**Using Boyum External Launcher**

**Mascidon, LLC**

**May 2023**

Table of Contents

[Using External Launcher in Boyum 3](#_Toc134118737)

[Figure 1.1 External Launcher Common Programs List 3](#_Toc134118738)

[EDI Customer Use of This Function 3](#_Toc134118739)

[Figure 1.2 SQL Report Variables – Input 4](#_Toc134118740)

[Figure 1.3 View an Incoming XML Sales Order 4](#_Toc134118741)

[Figure 1.4 SQL Report – Used to ‘Select’ a File to Review 5](#_Toc134118742)

[Figure 1.5 XML Sales Order Display in Notepad 5](#_Toc134118743)

[Review of Boyum Functions Used 6](#_Toc134118744)

[Figure 2.1 Add / Edit Menu Used to Reprocess a File 6](#_Toc134118745)

[Figure 2.2 SQL Report Used for Selections 6](#_Toc134118746)

[Figure 2.3 SQL Variables for EDI-40 Universal Function 7](#_Toc134118747)

[Figure 2.4 SQL Report Format Options 7](#_Toc134118748)

[Figure 2.5 SQL Report Within EDI-40 8](#_Toc134118749)

[Figure 2.6 ‘Process’ Button Action 9](#_Toc134118750)

[Figure 2.7 Line Loop Universal Function to Process Selected File(s) 9](#_Toc134118751)

[Figure 2.8 External Launcher Used to View the File 10](#_Toc134118752)

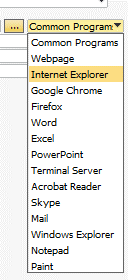
[Figure 2.9 Macro to Store the Field Name and Directory Where the File Resides 11](#_Toc134118753)

[Figure 2.10 SQL UF To Initiate the Reprocessing 11](#_Toc134118754)

[Figure 2.11 Close the Original SQL Report 12](#_Toc134118755)

# Using External Launcher in Boyum

The external launcher function within Boyum allows you to initiate the use of programs outside of SAP. Figure 1.1 shows ‘Common programs’ used for this purpose.



## Figure 1.1 External Launcher Common Programs List

In this demonstration of Boyum, the following features will be detailed:

* Using the Universal Function ‘External Launcher’
* Using a SQL report to provide the means to ‘select’ documents to view using Notepad via the External Launcher
* Within the SQL report, use SQL statements to define the SQL variable selections
* Within the SQL report, use the Edit and Hide options to allow entry when needed and to not display background fields required for processing (that may confuse the user)
* Use a Line Loop Universal Function to process the calls to the External Launcher

## EDI Customer Use of This Function

I support EDI import of sales orders for a number of customers and needed to address a problem with the EDI documents being sent. The basic problem is this:

* Mascidon receives an EDI sales order document in XML format from GM via a third party EDI translator program. i.e. GM sends an ASNII X12 document and this picked up by the third part EDI vendor. They do all of the hand-shaking with GM and adhere to GM rules. They periodically (hourly) send XML sales order documents to my customer’s FTP site.
* Mascidon checks every hour for the existence of new files in the FTP site.
* If new files are found – they get processed as SAP sales orders via an SDK developed by Mascidon.
* If there is an error when the SDK process runs, the user gets an error message. Normal messages deal with part setup problems particularly for new parts being introduced. It doesn’t matter the reason – we need to fix the problem and then reprocess the original XML file so these orders are logged in SAP.

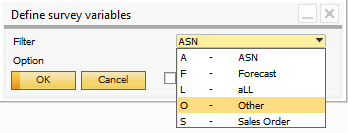
At this point the customer has:

* Corrected the problem
* Has a backup copy of the original XML file
* Wants to reprocess this file and create the sales order in SAP
* Has several hundred XML orders in backup directories

The challenge using Boyum is to do the following:

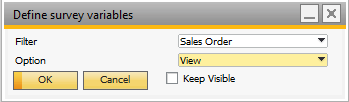
* Present the list of XML orders potentially available for reprocessing
* Let them view the XML file(s) so they can verify data
* Initiate the ‘reprocessing’ of the XML orders

A Boyum SQL Universal Function is called from Boyum’s Add/Edit menu. The variables required for the SQL report are input as requested variables – as shown in Figure 1.2.



