRTDBFXML above, with the exception that all parameters other than MAPNAME default to *SAME (no change).

CPYDBFXL and CPYDBFXML commands

These commands copy existing Database-to-Excel or Database-to-XML map respectively.

Command parameters are the same as for CRTDBFXL and CRTDBFXML above, with the exception that the from- and to-map names must be specified on the FROMMAP and parameter TOMAP parameters respectively. All other parameters default to *SAME (no change).

DLTDBFXL and DLTDBFXML commands

These commands delete an existing Database-to-Excel or Database-to-XML map respectively.

DSPDBFXL and DSPDBFXML commands

These commands display the details of a Database-to-Excel or Database-to-XML map respectively.

RNMDBFXL and RNMDBFXML commands

These commands rename an existing Database-to-Excel or Database-to-XML map respectively.

WRKDBFXL and WRKDBFXML commands

These commands let you work with existing Database-to-Excel or Database-to-XML maps respectively.

ADDDBFXLR (Add Database-to-Excel Map Row Group) command

The ADDDBFXLR (Add Database-to-Excel Map Row Group) command adds a row group to a Database-to-Excel map. A row group is a set of related rows that are output together to an Excel worksheet, for example a set of headings or totals.

MAPNAME –Database-to-Excel Map Name

Specify the name of the existing database-to-Excel map to which you wish to add the row group.

ROWGRPNAME –Row group name

Specify the name of the row group you wish to add.

Row group names conform to the normal rules for OS/400 object names except that they can be up to 20 characters long.

PARENT– Parent row group name

Specify the name of the parent row group, if any. Use of this parameter allows the creating of a nested tree of row groups.

This becomes important primarily in relation to determining when a new row group should be created (see NEWGRPOPT parameter below). When the condition specified for a new row group to be triggered is met, all row groups that are defined as children of the row group also become eligible to be output afresh.

For example, if you are creating an Excel spreadsheet from a classic order database consisting of order header records and associated order detail records, you might define an ORDER_HEADER row group where a new row group is created every time the order number in the input data changes and an ORDER_LINE row group where a new row group is created every time the order line number in the input data changes. By making ORDER_HEADER the parent of ORDER_LINE, you ensure that a new ORDER_LINE row group is output every time a new order starts, otherwise two successive orders consisting of a single line number 1 would generate only a single ORDER_LINE group, because the order line number would not have changed. (This is admittedly a slightly artificial example, as it would, in practice, be better to specify that a new ORDER_LINE row group be created for every record in the input data, or when either the order number or the order line number changed).

Options are:

***NONE**	The row group will be created in the root of the database-to-Excel map and will have no parent row group.
Parent_name	Specify the name of the parent row group.

SEQNBR – Sequence number

A number determining the order in which row groups are output. Row groups are output in the order of their sequence number within their parent row group.

Options are:

***NEXT**	The next highest available sequence number in the parent row group is assigned.

seq_number Specify the sequence number.

TEXT 'description'

Specify up to 50 characters of free-format descriptive text to help you identify the row group.

Options are:

***BLANK** No text is specified.

Text Specify the text 'description'.

NEWGRPOPT - New row group on change of

Specifies the conditions under which a new row group of this type, and all child row groups, will be reinitialized and output afresh.

Element 1 - Field /column name

Options are:

***RCD** A row group of this type is output for every record in the input source.

***NEVER** A new row group is never created. Only a single row group of this type will be created within each parent row group. This can be useful for defining headings which need to be output once at the start of a group but not subsequently.

***SELECT** You will be prompted to select one or more fields or database columns from the input source. A new row group of this type will be created whenever any one of those fields or database columns changes.

Field/column Specify the name of between 1 and 10 database fields or columns. A new row group of this type will be created whenever any one of those fields or database columns changes.

Options for the record format name are:

***ONLY** There is only a single record format in the input file. Specify this option if the input source is not a file.

record_name Specify the name of the record format in which the field or database column specified can be found.

Element 2 - Comparison

Options are:

***EQ** Equal to.

A single value must be specified on the Value element below. The rule is true if the field or cell value is equal to this value.

***GT** Greater than.

A single value must be specified on the Value element below. The rule is true if the field or cell value is greater than this value.

***LT** Less than.

A single value must be specified on the Value element below. The rule is true if the field or cell value is less than this value.

***GE** Greater than or equal to.

A single value must be specified on the Value element below. The rule is true if the field or cell value is greater than or equal to this value.

***LE** Less than or equal to.

A single value must be specified on the Value element below. The this value.

***NE** Not equal to.

A single value must be specified on the Value element below. The rule is true if the field or cell value is not equal to this value.

***CT** Contains.

A single value must be specified on the Value element below, and it is interpreted as a text string. The rule is true if the field or cell value contains the text string specified.

***NC** Does not contain.

A single value must be specified on the Value element below, and it is interpreted as a text string. The rule is true if the field or cell value does not contain the text string specified.

***BW** Begins with.

A single value must be specified on the Value element below, and it is interpreted as a text string. The rule is true if the field or cell value begins with the text string specified.

***EW** Ends with.

A single value must be specified on the Value element below, and it is interpreted as a text string. The rule is true if the field or cell value ends with the text string specified.

Element 3 - Value

Specify the value to compare to.

GRPDTAOPT - Row group data option

Specifies how the data within the group is to be shown. For example, if you have a group of records (see below, sorted by SALES), you may wish to show the details, contained in the first record, or the details contained in the last record. This can only be used, if you have specified a new group rule (NEWGRPOPT) of *NEVER, effectively showing the group as a break/summary.

MONTH	SALES
Jan	50,000
May	52,000
Feb	56,000
Apr	57,000
Mar	63,000

Options are:

*LAST	The data from the last entry of the group, would be shown (example above would show Mar 63,000).
*FIRST	The data from the first entry of the group, would be shown (example above would show Jan 50,000).

BFRACTION - Action before row group

Specifies the action to be carried before the row group is added.

Element 1 - Action

Options are:

*NEWSHEET	Start a new row group with a new worksheet, with the name specified in element 2.
*NEWPAGE	Start a new row group with a page break before it.
*NONE	No action is taken.

Element 2 - Sheet Name

If the action on the previous element is *NEWSHEET, then you specify the name of the worksheet here.

AFTACTION - Action after row group

Specifies the action to be carried after the row group is added.

Element 1 - Action

Options are:

*NONE	No action is taken.
*NEWPAGE	End a new row group with a page break before it.
*NEWSHEET	End a new row group with a new worksheet, with the name specified in element 2.

Element 2 - Sheet Name

If the action on the previous element is *NEWSHEET, then you specify the name of the worksheet here.

ADDDBFXLC (Add Database-to-Excel Map Cell) command

The ADDDBFXLC (Add Database-to-Excel Map Cell) command adds a cell to a Database-to-Excel map row group.

MAPNAME –Database-to-Excel Map Name

Specify the name of the existing database-to-Excel map to which you wish to add the cell.

ROWGRPNAME –Row group name

Specify the name of the row group to which you wish to add the cell.

Row group names conform to the normal rules for OS/400 object names except that they can be up to 20 characters long.

ROWNBR – Row number

Specify the row number within the row group. For example, if the row group consists of 4 rows in the Excel file, specifying ROWNBR(4) here adds this cell to the fourth row of the group. Note that this does ***not*** correspond to row 4 in the Excel file.

COLUMN– Column letter

Specify the column letter for the cell. For example, column A is the first column in the worksheet and column AA is the 27th.

CONTENT – Cell content

Specify the type of content to be assigned to the cell.

Options are:

***COLUMN**	The cell will be populated with data taken from a database field or column from the input source (specified on CELLCOLUMN below).
***TEXT**	The cell will be populated with constant text (specified on CELLTEXT below).
***EMPTY**	The cell will be empty (null). This can be useful for leaving spaces in a row of data while ensuring that formatting can be applied to those cells which would not be the case if they were simply undefined).
***IMAGE**	Specifies an image file to be included in the Excel file. Image types currently supported are: • JPEGs • GIFs • PNGs
***FORMULA**	Specifies that the cell will contain a formula not text.

CELLCOLUMN – Database field or column

Specifies the name of a database field or column from the input file the value of which will be used to populate the cell, when CONTENT(*COLUMN) is used.

Options are:

***SELECT**	You will be prompted to select the field or database column from the input source.
Field/column	Specify the name of the database fields or column.

Options for the record format name are:

***ONLY**	There is only a single record format in the input file. Specify this option if the input source is not a file.
record_format	Specify the name of the record format in which the field or database column specified can be found.

Options for the Data Type:

***COLUMN**	The cell will be populated with data taken from a database field or column from the input source.
***ALPHA**	The cell will be defined as Alpha.
***NUMERIC**	The cell will be defined as Numeric.
***DATE**	The cell will be defined as a Date.

CELLTEXT – Cell text

Specifies fixed constant text (for example, a label or heading) to be output to the cell, when CONTENT(*TEXT) is used.

Note that CoolSpools Database variables can be specified as part of the text.

IMGSRC – Source of image

How CoolSpools should locate the image file to include.

Options are :

***PATH**	The next element specifies the path to an image file to be included.
***COLUMN**	The next field specifies the name of a column from the input data which contains the path to an image file to be included.
***EXITPGM**	The next field specifies the name of an exit program that will be called to provide the path to an image file to be included.

IMGPATH– Path of the image

Specifies the path name of the image file to include.

Options are :

pathname	The pathname of the image is specified.

***NONE**	No pathname for the image is specified.

IMGSIZE– Image size option or unit

Determines the size of the image.

Element 1 - Option / unit :

Options are :

***IMAGE**	The size of the image is calculated from the image properties (width and height in pixels and resolution).
***SCALE**	The next two elements specify scaling factors by which the image size implied by the image properties will be multiplied.
***CELLREF**	The next two elements specify an Excel column letter and a row number where the bottom right corner of the image will be located.
***INCH**	The next two elements specify absolute coordinates on screen where the bottom right corner of the image will be located, measured in inches.
***CM**	The next two elements specify absolute coordinates on screen where the bottom right corner of the image will be located, measured in centimeters.
***MM**	The next two elements specify absolute coordinates on screen where the bottom right corner of the image will be located, measured in millimeters.
***PTS**	The next two elements specify absolute coordinates on screen where the bottom right corner of the image will be located, measure in points.

Element 2 - X dimension or scaling :

If the Option/unit is ***CELLREF**, you have to specify an Excel row and cell **(parameter IMGTOCELL)**, where the bottom right corner of the image will be located.

If the Option/unit is ***SCALE**, specify the horizontal scaling factor (1.00 = no scaling, 0.5 = half size, etc.)

If the Option/unit is neither *CELLREF or *SCALE, specify the horizontal coordinate of the bottom right corner of the image, measured in the unit defined on the Option/Unit element.

Options are :

integer	Specify X dimension or scaling value.

Element 3 - Y dimension or scaling :

This element has the same options as X dimension or scaling.

IMGRTT - Image rotation (degrees)

Specify how many degrees clockwise the image is to be rotated.

Values are:

0 The image will not be rotated.

integer Specify the angle through which the image is rotated, in degrees (0-360).

IMGCOLUMN - Column containing path to image

When the option/unit specified, for the IMGSIZE parameter, is *CELLREF, the bottom right row/cell of the image is required.

Element 1 - Extend to row number :

Options are :

***ROWNBR** Specify the bottom right row number.

Element 2 - Extend to cell number :

Options are :

***COLUMN** Specify the bottom right column letter.

IMGPGM – Exit program providing path to

Exit program containing the image path.

Element 1 - Qualified program name.

Options are :

***NONE** No exit program program is specified.

Qualified program name

Specify the name of the exit program.

Options for the library name are:

***LIBL** CoolSpools Database will use the library list to locate the file. .

***CURLIB** CoolSpools Database will look for the file in the current library.

library_name

Specify the library in which the file is located.

Element 2 - Parameter format name.

Enter the value SL_EPC01

IMGTOCELL – Image extends to cell reference

When the option/unit specified, for the IMGSIZE parameter, is *CELLREF, the bottom right row/cell of the image is required.

Element 1 - Extend to row number.

***ROWNBR** Specify the bottom right row number.

Element 2 - Extend to column letter.

***COLUMN** Specify the bottom right column letter.

FORMULA – Formula

Specified that the cell will contain a formula and not text. The formula must contain a string enclosed in the special marker characters ==XL().

For example,

==XL(put_your_formula_here)

There's an additional complication, in that often you will want the formula to refer to the values of other columns on the same row, which begs the question how the formula can "know" what row it's on.

The best way to solve this problem in our experience is to use the Excel INDIRECT function to build a "soft" reference to the current row.

For example, this text would tell CoolSpools to construct a formula which multiplies column F by columns E:

==XL(INDIRECT("F"&ROW())*INDIRECT("E"&ROW())) in other words, on row 2, this is equivalent to "F2 * E2".

MRGCELLS – Merge cells

Specifies if the cell should be merged with adjoining cells.

There is a single option:

***NONE** The cell is not merged with any adjoining cells.

The other options for this parameter specify the extent of the block of merged cells of which this cell forms a part. The block starts with the current cell and extends as far as the row and column specified below.

Merge to row number

The row number to which the block of merged cells extends. The row number specified cannot be lower than the row number specified on the ROWNBR parameter. In other words, cells cannot be merged with cells on earlier rows, only with the cells on the same or later rows.

Options are:

***ROWNBR**	The merged cell block does not extend beyond the current row.
Row_number	The row number within the row group to which the merged cell block extends.

Merge to column letter

The column letter to which the block of merged cells extends. The column letter specified cannot be to the left of the column letter specified on the COLUMN parameter. In other words, cells cannot be merged with cells to the left, only with the cells to the right.

Options are:

***COLUMN**	The merged cell block relates only to the current column.
Column_letter	The column letter to which the merged cell block extends.

ADDDBFXMLE (Add Database-to-XML Map Element) command

The ADDDBFXMLE (Add Database-to-XML Map Element) command adds an element to a Database-to-XML map. A database-to-XML element controls the creation of nodes within an XML document.

MAPNAME –Database-to-XML Map Name

Specify the name of the existing database-to-XML map to which you wish to add the row group.

ELEMENT–Element name

Specify the name of the XML element you wish to add.

Database-to-XML element names can be up to 50 characters in length and, like all things XML, are case-sensitive. They must conform to the rules for XML names.

PARENT– Parent element name

Specify the name of the parent element, if any. Use of this parameter allows the creating of a nested tree of XML elements.

Options are:

***NONE**	The element will be the document root element. Only one root element can exist for each database-to-XML map.
Parent_name	Specify the name of the parent element, which could be the root or a descendant of the root.

SEQNBR – Sequence number

A number determining the order in which elements are output within their parent elements. Elements are output in the order of their sequence number within their parent element.

Options for the library name are:

***NEXT**	The next highest available sequence number in the parent element is assigned.
seq_number	Specify the sequence number.

SOURCE - Source of Element Value

Specify whether the element value is derived from a constant value,

or from the column name from the Input File referenced in the Database-to-XML Map.

Options are:

***NONE**	No element value is derived.
***COLUMN**	The element value is derived from the Database field or column.
***CONSTANT**	The element value is derived entered Constant value.

CONSTANT - Constant Value

User-defined column name for the attribute.

Options are:

***NONE**	The column name of the attribute is left blank..
Constant value	User-defined column name.

COLUMN- Database field or column

Specify the name of a database field or column in the input source from which the value of the text node for this element will be derived.

Options are:

***NONE**	The element has no text node.
***SELECT**	You will be prompted to select the field or database column from the input source.
Field/column	Specify the name of the database field or column

Options for the record format name are:

***ONLY**	There is only a single record format in the input file. Specify this option if the input source is not a file.
record_name	Specify the name of the record format in which the field or database column specified can be found.

DATATYPE - Data Type

Specify the data type of the XML attribute.

Element 1 - Data type

Options are:

***SOURCE**	The data type of the specified database field or column is used.
***STRING**	The attribute is straight forward text, rather than numbers.
***DATE**	The attribute contains a date.
***INTEGER**	The attribute contains a number.
***TIME**	The attribute contains a time.
***DATETIME**	The attribute contains a combined date and time.
***BOOLEAN**	The attribute is true or false.
***FLOAT**	The attribute is a real number that can contain a fractional

part.

***DECIMAL** The attribute is decimal number.

TEXT – Text 'description'

Specify up to 50 characters of free-format descriptive text to help you identify the element.

Options are:

***BLANK** No text is specified.

Text Specify the text 'description'.

NEWELMOPT - New element on change of

Specifies the conditions under which a new element of this type, and all child elements, will be reinitialized and output afresh.

Options are:

***RCD** A new node of this type is output for every record in the input source.

***NEVER** A new node of this type is never created. This is valid only for the root and is the only valid option for the root.

***SELECT** You will be prompted to select one or more fields or database columns from the input source. A new XML node of this type will be created whenever any one of those fields or database columns changes.

Field/column Specify the name of between 1 and 10 database fields or columns. A new XML node of this type will be created whenever any one of those fields or database columns changes.

Options for the record format name are:

***ONLY** There is only a single record format in the input file. Specify this option if the input source is not a file.

record_name Specify the name of the record format in which the field or database column specified can be found.

XMLNAMESPC - XML namespace options

Specifies namespace-related options for XML output.

XML namespaces are used for providing uniquely named elements and attributes in an XML document. An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a namespace, ambiguity between identically named elements or attributes can be resolved.

A simple example would be to consider an XML instance that contained references to a customer and an ordered product. Both the customer element and the product element could have a child element named id.

References to the id. element would therefore be ambiguous; placing them in different namespaces would remove the ambiguity.

Element 1 - Namespace URI

A namespace name is a uniform resource identifier (URI).

Options are:

***PARENT**	Include the Namespace URI specified for the parent of this element.
***ROOT**	Include the Namespace URI specified for the root of this element.
***NONE**	No Namespace URI is defined for the XML document.
Namespace URI	Specify the Namespace URI.

Element 2 - Namespace prefix

The Namespace prefix forms part of the qualified name, and must be associated with a namespace URI reference in a namespace declaration.

Options are:

***PARENT**	Include the Namespace prefix specified for the parent of this element.
***ROOT**	Include the Namespace prefix specified for the root element of the document.
***NONE**	No namespace prefix is defined for the XML document.
Namespace prefix	Specify the Namespace prefix.

ADDDBFXMLA (Add Database-to-XML Map Attribute) command

The ADDDBFXMLA (Add Database-to-XML Map Attribute) command adds an attribute to an element in a Database-to-XML map.

MAPNAME –Database-to-XML Map Name

Specify the name of the existing database-to-XML map to which you wish to add the element.

ELEMENT–Element name

Specify the name of the existing XML element to which you wish to add an attribute.

Database-to-XML element names can be up to 50 characters in length and, like all things XML, are case-sensitive. They must conform to the rules for XML names.

ATTRIBUTE–Attribute name

Specify the name of the XML attribute you wish to add to the element.

Database-to-XML attribute names can be up to 50 characters in length and, like all things XML, are case-sensitive. They must conform to the rules for XML names.

