MICRO FOCUS

Data Express 4.0

Getting Started with z/OS Data Stores

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
http://www.microfocus.com

2017-07-21

Contents

Getting Started with z/OS Data Stores

Contains explanatory information and a series of tutorials that show you how to start using the product in order to analyze and manage data stores from System z servers running z/OS.

The tutorials introduce the major features in a number of short sessions, typically 10 - 30 minutes.

This guide is intended for customers who are using z/OS data stores. If you have the Data Express for z/OS solution but have also purchased the additional ODBC Extension or Oracle Extension, use the *Getting Started with Distributed Data Stores Guide* instead.

Product Components

The following is a brief description of the major components of Data Express:

Data Builder	The core Data Express module that enables you to take inventory of your data, and store information in the Data Express Knowledge Base, which resides on z/OS. Data Builder works with both Data Masking and Data Subset Extraction.
Data Masking	The module that enables you to generate a test environment where personal and sensitive data has been made anonymous by using predefined or customized masking routines.
Data Subset Extraction	The module that enables you to generate a test environment where a consistent and congruent subset of data has been extracted to allow for a manageable volume of data.
Workspace	A two-dimensional working environment definition from which you can categorize and classify data stores for your test environment.
Data Express Knowledge Base	A collection of tables within a database that contain all information about catalogued files, including properties and processing rules. The Knowledge Base for the Data Express for z/OS solution is a Mainframe DB/2 repository located in your z/OS environment.

Using z/OS Data Stores

This section provides a high-level overview of the work-flow for the Data Express for z/OS solution. For more information, see the . The work-flow process for z/OS data stores looks like this:

- Data Inventory Phase (Data Builder, Work with Jobs)
  - Non-relational data stores
    - Load and analysis of the copybook
    - Load, analysis, and association of the data store
  - Relational data stores
    - Load DB2 or ADABAS tables
- Data Classification Phase (Data Builder, Work with Jobs)
  - Manual Association
  - Import Classification from Referential Integrity
  - Import Classification from Data Dictionary

    - Data Analysis
        - Sampling
        - Fingerprinting
- Data Masking Phase (Data Masking)
    - Method creation (not mandatory and only used in conjunction with Data Subset Extraction to create a test environment)
    - BTUNLDR - Create Data Changer Jobs (Data Builder, Work with Jobs)
- Data Subsetting Phase (Data Subset Extraction)
    - Method creation
        - BTEDDUR - Import Method From Referential Integrity job (Data Builder, Work with Jobs)
        - Create New Method Wizard (Data Subset Extraction)
        - Data Masking module
    - Test Environment Creation (Data Builder, Work with Jobs)
- Life Cycle (Data Builder)

Data Express for z/OS Tutorials

These tutorials show you how to prepare your Data Express working environment in z/OS and introduce the Data Express Windows client, which consists of three Data Express modules: Data Builder, Data Masking, and Data Subset Extraction. Follow the tutorials in this guide in order after you have completed your product installation.

The easiest way to learn about Data Express is to complete a few simple tasks. This guide takes you through configuring Data Builder, loading metadata into the Data Express Knowledge Base, using Data Masking to implement masking rules on a small amount of data, using Data Subset Extraction to create a simple subset of data, and finally using Data Subset Extraction twice more to create subsets of data using referential integrity. Please note that some of the data reduction examples also mask data, showing that Data Express can do both functions in one method invocation.

The three modules, Data Builder, Data Masking, and Data Subset Extraction, and tutorial files contain all the tools you need to perform these tasks.

This tutorial describes the process of configuring your z/OS environment so that you can use the Data Express Windows client (as demonstrated in subsequent tutorials) for data masking and data reduction.

Machine IDs and Company Codes

A Machine ID and Company code pair is essentially an identifier for your Data Express working environment or workspace. The Machine ID is the higher-level identifier, while the Company code must be identified by a Machine ID. You can create a new Machine ID and a new Company code for each test environment, or you can create new Company codes under a single Machine ID as needed for each test environment you want to create.

Note: A single machine ID can correspond to numerous company codes. A new company code can be created only if an existing machine ID is first selected.

Sample Data

Data Express includes the files that create the sample DB2 tables and sequential files that are used throughout the exercises in this tutorial . The database and its sample tables must be created and populated after product installation.

Note: The JCL job streams, files, and scripts needed to create your tutorial environment are located in the `tutorial` folder of your installation directory.

In this session, you load data needed for this tutorial into your mainframe environment and you use Data Builder to define your workspace. This involves editing several files, executing programs, creating a

machine ID and company code pair for your workspace, and submitting jobs that will load information into your mainframe environment from external files.

You can follow the tutorials for either DB2 data stores, sequential files or both DB2 data stores and sequential files.

Sample Sessions

The sample sessions are broken out into two parts:

- DB2 data stores
- sequential files

Note: Sessions in this tutorial must be done in order. If you choose to complete the tutorial for both DB2 and sequential data, then do all sessions. If you choose to complete the tutorial for only DB2 or sequential data, then follow the sessions only for that data type.

DB2 Data Stores Session

In this session, you prepare your z/OS environment for DB2 data stores:

1. Import JCL job streams into your mainframe environment.
2. Create DB2 tablespace and tables.
3. Start Data Builder.
4. Set your workspace.
5. Load data store information from an external interface.

Importing JCL Job Streams into Your Mainframe Environment

In order to import the `JCL CREATBT` and `CREASEQD` into your mainframe environment, you must first edit an FTP script, `PUT_JCL_DB2`, to include your login credentials.

Next, you run `FTPJCLDB2.BAT`, which executes the FTP commands listed in the file `PUT_JCL_DB2.txt` to transfer the JCL job streams to an existing library in your mainframe environment.

To import the JCL job streams `CREATBT` and `CREASEQD` into your mainframe environment:

1. Edit the FTP script in the **PUT_JCL_DB2.TXT** file with the following information:

   ```
   123.123.123.123   the IP address of your mainframe
   YOURUSERNAME      your mainframe user name
   YOURPASSWORD      your mainframe password
   YOURJCLLIBRARY    the name of the library that you want to contain the JCL
   ```

   This FTP script transfers the JCL job streams `CREATBT` and `CREASEQD` onto the mainframe environment into your the specified JCL library when `FTPJCLDB2.BAT` is executed.
2. Ask the DBA to customize the following parameters in the `CREATBT.TXT` file:

   - DSN
   - DSN SYSTEM
   - PLAN
   - LIB
   - TABLESPACE
   - DATABASE NAME
   - STOGROUP
   - PRIQTY
   - SECQTY
   - FREEPAGE
   - PCTFREE

- BUFFERPOOL
- SEGSIZE
- COMPRESS

The CREATBT.TXT file contains the JCL that is used to create the tablespace DETESTLG and the following DB2 tables:

- CUSTOMER
- ACCOUNT
- CCARD
- OPERAT

3. Change each instance of the XXXXXX.XXXXXX library in the CREASEQD.TXT file to URADAR.DEMOT. For example, DSN=XXXXXX.XXXXXX.LISTDB2 becomes: DSN=URADAR.DEMOT.LISTDB2.

   Note: The library URADAR.DEMOT is used as an example throughout this tutorial. You can use any library that suits your mainframe library naming conventions. However, it is assumed that the mainframe user ID you are using has no restrictions and can therefore access this library.
4. Execute FTPJCLDB2.BAT to import the JCL job streams into your mainframe environment.
5. Continue to the next section.

Creating the DB2 Tablespace and Tables

In this exercise, you submit the CREATBT JCL (to create the tablespace and tables) and the CREASEQD JCL needed to complete your environment creation. To create the DB2 tablespace and tables:

1. Within your TSO environment, submit the JCL job streams CREATBT and CREASEQD.
2. Review results to verify that the jobs ran successfully.
3. Edit the FTP script in the PUT_FILE_DB2.txt file with the following information:

```
123.123.123.123    the IP address of your mainframe
YOURUSERNAME             your mainframe user name
YOURPASSWORD                your mainframe password
XXXXXX.XXXXXX      the name of the library created by JCL CREASEQD, for
example: URADAR.DEMOT.
```

   This FTP script creates the files **LISTDB2** and **REFINTEG** specified by the **CREASEQD.TXT** JCL.

   Note: The library URADAR.DEMOT is used as an example throughout this tutorial. You can use any library that suits your mainframe library naming conventions. However, it is assumed that the mainframe user ID you are using has no restrictions and can therefore access this library.
4. Within your Windows environment, execute FTPTUTORIALDB2.BAT.BAT to import the informaiton into your mainframe environment.
5. Continue to the next section.

Starting Data Builder

Data Builder enables functionality for the Data Masking and Data Subset Extraction modules. With Data Builder, you can take an inventory of data, and classify the data as you see fit. These processes store information about the data in the Data Express Knowledge Base repository.

Note: You will use the settings shown in this section throughout each subsequent chapter of this Getting Started guide.

1. Click **Start > All Programs > Micro Focus Data Express 4.0 > Data Builder**.
2. Select the mainframe Knowledge Base as the required database with which to launch the connection. In this example, the database is MVSDB2. By default, the highlighted item in the list of available databases is the last database you connected to using the Data Builder module. However, for the very first connection, MVSDB2 – OWUR40SV is selected by default. Note: By selecting MVSDB2 – OWUR40SV, you do not need to choose the database and then the schema OWUR40SV.

3. Click **OK**.
4. Log in using your mainframe user name and password.

   The **[Work with Data Stores]** window provides a list of files cataloged using different sets of search criteria, including:

   - Machine identifiers
   - Company names
   - Application identifiers
   - Data store types
   - Data stores with assigned classes (Multiple class Selection)
   - Data stores with assigned copybooks
   - Data store with sampling activated
   - Data stores with Data Changer activated
   - Data stores with Data Changer inactivated
   - Data stores set as Register Table
   - Data stores without Register Table attribute

   This window is comprised of four tabs: **Structure**, **Classes**, **Data Changer**, and **Register Tables**. You can select a tab to view files using a specific set of criteria as listed previously. Each tab is associated with different grouping criteria.

   The list of displayed data stores can be further restricted by applying filters. Filters can be selected from the context menu displayed by right-clicking the **List of Files** grid, or by selecting the specific icons displayed on the left side of the screen.

   Note: The Data Masking and the Data Subset Extraction modules are launched as separate applications.
5. Continue to the next section.

Setting Your Workspace

Data Builder lets you set your workspace as part of the Data Inventory process. You must specify your workspace to organize and sort data on the basis of a logical model.

Note: If you have already set your workspace for your sequential files environment, you do not need to perform this exercise. Both DB2 and sequential file data stores share the same workspace for this sample tutorial.

1. From Data Builder, open the **Work with Machine IDs** window (click or access it through **Environment** > **Work with Machine IDs**) to create a new Machine ID.
2. Click **New**.
3. Specify your name as the name of the machine ID in the **Machine ID** box. For this tutorial, type: `DATAEXPRES`.
4. Type a description of your machine ID in the **Description** box. For this tutorial, type: `DEMO ENVIRONMENT`.
5. Specify additional information if appropriate. For this tutorial, the additional fields have been left blank. Click **Apply**.
6. Click **OK** to save your specification and to close the window.
7. From Data Builder, open the **Work with Companies** window (click or access it through **Environment** > **Work with Companies**) to create a new company code.
8. Click **New**.
9. Select the `DATAEXPRES` Machine ID you just created.
10. Specify `DEMO` as the name of your company code in the **Company Code** box.
11. In the **Company Description** box, type: `DX GETTING STARTED GUIDE`.

12. By default, the **Life Cycle** is activated. Leave this box selected.
13. By default, the **Is a Backup Company** checkbox is not selected. Leave this box unselected.

    During the definition of a company, the backup company associated with it can also be defined. A Backup Company is useful when all the information concerning the files that undergo the Life Cycle procedure needs to be preserved.

    Note: Only one version for each file is stored in the backup company.

14. Click **Apply**. The company you just created appears in the **List of Companies**.
15. Click **OK**.
16. Continue to the next section.

Loading Data Store Information into Your Mainframe Environment

Perform this procedure to map and load the definitions from your data store into the Knowledge Base. The information from your data store will be mapped to your workspace.

1. In your mainframe environment, execute the following from an ISPF command line: `TSO EXEC 'xxxxxxxx.xxxxxxxx.CLIST(MFDATA)'` where `xxxxxxxx.xxxxxxxx` 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.
2. Enter the following command: `TSO CUINFIL`.
3. Select the **Machine ID** and **Company** you created.
4. Specify the following parameters:

   Data Set Name Inp — Name of the input data set that contains the table and owner names. For this demo, use `URADAR.DEMOT.LISTDB2`.

   Data Set Name Out — the name of the output data set that contains the table names, column names, and column descriptions. For this demo, use `URADAR.DEMOT.LISTODB2`.

   Note: The names for input and output datasets must be different.

   Process Id — Process identifier associated with the elaborated file. For example, for a DB2 data store with direct access, use `DB2DA`.

5. Open the **[Work with Jobs]** window by clicking **Environment** > **Work with Jobs** to schedule a new job.
6. Click **New**.
7. In **List of Jobs**, select the job **BURLFIL** (Load Data Store Information From External Interface).
8. Select the `DATAEXPRES` **Machine ID** you just created.
9. Specify `DEMO` as the name of your company code in the **Company name** box.
10. Click **Apply**.
11. In the **Sequential data store** box, enter: `URADAR.DEMOT.LISTODB2`.
12. Click **OK**. The job appears in the **List of Scheduled Jobs**. The pause symbol indicates that the job is awaiting execution. Once the job has completed successfully, the job is listed with a green check mark.
13. In your TSO environment, submit the `BURLFIL` job by using the **Submit Client Scheduled Job** panel. The **Submit Client Scheduled Job** panel is accessible by selecting option **Submit Client Scheduled Job** from the **Data Express for z/OS - Main Menu** panel. For more information, see the `Submission Function` in the *Process Guide for z/OS*.
14. In the **Work with Jobs** area, click the **Refresh** button to verify that the jobs completed successfully.
15. Close the **Work with Jobs** window.
16. In the **[Work with Data Stores]** window, expand the **Machine ID** node, and select **DATAEXPRES**, to view data stores associated with the **DATAEXPRES** machine ID:

17. If you're planning to go straight on to the next session, you can keep Data Builder open. Otherwise, either click **Exit** and then **Yes**.
18. Continue to the next section if you want to complete the sequential files session. Otherwise, go to the *Data Masking*.