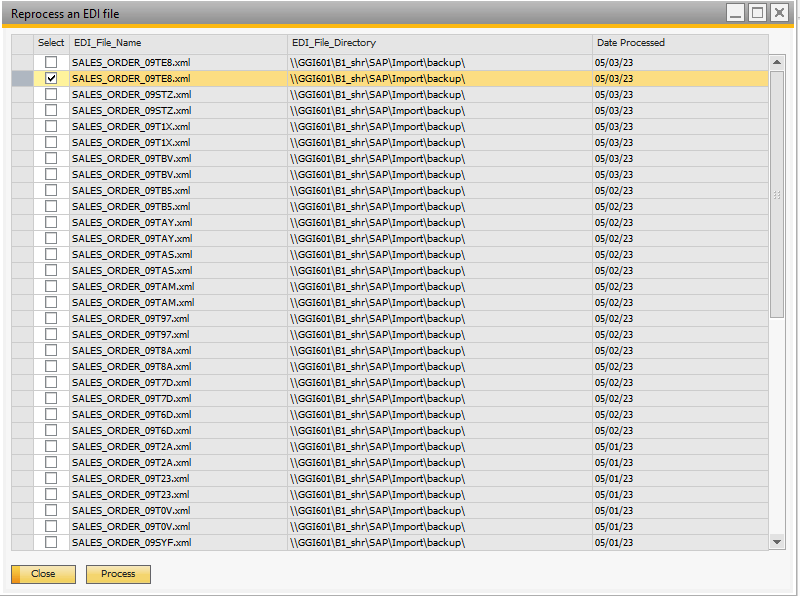
## Figure 1.2 SQL Report Variables – Input

To view an XML incoming sales order the selections would be as shown in Figure 1.3.



## Figure 1.3 View an Incoming XML Sales Order

When the user clicks OK the actual SQL report is run. This is shown in Figure 1.4.



## Figure 1.4 SQL Report – Used to ‘Select’ a File to Review

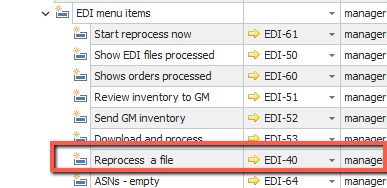
When the user clicks the ‘Process’ button, the External Launch Universal Function executes and displays the XML file data – as shown in Figure1.5.



## Figure 1.5 XML Sales Order Display in Notepad

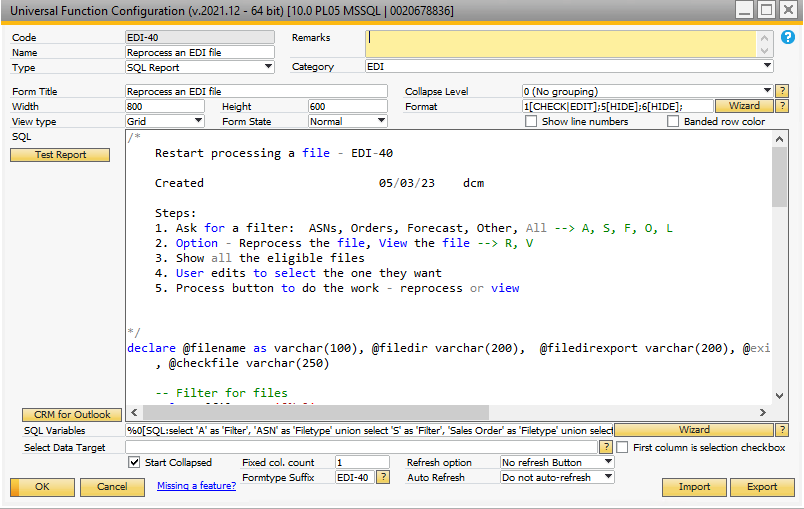
## Review of Boyum Functions Used

The first Boyum usage is the Add / Edit menus. This is shown in Figure 2.1.