SEQNBR – Sequence number

A number determining the order in which attributes are output on their associated elements. Attributes are output in the order of their sequence number specified.

Options for the sequence number are:

***NEXT**	The next highest available sequence number in the element is assigned.
seq_number	Specify the sequence number.

SOURCE - Source of Element Value

Specify whether the element value is derived from a constant value,

or from the column name from the Input File referenced in the Database-to-XML Map.

Options are:

***NONE**	No element value is derived.
***COLUMN**	The element value is derived from the Database field or column.
***CONSTANT**	The element value is derived entered Constant value.

CONSTANT - Constant Value

User-defined column name for the attribute.

Options are:

***NONE**	The column name of the attribute is left blank..
Constant value	User-defined column name.

COLUMN- Database field or column

Specify the name of a database field or column in the input source from which the value of the attribute for this element will be derived.

Options are:

***SELECT**	You will be prompted to select the field or database column from the input source.
Field/column	Specify the name of the database field or column

Options for the record format name are:

***ONLY**	There is only a single record format in the input file. Specify this option if the input source is not a file.
record_name	Specify the name of the record format in which the field or database column specified can be found.

DATATYPE - Data Type

Specify the data type of the XML attribute.

Element 1 - Data type

Options are:

***SOURCE**	The data type of the specified database field or column is used.
***STRING**	The attribute is straight forward text, rather than numbers.
***DATE**	The attribute contains a date.
***INTEGER**	The attribute contains a number.
***TIME**	The attribute contains a time.
***DATETIME**	The attribute contains a combined date and time.
***BOOLEAN**	The attribute is true or false.
***FLOAT**	The attribute is a real number that can contain a fractional part.
***DECIMAL**	The attribute is decimal number.

CVTXLDBF (Convert Excel to Database) Command

The CVTXLDBF command lets you extract the contents of cells in an Excel spreadsheet. The data is written in a standard database “outfile” format. Your applications can easily read this outfile and process its contents, for example to import the data into your own database.

This provides a convenient means of getting data stored in an Excel spreadsheet into an AS/400 database without the need for a PC. Thus, processing can be carried out automatically, and without user intervention, perhaps as part of your overnight batch job suite. For example, users might enter information into a spreadsheet during the day which is subsequently picked up and processed by your program after close of business. Your program might use this command to extract the data from the spreadsheet and write it to your AS/400 database files or perform other processing on it.

Note that the outfile contains null-capable fields and ILE RPG programs will need to have the header specification (H-spec) **ALWNULL(*INPUTONLY)** or **ALWNULL(*USRCTL)** defined, or specify the equivalent value on your compile command, in order to process the file.

See below for a discussion of the format in which the data extracted from your spreadsheet is stored.

Be sure to recalculate the values of formulas and then save and close your Excel file before running this command against it.

The command parameters are as follows.

FROMSTMF – Excel file to convert

Specifies the path name of the Excel file from which the cell contents will be extracted.

FROMSHEETS –Worksheet(s) to convert

Specifies one or more worksheets in the Excel spreadsheet from which cell contents will be extracted.

The default is the special value:

***ALL**	Data will be extracted from all worksheets in the workbook.

TOFILE – File to receive output

The name of the database file which will receive the output.

If the file does not exist, it will be created.

If the file does exist, it must be a file in the correct format (for example, a file previously created by this command). If the file is not in the correct format, an error will occur.

The default is the special value:

***FROMSTMF**	The name of the file will be derived from the name of the spreadsheet specified on the FROMSTMF parameter by removing any extension from the file name part of the path name and remainder of the file name up to a maximum of 10 characters.

Options for the library name are:

***LIBL**	CoolSpools Database will use the library list to locate an existing file. If no file is found, a new file will be created in the current library.
***CURLIB**	CoolSpools Database will look for an existing file in the current library, or create a new file in the current library.
library_name	Specify the library in which an existing file is located or in which to create a new file.

TOMBR – Member to receive output

Name

The name of the member in the database file which will receive the output.

If the member does not exist, it will be added to the file.

If the file does exist, data in that member will be replaced or the new data will be appended to that member depending on the setting of the “Replace or add records” option below.

Options are:

***FROMSTMF**	The name of the member will be derived from the name of the spreadsheet specified on the FROMSTMF parameter by removing any extension from the file name part of the path name and remainder of the file name up to a maximum of 10 characters.
***FIRST**	The data will be written to the first member in the file. If there are no members in the file, a new member with the same name as the file will be added and used.
member_name	Specify the name of the member to receive the data.

Replace or add records

Determines whether any existing data in the file is replaced or whether the new data is appended to any existing data.

Options are:

***REPLACE**	Any existing data in the file is replaced.
***ADD**	The new data is appended to any existing data.

TOCCSID – CCSID to convert to

Specifies the CCSID (character encoding) to which the cell contents will be converted. Data in Excel files is typically stored in ASCII or unicode and you will normally wish to convert that data to an appropriate EBCDIC CCSID for processing on the system i.

Options are:

***JOB**	The CCSID of the current job.
***SYSVAL**	The CCSID indicated by the QCCSID system value.
***USER**	The CCSID associated with the current user’s user profile.
CCSID	Specify a CCSID to use.

CVTDATES – If date formatting, cvt to

Controls how cells in the Excel file which look like dates are converted.

Excel does not have any concept of a date cell. Dates in Excel spreadsheets are held as numbers (specifically, a count of days since January 1st, 1900, but with 1900 treated, erroneously, as a leap year). You can ask Excel to display a cell as a date by applying date-related formatting, such as “mm/dd/yyyy”.

Single option:

***NUMBER**	Cells are not checked to see if they are formatted as dates and are output as a number. This number can be converted to a date in your program by treating it as a count of days since 1899-12-30.

Other options:

Where a numeric cell is being converted, its formatting will be checked. Where that formatting is one of the Excel built-in date-related format options, or if a custom format string is applied which contains any of the “yy”, “mm” or “dd” options, the cell will be treated as a date.

Date Format

The format to which the date is converted.

***ISO**	YYYY-MM-DD
***USA**	MM/DD/YYYY
***EUR**	DD.MM.YYYY
***YMD**	YYMMDD
***MDY**	MMDDYY
***DMY**	YYMMDD
***SYSVAL**	The date format implied by system value QDATFMT is used.
***JOB**	The date format implied by job attribute DATFMT is used.

Date Separator

The separator character used.

***DATFMT**	The separator implied by the date format is used. For *ISO, this is a hyphen (–). For *USA, this is a slash (/). For *EUR, this is a period (.) In all other cases, the date separator implied by the job attributes is used.
***SYSVAL**	The date separator implied by system value QDATSEP is used.

***JOB**	The date separator implied by job attribute DATSEP is used.
***NONE**	No separator is used.
***SLASH**	/
***HYPHEN**	-
***PERIOD**	.
***COMMA**	,
***COLON**	:
***BLANK**	A blank (space)
sep_char	Specify a separator character

BLANKS – Output blank cells

Determines whether anything is written to the output for cells that contain blanks (Excel BLANK and/or MULBLANK records). These records typically denote empty cells that contain no data and are not the same as cells that contain labels that consist of all blank (space) characters).

Options are:

***NO**	Cells that are defined but empty (represented by contain Excel BLANK or MULBLANK records) are ignored and not written to the output.
***YES**	Cells that are defined but empty (represented by contain Excel BLANK or MULBLANK records) are not ignored and written to the output.

ROUNDING – Floating point rounding

Specifies the number of decimal places to which floating point values are rounded.

Numeric values in cells in an Excel spreadsheet are typically held either as integer values or as floating point values. These are written to the output file in packed decimal format as that format is usually more convenient for system i applications to process than floating point. However, the process of converting floating point numbers to packed decimal can result in a very small loss of fidelity, which is normally of no consequence and can be ignored.

For example, because of floating point conversion errors, a number which should be an integer value 123, might be written to the file as 123.000000001.

In order to avoid this issue, numbers can be rounded to a set number of decimal places.

Options are:

5	Floating point values are rounded to 5 decimal places before being written to the file.
decimal_places	Specify between 0 and 9 decimal places to which floating point values will be rounded.

CVTXLDBF output format

The format of files created by CVTDBFXL is described below.

These files are organized in such a way that data from any Excel spreadsheet can be written to them and processed by application programs, irrespective of the layout of the original spreadsheet. Inevitably, to achieve this level of flexibility, the design of this generic outfile has involved a degree of compromise. The structure is denormalized in order to allow all data to be written to a single file and data is consequently not held as efficiently as it might be. You should be aware, therefore, that the outfile created when extracting information from a large spreadsheet could occupy a significant amount of storage.

The basic organization of the file is that each record represents a single cell in the spreadsheet, but only cells that contain data are represented in the outfile (whether cells which contain empty values are represented is controlled by the BLANKS parameter – see above).

The file that is created is a multi-member uniquely keyed physical with a record format called SL_XLSOUTR. Each record in the file is uniquely identified by a combination of sheet number, row number and column number.

The layout of records in the file is as follows:

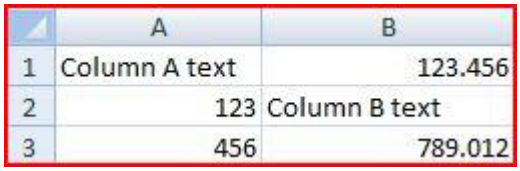
Field name	Format	DDS definition	Description	Comments
SHEETNBR	BINARY(4)	9B 0	Sheet number	Identifies the worksheet in the Excel workbook from which the cell data was extracted. Worksheets are numbered from 1 starting with the first worksheet in the workbook.
SHEETNAME	CHAR(32)	32A	Sheet name	Identifies the worksheet in the Excel workbook from which the cell data was extracted. The worksheet name repeated in each record for reasons of convenience in order to make it possible to write all data to a single outfile.
ROWNBR	BINARY(4)	9B 0	Row number	Identifies the row in the worksheet from which the cell data was extracted. Note that rows are numbered from 1 as per the Excel user interface.
COLUMNNBR	BINARY(4)	9 B 0	Column number	Identifies the column in the worksheet from which the cell data was extracted. Note that columns are numbered from 1, equivalent to column A in the Excel user interface. Columns numbers rather than letters are used here to denote cells in the worksheet

				in order to ensure proper sequencing. For example, column Z must precede column AA. In this file, column Z is numbered 26 and column AA is numbered 27 in order to sort records correctly.
COLUMNREF	CHAR(2)	2A	Column reference	The column reference of the column identified by COLUMNNBR above, in the form in which it is usually represented, i.e. a letter A-Z or pair of letters AA-ZZ.
DATATYPE	CHAR(1)	1A	Data type	The data type of the cell value. The value of this field determines which (if any) of the fields NUMBER and TEXT contains the actual cell value. Options are: N = Numeric. The cell contains a numeric value and that value is stored in the field NUMBER below. A = Alphanumeric. The cell contains a text label and that label value is stored in the field TEXT below. D = Date. The cell has been interpreted as a date (day count) because it has date-related formatting and the converted date value is stored in the field TEXT below. B = Blank. The cell is defined but empty. Neither NUMBER not TEXT will contain a value.
NUMBER	PACKED(31,9)	31P 9	Numeric value	If the Excel cell contains a floating point or integer value, that numeric value is returned in this field, and DATATYPE contains an “N”, otherwise this field is NULL.
TEXTCCSID	BINARY(4)	9B 0	CCSID of text value	If the Excel cell contains a text label, this field holds the CCSID to which that text value was converted before being stored in TEXT below, otherwise this field is NULL.
TEXT	VARLEN(1024)	1024A	Text value	If the Excel cell contains a text label, that text value is

		(varying)		returned in this field, and DATATYPE contains an “A”. If the Excel cell contains a number formatted as a date and you requested that dates be output by specifying something other than CVTDATES(*NUMBER) the converted date value is returned in this field, and DATATYPE contains a “D”. Otherwise this field is NULL.

Example:

Imagine a spreadsheet called **example.xls** containing a single worksheet called Sheet1 with the following contents:



	A	B
1	Column A text	123.456
2	123	Column B text
3	456	789.012

Running the following command:

CVTXLDBF FROMSTMF(example.xls)

Creates a file called EXAMPLE in the current library which contains the following data:

Sheet Number	Sheet Name	Row Number	Column Number	Column Ref	Data Type	Numeric Value	CSSID of Text Data	Text Value
1	Sheet1	1	1	A	A		37	Column A text
1	Sheet1	1	2	B	N	123.456		
1	Sheet1	2	1	A	N	123		
1	Sheet1	2	2	B	A		37	Column B text
1	Sheet1	3	1	A	N	456		
1	Sheet1	3	2	B	N	789.012		

The value of the TEXTCCSID field will be dependent on the job CCSID attribute.

IMPXLDBF (Import Excel to Database) command

The IMPXLDBF command lets you extract the contents of cells in an Excel spreadsheet. The data is written to either an existing file, a new file based on an existing file or a new file based on the spreadsheet.
This provides a convenient means of getting data stored in an Excel spreadsheet into an AS/400 database without the need for a PC. Thus, processing can be carried out automatically, and without user intervention, perhaps as part of your overnight batch job suite. For example, users might enter information into a spreadsheet during the day which is subsequently picked up and processed by your program after close of business. Your program might use this command to extract the data from the spreadsheet and write it to your AS/400 database files or perform other processing on it.
Be sure to recalculate the values of formulas and then save and close your Excel file before running this command against it.
The command parameters are as follows:

FROMSTMF – Excel file to convert

Specifies the path name of the Excel file from which the cell contents will be extracted.

FROMSHEETS – Worksheet(s) to convert

Specifies one or more worksheets in the Excel spreadsheet from which cell contents will be extracted.

The default is the special value:

***FIRST**	This is the default value. Only the first worksheet will be extracted.
***ALL**	Data will be extracted from all worksheets in the workbook.

TOFILE – File to receive output

The name of the database file which will receive the output. If the file does not exist, it will be created. If the file does exist, it must be a file in the correct format (for example, a file previously created by this command). If the file is not in the correct format, an error will occur.

The default is the special value:

***FROMSTMF**

The name of the file will be derived from the name of the spreadsheet specified on the FROMSTMF parameter by removing any extension from the file name part of the path name and remainder of the file name up to a maximum of 10 characters.

Options for the library name are:

***LIBL** CoolSpools Database will use the library list to locate an existing file. If no file is found, a new file will be created in the current library.

***CURLIB** CoolSpools Database will look for an existing file in the current library, or create a new file in the current library.

library_name

Specify the library in which an existing file is located or in which to create a new file.

TOMBR – Member to receive output

Specifies the database file member name that recevies the copied records.

Name The name of the member in the database file which will receive the output.
If the member does not exist, it will be added to the file.
If the file does exist, data in that member will be replaced or the new data will be appended to that member depending on the setting of the “Replace or add records” option below.

Options are:

***FROMSTMF**

The name of the member will be derived from the name of the spreadsheet specified on the FROMSTMF parameter by removing any extension from the file name part of the path name and remainder of the file name up to a maximum of 10 characters.

***FIRST** The data will be written to the first member in the file.
If there are no members in the file, a new member with the same name as the file will be added and used.

member_name

Specify the name of the member to receive the data.

Replace or add records

Determines whether any existing data in the file is replaced or whether the new data is appended to any existing data.

Options are:

***REPLACE** Any existing data in the file is replaced.

***ADD** The new data is appended to any existing data.

***NONE** This parameter is used when using CRTFILE(*YES).

CRTFILE – Create file

Specifies, when this command is used to copy to a physical or a logical file, whether a physical file is created to receive the data, if the specifed to-file does not exist. If the to-file

is a Distributed Data Management (DDM) file that identifies a remote file that does not exist, the to-file is created on the target system.

Options are:

***NO** The to-file must exist when this command is started. A physical file is not created to receive the data.

***YES** If the to-file does not exist, a physical file is created with the name specified on the To file **(TOFILE)** parameter. If the from-file is an SQL table, view, or index, that contains a user defined type, datalink, or LOB field type, the physical file created will be an SQL table. In all other instances the to-file created will be a database physical file that is not an SQL table. In addition to the normal copy operation validity checks, the following special conditions must all be true for the copy operation to create a to-file:

- The from-file must be either a physical or logical file.
- A library name must be specified on the To file **(TOFILE)** parameter. The default value, *LIBL, is not allowed.
- There cannot be an override to a different file or library name. The values specified on this command for the to-file must be used.
- The user running this command must be authorized to add the file to the to-file library, and must also have operational authority to the Create Physical File (CRTPF) command.
- A single record format must be used in the from-file. If the from-file is a logical file with multiple formats, the **Record format of logical file (RCDFMT)** parameter must specify a record format name.

COLNAMING – Column naming

Specifies, when this command is used to import the data from an excel spreadsheet, how the fields are labelled.

Options are:

***HEADER** The header row of the spreadsheet is used, to label each field within the database file.

***TABLE** The fieldnames are used, from the file that was referenced in the parameter, REFFILE.

***AUTONAME**

The fields within the database are automatically labelled, regardless of the spreadsheet header, COL0000000, COL0000001 etc.

NAMING – Naming convention

Specifies the naming convention used for objects in SQL statements.

Options are:

<u>*SQL</u> The SQL naming convention (collection-name.table-name) is used.

***SYS** The system naming convention (library-name/file-name) is used.

MAXMBRS – Maximum members

Specifies the maximum number of members that the physical file can contain.

Please note, if the spreadsheet you are importing has multiple worksheets, you have to specify enough members to accommodate them, otherwise you will get an error.

Options are:

<u>1</u> Only one member can be contained in the physical file.

***NOMAX** The number of members that can be contained in the file is the system maximum of 32,767 members.

1-32767 Specify the maximum number of members that can be contained in the physical file.

SIZE – Member size

Specifies the initial number of records in each member of the file, the number of records for each increment added to the member, and the number of times the increment is automatically applied. The number of records for each file member is specified as the number of records that can be placed in it (this number includes any deleted records).

When the maximum number of records has been reached, a message (stating that the member is full) is sent to the system operator, giving the choice of ending the request or extending the member's number of records. When the operator chooses to extend the member, the maximum number of records for the member will be increased by the increment number of records times the number of increments specified. However, this increase in maximum number of records will not always have the same effect on the actual member size (in bytes).

Initial number of records

<u>10000</u> Initially, up to 10000 records can be written to each member of the file.

1-2147483646
Specify the number of records that can be written to each member of the file before the member is automatically extended.

***NOMAX** The number of records that can be added to each member of the file is not limited by the user. The maximum number of records for each member is determined by the system. If *NOMAX is specified, *NO must be specified for the **Allocate storage (ALLOCATE)** parameter.

Increment number of records

Specify the number of additional records that are added to the member when the number of records in the member will exceed the initial number of records, or will exceed the current increment's number of records.

1000 Initially, up to 1000 records can be written to each member of the file.

integer Specify the number of additional records which are to be added to the member. If 0 is specified for the increment number of records value, the member is not automatically extended. This value must be 0 if the value for the maximum increments is 0.

Maximum increment

Specify the number of additional records that are added to the member when the number of records in the member will exceed the initial number of records, or will exceed the current increment's number of records.

