DSO Job Log and Activation Parameters

Applies to:
SAP BW 7.x

Summary
This article explains the job log during activation of requests in standard DSO. It explains Transaction RSODSO_SETTINGS and how the settings influence activation performance.

Author: Jürgen Noe
Company: Kheto Consulting GmbH
Created on: 8 January 2011

Author Bio
Jürgen Noe has over 9 years consulting experience in SAP BW projects and ABAP/OO programming. Jürgen is presently working for Kheto Consulting GmbH, Germany. He's working mainly on SAP BI 7.0 and SAP BO front end tools. He is specialised in SAP BW backend, performance and administration.
Table of Contents
1. DataStore Objects in BW 7.x ................................................................. 3
2. Manually activation of a request .......................................................... 3
3. BI_ODSA* and BIBCT* Jobs ............................................................... 4
4. Transaction for DSO settings ............................................................... 5
5. Settings for activation .......................................................................... 6
6. Settings for SID generation ................................................................. 7
7. Settings for Rollback ........................................................................... 8
8. DataStore-specific settings ................................................................. 8
9. Activation in Process chains ............................................................... 8
Related Content ..................................................................................... 9
Copyright ............................................................................................... 10
1. DataStore Objects in BW 7.x

In BI 7.x you have three different kinds of DataStoreObjects (DSO): Standard, write-optimized and direct update. A Standard DSO consists of a new data and an active date table and a changelog table, which records the changes. Write-optimized DSO and DSO for direct update consist of an active table only.

In BI 7.x the background process how data in standard DataStoreObjects is activated has changed in comparison to BW 3.5 or prior. In this article I will explain the DSO activation job log and the settings / parameters of transaction “RSODSO_Settings”. I will describe how the parameters you can set in this transaction influence DSO activation performance. I will not describe the different activation types.

2. Manually activation of a request

If you loaded a new request with a Data Transfer Process (DTP) in your standard DSO, data is written to new data table. You can manually activate the request or within a process chain.

If you manually activate requests, you get following popup screen:

![Picture 1: Manual DSO Activation]

Button "Activate in Parallel" sets the settings for parallel activating. In this popup you select either dialog or background. In background you select job class and server. For both you define the number of jobs for parallel processing. By default it is set to '3'. This means, you have two jobs that can be scheduled parallel to activate your data, the BIBCTL* jobs. The third job is needed for controlling the activation process and scheduling the processes. That's the BI_ODSA* job.
3. BI_ODSA* and BIBCT* Jobs

The main job for activating your data is "BI_ODSAXXXXXXXXXXXXXXXXXXXXXXXXXX" with a unique 25-letter-GUID at the end. Let's have a look at the job log with SM37.

Activating data is done in 3 steps. First it checks status of request in DSO if it can be activated, marked green in the log. If there is another yellow or red request before this request in the DSO, activation terminates. In a second step data is checked against archived data, marked blue in the log. In a third step the activation of data takes place, marked red in the log. During step 3 a number of sub-jobs "BIBCTL_xxxxxxxxxxxxxxxxxxxxx" with a unique 25-letter-GUID at the end are scheduled. This is done for the reason to get a higher parallelism and so a better performance.

But how is the data split up into the BIBCTL* jobs? How does the system know, how many jobs should be scheduled? I will answer this question in chapter 4.

I will show you in chapter 4, which settings can be the reason for these long-running activations. After the last "BIBCTL*" has been executed the SID activation will be started. Unfortunately this is not written to the job log for each generation step but at the end if the last of your SID generation jobs has been finished.

Let's look at the details and performance settings, how they influence DSO activating so that you may reduce your DSO activation time.
4. Transaction for DSO settings

You can view and change this DSO settings with "Goto->Customizing DataStore" in your manage DSO view. You’re now in transaction RSODSO_SETTINGS. In Reference 4 at the end of this document you can find a note for runtime parameters of DSO.

**Maintenance of Runtime Parameter of DataStore Objects**

As you can see, you can make cross-DataStore settings or DataStore specific settings. I choose cross-DataStore and choose "Change" for now. A new window opens which is divided into three sections:

**Picture 3: RSODSO_SETTINGS**

Section 1 for activation, section 2 for SID generation and section 3 for rollback. Let's have a look at the sections one after another.

**Picture 4: Parameters for cross-datastore settings**
5. Settings for activation

In the first section you can set the data package size for activation, the maximum wait time and change process parameters.

If you click on the button "Change process params" in part "Parameter for activation" a new popup window opens:

![Settings for Parallel Processing](image-url)

**Picture 5: Job parameters**

The parameter described here are your default '3' processes provided for parallel processing. By default also background activation is chosen. You can now save and transport these settings to your QAS or productive system. But be careful: These settings are valid for any DSO in your system. As you can see from picture 4 the number of data records per data package is set to 20000 and wait time for process is set to 300, it may defer from your system. What does this mean? This means simply that all your records which have to be activated are split into smaller packages with maximum of 20000 records each package. A new job "BIBCTL**" is scheduled for each data package.

The main activation job calculates the number of "BIBCTL**" jobs to be scheduled with this formula:

\[
\text{Maximum number of jobs} = \text{Max(parallel processes for activation, number of records in new data)} \div \text{maximum package size} + 2.
\]

It is the maximum of your number of available processes for activation and the integer part of your number of records divided by your maximum package size plus 2. One process for the fractal part of your division and one control process for your BI_ODSA job. So if you have 10000 records only to activate and four processes for activation, what happens? The result of the Formula Max (4, 2) is 4. Your BW will schedule 3 jobs and split your 10000 records into your 3 processes to 3333 each.