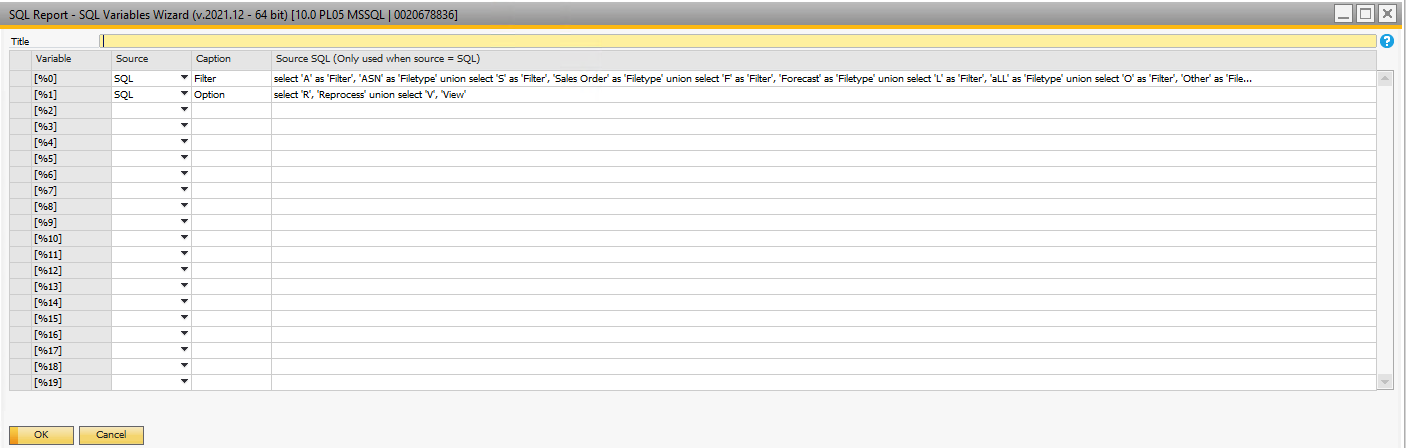
## Figure 2.1 Add / Edit Menu Used to Reprocess a File

The Universal Function SQL EDI-40 is shown in Figure 2.2.



## Figure 2.2 SQL Report Used for Selections

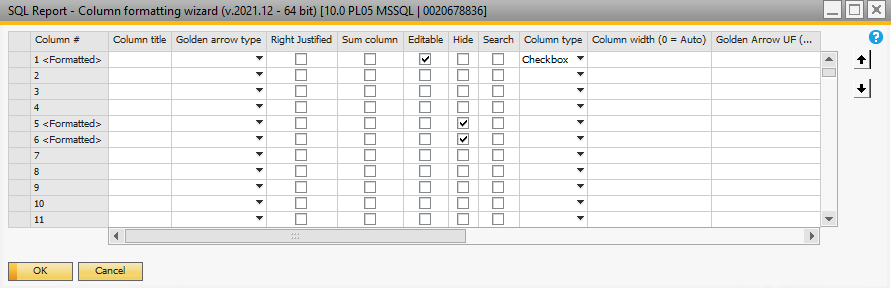
Note the use of a SQL select statement to prepare the SQL variables in a ‘drop down’ options presentation.



## Figure 2.3 SQL Variables for EDI-40 Universal Function

Note the format wizard used to format the 1st field in the SQL report as a ‘Check Box’ that is editable by the user. The report output as shown in Figure 1.4 shows the check box. The 5th and 6th columns of data in the SQL report are hidden from the user.

Figure 2.6 shows the contents of the SQL report.

