MICRO FOCUS

Data Express 4.0

Process for z/OS Guide

2020-01-31

Contents

Process for z/OS Guide

Outlines the concepts and procedures used by Micro Focus Data Express for z/OS. It shows how to start and work with Data Express in the mainframe environment.

Data Builder is a general-purpose data analysis module for z/OS environment. The module comprises a part on the z/OS platform and one on a PC. All operations and analysis are performed in the z/OS environment. All the analysis results are stored in a DB2 repository.

Who Should Read this Guide

This guide is intended for z/OS users who want to verify the integrity of data in databases (for example, DB2 or DL/I) or data sets (for example, VSAM, SEQUENTIAL, or GDG) of applications running on z/OS that are either developed internally or purchased by external suppliers.

The use of Data Express requires a minimum experience of the mainframe platform z/OS.

Considerations

Bear in mind the following points about Data Express:

- To use this product, you must have the correct licence. See the *Installation Guide* for more details.
- You do not need to analyze the sources of the programs in the application in order to use Data Express.
- Data Express does not perform any operation capable of altering the analyzed data.

Project Implementation

Data Inventory

The information about your data store that is loaded into the Data Express Knowledge Base is as follows:

- Data store (or table) inventory
- Type classification (for example, DB2, SEQ, or VSAM)
- Record number and record length
- Data store (or table) structure
- •Application structure

For non-relational data stores, the application structure of the Data Inventory process is organized into two phases:

- Load and analysis of the copybook that describes the structure of the data stores
- Load, analysis, and association of the data store

For relational data stores (for example, DB2), the application structure of the Data Inventory process is organized into one phase:

- Load DB2 tables

For ADABAS data stores, the application structure of the Data Inventory process is organized into one phase:

- Load ADABAS tables

Load Copybook Information

The load and analysis of the copybook can be done using the job **Load Copybook from External Interface** (see the sections *Load Copy Information From External Interface Job* in the chapter *Work with Jobs* in the *Front End Guide* and *Load Data Store Information from External Interface* in the chapter *Sequential Files* in the*Data Model Guide*.)

Load DB2 Tables

For DB2 tables with direct access, you can use the job **Load Data Store Information from External Interface** to load these tables into the Data Express Knowledge Base (see the section *Load Data Store Information From External Interface Job* in the chapter *Work with Jobs* in the*Front End Guide*).

All the information concerning the structure of the DB2 tables must be in a sequential file that will be processed to find the inventory information needed (see the chapter *Load Data Store Information from External Interface Sequential File Job* in the *Data Model Guide*).

For information about the sequential file used as input for the job **Load Data Store Information from External Interface**, see the chapter *Load Sequential File with DB2 Catalog* in the *Toolkit (z/OS)*.

Load ADABAS Data Stores

Data Express does not support a direct link to the ADABAS catalog. Instead, you must use the ADABAS FDT Report to generate an output sequential file that details the structure of the ADABAS tables. This sequential file is then used to load your ADABAS data store information into the Data Express Knowledge Base.

However, in order to load the table information into the KB, you must specify the output sequential file while submitting the job **Load Data Store Information from External Interface** (see the section *Load Data Store Information From External Interface Job* in the chapter *Work with Jobs* in the *Front End Guide.*)

For information about the sequential file that is used as input for the job **Load Data Store Information from External Interface**, see the chapter *Load Sequential File with ADABAS FDT report* in the **Toolkit (z/OS)** guide.

For information about the structure of the sequential file, see the section *Load File Information from External Interface* from chapter *Sequential File* in the *Data Model Guide*.

Load Other Data Stores

Loading all the other types of data stores into the Knowledge Base can be done by executing the following:

- Job **Load Data Store Information from External Interface** (see the section *Load Data Store Information From External Interface Job* in the chapter *Work with Jobs* from the*Front End Guide*).
- Single load job for all the other types of data store including VSAM or GDG (see the sections *Load VSAM data set information Job*, *Load SEQ/GDG Data Set Information Job*, *Load DL/I Database Information Job*, and *Load DB2 table information by unload Job* in the chapter *Work with Jobs* in the *Front End Guide*).