The default setting of 20000 records can be enough for small DSO or delta DSO where you expect only few data per load. But if you have mass data like in an initial load for CO-PA with millions of records you should increase this parameter to speed up your CO-PA activation. Keep this rule of thumb in mind taken from Note 1392715:

\[
\text{Number of records to activate at once} = \text{Number of requests} \times \text{Sum of all records in these requests} \div \text{Package size of the activation step} \leq 100.000.
\]

This rule is implemented by note 1157070.
So if you have only a couple of records to activate it can make sense to set your number of parallel processes to 2 instead of 3. For mass data with expected number of millions of records you can set the number of parallel process to a higher value, for example 6, so that you have 5 processes for activation and 1 control process. This parameter depends on your machine size and memory.

In this note 1392715 you also find SAP some common recommendations for your system and other useful performance hints:

Batch processes \[= (All \text{ Processes}/2)+1\]
Wait Time in Sec. \[= 3 \times \text{rdisp/\text{max\_wprun\_time}}\]
\text{rdisp/\text{max\_wprun\_time}} - RZ11 (Note 25528)

The maximum wait time for process defines the time how long the job is allowed to run and activate data. If the activation job takes longer than defined, BW assumes system problems or database problems and aborts this job. If you have high number of data packages you should set the wait time to a higher value, if you have less data packages set a low value.

One little trick: If you simply count the number of your BIBCTL* jobs in the job log and multiply it with maximum package size, you know how many records have been activated and how many records still have to be activated. As further approximation you can take the time how long each activation job BIBCTL* runs and multiply it by the maximum number of jobs you can calculate how long your activation job will run.

You can change the parameters for data package size and maximum wait time for process and they will be used for the next load.

6. Settings for SID generation

In section SID generation you can set parameters for maximum Package size and wait time for processes too. With button "Change process params" popup described in picture 5 appears. In this popup you define how many processes will be used for SID generation in parallel. It's again your default value. Minimum package size describes the minimum number of records that are bundled into one package for SID activation. With SAP Layered Scalable Architecture (LSA) in mind, you need SID generation for your DSO only if you want to report on them and have queries built on them. Even if you have queries built on top of DSO without SID generation at query execution time missing SIDs will be generated, which slows down query execution. For more information to LSA you can find really good webinar from Jürgen Haupt in the references at the end of the document. Unfortunately SID generation is set as default if you create your DSO. My recommendation is: Switch off SID generation for any DSO! If you use the DataStore object as the consolidation level, SAP recommends that you use the write-optimized DataStore object instead. This makes it possible to provide data in the Data Warehouse layer 2 to 2.5 times faster than with a standard DataStore object with unique data records and without SID generation! See performance tips for details.

From performance tips for DataStore Objects in help.sap.com you can also find this performance table and how the parameters "SID generation" and "Unique records" influence DSO activation:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Saving in Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation of SIDs During Activation Unique Data Records</td>
<td>x</td>
</tr>
<tr>
<td>Generation of SIDs During Activation Unique Data Records</td>
<td></td>
</tr>
<tr>
<td>Generation of SIDs During Activation Unique Data Records</td>
<td>approx. 35%</td>
</tr>
<tr>
<td>Generation of SIDs During Activation Unique Data Records</td>
<td>approx. 45%</td>
</tr>
</tbody>
</table>

The saving in runtime is influenced primarily by the SID determination. Other factors that have a favorable influence on the runtime are a low number of characteristics and a low number of disjointed characteristic attributes.
7. Settings for Rollback

Finally last section describes rollback. Here you set the maximum wait time for rollback processes and with button “Change process params” you set the number of processes available for rollback. If anything goes wrong during activation, e.g. your database runs out of table space, an error during SID generation occurs, rollback will be started and your data is reset to the state before activation. The most important parameter is maximum wait time for Rollback. If time is over, rollback job will be canceled. This could leave your DSO in an unstable state. My recommendation set this parameter to a high value. If you’ve large amount of data to activate you should take at least double the time of maximum wait time for activation for rollback. You should give your database enough time to execute rollback and reset your DSO to the state before activation started.

Button “Save” saves all your cross-datastore settings.

8. DataStore-specific settings

For a DataStore-specific setting you enter your DSO in the input field as you can see from picture 3. With this DSO local setting you overwrite the global DSO settings for this selected DSO. Especially if you expect to have very large DSOs with lot of records you can change your parameters here. If you press button “Change process params” the same popup opens as under global settings, see picture 5.

9. Activation in Process chains

I explained the settings for manual activation of requests in a standard DSO. For process chains you have to create a variant for DSO activation as step in your chain, see picture 6. In this variant you can set the number of parallel jobs for activation accordingly with button ”Parallel Processing”.

Other parameters for your standard DSO are taken from global or local DSO settings in TA "RSODSO_SETTINGS" during run of the process chain and activation.
Related Content

Reference 1: Webinar about Layered Scalable Architecture
Reference 2: DSO request activation: collective performance problem note
Reference 3: Performance tips for DataStore Objects
Reference 4: Runtime parameters of DataStore Objects