3 A maximum of 3 increments is automatically added to the member(s).

integer Specify the maximum number of increments automatically added to the member(s). Valid values range from 0 through 32767. If 0 is specified, the member is not automatically extended.

HDRNAMES – Header naming options

Specifies how the headers are named and treated.

Action if naming fails

***ABORT** If there are any columns, with conflicting headers names, the command will not import the excel spreadsheet.

***AUTONAME**

If there are any columns, with conflicting header names, the command will automatically rename the conflicting columns.

Header row number

***FIRST** Specifies that the first row of the spreadsheet, contains the headers for the fields, that are used for the database file.

decimal-number

Specifies the row number of the spreadsheet, where the headers will be used for the fields, for the database file.

REFFILE – Naming based-on file

Specifies the database file being used as a basis for the file being written to. The parameter is dependent on the parameter COLHEADING being *TABLE.

Naming based-in file

name — Specify the name of database file being used as a basis.

Library

***LIBL** — CoolSpools Database will use the library list to locate an existing file. If no file is found, a new file will be created in the current library.

***CURLIB** — CoolSpools Database will look for an existing file in the current library, or create a new file in the current library.

library_name — Specify the library in which an existing file is located or in which to create a new file.

COLDEFN – Column definition

Specifies how the columns are to be defined.

Note. This parameter may undergo further development in the future.

Options are:

***AUTO** — The columns are automatically defined by the command.

NBRROWS – Number of rows to convert

Specifies the number of rows, from th***e imported excel spreadsheet,*** to be converted.

Options are:

***ALL** — Convert all of the excel spreadsheet

decimal-number
Specify the number of rows to import (1 to 65565).

FIRSTROW – First row to convert

Specifies the first row from the exce***l spreadsheet, where the import*** will start.

1 to 65565 — Specify the start row to convert.

NBRCOLS – Number of columns to convert

Specifies the number of columns, from the imported excel spreadsheet, to be converted.

Options are:

***ALL** Convert all of the excel spreadsheet

decimal-number
Specify the number of columns to import.

FIRSTCOLS – First column to convert

Specifies the first column from the *excel spreadsheet, where the* import will start.

A-XFD *Excel column reference.*

IGNSUBTL – Ignore subtotals

Specifies whether the sub-totals within the spreadsheet, are to be ignored or not.

Options are:

***YES** Ignore the sub-totals.

***NO** Include the sub-totals.

WKSTONBR – Convert each sheet to a member

Specifies how the worksheets, from the excel spreadsheet are imported.

Options are:

***SAME** All of the worksheets will be imported into the same database file member.

***WORKSHEET**
A database file member will be created, for each worksheet.

MBRNAMES – New member naming

Specifies how the database file members will named.

Options are:

***AUTO** The database file members will be automatically named, for example, CSM0000001, CSM0000002.

***WORKSHEET**
The database file members, will be named as per the worksheet within the imported excel spreadsheet.

BLKCELLS- Treatment of blank cells

Specifies how blank cells are treated, when importing from an excel spreadsheet. Rows from the spreadsheet can be excluded, dependant on this parameter.

Options are:

<u>*BLANKZERO</u>
If any rows contain blank/empty cells, the rows are still included in the import.

***NULL** If any rows contain blank/empty cells, the rows are not imported. The database file, will represent these rows as a ***Deleted Record**.

FORMULAS – Convert formulas

Specifies how formulae are imported from the spreadsheet into the database file.

Options are:

<u>*YES</u> The cells containing formulae, will be imported showing the result, in the database field.

***NO** The cells containing formulae, will be imported not showing any result, in the database field.

CCSID1 – CCSID single byte

Specifies the CCSID (character encoding) of single byte characters to which the cell contents will be converted. Data in Excel files is typically stored in ASCII or unicode and you will normally wish to convert that data to an appropriate EBCDIC CCSID for processing on the system i.

Options are:

***JOB** The CCSID of the current job.

***SYSVAL** The CCSID indicated by the QCCSID system value.

***USER** The CCSID associated with the current user's user profile.

CCSID Specify a CCSID to use.

CCSID2 – CCSID double byte

Specifies the CCSID (character encoding) of double byte characters to which the cell contents will be converted. Data in Excel files is typically stored in ASCII or unicode and you will normally wish to convert that data to an appropriate EBCDIC CCSID for processing on the system i.

Options are:

***JOB** The CCSID of the current job.

***SYSVAL** The CCSID indicated by the QCCSID system value.

***USER** The CCSID associated with the current user's user profile.

CCSID Specify a CCSID to use.

IMPCSVDBF (Import Delimited Text to Database) command

The Import CSV to Database (IMPCSVDBF) command allows you to extract data from a delimited text file and place that data into a database file. Comma separated, pipe delimited and tab delimited text files can all be converted using this command.

If the target database file does not already exist then it can be created using field names and data types derived from the text file content. You also have the option to replace or append to the data within an existing database file.

This command provides a convenient method of loading data from a text file into the IBM i database without a need for manual processes on a PC. The conversion can be carried out automatically, and without user intervention (e.g. as part of your overnight batch job suite).

Note that the command provides additional functionality not available with the system command CPYFRMIMPF, such as the ability to create the target file (with control over the field naming and with data types automatically assigned based upon the content of the source text file) and the ability to select and order the columns from the source file that are to be converted.

The command also correctly handles the scenario where one or more rows contain an embedded line feed and/or carriage return character within a delimited text column - in these circumstances the data will be preserved as a single data value, rather than causing the row to be split and mishandled as multiple (incomplete) rows of data.

If conversion is performed to an existing database file then the data types and lengths of the database fields should match those of the text file columns, although you have control over the type of difference that will be reported as an error (for example, is truncation of text fields or rounding of numerics permitted).

The command parameters are as follows:

FROMSTMF - From Stream File

Specifies the path name of the delimited text file from which the data content will be converted.

This is a required parameter.

Options are:

path-name Specify the path name of the delimited text file

TOFILE - To Database File

The name of the database file which will receive the output.

If the file does not exist, then CREATE(*YES) must be specified, and parameters to assign field names and data types should be set.

If the file does exist, then CREATE(*NO) must be specified and either MBROPT(*ADD) or MBROPT(*REPLACE) used to indicate whether any existing data in the file is to be replaced. The field data types and lengths in an existing file should correspond to those of the text file columns.

File to receive output

name Specify the name of the file into which the data will be extracted.

Library

***LIBL** CoolSpools Database will use the library list to locate an existing file. This cannot be used with parameter CREATE(*YES).

***CURLIB** CoolSpools Database will output to a database file in the current library of the job.

name Specify the library in which an existing file is located or within which to create a new file.

MBR - Member to receive output

The name of the member in the database file which will receive the output.

Options are:

***FIRST** The data will be written to the first member in the file. If used with CREATE(*YES) or if there are no members in an existing file, then a new member with the same name as the database file will be used.

name Specify the name of the member to receive the data.

MBROPT - Replace or add records

Specifies whether the converted data replaces or is appended to any existing data records within the target database file member. When creating a new database file with CREATE(*YES) this parameter should be set to MBROPT(*NONE).

Options are:

***NONE** When creating a new file *NONE is required.

***REPLACE** Any existing data records in the file member are replaced with the converted data.

***ADD** The converted data is appended to any existing data records.

CREATE - Create File

Specifies whether a new database file is to be created in order to receive the converted data, or if an existing database file will be used.

When using CREATE(*YES) to create a new database file, parameters that control the assignment of database field names and data types should also be set.

When using CREATE(*NO) to output the converted data to an existing file, the database file named in the TOFILE() parameter must already exist, and the MBROPT() parameter must be used to specify whether to replace existing data records in that file. Field data types and lengths in the existing file should match the columns within the text file.

Options are:

***YES** A new database file is created.

***NO** The target database file already exists.

RCDFMT - Record Format Name

Specifies the record format name of the target database file.

When using CREATE(*YES) this parameter controls the record format name that will be used in the new database file. A value can be specified, or the special values *TOFILE and *TOFILER can be used to automatically generate a record format name based upon the name of the database file.

When using CREATE(*NO) the special value *ONLY indicates that the existing record format name of the database file should be used.

Options are:

***TOFILE** The record format will be the same as the database file name. For example, if the file to be created is named ORDLIN then the record format will also be named ORDLIN.

***TOFILER** The record format will be the same as the database file name, but with the character "R" appended. For example, if the file to be created is named ORDLIN then the record format will be named ORDLINR.

***ONLY** The existing record format name will be used - this value is only suitable for use when using CREATE(*NO) to output to an existing database file.

name Specify the required name of the record format.

FLDDLM - Field Delimiter

Specifies the character that is used within the delimited text file to separate columns. Typically this is a comma, pipe or tab, although any character can be specified.

This parameter allows CoolSpools to differentiate the data columns within the source text file, and is required whether you are creating a new database file or outputing to an existing file.

Options are:

***COMMA** Columns in the text file are separated by a comma character.

***PIPE** Columns in the text file are separated by a pipe character.

***TAB** Columns in the text file are separated by a tab character.

character-value Specify the separator character used to delimit columns in the text file.

STRDLM - String Delimiter

Specifies the character that is used within the delimited text file to enclose character strings. Typically this is a single quote or double quote character. If character data is not delimited then specify *NONE.

This parameter helps CoolSpools to identify data types within the source text file, and to exclude string delimiter characters from the converted data values. It should be accurately specified whether you are creating a new database file or outputting to an existing file.

Options are:

***DBLQTE** Character columns in the text file are enclosed within double quotes.

***SGLQTE** Character columns in the text file are enclosed within single quotes.

***NONE** Character columns in the text file are not enclosed within identifiable characters.

character-value Specify the character used to enclose character strings in the text file.

HDRROW - Header Rows

Specifies the number of rows at the top of the delimited text file that contain headers. Correctly identifying the header rows ensures that CoolSpools will not attempt to convert the header rows as data and also allows the use of header text when generating field names for a new database file with CREATE(*YES).

This parameter allows CoolSpools to differentiate between header and data rows in the source text file, and is required whether you are creating a new database file or outputting to an existing file.

Options are:

***NONE** The source text file does not contain any headers, and consists entirely of data rows. If there are no headers then the options FLDNAM(*HEADER) and FLDALS(*HEADER) are not available for generation of field name and alias values when creating a new file with CREATE(*YES).

number The number of rows that contain header information.

FROMROW - From Data Row

The data row from which data is to be converted. Used with TOROW() to specify the range of data rows that are to be converted.

Note that header rows identified with parameter HDRROW() are automatically excluded from conversion, so there is no need to use this parameter to exclude headers as FROMROW(*START) will convert from the first data row.

When identifying row numbers in the source text file, bear in mind that any data rows containing carriage return/line feed characters embedded within a delimited character string may be presented as separate (incomplete) rows within a text viewer, but will be correctly handled as a single row of data by CoolSpools.

Options are:

***START** Data conversion will start at the first data row in the source text file.

number The data row number from which data will be converted. If this is greater than the number of rows in the file then no data will be converted. This may be useful if you wish to use CREATE(*YES) to create an empty database file with field data types and lengths generated based on the text file columns.

TOROW - To Data Row

The data row upto which data is to be converted. Used with FROMROW() to specify the range of data rows that are to be converted.

When identifying row numbers in the source text file, bear in mind that header rows identified using HDRROW() are already excluded, and that any data rows containing carriage return/line feed characters embedded within a delimited character string may be presented as separate (incomplete) rows within a text viewer, but will be correctly handled as a single row of data by CoolSpools.

Options are:

***END** Data conversion will include rows to the end of the source text file.

number The data row number upto which data will be converted. This value cannot be lower than the row number specified in FROMROW().

SLTROW - Select Rows

Select data rows to be converted based on the data content of the source data columns.

This can be useful if loading a multi-format text file with a record type identifier, since records of each record type can be converted separately.

When row selection is to be used, specify a column identifier, expression and value (e.g. COL1 *EQ '1'). Where the conditions specified equate to true then the row will be included in the conversion. Where multiple conditions are specified these should be accompanied by logical expression *AND or *OR.

Single values:

***ALL** All rows are selected (unless omitted by FROMROW and TOROW).

Column Identifier
Column identifier as column number (from left to right) expressed either as COLn (e.g. COL1-COL99) or using Excel column notation (A-Z, then AA-AZ).

column-id Column identifier as COLn or Excel notation (A-Z then AA-AZ).

Expression
Logical expression to be used in comparison of the column data and value.

***EQ**	Column data is equal to specified value.
***NE**	Column data is not equal to specified value.
***LT**	Column data is less than specified value.
***LE**	Column data is less than or equal to specified value.
***GT**	Column data is greater than specified value.
***GE**	Column data is greater than or equal to specified value.

Value
Value against which the column data is to be compared.

text	Value against which the column data is to be compared.

Logical Expression
How to combine multiple expressions.

For example (COL1 *EQ '1') *AND (COL1 *EQ '2') will never be true since COL1 cannot equal both '1' and '2', but (COL1 *EQ '1') *OR (COL1 *EQ '2') will be true if COL1 is equal to '1' or '2'.

***AND**	Rows are selected if both conditions are true.
***OR**	Rows are selected if either condition is true.

INCLCOL - Include Columns

Optionally identify and sequence the text file columns that are to be included in the conversion.

By default all columns of the source text file are converted in the order that they appear. By using this parameter you can select specific columns to convert, and the order in which they will appear in the output database file.

Fields can be identified by their column number in the source file, either as COLn (e.g. COL1-COL999) or Excel column notation (e.g.]A-Z, then AA-AZ), or by the field name assigned using FLDNAM(*HEADER), which is derived from the column header in the text file.

Options are:

***ALL**	All columns will be included in the conversion, in the order that they appear in the source text file, unless used in combination with EXCLCOL() in which case all columns are included except those explicitly excluded.
column-identifier	One or more column identifiers. Columns can be identified using any combination of the following:

- o Column number — The position of the column within the text file (from left to right) formatted as COLn (e.g. first column is COL1, thirtieth column is COL30).
- o Excel column notation — The position of the column within the text file (from left to right) using Excel notation A-Z, then AA-AZ (e.g. first column is A, thirtieth column is AD).
- o Field name derived from header — Where field names are generated from the header using FLDNAM(*HEADER) the generated field name can be used to identify columns. Note that the header text may be truncated based on parameter FLDNAMLEN() and if there are any duplicates amongst the truncated field names then a numeric suffix may have been added. The value used as an identifier must match the truncated text, with possible numeric suffix, which might differ from the raw header text in the source text file.

EXCLCOL - Exclude Columns

Optionally identify text file columns that are to be excluded from the converted data set.

This can only be used when INCLCOL() is set to *ALL.

By default all columns of the source text file are converted to database field values. By using this parameter you can select specific columns to be excluded from the conversion.

Only those columns that are not identified in this list will be included in the conversion.

Fields can be identified by their column number in the source file, either as COLn (e.g. COL1-COL999) or Excel column notation (e.g. A-Z, then AA-AZ), or by a field name assigned from the column header text using FLDNAM(*HEADER).

Options are:

***NONE** — No columns are to be excluded from the conversion.

column-identifier — One or more column identifiers. Columns can be identified using any combination of the following:

- o Column number — The position of the column within the text file (from left to right) formatted as COLn (e.g. first column is COL1, thirtieth column is COL30).

- Excel column notation — The position of the column within the text file (from left to right) using Excel notation A-Z, then AA-AZ (e.g. first column is A, thirtieth column is AD).

- Field name derived from header — Where field names are generated from the header using FLDNAM(*HEADER) the generated field name can be used to identify columns. Note that the header text may be truncated based on parameter FLDNAMLEN() and if there are any duplicates amongst the truncated field names then a numeric suffix may have been added. The value used as an identifier must match the truncated text, with possible numeric suffix, which might differ from the raw header text in the source text file.

FLDNAM - Field Name

Specifies how the field names are to be assigned when creating a new database file with CREATE(*YES).

When the conversion is placing data into an existing database file any field names assigned here are ignored and the existing database field names are retained.

When mapping header text to field name with FLDNAM(*HEADER) the result can be used in identification of columns for the INCLCOL() and EXCLCOL() parameters whether a new database file is being created or data is being output to an existing file.

Options are:

***COLNO** The number of the column in the source text file (from left to right) is formatted as COLn. By default fields in the output file will be named COL1, COL2, COL3, etc. If INCLCOL() is used to select and order columns then the field names will relate to the column position within the source text file, and may not correspond to the order of fields in the generated file.

***COLXL** The number of the column in the source text file (from left to right) is converted to Excel column notation (A-Z, then AA-AZ...). Fields in the output file will be named A, B, C, etc. If INCLCOL() is used to select and order columns then the field names will relate to the column position within the source text file, and may not correspond to the order of fields in the generated file.

***HEADER** The header text of the column is to be used as the source of the field name. The parameter FLDNAMLEN() allows you to set a maximum length for field name values, for example FLDNAMLEN(6) will truncate a header of "Order Number" to a field name ORDERN. If there is any duplication of field names

derived from headers, then a numeric suffix is automatically applied, for example headers "Order Number" and "Order Name" would both be truncated to field name ORDERN, so the field names ORDE01 and ORDE02 would be assigned. The chances of duplication can be reduced by using a longer field name length FLDNAMLEN(10) or FLDNAMLEN(20).

FLDNAM(*HEADER) cannot be used if there are no header rows in the text file - HDRROW(*NONE) or HDRROW(0).

***FLDNO** A field name of FLDn is assigned based on the position of the field within the output data record. This field name is assigned without reference to the column position or header text in the source text file, so output fields will always be named FLD1, FLD2, FLD3, etc. even if INCLCOL() is used to amend the order of the columns. This field name cannot be used as a method of column identification in the INCLCOL() and EXCLCOL() parameters.

FLDNAMLEN - Field Name Length

When field name is to be generated from column header text, this parameter allows you to specify the maximum length of the field names to be generated. The three possible values are 6, 10 and 20.

Header text that is longer than the specified length will be truncated. If there are any duplicates amongst the truncated field names then a numeric suffix will be appended to ensure uniqueness.

Options are:

6 The maximum allowed field name length is six characters, and any column headers that exceed this will be truncated.

10 The maximum allowed field name length is ten characters, and any column headers that exceed this will be truncated.

20 The maximum allowed field name length is twenty characters, and any column headers that exceed this will be truncated.

FLDALS - Field Alias

When generating a new database file with CREATE(*YES), in addition to assigning field names you can also assign an optional field alias. A field alias can be used interchangeably with the field name in languages such as SQL and RPG free, and allows a longer name to be assigned than is supported by the field name.

Assigning a field name using FLDNAM(*COLNO) or FLDNAM(*FLDNO) in conjunction with FLDALS(*HEADER) allows you to create fields that can be referenced in your programs by

both a short id (e.g. COL1, FLD12) and a longer meaningful name (e.g. CUSTOMERNAME, ORDERNUMBER).

Options are:

***NONE** Field aliases will not be generated. The fields in the created file will be identifiable by field name only.

***COLNO** The number of the column in the source text file (from left to right) is formatted as COLn. By default fields in the output file will have an alias COL1, COL2, COL3, etc. If INCLCOL() is used to select and order columns then the field aliases will relate to the column position within the source text file, and may not correspond to the order of fields in the generated file.