Sequential Files Session

In this session, you prepare your z/OS environment for sequential files:

- Import JCL CREASEQS into your mainframe environment.
- Import the copybook and sequential file information into your mainframe environment.
- Start Data Builder.
- Set your workspace.
- Load copybook information from an external interface.
- Load data store information from an external interface.

Importing the JCL CREASEQS into Your Mainframe Environment

In order to import the JCL CREASEQS into your mainframe environment, you must first edit an FTP script, a JCL, list of copybook files, and a list of sequential files. The FTP script, PUT_JCL_SEQ, must be modified to include your login credentials so that you can transfer the JCL CREASEQS to an existing library in your mainframe environment. Then, the CREASEQS JCL must be modified to list the name of the library that will contain it.

After editing the provided files, you run FTPJCLSEQ.BAT, which executes the FTP commands listed in the file PUT_JCL_SEQ.txt.

1. Edit the FTP script in the PUT_JCL_SEQ.TXT file with the following information:

```
123.123.123.123  the IP address of your mainframe
YOURUSERNAME           your mainframe user name
YOURPASSWORD              your mainframe password
YOURJCLLIBRARY   the name of the library that you want to contain the JCL
```

   This FTP script transfers the JCL CREASEQS onto the mainframe environment into your the specified JCL library when FTPJCLSEQ.BAT is executed.
2. Change each instance of the *XXXXXX.XXXXXX* library name in the **CREASEQS.TXT** file to URADAR.DEMOT.

   Note: The library URADAR.DEMOT is used as an example during instructional steps throughout this tutorial. You can use any library that suits your mainframe library naming conventions. However, it is assumed that the mainframe user ID you are using has no restrictions and can therefore access this library.

   The CREASEQS.TXT file contains the JCL that is used to create the PDS library COPYBOOK and the libraries for the following sequential files:

   - CUSTOMER
   - PHCONTRA
   - PHNUMBER
   - PHONBILL
   - EXTSEQ
   - EXTCPY

   Note: The library name URADAR.DEMOT used in this JCL must also be inserted into the FTP job within the PUT_FILE_SEQ.TXT file.
3. Execute FTPJCLSEQ.BAT to import the JCL into your mainframe environment.
4. Continue to the next section.

Importing the Sequential File and Copybook Information into Your Mainframe Environment

In this exercise, you submit the CREASEQS JCL to create the library URADAR.DEMOT in order to complete your environment creation. Then, you import sequential file and copybook into your mainframe environment.

1. Edit the FTP script in the PUT_FILE_SEQ.txt file with the following information:

```
123.123.123.123   the IP address of your mainframe
YOURUSERNAME            your mainframe user name
YOURPASSWORD               your mainframe password
XXXXXX.XXXXXX     the name of the library created by JCL CREASEQS, for
example: URADAR.DEMOT.
```

2. This FTP script inserts the COPY and RECORD lines of the sequential files into the libraries created by the CREASEQS.TXT JCL.
3. Within your TSO environment, submit the CREASEQS JCL.
4. Review results to verify that the job ran successfully.
5. Within your Windows environment, execute **FTPTUTORIALSEQ.BAT** to import the information into your mainframe environment.

   The information from the following sequential files is loaded into the output dataset you specified:

Sequential File	Related Data Element File	Related Copybook File	Data Elements
CUSTOMER	CUSTOMER_REC	CUSTOMER_COPY	• COD-CUS (customer code) • NAME (customer name) • SURNAME (customer surname) • DATE-OF-BIRTH (customer birth date) • PLACE-OF-BIRTH (customer place of birth)
PHCONTRA	PHCONTRA_REC	PHCONTRA_COPY	• COD-CUS (customer code) • OPERAT-NUM (operator number) • CONTR-NUM (contract number) • DATE-SIGN (date contract signed)
PHNUMBER	PHNUMBER_REC	PHNUMBER_COPY	• OPERAT-NUM (operator number) • CONTR-NUM (contract number) • PHON-TYPE (phone type: S=smart phone, M=mobile) • PHON-NUM (phone number)
PHONBILL	PHONBILL_REC	PHONBILL_COPY	• PHON-TYPE (phone type: S=smart phone, M=mobile) • PHON-NUMBER (phone number) • TOT-CALL (total number of incoming and outgoing calls) • PERIOD (billing cycle)

6. Continue to the next section.

Starting Data Builder

Data Builder enables functionality for the Data Masking and Data Subset Extraction modules. With Data Builder, you can take an inventory of data, and classify the data as you see fit. These processes store information about the data in the Data Express Knowledge Base repository.

Note: You will use the settings shown in this section throughout each subsequent chapter of this Getting Started guide.

1. Click **Start > All Programs > Micro Focus Data Express 4.0 > Data Builder**.
2. Select the mainframe Knowledge Base as the required database with which to launch the connection. In this example, the database is MVSDB2. By default, the highlighted item in the list of available databases is the last database you connected to using the Data Builder module. However, for the very first connection, MVSDB2 – OWUR40SV is selected by default. Note: By selecting MVSDB2 – OWUR40SV, you do not need to choose the database and then the schema OWUR40SV.
3. Click **OK**.
4. Log in using your mainframe user name and password.

   The **[Work with Data Stores]** window provides a list of files cataloged using different sets of search criteria, including:

   - Machine identifiers
   - Company names
   - Application identifiers
   - Data store types
   - Data stores with assigned classes (Multiple class Selection)
   - Data stores with assigned copybooks
   - Data store with sampling activated
   - Data stores with Data Changer activated
   - Data stores with Data Changer inactivated
   - Data stores set as Register Table
   - Data stores without Register Table attribute

   This window is comprised of four tabs: **Structure**, **Classes**, **Data Changer**, and **Register Tables**. You can select a tab to view files using a specific set of criteria as listed previously. Each tab is associated with different grouping criteria.

   The list of displayed data stores can be further restricted by applying filters. Filters can be selected from the context menu displayed by right-clicking the **List of Files** grid, or by selecting the specific icons displayed on the left side of the screen.

   Note: The Data Masking and the Data Subset Extraction modules are launched as separate applications.
5. Continue to the next section.

Setting Your Workspace

Data Builder lets you set your workspace as part of the Data Inventory process. You must specify your workspace to organize and sort data on the basis of a logical model.

Note: If you have already set your workspace for your sequential files environment, you do not need to perform this exercise. Both DB2 and sequential file data stores share the same workspace for this sample tutorial.

1. From Data Builder, open the **Work with Machine IDs** window (click or access it through **Environment** > **Work with Machine IDs**) to create a new Machine ID.
2. Click **New**.
3. Specify your name as the name of the machine ID in the **Machine ID** box. For this tutorial, type: `DATAEXPRES`.
4. Type a description of your machine ID in the **Description** box. For this tutorial, type: `DEMO ENVIRONMENT`.
5. Specify additional information if appropriate. For this tutorial, the additional fields have been left blank. Click **Apply**.

6. Click **OK** to save your specification and to close the window.
7. From Data Builder, open the **Work with Companies** window (click or access it through **Environment** > **Work with Companies**) to create a new company code.
8. Click **New**.
9. Select the DATAEXPRES Machine ID you just created.
10. Specify DEMO as the name of your company code in the **Company Code** box.
11. In the **Company Description** box, type: DX GETTING STARTED GUIDE.
12. By default, the **Life Cycle** is activated. Leave this box selected.
13. By default, the **Is a Backup Company** checkbox is not selected. Leave this box unselected.

    During the definition of a company, the backup company associated with it can also be defined. A Backup Company is useful when all the information concerning the files that undergo the Life Cycle procedure needs to be preserved.

    Note: Only one version for each file is stored in the backup company.
14. Click **Apply**. The company you just created appears in the **List of Companies**.
15. Click **OK**.
16. Continue to the next section.

Loading Copybook Information into Your Mainframe Environment

Perform this procedure to map and load copybook information for your data store into the Knowledge Base. The information from the copybook will be mapped to your workspace.

Load copybook information into the Knowledge Base:

1. 1. From Data Builder, open the **Work with Jobs** window by clicking **Environment** > **Work with Jobs**.
2. 2. Click **New**.
3. 3. In the **List of Jobs**, select the job BURLCPY (Load Copybook Information From External Interface).
4. 4. Select the DATAEXPRES **Machine ID** you just created.
5. 5. Specify DEMO as the name of your company code in the **Company name** box.
6. Click **Apply**.
7. 6. In the **Sequential data store** box, enter: URADAR.DEMOT.EXTCPY.
8. 7. Click **OK**. The job appears in the **List of Scheduled** Jobs. The pause symbol indicates that the job is awaiting execution.
9. 8. Continue to the next section.

Loading Data Store Information into Your Mainframe Environment

Perform this procedure to map and load the definitions from your data store into the Knowledge Base. The information from your data store will be mapped to your workspace.

1. From the **Work with Jobs** window, click **New**.
2. In **List of Jobs**, select the job BURLFIL (Load Data Store Information From External Interface).
3. Select the DATAEXPRES **Machine ID** you just created.
4. Specify DEMO as the name of your company code in the **Company name** box.
5. Click **Apply**.
6. In the **Sequential data store** box, enter: URADAR.DEMOT.EXTSEQ.
7. Click **OK**. The job appears in the **List of Scheduled Jobs**. The pause symbol indicates that the job is awaiting execution. Once the job has been submitted on the mainframe and has completed successfully, the job is listed with a green check mark.

8. In your TSO environment, submit the BURLFIL and BURLCPY jobs by using the **Submit Client Scheduled Job** panel.

   Note: The **Submit Client Scheduled Job** panel is accessible by selecting option **Submit Client Scheduled Job** from the **Data Express for z/OS - Main Menu** panel. For more information, see the *Submission Function* in the *Process Guide for z/OS*.
9. In the **Work with Jobs** area, click the **Refresh** button to verify that jobs completed successfully.
10. Close the **Work with Jobs** window.
11. In the **[Work with Data Stores]** window, expand the **Machine ID** node, and select DATAEXPRES, to view data stores associated with the DATAEXPRES machine ID:
12. If you're planning to go straight on to the next session, you can keep Data Builder open. Otherwise, click **Exit** and then click **Yes**.

Data Masking

The Data Masking module is used to manage the privacy of data contained in databases of applications that are either developed internally or purchased by external suppliers. The aim is to protect sensitive data such as name, surname, or company name according to privacy laws and regulations.

This tutorial explains the Data Masking configuration procedure and describes the steps required to migrate the files in your application.

Note: Sessions in this tutorial should be done in order.

This tutorial walks you through the data classification and the data masking phases.

Begin this session only after you have set your workspace and loaded data store information into the Knowledge Base in tutorial *Using Data Express* and *Data Inventory*. You must already be logged in to Data Builder to begin.

The sample sessions are broken out into two parts: one for DB2 data stores and one for sequential files.

Note: Sessions in this tutorial must be done in order. If you choose to complete the tutorial for both DB2 and sequential data, then do all sessions. If you choose to complete the tutorial for only DB2 or sequential data, then follow the sessions only for that data type.

DB2 Data Stores Session

In this session, you :

- Associate supplied masking classes to data elements
- Start the Data Masking module
- Enable masking on the DB2 table CUSTOMER for NAME and SURNAME classes
- Submit a JCL to mask customer first and last names
- Create a masking class
- Associate a masking class to data elements
- Enable masking on the DB2 table for the CODCUS class
- Submit a JCL to mask customer IDs

Associating Supplied Masking Classes to Data Elements

In this exercise you will be manually associating classes supplied with your installation with columns NAME and SURNAME in the table CUSTOMER:

1. In the **[Work with Data Stores]** window, ensure that the **Machine ID** node is expanded, and that DATAEXPRES is selected:
2. In the **List of Data Stores**, select the row for the DELGT.CUSTOMER data store.

3. Right-click and then select **Work with Data Elements of the Selected Data Store**.

   If a message appears indicating that the default confirmation options have not been set, click **Yes** to set them. Click **Apply**, and then **OK** in the **Options** window.

   You can also access the **Work with Data Elements** window by clicking or by accessing it through **Environment** > **Work with Data Elements** . However, if you choose to access the **Work with Data Elements** window directly, you may want to specify information in the fields above the **List of Data Elements** and then click **Apply Filter**.
4. Click the **Classes - Assignment** tab.
   This screen shows a filtered list based on **Machine ID** and **Company Name**.
5. In the **All Classes** pane, expand the **Privacy** Super Class, and then select the class **NAME**.
6. In the **List of Data Elements**, select the row for **CUSTOMER.NAME**. (That is, select the row for the data store `CUSTOMER` and the name `NAME`.)

   Note: If you are viewing all data stores, you could click the **Data Store** column to s sort data elements by data store.
7. Click the **Assign Class** button to assign the **NAME** class to **CUSTOMER.NAME**. `NAME` now appears in the **Assigned Class** column for **CUSTOMER.NAME**.
8. In the **List of Data Elements**, select the row for **CUSTOMER.SURNAME**. (That is, select the row for the data store `CUSTOMER` and the name `SURNAME`.)
9. Click the **Assign Class** button to assign the **SURNAME** class to **CUSTOMER.SURNAME**. `SURNAME` now appears in the **Assigned Class** column for **CUSTOMER.SURNAME**.
10. Click **Close**.
11. Leave Data Builder running, and continue to the next section.

Starting the Data Masking Module

1. Click **Start > All Programs > Micro Focus Data Express 4.0 > Data Masking**.
2. Click **OK**.
3. Select the mainframe Knowledge Base as the required database with which to launch the connection. In this example, the database is `MVSDB2`. By default, the highlighted item in the list of available databases is the last database you connected to using Data Masking. This becomes apparent after you click **Available databases**.
4. Click **OK**.
5. Log in using your mainframe user name and password.
6. Click **OK**.
7. Select the schema `OWUR40SV`, and click **OK**.

