

Z Accelerators - Developers Guide

8.2 — Last update: 2018/03/20

Basis Technologies

Table of Contents

Audience	3
Introduction	4
Prerequisites	5
Basic Concepts	6
Architecture.....	7
Diffuser	8
MiniCube.....	9
Instance	10
Intervals	11
Capacity Groups	13
When to use Z Accelerators.....	14
Main Program	15
General.....	16
Declaration.....	17
Selection Screen Definition	18
START-OF-SELECTION Event	19
Interval Processing Subroutine.....	20
Collating Interval Results	22
AT SELECTION-SCREEN OUTPUT	24
Transformation Program.....	25
Regular Transformation Program	26
Selection Screens and Transformation Programs.....	28
Setting up a Diffuser program	29
Program Definition	30
Default Technical Settings.....	32
Interval Generation	34
Running Diffuser Programs	36
Technical Settings.....	37
Developer Notes	41
Administering Diffuser Programs.....	43
Diffuser Mode	44

Results	45
Intervals	46
Variants	47
App Servers	48
Increase or Decrease Jobs	49
Decrease Jobs	50
Pause	51
Resume	53
Delete	54
Force Error.....	55
Reprocess Error.....	56
Rename Instance	59
Debug an Interval.....	61
Scheduling Diffuser Programs	63
Exception Handling and Application Logs.....	64
Email Notification	66
Debugging and Troubleshooting	68
Debugging Programs	69
Troubleshooting	70
Advanced Concepts	71
Dynamic Interval Generation	72
Authority Checks	75
Authorisations.....	76
Implementation	79
Technical Settings	81
Diffuser Mode	85
Individual Actions.....	87
Application Program Interface	89
Overview	90
Selecting Instances.....	91
Get Details.....	93
Pause	94
Restart.....	95
Change Jobs	96
Software Support.....	97
Online Forum	98

Support from Basis Technologies	99
---------------------------------------	----

Audience

This guide has been developed for the following audience:

- ABAP™ Developers intending to use Z Accelerators
- Performance Specialists or Advisers
- SAP® Architects

Introduction

Basis Technologies' Diffuser provides a framework and run-time environment for custom developed SAP reports that must work with large volumes of a program running this we call a Z Accelerator. It provides for use of multiple processors with custom reports such that these programs can run within acceptable time constraints to deliver timelier business information.

Z Accelerators can be easily developed using the standard SAP ABAP Workbench. The Z Accelerators methodology guides the developer, by making them adhere to a structured program design and standardized technical program structure that separates data processing logic from presentation logic. For each Z Accelerator, there are three main components that need to be set up. This refers to the Diffuser definition and also two programs in the ABAP repository for each Z Accelerator, these three components are:

1. Main Program – This ABAP program retrieves data from the database, processes this information and then stores the data
2. Transformation Program – This ABAP program retrieves the data that was processed in the main program and presents this to the user
3. [Diffuser](#) Definition – This defines the name of the technical components of each Z Accelerator, as well as a number of other settings relevant to using the [Diffuser](#)

This guide will take you firstly through the Basic concepts of using Z Accelerators and then get you started on how to develop your own Z Accelerators. This will allow you to write ABAP reports that process large volumes of data in your SAP system.

Prerequisites

Systems (currently supported)

SAP® Web Application server release 7.00 (or higher)

Prerequisites for the SAP UI

Minimum:

SAP® GUI 7.10

Recommended:

Current Version SAP GUI



Older versions of SAP might be supported by older versions of Diffuser please contact us for details

Basic Concepts

The basic concepts of operating Diffuser are shown as below:

- [Architecture](#)
- [Diffuser](#)
- [MiniCube](#)
- [Instance](#)
- [Intervals](#)
- [Capacity Groups](#)

Architecture

For a program to be accelerated by the Diffuser, it can either be developed as a custom Z Accelerator or provided as a prepackaged program supplied by Basis Technologies (as a GT, GTi or BDi App). The key features to accelerate a program are the [Diffuser](#) and [MiniCube](#).

Diffuser

Diffuser takes a large data set and splits it up into small pieces of data called [Intervals](#) these can then be processed with multiple processors running. The Diffuser allows the number of processors to be increased or decreased at runtime, including pausing and restarting a run. A group of processors can be assigned to a Capacity Group to allow processing power to be dynamically distributed across programs.

This can be utilised by custom programs as a Z Accelerator or by programs supplied prepackaged by Basis Technologies (the GT, GTi and BDi Apps).

MiniCube

When a Diffuser run completes it can store the results as a MiniCube inside the architecture, these can be retrieved and further selections applied to this data. The data can also be supplied in an interactive way so that for example ALV or drilldown features can be used.

The transaction /BTR/MINICUBE also allows the monitoring of runs and with the Diffuser Mode the instance resources can be increased at runtime, see [Administering Diffuser Programs.](#)

Instance

Each run of a program using the [Diffuser](#) is called an 'Instance'. This can be given a separate label and, using the framework, a user can view a number of previous instances and pick which they wish to view the details of. The framework allows the saving of data against the instance.

An instance can have the following states:

Technical Code	Description
10	Created
20	Generating Intervals
25	Intervals Generated (Ready to Process)
30	In Process
40	Error
50	Stopped
55	Stopping
60	Ready to Collate
70	Collating
80	Finished

Intervals

[Diffuser](#) works on the principal that data processing can be divided into independent “pieces” of work. These pieces of work are referred to as Intervals and usually represent a range of master or transactional data that needs to be worked upon. This, for example, might be Intervals of Business Partner Numbers, Sales Order Numbers, or Account Numbers. Diffuser ensures that these Intervals of data are processed using multiple processors instead of the traditional sequential approach with one processor.

An interval can be thought of as a range of values that represent X number of master or transaction data objects. It is based on some data domain having a data type and a length. An example interval might be “All Sales Order’s from 1000 through to 2000”. It can then be implied that an interval has a Low value (e.g. 1000) and a High value (e.g. 2000).

If a SAP system has 100,000 Sales Orders, and an ABAP report is required to process all of them, then this range of Sales Orders can be broken down into, for example, 100 intervals, each representing 1000 Sales Orders. The list of 100 intervals might look something like:

Interval	Low	High
1	1	1000
2	1001	2000
3	2001	3000
...
100	99001	100000

The concept of an Interval Object is used to create these Intervals for use by a Diffuser program this is where an object such as a Sales Orders are broken down before the Diffuser program is run. As part of the definition of a Diffuser program, an Interval Object is selected. This refers directly to the type of Intervals the program uses, for example, some Interval Objects that are provided with Diffuser are:

- Sales Order
- Business Partner
- Contract Account

The generation of Interval Objects is detailed in the Setting up a Diffuser program [Interval Generation](#) section.

When using Z Accelerators intervals can also be built at runtime. This is programmed into the Diffuser program and is useful where you have complex intervals or need to define the number of intervals down to a

smaller size; a subset of materials, for example. This concept is detailed in the Z Accelerators – Advanced Concepts – [Dynamic Interval Generation](#) section.

An interval can have the following states:

Technical Code	Description
01	Available
02	Selected
03	In Process
04	Error
05	Stopped
06	Completed

Capacity Groups

Capacity Groups is a powerful tool of the Diffuser created to enhance its system resource administration capabilities. While the Diffuser provides a parallel processing platform and runtime environment which makes the execution of ABAP code faster and more efficient, Capacity Groups offer an advanced administration framework for the consumption of system resources by Diffuser enabled programs. The tool determines how many background processes a Diffuser program can use on one or more selected servers, on a specific time pattern, and its relative priority to other Diffuser programs running at that point.

When to use Z Accelerators

Z Accelerators are essential when the data volumes are so large that the processing time to run your reports is unacceptable. If you are able to write a report that runs within acceptable time-constraints then the use of Z Accelerators may not be required. However, even if your report runs within 2 hours (and this is considered acceptable) you are still able to use Z Accelerators to bring this run-time down even further.

It is our consideration that almost any report that executes in the background can gain from being run using Z Accelerators. This is because the Z Accelerator format not only promotes performance improvements, but also cost reductions in maintenance by having a generic format. It also provides the benefit of separating the processing and presentation logic. Separate presentation logic allows the data to be viewed interactively and in a user-friendly manner – far easier than lengthy static list output.

Main Program

This section takes you through the steps to construct the main program that uses Z Accelerators.

- [General](#)
- [Declaration](#)
- [Selection Screen Definition](#)
- [START-OF-SELECTION Event](#)
- [Interval Processing Subroutine](#)
- [Collating Interval Results](#)
- [AT SELECTION-SCREEN OUTPUT](#)

General

The Main Program is a simple “Type 1” (Executable) ABAP program/report and can be created or changed via the transaction /BTR/DIFFUSER, or the standard SAP transaction SE38. For ease we recommend the transaction /BTR/DIFFUSER. Within the Main program, the basic structure of a custom ABAP report is:

- Declaration – Declaration of working variables, include files, type declarations and so forth
- Selection Screen – Parameter screen definition of select-options and parameters
- Start-Of-Selection Event – Program Initialization
- Interval Processing Subroutine – Initial Data Restriction logic and Processing logic
- Collating Interval Subroutine – Collation logic to combine interval results (This is optional)

Note that there is no mention of result presentation in the Main Program; this is implemented in the Transformation program. This means that you should not do the following in the Main program:

- Use the WRITE statement for display to the SAP spool or screen (You can of course use the WRITE statement to format data into variables)
- Use the AT USER-COMMAND or AT LINE-SELECTION processing blocks
- Declare headings or text elements that are to be used for presentation
- Perform any sort of presentation logic

Declaration

When you create the Main program, you must include /BTR/MDR_INCLUDE. This can be done at the start of the Main program as in the below example:

```
*-----*
* Sample Diffuser Program
*-----*
REPORT Z_SAMPLE_PROGRAM.

INCLUDE:
  /btr/mdr_include.
```

This is also where you should include other declarations such as other includes, type declarations, global data declarations and table statements.

Selection Screen Definition

Almost all custom ABAP reports enable the user (or background variant) to supply select-options to restrict the data that is retrieved and processed. Standard ABAP reports allow the developer to use the keywords:

- SELECTION-SCREEN
- SELECT-OPTIONS
- PARAMETERS

In your Diffuser program, you must define the select-options and parameters of a program within the following Diffuser statements:

mdr-begin-selection-screen and mdr-end-selection-screen

Everything else regarding syntax remains exactly the same. This allows the [technical settings](#) button to be seen on screen.

An example declaration of parameters and select-options is as follows:

```
*-----*
* PARAMETERS and SELECT-OPTIONS
*-----*

mdr-begin-selection-screen.

SELECTION-SCREEN BEGIN OF BLOCK general WITH FRAME TITLE text-t01.
PARAMETERS:
  p_bukrs TYPE vbak-bukrs OBLIGATORY,    p_keydat TYPE d OBLIGATORY.

SELECT-OPTIONS:
  s_belnr FOR vbak-belnr,    s_crdat FOR vbak-erdat.
SELECTION-SCREEN END OF BLOCK general.

mdr-end-selection-screen.
```

START-OF-SELECTION Event

The START-OF-SELECTION event must be defined in the Diffuser main report. However, unlike standard ABAP reports this event does nothing except execute an Diffuser statement. This statement must immediately follow the START-OF-SELECTION keyword and nothing else should follow it. In effect, this single Diffuser statement begins processing the entire report. The following is a sample of how the START-OF-SELECTION event should appear:

```
*-----*  
* START-OF-SELECTION event  
*-----*  
START-OF-SELECTION.  
    mdr_program_initialize.
```

This statement will begin execution of the Diffuser program. Intervals will be generated (or retrieved) and one or more jobs started. For each interval, the interval processing subroutine will be called (defined next).

Interval Processing Subroutine

The special subroutine `mdr_interval_processing` is called once for each interval to be processed. It is passed a data dictionary structure of type `/BTR/ST_INTERVAL_VALUES` that has a number of fields defined. The most important of these fields is “LOW” and “HIGH”. At the start of the interval processing, you should typecast these two values to your own local variables.

The next step is to retrieve the data you require in your application specific program. However, you need to ensure that your program only retrieves the data relevant for that particular interval. This means your `SELECT` statements (if appropriate) must use the LOW and the HIGH values in the restriction of data retrieved.

This differs only slightly from a standard ABAP report. Put simply, the selection of the master/transactional data must be extended to include a further `WHERE` clause.

Once all data in the interval has been retrieved and processed appropriately, you need to save off the results of the processing. This is done using the macro `mdr_interval_result_put`. It expects two parameters (1) a label identifying the data and (2) the data to be saved. The data to be saved can take any form whether it is a locally declared type or a data type defined in the data dictionary. The data can also be an internal table or even a complex type containing both structures and internal tables.

An example Interval Processing subroutine is shown below:

```
*-----*
* FORM mdr_interval_processing
*-----*
* This form is called by the Diffuser to process each
* interval range. Results for each interval should be calculated
* within this subroutine and be saved for later collation
*-----*
FORM mdr_interval_processing
  USING x_interval TYPE /btr/st_interval_values.

DATA:
  lt_vbak      LIKE vbak OCCURS 0 WITH HEADER LINE,
  lv_belnr_low TYPE vbak-belnr,
  lv_belnr_high TYPE vbak-belnr.

* Type cast the interval low and high values
  lv_belnr_low = x_interval-low.   lv_belnr_high = x_interval-high.

* Retrieve data
SELECT *
  FROM vbak
  INTO TABLE lt_vbak
  WHERE belnr BETWEEN lv_belnr_low AND lv_belnr_high
```

```
    AND buhrs = p_buhrs.  
  
* Process data  
  PERFORM process_data TABLES lt_vbak.  
  
* Save results of interval  
  mdr_interval_result_put 'RESULTS' gt_results.  
  
ENDFORM.
```

✿ Note that the interval low and high range values have been typecast at the start of the subroutine into the local variables. The data is then retrieved using the LOW and HIGH values to restrict the list of Sales Orders returned. Furthermore, the parameters declared in the Selection screen definition are also used to restrict the data that is retrieved.

Once the data has been retrieved, a subroutine is called to process the data. This subroutine uses the Sales Order data and accumulates a global internal table called GT_RESULTS. It is not important what logic this subroutine performs, or the structure of this data, but rather that once this internal table is built, it is then saved using the statement `mdr_interval_result_put`.

Note that you can save multiple results for a particular interval. For example:

```
* Save results of interval  
  mdr_interval_result_put 'RESULTS' gt_results.  
  mdr_interval_result_put 'OTHERRESULTS' gt_other_results.
```

✿ The important point to note is that the label must be unique. Now that the results have been determined and saved for the interval, the results of all intervals can be combined into a single “final” result. This is the function of the Collation subroutine, which is described in the following section.

Collating Interval Results

After all the intervals of the program run have been processed, Diffuser calls a special subroutine to collate the interval results back together into a single result. Once collated, sets of results exist for the program run. The logic required to collate the interval results is often quite simple but will vary depending upon the nature of the report.

In order to develop your own collation routine, it is a simple matter of adding a new subroutine to the Main program. This subroutine must be called `mdr_interval_collation`. It has a single parameter passed to it, which is of type `/BTR/TT_MDR_INTERVALS`. This parameter contains the complete list of intervals and the associated result(s) that have been calculated for each interval in the interval processing subroutine. It is the task of the Collation routine to loop through each interval and retrieve the result for that interval. It should then combine the data results of each interval together to form one large combined result. An example collation routine is shown below:

```
*-----*
* FORM mdr_interval_collation
*-----*
* This form is called by once all intervals have been processed.      *
* It can be used to collate the results of the intervals together.    *
*-----*
FORM mdr_interval_collation
  USING xt_intervals TYPE /btr/tt_mdr_intervals.
  DATA:
    lv_interval    LIKE LINE OF xt_intervals,
    lt_summary     LIKE gt_summary_list[],
    lv_parameters  TYPE ty_parameters.

  REFRESH gt_summary_list.

  LOOP AT xt_intervals INTO lv_interval.
*   Get the summary results for this interval
    mdr_interval_result_get lv_interval co_label_summary lt_summary[].

*   Accumulate in the collated summary
    LOOP AT lt_summary INTO gt_summary_list.
      COLLECT gt_summary_list.
    ENDLOOP.
  ENDLOOP.

* Save the summary list for access by the transformation program.
  mdr_instance_result_put co_label_summary gt_summary_list[].
ENDFORM.
```


As can be seen in the example code the steps of the collation routine are to:

1. Loop through each interval passed into the subroutine
2. For each interval, retrieve the results that have been calculated by the interval processing routine by using the statement `mdr_interval_result_get`
3. For each result of the interval, combine the data into a “complete” result
4. Store the “complete” result back into Diffuser using the statement `mdr_instance_result_put`

Let's first look at the statement `mdr_interval_result_get`.

```
mdr_interval_result_get lv_interval co_label_summary lt_summary[].
```

The first parameter of this statement “`lv_interval`” is the interval detail that we are retrieving the data from; this is simply the appropriate row of the parameter `xt_intervals`.

The second parameter of this statement “`co_label_summary`” is a character string that represents the label of the data. This must be the label that was used to originally store the result during the interval processing.

The final parameter is the result variable itself “`lt_summary[]`”. This must have the same structure as was originally stored when the result was “put” during the interval processing routine using the statement `mdr_interval_result_put`. It is crucial that this data type is the same.

Once the Collation subroutine executes, the program will now have a single “complete” result (or results) for the entire interval set. This complete result is a combination of the results of each interval of work as though one job had executed the entire report.

With the processing component of your Diffuser program finished and the end result collated – the final step is to present this data to the user.

AT SELECTION-SCREEN OUTPUT

If you intend to use the AT SELECTION-SCREEN OUTPUT event in your Diffuser program, you will find that Diffuser does not allow you to use this, this is due to some constraints in defining multiple AT SELECTION-SCREEN OUTPUT events in a program (Diffuser also uses this for the technical settings tab of your program). You can still implement this code by creating the subroutine `AT_SELECTION_SCREEN_OUTPUT` in your Main Diffuser Program as below. There are no parameters required to this form, and you can access the `SCREEN` structure as normal.

```
*-----* FORM at_selection_screen_output *-----*
* This form allows custom AT SELECTION-SCREEN OUTPUT events to *
* be used                                                         *
*-----*
FORM mdr_at_selection_screen_output.

* Here you can access the screen variable as if you were using
* the standard AT SELECTION-SCREEN OUTPUT event. All other
* report events can be defined as normal.
LOOP AT SCREEN.
    CASE screen-name.
        WHEN 'S_CARRID-LOW'.
        WHEN 'S_CARRID-HIGH'.
    ENDCASE.

    MODIFY SCREEN.
ENDLOOP.

ENDFORM.                                "at_selection_screen_output
```

Transformation Program

- [Regular Transformation Program](#)
- [Selection Screens and Transformation Programs](#)

Regular Transformation Program

As mentioned earlier, Diffuser requires that the processing and presentation components of a report/program are separate. This section describes how the Transformation (Presentation) program is to be structured.