***COLXL** The number of the column in the source text file (from left to right) is converted to Excel column notation (A-Z, then AA-AZ...). Fields in the output file will have an alias A, B, C, etc. If INCLCOL() is used to select and order columns then the field aliases will relate to the column position within the source text file, and may not correspond to the order of fields in the generated file.

***HEADER** The header text of the column is to be used as the source of the field alias. The parameter FLDALSLEN() allows you to set a maximum length for field aliases, for example FLDALSLEN(20) will truncate a header of "Order Original Dispatch Date" to field alias ORDERORIGINALDISPATC. If there is any duplication of field aliases derived from headers, then a numeric suffix is automatically applied, for example headers "Order Original Dispatch Date" and "Order Original Dispatch Time" would both be truncated to field alias ORDERORIGINALDISPATC, so the alias values ORDERORIGINALDISPA01 and ORDERORIGINALDISPA02 would be assigned. The chances of duplication can be reduced by using the longest alias length FLDALSLEN(128). FLDALS(*HEADER) cannot be used if there are no header rows in the text file - HDRROW(*NONE) or HDRROW(0).

***FLDNO** A field alias of FLDn is assigned based on the position of the field within the output data record. This field alias is ssigned without reference to the column position or header text in the source text file, so output field aliases will always be FLD1, FLD2, FLD3, etc. even if INCLCOL() is used to amend the order of the columns.

FLDALSLEN - Field Alias Length

When field alias is to be generated from column header text, this parameter allows you to specify the maximum length of the field aliases to be generated. The three possible values are 10, 20 and 128.

Header text that is longer than the specified length will be truncated. If there are any duplicates amongst the truncated field aliases then a numeric suffix will be appended to ensure uniqueness.

Options are:

10 The maximum allowed field alias length is ten characters, and any column headers that exceed this will be truncated.

20 The maximum allowed field alias length is twenty characters, and any column headers that exceed this will be truncated.

128 The maximum allowed field alias length is 128 characters, and any column headers that exceed this will be truncated.

DTATYP - Data Type

When converting column data, by default the output data type will be derived from the content of the text file. This will result in CHAR or VARCHAR data if the column data appears to be character in nature, or as PACKED DECIMAL or FLOAT if the column data appears to be numeric. Alternatively you can specify that all columns should be treated as character data and converted to CHAR or VARCHAR field data.

When generating a new database file with CREATE(*YES) this parameter will decide the data type of the fields within the created file.

When outputting to an existing database file this parameter will have no effect as the data types of the existing fields will be retained.

Options are:

***CALC** Calculate the required field data type based on the text file content.

***CHAR** Treat all columns as character data.

VARCHAR - VARCHAR Threshold

When converting column data to a character field you can optionally set a field length from which character fields will be defined as VARCHAR rather than CHAR.

For example, specifying VARCHAR(10) will result in all columns containing character data values that exceed ten characters in length being created as VARCHAR fields and columns containing character data values upto ten characters being created as CHAR fields.

Specify *NEVER if you do not wish any fields to be created as VARCHAR.

When generating a new database file with CREATE(*YES) this parameter will decide whether character fields within the created file are CHAR or VARCHAR.

When outputting to an existing database file this parameter will have no effect as the data types of the existing fields will be retained.

Options are:

***NEVER** All character data will be converted to CHAR data type, and never to VARCHAR.

number Character field length above which data will be converted to VARCHAR.

OPTIONS - Conversion Options

Specify optional conversion options to control how data will be converted, and how errors encountered during conversion will be handled.

Options are:

***NONE** No special options are to be used.

***TRUNCATE** Where character data is longer than the database field to which it is being converted, the character data will be truncated to fit. If this option is not specified then oversized character data will result in an error being reported. *TRUNCATE cannot be used with option *FASTMODE.

***NOROUND** By default the import will apply rounding to any numeric data within the text file that has greater decimal precision than the field to which it is being loaded. This can result in minor differences in the imported data. If you do not wish rounding to be applied and for this to be reported as an error then specify option *NOROUND.

***MAPDROP** Where the number of the columns selected from the text file does not correspond to the number of fields in the target database file, additional columns will be dropped and additional fields will not be populated. Where the data types do not match then fields will be populated with default values of zero (numeric), 0001-01-01 (date) or 00.00.00 (time) if the source data cannot be converted. If this option is not specified then differences in the number or data type of columns will result in an error being reported.

***FASTMODE** When conversion is run in fast mode no pre-validation of the data is performed, and data values from the source text file are inserted directly into the database file without being checked. This option is incompatible with the *TRUNCATE option, as truncation is applied as part of the pre-validation step. Running in fast mode can result in shorter conversion times, but is reliant on compatibility between your source data columns and target database fields.

***NOMSG** By default every data issue (such as truncated character data or invalid numeric data) results in a message being issued to the joblog. By specifying the *NOMSG option these messages will be suppressed, and the only message issued will be the final status message confirming the number of data rows converted and the number that failed to convert. This option is incompatible with the error action ERRACT(*STOP) as the conversion will stop on the first data issue that is encountered.

***IGNORECASE** When using the SLTROW() parameter to select rows based on their data content, this option causes data comparisons to ignore case. For example SLTROW((COL1 *EQ 'FRED')) will not convert a row where column 1 equals 'Fred' unless *IGNORECASE is specified.

***DIAGNOSTIC** When run in diagnostic mode, detailed messages are output to the joblog. This may be useful to investigate when a data import does not behave as expected, or reports an unexpected error. The CoolSpools technical support team may request that you use this option when raising a support ticket.

ERRACT - Error Action

Action to be taken should any errors be identified during data conversion.

Errors may result from mismatches in data types between the source text columns and the target database fields. If an existing database file includes a constraint then this may cause an error if the converted data does not satisfy the constraint.

This parameter can be used to define what should happen if an error is encountered.

If set to *FILE then the accompanying parameter ERRFIL() must also be supplied to specify the required location of the error file.

Options are:

***CONTINUE** Issue a message to report the error but continue processing the remaining data rows.

***STOP** Issue a message to report the error and stop processing data rows.

***FILE** Issue a message to report the error and copy the data row into an error file on the IFS, then continue processing the remaining data rows. At the end of processing the error file will contain all data rows that failed to load. The required location of the error file must be supplied with ERRFIL().

ERRFIL - Error File

When ERRACT(*FILE) is used this parameter must be set to the required path of the IFS stream file that is to be populated with all data rows that fail to load due to errors.

If the stream file already exists then its current content will be replaced. If the stream file does not exist then it will be created.

If a relative path is used then the file location will be dependent upon the current directory of the runtime user. In order to guarantee the file location an absolute path can be used.

If the specified file location cannot be used, for example due to permissions or an incorrect directory path, then this will prevent the data conversion from running.

Options are:

path-name Specify the required path to the IFS stream file to be created or replaced with any data rows that fail to load due to errors.

IMPFXDDBF (Import Fixed Pos Text to Database) command

The Import Fixed Pos to Database (IMPFXDDBF) command allows you to extract data from a fixed position text file and place that data into a database file. A fixed position text file is a file where columns of data are not delimited, but are located in a predefined location within each row.

If the target database file does not already exist then it can be created. You also have the option to replace or append to the data within an existing database file.

This command provides a convenient method of loading data from a text file into the IBM i database without a need for manual processes on a PC. The conversion can be carried out automatically, and without user intervention (e.g. as part of your overnight batch job suite).

Note that the command provides additional functionality not available with the system command CPYFRMSTMF, such as the ability to create the target file and the ability to select and order the columns from the source file that are to be converted. You can also select the rows to be loaded, allowing processing of a text file with multiple record types.

If conversion is performed to an existing database file then the data types and lengths of the database fields should match those of the text file columns, although you have control

over the type of difference that will be reported as an error (for example, is truncation of text fields or rounding of numerics permitted).

The command parameters are as follows:

FROMSTMF - From Stream File

Specifies the path name of the fixed position text file from which the data content will be converted.

This is a required parameter.

Options are:

path-name Specify the path name of the fixed position text file

TOFILE - To Database File

The name of the database file which will receive the output.

If the file does not exist, then CREATE(*YES) must be specified, and parameters to assign field names and data types should be set.

If the file does exist, then CREATE(*NO) must be specified and either MBROPT(*ADD) or MBROPT(*REPLACE) used to indicate whether any existing data in the file is to be replaced. The field data types and lengths in an existing file should correspond to those of the text file columns.

File to receive output

name Specify the name of the file into which the data will be extracted.

Library

***LIBL** CoolSpools Database will use the library list to locate an existing file. This cannot be used with parameter CREATE(*YES).

***CURLIB** CoolSpools Database will output to a database file in the current library of the job.

name Specify the library in which an existing file is located or within which to create a new file.

MBR - Member to receive output

The name of the member in the database file which will receive the output.

Options are:

***FIRST** The data will be written to the first member in the file. If used with CREATE(*YES) or if there are no members in an existing file, then a new member with the same name as the database file will be used.

name Specify the name of the member to receive the data.

MBROPT - Replace or add records

Specifies whether the converted data replaces or is appended to any existing data records within the target database file member. When creating a new database file with CREATE(*YES) this parameter should be set to MBROPT(*NONE).

Options are:

***NONE** When creating a new file *NONE is required.

***REPLACE** Any existing data records in the file member are replaced with the converted data.

***ADD** The converted data is appended to any existing data records.

CREATE - Create File

Specifies whether a new database file is to be created in order to receive the converted data, or if an existing database file will be used.

When using CREATE(*YES) to create a new database file, parameters that control the assignment of database field names and data types should also be set.

When using CREATE(*NO) to output the converted data to an existing file, the database file named in the TOFILE() parameter must already exist, and the MBROPT() parameter must be used to specify whether to replace existing data records in that file. Field data types and lengths in the existing file should match the columns within the text file.

Options are:

***YES** A new database file is created.

***NO** The target database file already exists.

RCDFMT - Record Format Name

Specifies the record format name of the target database file.

When using CREATE(*YES) this parameter controls the record format name that will be used in the new database file. A value can be specified, or the special values *TOFILE and *TOFILER can be used to automatically generate a record format name based upon the name of the database file.

When using CREATE(*NO) the special value *ONLY indicates that the existing record format name of the database file should be used.

Options are:

***TOFILE** The record format will be the same as the database file name. For example, if the file to be created is named ORDLIN then the record format will also be named ORDLIN.

***TOFILER** The record format will be the same as the database file name, but with the character "R" appended. For example, if the file to be created is named ORDLIN then the record format will be named ORDLINR.

***ONLY** The existing record format name will be used - this value is only suitable for use when using CREATE(*NO) to output to an existing database file.

name Specify the required name of the record format.

COLUMNS - Columns

Specify the position, length and type of the data columns to be imported from the source text file. Note that it is not necessary to specify all columns of the source file, only those that are to be imported. Note also that the order in which the columns are specified does not need to correspond to their order within the source file, and it is possible to specify the same data position more than once if it is required to populate more than one field in the database file.

This parameter also allows you to specify field names to be used when output is to a new database file.

Start Position
Starting character position (from left to right) of the column data within the source text file.

Specify special value *NEXT to start in the next character following the end of the previous column (or the start of the record for the first column).

Options are:

***NEXT** Column data starts in the next character following the end of the previous column. If this is the first column then the column starts in the first character of the text.

number Start position of the column data (from left to right).

Length
Length of the column data within the source text file. For numeric values this is the entire length used to contain the value, inclusive of commas, decimal point and minus sign. For date or time values this is the entire length used to contain the value including any separator characters that are present.

Options are:

number Length of the column data within the text file.

Data Type
The type of data contained within the data column.

Specify special value *NEXT to start in the next character following the end of the previous column (or the start of the record for the first column).

Options are:

***TOFILE** When output is to an existing database file, the type of data is decided by the field type of the corresponding field within the existing file. This option is not valid when CREATE(*YES) is specified to create a new file.

***CHAR** The column contains alphanumeric character data.

***NUMERIC** The column contains integer or decimal numeric data.

***FLOAT** The column contains floating point numeric data.

***DATE** The column contains a date formatted as YYYY-MM-DD.

***TIME** The column contains a date formatted as hh.mm.ss.

***TIMESTAMP** The column contains a timestamp formatted as YYYY-MM-DD-hh.mm.ss.tttttt.

***SLTONLY** The column is not to be imported to database, but is defined for use in SLTROW() record selection only (e.g. a record type indicator).

Decimals
Where the data type is *NUMERIC this parameter allows you to specify the decimal precision of the numeric value. Where output is to a new database file this parameter is used to assign decimal precision to numeric fields.

Options are:

***NONE** No decimals are needed, because the data type is integer or not numeric.

number The number of digits after the decimal point.

Field Name
When output is to a new database file you can use this parameter to assign the field name. If left as *DFT the fields will be named COL1, COL2, etc. When output is to an existing file this parameter is not used and can be left as the default value *DFT. Where the file contains one or more header rows, the header text can be used as the basis of the filename by specifying *HEADER.

The field name assigned here can also be referenced within the SLTROW() row selection expressions.

Options are:

<u>*DFT</u> A default field name of COL*n* will be used, based on the sequence of the column in the output record set (e.g. the first column will be COL1 and so on).

***HEADER** The field name will be derived from the text in the header row(s). If no header text is found then the default COL*n* will be used instead. This option is only available where the number of header rows specified with HDRROW() is greater than zero.

character Character value to be used as the field name. You cannot specify the same field name for more than one column, and the value must adhere to IBM i field naming conventions (e.g. cannot contain restricted characters).

HDRROW - Header Rows

Specifies the number of rows at the top of the delimited text file that contain headers. Correctly identifying the header rows ensures that CoolSpools will not attempt to convert the header rows as data and also allows the use of header text when generating field names for a new database file with CREATE(*YES).

This parameter allows CoolSpools to differentiate between header and data rows in the source text file, and is required whether you are creating a new database file or outputting to an existing file.

When creating a new database file with CREATE(*YES) field names can only be set using special value *HEADER if one or more header rows exist.

Options are:

<u>*NONE</u> The source text file does not contain any headers, and consists entirely of data rows.

number The number of rows that contain header information.

FROMROW - From Data Row

The data row from which data is to be converted. Used with TOROW() to specify the range of data rows that are to be converted.

Note that header rows identified with parameter HDRROW() are automatically excluded from conversion, so there is no need to use this parameter to exclude headers as FROMROW(*START) will convert from the first data row.

Options are:

***START** Data conversion will start at the first data row in the source text file.

number The data row number from which data will be converted. If this is greater than the number of rows in the file then no data will be converted. This may be useful if you wish to use CREATE(*YES) to create an empty database file with field data types and lengths generated based on the text file columns.

TOROW - To Data Row

The data row upto which data is to be converted. Used with FROMROW() to specify the range of data rows that are to be converted.

When identifying row numbers in the source text file, bear in mind that header rows identified using HDRROW() are already excluded.

Options are:

***END** Data conversion will include rows to the end of the source text file.

number The data row number upto which data will be converted. This value cannot be lower than the row number specified in FROMROW().

SLTROW - Select Rows

Select data rows to be converted based on the data content of the source data columns.

This can be useful if loading a multi-format text file with a record type identifier, since records of each record type can be converted separately.

Single values:

***ALL** All rows are selected (unless omitted by FROMROW and TOROW).

Column Identifier
Column identifier as column number (from left to right) expressed as COL*n* (e.g. COL1-COL99), or a field name assigned within the COLUMNS() parameter.

column-id Column identifier as COL*n*. This reflects the position of the column within the list of columns specified in the COLUMNS() parameter.

field-name Field name as specified within the COLUMNS() parameter. This includes fields defined with data type *SLTONLY.

Expression
Logical expression to be used in comparison of the column data and value.

***EQ** Column data is equal to specified value.
***NE** Column data is not equal to specified value.
***LT** Column data is less than specified value.
***LE** Column data is less than or equal to specified value.
***GT** Column data is greater than specified value.
***GE** Column data is greater than or equal to specified value.

Value
Value against which the column data is to be compared.

text Value against which the column data is to be compared.

Logical Expression
How to combine multiple expressions.

For example (COL1 *EQ '1') *AND (COL1 *EQ '2') will never be true since COL1 cannot equal both '1' and '2', but (COL1 *EQ '1') *OR (COL1 *EQ '2') will be true if COL1 is equal to '1' or '2'.

***AND** Rows are selected if both conditions are true.

***OR** Rows are selected if either condition is true.

OPTIONS - Conversion Options

Specify optional conversion options to control how data will be converted, and how errors encountered during conversion will be handled.

Options are:

***NONE** No special options are to be used.

***TRUNCATE** Where character data is longer than the database field to which it is being converted, the character data will be truncated to fit. If this option is not specified then oversized character data will

result in an error being reported. *TRUNCATE cannot be used with option *FASTMODE.

***NOROUND** By default the import will apply rounding to any numeric data within the text file that has greater decimal precision than the field to which it is being loaded. This can result in minor differences in the imported data. If you do not wish rounding to be applied and for this to be reported as an error then specify option *NOROUND.

***MAPDROP** Where the number of the columns selected from the text file does not correspond to the number of fields in the target database file, additional columns will be dropped and additional fields will not be populated. Where the data types do not match then fields will be populated with default values of zero (numeric), 0001-01-01 (date) or 00.00.00 (time) if the source data cannot be converted. If this option is not specified then differences in the number or data type of columns will result in an error being reported.

***FASTMODE** When conversion is run in fast mode no pre-validation of the data is performed, and data values from the source text file are inserted directly into the database file without being checked. This option is incompatible with the *TRUNCATE option, as truncation is applied as part of the pre-validation step. Running in fast mode can result in shorter conversion times, but is reliant on compatibility between your source data columns and target database fields.

***NOMSG** By default every data issue (such as truncated character data or invalid numeric data) results in a message being issued to the joblog. By specifying the *NOMSG option these messages will be suppressed, and the only message issued will be the final status message confirming the number of data rows converted and the number that failed to convert. This option is incompatible with the error action ERRACT(*STOP) as the conversion will stop on the first data issue that is encountered.

***IGNORECASE** When using the SLTROW() parameter to select rows based on their data content, this option causes data comparisons to ignore case. For example SLTROW((COL1 *EQ 'FRED')) will not convert a row where column 1 equals 'Fred' unless *IGNORECASE is specified.

***DIAGNOSTIC** When run in diagnostic mode, detailed messages are output to the joblog. This may be useful to investigate when a data import does not behave as expected, or reports an unexpected error. The CoolSpools technical

support team may request that you use this option when raising a support ticket.

ERRACT - Error Action

Action to be taken should any errors be identified during data conversion.

Errors may result from mismatches in data types between the source text columns and the target database fields. If an existing database file includes a constraint then this may cause an error if the converted data does not satisfy the constraint.

This parameter can be used to define what should happen if an error is encountered.

If set to *FILE then the accompanying parameter ERRFIL() must also be supplied to specify the required location of the error file.