   The **Main Window** comprises four main areas:

   - **Machine ID./Company** grid
   - **All Machine ID./Companies**
   - **Multiple Class Selection**
   - **Object Distribution**

   The area containing the grid and the tabs **Enabled** and **Disabled** is referred to as the **Machine ID./Company** area or the **Grid** area. The grid lists files related to the selected class and to the selected machine ID and company. The names of the forms, their meaning and the included information are outlined below.

   The **Enabled** tab shows a list of all active files (enabling a file means that it is visible in the export functionality in order to be prepared for masking), and all files, which have one or more specific classes related to their data elements.

   The **Object Distribution** area displays a pie chart and bar chart representation of the file types for the classes selected in the **Machine ID./Companies** or **Multiple Class Section** area. The pie chart shows

the percentage distribution of file types, while the bar chart shows the numerical distribution of file types. For both charts, both enabled and disabled files are included. Click the minimize button to minimize the **Object Distribution** area.

8. Continue to the next section.

Enabling Masking on the DB2 Table CUSTOMER

Data masking lets you to enable masking for individual data stores or all data stores, and to enable masking at the data-element level or class level. You can mask at the class level or at the data-element level. In this tutorial, we will be masking at the class level.

In this exercise you will activate masking for the data store CUSTOMER at the class (NAME and SURNAME) level.

To mask at the data-element level, double-click the desired data store from the **Main Window**, right-click the desired data element, and then click **Data Changer Data Element Properties**. Click **Change**, click **Yes** to activate the **Changer** option, and finally click **Apply** to activate your setting.

1. In data masking, click the **ALL Machine ID./Companies** list.
2. Expand the **All Machine ID./Companies** node.
3. Double-click the `DATAEXPRES` machine ID.
4. Select the row for the data store `DELGT.CUSTOMER`.
5. Right-click and select **Enable**. The data store disappears from the list.
6. Verify `CUSTOMER` is set as the **Input Unload data store name** and specify `URADAR.DEMOT.MASK.new_library_name` as the **Output Unload data store name**. For example: `URADAR.DEMOT.MASK.DB2CUST`.
7. Click **OK**.
8. Click the **Enabled** tab.

   Note: You can also choose to enable all data stores that have assigned classes with masking routines by choosing **Tools** > **Enable All Data Stores**.
9. Select the data store `DELGT.CUSTOMER`, right-click, and then click **Open Selected Data Store.**
10. Expand the row for the **SURNAME** node.
11. Click the **SURNAME** class, right-click, and then click **Enable**.
12. Expand the row for the **NAME** node.
13. Click the **NAME** class, right-click, and then click **Enable**.
14. Continue to the next section.

Submitting a JCL to Mask Customer First and Last Names

Use the **Work with Jobs** tool in data builder to submit the JCL that does the actual masking on the `CUSTOMER` table. Then, submit the job in the mainframe environment and verify your results.

1. Data Builder, open the **Work with Jobs** window (click or access it through **Environment** > **Work with Jobs**) to schedule a new job.
2. Click **New**.
3. In the **List of Jobs**, select the job **BDCCHGR** (Data Store Data Changer).
4. Select the `DATAEXPRES` Machine ID and the `DEMO` company name.
5. In the **Data store type** list, click `DB2`.
6. Click **Apply**.

   The job appears in the **List of Scheduled Jobs**. The pause symbol indicates that the job is awaiting execution. Once the job has completed successfully, the job is listed with a green check mark.

7. In your TSO environment, submit the BDCCHGR job by using the **Submit Client Scheduled Job** panel.
8. In the **Work with Jobs** area, click the refresh button to verify that the jobs completed successfully.
9. Click **OK** or the button to close the **Work with Jobs** window.
10. In your mainframe environment, notice that the values in the **NAME** and **SURNAME** columns for your input and output datasets are different.
11. Continue to the next section.

Creating a Class

You can create new classes to be used for masking purposes, or you can use a predefined class. In this exercise you will create a masking class named CODCUS, which will enable us to mask the data element COD_CUS in the CUSTOMER table.

1. In **Data Builder**, open the **Work with Classes** window (access it through **Environment** > **Work with Classes**)
2. Click **New**.
3. In the **Name** field, specify CODCUS as the name of the class you want to create.
4. In the **Full description** field, specify COD_CUS MASK EXAMPLE as the description of the class. The description will be visible as a label when you assign the class to data elements within data stores.
5. Specify UDCPIVC in the **Data Masking Routine** box.

   Note: This masking routine is provided during product installation. This is just an example routine for the purpose of creating a class. The masking logic for the routine is not in consideration for this demonstration.

   Note: You can mask at the class level or at the data-element level. In this tutorial, we will be masking at the class level.
6. Select the Super Class **Generated**.
7. Select the data type **Alpha**.
8. Leave the remainder of the fields at their default values. Click **Apply**.

   Note: The class CODCUS is now listed in the **List of Classes** and is given the first free class ID.
9. Click **OK**.
10. Continue to the next section.

Associating a Class to Data Elements

In this exercise you will be manually associating the CODCUS class you just created with columns in the table CUSTOMER.

1. In the **[Work with Data Stores]** window, expand the **Machine ID** node, select DATAEXPRES, and then select the row for the DELGT.CUSTOMER data store in the **List of Data Stores**.
2. Right-click and then select **Work with Data Elements of the Selected Data Store**.
3. Click the **Classes - Assignment** tab.
4. In the **All Classes** pane, expand the **Generated** Super Class, and then select the class **COD_CUS MASK EXAMPLE**.
5. In the **List of Data Elements**, select the row for CUSTOMER.COD_CUS. (That is, select the row for the data store CUSTOMER and the name COD_CUS.)

   Note: If you are viewing all data stores, you could click the **Data Store** column to s sort data elements by data store.
6. Click the **Assign Class** button to assign the CODCUS class to CUSTOMER.COD_CUS. COD_CUS MASK EXAMPLE now appears in the **Assigned Class** column for CUSTOMER.COD_CUS.

7. Click the **Close** button to close the **Work with Data Elements** window.
8. Continue to the next section.

Enabling Masking on the DB2 Table CUSTOMER for the CODCUS Class

In this exercise you will activate masking for the data store `CUSTOMER` at the class (`CODCUS`) level.

1. In **Data Masking**, click the **ALL Machine ID./Companies** list.
2. Expand the **All Machine ID./Companies** node.
3. Double-click the **DATAEXPRES** machine ID.
4. On the **Enabled** tab, select the row for the data store `DELGT.CUSTOMER`.
5. Right-click and select **Data Changer Properties**.
6. Click Change.
7. Verify **CUSTOMER** is set as the **Input Unload data store name** and specify `URADAR.DEMOT.MASK.new_library_name` as the **Output Unload data store name**. For example: `URADAR.DEMOT.MASK.DB2CUST2`.

   Note: Be sure that you change the output data store name in this exercise or you will overwrite the data from the previous masking exercise.
8. Click **OK**.
9. From the **Enabled** tab, select the data store `DELGT.CUSTOMER`, right-click and then click **Open Selected Data Store**.
10. Expand the row for the node **COD_CUS MASK EXAMPLE** if it is not already expanded:
11. Click the **COD_CUS** class, right-click, and then click **Enable**.
12. Continue to the next section.

Submitting a JCL to Mask Customer IDs

Perform this procedure to map and load the definitions from your data store.

1. 1. In **Data Builder**, open the **Work with Jobs** window (click **Environment** > **Work with Jobs**) to schedule a new job.
2. Click **New**.
3. In the **List of Jobs**, select the job `BDCCHGR` (Data Store Data Changer).
4. Select the `DATAEXPRES` **Machine ID** and the `DEMO` **company name**.
5. In the **Data store type** list, click `DB2`.
6. Click **Apply**. The job appears in the **List of Scheduled Jobs**. The pause symbol indicates that the job is awaiting execution. Once the job has completed successfully, the job is listed with a green check mark.
7. In your TSO environment, submit the `BDCCHGR` job by using the **Submit Client Scheduled Job** panel.
8. In the **Work with Jobs** area, click the **Refresh** button to verify that the jobs completed successfully.
9. Click **OK** or the button to close the **Work with Jobs** window.
10. In your mainframe environment, notice that the values in the `COD_CUS` column for your input and output datasets are different.
11. If you are planning to go straight on to the next tutorial to complete masking on sequential data, you can keep **Data Builder** open. Otherwise, click **File** > **Exit**, and then click **Yes**.

Sequential Files Session

In this session, you:

- Associate supplied classes to data elements.

- Start the Data Masking module.
- Enable masking on the sequential file CUSTOMER for NAME and SURNAME classes.
- Submit a JCL to mask customer first and last names.

Note: The sequential files session does not cover creating a masking class like the DB2 session does.

Associating Supplied Classes to Data Elements

In this exercise you will manually associate classes supplied with your installation with columns NAME and SURNAME in the table CUSTOMER.

1. In the **[Work with Data Stores]** window, ensure that the **Machine ID** node is expanded, and that DATAEXPRES is selected:
2. In the **List of Data Stores**, select the row for the URADAR.DEMOT.CUSTOMER data store.
3. Right-click and then select **Work with Data Elements of the Selected Data Store**.

   Note: If a message appears indicating that the default confirmation options have not been set, click **Yes** to set them. Click **Apply**, and then **OK** in the **Options** window.

   Note: You can also access the **Work with Data Elements** window by clicking or by accessing it through **Environment** > **Work with Data Elements**. However, if you choose to access the **Work with Data Elements** window directly, you may want to specify information in the fields above the **List of Data Elements** and then click **Apply Filter**.
4. Click the **Classes - Assignment** tab.
5. In the **All Classes** pane, expand the Privacy **Super Class**, and then select the class **NAME**.
6. In the **List of Data Elements**, select the row for CUSTOMER.NAME. (That is, select the row for the data store CUSTOMER and the name NAME.)

   Note: If you are viewing all data stores, you could click the **Data Store** column to sort data elements by data store.
7. Click the **Assign Class** button to assign the NAME class to CUSTOMER.NAME. NAME now appears in the **Assigned Class** column for CUSTOMER.NAME.
8. In the **List of Data Elements**, select the row for CUSTOMER.SURNAME. (That is, select the row for the data store CUSTOMER and the name SURNAME.)
9. Click the **Assign Class** button to assign the SURNAME class to CUSTOMER.SURNAME. SURNAME now appears in the Assigned Class column for CUSTOMER.SURNAME.
10. Click the **Close** button to close the **Work with Data Elements** window.
11. Leave **Data Builder** running, and continue to the next section.

Starting the Data Masking Module

1. Click **Start > All Programs > Micro Focus Data Express 4.0 > Data Masking**.
2. Click **OK**.
3. Select the mainframe Knowledge Base as the required database with which to launch the connection. In this example, the database is MVSDB2. By default, the highlighted item in the list of available databases is the last database you connected to using Data Masking. This becomes apparent after you click **Available databases**.
4. Click **OK**.
5. Log in using your mainframe user name and password.
6. Click **OK**.
7. Select the schema OWUR40SV, and click **OK**.

   The **Main Window** comprises four main areas:

- **Machine ID./Company** grid
- **All Machine ID./Companies**
- **Multiple Class Selection**
- **Object Distribution**

The area containing the grid and the tabs **Enabled** and **Disabled** is referred to as the **Machine ID./Company** area or the **Grid** area. The grid lists files related to the selected class and to the selected machine ID and company. The names of the forms, their meaning and the included information are outlined below.

The **Enabled** tab shows a list of all active files (enabling a file means that it is visible in the export functionality in order to be prepared for masking), and all files, which have one or more specific classes related to their data elements.

The **Object Distribution** area displays a pie chart and bar chart representation of the file types for the classes selected in the **Machine ID./Companies** or **Multiple Class Section** area. The pie chart shows the percentage distribution of file types, while the bar chart shows the numerical distribution of file types. For both charts, both enabled and disabled files are included. Click the minimize button to minimize the **Object Distribution** area.

8. Continue to the next section.

Enabling Masking on the Sequential File CUSTOMER and NAME and SURNAME classes

Data Masking lets you enable masking for individual data stores or all data stores, and to enable masking at the data-element level or class level.

In this exercise you will be masking at the class level by activating masking for the data store `CUSTOMER` at the class (`NAME` and `SURNAME`) level.

Note: To mask at the data-element level, double-click the desired data store from the **Main Window**, right-click the desired data element, and then click **Data Changer Data Element Properties**. Click **Change**, click **Yes** to activate the **Changer** option, and finally click **Apply** to activate your setting.

1. In **Data Masking**, click the **ALL Machine ID./Companies** list.
2. Expand the **All Machine ID./Companies** node
3. Double-click the `DATAEXPRES` **machine ID**.
4. Select the row for the data store `URADAR.DEMOT.CUSTOMER`.
5. Right-click and select **Enable**.
6. Click the **Enabled** tab, and then right-click and select **Data Changer Properties**.
7. Verify `URADAR.DEMOT.CUSTOMER` is set as the **Input Unload** data store name and specify `URADAR.DEMOT.MASK.new_library_name` as the **Output Unload** data store name. For example: `URADAR.DEMOT.MASK.SEQCUST`.
8. Click **OK**.
9. Click the **Enabled** tab.
10. Select the data store `URADAR.DEMOT.CUSTOMER`, right-click, and then click **Open Selected Data Store**.
11. Expand the row for the `SURNAME` node.
12. Click the `SURNAME` class, right-click, and then click **Enable**.
13. Expand the row for the `NAME` node.
14. Click the `NAME` class, right-click, and then click **Enable**.
15. Continue to the next section.

Submitting a JCL to Mask Customer First and Last Names

Use the **Work with Jobs** tool in **Data Builder** to submit the JCL that does the actual masking on the CUSTOMER table. Then, submit the job in the mainframe environment and verify your results.

1. In **Data Builder**, open the **Work with Jobs** window (click **Environment** > **Work with Jobs**) to schedule a new job.
2. Click **New**.
3. In the **List of Jobs**, select the job BDCCHGR (Data Store Data Changer).
4. Select the DATAEXPRES **Machine ID** and the DEMO **company name**.
5. In the **Data store name** box, type: *CUSTOMER*.
6. Click **Apply**. The job appears in the **List of Scheduled Jobs**. The pause icon indicates that the job is awaiting execution. Once the job has completed successfully, the job is listed with a green check mark.
7. In your TSO environment, submit the BDCCHGR job by using the **Submit Client Scheduled Job** panel.
8. In the **Work with Jobs** area, click the refresh button to verify that the jobs completed successfully.
9. Click **OK** to close the **Work with Jobs** window.
10. In your mainframe environment, notice that the values in the NAME and SURNAME columns for your input and output datasets are different.

