Parallel Processing – Example Code

**Most of the time when dealing with huge amount of data, the run time / execution time of a program should be with in the specified time limit. To overcome this, SAP has provided a wonderful tool called “Parallel processing”.**

**My attempt to provide an example for parallel processing.**

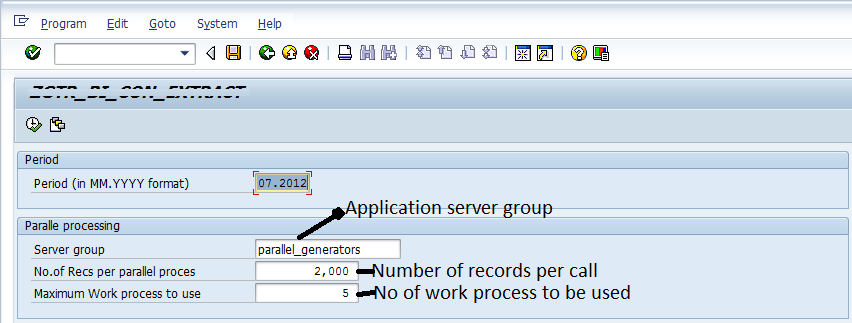
**Requirement:**  
Get the list of Contracts (New, Terminated and Active as on date) for a given period and retrieve the additional information using parallel processing.

###### Solution:

Requirement is seems to be simple. But, yes scanning CRMD\_ORDERADM\_H for all the contracts and retrieving the additional data (Partner, Product, Customer etc..) for each contract is not an easy and quick task.

Parallel processing has been introduced to complete the report with in the specified time.

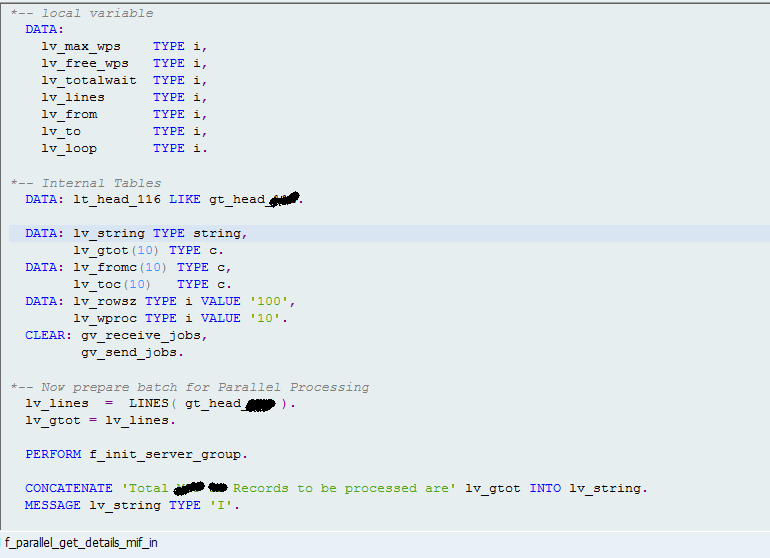
###### Selection Screen:

**[](https://blogs.sap.com/wp-content/uploads/2012/08/selection_screen_130478.png)**

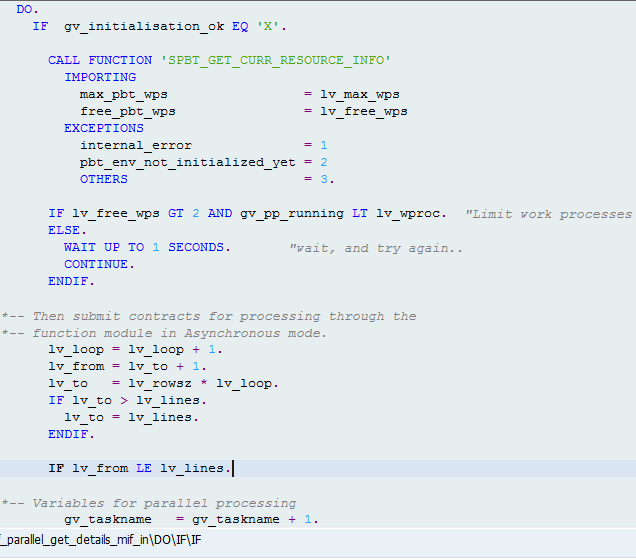
###### Parallel processing code:

1. Do the Initial selection of contracts based on the Period.
2. Call the FM ‘SPBT\_INITIALIZE’ to know the Maximum work process and free work process by passing the server group name (parallel\_generators)
3. [](https://blogs.sap.com/wp-content/uploads/2012/08/initialize_130480.png)
4. Note: For server group details, the number of maximum work process you can use under this group are best answered by your BASIS team. In practice, BASIS will provide a dedicated application sever group for parallel processing.
5. Based on the selection screen entry (number of records per call),  move these many records into an internal table for parallel processing.
6. If you want to track the number of records processed, use below statements to see the job log while the program is running.
7. CONCATENATE *‘Total Records to be processed are’*lv\_gtot *INTO*lv\_string*.*  
      *MESSAGE*lv\_string *TYPE ‘I’.*
8. Here lv\_gtot is the number of lines of the internal table which is passed to parallel processing FM
9. Retrieve the resources info by calling FM ‘**SPBT\_GET\_CURR\_RESOURCE\_INFO**‘ and check against the selection screen entry “Maximum no of work process to be used”.
10. If the available work process is greater than specified, and the running work process are within the max limit, call the RFC Function module in Asynchronous mode. If not wait for the free work process and try again.
11. When the RFC is called, please check the returned sy-subrc value. If it is initial, count the number of submitted RFC.
12. Continue the submission of Asynchronous RFC till all the records are processed.
13. The RFC is executed in the next available work process in the said server group and the results are captured under “Call back routine” ON END OF TASK.
14. In the receiving routine count the number of RFC received.
15. As this is asynchronous calls, we need to wait until the number of sent calls and received should be equal.