  Note: When using VSAM RRDS data sets, you must specify RRDS in the interface instead of a generic VSAM term. In addition, you must create a process ID similar to GENDA for RRDS data sets, but use I/O program KURRDS instead of KURFIO.

For the DB2 tables with unload, the inventory information is located in the SYSPUNCH, while for the other types of data stores, the inventory information is located in the copybook associated with them.

The copybook association can be manual or automatic:

- Manual (see the section *Load Copybook Information from External Interface Job* in the chapter *Work with Jobs* from the*Front End Guide*).

- Automatic with the job **Load Data Store Information from External Interface** (see the section *Load Data Store Information from External Interface* in the chapter *Sequential Files* from the *Data Model Guide*).

Data Classification

Data Express groups fields automatically into homogeneous categories (or classes), on the basis of both their contents "snapshot" and their format and naming convention.

Data classification can be performed in four different ways:

1. Using the job **Import Classification from Referential Integrity**
2. Using the job **Import Classification from Data Dictionary**
3. Through manual association
4. Through Data Analysis

Import Classification from Referential Integrity Job

The job **Import Classification from Referential Integrity** lets you import the data classification from a CSV file with Referential Integrity relations (see the sections *Import Classification From Data Dictionary Job* in the chapter *Work with Jobs* in the *Front End Guide* and *Load Referential Integrity Relation Information* in the chapter *Sequential Files* in the *Data Model Guide*). This job can be used for data subsetting. This job sets the estimated class assignment, which can be confirmed in the **Work with Fields** area of Data Builder (see the chapter *Work with Fields* in the *Front End Guide*).

> **Note:** If you are using Data Express for z/OS with the optional ODBC Extension, you can specify your ODBC DSN in the **ODBC DSN Name** field, provided that it is less than 44 characters in length.

Import Classification from Data Dictionary Job

The job **Import Classification from Data Dictionary** lets you import the data classification from a sequential file with the list of the field and the related class that must be associated with it (see the chapter *Import Classification From Data Dictionary Job* in the *Front End Guide*). This job can be used for data masking.

> **Note:** If you are using Data Express for z/OS with the optional ODBC Extension, you can specify your ODBC DSN in the **ODBC DSN Name** field, provided that it is less than 44 characters in length.

Manual Association

Manual association lets you associate a class with a specific field or a list of fields using Data Builder (see the section *Window Contents* in the chapter *Work with Classes* in the *Front End Guide*).

Data Analysis

In the same way a fingerprint is used to identify a person, the fingerprint is a way to identify a field by its characteristics. The base concept is that different kinds of fields have different types of content and therefore have different fingerprints.

The fingerprint is calculated by the Sampling function, which displays concisely the content of the data store data elements or a sample of the values contained in the fields with the number of occurrences for each of them.

After having calculated the fingerprint, if you know that the corresponding data element belongs to a specific class of data element (for example, customer code), you can use the fingerprint of this data element as a prototype, and assign it to this class.

The next step is to search all the data elements that have a fingerprint similar to the prototype assigned to the class with the job **Class Data Element Assignment** (see the chapter *Class Data Element Assignment Job* in the *Front End Guide*). This job suggests a class that is assigned a level of confidence calculated using recognition algorithms based on the compatibility of definition attributes and similarity of contents.

After job execution, the suggested class can be confirmed (see the section *Estimated ClassesConfirm Tab* in the chapter *Work with Data Elements* in the *Front End Guide.*)

DB2 Catalog Synchronization

The job **DB2 Catalog Synchronization** loads information for the cardinality for DB2 tables and the existence of a key or the existence of an image copy into the Data Express Knowledge Base.

The job must be executed before subsetting the data using Data Subset Extraction. If cardinality information or the existence of a key or image copy stored in the system catalog has changed, the job must be executed again.

For more information, see chapter *DB2 Catalog Synchronization* in the *Toolkit (z/OS).*

Data Subsetting

The Data Subsetting function enables the creation of a new reduced environment starting from an existing environment using selection criteria specified by the user.