Simple Data Subset Extraction

The **Data Subset Extraction** module is used to generate a reduced test environment. In this tutorial, you perform basic data subset extraction.

Note: Sessions in this tutorial should be done in order.

This tutorial walks you through the data classification and the data subset extraction phases. This session involves creating a simple subsetting class.

Begin this session only after you have set your workspace and loaded data store information into the Knowledge Base in tutorials *Using Data Express* and *Data Inventory*. You must already be logged in to **Data Builder** to begin.

The sample sessions are broken out into two parts: one for DB2 data stores and one for sequential files.

Note: Sessions in this tutorial must be done in order. If you choose to complete the tutorial for both DB2 and sequential data, then do all sessions. If you choose to complete the tutorial for only DB2 or sequential data, then follow the sessions only for that data type.

DB2 Data Stores Session

In this session, you:

- Associate a class to a data element.
- Start the Data Subset Extraction module.
- Create a group.
- Create a simple method within that group.
- Submit a JCL to subset customer names.

Associating a Class to a Data Element

In this exercise, you will be manually associating the NAME predefined class supplied with your installation with the column NAME in the table CUSTOMER.

Note: Similarly, if you performed the masking tutorial for DB2 data stores, then you already have the correct class associations for your data elements; you can skip the instructions in this section. If you have not performed the masking tutorial, then you cannot skip this section.

1. In the **[Work with Data Stores]** window, ensure that the **Machine ID** node is expanded, and that DATAEXPRES is selected.
2. In the **List of Data Stores**, select the row for the DELGT.CUSTOMER data store.
3. Right-click and then select **Work with Data Elements of the Selected Data Store**.

   Note: If a message appears indicating that the default confirmation options have not been set, click **Yes** to set them. Click **Apply**, and then **OK** in the **Options** window.

   Note: You can also access the **Work with Data Elements** window by clicking or by accessing it through **Environment** > **Work with Data Elements**. However, if you choose to access the **Work with Data Elements** window directly, you may want to specify information in the fields above the **List of Data Elements** and then click **Apply Filter**.
4. Click the **Classes - Assignment** tab.

   Note: This screen capture shows a filtered list based on **Machine ID** and **Company Name**.
5. In the All Classes pane, expand the Privacy Super Class, and then select the class NAME.
6. In the List of Data Elements, select the row for CUSTOMER.NAME. (That is, select the row for the data store CUSTOMER and the name NAME.)

   Note: If you are viewing all data stores, you could click the **Data Store** column to sort data elements by data store.
7. Click the **Assign Class** button to assign the **NAME** class to **CUSTOMER.NAME**. **NAME** now appears in the **Assigned Class** column for CUSTOMER.NAME.
8. Click the Close button to close the Work with Data Elements window.
9. Leave Data Builder running, and continue to the next section.

Starting Data Subset Extraction

Enter the context of your task here (optional).

1. Click**Start > All Programs > Micro Focus Data Express 4.0 > Data Subset Extraction**.

   Note: If you are using Data Express on Windows Vista and User Access Control is enabled, you must run **Data Subset Extraction** as *Administrator*.
2. Select the **mainframe Knowledge Base** as the required database with which to launch the connection. In this example, the database is MVSDB2. By default, the highlighted item in the list of available databases is the last database you connected to using **Data Masking**. This becomes apparent after you click **Available databases**.
3. Click **OK**.
4. Log in using your mainframe username and password.
5. Click **OK**.
6. Select the schema OWUR40SV, and click **OK**.

   The **Data Subset Extraction** window comprises two main areas:

   - The node pane.
   - The grid area.

   The node pane contains a hierarchical structure that shows the grouping associated with each root-level node. The two root-level nodes are **Groups** and **Creators**.
7. Continue to the next section.

Creating a Group

The **Work with Groups** window lets you define, view, modify, and delete groups.

Groups can be used to logically separate the methods created for each workspace. Each method must belong to a group.

1. Open the **Work with Groups** window (**Environment** > **Work with Groups**).
2. Click **New**.
3. In the **Machine ID** list, click `DATAEXPRES`.
4. In the **Company** list, click `DEMO`.
5. In the **Group name** box, specify `SIMPLESUB`.
6. In the **Group description** box, type `GROUP FOR SIMPLE SUBSET EXERCISE`.
7. Click **Apply**.
8. Click **OK**.
9. Continue to the next section.

Creating a Simple Method within a Group

The **New Method Wizard** is used to create a new method, which is the extraction proposal containing the set of operations needed for the creation of the test environment. During method creation, all the information concerning the method is saved locally in a directory indicated by the user

Reduction can be obtained for data stores of a certain size and/or a particular type. In this case, you must be knowledgeable about the application to determine the appropriate selections. However, the **Data Subset Extraction** module can recognize and automatically create a first subset containing a series of files (called register table files) that are initially brought into the new test environment. Identifying register table files has the following advantages:

- You can create a method to extract only the register table file subset, which is normally valid for the creation of any test environment independently of the selection criteria applied. Once this method has been saved and confirmed, it can be run whenever necessary.
- You can create a second file subset, which filters the register file subset you created, to produce a file subset that is reduced even further.

To create a simple method:

1. Launch the **New Method Wizard** (**Environment** > **Create New Method**).
2. In the **Machine ID** list, click `DATAEXPRES`.
3. In the **Company** list, click `DEMO`. Files belonging to different workspaces cannot be included.
4. Click **Next** to go to the next screen.

   Note: If a group has not previously been defined in the **Work with Groups** area, it can be defined from this screen, by simply specifying its name and description in a specific field.
5. In the **Group name** box, click `SIMPLESUB.Text` in the **Group description** box populates automatically.
6. In the **Method** box, type `SIMPLEDB2`.
7. In the Method description box, type: `SIMPLE DB2 METHOD`.
8. Click **Next**.
9. Select **Import all non-register table Data Stores** to include demographic files are registered in a dedicated elaboration step of the method.
10. Click to clear the **Import all register table Data Stores** box.
11. Click **Next** to go the next screen.
12. Select NAME to be used as the primary extraction criteria.

13. Click **Next**.

    The **Data Subset Extraction** module automatically extracts all the environment files containing the selected classes, entering them into a specific step. Once the method has been created, the remaining files (that is the files that are not sensitive to the selected classes) will initially be excluded from elaboration. During the *Method Confirmation* phase, they can be included in the method again. These files will then be integrally copied into the new environment generated by the method involved.

    The primary extraction criteria are listed in the **Selection class** field.

14. For this tutorial, specifying a selection class in the Other Class is not necessary because you are creating a method for only one data store. You do not need another class. Note that both the DB2 and sequential data stores used throughout these tutorials both use the class NAME. Click Finish.
15. Click **Yes**.

    The **Data Subset Extraction** module automatically extracts all the environment files containing the related classes and inserts them in a further elaboration step.

    The **Create Method** window is displayed:

16. Click **Start**.
17. Click **OK**.
18. Click **Confirm**.

    During method confirmation, specified filters are applied to the classes needed to perform the extraction, and the method information is imported from a temporary file to the Knowledge Base.

    In this example, we will simply create a method based on the data stores that use the class NAME.

19. In the **Prefix** box, type: `URADAR.DEMOT.SUB.DB2CUST`.
20. Click to clear the **Move excluded data stores** into new step box.
21. Click **OK**.
22. In the **Choose a Filter Type** list, click `FILTER BY VALUE LIST`.
23. In one of the boxes on the lower left side, type: `Rick`.
24. Click **OK** to close the **Set Filter to Selection Classes** window.
25. In the **Unload Output Data Store Name** box, type: `URADAR.DEMOT.SUB.DB2CUST`
26. Click **OK** to confirm the method.
27. In the **Work with Method** window, click **Properties**:
28. Click **Active**.
29. Click **OK**.
30. Click **OK**.
31. Continue to the next section.

Submitting a JCL to Subset Customer Names

Use the **Work with Jobs** tool in **Data Builder** to submit the JCL that does the actual subsetting on the `CUSTOMER` table. Then, submit the job in the mainframe environment and verify your results.

1. In **Data Builder**, open the **Work with Jobs** window (**Environment** > **Work with Jobs**) to schedule a new job.
2. Click **New**.
3. In the **List of Jobs**, select the job `BTESBMR` (test environment creation).
4. Select the `DATAEXPRES` **Machine ID** and the **DEMO** company name.
5. In the **Data store name** box, type: `*CUSTOMER*`.
6. Click **Apply**.
7. In the **Group** list, click `SIMPLESUB`.

8. In the **Method** list, click **SIMPLE**.
9. Click **OK**. The job appears in the **List of Scheduled Jobs**. The pause symbol indicates that the job is awaiting execution. Once the job has completed successfully, the job is listed with a green check mark.
10. In your TSO environment, submit the `BTESBMR` job by using the **Submit Client Scheduled Job** panel.
11. In the **Work with Jobs** area, click the **Refresh** button to verify that the jobs completed successfully.
12. Click **OK** or the button to close the **Work with Jobs** window.
13. In your mainframe environment, notice that only information for the value `Rick` in the **NAME** column for your output dataset is listed.

    Note: Because the **Name** class is associated to a masking routine, the data was masked as well as reduced.
14. If you are planning to go straight on to the next tutorial to complete subsetting on sequential data, you can keep **Data Builder** open. Otherwise, either click **File** > **Exit** and then click **Yes**.

Sequential Files Session

In this session, you:

- Associate a class to a data element.
- Start the Data Subset Extraction module.
- Create a group.
- Create a simple method within that group.
- Submit a JCL to subset customer names.

Associating a Class to a Data Element

In this exercise, you will be manually associating the `NAME` predefined class supplied with your installation with the column `NAME` in the table `CUSTOMER`.

Note: Similarly, if you performed the masking tutorial for DB2 data stores, then you already have the correct class associations for your data elements; you can skip the instructions in this section. If you have not performed the masking tutorial, then you cannot skip this section.

1. In the **[Work with Data Stores]** window, ensure that the **Machine ID** node is expanded, and that `DATAEXPRES` is selected.
2. In the **List of Data Stores**, select the row for the `DELGT.CUSTOMER` data store.
3. Right-click and then select **Work with Data Elements of the Selected Data Store**.

   Note: If a message appears indicating that the default confirmation options have not been set, click **Yes** to set them. Click **Apply**, and then **OK** in the **Options** window.

   Note: You can also access the **Work with Data Elements** window by clicking or by accessing it through **Environment** > **Work with Data Elements**. However, if you choose to access the **Work with Data Elements** window directly, you may want to specify information in the fields above the **List of Data Elements** and then click **Apply Filter**.
4. Click the **Classes - Assignment** tab.

   Note: This screen capture shows a filtered list based on **Machine ID** and **Company Name**.
5. In the All Classes pane, expand the Privacy Super Class, and then select the class NAME.
6. In the List of Data Elements, select the row for CUSTOMER.NAME. (That is, select the row for the data store CUSTOMER and the name NAME.)

   Note: If you are viewing all data stores, you could click the **Data Store** column to sort data elements by data store.
7. Click the **Assign Class** button to assign the **NAME** class to **CUSTOMER.NAME**. **NAME** now appears in the **Assigned Class** column for `CUSTOMER.NAME`.

8. Click the Close button to close the Work with Data Elements window.
9. Leave Data Builder running, and continue to the next section.

Starting Data Subset Extraction

Enter the context of your task here (optional).

1. Click**Start > All Programs > Micro Focus Data Express 4.0 > Data Subset Extraction**.

   Note: If you are using Data Express on Windows Vista and User Access Control is enabled, you must run **Data Subset Extraction** as *Administrator*.
2. Select the **mainframe Knowledge Base** as the required database with which to launch the connection. In this example, the database is `MVSDB2`. By default, the highlighted item in the list of available databases is the last database you connected to using **Data Masking**. This becomes apparent after you click **Available databases**.
3. Click **OK**.
4. Log in using your mainframe username and password.
5. Click **OK**.
6. Select the schema `OWUR40SV`, and click **OK**.

   The **Data Subset Extraction** window comprises two main areas:

   - The node pane.
   - The grid area.

   The node pane contains a hierarchical structure that shows the grouping associated with each root-level node. The two root-level nodes are **Groups** and **Creators**.
7. Continue to the next section.

Creating a Group

The **Work with Groups** window lets you define, view, modify, and delete groups.

Groups can be used to logically separate the methods created for each workspace. Each method must belong to a group.

1. Open the **Work with Groups** window (**Environment** > **Work with Groups**).
2. Click **New**.
3. In the **Machine ID** list, click `DATAEXPRES`.
4. In the **Company** list, click `DEMO`.
5. In the **Group name** box, specify `SIMPLESUB`.
6. In the **Group description** box, type `GROUP FOR SIMPLE SUBSET EXERCISE`.
7. Click **Apply**.
8. Click **OK**.
9. Continue to the next section.

Creating a Simple Method within a Group

The **New Method Wizard** is used to create a new method, which is the extraction proposal containing the set of operations needed for the creation of the test environment. During method creation, all the information concerning the method is saved locally in a directory indicated by the user

Reduction can be obtained for data stores of a certain size and/or a particular type. In this case, you must be knowledgeable about the application to determine the appropriate selections. However, the **Data Subset Extraction** module can recognize and automatically create a first subset containing a series of files (called register table files) that are initially brought into the new test environment. Identifying register table files has the following advantages:

- You can create a method to extract only the register table file subset, which is normally valid for the creation of any test environment independently of the selection criteria applied. Once this method has been saved and confirmed, it can be run whenever necessary.
- You can create a second file subset, which filters the register file subset you created, to produce a file subset that is reduced even further.

To create a simple method:

1. Launch the **New Method Wizard** (**Environment** > **Create New Method**).
2. In the **Machine ID** list, click `DATAEXPRES`.
3. In the **Company** list, click `DEMO`. Files belonging to different workspaces cannot be included.
4. Click **Next** to go to the next screen.