###### Code snippet:

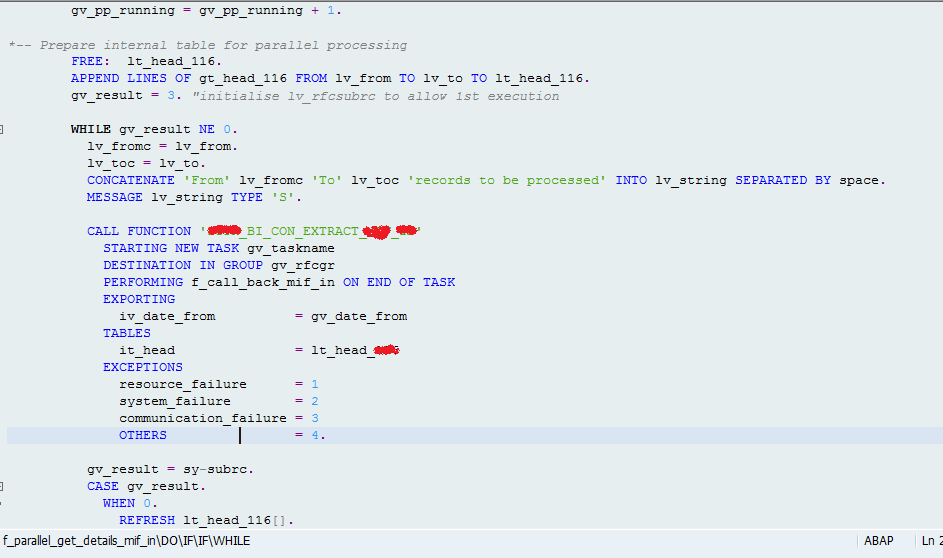
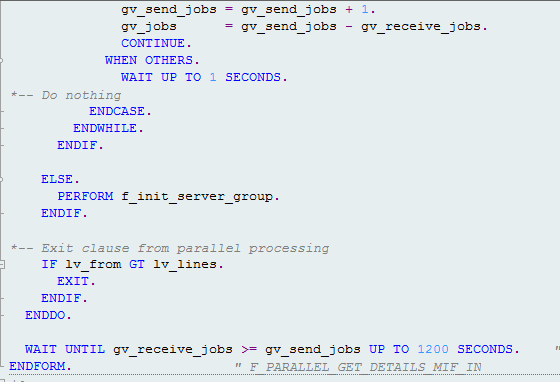
[](https://blogs.sap.com/wp-content/uploads/2012/08/01_130490.png)

Perform f\_init\_server\_group screen shot attached at the top of the page.

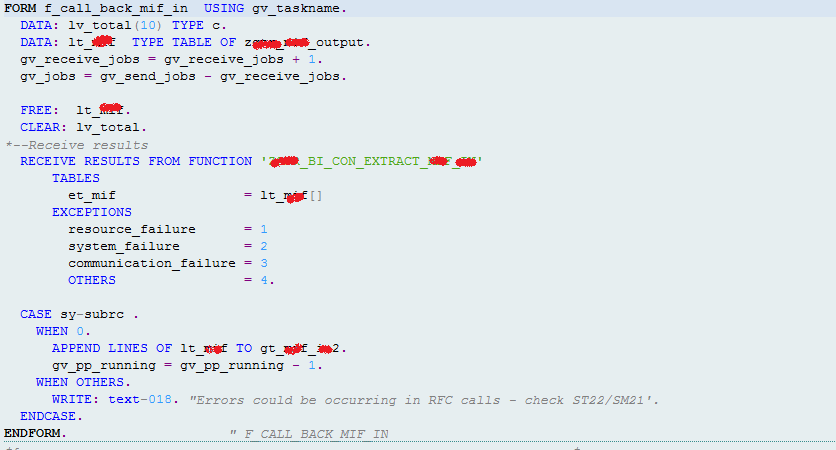
[](https://blogs.sap.com/wp-content/uploads/2012/08/02_130491.png)

     Here in below, we are calling the RFC Function module which does the actual work. It retries the all the requested information for the passed chunk of data.

     Once the FM is executed fully, then the call back routine “f\_call\_back\*” is called.

[](https://blogs.sap.com/wp-content/uploads/2012/08/03_130492.png)[](https://blogs.sap.com/wp-content/uploads/2012/08/05_130493.png)

     In the below call back routine, we will capture the processed records with all the information.

[](https://blogs.sap.com/wp-content/uploads/2012/08/07_130495.png)

     The final internal table gt\_\*\_in2 will holds the all the information.

      If any data base update is required then it should be done in the RFC itself.

      For tracking of the records processed, use “MESSAGE” statement where ever required.

      And last but not least, don’t forget to wait till the sent RFC calls should be equal to received RFC Calls.