Options are:

***CONTINUE** Issue a message to report the error but continue processing the remaining data rows.

***STOP** Issue a message to report the error and stop processing data rows.

***FILE** Issue a message to report the error and copy the data row into an error file on the IFS, then continue processing the remaining data rows. At the end of processing the error file will contain all data rows that failed to load. The required location of the error file must be supplied with ERRFIL().

ERRFIL - Error File

When ERRACT(*FILE) is used this parameter must be set to the required path of the IFS stream file that is to be populated with all data rows that fail to load due to errors.

If the stream file already exists then its current content will be replaced. If the stream file does not exist then it will be created.

If a relative path is used then the file location will be dependent upon the current directory of the runtime user. In order to guarantee the file location an absolute path can be used.

If the specified file location cannot be used, for example due to permissions or an incorrect directory path, then this will prevent the data conversion from running.

Options are:

path-name Specify the required path to the IFS stream file to be created or replaced with any data rows that fail to load due to errors.

BLDXMLDBF (Create XML-to-DBF Map from XML) Command

CoolSpools XML-to-DBF is a tool for importing XML data into IBM i database files. The conversion process uses a defined XML-to-Database Map to translate the content of an XML document to database files and fields.

The Create XML-to-Database Map (BLDXMLDBF) command parses a sample XML file and creates a XML-to-Database Map based on its structure. The map will contain detailed structure of the XML elements and attributes, but will not contain any references to database files and fields. These need to be supplied using the command **WRKXMLDBF**.

The command parameters are as follows:

PATH – Sample XML document

The path specifies the location on the IFS of the XML document whose structure is to be used as the basis for the XML-to-Database Map. This must be an existing XML document.

Either an absolute path or a relative path may be specified, however use of an absolute path is recommended. If a relative path is specified, then the IFS location will be relative to the current job's home directory, and this may vary from user to user.

MAPNAME – XML-to-Database Map to be Created

The name of the XML-to-Database Map to be created. The name specified here can later be used in commands **WRKXMLDBF** and **IMPXMLDBF** to refer to the map.

If an XML-to-Database Map with the specified name already exists, then the existing map will not be replaced, and the build of a new XML-to-Database Map will fail.

WRKXMLDBF (Work with XML-to-DBF Map) Command

CoolSpools XML-to-DBF is a tool for importing XML data into IBM i database files. The conversion process uses a defined XML-to-Database Map to translate the content of an XML document to database files and fields.

The Work with XML-to-Database Map (WRKXMLDBF) command allows you maintain existing maps, and to specify the database files and fields that are to be populated from each XML element and attribute. It is possible to manually define the XML structure by adding elements and attributes with this command, but it is more likely that you will create the structure of the map using the **BLDXMLDBF** command.

The mappings defined here are used by command **IMPXMLDBF** to import from XML.

The command parameters are as follows:

MAPNAME – XML-to-Database Map to work with

The name of the existing XML-to-Database Map to be maintained. If *SELECT is used then you will be presented with a list of all existing XML-to-Database Maps, otherwise you will be taken straight to the Work with XML-to-DBF Map Elements screen for the specified map.

The default is the special value:

***SELECT**

A list of all existing XML-to-Database Maps will be presented. The screen that is presented provides the following options:

4=Delete Map

Runs command DLTXMLDBF to delete the XML-to-Database Map.

6=Print Map

A spooled file report of the XML-to-Database Map content will be produced.

8=Elements

Drill down to the Work with XML-to-DBF Map Elements screen. This is equivalent to specifying a map name rather than *SELECT for the MAPNAME parameter. See below for details of this screen.

15=Save

Runs command SAVXMLDBF to save the map definition to a stream file on the IFS. This can be used as a backup, for distribution of maps between servers or for version control purposes.

16=Restore

Runs command RSTXMLDBF to restore a map definition from a stream file on the IFS created using the SAVXMLDBF command.

F6=Build New Map

Runs command BLDXMLDBF to build a new map definition from an XML document.

Work with XML-to-DBF Map Elements screen

This screen presents a list of the XML elements that exist at each level. When you first open the element mapping you will be presented with the top-level root element of the XML structure. You can navigate between XML elements and attributes using the screen options:

2=Edit DBF

This will take you to the Change XML-to-DBF Map Element screen, where you can specify database file and field details.

4=Delete

This will delete the selected element, together with any associated sub-elements and attributes.

6=Link DBF

This will take you to the Change DBF Link screen to link a real XML data element to a virtual element at another level of the XML structure.

8=Elements

This will drill down to the sub-elements of the selected element and present the Work with XML-to-DBF Map Elements screen again. This allows you to navigate through a nested XML element structure.

9=Attributes

This will take you to the Work with XML-to-DBF Map Attributes screen for attributes of the selected element.

F6=Add

This will take you to the Add XML-to-DBF Map Element screen, where you can create additional XML elements at the current element level.

F11=View 1/2

This will switch between two alternate views. View 1 presents the Element name, details of the linked database file and/or field, whether the Link DBF option is in use, and whether the element contains any sub-elements or attributes. View 2 is the same, but hides the database details in order to present a longer Element name – this is useful for working with XML documents that use long Element naming.

Add XML-to-DBF Map Element screen

This screen allows you to create a new element within the XML structure. This may be because an optional element was not present in the XML document sampled by BLDXMLDBF, or because you want to create a virtual element that does not exist in the source XML, but which is to be populated using the DBF Link option.

The screen allows you to enter the following settings:

Element Name

The name of the XML element to be added.

Action

The database action to be performed when this element is encountered. This can be any one of the following:

N/blank = No action

No database activity will take place.

D = Data

Database field level mapping for this data is to take place. This should be specified for an element that contains a data value. The database file and field names must be specified.

R = Write Record

A Database record will be written. This is usually specified at a parent "container" element. The database file name must be specified.

File Name

The name of the database file to which data is to be written. This is required if action is D (field level data mapping) or R (write file).

Library

The library of the database file to which data is to be written. This is required if action is D (field level data mapping) or R (write file). A library name can be specified, or the special value *LIBL if the file is to be found in the library list at runtime when the conversion is performed. Using *LIBL allows flexibility in use of the same map with different data libraries, but does rely on the library list being set correctly when the conversion is run.

Field

The database field to which the element data is to be mapped. This is only required if action is D (field level data mapping). In addition to a field name, you can specify the following:

****NONE***

Required if action is not D (field level data mapping).

****SELECT***

Presents a list of field names from the database file specified in the file name and library settings. Enter **1=Select** to select the required field.

Data Type

The type of data that is being mapped from the XML document. This is only required if action is D (field level data mapping), and assists CoolSpools in correctly mapping data to the database. You can specify the following:

****ALPHA***

The XML element contains alphanumeric character data.

DATE

The XML element contains date data.

NUMERIC

The XML element contains numeric data.

TIME

The XML element contains time data.

Display Sequence

A sequence number that controls the order in which elements at the same level (with the same parent element) are presented.

Change XML-to-DBF Map Element screen

This screen allows you to change the database mapping associated with an element within the XML structure.

The screen allows you to modify the following settings:

Action

The database action to be performed when this element is encountered. This can be any one of the following:

N/blank = No action

No database activity will take place.

D = Data

Database field level mapping for this data is to take place. This should be specified for an element that contains a data value. The database file and field names must be specified.

R = Write Record

A Database record will be written. This is usually specified at a parent "container" element. The database file name must be specified.

File Name

The name of the database file to which data is to be written. This is required if action is D (field level data mapping) or R (write file).

Library

The library of the database file to which data is to be written. This is required if action is D (field level data mapping) or R (write file). A library name can be specified, or the special value *LIBL if the file is to be found in the library list at runtime when the conversion is performed. Using *LIBL allows flexibility in use of the same map with different data libraries, but does rely on the library list being set correctly when the conversion is run.

Field

The database field to which the element data is to be mapped. This is only required if action is D (field level data mapping). In addition to a field name, you can specify the following:

***NONE**

Required if action is not D (field level data mapping).

***SELECT**

Presents a list of field names from the database file specified in the file name and library settings. Enter **1=Select** to select the required field.

Data Type

The type of data that is being mapped from the XML document. This is only required if action is D (field level data mapping), and assists CoolSpools in correctly mapping data to the database. You can specify the following:

***ALPHA**

The XML element contains alphanumeric character data.

***DATE**

The XML element contains date data.

***NUMERIC**

The XML element contains numeric data.

***TIME**

The XML element contains time data.

Display Sequence

A sequence number that controls the order in which elements at the same level (with the same parent element) are presented.

Work with XML-to-DBF Map Attributes screen

This screen presents a list of the XML attributes associated with an element. For example, in the XML tag <customer ID=”100”> *customer* is an element and *ID* is an attribute of the *customer* element. The screen provides the following options:

2=Edit DBF

This will take you to the Change XML-to-DBF Map Attribute screen, where you can specify database file and field details.

4=Delete

This will delete the selected attribute.

F6=Add

This will take you to the Add XML-to-DBF Map Attribute screen, where you can create additional XML attributes under the current element.

F11=View 1/2

This will switch between two alternate views. View 1 presents the Attribute name, details of the linked database file and/or field, and whether the Link DBF option is in use. View 2 is the same, but hides the database details in order to present a longer Attribute name – this is useful for working with XML documents that use long Attribute naming.

Add XML-to-DBF Map Attribute screen

This screen allows you to create a new attribute within the XML structure. This may be because an optional attribute was not present in the XML document sampled by BLDXMLDBF, or because you want to create a virtual attribute that does not exist in the source XML, but which is to be populated using the DBF Link option.

The screen allows you to enter the following settings:

Attribute Name

The name of the XML attribute to be added.

Action

The database action to be performed when this attribute is encountered. This can be any one of the following:

N/blank = No action

No database activity will take place.

D = Data

Database field level mapping for this data is to take place. The database file and field names must be specified.

File Name

The name of the database file to which data is to be written. This is only required if action is D (field level data mapping).

Library

The library of the database file to which data is to be written. This is only required if action is D (field level data mapping). A library name can be specified, or the special value *LIBL if the file is to be found in the library list at runtime when the conversion is performed. Using *LIBL allows flexibility in use of the same map with different data libraries, but does rely on the library list being set correctly when the conversion is run.

Field

The database field to which the attribute data is to be mapped. This is only required if action is D (field level data mapping). In addition to a field name, you can specify the following:

***NONE**

Required if action is not D (field level data mapping).

***SELECT**

Presents a list of field names from the database file specified in the file name and library settings. Enter **1=Select** to select the required field.

Data Type

The type of data that is being mapped from the XML document. This is only required if action is D (field level data mapping), and assists CoolSpools in correctly mapping data to the database. You can specify the following:

***ALPHA**

The XML attribute contains alphanumeric character data.

***DATE**

The XML attribute contains date data.

***NUMERIC**

The XML attribute contains numeric data.

***TIME**

The XML attribute contains time data.

Display Sequence

A sequence number that controls the order in which attributes with the same parent element are presented.

Change XML-to-DBF Map Attribute screen

This screen allows you to change the database mapping associated with an attribute within the XML structure.

The screen allows you to enter the following settings:

Action

The database action to be performed when this attribute is encountered. This can be any one of the following:

***N/blank = No action**

No database activity will take place.

D = Data

Database field level mapping for this data is to take place. The database file and field names must be specified.

File Name

The name of the database file to which data is to be written. This is only required if action is D (field level data mapping).

Library

The library of the database file to which data is to be written. This is only required if action is D (field level data mapping). A library name can be specified, or the special value *LIBL if the file is to be found in the library list at runtime when the conversion is performed. Using *LIBL allows flexibility in use of the same map with different data libraries, but does rely on the library list being set correctly when the conversion is run.

Field

The database field to which the attribute data is to be mapped. This is only required if action is D (field level data mapping). In addition to a field name, you can specify the following:

NONE

Required if action is not D (field level data mapping).

SELECT

Presents a list of field names from the database file specified in the file name and library settings. Enter **1=Select** to select the required field.

Data Type

The type of data that is being mapped from the XML document. This is only required if action is D (field level data mapping), and assists CoolSpools in correctly mapping data to the database. You can specify the following:

ALPHA

The XML attribute contains alphanumeric character data.

DATE

The XML attribute contains date data.

NUMERIC

The XML attribute contains numeric data.

TIME

The XML attribute contains time data.

Display Sequence

A sequence number that controls the order in which attributes with the same parent element are presented.

Change DBF Link Screen

The DBF Link functionality allows you to replicate data elements between XML levels. This allows you to reference data at one level in the structure that only appears within an element at a different level in the XML document.

For example, in an XML Order document you may have an order header element with a repeating order line structure as a sub-element. If you want to write the header and line data to separate order header and order line database files then you can do this by specifying the appropriate actions D and R to map the fields and write records to the two files. If order number is required in both the header and line files, and only appears at the header level in the XML document, then to populate order number at line level you can create a virtual order number element at the line level within the map, and then use DBF Link to populate the virtual line level element from the order number in the parent.

When creating a DBF Link specify the option 6=DBF Link against the element or attribute that data is to be copied TO, not against the element that data is being copied FROM. Note that you can create multiple DBF Links from the same element.

The Change DBF Link screen has the following options:

F6=Select From LINK PATH

You will be presented with the top level (root) element of the XML structure. Use options 8=Elements and 9=Attributes to navigate through the XML structure, then use 1=Select to select the Element or Attribute from which data is to be copied.

F23=REMOVE Link

Clear an existing DBF Link from the selected element or attribute.

IMPXMLDBF (Import XML to Database) Command

CoolSpools XML-to-DBF is a tool for importing XML data into IBM i database files. The conversion process uses a defined XML-to-Database Map to translate the content of an XML document to database files and fields.

The Import XML to Database (IMPXMLDBF) command parses an XML document, extracting the data and populating one or more IBM i database files as specified within a defined XML-to-Database Map, created using the BLDXMLDBF and WRKXMLDBF commands.

The location of the source XML document must be specified, but the database files into which the data will be placed are defined within the XML-to-Database Map.

The command parameters are as follows:

PATH – XML Document

The PATH parameter specifies the location on the IFS of the XML document to be imported to IBM i Database files as defined in the map. This must be an existing XML document, and the XML structure should match that defined within the mapping.

MAPNAME – Map Name

Name of the XML-to-Database Map to be used in the mapping of data from XML elements and attributes to database files and fields. This must be an existing XML-to-Database Map and the structure of the map should match that of the XML document.

DLTXMLDBF (Delete XML to Database Map) Command

CoolSpools XML-to-DBF is a tool for importing XML data into IBM i database files. The conversion process uses a defined XML-to-Database Map to translate the content of an XML document to database files and fields.

The Delete XML-to-Database Map (DLTXMLDBF) command deletes the specified XML-to-Database Map including all of the element and attribute definitions, and any database file/field mappings.

Once the map has been deleted it can no longer be used by the IMPXMLDBF command to import data from XML to IBM i database files.

The command parameters are as follows:

MAPNAME – Map Name

Name of the XML-to-Database Map to be deleted.

SAVXMLDBF (Save XML to Database Map) Command

CoolSpools XML-to-DBF is a tool for importing XML data into IBM i database files. The conversion process uses a defined XML-to-Database Map to translate the content of an XML document to database files and fields.

The Save XML-to-Database Map (SAVXMLDBF) command extracts map data, including the XML element structure and the database file/field level mappings, and populates a stream file document on the IFS. This stream file document can be retained as a backup or for version control purposes, or can be transferred to another IBM i server and used to recreate the map. For example, to transfer a tested XML-to-database map from a development or UAT environment to a production system.

The command parameters are as follows:

MAPNAME – Map Name

Name of the XML-to-Database Map to be saved.

TOSTMF – Save to Stream file name

Specify an IFS path for the stream file to which the XML-to-database map should be saved. The stream file that is created is an XML document.

Options are:

***MAPNAME**	A stream file name of *mapname*.xls will be created in the job's current directory, where *mapname* is the name of the map being saved.
Stream file	Specify the path of the stream file.

REPLACE – Replace stream file

Should the stream file be replaced if it already exists?

Options are:

***NO**	An error will be reported in the stream file already exists.
***YES**	The stream file will be replaced if it already exists.

RSTXMLDBF (Restore XML to Database Map) Command

CoolSpools XML-to-DBF is a tool for importing XML data into IBM i database files. The conversion process uses a defined XML-to-Database Map to translate the content of an XML document to database files and fields.

The Restore XML-to-Database Map (RSTXMLDBF) command restores an XML-to-Database Map from a stream file document that was previously created by using the corresponding Save XML-to-Database Map (SAVXMLDBF) command.

The command parameters are as follows:

MAPNAME – Map Name

Name of the XML-to-Database Map that was saved to the stream file.

FROMSTMF – Restore from Stream file name

Specify an IFS path of an existing stream file to that was created using the SAVXMLDBF command.

RSTASMAP – Restore as Map Name

Name of the XML-to-Database Map that is to be created from the content of the stream file. By default this is the same as the map name that was saved to the stream file.

Options are:

***MAPNAME**	Restore using the same map name as was saved.
Map name	Specify a new map name to which the saved map is to be restored.

Worked Examples

Example database

Imagine the following simple order database.

Customer file CUSTOMER:

```
                R CUSTOMERR                    TEXT('Customer Record')
                  CUSTNBR         9B 0         TEXT('Customer number')
                                               COLHDG('Customer' 'Number')
                                               ALIAS(CUSTOMER_NUMBER)
                  NAME           30A           TEXT('Customer name')
                                               COLHDG('Customer' 'Name')
                                               ALIAS(CUSTOMER_NAME)
```

Order header file ORDERHDR:

```
                R ORDERHDRR                    TEXT('Order Header Record')
                  ORDERNBR        9B 0         TEXT('Order number')
                                               COLHDG('Order' 'Number')
                                               ALIAS(ORDER_NUMBER)
                  CUSTNBR         9B 0         TEXT('Customer number')
                                               COLHDG('Customer' 'Number')
                                               ALIAS(CUSTOMER_NUMBER)
                  ORDERDATE        L           TEXT('Order date')
                                               COLHDG('Order' 'Date')
                                               ALIAS(ORDER_DATE)
```

Order detail file ORDERDTL:

```
                R ORDERDTLR                    TEXT('Order Detail Record')
                  ORDERNBR        9B 0         TEXT('Order number')
                                               COLHDG('Order' 'Number')
                                               ALIAS(ORDER_NUMBER)
                  ORDERLINE       4B 0         TEXT('Order line number')
                                               COLHDG('Order' 'Line'
'Number')
                                               ALIAS(ORDER_LINE_NUMBER)
                  ITEMNBR         9B 0         TEXT('Item number')
                                               COLHDG('Item' 'Number')
                                               ALIAS(ITEM_NUMBER)
                  QUANTITY       11P 4         TEXT('Quantity ordered')
                                               COLHDG('Quantity' 'Ordered')
                                               ALIAS(QUANTITY_ORDERED)
```

Item file ITEM:

```
                R ITEMR                        TEXT('Item Record')
                  ITEMNBR         9B 0         TEXT('Item number')
                                               COLHDG('Item' 'Number')
                                               ALIAS(ITEM_NUMBER)
                  ITEMDESC       30A           TEXT('Item description')
                                               COLHDG('Item' 'Description')
                                               ALIAS(ITEM_DESCRIPTION)
                  UOM             2A           TEXT('Unit of measure')
                                               COLHDG('UOM')
```

```
                                        ALIAS(UNIT_OF_MEASURE)
                  COST          9P 2    TEXT('Item cost')
                                        COLHDG('Item' 'Cost')
                                        ALIAS(ITEM_COST)
                  PRICE         9P 2    TEXT('Selling price')
                                        COLHDG('Selling' 'Price')
                                        ALIAS(SELLING_PRICE)
```

Database-to-Excel Map

Assume source member ORDERS in source file QSQLSRC contains the following SQL statement:

```
SELECT   h.ORDER_NUMBER,
         h.ORDER_DATE,
         c.CUSTOMER_NUMBER,
         c.CUSTOMER_NAME,
         d.QUANTITY,
         i.ITEM_NUMBER,
         i.ITEM_DESCRIPTION,
         i.SELLING_PRICE,
         d.QUANTITY*i.SELLING_PRICE as LINE_AMOUNT
FROM     ORDERHDR h, ORDERDTL d, CUSTOMER c, ITEM i
WHERE    h.ORDER_NUMBER = d.ORDER_NUMBER
AND      h.CUSTOMER_NUMBER = c.CUSTOMER_NUMBER
AND      d.ITEM_NUMBER = i.ITEM_NUMBER
ORDER BY h.ORDER_NUMBER, d.ORDER_LINE_NUMBER
```

The following commands create a database-to-Excel map for use with this SQL statement.