The Transformation program is its own stand-alone program with a different name to that of the Main program. The common component of the Transformation and Main program should be the data type(s) of the results. If you have defined your own data type to store the result sets, then either make sure they are defined in the Data Dictionary, or they have been defined in a common include file for use by both the Main and Transformation programs.

The Transformation program is designed to run every time that the user is to be presented with the end result data. This means it needs to first retrieve the results and then present them to the user. One of the major benefits of having the Transformation program separate to the Main program is that it can have its own select-options/parameters. This can be used to further restrict the “collated” data when it is presented. An example transformation program is shown below:

```
*-----*
* Sample Transformation Program
*-----*
REPORT z_sample_mdr_transform_program.

INCLUDE:
  /btr/mdr_include.

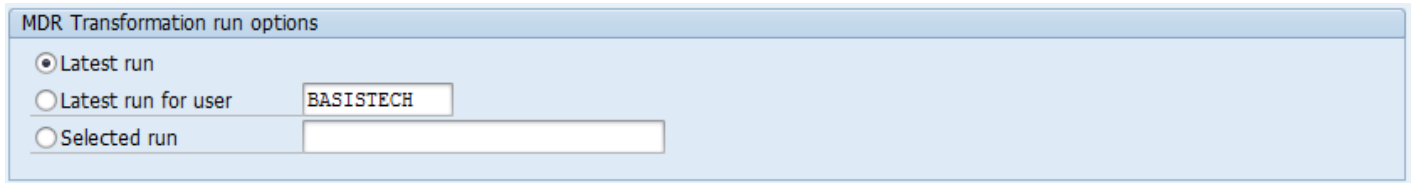
* Transformation run options
  mdr-begin-select_screen_trans.

START-OF-SELECTION.

* Get the summary results
  mdr_instance_result_get co_label_summary gt_summary_list[].

* Display the details to the user
  PERFORM display_result USING gt_summary_list.
```

The statement `mdr-begin-select_screen_trans` must be used before the `START-OF-SELECTION` event. This allows the transformation program to be run separately from the run history and doing so will produce the options below the main part of the selection screen allowing the user to check their latest runs or select runs.



The image shows a dialog box titled "MDR Transformation run options". It contains three radio button options: "Latest run" (which is selected), "Latest run for user", and "Selected run". To the right of the "Latest run for user" option is a text input field containing the text "BASISTECH". To the right of the "Selected run" option is an empty text input field.

The next statement in the transformation program is `mdr_instance_result_get`. This statement will retrieve the “complete” result that was calculated in the Collation subroutine. The label that is passed is very important in that it identifies which result you are trying to retrieve since there may be more than one. The last parameter to the statement is the actual variable that you have defined and must be the exact type as what you have defined in the Collation subroutine when you saved the “complete” result.

Now that the complete result (or results) has been retrieved, you can display them to the user. It is up to you how you do this. In the example on the previous page, a subroutine has been defined that will perform whatever steps are required to display the data. This might involve using `WRITE` statements and outputting to the spool. In many cases, a better approach would be to use the ABAP List viewer and call the `REUSE*` function modules. There are no restrictions to the presentation possibilities.

An important feature of the Transformation program is that it can be interactive. Hence the `AT USERCOMMAND` and `AT LINE-SELECTION` events can be implemented, unlike a traditional batch report that would have the results stored in the spool. The Transformation program will be executed every time a user decides to view the results of a Diffuser program run.

Selection Screens and Transformation Programs

Transformation programs can also be enhanced with selection screens to enable the data retrieved from the Diffuser framework to be filtered as per the example below filtering by company code. Or alternatively selection screen options could impact the output or offer file download options in the usual way you might control any ABAP program.

```
*-----*
* Sample Transformation Program
*-----*
REPORT z_sample_mdr_transform_program.

INCLUDE:
  /btr/mdr_include.

SELECTION-SCREEN BEGIN OF BLOCK seloptions WITH FRME TITLE texts01.

SELECT-OPTIONS s_bukrs FOR vbak-bukrs.

SELECTION-SCREEN END OF BLOCK seloptions.

* Transformation run options
  mdr-begin-select_screen_trans.

START-OF-SELECTION.

* Get the summary results
  mdr_instance_result_get co_label_summary gt_summary_list[].

* Filter the summary results
  delete gt_summary_list where bukrs not in s_bukrs.

* Display the details to the user
  PERFORM display_result USING gt_summary_list.
```

Setting up a Diffuser program

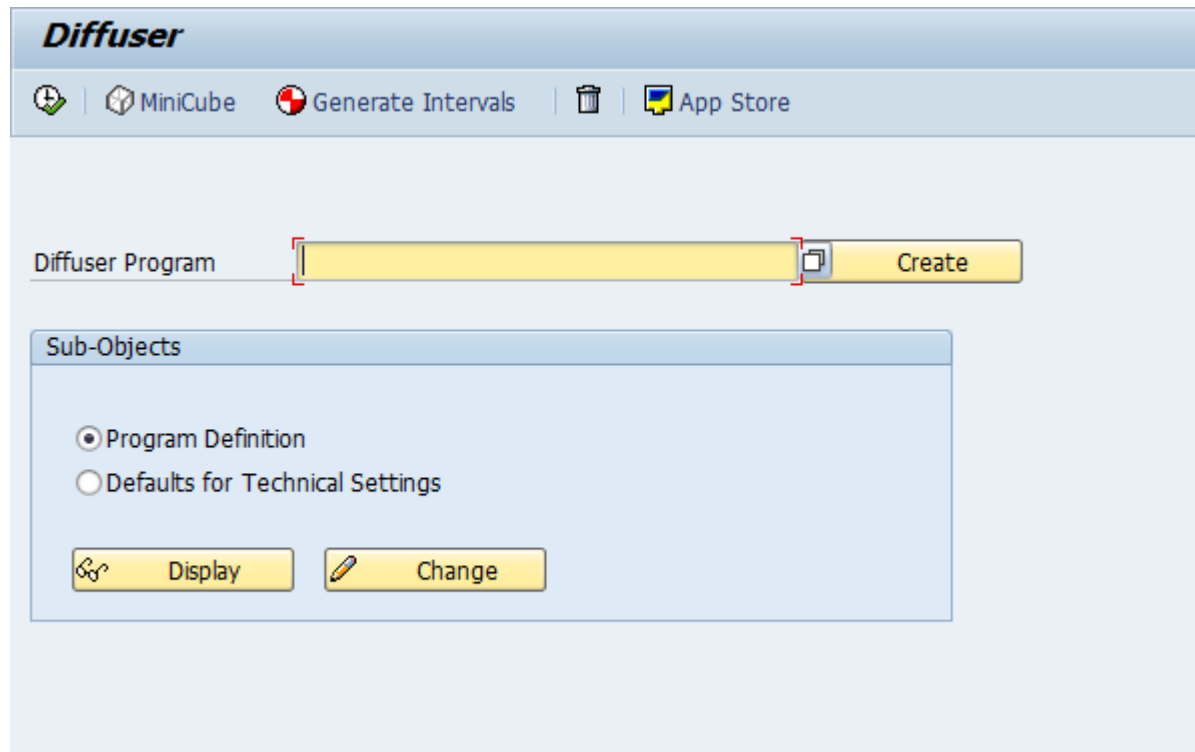
For a program to use Diffuser, it can either be developed as a custom Z Accelerator or provided as a prepackaged program supplied by Basis Technologies (as a GT, GTi or BDi App).

To setup a program to use Diffuser use the transaction /BTR/DIFFUSER here as a minimum define the Diffuser program in the Program Definition, then choose to setup the technical settings and generate intervals where required.

- [Program Definition](#)
- [Default Technical Settings](#)
- [Interval Generation](#)

Program Definition

The definition of a Diffuser program is set up via the transaction /N/BTR/DIFFUSER.



The screenshot shows the 'Diffuser' application window. At the top, there is a header bar with the title 'Diffuser' and a toolbar containing icons for MiniCube, Generate Intervals, a trash can, and App Store. Below the header, there is a text input field labeled 'Diffuser Program' with a yellow highlight and a 'Create' button. Below this, there is a 'Sub-Objects' section with two radio buttons: 'Program Definition' (selected) and 'Defaults for Technical Settings'. At the bottom of the 'Sub-Objects' section, there are two buttons: 'Display' and 'Change'.

Enter the program name and press the Create button for new programs or Change button for an existing program with the sub-object as program definition.

The definition is now displayed as below.

Diffuser

MiniCube

Diffuser Program: /BTR/SAMPLE_FLIGHT_REPORT

Diffuser Sample: Flight Report

Transform Prog.: /BTR/SAMPLE_FLIGHT_TRANALV ☐ Launch Transform program

Interval Obj.: Sample: Flight Customers ☐ Generate variant internally

Object: /BTR/MDR Diffuser application logs

Subobject: DEFAULT Deafult sub-object

Job Format ID:

Main Program: Diffuser Forms Interval Generation Interval Processing Interval Collation

The main program is already populated from the first screen. A transformation program can now be configured if required.

The “Interval Object” object is also populated here, refer [here](#) for more information on intervals.

The program definition also allows the user to configure which application log object and sub-object any messages are written to that are called during the execution of the program. The default object and sub-object are /BTR/MDR/ and DEFAULT respectively.


Furthermore, the transaction N/BTR/DIFFUSER allows the developer to maintain the Main Program and Transformation Program directly instead of using the standard SAP transaction SE38, with the code buttons at the bottom linking directly to the relevant subroutines.

Default Technical Settings

The second sub-object managed through the transaction /BTR/DIFFUSER is “Defaults for Technical Settings”. This screen contains two main sections.

1. “Defaults for Technical Settings” allows to set default values for a specific Diffuser program. Once set, these values will always appear on the Technical Settings’ sub screen for that program see [Technical Settings](#) under Running Diffuser Programs for more details.

Diffuser



Instance Settings

Label

Interval Settings

Perform processing using intervals of

Interval variant

Distribution

☒ Number of batch jobs across all servers


☐ Distribution according to server group

☐ Manual Distribution

☐ Run online as a single process (debugging mode)

Other settings

☐ Wait for run to complete ☐ Launch Transformation Program after completed run

Distribution List 

Message log level

Technical Settings Access

☐ Background Program

☐ Lock Technical Settings

☐ Lock Expert Mode

1. “Technical Settings Access” we will explore in more detail.

Background Program

NOTE: This functionality requires “Wait for run to complete” to be set.

This option suppresses the default display of the Run History screen after completion of an instance run. This is a useful feature that allows a Diffuser program to be called from another program without interrupting the latter with the MiniCube display.

The input field “Check for completion (in sec)” allows to set in seconds a time interval in which the parent job of a running Diffuser instance will wake up and check if all child parallel processes have completed. The default wake up and check time is 30 seconds which is suitable for very long running programs but not for speeding up web services where every second counts.

Lock Technical Settings

This option allows to lock all input fields for Technical Settings. This is useful if when a program can repeatedly run with the same default values and users should not change those values. When this option is set, the Diffuser Mode in the MiniCube will be locked as well.

This restriction applies at program level and not at user level. That is, once set the Technical Settings will be locked for all users. Restrictions at user level can be implemented, see the section [Authority Checks](#) in the Z Accelerators Guide.

Lock Expert Mode

This option is similar to “Lock Technical Settings”. The only difference is that on the Technical settings screen only the input fields under “Distribution” are locked. This allows the user to change settings like label name while protecting the more critical job distribution section from potential misuse. This option applies at program level as well. Restrictions at user level can be implemented with see the section [Authority Checks](#) in the Z Accelerators Guide.

Interval Generation

When a [Diffuser](#) program uses Interval Objects, an Interval Variant needs to be created from the Interval Object, before the Diffuser program is run. An Interval Variant can be thought of as the set of Intervals that the Diffuser program is going to use. It is necessary that new Interval Variants are generated regularly (potentially before each batch run) to ensure that the intervals are split evenly as the data in the system grows.

To use an Interval Objects, they must first be configured into the framework via table /BTR/INTVALOBJ.

There are two different types of Interval Objects; standard SAP Mass Run Interval Objects and Diffuser Interval Objects. They have similar operation except for the generation of the Interval Variants. The Intervals (or Interval Variant) are created before the Diffuser program is executed; this is done via the program /BTR/MDR_INTERVAL_GENERATION or alternately for standard SAP Interval variants via transaction FQD2. Either an Interval Count or Interval Size can be used as parameters to how the Intervals get generated.

Diffuser: Generate Interval Variant

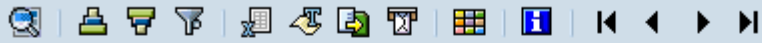
Interval Object: Sample: Flight Customers

Interval Variant: INT:CNT:10

Intervals	
Interval Size	
Interval Count	10

The above example shows the generation of a new Interval Variant called INT:CNT:10 from the Interval Object "Customer Interval Object", with the requirement that 10 Intervals are to be created.

As can be seen below, the result is 10 generated Intervals.

Diffuser: Generate Interval Variant

Int.Object	Variant	Interval	Low	High
SCUSTOM	INT:CNT:10	1		00000514
SCUSTOM	INT:CNT:10	2	00000514	00000978
SCUSTOM	INT:CNT:10	3	00000978	00001442
SCUSTOM	INT:CNT:10	4	00001442	00001906
SCUSTOM	INT:CNT:10	5	00001906	00002370
SCUSTOM	INT:CNT:10	6	00002370	00002834
SCUSTOM	INT:CNT:10	7	00002834	00003298
SCUSTOM	INT:CNT:10	8	00003298	00003762
SCUSTOM	INT:CNT:10	9	00003762	00004226
SCUSTOM	INT:CNT:10	10	00004226	99999999

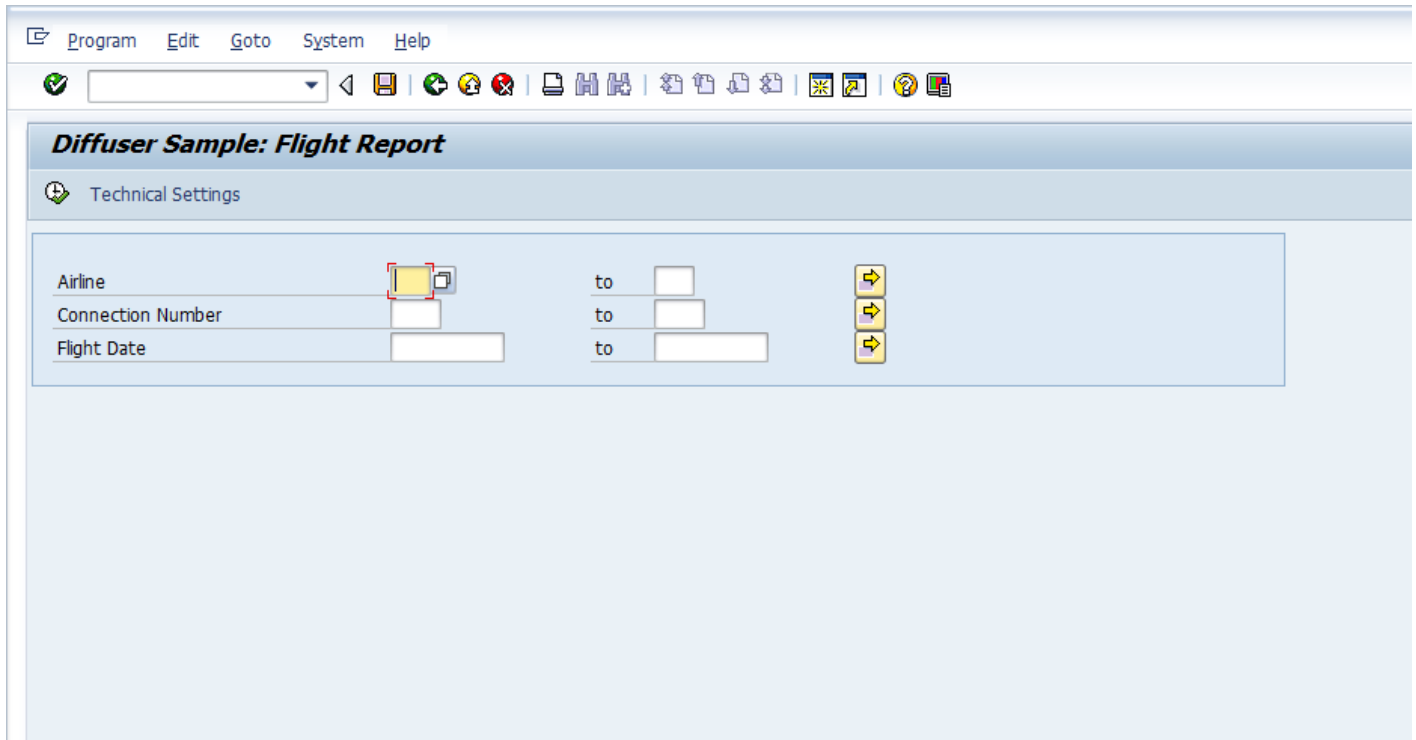
Running Diffuser Programs

The following sections should be considered before running a Diffuser program

- [Technical Settings](#)
- [Developer Notes](#)

Technical Settings

The key part that the user sees is the “Technical Settings” button as below.



If you select the “Technical Settings” button, you will be prompted for Diffuser specific technical settings.

The screenshot shows a software window titled "Diffuser Sample: Flight Report". It contains several sections for configuring a Diffuser program:

- Instance Settings:** A text field labeled "Label" contains the text "Lufthansa Flights".
- Interval Settings:** Two dropdown menus. The first is labeled "Perform processing using intervals of" and shows "Sample: Flight Customers". The second is labeled "Interval variant" and shows "INT:100 : 99 intervals".
- Distribution:** Four radio buttons:
 - ☒ Number of batch jobs across all servers: A text field next to it contains the number "5".
 - ☐ Distribution according to server group: A dropdown menu.
 - ☐ Manual Distribution: Two empty text fields.
 - ☐ Run online as a single process (debugging mode)
- Other settings:**
 - Two checkboxes: "Wait for run to complete" and "Launch Transformation Program after completed run", both are unchecked.
 - "Distribution List": A text field with an empty dropdown arrow icon.
 - "Message log level": A dropdown menu showing "Other".