The application structure of the Data Subsetting function is organized in two phases:

1. Creation of methods to generate a test environment
2. Submission of extraction jobs (see the chapter *Phase 4 Launch of the Extraction Jobs* in the *Data Subset Extraction Guide*).

Method Creation

A method can be created by:

- The job **Import Method from Referential Integrity** (see the chapter *Import Method From Referential Integrity Job* in the *Front End Guide*). The method is created starting from a data store for a specific class that includes all the related files indicated in the CSV file that contains the Referential Integrity relation.
- The Data Subset Extraction wizard (see the chapter *Create New Method Wizard* in the*Data Subset Extraction Guide*). Starting from a class or from a related class, the wizard guides you in creating a method with all the data stores that have the specified classes; with this wizard, you can define the logical model of data, define the rules of subsetting, and add or remove files and specify the input and output data set for files involved in the method.
- The Data Masking module. This module can be used only if the new environments are created only applying data masking techniques and all the data stores will be expanded separately (see the *Data Masking Guide*).

Work with Method Function

After the creation of the method, it can be edited in the "Work with method" function. The following list describes the capabilities of this function:

1. Add or exclude a data store from the method.
2. Change the output file.
3. Modify the filter (reduction criterion) for a data store (or for all data stores with the same filter).
4. Specify one or more output classes for a data store (if they are not generated during the initial method creation) using the same values for the pairs: (selection class, input filter) and (out class, output filter). In order for an output class to be used as an input class for another data store, during the processing of the data store (in a table of the Knowledge Base called **ANENVLST**) the values for the data element containing the output class are created.

    Note: In this example, two entries are generated for the same data store with all the same values except for the output class. During the extraction, all the entries for the same data store are

processed together, and the class information is stored in the table **ANENVLST** for each output class.

5. Specify one or more input classes (multiple filters) for a data store. In this case, an OR condition is implicitly generated by Data Express. (If you want to use an AND condition, you should create a combined data element and specify a single condition on the data element.)

See the chapter *Work with Method Multiple Filters* in the *Data Subset Extraction Guide*.

Extraction Jobs Submission

During the extraction job, all the data stores that have data elements with a data masking routine associated with them will be automatically masked.

The Data Subsetting process manages all types of data stores processed by Data Builder with the following exceptions:

- DB2 with direct access residing in a different DB2 subsystem from the one in which the product is installed
- DL/I file with direct access

In these two cases, before creating the extraction method, you must specify a sequential unload of the files and insert the Process Identifier for reading the files, as described in the section *Specifying Unload Files for DL/I and DB2 Direct Access Files* in the chapter *Data Subset Extraction Creation* in the *Data Subset Extraction Guide*.

Output Flat File

For all files involved in the subsetting, the product output is sequential files in the same format of the input. Depending on the mode used to create the method, there are different ways to indicate this output flat file.

If the method was not created (as the new environments must be created only applying data masking techniques), the name of the flat file must be indicated for all the files to be expanded.

Import Method from Referential Integrity

If the method is created by the job **Import Method from Referential Integrity**the name of the output flat file and the routine used to write it can be specified using Data Subset Extraction. In this case, you must manually update this information in the **Elaboration properties** windows for all the files involved in the method (see the chapter *Work with MethodElaboration Properties* in the *Data Subset Extraction Guide*). Otherwise, this operation can be done with a query that directly updates the DB2 table **HSDCHFIL** that contains this information. For example, the following query can be used:

```
UPDATE EXN01.HSDCHFIL A
SET WRTPGM = 'KDCWRT', UNLOUTNAM = (SELECT 'HAL.KBAIMD.UNLOAD.' CONCAT TSNAME
FROM EXN01.HSURDFIL B
WHERE A.MCRECID = B.MCRECID AND A.FILRECID = B.FILRECID)
WHERE METHOD = 'METHOD4'
```

This query updates the name of the routine (field **WRTPGM**) and the flat file (**UNLOUTNAM**) for the DB2 tables. The name of the flat file is based on a string to which is added the name of the tablespaces (present in the field **TSNAME** of the table **HSURDFIL**: this table contains all the information concerning data stores loaded into Data Express). It is assumed that the names of the tablespaces are always different in order to obtain different flat files. However, if this is not so, this query can be customized for your user environment.