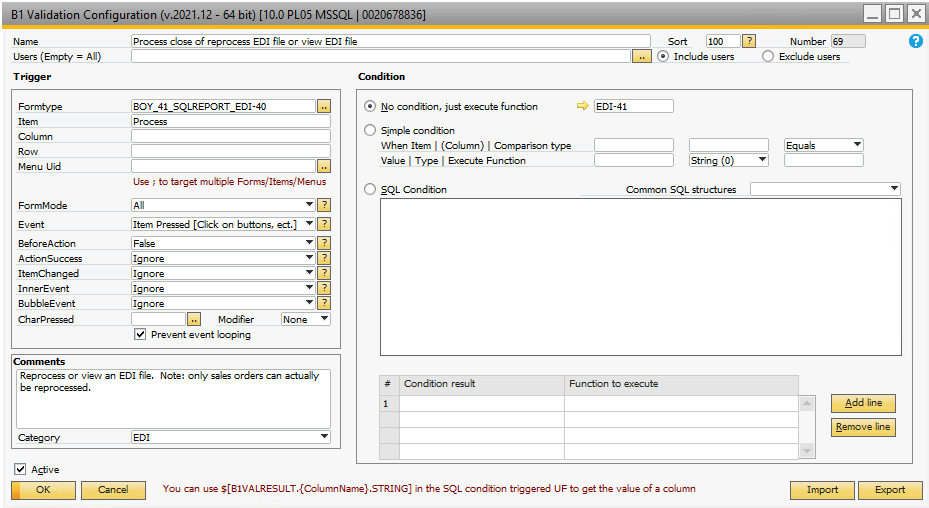


## Figure 2.4 SQL Report Format Options

*/\* Restart processing a file - EDI-40 Created 05/03/23 dcm Steps: 1. Ask for a filter: ASNs, Orders, Forecast, Other, All --> A, S, F, O, L 2. Option - Reprocess the file, View the file --> R, V 3. Show all the eligible files 4. User edits to select the one they want 5. Process button to do the work - reprocess or view\*/declare @filename as varchar(100), @filedir varchar(200), @filedirexport varchar(200), @exists int, @filter varchar(1), @option varchar(1), @processdate datetime , @checkfile varchar(250) -- Filter for files select @filter = '[%0]' -- Option select @option = '[%1]' create table #files(filenames varchar(200), filedir varchar(200), processdate datetime) select top 1 @filedirexport = u\_xmlexport from [@DCMEDIPARAMETERS] declare mainloop cursor for select u\_filename , u\_directory , u\_processdate from [@DCMEDIFILES] where (@filter = 'L' --aLL or (@filter = 'A' and left(u\_filename,3) = 'ASN') or (@filter = 'S' and left(u\_filename,5) = 'SALES') or (@filter = 'F' and left(u\_filename,4) = 'FORE') or (@filter = 'O' and (left(u\_filename,3) not in ('FOR','SAL' ,'ASN'))) ) and CHARINDEX('.xml',u\_filename) > 0 -- must be an XML file order by 3 -- processdate open mainloop fetch mainloop into @filename, @filedir, @processdate while @@FETCH\_STATUS = 0 begin if isnull(@filedir,'') = '' -- ASNs do not have directory stored begin select @checkfile = @filedirexport + 'backup\' + @filename Exec master.dbo.xp\_fileexist @checkfile, @exists output if @exists = 1 select @filedir = @filedirexport else -- not found - check if in Corrugated area begin select @checkfile = @filedirexport + 'Corrugated\backup\' + @filename Exec master.dbo.xp\_fileexist @checkfile, @exists output if @exists = 1 select @filedir = @filedirexport + 'Corrugated\' -- cannot find it - omit from list end end select @filedir = @filedir +'backup\' insert into #files select @filename, @filedir, @processdate fetch mainloop into @filename, @filedir, @processdate end close mainloop deallocate mainloop select 'N' as 'Select' , filenames as 'EDI\_File\_Name' , filedir as 'EDI\_File\_Directory' , processdate as 'Date Processed' , @option as 'ProcessOption' ,* ***case when @filter = 'S' then @option when @filter = 'L' and left(filenames,4) = 'SALE' then @option else 'V' end as 'HowtoProcess'*** *from #files order by 4 desc drop table #files*

## Figure 2.5 SQL Report Within EDI-40

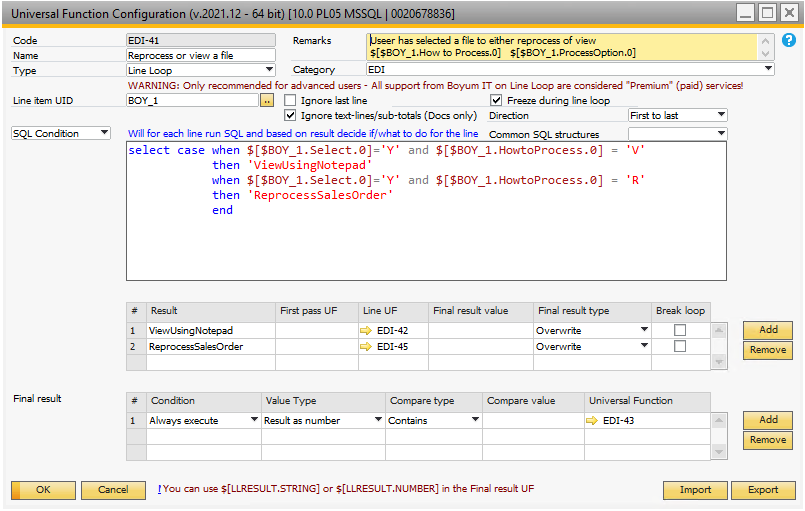
The ‘Process’ button on the SQL report was added using the Boyum Item Placement Tool. When the user clicks the ‘Process’ button a B1 Validation as shown in Figure 2.6.