1. Define map

CRTDBFXL
    MAPNAME(ORDERS)
    INPUT(*SQLSRC)
    SQLSRC(QSQLSRC ORDERS)
    TEXT('Order map')

2. Define customer line

ADDDBFXLR
    MAPNAME(ORDERS)
    ROWGRPNAME(CUSTOMER_LINE)
    TEXT('Customer line')
    NEWGRPOPT(CUSTOMER_NUMBER)

3. Add label for customer

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(CUSTOMER_LINE)
    ROWNBR(1)
    COLUMN(A)
    CONTENT(*TEXT)
    CELLTEXT('Customer:')

4. Add customer number

ADDDBFXLC

```
    MAPNAME(ORDERS)
    ROWGRPNAME(CUSTOMER_LINE)
    ROWNBR(1)
    COLUMN(B)
    CONTENT(*COLUMN)
    CELLCOLUMN(CUSTOMER_NUMBER)
```

5. Add customer name

```
ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(CUSTOMER_LINE)
    ROWNBR(1)
    COLUMN(C)
    CONTENT(*COLUMN)
    CELLCOLUMN(CUSTOMER_NAME)
```

6. Define unused columns as empty

```
ADDDBFXLC
    MAPNAME(&MAPNAME)
    ROWGRPNAME(CUSTOMER_LINE)
    ROWNBR(1)
    COLUMN(D)
    CONTENT(*EMPTY)
    MRGCELLS(1 G)
```

7. Define order header row group

```
ADDDBFXLR
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_HEADER)
    PARENT(CUSTOMER_LINE)
    TEXT('Order line')
    NEWGRPOPT(ORDER_NUMBER)
```

8. Add label for order number

```
ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_HEADER)
    ROWNBR(1)
    COLUMN(A)
    CONTENT(*TEXT)
    CELLTEXT('Order number:')
```

9. Add order number

```
ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_HEADER)
    ROWNBR(1)
    COLUMN(B)
    CONTENT(*COLUMN)
    CELLCOLUMN(ORDER_NUMBER)
```

10. Add label for order date

```
ADDDBFXLC
     MAPNAME(ORDERS)
     ROWGRPNAME(ORDER_HEADER)
     ROWNBR(1)
     COLUMN(C)
     CONTENT(*TEXT)
     CELLTEXT('Order date:')
```

11. Add order date

```
ADDDBFXLC
     MAPNAME(ORDERS)
     ROWGRPNAME(ORDER_HEADER)
     ROWNBR(1)
     COLUMN(D)
     CONTENT(*COLUMN)
     CELLCOLUMN(ORDER_DATE)
```

12. Define unused columns as empty

```
ADDDBFXLC
     MAPNAME(&MAPNAME)
     ROWGRPNAME(ORDER_HEADER)
     ROWNBR(1)
     COLUMN(E)
     CONTENT(*EMPTY)
     MRGCELLS(1 G)
```

13. Define order detail column headings

```
ADDDBFXLR
     MAPNAME(ORDERS)
     ROWGRPNAME(ORDER_COL_HEADINGS)
     PARENT(ORDER_HEADER)
     TEXT('Order detail column headings')
     NEWGRPOPT(*NEVER)
```

14. Add column headings

```
ADDDBFXLC
     MAPNAME(ORDERS)
     ROWGRPNAME(ORDER_COL_HEADINGS)
     ROWNBR(1)
     COLUMN(A)
     CONTENT(*TEXT)
     CELLTEXT('Line number')

ADDDBFXLC
     MAPNAME(ORDERS)
     ROWGRPNAME(ORDER_COL_HEADINGS)
     ROWNBR(1)
     COLUMN(B)
```

```
    CONTENT(*TEXT)
    CELLTEXT('Item number')

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_COL_HEADINGS)
    ROWNBR(1)
    COLUMN(C)
    CONTENT(*TEXT)
    CELLTEXT('Description')

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_COL_HEADINGS)
    ROWNBR(1)
    COLUMN(D)
    CONTENT(*TEXT)
    CELLTEXT('Quantity')

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_COL_HEADINGS)
    ROWNBR(1)
    COLUMN(E)
    CONTENT(*TEXT)
    CELLTEXT('UOM')

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_COL_HEADINGS)
    ROWNBR(1)
    COLUMN(F)
    CONTENT(*TEXT)
    CELLTEXT('Price')

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_COL_HEADINGS)
    ROWNBR(1)
    COLUMN(G)
    CONTENT(*TEXT)
    CELLTEXT('Line amount')
```

15. Define order detail line

```
ADDDBFXLR
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_ITEM_LINE)
    PARENT(ORDER_HEADER)
    TEXT('Order item line')
```

```
    NEWGRPOPT(*RCD)

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_ITEM_LINE)
    ROWNBR(1)
    COLUMN(A)
    CONTENT(*COLUMN)
    CELLCOLUMN(ORDER_LINE_NUMBER)

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_ITEM_LINE)
    ROWNBR(1)
    COLUMN(B)
    CONTENT(*COLUMN)
    CELLCOLUMN(ITEM_NUMBER)

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_ITEM_LINE)
    ROWNBR(1)
    COLUMN(C)
    CONTENT(*COLUMN)
    CELLCOLUMN(ITEM_DESCRIPTION)

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_ITEM_LINE)
    ROWNBR(1)
    COLUMN(D)
    CONTENT(*COLUMN)
    CELLCOLUMN(QUANTITY)

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_ITEM_LINE)
    ROWNBR(1)
    COLUMN(E)
    CONTENT(*COLUMN)
    CELLCOLUMN(UOM)

ADDDBFXLC
    MAPNAME(ORDERS)
    ROWGRPNAME(ORDER_ITEM_LINE)
    ROWNBR(1)
    COLUMN(F)
    CONTENT(*COLUMN)
    CELLCOLUMN(PRICE)
```

```
ADDDBFXLC
      MAPNAME(ORDERS)
      ROWGRPNAME(ORDER_ITEM_LINE)
      ROWNBR(1)
      COLUMN(G)
      CONTENT(*COLUMN)
      CELLCOLUMN(LINE_AMOUNT)
```

16. Define styles

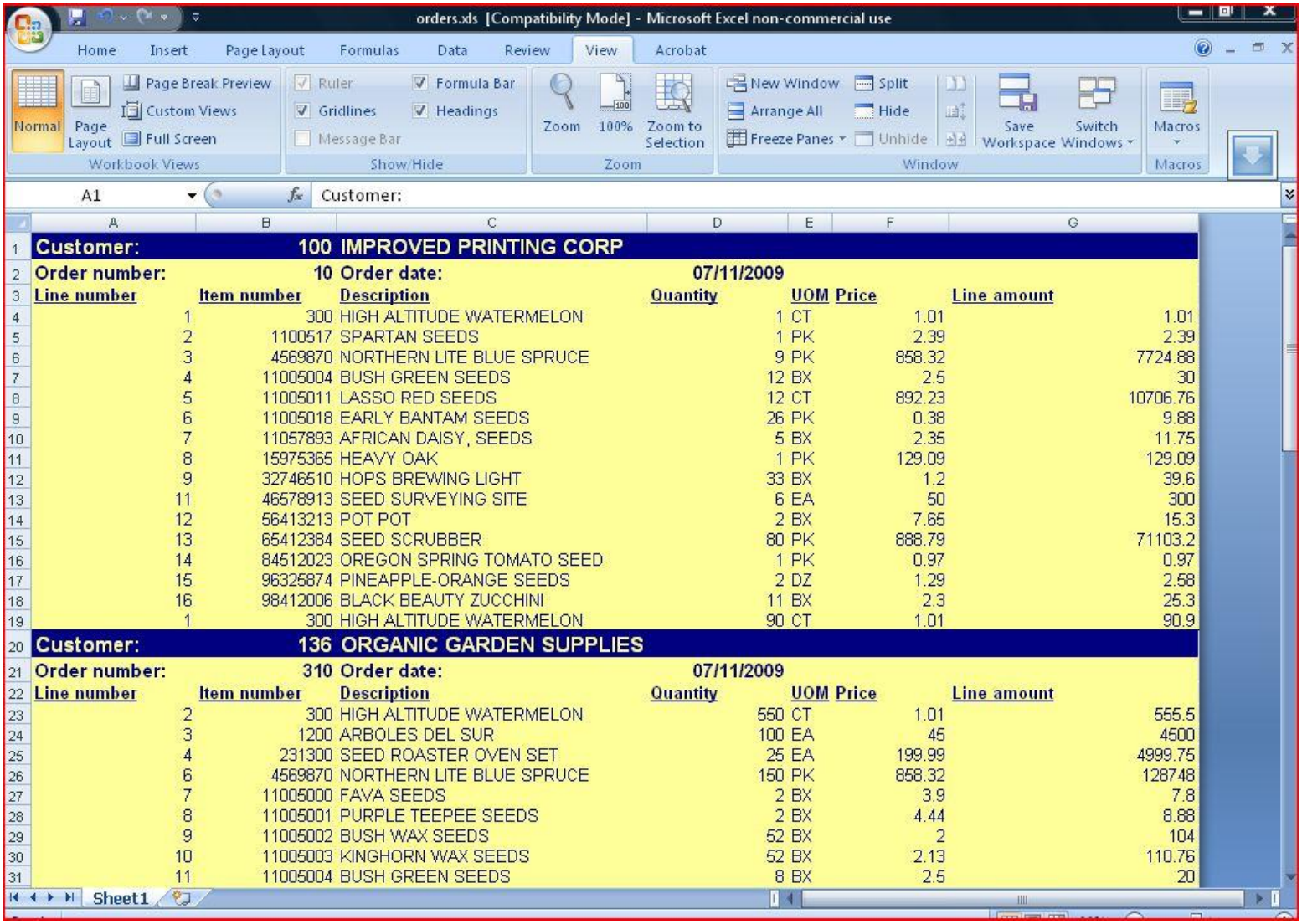
```
CRTSTLDFN
      STYLENAME(CUSTOMER_LINE)
      TEXT('Customer line')
      FONTSIZE(16)
      BOLD(*YES)
      TEXTCOLOR(*LIGHTYELLOW)
      BACKCOLOR(*DARKBLUE)

CRTSTLDFN
      STYLENAME(ORDER_HEADER)
      TEXT('Order line')
      FONTSIZE(14)
      BOLD(*YES)
      TEXTCOLOR(*DARKBLUE)
      BACKCOLOR(*LIGHTYELLOW)

CRTSTLDFN
      STYLENAME(ORDER_COL_HEADINGS)
      TEXT('Order column headings')
      FONTSIZE(12)
      BOLD(*YES)
      UNDERLINE(*SINGLE)
      TEXTCOLOR(*DARKBLUE)
      BACKCOLOR(*LIGHTYELLOW)

CRTSTLDFN
      STYLENAME(ORDER_ITEM_LINE)
      TEXT('Order item line')
      FONTSIZE(12)
      BOLD(*NO)
      TEXTCOLOR(*DARKBLUE)
      BACKCOLOR(*LIGHTYELLOW)
```

17. Use the map to create a spreadsheet

```
CVTDBFXL
        FROMFILE(*MAP)
        MAPNAME(ORDERS)
        TOSTMF(orders.xls)
        STMFOPT(*REPLACE)
```

```
EXCEL(*XLS *MM *HMS *DFT *FIELDSIZE *XLSVER *YES *YES)
```

The resultant spreadsheet should look something like this.



Database-to-XML Map

Using the same database and SQL statement as the above example, the following commands create a database-to-XML map for use with this SQL statement.

1. Define map

```
CRTDBFXML
      MAPNAME(&MAPNAME)
      INPUT(*SQLSRC)
      SQLSRC(QSQLSRC ORDERS)
      TEXT('Order map')
```

2. Define root element

```
ADDDBFXMLE
      MAPNAME(&MAPNAME)
      ELEMENT(customerOrders)
      PARENT(*NONE)
      TEXT('Root element for customer order list')
      NEWELMOPT(*NEVER)
```

3. Define customer element

```
ADDDBFXMLE
```

```
    MAPNAME(&MAPNAME)
    ELEMENT(customer)
    PARENT(customerOrders)
    TEXT('Customer element')
    NEWELMOPT(CUSTOMER_NUMBER)
```

4. Add customer number attribute

```
ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT('customer')
    ATTRIBUTE('ID')
    COLUMN(CUSTOMER_NUMBER)
    TEXT('Customer number')
```

5. Add customer name attribute

```
ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT('customer')
    ATTRIBUTE('name')
    COLUMN(CUSTOMER_NAME)
    TEXT('Customer name')
```

6. Define order element

```
ADDDBFXMLE
    MAPNAME(&MAPNAME)
    ELEMENT(order)
    PARENT(customer)
    TEXT('Order header')
    NEWELMOPT(ORDER_NUMBER)
```

7. Add order number

```
ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT(order)
    ATTRIBUTE('ID')
    COLUMN(ORDER_NUMBER)
    TEXT('Order number')
```

8. Add order date

```
ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT(order)
    ATTRIBUTE('date')
    COLUMN(ORDER_DATE)
    TEXT('Order date')
```

9. Define order item element

```
ADDDBFXMLE
    MAPNAME(&MAPNAME)
    ELEMENT(orderItem)
```

```
    PARENT(order)
    TEXT('Order item element')
    NEWELMOPT(*RCD)

ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT(orderItem)
    ATTRIBUTE('lineNumber')
    COLUMN(ORDER_LINE_NUMBER)
    TEXT('Order line number')

ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT(orderItem)
    ATTRIBUTE('itemNumber')
    COLUMN(ITEM_NUMBER)
    TEXT('Item number')

ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT(orderItem)
    ATTRIBUTE('itemDescription')
    COLUMN(ITEM_DESCRIPTION)
    TEXT('Item description')

ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT(orderItem)
    ATTRIBUTE('quantity')
    COLUMN(QUANTITY)
    TEXT('Quantity')

ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT(orderItem)
    ATTRIBUTE('UOM')
    COLUMN(UOM)
    TEXT('UOM')

ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT(orderItem)
    ATTRIBUTE('price')
    COLUMN(PRICE)
    TEXT('Price')

ADDDBFXMLA
    MAPNAME(&MAPNAME)
    ELEMENT(orderItem)
```

ATTRIBUTE('amount')
COLUMN(LINE_AMOUNT)
TEXT('Line amount')

10. Use the map to create a spreadsheet

CVTDBFXML
FROMFILE(*MAP)
MAPNAME(ORDERS)
TOSTMF(orders.xml)
STMFOPT(*REPLACE)

The resultant XML document should look something like this.

```
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="orders.XSLT"?>

<customerOrders xml:space="preserve">

	<customer ID="100" name="IMPROVED PRINTING CORP">

		<order ID="10" date="2009-11-07">

			<orderItem lineNumber="1" itemNumber="300" itemDescription="HIGH ALTITUDE
			WATERMELON" quantity="1.0000" UOM="CT" price="1.01" amount="1.010000"/>

			<orderItem lineNumber="2" itemNumber="1100517" itemDescription="SPARTAN SEEDS"
			quantity="1.0000" UOM="PK" price="2.39" amount="2.390000"/>

			<orderItem lineNumber="3" itemNumber="4569870" itemDescription="NORTHERN LITE
			BLUE SPRUCE" quantity="9.0000" UOM="PK" price="858.32" amount="7724.880000"/>

			<orderItem lineNumber="4" itemNumber="11005004" itemDescription="BUSH GREEN
			SEEDS" quantity="12.0000" UOM="BX" price="2.50" amount="30.000000"/>

			<orderItem lineNumber="5" itemNumber="11005011" itemDescription="LASSO RED
			SEEDS" quantity="12.0000" UOM="CT" price="892.23" amount="10706.760000"/>

			<orderItem lineNumber="6" itemNumber="11005018" itemDescription="EARLY BANTAM
			SEEDS" quantity="26.0000" UOM="PK" price=".38" amount="9.880000"/>

			<orderItem lineNumber="7" itemNumber="11057893" itemDescription="AFRICAN DAISY,
			SEEDS" quantity="5.0000" UOM="BX" price="2.35" amount="11.750000"/>

			<orderItem lineNumber="8" itemNumber="15975365" itemDescription="HEAVY OAK"
			quantity="1.0000" UOM="PK" price="129.09" amount="129.090000"/>

			<orderItem lineNumber="9" itemNumber="32746510" itemDescription="HOPS BREWING
			LIGHT" quantity="33.0000" UOM="BX" price="1.20" amount="39.600000"/>

			<orderItem lineNumber="11" itemNumber="46578913" itemDescription="SEED
			SURVEYING SITE" quantity="6.0000" UOM="EA" price="50.00" amount="300.000000"/>

			<orderItem lineNumber="12" itemNumber="56413213" itemDescription="POT POT"
			quantity="2.0000" UOM="BX" price="7.65" amount="15.300000"/>

			<orderItem lineNumber="13" itemNumber="65412384" itemDescription="SEED
			SCRUBBER" quantity="80.0000" UOM="PK" price="888.79" amount="71103.200000"/>

			<orderItem lineNumber="14" itemNumber="84512023" itemDescription="OREGON SPRING
			TOMATO SEED" quantity="1.0000" UOM="PK" price=".97" amount=".970000"/>

			<orderItem lineNumber="15" itemNumber="96325874" itemDescription="PINEAPPLE-
			ORANGE SEEDS" quantity="2.0000" UOM="DZ" price="1.29" amount="2.580000"/>

			<orderItem lineNumber="16" itemNumber="98412006" itemDescription="BLACK BEAUTY
			ZUCCHINI" quantity="11.0000" UOM="BX" price="2.30" amount="25.300000"/>

			<orderItem lineNumber="17" itemNumber="98546320" itemDescription="FROZEN JUICE
			PROCESSOR" quantity="5.0000" UOM="EA" price="109.90" amount="549.500000"/>

		</order>

	</customer>

	<customer ID="136" name="ORGANIC GARDEN SUPPLIES">

		<order ID="310" date="2009-11-07">
```