At the bottom right, there is a toolbar with icons for "Check", "Save", and "Close".

These “Technical Settings” are important when executing an Diffuser program be it in a production environment, or when performing unit testing of your Diffuser program.


Note that you can set up [Default Technical Settings](#) and set up user authorizations to control the users ability to change these settings, for more information refer to the Z Accelerators – Developers Guide [Authority Checks](#).

An explanation for the function of each field on the “Technical Settings” screen is as below:

- Label – The first is a label that can be specified to identify this particular execution.
- Perform processing using intervals of – This is the interval object confirmed in the [Program Definition](#)
- Interval Variant – The Interval variant provides you with a list of different pre-generated Intervals. As detailed in the section [Interval Generation](#), the interval variants are pre-generated using program /BTR/MDR_INTERVAL_GENERATION
- Number of batch jobs across all servers – This specifies the number of processes with which to run the Diffuser program.

- Distribution according to server grouping – This allows the distribution of jobs over one server group to control the number of processors available to this Diffuser instance.
- Manual Distribution – The server grouping above can also be distributed manually.
- Run online as a single process (debugging mode) – This is only used when debugging Diffuser programs and ensures the whole program runs sequentially, in a foreground process.
- Wait for run to complete before finishing – This is often used when running Diffuser programs on-line or when executing them via a job scheduler. It will ensure the parent process waits until all child processes have completed. Once all child processes have finished, control is returned to the parent for completion.
- Launch Transformation Program after completed run – This means the MiniCube screen is skipped so the user gets straight to their results when the run completes
- Distribution List – After a Diffuser program completes it is able to send a SAP office document or external email to a set of recipients that can be specified here.
- Message log level – Lower limit for the priority of messages output to the application log. For example, you can restrict output of informational application log messages by increasing the log level via this parameter.

When using Dynamic Intervals as set out in the Z Accelerators – Developers Guide [Dynamic Interval Generation](#) the “Interval Settings” section will change and be replaced with the two options Interval Count and Interval Size introduced, this looks as per the screenshot below.

Instance Settings	
Label	<input type="text"/>
Interval Settings	
Interval Count	<input type="text"/>
Interval Size	<input type="text"/>
Distribution	
<input checked="" type="radio"/> Number of batch jobs across all servers	<input type="text" value="3"/>
<input type="radio"/> Distribution according to server group	<input type="text"/> <input type="text"/>
<input type="radio"/> Manual Distribution	<input type="text"/> <input type="text"/>
<input type="radio"/> Run online as a single process (debugging mode)	
Other settings	
<input type="checkbox"/> Wait for run to complete	
Distribution List	<input type="text"/> 
Message log level	<input type="text" value="Other"/>

The impact of the two fields is as below:

- Interval Count – This specifies the number of intervals (pieces) that the total amount of work to be done is to be broken up into
- Interval Size – This specifies the “number of objects” to be put into each interval to be then worked upon independently

Developer Notes

As a developer, your first step will be to ensure the program runs correctly with a single job, but with more than one interval. You should have the first run option selected for this, with 1 job in the corresponding field. When your interval processing, collation and presentation all work correctly with a single job (but more than one interval) then most likely your program is ready for running in parallel mode on large volumes of data. This can be done using the same run option but with a higher number of jobs specified.

When you execute your Diffuser program and you specify the first run option, the execution will be in the background; Diffuser will immediately start the jobs and take you into the MiniCube Monitoring transaction. Here you are able to refresh the display in order to determine the current status and progress of the execution.

The setting “Wait for run to complete before finishing” ensures that the master job stays active until all N sub jobs have completed. This may be useful if you require another process to begin when this one completes. The “Distribution List” parameter is described in the following section Email Notification.

After the program has completed you will be able to view the results. You can access the run history via transaction /BTR/MINICUBE, or via transaction /BTR/DIFFUSER and the “MiniCube” button. You then view the results by selecting the Program run, and selecting the Transform button (or F5). This will display your results as you have implemented in the Transformation program.

Instance List Edit Goto Settings System Help

MDR: Run History

Transform Application Log Results Intervals Variants Jobs App. Servers

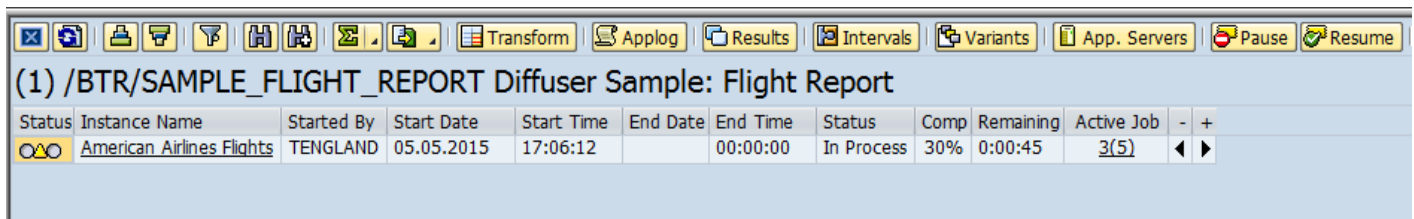
MDR Program	Report title	Run Count
/BTR/MDR_SAMPLE_FLIGHT_REPORT	MDR Sample: Flight Report	1

Instance Name	Started By	Start Date	Start Time	End Date	End Time	Instance Status	Comp	Remaining	Active Job	-	+
American Airlines Flights	TENGLAND	16.01.2015	19:52:56		00:00:00	In Process	45%	0:00:59	4 (9)	◀	▶

Interval	Low	High	Status	Results
1	00000001	00000005	Completed	1
2	00000006	00000010	Completed	1
3	00000011	00000015	Completed	1
4	00000016	00000020	Completed	1
5	00000021	00000025	Completed	1
6	00000026	00000030	Completed	1
7	00000031	00000035	Completed	1
8	00000036	00000040	Completed	1
9	00000041	00000045	Completed	1
10	00000046	00000050	Completed	1
11	00000051	00000056	Completed	1
12	00000057	00000062	Completed	1
13	00000063	00000067	Completed	1
14	00000068	00000072	Completed	1
15	00000073	00000077	Completed	1
16	00000078	00000082	Completed	1
17	00000083	00000088	Completed	1
18	00000089	00000094	Completed	1

Administering Diffuser Programs

The Diffuser provides the advanced user a number of powerful administrative capabilities via the MiniCube transaction /N/BTR/MINICUBE (see screen below). These capabilities provide a powerful way of managing your Diffuser programs.



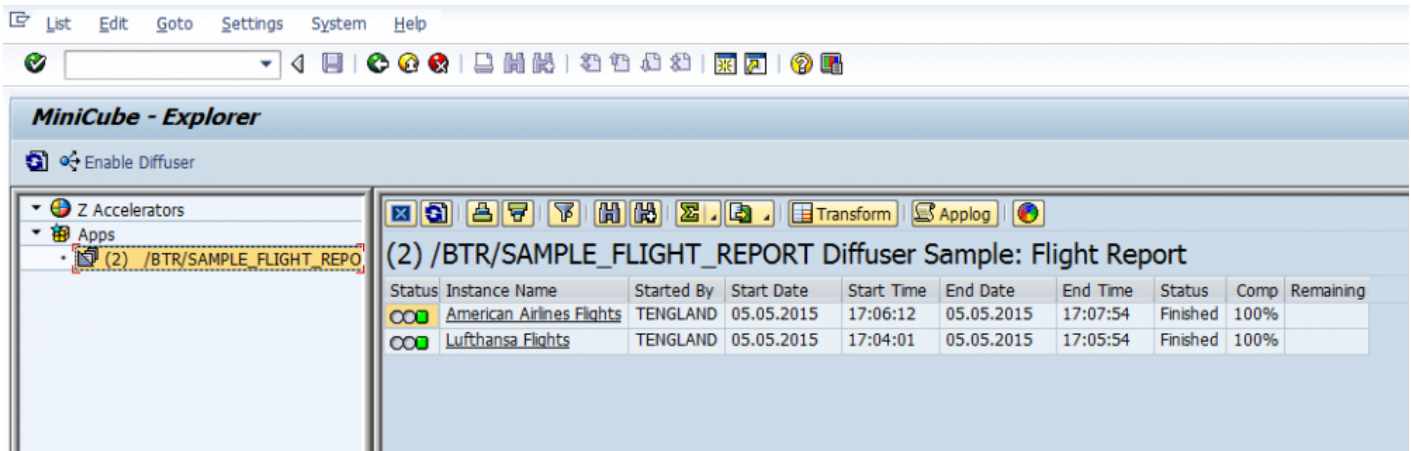
The screenshot shows a software interface with a toolbar at the top containing icons for file operations, a search filter, and buttons for Transform, Applog, Results, Intervals, Variants, App. Servers, Pause, and Resume. Below the toolbar, the title bar reads "(1) /BTR/SAMPLE_FLIGHT_REPORT Diffuser Sample: Flight Report". The main area contains a table with the following data:

Status	Instance Name	Started By	Start Date	Start Time	End Date	End Time	Status	Comp	Remaining	Active Job	-	+
	American Airlines Flights	TENGLAND	05.05.2015	17:06:12		00:00:00	In Process	30%	0:00:45	3(5)	◀	▶

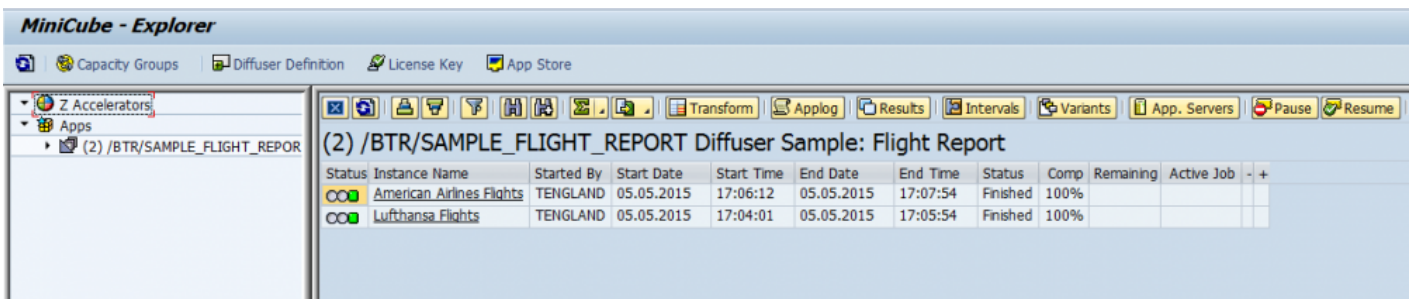
- [Diffuser Mode](#)
- [Intervals](#)
- [Results](#)
- [Variants](#)
- [App Servers](#)
- [Increase or Decrease Jobs](#)
- [Pause](#)
- [Resume](#)
- [Delete](#)
- [Force Error](#)
- [Reprocess Error](#)
- [Debug an Interval](#)
- [Rename Instance](#)

Diffuser Mode

To access the expert mode click the Diffuser mode button as below.

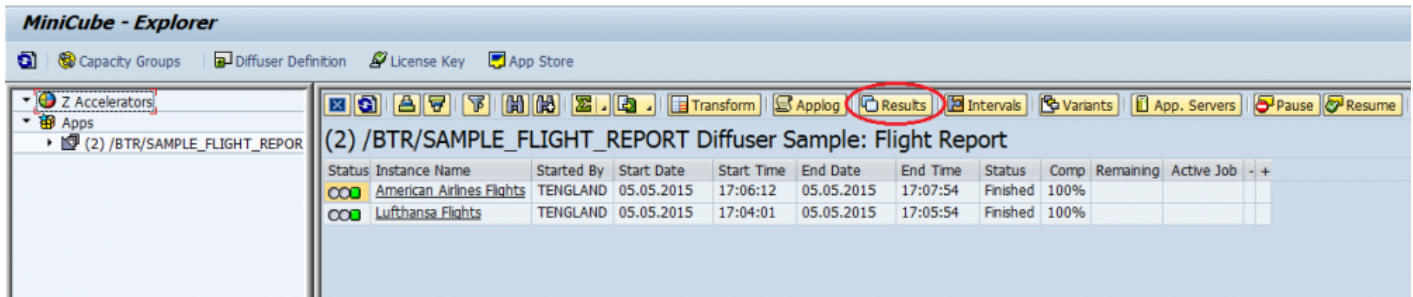


If authorized a number of other functions will be revealed.

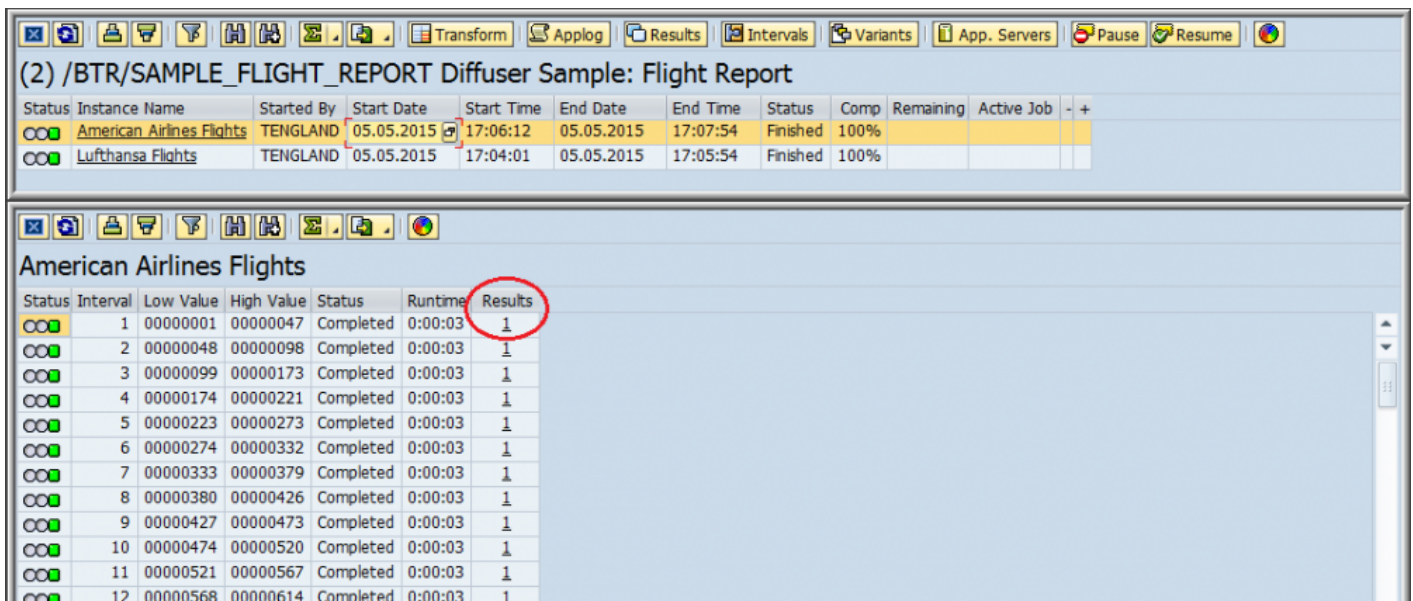


Results

To access the raw results stored against the instance click the results button as below.

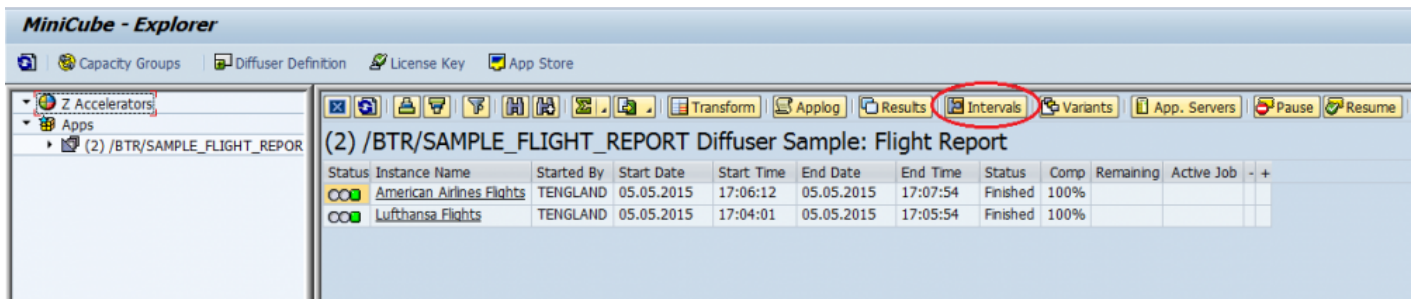


You can also select an interval and view the raw results stored against each interval, by double-clicking the number of results.

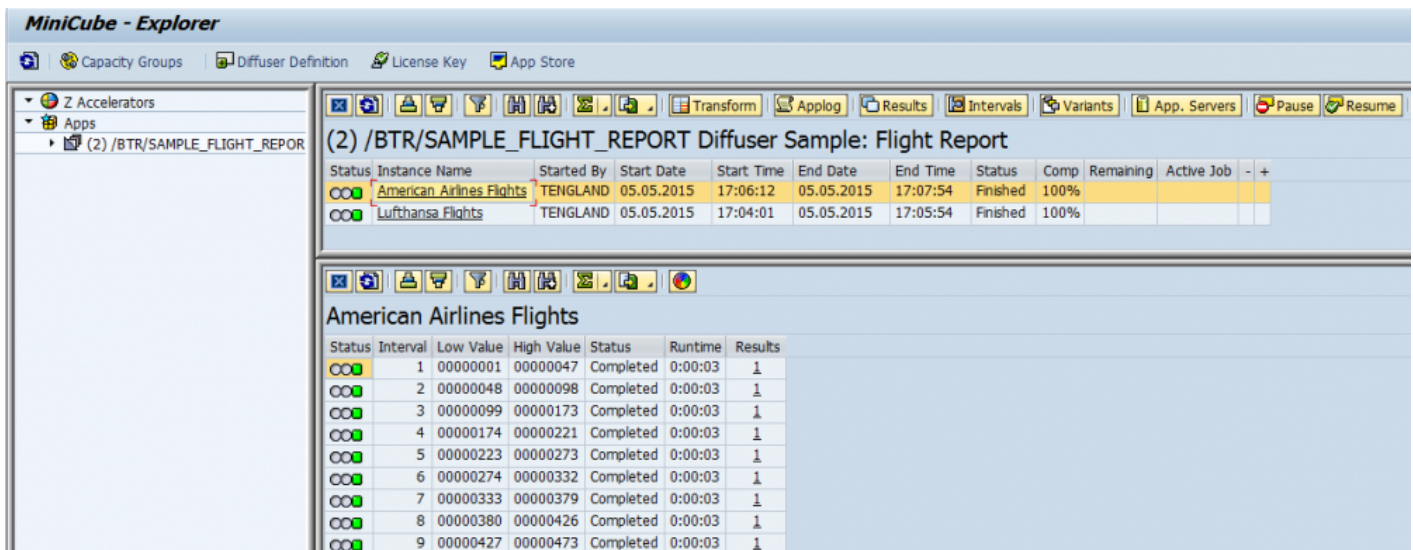


Intervals

By drilling down on the program name the user will access the programs instance runs. Select an instance and in Diffuser mode double click the instance or click “Intervals” to display the intervals to that specific instance run.