   Note: If a group has not previously been defined in the **Work with Groups** area, it can be defined from this screen, by simply specifying its name and description in a specific field.
5. In the **Group name** box, click `SIMPLESUB.Text` in the **Group description** box populates automatically.
6. In the **Method** box, type `SIMPLEDB2`.
7. In the Method description box, type: `SIMPLE DB2 METHOD`.
8. Click **Next**.
9. Select **Import all non-register table Data Stores** to include demographic files are registered in a dedicated elaboration step of the method.
10. Click to clear the **Import all register table Data Stores** box.
11. Click **Next** to go the next screen.
12. Select NAME to be used as the primary extraction criteria.
13. Click **Next**.

    The **Data Subset Extraction** module automatically extracts all the environment files containing the selected classes, entering them into a specific step. Once the method has been created, the remaining files (that is the files that are not sensitive to the selected classes) will initially be excluded from elaboration. During the *Method Confirmation* phase, they can be included in the method again. These files will then be integrally copied into the new environment generated by the method involved.

    The primary extraction criteria are listed in the **Selection class** field.
14. For this tutorial, specifying a selection class in the Other Class is not necessary because you are creating a method for only one data store. You do not need another class. Note that both the DB2 and sequential data stores used throughout these tutorials both use the class NAME. Click Finish.
15. Click **Yes**.

    The **Data Subset Extraction** module automatically extracts all the environment files containing the related classes and inserts them in a further elaboration step.

    The **Create Method** window is displayed:
16. Click **Start**.
17. Click **OK**.
18. Click **Confirm**.

    During method confirmation, specified filters are applied to the classes needed to perform the extraction, and the method information is imported from a temporary file to the Knowledge Base.

    In this example, we will simply create a method based on the data stores that use the class NAME.
19. In the **Prefix** box, type: `URADAR.DEMOT.SUB.DB2CUST`.
20. Click to clear the **Move excluded data stores** into new step box.
21. Click **OK**.

22. In the **Choose a Filter Type** list, click `FILTER BY VALUE LIST`.
23. In one of the boxes on the lower left side, type: `Rick`.
24. Click **OK** to close the **Set Filter to Selection Classes** window.
25. In the **Unload Output Data Store Name** box, type: `URADAR.DEMOT.SUB.DB2CUST`
26. Click **OK** to confirm the method.
27. In the **Work with Method** window, click **Properties**:
28. Click **Active**.
29. Click **OK**.
30. Click **OK**.
31. Continue to the next section.

Submitting a JCL to Subset Customer Names

Use the **Work with Jobs** tool in **Data Builder** to submit the JCL that does the actual subsetting on the `CUSTOMER` table. Then, submit the job in the mainframe environment and verify your results.

1. In **Data Builder**, open the **Work with Jobs** window (**Environment** > **Work with Jobs**) to schedule a new job.
2. Click **New**.
3. In the **List of Jobs**, select the job `BTESBMR` (test environment creation).
4. Select the `DATAEXPRES` **Machine ID** and the **DEMO** company name.
5. In the **Data store name** box, type: `*CUSTOMER*`.
6. Click **Apply**.
7. In the **Group** list, click `SIMPLESUB`.
8. In the **Method** list, click **SIMPLE**.
9. Click **OK**. The job appears in the **List of Scheduled Jobs**. The pause symbol indicates that the job is awaiting execution. Once the job has completed successfully, the job is listed with a green check mark.
10. In your TSO environment, submit the `BTESBMR` job by using the **Submit Client Scheduled Job** panel.
11. In the **Work with Jobs** area, click the **Refresh** button to verify that the jobs completed successfully.
12. Click **OK** or the button to close the **Work with Jobs** window.
13. In your mainframe environment, notice that only information for the value `Rick` in the **NAME** column for your output dataset is listed.

    Note: Because the **Name** class is associated to a masking routine, the data was masked as well as reduced.
14. If you are planning to go straight on to the next tutorial to complete subsetting on sequential data, you can keep **Data Builder** open. Otherwise, either click **File** > **Exit** and then click **Yes**.

Data Subset Extraction using Referential Integrity and Combined Data Elements

In this tutorial, you import class information using referential integrity for a DB2 data store. Then you create a method `RIMETHOD` to use a *combined data element*.

A combined data element is essentially a *dummy* data element that you create for ease of use when it comes to manipulating classes. Once a combined data element is created, you can perform operations against the data element like you would for other elements.

Note: Sessions in this tutorial should be done in order.

This tutorial walks you through the data classification and the data subsetting phases. In this session, subsetting is based on referential integrity rules.

Begin this session only after you have completed the *DB2 Data Stores Session* for the tutorial *Simple Data Subset Extraction*. You must already be logged in to Data Builder to begin.

In this session, you:

- Import referential integrity classes.
- Assign estimated classes.
- Verify combined data element information.
- Create a referential integrity method.
- Edit method properties.
- Submit a JCL to create the test environment.

Creating the HSURDRIA table

The ability to import referential integrity class is based on the **ACTIVATION** field in the **HSURDRIA** table located in the Data Express knowledge base. During the z/OS Host installation this table is not created in the knowledge base. It must be created manually in case the import type **Referential Integrity** in the **Import Class** window will be used. This way of handling automatic referential integrity has been chosen in order to prevent uncontrolled access to z-OS knowledge base from user interface, because the referential integrity process is a time consuming one and it can affect the z-OS performances.

1. Ensure administrative privileges for the DB2 userid.
2. Create the table **HSURDRIA** using the DDL below:

```
CREATE TABLESPACE      HSURDRIA
       IN               &URDBASE
       USING STOGROUP   &URSGROUP
       PRIQTY          12
       SECQTY          12
       FREEPAGE        0
       PCTFREE         0
       BUFFERPOOL         &URBPTS
       SEGSIZE         64
       COMPRESS           NO;

CREATE TABLE &UROWNER.HSURDRIA
       FREEPAGE    0
       PCTFREE     0
       BUFFERPOOL    &URBPTS
       SEGSIZE     64
       COMPRESS      NO;

CREATE TABLE  &UROWNER.HSURDRIA
   (ACTIVATION CHAR(1) NOT NULL WITH DEFAULT)
 IN  &URDBASE..HSURDRIA;

CREATE UNIQUE INDEX &UROWNER.HSURDRIAX1 ON &UROWNER..HSURDRIA
 (ACTIVATION)
        USING STOGROUP &URSGROUP
        PRIQTY 12
        SECQTY 12
        CLUSTER
        BUFFERPOOL  &URBPIX;
```

&URDBASE	Name of the DB2 database where the product tables are created.
&URSGROUP	Name of the DB2 storage group.
&UROWNER	Name of the DB2 creator that is used in order to create all the product tables.

&URBPTS	Name of the buffer pool containing the data of the product tables.
&URBPIX	Name of the buffer pool containing the indexes of the product tables.

3. To activate the import type **Referential Integrity**, insert a value Y in the field **ACTIVATION** using the command: `INSERT INTO &UROWNER.HSURDRIA (ACTIVATION) VALUES ('Y')`
4. After restarting Data Builder you should see import type **Referential Integrity** active.
5. To deactivate the import type **Referential Integrity** perform the command below and then restart the Data Builder: `DELETE FROM &UROWNER.HSURDRIA WHERE ACTIVATION = 'Y'`

Importing Referential Integrity Classes

In this exercise you import referential integrity classes by running the BURDDUR (Import Classification From Referential Integrity) job, which creates a referential integrity class and combined data elements, and creates an estimated class assignment for those data elements.

Note: You can also use the Import Class window to import referential integrity classes. However, this mechanism is designed to be used with the New Method Wizard to create a simple method with referential integrity that supports two-level relationships between data stores.

1. In **Data Builder**, open the **Work with Jobs** window (click or access it through **Environment** > **Work with Jobs**).
2. Click **New**.
3. In **List of Jobs**, select the job BURDDUR (Import Classification From Referential Integrity).
4. Select the DATAEXPRES **Machine ID**.
5. Specify **DEMO** as the name of your company code in the **Company name** box.
6. Click **Apply**.
7. In the **Referential Integrity Interface** box, type: `URADAR.DEMOT.REFINTEG 8`. Click **OK**.

   The job appears in the **List of Scheduled Jobs**. The pause symbol indicates that the job is awaiting execution. Once the job has completed successfully, the job is listed with a green check mark.
8. In your TSO environment, submit the BURDDUR job by using the **Submit Client Scheduled Job** panel.
9. In the **Work with Jobs** area, click the **Refresh** button to verify that the jobs completed successfully.
10. Close the **Work with Jobs** window.
11. Continue to the next section.

Assigning Estimated Classes

In this exercise, you assign estimated classes to the combined data elements created by the BURDDUR job and to the data elements that compose those combined data elements.

1. Open the Work with Data Elements window (**Environment** > **Work with Data Elements**).

   Note: You can also access the **Work with Data Elements** window from the main window by selecting the file name, right-clicking to view the context menu, and then by clicking **Work with Data Elements of the Selected Data Store**.
2. From the **Select Data Elements** tab, specify DATAEXPRES as the **Machine ID** and DEMO as the **Company name**.
3. Click the **Data Store Attributes** tab.
4. In the **DB2 Owner, DL/I DBD** list, click **DELGT**.
5. Click **Apply Filter** to further narrow down the list of data elements displayed.
6. In the **List of Data Elements (Filtered)** table, view the column **Estimated Class**.

   The BURDDUR job you ran has created four combined data elements and has automatically estimated class assignments. These assignments are estimated for the newly created combined data elements and for the data elements that compose these combined data elements. (Likewise, if a data element

that composes a combined data element already has an assigned class, the data element is not associated with an estimated class.)

Note: When you performed the *Data Masking* exercise, the class CODCUS (displayed by its class description COD_CUS MASK EXAMPLE) was assigned to the COD_CUS data element in the CUSTOMER data store. Then, when you ran the BURDDUR job as part of this exercise, the class CODUS became an estimated class assignment for the COD_CUS data element in the ACCOUNT data store.

7. In the **List of Data Elements (Filtered)**, view the newly assigned estimated classes created for the following data stores:

ACCOUNT and CCARD data stores	• **CLA0449** - created for the data element OFF_NUM. • **CLA0450** - created for the data element ACC_NUM. • **CLA0451** - created for the combined data element, which is comprised of both OFF_NUM and ACC_NUM data elements.
CCARD and OPERAT data stores	• **CLA0452** - created for the data element CARD_TYPE • **CLA0453** - created for the data element CARD_NUM • **CLA0454** - created for the combined data element, which is comprised of both CARD_TYPE and CARD_NUM

Note: The actual names of your estimated classes will differ as they are system-generated.

8. Click the **Estimated Classes - Confirm** tab
9. In the **List of Data Elements (Filtered)**, select each row that has an estimated class but not an assigned class.

Note: Not all selected rows are displayed.

10. Click **Confirm Class**.

    The **Estimated Class** names now appear in the **Assigned Class** column.
11. Close the **Work with Data Elements** window to return to the **Work with Data Stores** window.
12. Continue to the next section.

Verifying Combined Data Element Information

In this exercise, you will view and edit combined data element properties.

1. From the **Work with Data Stores** window, select a data store that you know has a combined data element. (For this exercise, you can select DELGT. ACCOUNT, DELGT.CCARD, or DELGT.OPERAT.)
2. Right-click your selection, and choose **Open Selected Data Store**. For this demonstration, the data store DELGT.ACCOUNT was chosen.
3. In the **List of Data Elements**, select the row for the combined data element, right-click, and then click **Work with Combined Data Element**.
4. View properties of the system-generated combined data element. Change the **Combined Data Element Attributes** if desired.
5. Click **Apply** or **Close** as appropriate.
6. Close the **Work With Data Store** window.
7. Continue to the next section.

Creating a Referential Integrity Method

In this exercise you will create a referential integrity method by running the BTEDDUR (Import Method From Referential Integrity) job.

Note:

- After class assignments have been established, the BTEDDUR job is used to create a complex method with referential integrity that supports multiple-level relationships between data stores.
- You can also use the **New Method Wizard** to create a simple method with referential integrity, but the wizard is limited in that it only supports two-level relationships between data stores.
- The method created by the BTEDDUR is automatically activated during job execution. Therefore, you do not need to activate the method in the **Method Properties** window.

1. 1. In **Data Builder**, open the **Work with Jobs** window (**Environment** > **Work with Jobs**).
2. Click **New**.
3. In **List of Jobs**, select the job BTEDDUR (Import Method From Referential Integrity).
4. Select the DATAEXPRES **Machine ID**.
5. Specify DEMO as the name of your company code in the **Company name** box.
6. Click **Apply**.
7. In the **Referential Integrity Interface** box, type: URADAR.DEMOT.REFINTEG.
8. In the **Group** list, click SIMPLESUB.

   Note: You created the SIMPLESUB group during the Simple Subset Extraction exercise for DB2 data stores.
9. In the Method field, type: RIMETHOD.

   Note: The method RIMETHOD is created when the job runs.
10. In the Method text box, type: METHOD FOR RI AND COMBINED DATA ELEMENT.
11. In the **Class** list, click COD_CUS MASK EXAMPLE.

    Note: You created the class COD_CUS MASK EXAMPLE during the Data Masking exercise.
12. Click **OK**.

Editing Method Properties

In this exercise, you will edit the properties of the referential integrity method you just created.