For a sequential file, add a string to the file name as shown in the following query:

```
UPDATE EXN01.HSDCHFIL A
SET WRTPGM = 'KDCWRT', UNLOUTNAM = (SELECT 'HAL.KBAIMD.UNLOAD.' CONCAT FILNAME
FROM EXN01.HSURDFIL B
WHERE A.MCRECID = B.MCRECID AND A.FILRECID = B.FILRECID)
WHERE METHOD = 'METHOD4'
```

For both previous queries, the owner of the tables **HSDCHFIL** and **HSURDFIL** must be changed in accordance with the parameters specified during the installation of Data Express. Moreover, instead of using the string **'METHOD4'** in the field **METHOD**, you must indicate the name of the method to be executed, and instead of using the string **'HAL .KBAIMD.UNLOAD.'** you can indicate another string with the initial part of the output flat file.

Data Extraction Wizard

If the method is created with the Data Subset Extraction wizard, the name of the flat file and the name of the routine used to write it are automatically inserted in the table **HSDCHFIL**.

The name of the flat file can be generated following one of these three rules:

1. A prefix is added to the unload data set name, or, if it doesn't exist, to the file name.
2. A suffix is added to the unload data set name, or, if it doesn't exist, to the file name.
3. A string replaces a part of the unload data set name or, if it doesn't exist, the file name.

One of these three rules generates a consistent flat file name for all files except the DB2 tables loaded with direct access. For this kind of file, all these three rules would produce a name with the wrong syntax. So, as the DB2 tables with direct access do not have the unload data set name, you can update the field **UNLFILNAM** in the table **HSURDFIL** before creating the method. For example, you can use the following query:

```
UPDATE EXN01.HSURDFIL A
SET UNLFILTYP = 'SEQ', UNLFILNAM = CONCAT ('HAL.KBAIMD.UNLOAD.', TSNAME)
WHERE UNLFILNAM =' AND FILTYPE = 'DB2'
```

Also in this query, the owner of the table **HSURDFIL** must be changed in accordance with the parameters specified during the installation of Data Express, and instead of the string **'HAL .KBAIMD.UNLOAD.**you can indicate another string that will be the initial part of the output flat file.

Data Masking

This process allows for the creation of a new environment, starting from an existing environment and based on masking routines that must preserve:

- the original meaning of the information
- data referential integrity
- the unique key

Data Express provides a standard masking routine for:

- NamesSurnames
- Company names
- Addresses
- Unique alphanumeric codes
- Telephone numbers

However, you can develop customized masking routines for each standard case (for example, compress data or data with control characters) using standard programming languages (COBOL, PL/I, or C).

The application structure of Data Masking is organized in two phases:

1. Creation of methods to generate a test environment (see the chapter *Create New MethodWizard* in the *Data Subset Extraction Guide*). This phase is not mandatory in that if the method is not created; all the files are expanded separately.
2. Submission of masking jobs (see the chapter *Work with Data Changer Jobs* in the*Data Masking Guide*).

Life Cycle

This chapter explains the day-to-day activities to update Data Express with the normal changes that occur to the files, tables, and copybooks in the user environment.

The Life Cycle process manages modifications within file layouts, preserving any manual information already entered into the Data Express Knowledge Base.

The application structure of Life Cycle is organized in three phases:

    Phase 1: Changed Data Stores Identification
    Phase 2: Changed Data Elements Identification
    Phase 3: Apply Identified Changes

For known restrictions pertaining to the Life Cycle process, see chapter *Life Cycle Introduction* in the *Life Cycle Guide*

Phase 1

Phase 1, starting from a list of copies or of new or modified DB2 tables, verifies if they are already in the Data Express Knowledge Base and then produces the list of data stores seen in the **Work with Changed Data Stores** area.