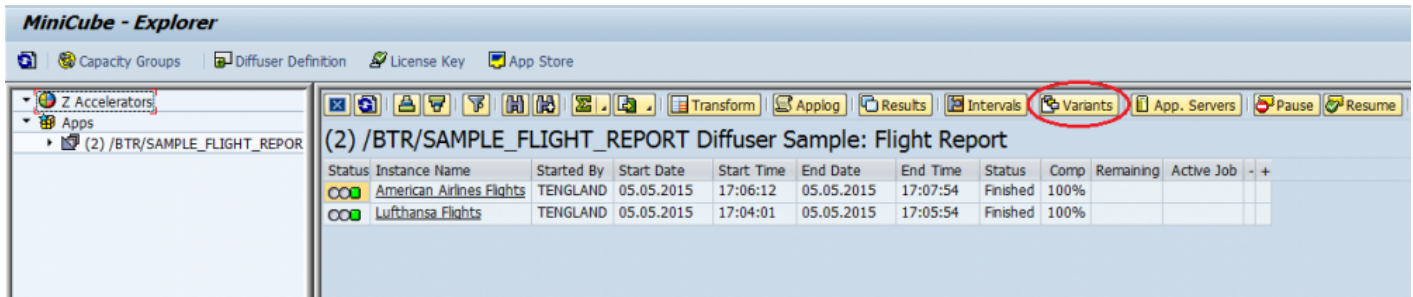


The details of all the intervals are then displayed as below.

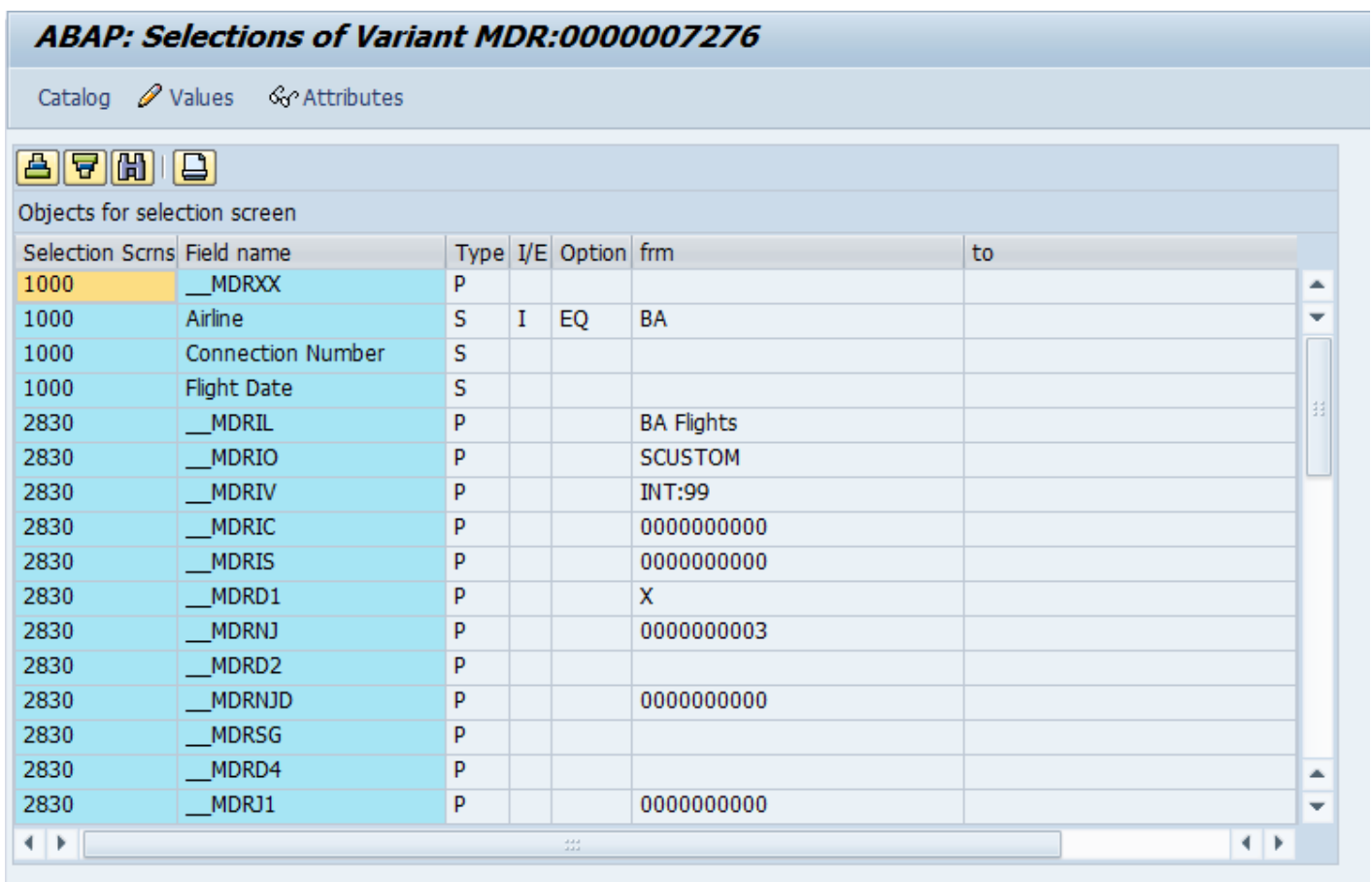


Variants

To access the details entered on the selection screen for an instance click the variant button as below.

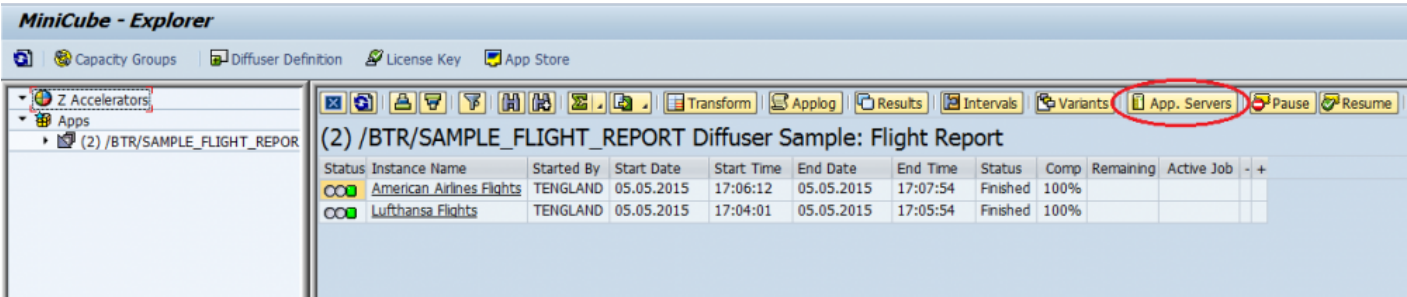


This enables the variant details entered on the selection screen to be viewed.

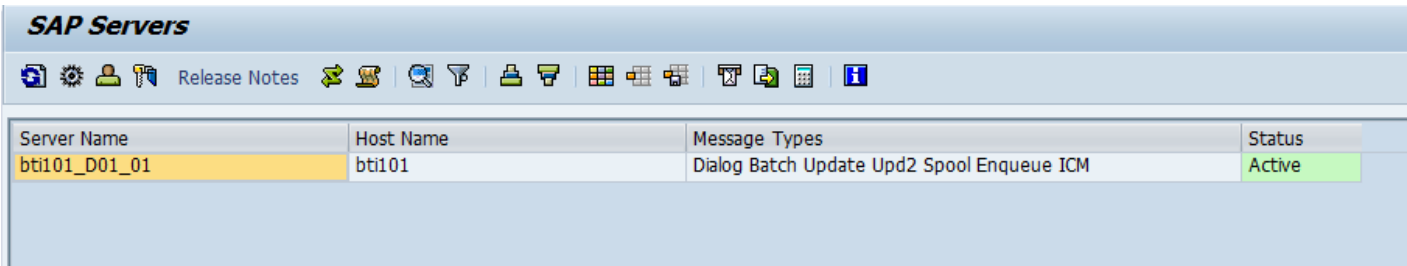


App Servers

To view the application servers click the App Server button as below.



This then displays the available App Servers

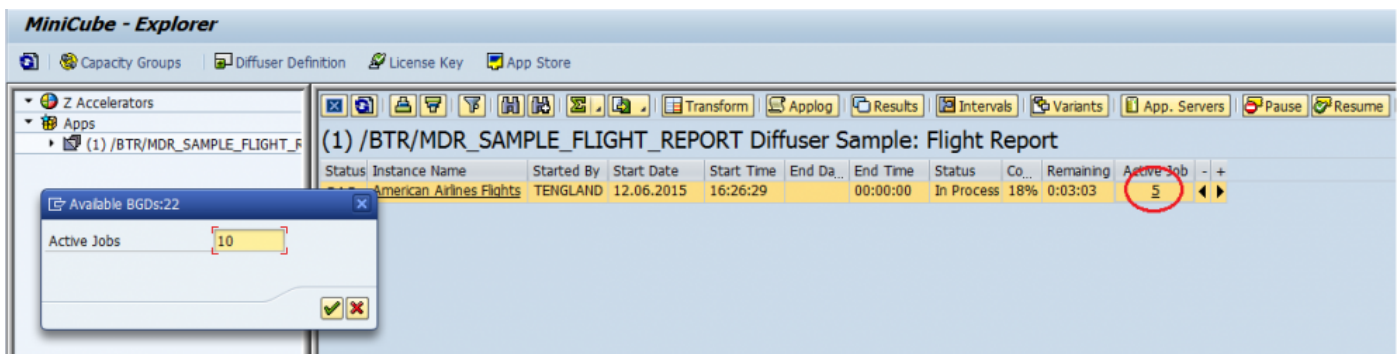


Increase or Decrease Jobs

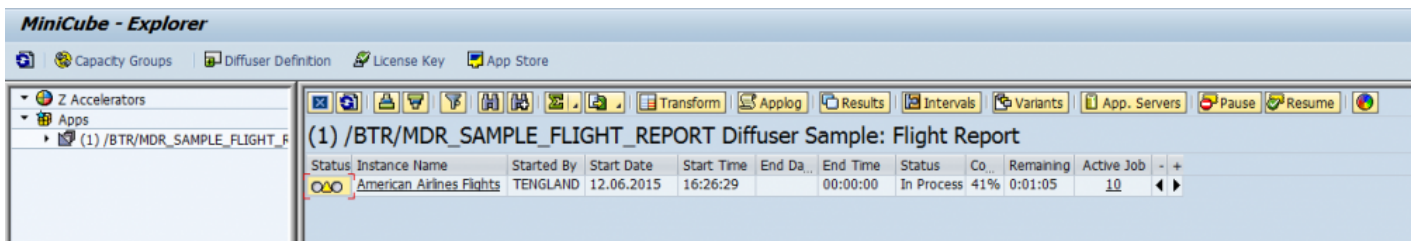
Through the MiniCube transaction, you can see historical instances of a program as well as any currently executing program instances. You can also see the number of active jobs for each program instance currently running, and it is possible to change the number of jobs running for a particular active job.

Adding more jobs can help decrease the run time.

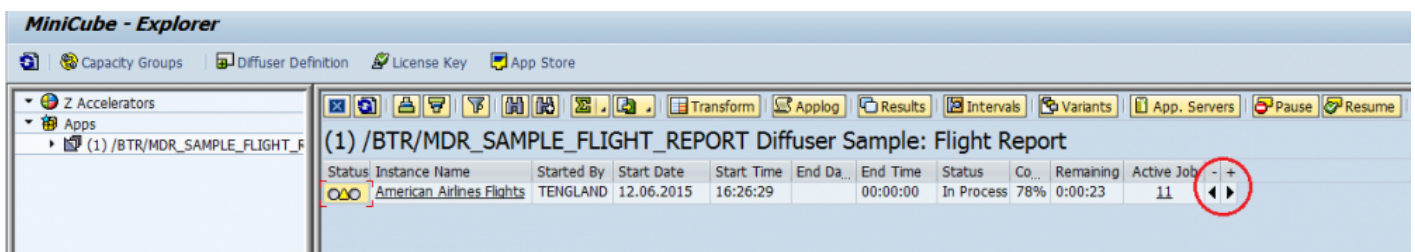
To change the number of jobs click the number of jobs currently running and a popup appears where you can enter the new number of jobs you want the instance to run. Note the top of the popup box shows the number of unused background jobs in the system at that point in time, in this case 22.



Instance is now running 10 jobs.



Alternatively you can click the arrow buttons to increase or decrease the jobs one at a time.



Decrease Jobs

It is also possible to select an instance and decrease the number of jobs by selecting the “Decrease Jobs” option, or selecting the left arrow icon beside the instance. It is only possible to select this option for instances that are currently “In Progress”, and have more than one active job. MDR will prevent you decreasing the number of jobs to zero, if it is the users intention to stop the processing of the Diffuser report, then the user should select the “Stop” option. After a short period, and after selecting refresh the user will notice that the number of active jobs decreases by 1.

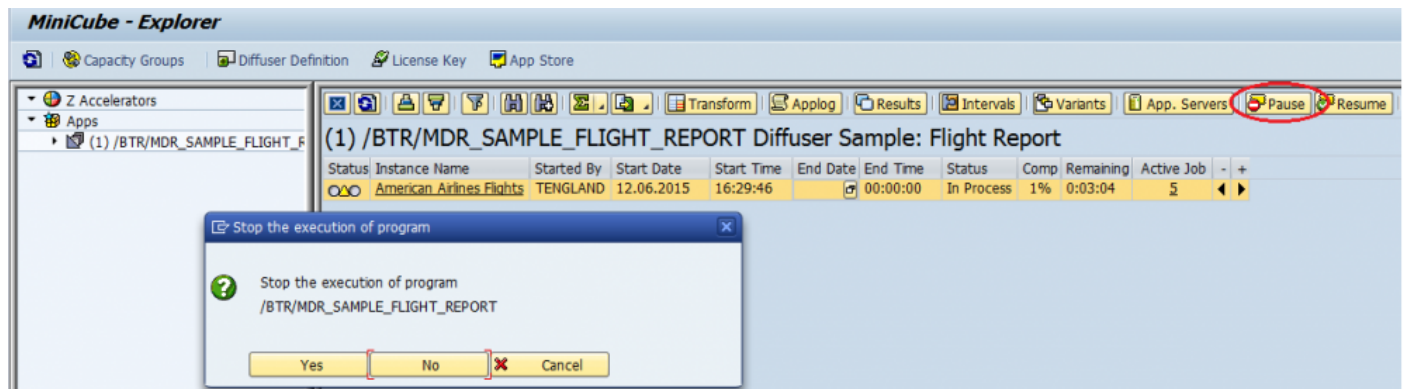
Instance Status	Comp	Remaining	Active Job	-	+
In Progress	87%	00:00:24	2	◀	▶

Pause

It is possible to Pause (or Stop) a program instance using this option. By selecting this option after selecting a program instance, Diffuser tells the currently executing jobs to no-longer process any more intervals after it completes the processing of the current intervals. The status of the Instance, and unprocessed intervals changes to “Stopped”. You will need to click Refresh to update the status. This is a powerful option that is used typically when a Diffuser program needs to be stopped temporarily due to the need to free up batch resources, or stopped permanently if the report run is no longer required. When the instance is paused, the Diffuser framework will not immediately stop all jobs that are currently running. It will instead prevent any new intervals from being started. The more intervals there are the more control over the execution of the instance an administrator will have.

The benefit Diffuser has over the traditional approach to executing reports is that the Diffuser program does not need to start over again, execution can continue from where it left off. The intervals that have already been processed do not need to be reprocessed unless of course it is deemed necessary by the user due to perhaps a substantial amount of time passing before the program is allowed to continue. It is only possible to pause an Instance that is currently in the “In Progress” status.

To pause a program simply select the instance and hit the pause button, you will be asked to confirm that you want to.



Once all the intervals have completed the status of the instance changes as below.

MiniCube - Explorer

Capacity Groups | Diffuser Definition | License Key | App Store

▼ Z Accelerators
▼ Apps
 ▶ (1) /BTR/MDR_SAMPLE_FLIGHT_F

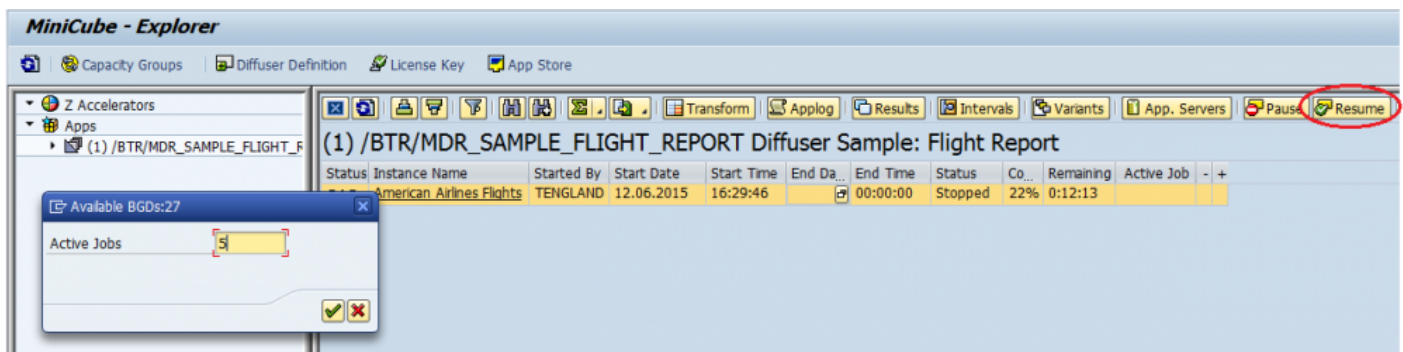
(1) /BTR/MDR_SAMPLE_FLIGHT_REPORT Diffuser Sample: Flight Report

Status	Instance Name	Started By	Start Date	Start Time	End Da..	End Time	Status	Co..	Remaining	Active Job	-	+
○○○	American Airlines Flights	TENGLAND	12.06.2015	16:29:46		00:00:00	Stopped	22%	0:12:13			

Resume

The “Resume” option allows the selected program instance to continue from the point it was stopped or paused. This option uses the Technical Settings of the original program instance to reschedule the report. By resuming an instance it does not reprocess any intervals that have a status of “Completed”, it changes the status of a “Stopped” interval to “Available”. The restart option can only be selected for instances with the status “Paused” or “Error”.