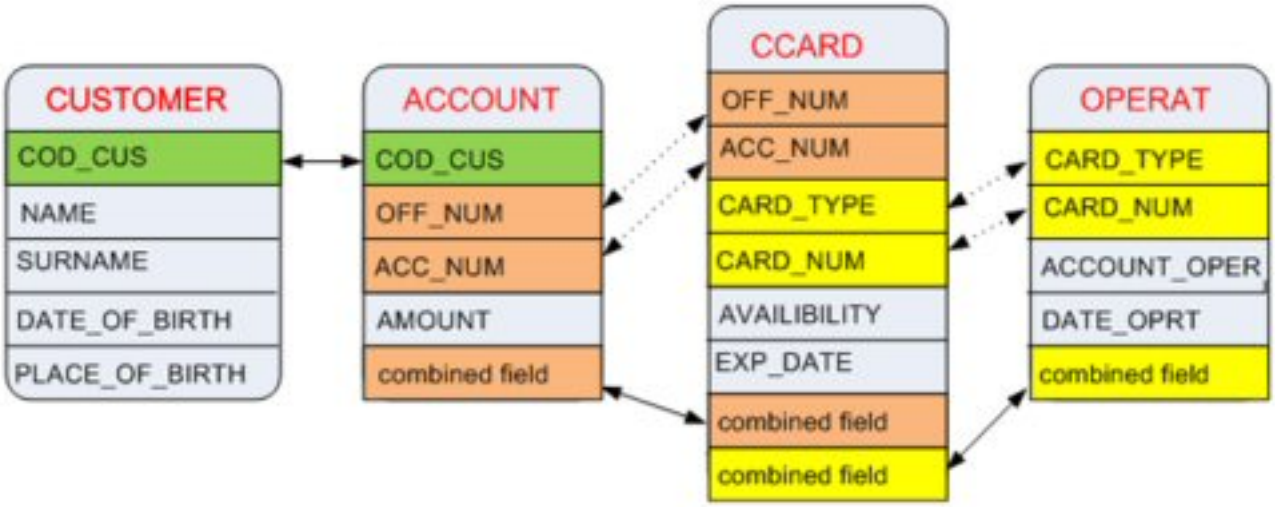
1. Launch **Data Subset Extraction**.
2. In the left pane, expand the **Group** list, expand the **SIMPLESUB** list, and then select Methods.
3. Select the row for the method RIMETHOD, right-click, and then choose **Work with selected Method**.

   The table information for all data stores has been exported because all data stores are related.

   Notice that the data stores DELGT.CUSTOMER and DELGT.ACCOUNT are elaborated in the step, and that the output class for the DELGT.ACCOUNT data store is the input class for DELGT.CCARD in the subsequent step. Likewise, the output class DELGT.CCARD is the input class for DELGT.OPERAT in the subsequent step.

   Elaborating the data stores in this way preserves referential integrity and shows that the DELGT.CUSTOMER data stores is indirectly related to the DELGT.OPERAT data store.

   The following diagram further demonstrations the data store relations:



Note: The dashed line represents a relationship that exists between data stores because of the combined field.

4. In the **Data Stores Elaboration** list, select a row for a `DELGT.DataStoreName` data store.
5. Right-click and select **Elaboration Properties**
6. In the **Unload Output Data Store Properties** section, in the **Data store name** box, type: `URADAR.DEMOT.SUB2.DataStoreName`. For example, type: `URADAR.DEMOT.SUB2.CUSTOMER`.
7. Click **OK**.
8. Repeat steps 4-7 for each `DELGT.DataStoreName` data store.
9. In the **Data Stores Elaboration** list of the **Work with Method** window, select the row for the `DELGT.CUSTOMER` data store.
10. Right-click, and then select **Elaboration** > **Selection class / Filter properties**.
11. Choose the filter type **FILTER BY LIST VALUE**.
12. In one of the boxes in the lower-left corner, type the following `CUSTOMER CODE: 0000000001`

Note: You can verify the customer code in your z/OS DB2 window. However, this customer code is specified in an `INSERT` statement in the `CREATBT.TXT` file used to create your DB2 tables.

13. Click **OK** to close the **Set Filter to Selection Classes** window.

Note: The filter only needs to be specified for the `CUSTOMER` data store because the `BTEDDUR` job establishes the relations between all of the other data stores.

14. Exit **Data Subset Extraction** by clicking **File** > **Exit**.
15. Continue to the next section.

Submitting a JCL to Create the Test Environment

Use the **Work with Jobs** tool in **Data Builder** to submit the JCL that does the actual subsetting on the `CUSTOMER`, `ACCOUNT`, `CCARD`, and `OPERAT` data stores. Then, submit the job in the mainframe environment and verify your results.

1. 1. In **Data Builder**, open the **Work with Jobs** window (**Environment** > **Work with Jobs**) to schedule a new job.
2. Click **New**.
3. In the List of Jobs, select the job `BTESBMR` (Test Environment Creation).
4. Select the DATAEXPRES Machine ID and the DEMO company name.
5. Click **Apply**.

6. In the **Group** list, click SIMPLESUB.
7. In the **Method** list, click RIMETHOD.
8. Click **OK**. The job appears in the **List of Scheduled Jobs**. The **Pause** symbol indicates that the job is awaiting execution. Once the job has completed successfully, the job is listed with a green check mark.
9. In your TSO environment, submit the BTESBMR job by using the **Submit Client Scheduled Job** panel.
10. In the **Work with Jobs** area, click **Refresh** to verify that the jobs completed successfully.
11. Click **OK**.
12. In your mainframe environment, verify the subset extraction results by viewing the sequential files created by the BTESBMR job:
    - URADAR.DEMOT.SUB2.CUSTOMER
    - URADAR.DEMOT.SUB2.ACCOUNT
    - URADAR.DEMOT.SUB2.CCARD
    - URADAR.DEMOT.SUB2.OPERAT

    Note: Because the COD_CUS MASK EXAMPLE class is associated to a masking routine, the data was masked as well as reduced.
13. Exit **Data Builder**.

Data Element Sampling

This tutorial walks you through the data sampling phases. In this tutorial, you perform sampling on sequential data stores to obtain information about the distribution of data. Then you use that distribution as a way to assign classes to other data elements with similar data distributions.

Note: Sessions in this tutorial should be done in order.

Complete this session only after you have completed the Configuring your z/OS Data Express Environment, Data Masking, and the Simple Data Subset Extraction tutorials.

You must already be logged in to **Data Builder** to begin. In this session, you:

- Submit a JCL to perform sampling
- Verify results
- Create a sampling class
- Associate a fingerprint to a class
- Associate a class to data elements

Submitting a JCL to Perform Sampling

Use the Work with Jobs tool in Data Builder to submit the JCL that performs the actual sampling.

1. From **Data Builder**, in the **List of Data Stores** in the **Work with Data Stores** window, expand the DATA EXPRES **Machine ID** and select the DEMO company.
2. Select a data store with the **Machine ID** DATA EXPRES and the **company** DEMO, right-click, and then click Data Store Properties.

   Note that the Classification number is 1. By default all data stores are assigned a Classification number of 1, which means that sampling is enabled. Classification numbers are used to restrict the data stores that get sampled and to and to provide further granularity in restricting which data stores get sampled as based on user-defined criteria. When sampling the data stores within your work environment, all enabled data stores with Classification number values of 1 will be sampled. To disable sampling, change the Classification number to 0 in the Properties – File window or right-click the row for the data store in the Work with Data Stores and click Disable Sampling.

   The data store remains enabled as long as the Classification number is not 0. However, just because sampling is enabled does not mean that the data store is a candidate for sampling as this is controlled

by the Classification number specified in the Secondary Options for the job that submits the JCL. If the Classification number set for the data store is less than or equal to the Classification number specified in the Distributed Sampler, the data store will be sampled. Likewise, if the data store Classification number is greater than the Classification number in the Distributed Sampler, the data store will not be sampled.

Tip: You can assign classification numbers to your data stores based on how often you want to sample data. For example, 1 could represent daily sampling, 2 could represent weekly sampling, and 3 could represent monthly sampling.

3. Click **OK**.
4. Open the **Work with Jobs** window (**Environment** > **Work with Jobs**) to schedule a new job.
5. Click **New**.
6. In the **List of Jobs**, select the job `BURSAMR` (Data Store Data Element Sampling).
7. Select the `DATAEXPRES` **Machine ID** and the `DEMO` **company name**.
8. Click **Apply**.
9. Retain all default sampling values. For this exercise, all data stores should be sampling candidates and all sampling options should be utilized.

   The following list describes the sampling **Options**:

Compressed sampling	Produces the data element `fingerprint`. The fingerprint graphically shows the distribution of values within a given range for the sampled data element. The fingerprints for numeric and alphanumeric data elements differ in that the fingerprint for an alphanumeric data element shows the distribution of values based on the first character and provides additional information.
Standard sampling	Displays information for each data element value including the number of times each value occurred and the percentage that value represents in the total population.
Min/Max calculation	Displays the minimum and maximum values for the data element.

10. Click **Advanced**.
11. Retain all default values.

Note: The **Data Element Size** of `0` (zero) means to include all sizes in the sampling results and does not actually reflect the value of `0` when **Ignore Special Values Zero / Space** is checked. Likewise, the default `0` values in **Max. And Min. Recalculation Additional Options** mean to consider all values to be in-range and does not reflect the value of `0`.

12. Click **OK**.
13. In your TSO environment, submit the `BURSAMR` job by using the **Submit Client Scheduled Job** panel.
14. In the **Work with Jobs** area, click **Refresh** to verify that the job completed successfully.
15. Close the **Work with Jobs** window.
16. In your mainframe environment, verify the sampling results.
17. Continue to the next section.

Verifying Results

Once you have executed sampling, you can view the results.

When compressed sampling is performed, a `fingerprint` is created. The fingerprint is a unique graphical representation of the distribution of data for the sampled data element. When the results for compressed sampling are loaded, the fingerprint is created.

1. In the **List of Data Stores** in the **Work with Data Stores** window, select the row for the `URADAR.DEMOT.TableName` data store.

2. Click the **Show Synthetic Data Elements** button. The **Data Store Data Elements** list in the left pane shows the data elements for the data store you selected.
3. In the **Data Store Data Elements** list, select the data element `OPERAT-NUM`.

   You can also select the image for the data element in the **Data Element Samples** grid. To ensure that you have selected the correct one, the related data element name is highlighted in the **Data Store Data Elements** list.

   Sampling results for the numeric data element `OPERAT-NUM` are displayed.
4. Review the data element fingerprint in the **Zoomed Data Element Sample**. The fingerprint is the result of performing compressed sampling. Notice the following:

   - The numbers on the vertical y-axes represent the ranges of values based on the decimal place, where each number represents one decimal place to the left of the decimal point. For instance, 0 represents the ones place (0-9), 1 represents the tens place (10-99), 2 represents the hundreds place (100-999), and so on.
   - The numbers on the horizontal x-axes represent a specific range based on the y-coordinate corresponding to the first digit. For instance, 1 represents the value 1 for the y-coordinate 0, while 1 represents values 10-19 for the y-coordinate 1.
   - The values in each cell indicate the number of times a value in the data element falls within the specified range.

   Note: The fingerprint can also be viewed as a bar graph by clicking **Graph**.
5. Review the items in the list **Sample Analysis of the Selected Data Element**. This list shows the actual data element values when standard sampling was performed.

   Note: Only the first 1500 distinct values and ranges display. They are sorted either numerically or alphanumerically based on the data element type.
6. In the In the **Data Store Data Elements** list, select the data element **COD-CUS**. Sampling results for alphanumeric data element **COD-CUS** are displayed.
7. Review the data element fingerprint in the **Zoomed Data Element Sample**. The fingerprint is the result of performing compressed sampling. Notice that the fingerprint for an alphanumeric data element contains four distinct sections:

Section Type	Description	Example
Character Description	A range of characters in alphabetical order is provided to show how many data elements begin with that character.	Spa A B C D E F G H I J K L M N O P Q R S T U V W X Y Z *
Number Distribution	A range of numbers is provided to show how many data elements begin with the indicated number. In the provided example, no data elements begin with numbers.	0 1 2 3 4 5 6 7 8 9 4
Type Summary	An alphanumeric data element can actually be a numeric-only data element or alphanumeric data element. The type summary provides a count of how many data elements fall into either category. In the provided example, all data elements are alphanumeric in type.	Num Only Alpha Only 4

Section Type	Description	Example

Field Length	The numbers on the vertical y-axes represent the ranges of values specified. The numbers on the horizontal x-axes represent the position of the actual number in that range. For instance, 4 in the provided example means that there are 4 values with a length equal to 10.	`Field Len.  Val.`
		`0 1 2 3 4 5 6 7 8 9`
		
		` 0 -  9`
		`10 - 19  4`
		`20  -->`

Note: The fingerprint can also be viewed as a bar graph by clicking **Graph**.

8. Close the **Show Synthetic Data Element Contents** window to return to **Work with Data Stores**.
9. Open the **Work with Data Elements** window (**Environment** > **Work with Data Elements**).
10. In the **Machine ID** list, click `DATA EXPRES`.
11. In the **Company name** list, click `DEMO`.
12. Click **Apply Filter**.
13. Regard the **Minimum Value** and **Maximum Value** items in the **List of Data Elements**, which are the result of performing min/max calculation sampling.
14. Leave the **Work with Data Elements** window open, and continue to the next section.

Creating a Sampling Class

You can create a new class to be used for sampling purposes, or you can use a predefined class. In this exercise you will create a sampling class named `SAMPCUS`.

1. 1. Open the **Work with Classes** window (click or access it through **Environment** > **Work with Classes**).
2. Click **New**.
3. In the **Name** box, specify `SAMPCUS` as the name of the class you want to create.
4. In the Full description field, specify `SAMPLING CUSTOMER VALUES` as the description of the class.

   Note: The description will be visible as a label when you assign the class to data elements within data stores.
5. Select the **Super Class Generated**.
6. Check each data type box.
7. Leave the remainder of the fields at their default values. Click **Apply**. The class `SAMPCUS` is now listed in the **List of Classes**.
8. Click **OK** to return to the **Work with Data Elements** window.
9. Click **Refresh**.
10. Continue to the next section.

Associating a Fingerprint to a Class

In Data Express, you can correlate data elements if their value distributions are similar. To do this, you must first associate a fingerprint, which represents the desired distribution, to a class.

This fingerprint becomes the prototype that is used determine the class assignment for other data elements. If the prototype and the fingerprint for a data element are deemed similar based on a calculated confidence value, the data element is also associated with the prototype class.