## Figure 2.6 ‘Process’ Button Action

The Universal Function EDI-41 is called when the Process button is clicked – shown in Figure 2.7. Note these options in the line loop function:

* Since you are processing records from a SQL ‘select’ report, uncheck the ‘Ignore last line’ (otherwise the last file in the list cannot be reported)



## Figure 2.7 Line Loop Universal Function to Process Selected File(s)

* The SQL condition checks the 2 columns:
  + $[BOY\_1.Select.0] – this is the ‘Check Box’ in the report
  + $[BOY\_1\_HowtoProcess] – this is a hidden field in the SQL report – highlighted in ‘Red’ in the SQL shown above. It’s value is either ‘V’ = View the file, or ‘R’ = Reprocess the file

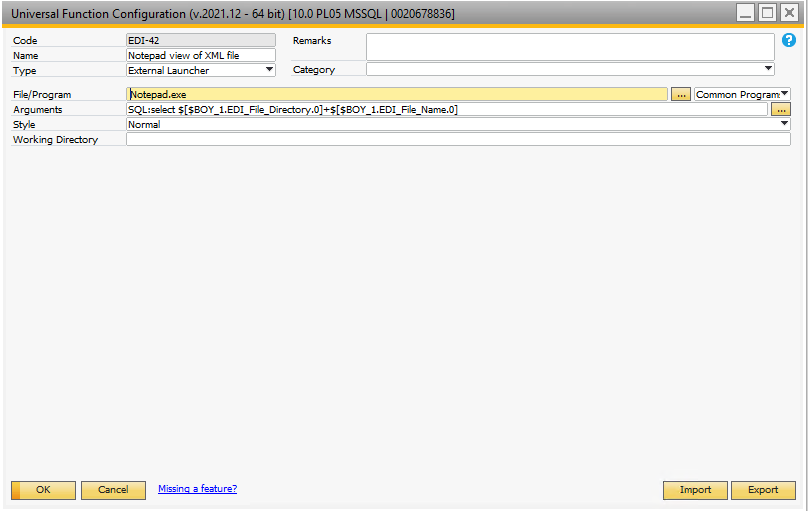
If the ‘How to Process’ field value is ‘View’ the file, then the UF EDI-42 is called by the line loop. If it is ‘Reprocess’ then the UF EDI-45 is called by the line loop.

**‘View Option’**

The view option EDI-42 is shown in Figure 2.8. Look closely at the ‘Arguments’ – a SQL command that concatenates the file directory with the file name to get the file to open using Notepad.

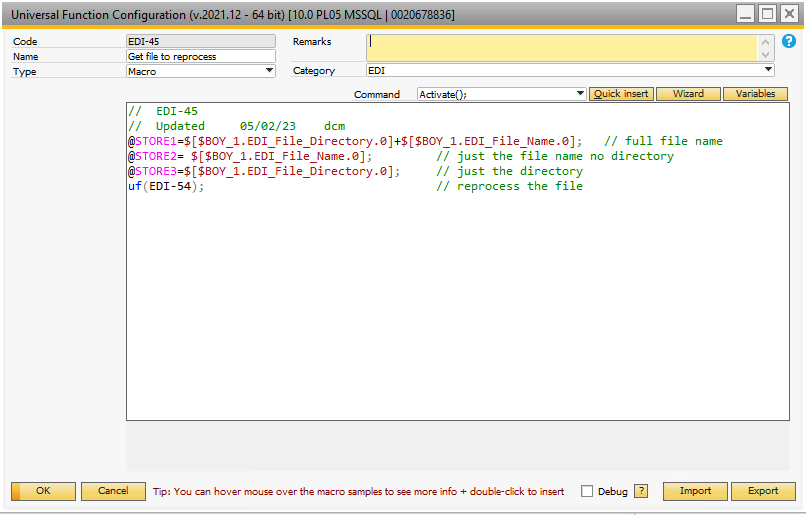
**NOTE:** An important note: if you are adding 2 fields together like $[$BOY\_1.EDI\_File\_Directory.0] and $[$BOY\_1.EDI\_File\_Name.0] must not contain spaces – Boyum rejects those fields. It’s a simple fix since these are fields in the SQL report with an ‘as’ designation controlling the field names.