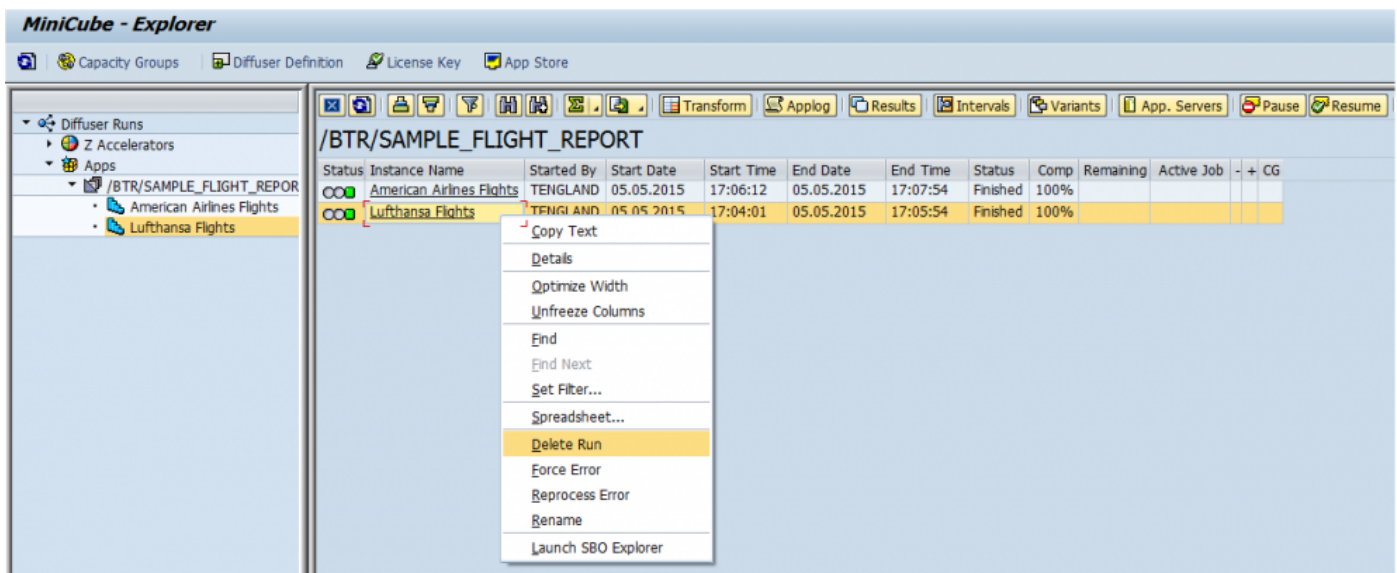
To resume a program select the instance and press the resume button, a pop appears for the number of jobs required to start processing intervals.



Delete

It is possible to delete a program instance by selecting the instance and then the “Delete” option. This in turn deletes all the intervals and results belonging to the program instance. After the delete option is selected the user is faced with a confirmation window to ensure the deletion was intentional. This option is particularly useful in a testing environment and with instances that have errored. It is only possible to delete instances with the status “Error” or “Completed”. The system will not allow an instance “In Progress” to be deleted due to possible data inconsistencies.

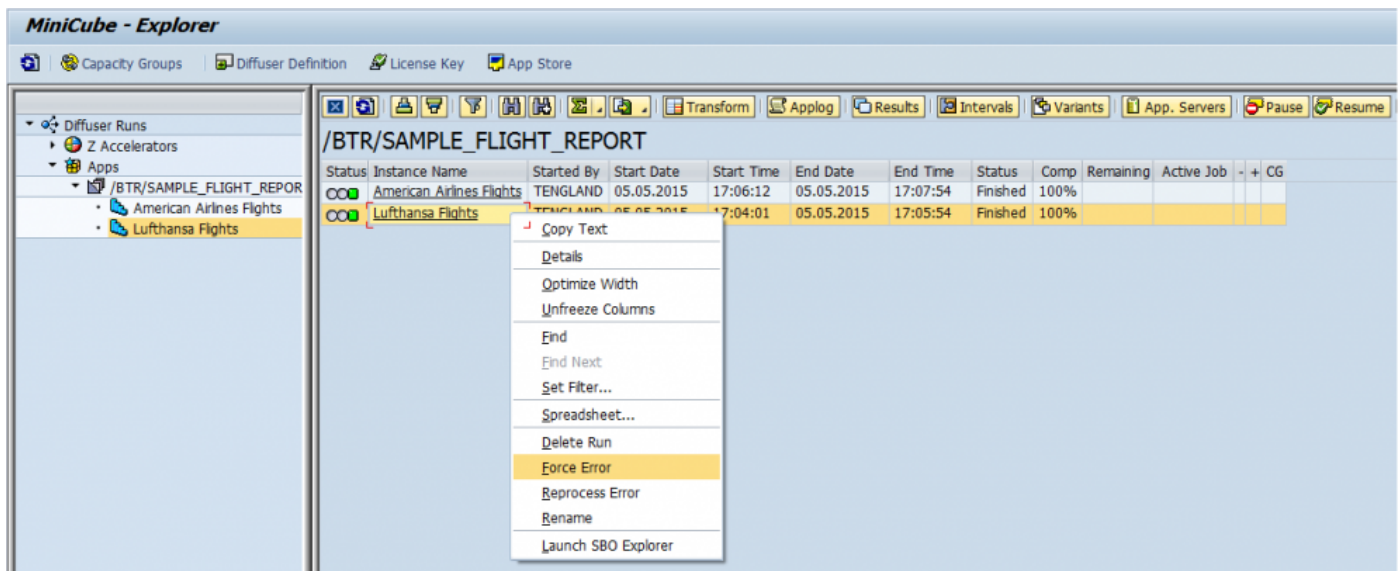
To delete an instance select it right-click and select the Delete option as below.



Force Error

By selecting an Instance, right-clicking and then the “Force error” option, the status of the program instance is changed to “Error”. This allows instances that have technically completed successfully to be changed to Error. This is basically an override function. It is only possible to set an instance to “Error” if there are no active jobs executing the instance.

To delete an instance select it right mouse click and select the force error option as below.



Reprocess Error

On finding an interval in error as below there is an option to reprocess where you have been able to fix the cause of the error, such as updating some master data.



Bear in mind the impact that running the interval out of sequence or at a later date may have on your report or processing of data.

ZERROR_FLIGHT_REPORT											
Status	Instance Name	Started By	Start Date	Start Time	End Date	End Time	Status	Comp	Remaining	Active Job	- + CG
	ZERROR_FLIGHT_REPORT	TENGLAND	12.06.2015	17:33:02		00:00:00	Error	100%			

ZERROR_FLIGHT_REPORT						
Status	Interval	Low Value	High Value	Status	Runtime	Results
	1	00000001	00000047	Completed	0:00:09	1
	2	00000048	00000098	Completed	0:00:10	1
	3	00000099	00000173	Completed	0:00:13	1
	4	00000174	00000221	Completed	0:00:04	1
	5	00000223	00000273	Error	0:00:00	0
	6	00000274	00000332	Completed	0:00:06	1
	7	00000333	00000379	Completed	0:00:04	1
	8	00000380	00000426	Completed	0:00:04	1
	9	00000427	00000473	Completed	0:00:04	1
	10	00000474	00000520	Completed	0:00:03	1
	11	00000521	00000567	Completed	0:00:03	1
	12	00000568	00000614	Completed	0:00:03	1
	13	00000615	00000661	Completed	0:00:04	1
	14	00000662	00000708	Completed	0:00:03	1
	15	00000709	00000755	Completed	0:00:04	1
	16	00000756	00000802	Completed	0:00:03	1
	17	00000803	00000849	Completed	0:00:03	1

To reprocess the error select the instance in the status of error and right-click for the “Reprocess Error” option as below.

The screenshot shows the ZERROR_FLIGHT_REPORT application interface. At the top, there is a toolbar with icons for Transform, Applog, Results, Intervals, Variants, App. Servers, Pause, and Resume. Below the toolbar, a summary bar displays the instance name 'ZERROR_FLIGHT_REPORT', started by 'TENGLAND', start date '12.06.2015', start time '17:33:02', end date '00:00:00', end time '00:00:00', status 'Error', completion '100%', and active job '+ CG'. The main area contains a table with columns: Status, Interval, Low Value, High Value, Status, Runtime, and Results. The table lists 15 intervals. Interval 5 is marked as 'Error' with a red status icon. A context menu is open over the error row, showing options: Copy Text, Details, Optimize Width, Unfreeze Columns, End, End Next, Set Filter..., Spreadsheet..., Delete Run, Force Error, Reprocess Error (highlighted), Rename, and Launch SBO Explorer.

Status	Interval	Low Value	High Value	Status	Runtime	Results
Completed	1	00000001	00000047	Completed	0:00:09	1
Completed	2	00000048	00000098	Completed	0:00:10	1
Completed	3	00000099	00000173	Completed	0:00:13	1
Completed	4	00000174	00000221	Completed	0:00:04	1
Error	5	00000223	00000273	Error	0:00:00	0
Completed	6	00000274	00000332	Completed	0:00:06	1
Completed	7	00000333	00000379	Completed	0:00:04	1
Completed	8	00000380	00000426	Completed	0:00:04	1
Completed	9	00000427	00000473	Completed	0:00:04	1
Completed	10	00000474	00000520	Completed	0:00:03	1
Completed	11	00000521	00000567	Completed	0:00:03	1
Completed	12	00000568	00000614	Completed	0:00:03	1
Completed	13	00000615	00000661	Completed	0:00:04	1
Completed	14	00000662	00000708	Completed	0:00:03	1
Completed	15	00000709	00000755	Completed	0:00:04	1

The same as resuming a Diffuser instance the popup for the number of processors you want to utilize appears.

The screenshot shows the same ZERROR_FLIGHT_REPORT application interface. A dialog box titled 'Available BGDs: 3' is open, showing 'Active Jobs' with a value of '1'. The dialog has a green checkmark button and a red X button. The background table and summary bar are the same as in the previous screenshot.

In this example the error is successfully reprocessed.

The screenshot displays the ZERROR_FLIGHT_REPORT application interface. The top section features a toolbar with icons for file operations and a menu bar with options: Transform, Applog, Results, Intervals, Variants, App. Servers, Pause, and Resume. Below the menu bar is a summary table for the ZERROR_FLIGHT_REPORT instance.

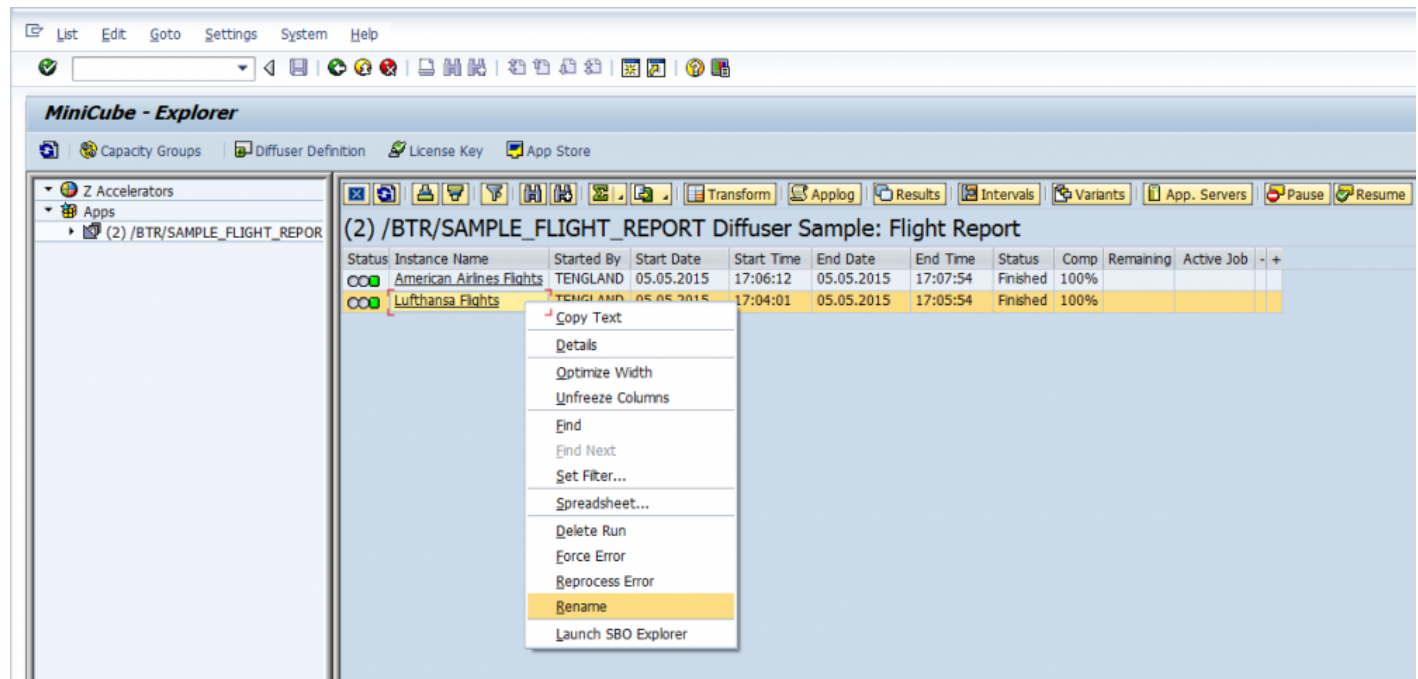
Status	Instance Name	Started By	Start Date	Start Time	End Da...	End Time	Status	Comp	Remaining	Active Job	-	+	CG
	ZERROR_FLIGHT_REPORT	TENGLAND	12.06.2015	17:33:02	12.06.20	17:56:50	Finishe	100%					

The bottom section displays a detailed table of error flight reports, also titled ZERROR_FLIGHT_REPORT. The table has columns for Status, Inter..., Low Value, High, Status, Runtime, and Results. The first row is highlighted with a red box.

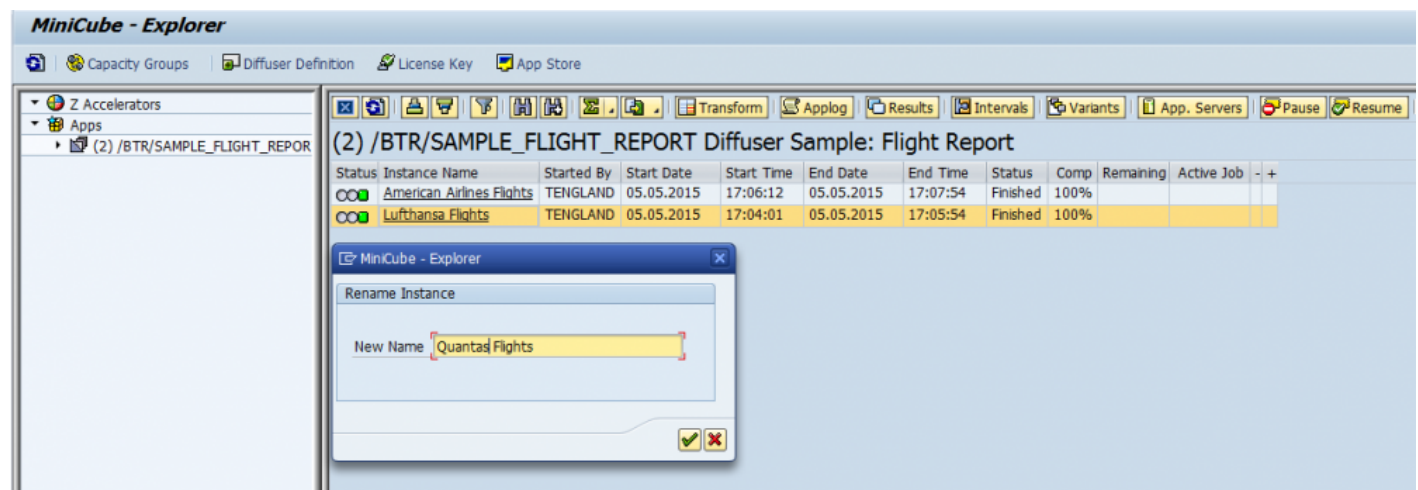
Status	Inter...	Low Value	High	Status	Runtime	Results
	1	00000001	00000047	Completed	0:00:09	1
	2	00000048	00000098	Completed	0:00:10	1
	3	00000099	00000173	Completed	0:00:13	1
	4	00000174	00000221	Completed	0:00:04	1
	5	00000223	00000273	Completed	0:00:03	1
	6	00000274	00000332	Completed	0:00:06	1
	7	00000333	00000379	Completed	0:00:04	1
	8	00000380	00000426	Completed	0:00:04	1
	9	00000427	00000473	Completed	0:00:04	1
	10	00000474	00000520	Completed	0:00:03	1
	11	00000521	00000567	Completed	0:00:03	1
	12	00000568	00000614	Completed	0:00:03	1
	13	00000615	00000661	Completed	0:00:04	1
	14	00000662	00000708	Completed	0:00:03	1

Rename Instance

To rename an instance select the instance and right-click.



Enter the new name.