1. Click the **Classes – Assignments** tab.

2. In the **All Classes** pane, expand the **Generated Super Class**, and select the class description SAMPLING CUSTOMER VALUES.
3. In the **List of Data Elements**, select the row for the data element COD-CUS in the data store URADAR.DEMOT.PHCONTRA.
4. Check the **Selected Data Element Attributes** check box
5. Drag the fingerprint that now appears beneath the text Selected Data Element Attributes, to the first Class Samples cell:

   By highlighting the SAMPCUS class description in the **All Classes** pane and then by dragging the fingerprint for the COD-CUS data element to the **Class Samples** cell, the SAMPCUS class is then assigned to the fingerprint. This fingerprint becomes the prototype. This action also associates the SAMPCUS class to the COD-CUS data element.
6. Continue to the next section.

Associating a Class to Data Elements

In this exercise you are associating the class SAMPCUS (which is associated to your sampling prototype) to other data elements with sampling data distributions similar to the prototype fingerprint.

The level of similarity between distributions depends on the thresholds you set when submitting the BURCLAR job. If the prototype and the fingerprint for a data element are deemed similar based on a calculated confidence value, the data element is also associated with the prototype class.

1. 1. Open the **Work with Jobs** window (**Environment** > **Work with Jobs**) to schedule a new job.
2. Click **New**.
3. In the **List of Jobs**, select the job BURCLAR (Class Data Element Assignment).
4. Select the DATAEXPRES **Machine ID** and the DEMO **company name**.
5. Click **Apply**. When comparing the fingerprint for a data element to the prototype fingerprint, internal program confidence values are calculated to provide measurements that illustrates the similarities between the data distributions. The internal confidence values are used as input for the threshold formulas.
6. In the **Select a class** list, click SAMPLING CUSTOMER VALUES.
7. Click **Add**. The class description now appears in the table.
8. In the **Rules** section, uncheck **By Attribute** if it is checked.
9. Set the **Threshold 1** value to 50. The value for **Threshold 1** represents a percentage where the similar areas in the two fingerprints are weighted more heavily than the dissimilar areas.
10. Let the default value of 0 for **Threshold 2**. The value for **Threshold 2** represents a percentage where all areas (both similar and dissimilar) in the two fingerprints are given equal weight.
11. Keep the logical operator set to AND.
12. Click **OK**.
13. In your TSO environment, submit the BURCLAR job by using the **Submit Client Scheduled Job** panel.
14. In the **Work with Jobs** area, click refresh to verify that the job completed successfully.
15. Close the **Work with Jobs** area.
16. In your mainframe environment, verify the class assignment.
17. Return to the **Work with Data Elements** window and refresh the list of data elements.
18. Select the **Select Data Elements** tab if it is not already selected.
19. Click the **Data Store Attributes** tab.
20. Delete any text in the **Data store Name** field.
21. Click **Apply Filter**.
22. Click the **Estimated Classes - Confirm** tab.

23. In the **List of Data Elements (Filtered)**, select the row that has an estimated class but not an assigned class.
24. Click **Confirm Class**. The **Estimated Class** names now appear in the **Assigned Class** column.
25. Close the **Work with Data Elements** window and return to the **Work with Data Stores** window. A sampling class has successfully been assigned to data elements with similar distributions to the prototype.
26. Close **Data Builder**.

Data Generation

In this tutorial, you perform basic data generation, working with the TGT.DEPT and TGT.EMPLOYEE tables to populate a table based on the content of another table and on data in a dictionary.

Begin this session only after you have set your workspace and loaded data store information into the Knowledge Base by following the instructions in tutorials *Using Data Express* and *Data Inventory*. You must be logged into Data Builder before you begin.

In this session, you:

- Catalog the XDB tables provided in the sample
- Start the Data Generator module.
- Create simple Data Generation code
- Submit JCL to generate data

Complete these tutorials in the following order:

Catalog and Classify Tables

1. Using a different machine and company, repeat the cataloguing phase, but loading only the TGT.DEPT and TGT.EMPLOYEE tables.
2. Execute the classification phase as described in the *Data Subset Extraction* part of this guide, assigning the same class to the DEPTNO columns of both TGT.DEPT and TGT.EMPLOYEE.

Start the Data Generation Module

1. Start the **Data Generation** module from your **Start** menu or tile, depending on your Windows operating system.

   Note: If you are using Data Express on Windows Vista and User Access Control is enabled, you must run the Data Generation module as Administrator.
2. Select the **XDB Knowledge Base** as the required database for your ODBC connection.

   By default, the highlighted item in the list of available databases is the last database you connected to using Data Masking. This becomes apparent after you click **Available databases**.
3. Click **OK**.
4. Log in using your username and password.
5. Click **OK**.
6. Select the schema **DX40**; click **OK**.

Create Simple Data Generation Code

1. Launch the New Code Wizard from the **New Code** icon.
2. Specify the proper machine and company.
3. Give the code the name `SIMPLEC` and provide a description.
4. Click **Apply**.

   This creates the code.

5. Select the code, and look at the lower part of the screen; it shows the laws that are used in the code, one law per table.
6. Ensure that the **TGT.EMPLOYEE** table appears before the **TGT.DEPT** table. If now, reorder the tables using the arrows on the right.
7. Right-click on **TGT.EMPLOYEE**; then select **Set directive** from the context menu.
8. Select the **DEPT** column; then right-click and select **Assign output class** specifying the **DEPT** column.
9. Right-click **TGT.EMPLOYEE**; then select **Don't create output** on the context menu.
10. Right-click **TGT.DEPT**; then select **Set directive** from the context menu.
11. Right-click the COD_CUS column; then select **Set directive** from the context menu, specifying **Other file dependencies** as the directive type.
12. Right-click **TGT.EMPLOYEE**; then select **Set directive** on the context menu.
13. Select the **DNAME** column; then select **Set directive** from the context menu, specifying **Dictionary** as the directive type and **USDCHSUR** as dictionary name.
14. Export the information about the tables DEPT and EMPLOYEE using the Distributed Exporter.

Execute a Simple Data Generation Law

By default, Extension Technology attempts to insert the processed data into the target database.

Execute a simple data generation law with Extension Technology:

1. From a command prompt, change to the executables directory, which by default is `%PROGRAMFILES%\Micro Focus\Data Express 4.0\Synthetic\ODBC`.
2. Enter `dxestart`.

   This populates the tables in the target ODBC data store using the specified data generation laws. After the execution finishes without errors, you are prompted to view the log file.
3. Type Y to view the file.
4. Close the text file.
5. Continue to the next section.

Verify Results

Verify that the tables are populated according to definition:

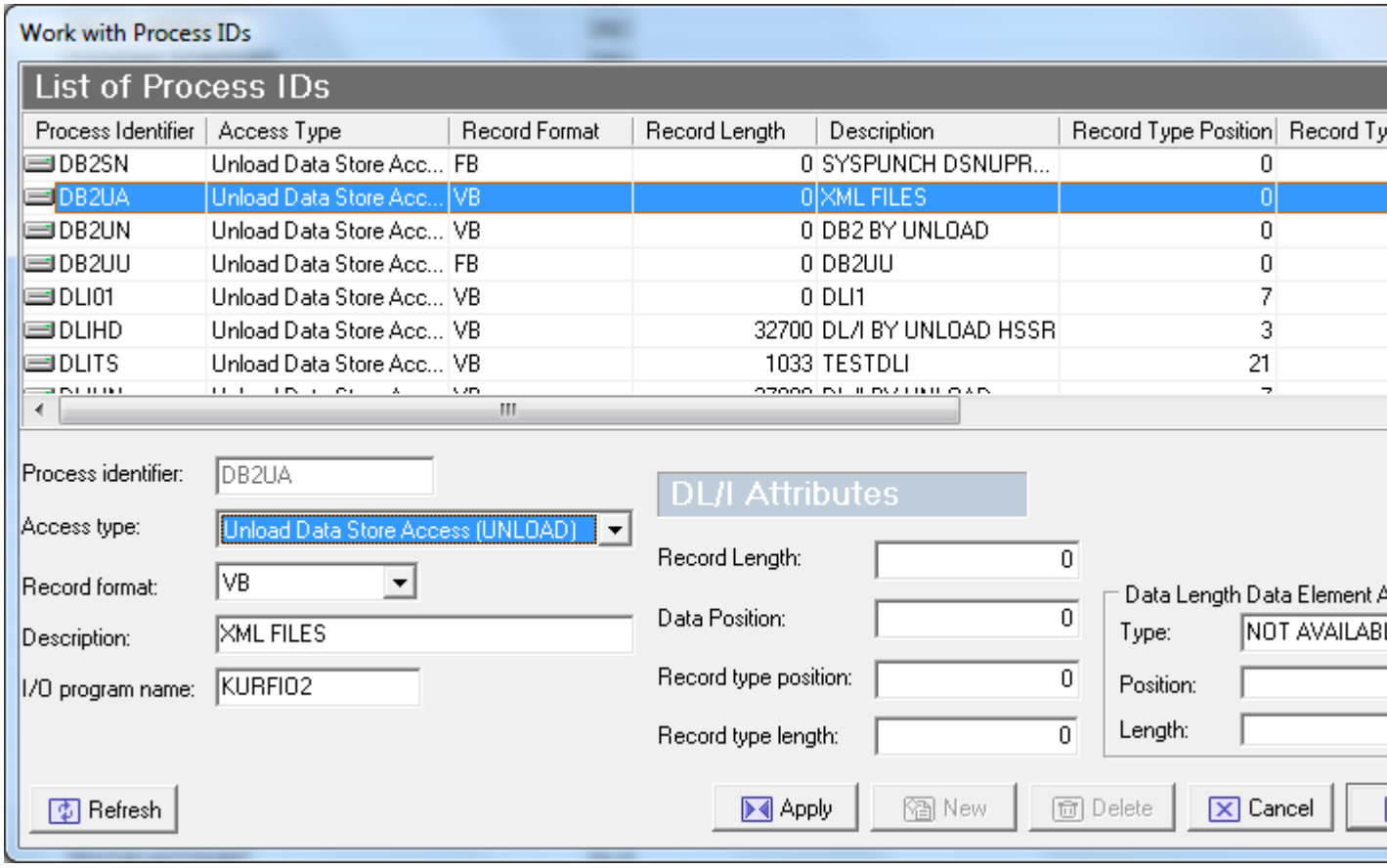
1. Launch the SQL Wizard: `..\mfsql\bin\xwiz40n`
2. Verify that the **TGT.EMPLOYEE** table contains its original values, and the **TGT.DEPT** table has been populated according to the specified laws.

Managing XML and Large-object Columns

Describes how to manage masking and subsetting of XML data stored in XML or LOB/BLOB/CLOB columns of z/OS DB2 tables, supporting both DSNTIAUL and UNLOAD formats.

Load and Classification

Structure load of tables containing XML data, both in XML and in LOB/BLOB/CLOB columns, is performed via direct access DB2 load from the Work with Process IDs window, using the same load process as is used for tables that do not have these types of columns.



XML and LOB/BLOB/CLOB columns are recorded in Data Express as VARCHAR(56), which is required to define a process identifier. In particular, be careful to always set the **Access type** for XML FILES correctly based on UNLOAD and DSNTIAUL keywords.

For example, the screenshot above cites `Unload Data Store Access (UNLOAD)`, which is to be used for UNLOAD formats. If you are using the DSNTIAUL format, use `Unload Data Store Access (DSNTIAUL)` instead.

Class definition and assignment are the same as for normal columns. For all XML or LOB/BLOB/CLOB columns that are not being masked, we recommend that you define a special class to assign them to, and to assign them the UDCXML0 "dummy" masking routine.

Note: The information shown here is used only for masking and subsetting. It is not used for loading and cataloging.

Masking

You can use either the UNLOAD or the DSNTIAUL format for masking.

UNLOAD Format

Masking using the UNLOAD format requires a sequential unload for each table, and a PDS for each XML/LOB/BLOB/CLOB column. The following JCL provides an example of the required code:

```
//LOADLOB JOB KRM,MSGCLASS=A,CLASS=1,NOTIFY=&SYSUID
JOB07656
//* ****************************************
//* TABLE OWURSVIL.BIG01
//* (PDFBLO BLOB (27994))
//* (TXTCLO CLOB (16094) )
```

```
//* (PDFBLO BLOB (16032) )
//* UNIT(SYSDA) SPACE ((20,2) MB)
//* (PDFBLO VARCHAR(54) BLOBF TSYSLOB)
//* ************************************
//STEP1 EXEC DSNUPROC,UTPROC=,SYSTEM=HADB,LIB=DSNA10.SDSNLOAD
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  TEMPLATE TSYSPUN
     DSN('URADAR.&SS..&DB..PUNCH.CREDB1')
     DISP(MOD,CATLG,CATLG)
  TEMPLATE TSYSREC
     DSN('URADAR.&SS..&DB..SYSREC.CREDB1')
     DISP(MOD,CATLG,CATLG)
  TEMPLATE TSYSLOB
     DSN('URADAR.&SS..&DB..PDS.CREDB1')
     DISP(MOD,CATLG,CATLG)
     DSNTYPE(PDS)
  UNLOAD DATA FROM TABLE OWURSVIL.CREDB1 HEADER NONE
   (A     , B                  ,C      VARCHAR(54) CLOBF TSYSLOB,
    D)
  UNLDDN(TSYSREC) PUNCHDDN(TSYSPUN)
/*
//
```

In particular, use must include the HEADER NONE clause, and specify VARCHAR(54) for each XML/LOB/CLOB/BLOB column.

Before creating the method, be sure that you have created an output PDS via the masking routine, and that you have generated a process ID from Data Builder. When you create the method, include the input and output data set names.

DSNTIAUL Format

Masking using the DSNTIAUL format requires a sequential unload for each table. The following JCL provides an example of the required code:

```
//DB2UNLO  JOB (00001),URADAR,MSGCLASS=1,CLASS=1,NOTIFY=&SYSUID
//STEP0100 EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(HADB)
 RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIB10)                -
       LIB('DSNA10.DB2HAL.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSREC00 DD DSN=URADAR.SV41.CREDB2.NEW,
//            DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(24,24),RLSE)
//SYSPUNCH DD DSN=URADAR.UTILITY.SYSPUNCH(CREDB2N),DISP=SHR
//SYSIN    DD *
                  OWURSVIL.CREDB2
```

Before creating the method, be sure that you have generated a process ID from Data Builder. When you create the method, include the input and output data set names.