The list of copies or DB2 tables from which phase 1 starts must be in a sequential file named like the project and group specification and must be followed by the suffix **IMPSRC**. (For the structure of this file, see chapter *IMPSRC sequential FileModified Source* in the *Data Model Guide.*) The file can be produced by:

- The Data Express Toolkit (See the chapter *Load IMPSRC for Life Cycle* in the guide *Toolkit (z/OS)*).
- The user with a change management system.

Before moving on to the next phase, you must examine the list in the **Work with Changed Data Stores** area to confirm what Data Express has found, and possibly integrate it with additional information. (See the section *Work with Changed Data Stores* in the *Life Cycle Guide.*)

Phase 2

Phase 2 starts from the list of copies or new or modified DB2 tables present in the **Work with Changed Data Stores** area that are correctly expanded by phase 1 (that is only the record with **File Confidence** = **Recognized**) and compares their current structure with what is already defined in the Data Express Knowledge Base to verify if there are matches between their data elements.

At the end of this phase, a list of changed data stores and data elements is produced, which you can verify and approve. Thanks to a confidence parameter, you can identify where the tool could recognize the current features automatically, where a manual confirmation of what has been found is required, or where the match between the original and the modified features must be set manually.

Before moving on to the next phase, you must examine the list in order to confirm what the product has found and, possibly, integrate it with additional information. (See the chapter *Work with Changed Data Stores* of the *Life Cycle Guide.*)

Phase 3

Phase 3 applies all the changes identified in the previous phases and, if required, makes a backup of the old data. This phase also expands all copies or DB2 tables that were correctly expanded by phase 2 (that is only the record with **Field Confidence** = **Recognized** and **Field State** = **Executed**).

All the information already contained in the Data Express Knowledge Base (for example, sampling rules, associations of classes with data elements, associations of the Data Changer function with routines, test

environment creation methods, and definition of combined data elements) are saved and applied to the new structure of the file.

A backup project is useful when all the information about the files that undergo the Life Cycle procedure needs to be preserved.

During the definition of a project, you can define the backup project associated with it. When the Life Cycle procedure is started on the files of a project associated with a backup project, the new file layouts will be visible within the project itself, whereas the information about the file with the layout considered to be obsolete may be retrieved from the backup project.

> **Note:** Only one version for each file is stored in the backup project. Thus, if the Life Cycle causes a file to be modified twice, the backup project will only include the version before last whereas the version prior to that will no longer be stored.

Image Copy

Data Express manages image copies of DB2 tables in the Data Masking and Data Subset Extraction modules.

The DB2 tables must be loaded into the Knowledge Base with direct access. Then, during the masking or subsetting process, you can get the data from the backup obtained by image copy instead of the production table.

To obtain information from the image COPY, create a process identifier with the appropriate access type as described in the *Front End Guide*. In the following example, we assume the process identifier is **DB2IC**.

Before beginning the extraction or masking phase, you must execute the synchronization phase, to load the information concerning the image copies into Data Express.

Then, the name of the image copies data set must be inserted in the field **UNLINPNAM** of the data store **HSDCHFIL** (This information is specified in the **File Name** data element in the **Unload Input File Properties** section of the **Elaboration properties** window of the **Work with Method** area in Data Subset Extraction) so the process can read the correct image copy as input file. (For more information, see the section *Window Contents* in the chapter *Work with MethodElaboration Properties* in the *Subset Extraction Guide*.)

To do this, use the exit routine **UTERTIC** (that must be customized to obtain the name of the image copies matching your environment) or use a query such as the following:

```
UPDATE KBAIMD.HSDCHFIL A
SET UNLINPTYP = 'SEQ', PROCIDINP = 'DB2IC', UNLINPNAM = (
SELECT DSNAME FROM KBAIMD.HSURDICP B, KBAIMD.HSURDFIL D
WHERE D.MCRECID = 1
AND D.FILRECID = A.FILRECID
AND D.TSNAME = B.TSNAME
AND D.DBNAME = B.DBNAME
AND B.DSNUM < 2
AND B.ICDATE = '050319')
WHERE A.MCRECID = 1
AND A.METHOD = 'PRIVACY1'
AND EXISTS (
SELECT DSNAME FROM KBAIMD.HSURDICP C, KBAIMD.HSURDFIL E
WHERE E.MCRECID = 1
AND E.FILRECID = A.FILRECID
AND E.TSNAME = C.TSNAME
AND E.DBNAME = C.DBNAME
AND C.DSNUM < 2
AND C.ICDATE = '050319')
```