The new name is updated as below:

MiniCube - Explorer

Capacity Groups | Diffuser Definition | License Key | App Store

Z Accelerators

Apps

(2) /BTR/SAMPLE_FLIGHT_REPOR

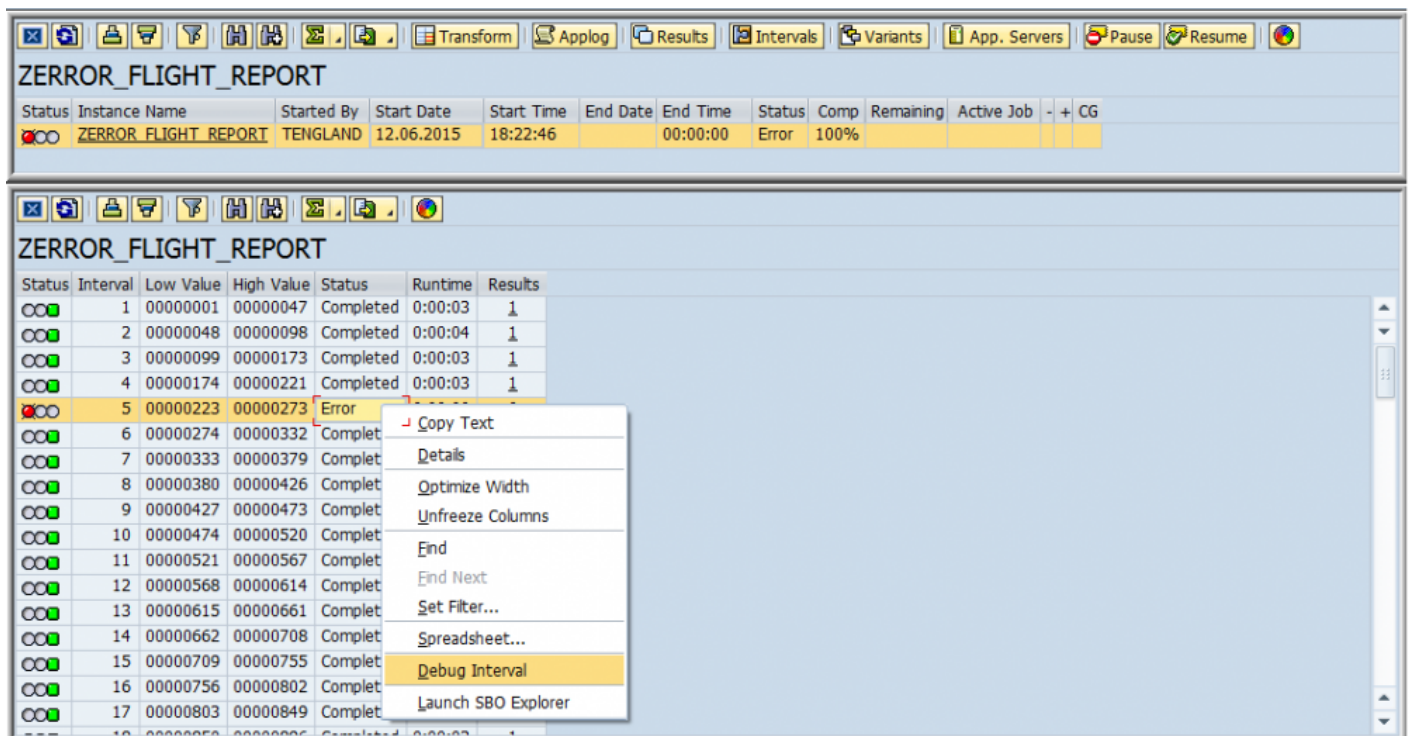
(2) /BTR/SAMPLE_FLIGHT_REPORT Diffuser Sample: Flight Report

Status	Instance Name	Started By	Start Date	Start Time	End Date	End Time	Status	Comp	Remaining	Active Job	-	+
COO	American Airlines Flights	TENGLAND	05.05.2015	17:06:12	05.05.2015	17:07:54	Finished	100%				
COO	Quantas Flights	TENGLAND	05.05.2015	17:04:01	05.05.2015	17:05:54	Finished	100%				

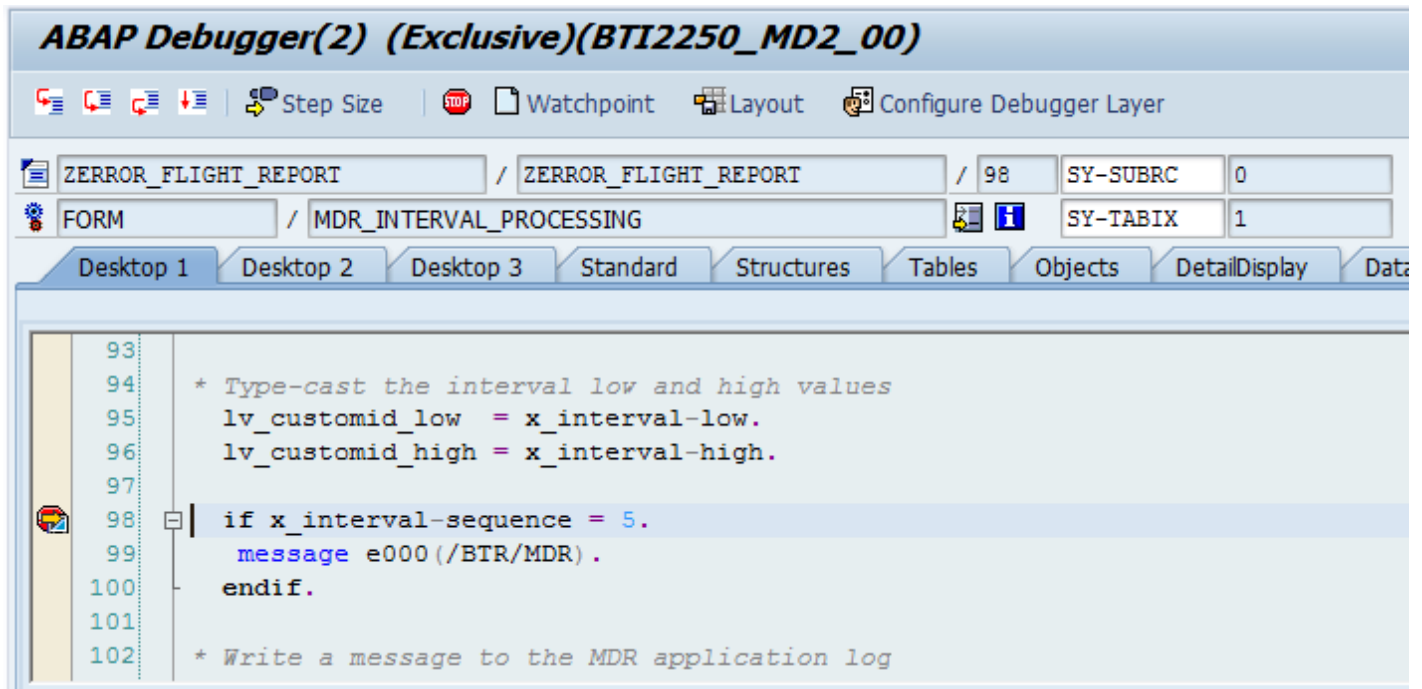
Debug an Interval

On finding an interval in error you also have the option of debugging the interval to try and work out what went wrong.

Firstly ensure you have positioned your break point in the code, then select the interval and right-click for the option to “Debug an Interval”



The debugger will then open at your break point.



Scheduling Diffuser Programs

A Diffuser program can be scheduled just like any other background program. Typically this is done using the standard transaction SM36. The program variants can also be saved as per normal.

Diffuser in most cases, however, does require another program to be scheduled for it to operate efficiently in a production environment. The program function is to regenerate the Interval Variant. The purpose of regenerating an Interval Variant is such that as the master or transactional data grows, the intervals can be recalculated to ensure that each interval is evenly spread. This then ensures the Diffuser program is processed as efficiently as possible.

The program /BTR/MDR_INTERVAL_REGENERATION is used for this purpose. This job should typically be scheduled nightly at the beginning of the batch window, and can be executed for individual Interval Objects, individual Interval Variants, or for all Interval Variants by adjusting the parameters on the selection screen. For the Interval Regeneration to operate, you will need to configure the table /BTR/INTVALVARC. Here you define an Interval Object, Interval Variant and the refresh age. The refresh age defines how frequently the Interval Variant is refreshed. For example if for Object SCUSTOMID, Variant SAMPLE, if the refresh age is 7, the interval variant will only be regenerated every 7 days, even if the regeneration job is scheduled nightly. This functionality allows you to avoid scheduling individual regeneration jobs in different reoccurring cycles. You can override the refresh age functionality by selecting the "Force regeneration" check-box.

Exception Handling and Application Logs

There are numerous situations where a Diffuser program is required to take some action. This action can range from simply outputting informational messages to extreme cases where the program is required to abend. A number of framework components exist in Diffuser so that your program can indicate to the framework the current processing status and the manner in which your program is reacting to a particular situation.

The interval processing subroutine is given a range of objects to process (the interval itself). During the processing of these objects, a situation may occur where a particular object in the range is invalid. An example of this may be that there is a problem with the database integrity of that particular object (e.g. an invalid enumeration or configuration is missing). The Diffuser program has the option to either:

1. Output a message to the application log
2. Skip the current object and move on to the next
3. Abend the program

Alternatively, the program may be required to do a combination of the above. MDR provides the developer with a number of statements to log exceptions:

The sample code below shows how to code messages raised inside MDR to be read via the application log.

```
FORM mdr_interval_processing
  USING x_interval TYPE /btr/st_interval_values.
  ...

  IF lv_error EQ gc_true.
    mdr_msg_put 1 'E' '100' 'MSGCLASS' space space space space.
  ENDIF.
  ...

ENDFORM.
```

The statement `mdr_msg_put` will output a message to the application log. It expects to be provided with the following parameters:

1. Priority – The log level represents the importance of the message. This allows the person running the program to determine what messages are outputted to the application log. The five categories that can be used are:
 - Critical (1) – Used to represent that the message is critical
 - Important (2) – Used to represent that the message is important

Normal (3) – Used to represent that the message is of normal importance

Information (4) – Used to represent that the message is an informational message

Debugging (5) – Used to represent that the message is for debugging purposes

2. Message Type – The message type is the standard SAP message type and represents the type of message. It must be one of the following five options:

Error (E) – Error message

Success (S) – Success message

Information (I) – Informational message

Warning (W) – Warning message

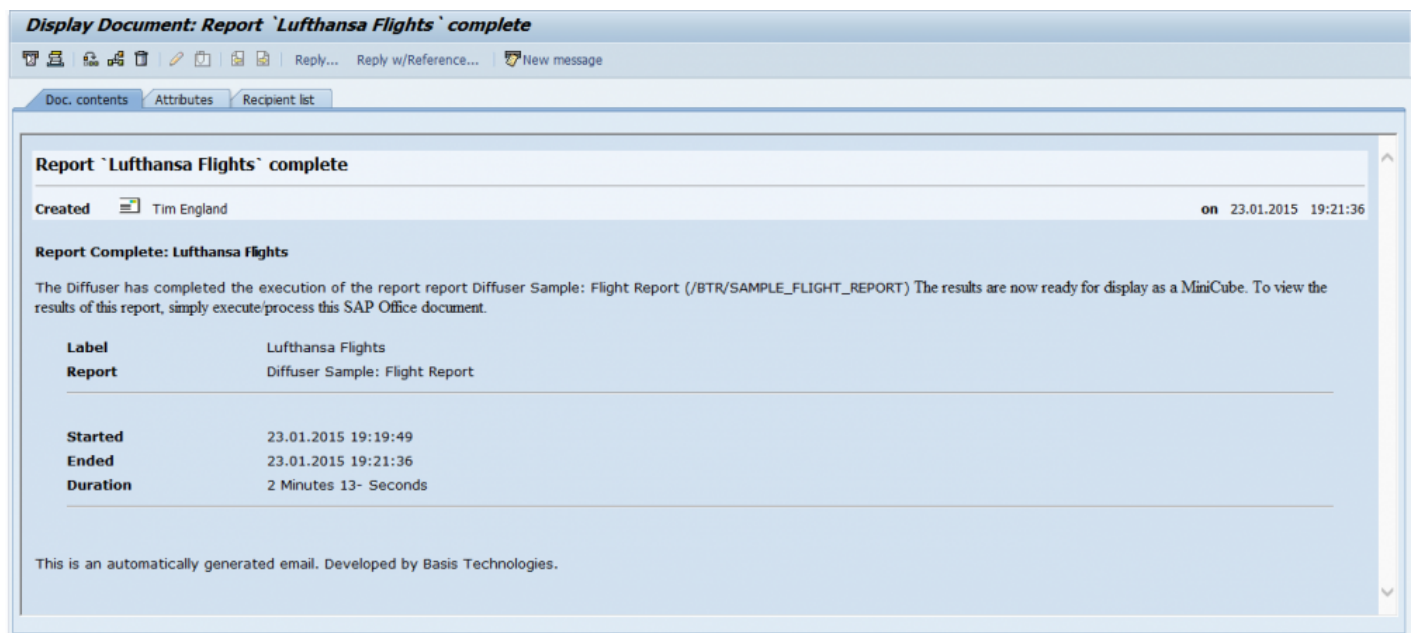
Abend (A) – Abend message – stops the program

3. Message Number – The message number is the standard SAP message number as defined in the message class.
4. Message Class – The message class represents the standard SAP collection of messages.
5. Message variables – When you define a message in the standard SAP message class, placeholders for variables may be defined. These placeholder variables can be passed in here. There may be up to four message variables. If the message has no variables or there are less than four, then you should use the “space” keyword – i.e. you must provide all four variables even if you simply use the space keyword four times.

Email Notification

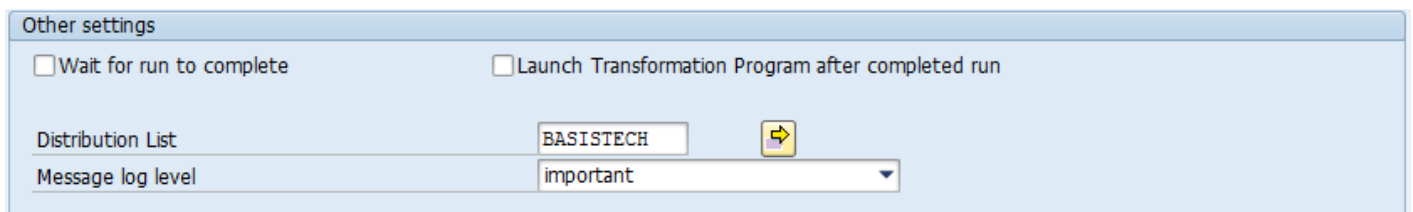
On the technical settings tab of a Diffuser program there is a parameter “Distribution List”.

Diffuser provides the functionality to automatically send a notification to users SAP inbox as defined on the Distribution List once the report results are complete. This improves the flow of information by ensuring that results are passed onto the business users at the soonest possible moment. It is no longer necessary to rely on business processes to notify and pass on a reports output to the business, instead a link to the report output is sent automatically to the users inbox. The results can be accessed by this link.



Providing SAP Connect is configured in your system, these results can also be sent directly to the corresponding users external email address. The link provides direct access to the report output, even if the user is not initially logged into your SAP system.

This configuration is extremely simple. All it involves is configuring the appropriate users on the Technical Settings screen, under Distribution List.



For consistency, Diffuser provides the ability to send email notification of results of reports not migrated to Diffuser format. Diffuser includes a program used to monitor background reports that have been scheduled and determine whether a notification is to be sent to a user or a group of users indicating the completion of the reports.

The notification is delivered in the same manner as Diffuser delivers the notification. This has the added advantage that a link is sent to a user rather than the actual spool attachment (which may be very large).

Furthermore, it is possible to configure the email template used for the delivery of the notification.

In order to deliver notifications for standard/custom-built non-Diffuser reports and programs, the following is required:

1. Initially, the configuration for the notification must be maintained in the view cluster (/BTR/CREPNTF) to determine whom the notification should be sent to and with what template. This is done via transaction SM34. The job name is critical in the configuration and will be used to determine if an entry has been executed in the job schedule. The job (when configured to run in background) must have the same name as entered in the configuration. If it doesn't have the same name – it will not be picked up and hence no notifications will be sent.

For each job you setup, you must also specify who will be notified when the report is finished.

Recipients are classified in various categories including:

- a. Internal SAP Username
- b. External Address
- c. Organizational Unit (As defined in the HR Org Structure)
- d. Internet Address

You can configure one or more of the above for each job setup. For each recipient, you may also specify the template that is used when the notification is sent. This can be defaulted to /BTR/MDR_REPORT_NOTIFICATION. If you want to customize the notification template at an individual level, this can also be specified against the recipient. The recipient level template will always take precedence over what has been specified at the job level.

2. During the normal batch schedule, the reports that have configured in the first step are executed as per normal (in background).
3. The report /BTR/MDR_REPORT_NOTIFICATION must then be scheduled daily at the end of the batch schedule. This will find all background jobs that have been executed (restricted on selection criteria if required) and send the appropriate notification to the configured user(s).

Notifications are generally delivered to a user's SAP Inbox. The document that is sent contains the particulars about when the report was started, when it completed and how long it took. It also specifies the report title and technical name.

The inbox item must be "Executed" by the user in order for them to be taken to the report output. This is a simple matter of right-clicking on the Inbox item and selecting "Execute". The user will then be taken to the spool output immediately.

Debugging and Troubleshooting

- [Debugging Programs](#)
- [Troubleshooting](#)

Debugging Programs

One of the major benefits provided is that the developer is able to debug their program in much the same way as they would when they develop a normal program.

When first executing a program with Diffuser at the top of the selection screen is the “Technical Settings” button, the option “Run online as a single process (debugging mode)”, allows the program to be run online. This means that any break points that the developer has in their code, either hard-coded or dynamically set from the editor, will be stopped at.

In addition on finding an interval in error you also have the option of debugging the interval to try and work out what went wrong see [“Debug an Interval”](#) for details.

The screenshot shows the 'Diffuser Sample: Flight Report' window. It contains several sections for configuring the program run:

- Instance Settings:** A text field labeled 'Label' with the value 'Lufthansa Flights'.
- Interval Settings:** Two dropdown menus. The first is 'Perform processing using intervals of' with the value 'Sample: Flight Customers'. The second is 'Interval variant' with the value 'INT:100 : 99 intervals'.
- Distribution:** Three radio buttons: 'Number of batch jobs across all servers' (selected), 'Distribution according to server group', and 'Manual Distribution'. Below the radio buttons are two empty text fields. At the bottom of this section is a radio button labeled 'Run online as a single process (debugging mode)' which is also selected.
- Other settings:** Two checkboxes: 'Wait for run to complete' and 'Launch Transformation Program after completed run'. Below these are two more fields: 'Distribution List' with an empty text field and a refresh icon, and 'Message log level' with a dropdown menu showing 'Other'.

The bottom right corner of the window has a toolbar with icons for a clock, a checkmark, a document, and a close button, along with the text 'Check'.