<orderItem lineNumber="1" itemNumber="300" itemDescription="HIGH ALTITUDE WATERMELON" quantity="90.0000" UOM="CT" price="1.01" amount="90.900000"/>

<orderItem lineNumber="2" itemNumber="300" itemDescription="HIGH ALTITUDE WATERMELON" quantity="550.0000" UOM="CT" price="1.01" amount="555.500000"/>

<orderItem lineNumber="3" itemNumber="1200" itemDescription="ARBOLES DEL SUR" quantity="100.0000" UOM="EA" price="45.00" amount="4500.000000"/>

<orderItem lineNumber="4" itemNumber="231300" itemDescription="SEED ROASTER OVEN SET" quantity="25.0000" UOM="EA" price="199.99" amount="4999.750000"/>

<orderItem lineNumber="6" itemNumber="4569870" itemDescription="NORTHERN LITE BLUE SPRUCE" quantity="150.0000" UOM="PK" price="858.32" amount="128748.000000"/>

<orderItem lineNumber="7" itemNumber="11005000" itemDescription="FAVA SEEDS" quantity="2.0000" UOM="BX" price="3.90" amount="7.800000"/>

<orderItem lineNumber="8" itemNumber="11005001" itemDescription="PURPLE TEEPEE SEEDS" quantity="2.0000" UOM="BX" price="4.44" amount="8.880000"/>

<orderItem lineNumber="9" itemNumber="11005002" itemDescription="BUSH WAX SEEDS" quantity="52.0000" UOM="BX" price="2.00" amount="104.000000"/>

<orderItem lineNumber="10" itemNumber="11005003" itemDescription="KINGHORN WAX SEEDS" quantity="52.0000" UOM="BX" price="2.13" amount="110.760000"/>

<orderItem lineNumber="11" itemNumber="11005004" itemDescription="BUSH GREEN SEEDS" quantity="8.0000" UOM="BX" price="2.50" amount="20.000000"/>

<orderItem lineNumber="12" itemNumber="11005005" itemDescription="BLUE LAKE GREEN SEEDS" quantity="8.0000" UOM="BX" price="4.00" amount="32.000000"/>

<orderItem lineNumber="13" itemNumber="11005006" itemDescription="KINGHORN WAX SEEDS" quantity="2.0000" UOM="BX" price="3.00" amount="6.000000"/>

<orderItem lineNumber="14" itemNumber="11005007" itemDescription="VENTURE GREEN SEEDS" quantity="2.0000" UOM="CT" price="1.50" amount="3.000000"/>

<orderItem lineNumber="15" itemNumber="11005008" itemDescription="NORTHEASTERN POLE SEEDS" quantity="100.0000" UOM="CT" price="1.29" amount="129.000000"/>

<orderItem lineNumber="16" itemNumber="11005009" itemDescription="KENTUCKY BLUE SEEDS" quantity="100.0000" UOM="CT" price="2.10" amount="210.000000"/>

<orderItem lineNumber="17" itemNumber="11005010" itemDescription="EARLY DWARF DANISH SEEDS" quantity="58.0000" UOM="CT" price="3.01" amount="174.580000"/>

<orderItem lineNumber="18" itemNumber="11005011" itemDescription="LASSO RED SEEDS" quantity="58.0000" UOM="CT" price="892.23" amount="51749.340000"/>

<orderItem lineNumber="19" itemNumber="11005012" itemDescription="BLUE MAX SAVOY BEANS" quantity="84.0000" UOM="EA" price="1.23" amount="103.320000"/>

<orderItem lineNumber="20" itemNumber="11005013" itemDescription="MINCOR NANTES CARROT SEED" quantity="84.0000" UOM="DZ" price=".87" amount="73.080000"/>

<orderItem lineNumber="21" itemNumber="11005014" itemDescription="SCARLET NANTES SEEDS" quantity="10.0000" UOM="DZ" price="5.90" amount="59.000000"/>

<orderItem lineNumber="22" itemNumber="11005014" itemDescription="SCARLET NANTES SEEDS" quantity="5.0000" UOM="DZ" price="5.90" amount="29.500000"/>

<orderItem lineNumber="23" itemNumber="11005015" itemDescription="CHANTENAY SEEDS" quantity="10.0000" UOM="BZ" price="2.19" amount="21.900000"/>

<orderItem lineNumber="24" itemNumber="11005016" itemDescription="TOUCHON SEEDS" quantity="63.0000" UOM="BZ" price="2.83" amount="178.290000"/>

<orderItem lineNumber="25" itemNumber="11005016" itemDescription="TOUCHON SEEDS" quantity="65.0000" UOM="BZ" price="2.83" amount="183.950000"/>

<orderItem lineNumber="27" itemNumber="11005018" itemDescription="EARLY BANTAM SEEDS" quantity="2.0000" UOM="PK" price=".38" amount=".760000"/>

<orderItem lineNumber="28" itemNumber="11005019" itemDescription="NORTHERN PICKLING SEEDS" quantity="2.0000" UOM="PK" price=".39" amount=".780000"/>

<orderItem lineNumber="29" itemNumber="11005020" itemDescription="FRENCH PICKLING SEEDS" quantity="90.0000" UOM="PK" price="2.39" amount="215.100000"/>

<orderItem lineNumber="30" itemNumber="11057893" itemDescription="AFRICAN DAISY, SEEDS" quantity="100.0000" UOM="BX" price="2.35" amount="235.000000"/>

<orderItem lineNumber="31" itemNumber="12382910" itemDescription="SUCCATASH SEEDS" quantity="25.0000" UOM="CT" price=".38" amount="9.500000"/>

```
                <orderItem lineNumber="32" itemNumber="13145340" itemDescription="SOUR GRAPE
                SEEDS" quantity="45.0000" UOM="CT" price=".15" amount="6.750000"/>

                <orderItem lineNumber="33" itemNumber="15789342" itemDescription="BLUE BELLES,
                BRIGHT BLUE" quantity="10.0000" UOM="PT" price="18.57" amount="185.700000"/>

                <orderItem lineNumber="34" itemNumber="15975365" itemDescription="HEAVY OAK"
                quantity="50.0000" UOM="PK" price="129.09" amount="6454.500000"/>

                <orderItem lineNumber="35" itemNumber="31321654" itemDescription="BELLSTAR
                SEEDS" quantity="25.0000" UOM="EA" price="7.88" amount="197.000000"/>

                <orderItem lineNumber="36" itemNumber="31321654" itemDescription="BELLSTAR
                SEEDS" quantity="2.0000" UOM="EA" price="7.88" amount="15.760000"/>

                <orderItem lineNumber="37" itemNumber="32154657" itemDescription="PETERSBURG
                PALM TREE" quantity="25.0000" UOM="DZ" price="34.90" amount="872.500000"/>

                <orderItem lineNumber="38" itemNumber="32165478" itemDescription="BLACK EYED
                BANANA" quantity="6.0000" UOM="BZ" price="3.01" amount="18.060000"/>

                <orderItem lineNumber="39" itemNumber="32746510" itemDescription="HOPS BREWING
                LIGHT" quantity="45.0000" UOM="BX" price="1.20" amount="54.000000"/>

                <orderItem lineNumber="40" itemNumber="35456031" itemDescription="SUNNY
                SUNFLOWER SEEDS" quantity="10.0000" UOM="CT" price="1.23" amount="12.300000"/>

                <orderItem lineNumber="41" itemNumber="35715924" itemDescription="SEED SIFTER
                SET" quantity="50.0000" UOM="EA" price="2900.00" amount="145000.000000"/>

                <orderItem lineNumber="42" itemNumber="40113254" itemDescription="FRESH FRUIT
                CANNED CANNER" quantity="18.0000" UOM="EA" price="22.97" amount="413.460000"/>

                <orderItem lineNumber="43" itemNumber="56413213" itemDescription="POT POT"
                quantity="6.0000" UOM="BX" price="7.65" amount="45.900000"/>

                <orderItem lineNumber="44" itemNumber="64132029" itemDescription="PITLESS PEACH
                SEEDS" quantity="1000.0000" UOM="PK" price=".97" amount="970.000000"/>

                <orderItem lineNumber="45" itemNumber="90978412" itemDescription="TREE TRIMMER
                TUBING" quantity="500.0000" UOM="EA" price=".20" amount="100.000000"/>

                <orderItem lineNumber="46" itemNumber="94875081" itemDescription="EARLIROUGE
                TOMATO SEEDS" quantity="6.0000" UOM="CT" price=".49" amount="2.940000"/>

                <orderItem lineNumber="47" itemNumber="98412006" itemDescription="BLACK BEAUTY
                ZUCCHINI" quantity="45.0000" UOM="BX" price="2.30" amount="103.500000"/>

                <orderItem lineNumber="48" itemNumber="98546320" itemDescription="FROZEN JUICE
                PROCESSOR" quantity="5.0000" UOM="EA" price="109.90" amount="549.500000"/>

        </order>

</customer>

<customer ID="141" name="LOS ARBOLES DEL MUNDO">

        <order ID="930" date="2009-11-07">

                <orderItem lineNumber="1" itemNumber="1200" itemDescription="ARBOLES DEL SUR"
                quantity="900.0000" UOM="EA" price="45.00" amount="40500.000000"/>

                <orderItem lineNumber="5" itemNumber="11005011" itemDescription="LASSO RED
                SEEDS" quantity="951.0000" UOM="CT" price="892.23" amount="848510.730000"/>

                <orderItem lineNumber="6" itemNumber="11005014" itemDescription="SCARLET NANTES
                SEEDS" quantity="46.0000" UOM="DZ" price="5.90" amount="271.400000"/>

                <orderItem lineNumber="7" itemNumber="11005015" itemDescription="CHANTENAY
                SEEDS" quantity="45.0000" UOM="BZ" price="2.19" amount="98.550000"/>

                <orderItem lineNumber="8" itemNumber="11005018" itemDescription="EARLY BANTAM
                SEEDS" quantity="951.0000" UOM="PK" price=".38" amount="361.380000"/>

                <orderItem lineNumber="11" itemNumber="11057893" itemDescription="AFRICAN
                DAISY, SEEDS" quantity="4.0000" UOM="BX" price="2.35" amount="9.400000"/>

                <orderItem lineNumber="13" itemNumber="31321655" itemDescription="SEMILLAS DEL
                SUS SOMBEROS" quantity="100.0000" UOM="EA" price="24.95" amount="2495.000000"/>

                <orderItem lineNumber="15" itemNumber="56413213" itemDescription="POT POT"
                quantity="1000.0000" UOM="BX" price="7.65" amount="7650.000000"/>

                <orderItem lineNumber="17" itemNumber="84512023" itemDescription="OREGON SPRING
                TOMATO SEED" quantity="98.0000" UOM="PK" price=".97" amount="95.060000"/>

        </order>

</customer>
```

```
        <customer ID="154" name="THE LAST LEAF">
                <order ID="1110" date="2009-11-07">
                        <orderItem lineNumber="1" itemNumber="231300" itemDescription="SEED ROASTER
                        OVEN SET" quantity="9.0000" UOM="EA" price="199.99" amount="1799.910000"/>
                        <orderItem lineNumber="3" itemNumber="3698741" itemDescription="STRING
                        GRAPEFRUIT" quantity="4.0000" UOM="PK" price="2.01" amount="8.040000"/>
                        <orderItem lineNumber="5" itemNumber="11000146" itemDescription="AZALIA, GIANT
                        ROSE SEEDS" quantity="300.0000" UOM="EA" price=".55" amount="165.000000"/>
                        <orderItem lineNumber="7" itemNumber="11005010" itemDescription="EARLY DWARF
                        DANISH SEEDS" quantity="6.0000" UOM="CT" price="3.01" amount="18.060000"/>
                        <orderItem lineNumber="8" itemNumber="11005013" itemDescription="MINCOR NANTES
                        CARROT SEED" quantity="24.0000" UOM="DZ" price=".87" amount="20.880000"/>
                        <orderItem lineNumber="10" itemNumber="11005020" itemDescription="FRENCH
                        PICKLING SEEDS" quantity="24.0000" UOM="PK" price="2.39" amount="57.360000"/>
                        <orderItem lineNumber="11" itemNumber="12382910" itemDescription="SUCCATASH
                        SEEDS" quantity="12.0000" UOM="CT" price=".38" amount="4.560000"/>
                        <orderItem lineNumber="13" itemNumber="13145340" itemDescription="SOUR GRAPE
                        SEEDS" quantity="55.0000" UOM="CT" price=".15" amount="8.250000"/>
                        <orderItem lineNumber="15" itemNumber="32165478" itemDescription="BLACK EYED
                        BANANA" quantity="14.0000" UOM="BZ" price="3.01" amount="42.140000"/>
                        <orderItem lineNumber="17" itemNumber="44646510" itemDescription="PLUMP RED
                        PLUMS" quantity="600.0000" UOM="DZ" price=".49" amount="294.000000"/>
                        <orderItem lineNumber="19" itemNumber="45613712" itemDescription="CRANAPPLE
                        BERRY SEEDS" quantity="40.0000" UOM="DZ" price="1.28" amount="51.200000"/>
                        <orderItem lineNumber="21" itemNumber="65412384" itemDescription="SEED
                        SCRUBBER" quantity="4.0000" UOM="PK" price="888.79" amount="3555.160000"/>
                        <orderItem lineNumber="25" itemNumber="84512130" itemDescription="SUB-ARTIC
                        TOMATO SEEDS" quantity="12.0000" UOM="DZ" price="3.84" amount="46.080000"/>
                </order>
        </customer>
</customerOrders>
```

Using Database-to-XML Maps: A Real-World Example

What follows is a description of a real-world implementation of a Database-to-XML map, one which ariadne software uses in its own internal accounting systems.

UK businesses that are registered for VAT (Value Added Tax) and which supply goods and certain services to a VAT-registered customers in other European Union (EU) countries are obliged to notify UK Revenue & Customs (HMRC) about those transactions so VAT can be collected by the tax authorities in the customer's own country.

HMRC requires the periodic submission of an EC Sales List (ESL) showing details of sales to such customers in the EU and the value (in sterling) of the supplies made to them in the period. There is an option to provide this information to HMRC in XML format electronically.

HMRC provides the following specification for this XML bulk upload file.

The XML structure required is as follows:

```
<Submission type='HMRC_VAT_ESL_BULK_SUBMISSION_FILE'>
        <TraderVRN></TraderVRN>
        <Branch></Branch>
        <Year></Year>
        <Period></Period>
        <CurrencyA3></CurrencyA3>
        <ContactName></ContactName>
        <Online></Online>
        <SubmissionLines>
                <SubmissionLine>
                        <CountryA2>/CountryA2>
                        <CustomerVRN></CustomerVRN>
                        <Value></Value>
                        <Indicator></Indicator>
                </SubmissionLine>
        </SubmissionLines>
</Submission>
```

Node explanation and contents:

<Submission>

This is the root node of the XML. It must contain a type attribute set to 'HMRC_VAT_ESL_BULK_SUBMISSION_FILE'

<TraderVRN>

Must be a valid UK VAT Registration Number (VRN). The VRN must match the VRN used by the trader on logging into the ECSL Application.

<Branch>

Must be three-digit numeric.

<Year>

Must be four-digit numeric.

<Period>

Must be two-digit numeric representing a month.

For quarterly traders this will be one of:

03, 06, 09, 12 (corresponding to the old quarter identifiers 1,2,3,4)

For monthly traders this will be one of:

01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12

<CurrencyA3>

Must be the letters GBP.

<ContactName>

Must contain at least one alpha or numeric character but no more than thirty-five characters.

<Online>

Must be a one-digit number set to 0.

<SubmissionLines>
Container node of one or more <SubmissionLine> nodes.

<SubmissionLine>

Repeating line can appear one or more times. Must contain the following nodes:

<CountryA2>, <CustomerVRN>, <Value>, <Indicator>

<CountryA2>

Must contain two alpha characters representing the EU Member States. AT, BE, BG, CY, CZ, DE, DK, EE, EL, ES, FI, FR, HU, IE, IT, LT, LU, LV, MT, NL, PL, PT, RO, SE, SI, or SK.

<CustomerVRN>
Must be valid for the Country entered in <CountryA2> as identified in the EC country codes and customer number formats page.

VAT Numbers can be entered on the European Commission website to check that they are valid for the country code entered.

<Value>

Containing the value of goods in pounds sterling. Must contain numerics. No decimal places, commas or brackets allowed. Minus figures must be prefixed with a '-' minus sign.

<Indicator>

Must be one digit numeric containing either 0, 2 or 3.

Enter code '0' for B2B Goods

Enter code '2' if the supplier is an intermediary in a triangular transaction.

Enter code ‘3’ for B2B Services

An example of an XML data file:

```
<?xml version='1.0' encoding='UTF-8'?>
<Submission type='HMRC_VAT_ESL_BULK_SUBMISSION_FILE'>
<TraderVRN>123456789</TraderVRN>
<Branch>123</Branch>
<Year>2010</Year>
<Period>03</Period>
<CurrencyA3>GBP</CurrencyA3>
<ContactName>Jim Bloggs</ContactName>
<Online>0</Online>
<SubmissionLines>
<SubmissionLine>
<CountryA2>AT</CountryA2>
<CustomerVRN>U12345678</CustomerVRN>
<Value>2000</Value>
<Indicator>2</Indicator>
</SubmissionLine>
<SubmissionLine>
<CountryA2>DE</CountryA2>
<CustomerVRN>123456789</CustomerVRN>
<Value>10</Value>
<Indicator>0</Indicator>
</SubmissionLine>
<SubmissionLine>
<CountryA2>ES</CountryA2>
<CustomerVRN>A12345678</CustomerVRN>
<Value>75</Value>
<Indicator>0</Indicator>
</SubmissionLine>
<SubmissionLine>
<CountryA2>FR</CountryA2>
<CustomerVRN>12345678901</CustomerVRN>
<Value>50</Value>
<Indicator>3</Indicator>
</SubmissionLine>
</SubmissionLines>
</Submission>
```

Validation checks are carried out during the upload of the file and any errors identified will be displayed by ‘X’ on the 'Review and Submit' screen. You will have the opportunity to amend or delete the data before resubmitting. All errors must be corrected and the file saved before uploading again. Only files free of errors will be accepted.

An acknowledgement message, with a unique 'Submission Reference Number', will be displayed which you should print and keep for your records.

ariadne software submits an EC Sales List bulk upload file to UK HMRC in the form of an XML file each quarter.

The application used to create this file is very simple. It consists of the following components:

1. A Query Management Query called **VAT_ECSL**. This is basically a piece of SQL which selects records from the ariadne invoice header file (sw_invhdr) where the invoice is not cancelled (status <> 'C'), the invoice date (invcdate) is in a specified range of dates and where the customer is in the EU, but not in the UK, and is registered for VAT (has a valid VAT number vatref).