That’s really all there is to defining the External Launcher and feeding it the arguments required.

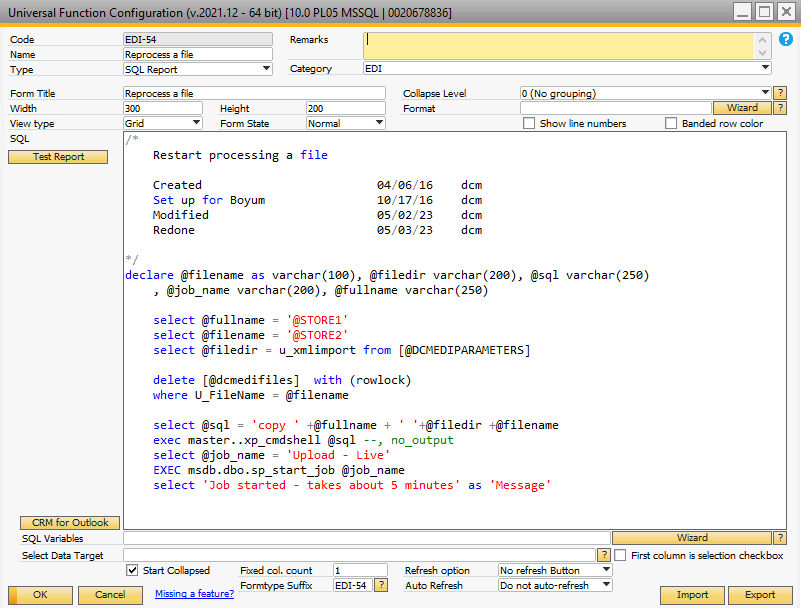


## Figure 2.8 External Launcher Used to View the File

The other option is to ‘Reprocess’ the XML sales order. In my system, the file to be reprocess needs to be copied to a specific directory on the network and then a SQL server agent is called to run an external SDK that loads the sales order into SAP. Figure 2.9 shows the first of 2 Universal Functions called to Reprocess the file.



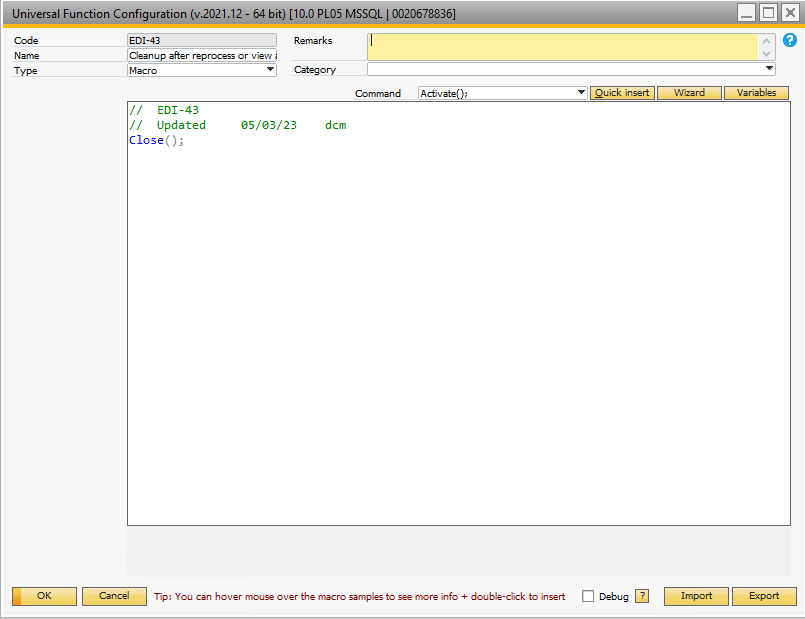
## Figure 2.9 Macro to Store the Field Name and Directory Where the File Resides



## Figure 2.10 SQL UF To Initiate the Reprocessing

Figure 2.10 also illustrates how to call a SQL Server Agent Job from within SQL.

Once all the line loop processes have completed, the last step is to call EDI-43 to close the SQL report – the work is done!



## Figure 2.11 Close the Original SQL Report