This query must be updated to change the owner of the tables **HSDCHFIL**, **HSURDICP**, and **HSURDFIL** in accordance with the parameters specified during the installation of Data Express.

Additionally, instead of using the value **'050319'** in the fields **ICDATE**, you must indicate the date of the image copy from which the process must read the data; instead of **'PRIVACY1'** in the field **METHOD**, you must indicate the name of the method that must be executed, and instead of **'1'** in the field **MCRECID**, the progressive number associated with the Machine ID and the Company present in the method must be indicated.

After inserting the name of the image copy, the job **Create Data Changer** (for data masking) or the job **Create Extraction Job** (for data subsetting) can be executed. Both these jobs automatically generate a JCL that, starting from the image copy, unloads the information about the DB2 tables and the data into a temporary flat file that will be deleted at the end of the job. After this first step, the masking or subsetting program is submitted.

🖊 **Restriction:** The jobs **Test Environment Creation** and **Data Store Data Changer** cannot be used on files with access from image copy.

A similar process can be applied to the backups of the DB2 tables obtained by the unload utility of CA in VARIABLE, FIX, or COMMA SEPARATED format.

Starting Data Express for z/OS

The Data Express for z/OS client runs on z/OS systems.

To start Data Express for z/OS:

1. Either execute the following command from a TSO environment:

```
EXEC 'xxxxxxxx.yyyyyyyy.CLIST(MFDATA)'
```

or execute the following command from an ISPF command line:

```
TSO EXEC 'xxxxxxxx.yyyyyyyy.CLIST(MFDATA)'
```

where *xxxxxxxx.yyyyyyyy* is a part of the PDS library name where member MFDATA is located. In both cases, this command allocates PDS libraries to ISPF and launches the **Main Menu** in an ISPF environment as shown below:

2. Specify the appropriate option from one of the three sections as listed below.

🖊 **Note:** If this is the first time you have started Data Express on z/OS, you must first enable your product. For more information, see the section *Product Enabling*.

Submission Function	Lists an option to submit scheduled jobs automatically for execution.
Product Enabling	Lists options to retrieve and specify the software key, and check product licensing status.
Environment Creation and PTF	List options to complete the product installation and to install both single PTFs and a whole modification level. For more information, see the*Installation Guide*.

Submission Function

Use the **Submit Client Schedule Job** panel to set options for job scheduling and execution.

For more information about scheduling jobs, see the chapter *Work with* Jobs in the *Front End Guide*

Restriction: You cannot use Data to schedule **Data Store Data Element Sampling** jobs for direct access DL/I databases.

This section describes the Submit Client Scheduled Job (BURSUBM) panel.

The client scheduled job reads the Knowledge Base to determine if a new job has been created in the client **Work with Jobs** area, and then submits the jobs accordingly. The client scheduled job is also called the submitter job.

To submit client scheduled jobs:

1. From the **Main Menu** panel, select the option **Submit Client Scheduled Job**to display the**Submit Client Scheduled Job (BURSUBM)**panel as shown below:

```
              Submit Client Scheduled Job (BURSUBM)

COMMAND ===> _____

Type choices, press Enter.

Expiration Date. . . . . . . . . . *TODAY     YYYYMMDD, *TODAY,
                                              blank=No Expiration Date
   Expiration Time. . . . . . . .  0928       HHMM

Maximum Number of Total Job. . . . _____      blank=No Limit

Cycle Sleep Time . . . . . . . . . 000100     HHMMSS
Maximum Number of Job for Cycle. . _____      blank=No Limit

Submit Already Scheduled Job . . . /          Enter "/" to Select Option
Enqueue Jobs with Same Name. . . . _          Enter "/" to Select Option

JES2 Input Class . . . . . . . . . A
```