Troubleshooting

When converting an existing program to use the Diffuser framework the obvious way to help find problems is to run the original program and MDR program side by side to check the results, to remove the multiple processing and intervals as possible issues keep the Diffuser program you can run it with just one large interval.

Typical bugs introduced from converting a program to use Diffuser are:

- Global variables not being cleared at the end of the interval processing subroutine, if you have no problem with one large interval the problem will most likely be around clearing global variables
- Repeating code in interval processing that is only required once per background job
- Variables not being stored for the transformation program, don't forget anything that needs to be displayed in the results needs to be stored into the results and retrieved by the transformation program

When seeing Intervals with an error status you should look for short dumps via transaction ST22 and messages against the background job as this should have also ended in failure. You can use the [Debug an Interval](#) option to rerun the interval via a dialog process to help you find the error.

Advanced Concepts

The details of advanced programming concepts are available here.

- [Dynamic Interval Generation](#)
- [Authority Checks](#)
- [Application Program Interface](#)

Dynamic Interval Generation

In some situations it may be necessary that you want the Intervals to dynamically change each time you execute a Diffuser program. This is an alternate method of Interval Generation to the more commonly used Interval Object method. Diffuser uses the concept of an Interval Generator to do this. An Interval Generator allows you to write ABAP code to do this dynamic interval creation. A custom Interval Generator can be implemented using the subroutine `mdr_interval_generator` in the Main program of the MDR program.

The impact on the selection screen is that the interval size and interval count options are displayed on the “Technical Setting” screen, see the [Technical Settings](#) section for details on this.

The subroutine `mdr_interval_generator` is called at the beginning of the Diffuser program. This subroutine is used to define the intervals that will be processed. The subroutine provides a single input structure that contains both an interval count and an interval size. These are values that are set at run-time, and provide information to the program on how it should break up the processing. The `CHANGING` parameter is a list of Intervals to be processed. The developer writing the Diffuser program needs to implement logic to populate the `LOW` and `HIGH` interval values of `yt_intervals`.

```
*-----*
* FORM mdr_interval_generation
*-----*
* This form is called by Diffuser to generate      *
* interval ranges that can be coded as per your own requirements *
*-----*
FORM mdr_interval_generation
  USING      x_input      TYPE /btr/st_intgen_input
  CHANGING yt_intervals  TYPE /btr/tt_mdr_interval_values.

DATA :
  lv_interval      LIKE LINE OF yt_intervals.

DATA:
  lv_interval_size      TYPE i,
  lv_interval_index     TYPE i,
  lv_interval_index_1   TYPE i,
  lv_remainder          TYPE i,
  lv_count              TYPE i,
  lv_count_numc         TYPE numc10,
  lv_count_intervals    TYPE i,
  lv_index              TYPE syindex,
  lv_low_index          TYPE syindex,
  lv_flag              TYPE c.
```

```
DATA: lt_sbook TYPE sbook,
      ls_sbook TYPE sbook.

* Select bookings
SELECT *
  FROM sbook
  INTO TABLE lt_sbook
 WHERE custid in s_custid

SORT lt_sbook.

DELETE ADJACENT DUPLICATES FROM lt_sbook.

DESCRIBE TABLE lt_sbook LINES lv_count.

IF x_input-interval_size IS INITIAL.

*   Determine the size of each interval
    lv_interval_size = lv_count DIV x_input-interval_count.
    lv_remainder      = lv_count MOD x_input-interval_count.

    IF NOT lv_remainder IS INITIAL.

        ADD 1 TO lv_interval_size.
    ENDIF.

ELSE.

    lv_interval_size = x_input-interval_size.

ENDIF.

lv_interval_index = lv_interval_size.

* Find the first low for the interval
CLEAR ls_sbook.
READ TABLE lt_sbook INTO ls_sbook INDEX 1.

lv_interval-low = ls_sbook-custid.

DO.

* Get the last number in this package of data
  CLEAR ls_sbook.
  READ TABLE lt_sbook INTO ls_sbook INDEX lv_interval_index.

  IF sy-subrc EQ 0.

*   Provided something was found store this in the high
```

```
lv_interval-high = ls_sbook-custid.

APPEND lv_interval TO yt_intervals.
CLEAR lv_interval.

* Add one to the interval index to find the new low for the next interval
lv_interval_index_1 = lv_interval_index + 1.

CLEAR ls_sbook.
READ TABLE lt_sbook INTO ls_sbook INDEX lv_interval_index_1.
IF sy-subrc = 0.
*   If a record is found we have a new interval to create so create the low
*   else interval creation is complete so exit
    lv_interval-low = ls_sbook-custid.
ELSE.
    EXIT.
ENDIF.
lv_interval_index = lv_interval_index + lv_interval_size.

ELSE.

*   Fill the last entry of the intervals with the last entry
    CLEAR ls_sbook.
    READ TABLE lt_sbook INTO ls_sbook INDEX lv_count.

    IF ls_sbook-custid >= lv_interval-low .
        lv_interval-high = ls_sbook-custid.
        APPEND lv_interval TO yt_intervals.
    ENDIF.
    EXIT.

ENDIF.

ENDDO.

ENDFORM.
```

Authority Checks

Authorizations are now delivered with Diffuser out of the box, see [Authorisations](#).

If you want to introduce more advanced security into Diffuser the following steps show how you can enhance the controls.

- [Implementation](#)
- [Technical Settings](#)
- [Expert Mode](#)
- [Individual Actions](#)

Authorisations

Out of the box authorisations are now supplied with Diffuser.

There is an expert and user role supplied further details are below.

Expert role /BTR/DIF:EXPERT

This involves access to the following transactions:

/BTR/MINICUBE

/BTR/DIFFUSER

/BTR/LICENSE

/BTR/MDR

/BTR/MDRH

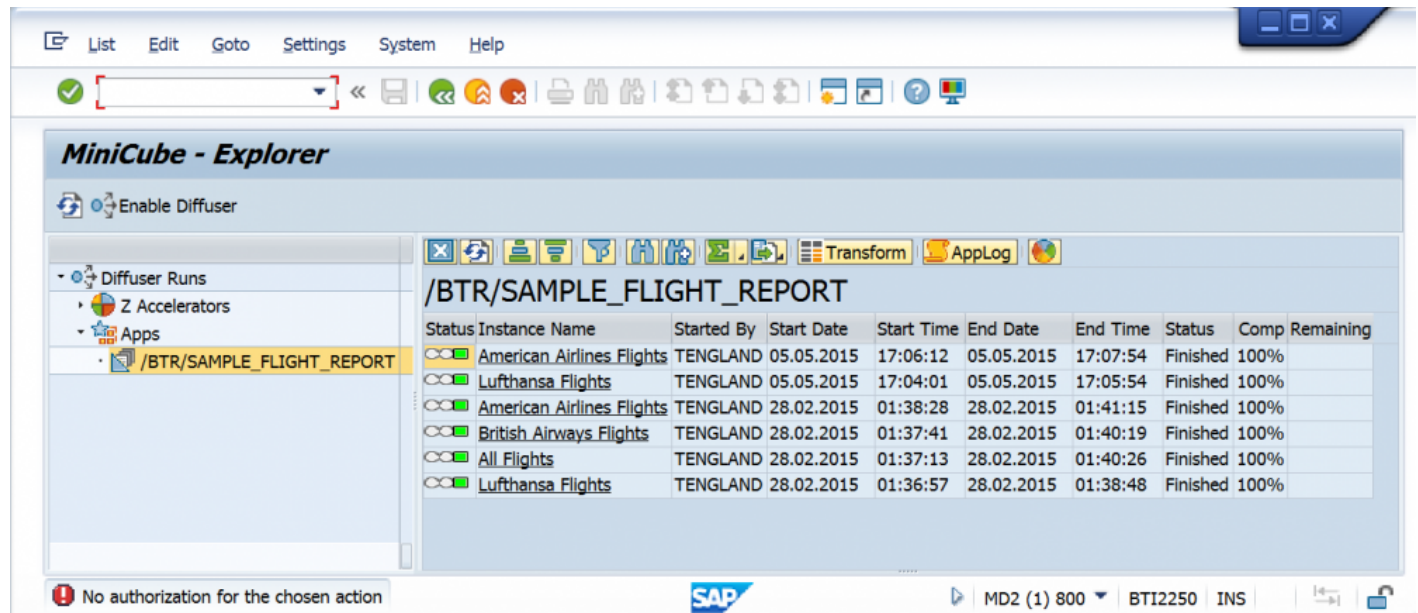
/BTR/MDRH_OLD



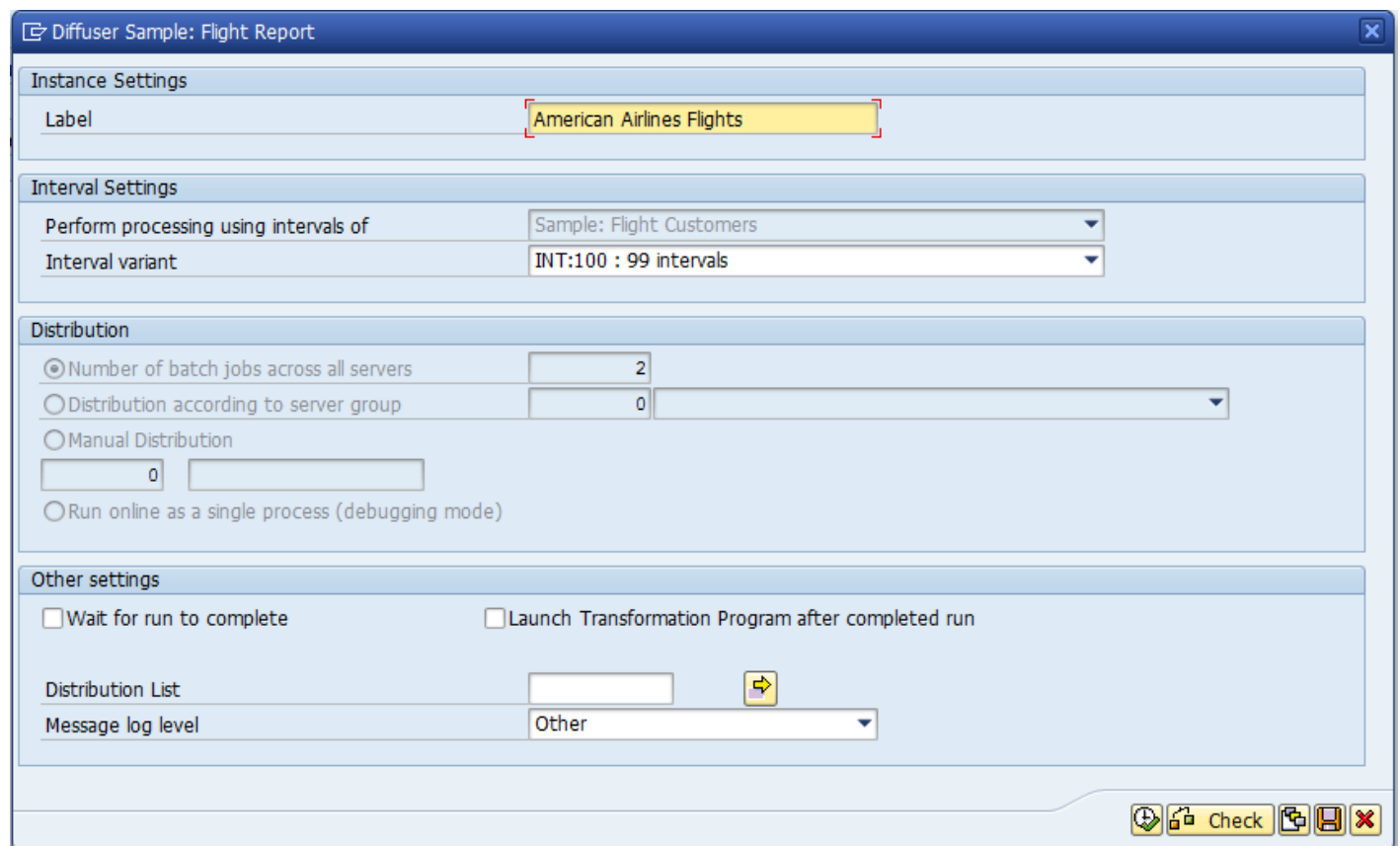
Note this role also includes the authorisation object for Background Processing: Background Administrator (S_BTCH_ADM) to allow the expert user to add and remove background processing jobs to a Diffuser run.

User role /BTR/DIF:USER

This involves access to the MiniCube transaction /BTR/MINICUBE with [Diffuser Mode](#) giving an error message as below.



The distribution component of [technical settings](#) is disabled so the user cannot determine the number of jobs that a program uses when running with Diffuser, as below.

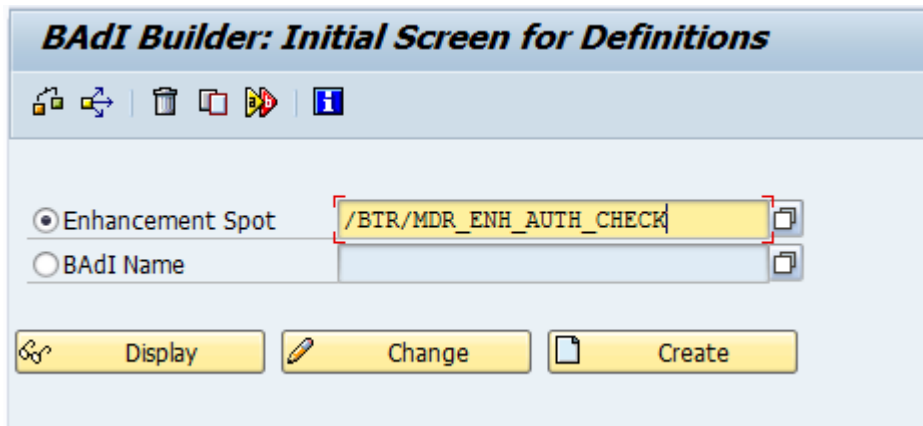


Diffuser also provides enhancement spots to allow developers to apply customer specific authority checks. This can be used to restrict technical as well as administrative settings at a user and program level.

For more information refer to the section [Authority Checks](#) in the Z Accelerators Guide.

Implementation

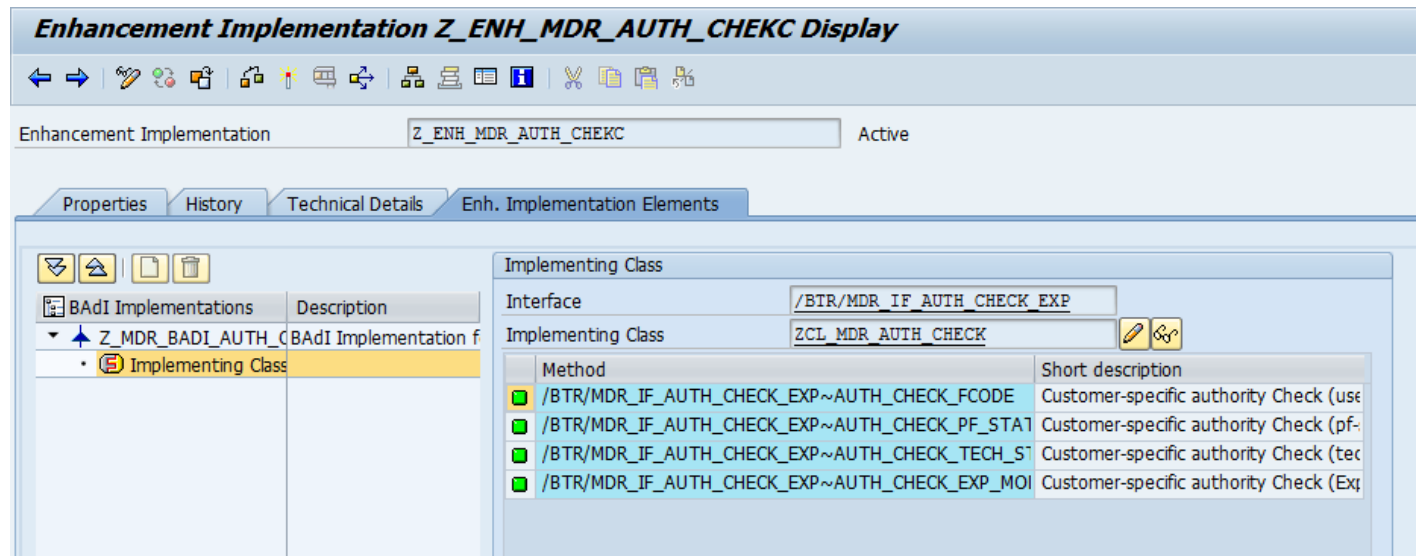
Diffuser contains enhancement spot /BTR/MDR_ENH_AUTH_CHECK. Go to SAP standard transaction SE18 to access it.



This enhancement spot contains BAdI definition /BTR/MDR_BADI_AUTH_CHECK_EXP which can be implemented to restrict access to both the Expert Mode functionality and the Technical Settings of Diffuser programs.

To implement BAdI definition /BTR/MDR_BADI_AUTH_CHECK_EXP create a custom class using interface /BTR/MDR_IF_AUTH_CHECK_EXP in SE18 (More information on how to implement a BAdI can be found in SAP standard documentation).

Once your class implementation of BAdI definition /BTR/MDR_BADI_AUTH_CHECK_EXP is in place, implement interface method AUTH_CHECK_TECH_STTGS.



The example above uses custom implementing class ZCL_MDR_AUTH_CHECK. Click on the method name to drill into its implementation and add your custom code. The signature of this method imports the program ID so you can set restrictions at program level.

The following sections will demonstrate how to implement controls around Technical Settings, Expert Mode and Individual Actions.

Technical Settings

There are two options to control the authorizations for Technical Settings.

When the changing parameter LCK_EXPT_MODE is set to true only the job distribution section of the Technical Settings screen is grayed out (see below).

Class Builder: Class ZCL_MDR_AUTH_CHECK Display

← → | | Pattern | Pretty Printer | Signature

Ty.	Parameter	Type spec.	Description
▶	PROGRAMOID	TYPE /BTR/OID OPTIONAL	MDR: Internal Object ID
▶▶	LCK_EXPT_MODE	TYPE /BTR/LCKEXMD OPTIONAL	MDR: Lock Expert Mode
▶▶	LCK_TECH_SETT	TYPE /BTR/LCKTSETT OPTIONAL	MDR: Lock Technical Settings

Method: /BTR/MDR_IF_AUTH_CHECK_EXP~AUTH_CHECK_TECH_STTGS Active

```

1  METHOD /btr/mdr_if_auth_check_exp~auth_check_tech_sttgs.
2
3      AUTHORITY-CHECK OBJECT 'S_TCH_S'
4                          ID 'PROID' FIELD programoid
5                          ID 'ACTVT' FIELD '03'.
6
7  IF sy-subrc <> 0.
8      lck_expt_mode = 'X'.
9  ENDIF.
10
11 ENDMETHOD.