   The source looks like this:

```
H QM4 05 Q 01 E V W E R 01 03 11/04/10 14:27
V 1001 050
select
substr(vatref,1,2) as COUNTRY,
trim(substr(vatref,3)) as VAT,
decimal(gbptotal,9,0) as AMOUNT
from sw_invhdr
where invcdate between &FROMDATE and &TODATE
and status <> 'C'
and vatref <> ' '
and vatref not in ('N/A','NONE','NOT REGISTERED')
and substr(vatref,1,2) <> 'GB'
order by invoice
```

2. A Database-to-XML map, also called **VAT_ECSL**. This specifies the QM query VAT_ECSL as its input source and defines how to build an XML file in the required format from the data returned by that query.

   The commands used to build this Database-to-XML map are encapsulated by the following CL code.

```
             PGM

/* Create Database-to-XML map. The input source is a QM Query called */
/* VAT_ECSL which selects invoices from the invoice file that match  */
/* suitable criteria                                                 */
             CRTDBFXML  MAPNAME(VAT_ECSL) INPUT(*QMQRY) +
                          QMQRY(VAT_ECSL *NONE *SYS) +
                          DFTUSEAUT(*DENIED) DFTCHGAUT(*DENIED) +
                          TEXT('VAT EC Sales list')

/* Define the Submission element.                                    */
/* This is the root element (PARENT(*NONE)) and there is only ever   */
/* one element of this type (NEWELMOPT(*NEVER))                      */
             ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(Submission) +
                          PARENT(*NONE) SEQNBR(*NONE) SOURCE(*NONE) +
                          TEXT(*BLANK) NEWELMOPT(*NEVER) +
                          XMLNAMESPC(*NONE *NONE)

/* Define the Submission element's attribute                         */
/* This is a constant "HMRC_VAT_ESL_BULK_SUBMISSION_FILE"            */
             ADDDBFXMLA MAPNAME(VAT_ECSL) ELEMENT(Submission) +
                          ATTRIBUTE(type) SEQNBR(100) +
                          SOURCE(*CONSTANT) +
                          CONSTANT('HMRC_VAT_ESL_BULK_SUBMISSION_FILE+
```

```
                          ') TEXT(*BLANK)

   /* Define the TraderVRN element as a sub-element of Submission       */
   /* There is only ever one element of this type (NEWELMOPT(*NEVER))   */
   /* This is a constant with value 123456789                           */
               ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(TraderVRN) +
                            PARENT(Submission) SEQNBR(100) +
                            SOURCE(*CONSTANT) CONSTANT('123456789') +
                            TEXT(*BLANK) NEWELMOPT(*NEVER) +
                            XMLNAMESPC(*NONE *NONE)

   /* Define the Branch element as a sub-element of Submission          */
   /* There is only ever one element of this type (NEWELMOPT(*NEVER))   */
   /* This is a constant with value 000                                 */
               ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(Branch) +
                            PARENT(Submission) SEQNBR(200) +
                            SOURCE(*CONSTANT) CONSTANT('000') +
                            TEXT(*BLANK) NEWELMOPT(*NEVER) +
                            XMLNAMESPC(*NONE *NONE)

   /* Define the Year element as a sub-element of Submission            */
   /* There is only ever one element of this type (NEWELMOPT(*NEVER))   */
   /* This is a constant the value of which is taken from the variable  */
   /* <:VAT_YEAR:> defined on the OPTIONS parameter of the CVTDBFXML    */
   /* command (see ILE RPG VAT_ECSLR)                                   */
               ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(Year) +
                            PARENT(Submission) SEQNBR(300) +
                            SOURCE(*CONSTANT) +
                            CONSTANT('<:VAT_YEAR:>') TEXT(*BLANK) +
                            NEWELMOPT(*NEVER) XMLNAMESPC(*NONE *NONE)

   /* Define the Period element as a sub-element of Submission          */
   /* There is only ever one element of this type (NEWELMOPT(*NEVER))   */
   /* This is a constant the value of which is taken from the variable  */
   /* <:VAT_PERIOD:> defined on the OPTIONS parameter of the CVTDBFXML  */
   /* command (see ILE RPG VAT_ECSLR)                                   */
               ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(Period) +
                            PARENT(Submission) SEQNBR(400) +
                            SOURCE(*CONSTANT) +
                            CONSTANT('<:VAT_PERIOD:>') TEXT(*BLANK) +
                            NEWELMOPT(*NEVER) XMLNAMESPC(*NONE *NONE)

   /* Define the Currency element as a sub-element of Submission        */
   /* There is only ever one element of this type (NEWELMOPT(*NEVER))   */
   /* This is a constant with value GBP                                 */
               ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(CurrencyA3) +
                            PARENT(Submission) SEQNBR(500) +
                            SOURCE(*CONSTANT) CONSTANT('GBP') +
                            TEXT(*BLANK) NEWELMOPT(*NEVER) +
                            XMLNAMESPC(*NONE *NONE)

   /* Define the ContactName element as a sub-element of Submission     */
   /* There is only ever one element of this type (NEWELMOPT(*NEVER))   */
   /* This is a constant with value John Smith                          */
               ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(ContactName) +
                            PARENT(Submission) SEQNBR(600) +
                            SOURCE(*CONSTANT) CONSTANT('John +
                            Smith') TEXT(*BLANK) NEWELMOPT(*NEVER) +
                            XMLNAMESPC(*NONE *NONE)

   /* Define the Online element as a sub-element of Submission          */
   /* There is only ever one element of this type (NEWELMOPT(*NEVER))   */
   /* This is a constant with value 0                                   */
```

```
            ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(Online) +
                         PARENT(Submission) SEQNBR(700) +
                         SOURCE(*CONSTANT) CONSTANT('0') +
                         TEXT(*BLANK) NEWELMOPT(*NEVER) +
                         XMLNAMESPC(*NONE *NONE)

/* Define the SubmissionLines element as a sub-element of Submission */
/* There is only ever one element of this type (NEWELMOPT(*NEVER))   */
/* This is just a container (SOURCE(*NONE))                          */
            ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(SubmissionLines) +
                         PARENT(Submission) SEQNBR(800) +
                         SOURCE(*NONE) TEXT(*BLANK) +
                         NEWELMOPT(*NEVER) XMLNAMESPC(*NONE *NONE)

/* Define the SubmissionLine element as a sub-element of             */
/* SubmissionLines                                                   */
/* There is a new element of this type for every record read from    */
/* the input source (NEWELMOPT(*RCD))                                */
/* This is just a container (SOURCE(*NONE))                          */
            ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(SubmissionLine) +
                         PARENT(SubmissionLines) SEQNBR(100) +
                         SOURCE(*NONE) TEXT(*BLANK) +
                         NEWELMOPT(*RCD) XMLNAMESPC(*NONE *NONE)

/* Define the CountryA2 element as a sub-element of SubmissionLine.  */
/* There is a new element of this type for every record read from    */
/* the input source (NEWELMOPT(*RCD))                                */
/* The value is derived from column COUNTRY in the input source      */
            ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(CountryA2) +
                         PARENT(SubmissionLine) SEQNBR(100) +
                         SOURCE(*COLUMN) COLUMN(*ONLY/COUNTRY) +
                         TEXT(*BLANK) NEWELMOPT(*RCD) +
                         XMLNAMESPC(*NONE *NONE)

/* Define the CustomerVRN element as a sub-element of SubmissionLine */
/* There is a new element of this type for every record read from    */
/* the input source (NEWELMOPT(*RCD))                                */
/* The value is derived from column VAT in the input source          */
            ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(CustomerVRN) +
                         PARENT(SubmissionLine) SEQNBR(200) +
                         SOURCE(*COLUMN) COLUMN(*ONLY/VAT) +
                         TEXT(*BLANK) NEWELMOPT(*RCD) +
                         XMLNAMESPC(*NONE *NONE)

/* Define the Value element as a sub-element of SubmissionLine.      */
/* There is a new element of this type for every record read from    */
/* the input source (NEWELMOPT(*RCD))                                */
/* The value is derived from column VAT in the input source          */
            ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(Value) +
                         PARENT(SubmissionLine) SEQNBR(300) +
                         SOURCE(*COLUMN) COLUMN(*ONLY/AMOUNT) +
                         TEXT(*BLANK) NEWELMOPT(*RCD) +
                         XMLNAMESPC(*NONE *NONE)

/* Define the Indicator element as a sub-element of SubmissionLine.  */
/* There is a new element of this type for every record read from    */
/* the input source (NEWELMOPT(*RCD))                                */
/* The value is a constant 0                                         */
            ADDDBFXMLE MAPNAME(VAT_ECSL) ELEMENT(Indicator) +
                         PARENT(SubmissionLine) SEQNBR(400) +
                         SOURCE(*CONSTANT) CONSTANT('0') +
                         TEXT(*BLANK) NEWELMOPT(*RCD) +
                         XMLNAMESPC(*NONE *NONE)
```

```
ENDPGM
```

3. An ILRE RPG program called VAT_ECSLR which is passed a year and quarter number as parameters. It calculates the from- and to- dates for the given year and period and then runs CVTDBFXML to build an XML document from the QM query output.

   The RPG source looks like this:

```
H ALWNULL(*USRCTL)
H EXTBININT
H DFTACTGRP(*NO)

 * Array of month numbers that start each quarter
D quarterStart    S              2S 0 DIM(4) CTDATA PERRCD(4)

 * Prototype for this program
D VAT_ECSLR       PR                  EXTPGM('VAT_ECSLR')
D  year                           4S 0 CONST
D  quarter                        1S 0 CONST

 * from- and to-dates calculated from the year and quarter passed
D fromDate        S               D
D toDate          S               D

 * ending month number of the range
D toMonth         S              2S 0

 * command string
D cmd             S           4096A   VARYING

 * Prototype for QCMDEXC
D Command         PR                  EXTPGM('QCMDEXC')
D   iaCommand                32000A   CONST OPTIONS(*VARSIZE)
D   inCmdLen                    15P 5 CONST

 * Structure used for calculating from- and to-dates
D Date            DS                  QUALIFIED
D   date                         D
D   year                         4S 0 OVERLAY(date:1)
D   month                        2S 0 OVERLAY(date:6)
D   day                          2S 0 OVERLAY(date:9)

 * Program interface
D VAT_ECSLR       PI
D  year                           4S 0 CONST
D  quarter                        1S 0 CONST

 /free

  // Determine the date range for the specified quarter
  clear Date.date;
  Date.year = year;
  Date.month = quarterStart(quarter);
  Date.day = 01;
  fromDate = Date.date;
  toDate = fromDate + %months(3);
  toDate = toDate - %days(1);
  toMonth = %subdt(toDate:*M);

  // Build the command string to run
  // For example, if year = 2011 and quarter = 4:
  // fromDate = 2011-10-01
```

```
        // toDate   = 2011-12-31
        // toMonth  = 12
        // cmd      = CVTDBFXML
        //              FROMFILE(*MAP) TOSTMF('/VAT/ecsl_20114.xml')
        //              STMFOPT(*REPLACE) MAPNAME(VAT_ECSL)
        //              QMQRY(*N *NONE *NO *QMQRY ((FROMDATE '2011-10-01')
        //                                         (TODATE   '2011-12-31')))
        //              XMLSTYLING(*NONE) OPTIONS(('<:VAT_YEAR:>' '2011')
        //                                        ('<:VAT_PERIOD:>' '12'))
        cmd = 'CVTDBFXML '
              + 'FROMFILE(*MAP) TOSTMF('''
              + '/VAT/ecsl_' + %char(year) + %char(quarter) +'.xml'
              + ''') STMFOPT(*REPLACE) MAPNAME(VAT_ECSL) '
              + 'QMQRY(*N *NONE *NO *QMQRY ((FROMDATE '''
              + %char(fromDate:*ISO)
              + ''') (TODATE '''
              + %char(toDate:*ISO)
              + '''))) XMLSTYLING(*NONE) OPTIONS((''<:VAT_YEAR:>'' '
              + %editc(year:'X')
              + ') (''<:VAT_PERIOD:>'' '
              + %editc(toMonth:'X')
              + '))';

        // Run the command to create the XML file
        Command(cmd:%len(cmd));

        *INLR = *ON;
        return;

       /end-free

** Period start month
01040710
```

The Integrated File System

The system i Integrated File System (IFS) provides a coherent, coordinated set of file systems which can used for storing a variety of data physically on the system i or for communicating with file systems on other platforms.

These file systems include the following that may be of use to you for storing stream files created by ***CoolSpools Database*** locally on your system i or remotely on another computer (PC, UNIX server etc.)

"root"	The **"root" (/)** file system. This file system takes full advantage of the stream file support and hierarchical directory structure of the integrated file system. The root file system has the characteristics of the Disk Operating System (DOS) and OS/2 file systems. You should typically use the root file system if you want to store stream files created by ***CoolSpools Database*** locally on the same system i where you run the CVTDBFXL command.
QNTC	Windows NT Server file system. This file system provides access to data and objects that are stored on a PC running Windows NT 4.0 or higher. It allows system i server applications to use the same data as Windows NT clients. If you prefer not to store stream file locally on your system i, you can use QNTC to enable ***CoolSpools Database*** to write stream files directly to a PC running NT, Windows 2000, Windows XP etc. instead. This can be an effective way of sharing data created by ***CoolSpools Database*** amongst your users and customers.
NFS	Network File System. This file system provides you with access to data and objects that are stored on a remote NFS server. An NFS server can export a network file system that NFS clients will then mount dynamically. This may be an option for sending stream files to a UNIX server.
QNetWare	The QNetWare file system. This file system provides access to local or remote data and objects that are stored on a server that runs Novell NetWare 4.10 or 4.11 or to standalone PC Servers running Novell Netware 4.12, 4.10 4.11 or 5.0. You can dynamically mount NetWare file systems over existing local file systems.
QOpenSys	The open systems file system. This file system is compatible with UNIX-based open system standards,

such as POSIX and XPG. Like the root file system, this file system takes advantage of the stream file and directory support that is provided by the integrated file system. In addition, it supports case-sensitive object names.

QDLS — The document library services file system (previously known as “shared folders”). This file system provides access to documents and folders. Use this file system only if you have applications which require it. QDLS is significantly slower and has major limitations (e.g. in relation to naming) compared with the root file system.

QFileSvr.400 — This file system provides access to other file systems that reside on remote system i servers. You can use QFileSvr.400 to save output from ***CoolSpools Database*** directly to another system i.

See ***http://publib.boulder.ibm.com/system i/V5R2/ic2924/info/rzaia/rzaiacon.htm*** for full details of the Integrated File System.

We will now focus a little more closely on the QNTC File System since this is little known area of system i functionality which may well be of serious interest to users of ***CoolSpools Database***.

QNTC

The QNTC file system is a subdivision of the IFS (Integrated File System) that enables the system i to access file and device shares (e.g. printers and CDROM) drives on a remote NT system. Please note that contrary to a commonly held fallacy this is ***NOT*** restricted to the Integrated XSeries Server (aka IPCS or FSIOP).

Using the QNTC file system, your system i can read and write files that reside physically on a PC running Windows NT 4 or above. This means that ***CoolSpools Database*** can output stream files directly to an NT server if you would prefer to store them there rather than in the root file system of your IFS.

To use QNTC, the only software you need other than the base operating system is TCP/IP Connectivity Utilities for system i 400 *(S722-TCI).* However, setting up QNTC can be tricky. For full setup information, refer to the article in the IBM Software Knowledgebase at ***http://www-912.ibm.com/s_dir/slkbase.nsf/1ac66549a21402188625680b0002037e/aea450153eebf8ff8625670f0072550f?OpenDocument&Highlight=0,QNTC***

However, here is a quick overview of the steps you need to follow to set up QNTC.

Domain Names

First you must ensure that the domain name defined by your system i NetServer configuration matches your PC's Windows network workgroup name. NetServer is the function on the system i that provides support for the Windows Network Neighborhood. You can use Operations Navigator (OpsNav) to set up and manage NetServer.

Please note that changing the system i domain name in NetServer may affect which PCs will be able to see the system i in their Network Neighborhood.

From the main OpsNav window, click the name of your machine, select File Systems, and then right-click File Shares. Choose Open system i NetServer from the menu to display the NetServer window. Right-click system i NetServer and choose Properties to display the Properties window, where you can change the domain. Click the General tab and press the Next Start button. A window appears where you can set properties that will be used the next time NetServer is restarted. In the domain name field, enter a name that matches the workgroup of the NT PC that you wish to access.

Now, if you are sure that NetServer is not currently in use, end and restart NetServer by clicking the Stop icon followed (once NetServer has fully ended) by the green triangle icon.

If you wish to change the Network Id on the PC side, this can be done through the Windows Control Panel. On Windows 2000, select Start, Settings, Control Panel, and System. Then choose the Network Identification tab, click the Properties button, select Workgroup, and set the workgroup to match the domain that NetServer on the system i was configured to use.

On XP, you'll find the option on the Computer Name tab.

On Windows NT, the option to change the workgroup name can be found if you select Start, Settings, Control Panel, and Network.

Note that you may have to reboot the PC for these changes to take effect.

User Id and Password

One other thing which must match between the system i and the NT PC is the user id and password you're going to use. It is vital that your system i user id be recognised by the NT PC as a valid network logon id, and that the passwords are the same on both the system i and the NT machine. It may well be advisable to create a special user id on the two platforms specifically for the purpose of communicating between them using QNTC. You can then ensure that when the password needs to be changed, it is changed on both systems, if this needs to be done manually.

Testing the Connection

To test your connection, first ensure that you are logged on to the system i using a shared user id/password (see above), then run the command WRKLNK '/QNTC/*' on the system i. This could take several minutes to complete the first time it is run. Your NT

system should appear on the list. If it doesn't, you may be able to manually establish a connection to the NT system using command CRTDIR /Q*NTC*/ *<servername>*', substituting the name of your NT system for *servername*.

Find your NT system in this list and choose Display (option 5) in WRKLNK (Work with Object Links) to display your file shares on NT. If the file shares don't appear, double-check that your user IDs and passwords match exactly on both systems.

You access the QNTC file system by including QNTC and the name of your PC and share name in the path name you specify on TOSTMF (To Stream File) parameter of the CVTDBFXL command.

For example, let's imagine you have a company server running Windows NT 4 or above and you have decided that this is a convenient place to save the output from ***CoolSpools Database*** so that all of your users can have shared access to the data. Let's imagine that this server is called **NTServer** and that it has a file share name set up called **NTFiles**. Below the share name, there is a directory called **CustData**. If you wanted to convert data from your customer file and save it as an Excel spreadsheet in this location under the file name **customer_file.xls**, you would run a command something like this:

```
CVTDBFXL      FROMFILE(custfile)
              TOSTMF('/QNTC/NTServer/NTFiles/CustData/customer_file.xls')
              TOFMT(*XLS)
```

Note that the path name starts with **/QNTC**. This indicates to the system i that you are referring to the QNTC File System. Following /QNTC is the name of the PC to which the data is to be sent: **NTServer**. Next comes the name of the file share: **NTFiles**. After that is the name of the directory below the share: **CustData**. Finally we have the name of the file itself: **customer_file.xls**.