2. Insert parameter names into fields as required:

Expiration Date/ Time	Day and time at which the submitter job will stop running. Setting the **Expiration Date** parameter to the value ***TODAY** causes the submitter job to expire automatically. If you leave the **Expiration Date** parameter blank, the submitter job does not expire.
Maximum Number of Total Job	Maximum number of jobs that can be submitted. At the end of the last of these submissions, the submitter job turns itself off even if the time to switch itself off has not been reached.
Cycle Sleep Time	Amount of sleep time (*hhmmss*: hours, minutes, and seconds) between two successive job submission cycles.
Maximum Number of Jobs for Cycle	Maximum number of jobs that can be submitted in a submission cycle. Assuming you want to submit one job per hour, set the **Cycle Sleep Time** parameter to **010000**and the**Maximum Number of Jobs for Cycle**parameter to**00001**.
Submit Already Scheduled Job	An indicator of whether to submit the jobs that have already been scheduled before the submitter job. A forward slash (*/*) indicates that the option is selected. If this option is not selected, the jobs previously scheduled from the PC will be ignored.
JES2 Input Class	Input class in which the submitter job must be executed. The **JES2 Input Class** parameter updates the **KURJBNPC** job tabsheet located in the PDS **SKEL** product library. Before submitting the submitter job, you must customize this tabsheet according to your company standards.
Enqueue Jobs with Same Name	Indicates whether or not to enqueue a job if another job with the same name is already in the queue. When the value is a forward slash (*/*), only jobs with unique names are submitted and jobs with the same name as a job that is already in the queue are enqueued in a Data Express queue. When blank, all jobs are submitted without regard to the names of other jobs already running.

GDG Support

The generation data group (GDG) data sets consist of one GDG base index file and generation data sets (GDSs) within a GDG data group. The generations within a GDG are assigned names derived from the name of the GDG base. Their names can be specified via relative name (signed integer) or absolute generation name.

Data Express loads GDG files into the knowledge base via **Load Data Store Information from External Interface** (job BURLFIL), passing the generations to the job using SEQ file with a special format.

Start with a GDG base file and three generations (GDSs) created on mainframe. For example:

```
XXXXXX.GDG.TEST
XXXXXX.GDG.TEST.G0001V00
XXXXXX.GDG.TEST.G0002V00
XXXXXX.GDG.TEST.G0003V00
```

You can specify the generation names in the SEQ file using either of the two following methods:

1. Relative Name (signed integer) The SEQ file passed to the BURLFIL job must contain these names:

```
GDG   XXXXXX.GDG.TEST(0)
GDG   XXXXXX.GDG.TEST(-1)
GDG   XXXXXX.GDG.TEST(-2)
```

The three GDG generations are loaded as GDG file types in the Data Builder:

Data Store Name	Data Store Type
XXXXXX.GDG.TEST(000)	GDG
XXXXXX.GDG.TEST(-001)	GDG
XXXXXX.GDG.TEST(-002)	GDG

**2. Absolute
Generation
Name**

The SEQ file passed to the BURLFIL job must contain these names:

```
SEQ    XXXXXX.GDG.TEST.G0001V00
SEQ    XXXXXX.GDG.TEST.G0002V00
SEQ    XXXXXX.GDG.TEST.G0003V00
```

The three GDG generations are loaded as SEQ file types in the Data Builder:

Data Store Name	Data Store Type
XXXXXX.GDG.TEST.G0001V00	SEQ
XXXXXX.GDG.TEST.G0002V00	SEQ
XXXXXX.GDG.TEST.G0003V00	SEQ

Index

A

ADABAS 5

C

copybook 5

D

data analysis 6
data dictionary 6
Data Inventory 4, 5
data masking 9
Data Subsetting 7, 8
DB2 5
DB2 Catalog Synchronization 7

I

Image Copy 11
Index Term 10

L

life cycle 10

R

referential integrity 6

S

starting 12