```

Technical Settings will look like this:

The screenshot shows a software window titled "Diffuser Sample: Flight Report". It contains several configuration sections:

- Instance Settings:** A "Label" field with the text "American Airlines Flights".
- Interval Settings:** Two dropdown menus. The first is "Perform processing using intervals of" with the value "Sample: Flight Customers". The second is "Interval variant" with the value "INT:100 : 99 intervals".
- Distribution:** Radio buttons for "Number of batch jobs across all servers" (selected, value 2), "Distribution according to server group" (value 0), and "Manual Distribution" (value 0). There is also an option "Run online as a single process (debugging mode)".
- Other settings:** Two checkboxes: "Wait for run to complete" and "Launch Transformation Program after completed run". Below these are a "Distribution List" field with a button and a "Message log level" dropdown set to "Other".

At the bottom right, there is a toolbar with icons for a clock, a lock, a "Check" button, and standard save/cancel/delete icons.

When the changing parameter LCK_TECH_SETT is set to true all input fields of the Technical Settings screen is grayed out (see below).

Class Builder: Class ZCL_MDR_AUTH_CHECK Display

← → | Pattern Pretty Printer Signature

Ty.	Parameter	Type spec.	Description
▶	PROGRAMOID	TYPE /BTR/OID OPTIONAL	MDR: Internal Object ID
▶▶	LCK_EXPT_MODE	TYPE /BTR/LCKEXMD OPTIONAL	MDR: Lock Expert Mode
▶▶	LCK_TECH_SETT	TYPE /BTR/LCKTSETT OPTIONAL	MDR: Lock Technical Settings

Method Active

```

1  METHOD /btr/mdr_if_auth_check_exp~auth_check_tech_stgs.
2
3      AUTHORITY-CHECK OBJECT 'S_TCH_S'
4          ID 'PROID' FIELD programoid
5          ID 'ACTVT' FIELD '03'.
6
7  IF sy-subrc <> 0.
8      lck_tech_sett = 'X'.
9  ENDIF.
10
11 ENDMETHOD.

```

Technical Settings will look like this:

Diffuser Sample: Flight Report

Instance Settings

Label American Airlines Flights

Interval Settings

Perform processing using intervals of Sample: Flight Customers

Interval variant INT:100 : 99 intervals

Distribution

☒ Number of batch jobs across all servers 2

☐ Distribution according to server group 0

☐ Manual Distribution

0

☐ Run online as a single process (debugging mode)

Other settings

☐ Wait for run to complete ☐ Launch Transformation Program after completed run

Distribution List


Message log level Other

Check

Diffuser Mode

Access to the Diffuser Mode button on the MiniCube transaction /N/BTR/MINICUBE is now restricted by default see [Authorisations](#) topic for details.

To hide the Diffuser Mode button use BAdI implementing class method AUTH_CHECK_PF_STATUS. An example can be seen below which shows how to hide the Diffuser Mode button as well as links to the Diffuser Definition, Capacity Groups and License Key buttons if you require any of these not to be shown.

Ty.	Parameter	Type spec.	Description
	value(XYT_EXCLUDING)	TYPE /BTR/SLIS_EXTAB OPTIONAL	Table with excluded fcodes

Method	/BTR/MDR_IF_AUTH_CHECK_EXP~AUTH_CHECK_PF_STATUS	Active
--------	---	--------

```

1  □ METHOD /btr/mdr_if_auth_check_exp~auth_check_pf_status.
2    * Note: PF status of minicube can be changed in here
3
4    INCLUDE /btr/mdr_fcodes.
5
6    * Remove Expert Mode Button
7    APPEND co_fcode_expert_mode TO xyt_excluding.
8    * Remove Diffuser Definition
9    APPEND /btr/cl_mdr_constants=>co_fcode_progdeff TO xyt_excluding.
10   * Remove License Keys
11   APPEND /btr/cl_mdr_constants=>co_fcode_mdrlicense TO xyt_excluding.
12   * Remove Capacity Groups
13   APPEND /btr/cl_mdr_constants=>co_fcode_capacity_groups TO xyt_excluding.
14
15  ENDMETHOD.
```

The Diffuser Mode button will then not be shown as below:

MiniCube - Explorer

Diffuser Runs

- Z Accelerators
- Apps
 - /BTR/SAMPLE_FLIGHT_REPORT

/BTR/SAMPLE_FLIGHT_REPORT

Status	Instance Name	Started By	Start Date	Start Time	End Date	End Time	Status	Comp	Remaining
	American Airlines Flights	TENGLAND	05.05.2015	17:06:12	05.05.2015	17:07:54	Finished	100%	
	Lufthansa Flights	TENGLAND	05.05.2015	17:04:01	05.05.2015	17:05:54	Finished	100%	
	American Airlines Flights	TENGLAND	28.02.2015	01:38:28	28.02.2015	01:41:15	Finished	100%	
	British Airways Flights	TENGLAND	28.02.2015	01:37:41	28.02.2015	01:40:19	Finished	100%	
	All Flights	TENGLAND	28.02.2015	01:37:13	28.02.2015	01:40:26	Finished	100%	
	Lufthansa Flights	TENGLAND	28.02.2015	01:36:57	28.02.2015	01:38:48	Finished	100%	

Individual Actions

BAdI definition /BTR/MDR_BADI_AUTH_CHECK_EXP offers the ability to restrict access to individual functions. For instance, actions like deleting an instance run, pausing or increasing number of batch jobs can be restricted at program level.

To restrict by individual buttons the technical codes are as below:

Button	Constant
Application Log	/BTR/CL_MDR_CONSTANTS-CO_FUNC_APPLOG
Results	/BTR/CL_MDR_CONSTANTS-CO_FUNC_RSET
Intervals	/BTR/CL_MDR_CONSTANTS-CO_FUNC_IVLS
Variants	/BTR/CL_MDR_CONSTANTS-CO_FUNC_VRNTS
Resume	/BTR/CL_MDR_CONSTANTS-CO_FUNC_RESUME
Stop	/BTR/CL_MDR_CONSTANTS-CO_FUNC_STOP
Application Servers	/BTR/CL_MDR_CONSTANTS-CO_FUNC_APPSVRS

BAdI implementing class method AUTH_CHECK_FCODE. An example of the code can be seen below:

Ty.	Parameter	Type spec.	Description
	EXPERT_MODE	TYPE CHAR1 OPTIONAL	Single-Character Indicator
	FCODE	TYPE SYUCOMM OPTIONAL	Function code that PAI triggered
	value(XYT_BUTTONS)	TYPE TTB_BUTTON OPTIONAL	Toolbar Buttons

Method	/BTR/MDR_IF_AUTH_CHECK_EXP~AUTH_CHECK_ALV_TOOL_BAR	Active
--------	--	--------

```

1  METHOD /btr/mdr_if_auth_check_exp~auth_check_alv_tool_bar.
2  * Note: Buttons in the ALV toolbar can be disabled in this method
3
4  *****SAMPLE CODE TO DISABLE BUTTONS *****
5  FIELD-SYMBOLS: <fs_buttons> LIKE LINE OF xyt_buttons.
6  LOOP AT xyt_buttons ASSIGNING <fs_buttons>.
7      IF <fs_buttons>-function = /btr/cl_mdr_constants=>co_func_rset OR
8          <fs_buttons>-function = /btr/cl_mdr_constants=>co_func_ivls OR
9          <fs_buttons>-function = /btr/cl_mdr_constants=>co_func_vrnts OR
10         <fs_buttons>-function = /btr/cl_mdr_constants=>co_func_resume OR
11         <fs_buttons>-function = /btr/cl_mdr_constants=>co_func_stop OR
12         <fs_buttons>-function = /btr/cl_mdr_constants=>co_func_appsvr.
13         DELETE TABLE xyt_buttons FROM <fs_buttons>.
14     ENDIF.
15 ENDLOOP.
16
17 ENDMETHOD.

```

Even with Diffuser mode selected the impact of the code is shown below:

The screenshot shows the MiniCube - Explorer application interface. The top menu bar includes List, Edit, Goto, Settings, System, and Help. Below the menu is a toolbar with various icons. The main window displays a tree view on the left with the following structure:

- Diffuser Runs
 - Z Accelerators
 - Apps
 - /BTR/SAMPLE_FLIGHT_REPORT

The main pane shows the details for /BTR/SAMPLE_FLIGHT_REPORT, displaying a table with the following data:

Status	Instance Name	Started By	Start Date	Start Time	End Date	End Time	Status	Comp	Remaining	Active Job	-	+	CG
Finished	American Airlines Flights	TENGLAND	05.05.2015	17:06:12	05.05.2015	17:07:54	Finished	100%					
Finished	Lufthansa Flights	TENGLAND	05.05.2015	17:04:01	05.05.2015	17:05:54	Finished	100%					
Finished	American Airlines Flights	TENGLAND	28.02.2015	01:38:28	28.02.2015	01:41:15	Finished	100%					
Finished	British Airways Flights	TENGLAND	28.02.2015	01:37:41	28.02.2015	01:40:19	Finished	100%					
Finished	All Flights	TENGLAND	28.02.2015	01:37:13	28.02.2015	01:40:26	Finished	100%					
Finished	Lufthansa Flights	TENGLAND	28.02.2015	01:36:57	28.02.2015	01:38:48	Finished	100%					

Application Program Interface

A suite of APIs have been introduced to allow the retrieval of information and the administration of a Diffuser instance from code external to Diffuser.

The following sections will provide details on how to use the APIs, if you do run into difficulties or have questions please reach out to our support team, details on how to do that are [here](#).

- [Overview](#)
- [Selecting Instances](#)
- [Get Details](#)
- [Pause](#)
- [Restart](#)
- [Change Jobs](#)

Overview

A suite of APIs have been introduced to allow the retrieval of information and the administration of a Diffuser instance, these have been built in the Function Group /BTR/MDR_API and have been remote-enabled.

Selections of Diffuser instances similar to the MiniCube selection can be made see [here](#) for details.

The Instance ID or Jobname and Jobcount can be used as parameters and used to perform the following actions on a Diffuser instance:

- Pause Instance, further details [here](#)
- Restart Instance further details [here](#)
- Change number of processors running against an instance further details [here](#)

The following information on a Diffuser instance can be retrieved:

- Status
- Estimated time remaining
- Percentage complete
- Number of intervals completed
- Number of intervals remaining
- Number of active background processes operating

Selecting Instances

To select a list of instances and their details use the Function Module /BTR/MDR_GET_INSTANCES here you can use a number of selection criteria to retrieve instances of programs that have run or are still running using Diffuser.

The following selection criteria can be used and ranges can be entered if required:

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Instance Object ID – The unique object ID generated for each instance run

Start User – User who started the instance run

Start Date – Start date of the instance run

Start Time – Start time of the instance run

End Date – End date of the instance run

End Time – End time of the instance run

Instance Status – Current status of the instance (see [instance](#) details for technical codes)

Program Name – Technical program name of the the Diffuser program



Note that Diffuser programs run in the foreground do not have a Background Job Name or a Job ID

The following details are returned for each instance that meets the selection criteria:

Client – The SAP client that the data was retrieved from

Instance Object ID – The unique object ID generated for each instance run

Program Object ID – The unique object ID generated for each program configured into the Diffuser framework

Program Version – The version of the man program

Result Object ID – The unique object ID for the result set

Run Variant – Name of variant (without program name)

Instance Label – The description given to the instance run

Searchterm – Instance Label

Log Handle – The handle for the application log

Instance Status – Current status of the Diffuser program (see [instance](#) details for technical codes)

Start User – User who started the instance run

Start Date – Start date of the instance run

Start Time – Start time of the instance run

Collation Start Date – Start date of the collation routine

Collation Start Time – Start time of the collation routine

End Date – End date of the instance run

End Time – End time of the instance run

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Archive – Archive flag

Get Details

The Function Module /BTR/MDR_GET_INSTANCE_DETAILS returns details for a single instance.

The following selection parameters can be used, however, you must use the Instance Object ID or Background Job Name and Job ID, it is recommended that the Instance Object ID is used as this is unique and always exists, job details are not present for instances run with a dialog session.

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Instance Object ID – The unique object ID generated for each instance run

Start User – User who started the instance run

Start Date – Start date of the instance run

Start Time – Start time of the instance run

End Date – End date of the instance run

End Time – End time of the instance run

Instance Status – Current status of the instance (see [instance](#) details for technical codes)

Program Name – Technical program name of the the Diffuser program

The following details are returned for the instance:

Instance Object ID – The unique object ID generated for each instance run

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Status – The technical code for the current status of the instance (see [instance](#) details for technical codes)

Status Description – Description for the current status of the instance (see [instance](#) details for technical codes)

Estimated Time to completion – Estimated amount of time left to complete the instance

Percentage – Percentage of intervals that are complete for the instance

Intervals Completed – Number of intervals with a completed

Intervals Remaining – Number of intervals which remain to be completed

Active Jobs – Number of active background jobs currently progressing against the instance

Pause

The Function Module /BTR/MDR_INSTANCE_PAUSE pauses an instance so that all jobs stop once they complete the interval they are processing, this is the same as using the [pause](#) option in the MiniCube transaction.

The following selection parameters can be used, however, you must use the Instance Object ID or Background Job Name and Job ID, **it is recommended that the Instance Object ID is used as this is unique and always exists**, job details are not present for instances run with a dialog session.

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Instance Object ID – The unique object ID generated for each instance run

The following details are returned for the instance:

Instance Object ID – The unique object ID generated for each instance run

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Status – The technical code for the current status of the instance (see [instance](#) details for technical codes)

Status Description – Description for the current status of the instance (see [instance](#) details for technical codes)

Estimated Time to completion – Estimated amount of time left to complete the instance

Percentage – Percentage of intervals that are complete for the instance

Intervals Completed – Number of intervals with a completed

Intervals Remaining – Number of intervals which remain to be completed

Active Jobs – Number of active background jobs currently progressing against the instance

Restart

The Function Module /BTR/MDR_INSTANCE_RESTART restarts an instance that had be paused, this is the similar to using the [resume](#) option in the MiniCube transaction, however, you must then call the [change_jobs](#) function module to provision jobs to the instance.

The following selection parameters can be used, however, you must use the Instance Object ID or Background Job Name and Job ID, **it is recommended that the Instance Object ID is used as this is unique and always exists**, job details are not present for instances run with a dialog session.

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Instance Object ID – The unique object ID generated for each instance run

The following details are returned for the instance:

Instance Object ID – The unique object ID generated for each instance run

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Status – The technical code for the current status of the instance (see [instance](#) details for technical codes)

Status Description – Description for the current status of the instance (see [instance](#) details for technical codes)

Estimated Time to completion – Estimated amount of time left to complete the instance

Percentage – Percentage of intervals that are complete for the instance

Intervals Completed – Number of intervals with a completed

Intervals Remaining – Number of intervals which remain to be completed

Active Jobs – Number of active background jobs currently progressing against the instance

Change Jobs

The Function Module /BTR/MDR_INSTANCE_CHANGE_JOBS the number of jobs operating on the Diffuser instance to be changed.

The following selection parameters can be used, however, you must use the Instance Object ID or Background Job Name and Job ID, **it is recommended that the Instance Object ID is used as this is unique and always exists**, job details are not present for instances run with a dialog session.

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Instance Object ID – The unique object ID generated for each instance run

Number of Jobs – Number of jobs to increase or decrease by

Action – To increase jobs by the number of jobs above use 'INCJOB' (default option) or to decrease the number of jobs use 'DECJOB'

The following details are returned for the instance:

Instance Object ID – The unique object ID generated for each instance run

Background Job Name – The SAP background job name for the instance run

Job ID – The SAP background job id for the instance run

Status – The technical code for the current status of the instance (see [instance](#) details for technical codes)

Status Description – Description for the current status of the instance (see [instance](#) details for technical codes)

Estimated Time to completion – Estimated amount of time left to complete the instance

Percentage – Percentage of intervals that are complete for the instance

Intervals Completed – Number of intervals with a completed

Intervals Remaining – Number of intervals which remain to be completed

Active Jobs – Number of active background jobs currently progressing against the instance

Software Support

Online Forum

Basis Technologies have an online forum containing over 250 searchable Frequently Asked Questions relating to our products.

These FAQs cover many of the common error / warning messages that can be experienced during normal usage and also useful HOW TO guides to perform many of the common operations.

The online forum can be accessed via the following URL:

<http://support.basistechnologies.com/forums>

You will need to register for a username and password before you can access the forum.

Support from Basis Technologies

Raising Support Tickets

To request support from Basis Technologies on any issue relating to our product sets (ActiveControl, Transport Espresso, DevOps, Testimony, Diffuser, Utilities or Transformation), a ticket should be raised via the following email address:

support@basistechnologies.com

Sending an email to this address will automatically create a ticket in Zendesk, the ticketing tool used by Basis Technologies.

To help us offer you the best service with your issue, please include as much information as possible about the issue, with particular attention to the following:

- **Customer:** Include the name of the customer you are representing, it may not always be obvious from your email address
- **Product and Version:** Include the Basis Technologies product and version that you are operating that has the issue
- **System & Client:** The system and client where the issue/fault occurred and if it's a license key issue provide the SAP system installation number (it is always ten digits long)
- **Description:** A clear description of the problem and the steps to replicate the issue, with screen shots
- **Data:** Any master or transactional data objects associated with the issue. E.g. Business Partner, BPEM Case ID, Plant
- **Error Messages:** Details of any error or warning messages given including where applicable run time errors, short dumps and error logs
- **User ID:** The User ID being used when the issue occurred
- **Authorisations:** Ensure transaction SU53 is run and results shared to help with authorisation issues
- **Contact Details:** Please include your own contact details in your email
- **Priority:** Reflect any high priority issues by including URGENT or HIGH PRIORITY at the start of the email subject

Support Escalation

If you have any concerns with the service you are getting from Basis Technologies support, or wish to escalate any high priority issues please email **supportescalation@basistechnologies.com**

Require additional Information or Services?

If additional information or services relating to any of Basis Technologies product sets is required, you can contact us via the above support@basistechnologies.com address, or alternatively by contacting your assigned Basis Technologies Account Director.