XML Tag Identifiers

To pass data to a data masking routine from XML, you need to identify each data masking object in your XML code. A data masking object is one or more XML fields that each contain functionally equivalent data to be masked. You assign a tag identifier to each data masking object, and pass the identifier to the data masking routine as a literal string. This requires that you assemble your XML tags such that each data masking object can be uniquely identified by its elementary tag alone, or by qualification using its elementary and parental tags. The following examples illustrate this point:

Elementary tag identifier:

```
<Company>
   <Nm>Yoga Life Spa</Nm>
</Company>

<Client>
   <Nm>Hope Lesley Stressing</Nm>
</Client>
```

In this example, the <Nm> tag contains either a company name or a client name. Each <Nm> elementary tag instance is a child of a different parent – <Company> to identify a company, and <Client> to identify a client. Assuming company and client data are functional equivalents, the tag identifier used to represent this data masking object and that is passed to the data masking routine is just the elementary tag name, Nm, without further qualification.

Hierarchical tag identifier:

```
<OrgId>
   <IBAN>
      <Id>IT77889900100000004245520999</Id>
   </IBAN>
</OrgId>

<OrgId>
   <TaxCode>
      <Id>XYZABC40S18L424B</Id>
   </TaxCode>
</OrgId>
```

In this example, the <Id> tag is used to provide either the IBAN (International Bank Account Number) or the tax code associated with an organization. These are not functional equivalents, and therefore represent two different data masking objects. However, each is contained in an elementary <Id> tag; therefore, the tag identifier for each must be further qualified by including both the parent and the child tag. To do this, you specify the parent tag and its child tag in sequence, separated with a comma. For example: IBAN,Id and TaxCode,Id respectively.

Each tag identifier must include as many levels in the tag hierarchy as are required to uniquely identify the elementary tag that contains the data. For example, to uniquely identify each <Id> tag in the following code requires four levels of qualification: IBAN,Country,CountryCode,Id and TaxCode,Country,CountryCode,Id respectively.

```
<OrgId>
   <IBAN>
      <Country>
         <CountryCode>
            <Id>IT77889900100000004245520999</Id>
         </CountryCode>
      </Country>
   </IBAN>
</OrgId>

<OrgId>
   <TaxCode>
      <Country>
         <CountryCode>
            <Id>XYZABC40S18L424B</Id>
         </CountryCode>
      </Country>
   </TaxCode>
</OrgId>
```

To learn how to pass a tag identifier to a data masking routine, see *XML Masking Template UDCXML1* .

Provided Templates

Data Express provides the following templates:

For the UNLOAD format

- UDCXML1 is a prototype for masking XML data
- UDCXML0 is a prototype for managing data stored in large object columns not needing masking
- UDCLOB1 is a prototype for masking positional data in a large object column

For the DSNTIAUL format

- UDCXML6 is a prototype for masking XML data
- UDCLOB6 is a prototype for masking positional data in a large object column

Note: When using the DSNTIAUL format with LOB columns that are not masked, a masking routine specification is not required.

XML Masking Template UDCXML1

Data Express provides the UDCXML1 data masking routine as a template for processing XML data. You can use the template as-is, or customize and/or replicate it according to your needs.

Note: The information in this topic that applies to UDCXML1 also applies to UDCXML6 except for the output PDS naming convention because PDSs are not present in DSNTIAUL format. In addition, the information in this topic that applies to UDCLOB1 also applies to UDCLOB6.

The sections that follow explain how to customize the STARTING-OPERATIONS section in the UDCXML1 template to manage:

- Character set
- Passing XML data
- XML masking routines
- The naming convention for output PDSs

Character Set

Source XML can be stored in DB2 using any of several supported character sets such as EBCDIC, ASCII, various dialects of EBCDIC and ASCII, or even custom character sets.

To correctly manage this, the engine called by the data masking routine needs to know the character set in use so it can appropriately process the data. Depending on the specified character set, the data masking engine performs certain tasks such as the transforming of single-byte characters from one set to another, transforming EBCDIC to ASCII before processing, and recoding any transformed characters back to the original character set before writing the output.

Standard character set

By default, the character set specified to Data Express is "*NONE", meaning that no character transforming is required. This assumes that the data being parsed is in EBCDIC and uses the same code page as is specified in your Data Express mainframe installation:

```
       STARTING-OPERATIONS.
      *
      * CHARACTER SET
      *
      *    MOVE "ASCII"              TO WS-CHARSET.
      *    MOVE "EBCDIC"             TO WS-CHARSET.
           MOVE "*NONE"              TO WS-CHARSET.
```

In addition to ASCII and EBCDIC, you can customize this value to represent an alternative or a custom character set.

Alternative or customized character set

To specify a character set other than ASCII or EBCDIC, you must change the value passed to the WS-CHARSET variable. To do this, edit the character set conversion logic in the provided KDCXML3W copybook, which is called by the UDCXML1 routine. Follow these steps:

1. Add 1 to the KDCXML3-NUM-TAB-USATE VALUE clause.
2. After the existing TRASC02 group field, add a new similarly-coded group field named TRASC03.

   Note: If TRASC03 or any number of additional TRASC0*n* fields already exist due to previous editing, name the new group field by incrementing *n* to the next consecutive number; for example, if TRASC03 exists, name the new group field TRASC04.
3. Code the new group field as follows:

   - An initial elementary field that is the name of the encoding.
   - A set of variables that associate the value of a character in the customized character set with the value of the corresponding character in the EBCDIC character set.

   For example:

```
            10 TRASC03.
               15 TAB-TRASC03 PIC X(16) VALUE 'NO-TRASX'.
               15 EL-TRASC03-001 PIC X(2) VALUE X'0000'.
               15 EL-TRASC03-002 PIC X(2) VALUE X'0101'.
               15 EL-TRASC03-003 PIC X(2) VALUE X'0202'.
               .
               .
               .
               15 EL-TRASC03-256 PIC X(2) VALUE X'0000'.
```

4. Following the new code group, locate and delete the following rows:

```
            10 TAB-VALUE03 PIC X(16).
            10 TAB-EL03 PIC X(2) OCCURS 256 TIMES.
```

   Note: If your group field name uses an incremented number other than 3, find and delete the two rows that correspond to the number used for your group field.
5. Recompile UDCXML1.

Passing XML Data to Masking Routines

To pass data to a masking routine, you code several MOVE statements in the STARTING-OPERATIONS section. Before starting:

- Assign a tag identifier to each data masking object in your XML code. See *XML Tag Identifiers* for details.
- Determine the total number of data masking object instances contained in your XML code. A data masking object instance is single occurrence of a data masking object.
- Associate the number 1 with the first data masking job instance, and continue numbering consecutively through the last instance. These numbers are for your reference.
- Determine the maximum length of each data masking object.
- Determine the name of each data masking routine you want to use. See *Masking Routines* for more information.

Data Express provides the **WS-NUMTAG(*n*)** field to hold the total number of data masking object instances to process, and provides the following three fields to identify the data and the masking routine for each instance. Each data masking object instance uses all three fields:

WS-TAG(*n*) Where *n* is the number (from 1 to WS-NUMTAG) that you have associated with the data masking object instance. Into this field, you move the tag identifier for a data masking object instance.

WS-LEN1(*n*) Where *n* is the number (from 1 to WS-NUMTAG) that you have associated with the data masking object instance. Into this field, move the length of the data masking object instance. You can set this to an exact field length for standard-sized data, such as an IBAN, or set it higher to accommodate dynamic field lengths. The limit is 255.

WS-ROUTINE(*n*) Where *n* is the number (from 1 to WS-NUMTAG) that you have associated with the data masking object instance. Into this field, move the name of the masking routine to use.

The following example shows code used to move the data from eight XML data object instances to their appropriate masking routines:

```
      *-----------------------------------------------------------------
      *
       STARTING-OPERATIONS.
           .
           .
           .
      *
      * MASKING ROUTINE
      *
           MOVE  8                    TO WS-NUMTAG.
           MOVE "Nm"                  TO WS-TAG      (1).
           MOVE 255                   TO WS-LEN1     (1).
           MOVE "ADLER"               TO WS-ROUTINE  (1).
           MOVE "AdrLine"             TO WS-TAG      (2).
           MOVE 255                   TO WS-LEN1     (2).
           MOVE "ADLER"               TO WS-ROUTINE  (2).
           MOVE
           "Mndt,Dbtr,Id,PrvtId,Othr,Id"
                                      TO WS-TAG      (3).
           MOVE 16                    TO WS-LEN1     (3).
           MOVE "UDCCDFC"             TO WS-ROUTINE  (3).
           MOVE SPACES                TO WS-TAG      (4).
           STRING "UndrlygAccptncDtls,OrgnlMndt,OrgnlMndt,"
                   DELIMITED BY SIZE
                  "Dbtr,Id,PrvtId,Othr,Id"
                   DELIMITED BY SIZE
                                     INTO WS-TAG      (4).
          MOVE 16                     TO WS-LEN1     (4).
          MOVE "UDCCDFC"              TO WS-ROUTINE  (4).
          MOVE "BirthDt"              TO WS-TAG      (5).
          MOVE 10                     TO WS-LEN1     (5).
          MOVE "UDCMDY0"              TO WS-ROUTINE  (5)
          MOVE "NOME"                 TO WS-TAG      (6).
          MOVE 255                    TO WS-LEN1     (6).
          MOVE "UDCNAMC"              TO WS-ROUTINE  (6)
          MOVE "COGNOME"              TO WS-TAG      (7).
          MOVE 255                    TO WS-LEN1     (7).
          MOVE "UDCSURC"              TO WS-ROUTINE  (7).
          MOVE "TAXCODE"              TO WS-TAG      (8).
          MOVE 16                     TO WS-LEN1     (8).
          MOVE "UDCCDFC"              TO WS-ROUTINE  (8).

      STARTING-OPERATIONS-EX.
```

Output PDS Naming Convention

You can customize the following algorithm to set the output PDS. The name of the variable to use is WS-PDS-OUT:

```
      *-------------------------------------------------------------
      *
       STARTING-OPERATIONS.
           .
           .
           .
      *
      * OUTPUT PDS NAMING CONVENTION
      *
           MOVE SPACES TO WS-PDS-OUT.
           STRING "XXX.YYY." DELIMITED BY SIZE
                  WS-METODO DELIMITED BY SPACES
                  "." DELIMITED BY SIZE
                  WS-TABELLA DELIMITED BY SPACES
                  ".LIB" DELIMITED BY SIZE
                  INTO WS-PDS-OUT.
```

Dummy Masking Template UDCXML0

In addition to the UDCXML1 XML masking template, Data Express also provides the UDCXML0 XML masking template. Use this simple template when character set and data masking object customizations are not required. You can edit UDCXML0 to customize the output PDS naming convention just as you would in UDCXML1. For details, see the *Output PDS Naming Convention* section in *XML Masking Template UDCXML1*.

LOB/BLOB/CLOB Masking Template UDCLOB1

Data Express provides the UDCLOB1 data masking routine as a template for managing specific positional LOB/BLOB/CLOB cases. You can use the template as-is, or customize and/or replicate it according to your needs.

The sections that follow explain how to customize the STARTING-OPERATIONS section in the UDCLOB1 template to manage:

- The starting and ending position for each masking object
- The naming convention for output PDSs

Passing LOB/BLOB/CLOB Data to Masking Routines

To pass data to a masking routine, you code several MOVE statements in the STARTING-OPERATIONS section. Before starting:

- Determine the total number of data masking object instances contained in your LOB/BLOB/CLOB code. A data masking object instance is single occurrence of a data masking object.
- Associate the number 1 with the first data masking job instance, and continue numbering consecutively through the last instance. These numbers are for your reference.
- Determine the starting position for each data masking object.
- Determine the ending position for each data masking object.
- Determine the name of each data masking routine you want to use. See *Masking Routines* for more information.

Data Express provides the **WS-NUMTAG(*n*)** field to hold the total number of data masking object instances to process, and provides the following three fields to identify the data and the masking routine for each instance. Each data masking object instance uses all three fields:

WS-POS1(*n*) Where *n* is the number (from 1 to WS-NUMTAG) that you have associated with the data masking object instance. Into this field, you move the starting position for a data masking object instance.

WS-END1(*n*) Where *n* is the number (from 1 to WS-NUMTAG) that you have associated with the data masking object instance. Into this field, you move the starting position for a data masking object instance.

WS-ROUTINE(*n*) Where *n* is the number (from 1 to WS-NUMTAG) that you have associated with the data masking object instance. Into this field, move the name of the masking routine to use.

The following example shows code used to move the data from three LOB/BLOB/CLOB data object instances to their appropriate masking routines. This code represents:

- A name at position 1 with length 20
- A surname at position 21 with length 20
- A tax code at position 190 with length 16

```
      *-------------------------------------------------------------
      *
       STARTING-OPERATIONS.
           .
           .
           .
      *
      * MASKING ROUTINE
      *
          MOVE 3                    TO WS-NUMTAG.
          MOVE 1                    TO WS-POS1     (1).
          MOVE 20                   TO WS-END1     (1).
          MOVE "UDCNAMC"            TO WS-ROUTINE  (1).
          MOVE 21                   TO WS-POS1     (2).
          MOVE 40                   TO WS-END1     (2).
          MOVE "UDCSURC"            TO WS-ROUTINE  (2).
          MOVE 190                  TO WS-POS1     (3).
          MOVE 205                  TO WS-END1     (3).
          MOVE "UDCCDFC"            TO WS-ROUTINE  (3).

      STARTING-OPERATIONS-EX.
```

Output PDS Naming Convention

You can customize the following algorithm to set the output PDS. The name of the variable to use is WS-PDS-OUT:

```
      *-------------------------------------------------------------
      *
       STARTING-OPERATIONS.
           .
           .
           .
      *
      * OUTPUT PDS NAMING CONVENTION
      *
           MOVE SPACES TO WS-PDS-OUT.
           STRING "XXX.YYY." DELIMITED BY SIZE
                  WS-METODO DELIMITED BY SPACES
                  "." DELIMITED BY SIZE
                  WS-TABELLA DELIMITED BY SPACES
                  ".LIB" DELIMITED BY SIZE
                  INTO WS-PDS-OUT.
```

Index