z/TPF EE V1.1
z/TPFDF V1.1
TPF Toolkit for WebSphere® Studio V3
TPF Operations Server V1.2

IBM Software Group

TPF Users Group Spring 2006

SDO Access to z/TPFDF Databases

Name: Sasha Krymer
Venue: Database Subcommittee

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

Agenda

- Why Service Data Objects (SDO)?
- Introduction to SDO on z/TPFDF
- Writing SDO-based applications to access z/TPFDF data
- Existing z/TPFDF code vs. SDO-based code

SDO

- New model of data access
- Complementary technology for SOA
- Developed jointly by IBM and BEA
- Standardized using Java Specification Request (JSR) 235

Why SDO?

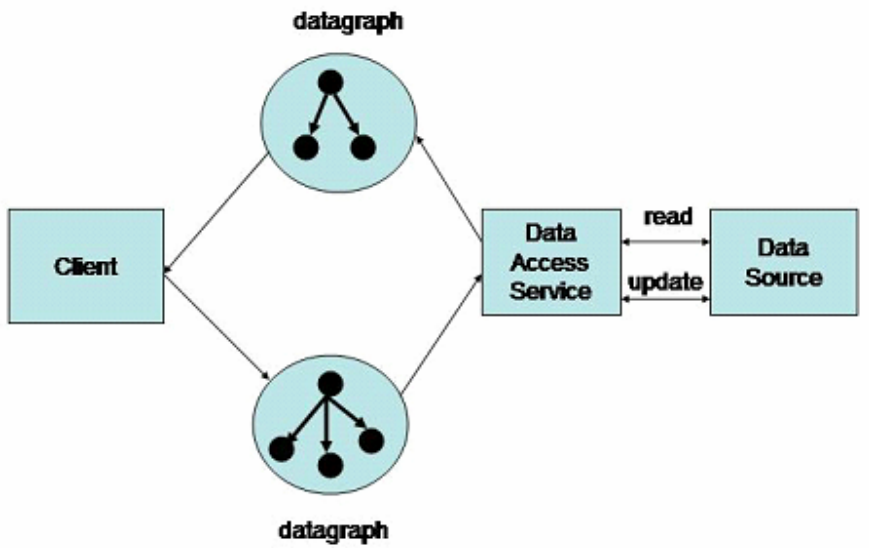
- Convenient and generic way to access data
  - Universal model for business data
  - Common unifying format for exchanging data between services
  - Includes dynamic interfaces
  - Not tied to the data organization, like SQL to relational databases
  - Object-oriented, thus maintenance is easier

SDO Concepts

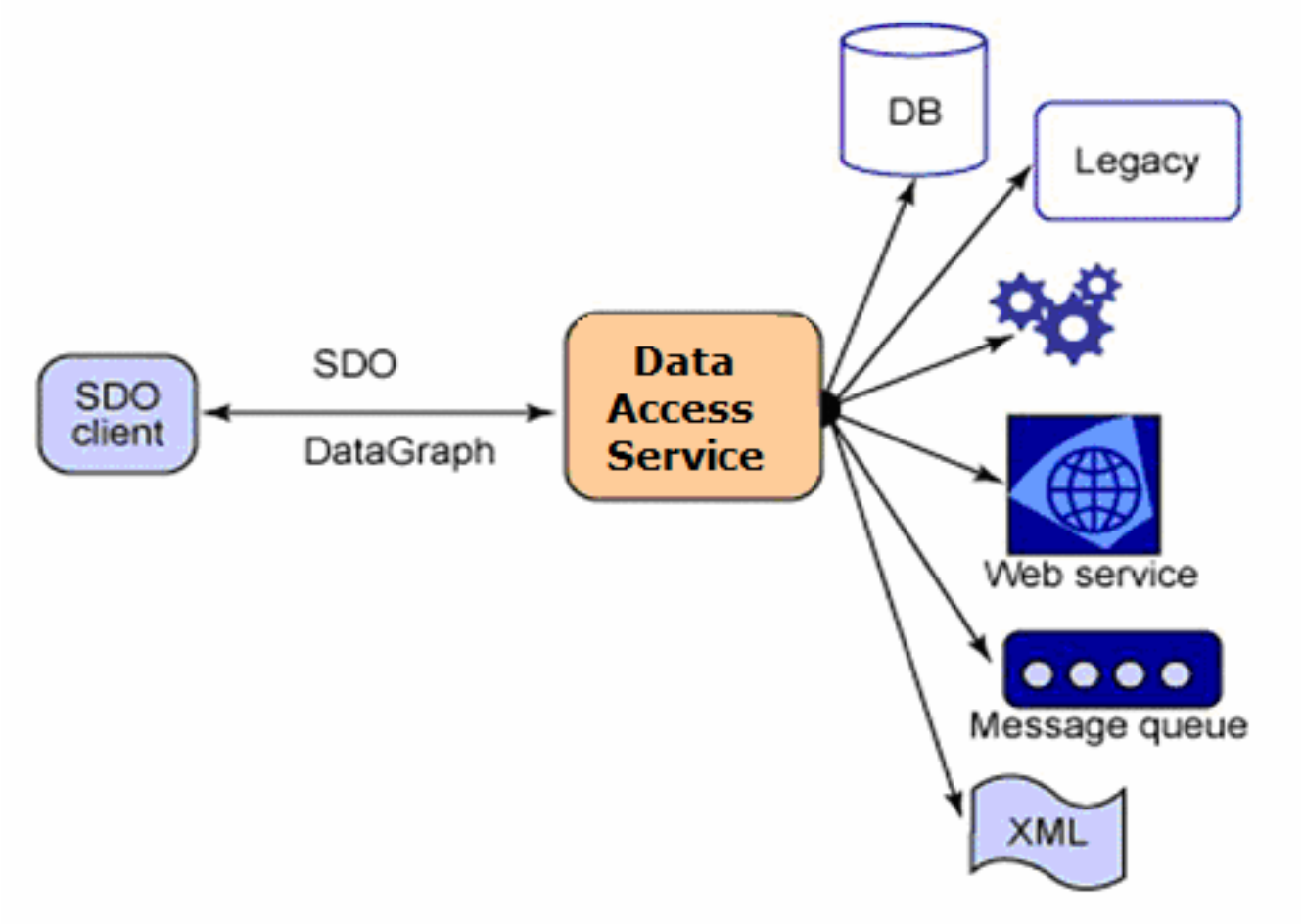
- DataObject – entity representing fragment of data
  - Property (single-valued, many-valued, “simple”, “complex”)
  - Type (String, Integer, Date, Boolean, DataObject)
- DataGraph – graph representing data (non-persistent)
  - Tree of DataObjects connected by references
  - Nodes are accessed via root & references
- ChangeSummary
  - change history for DataGraph and DataObjects

SDO Concepts (cont.)

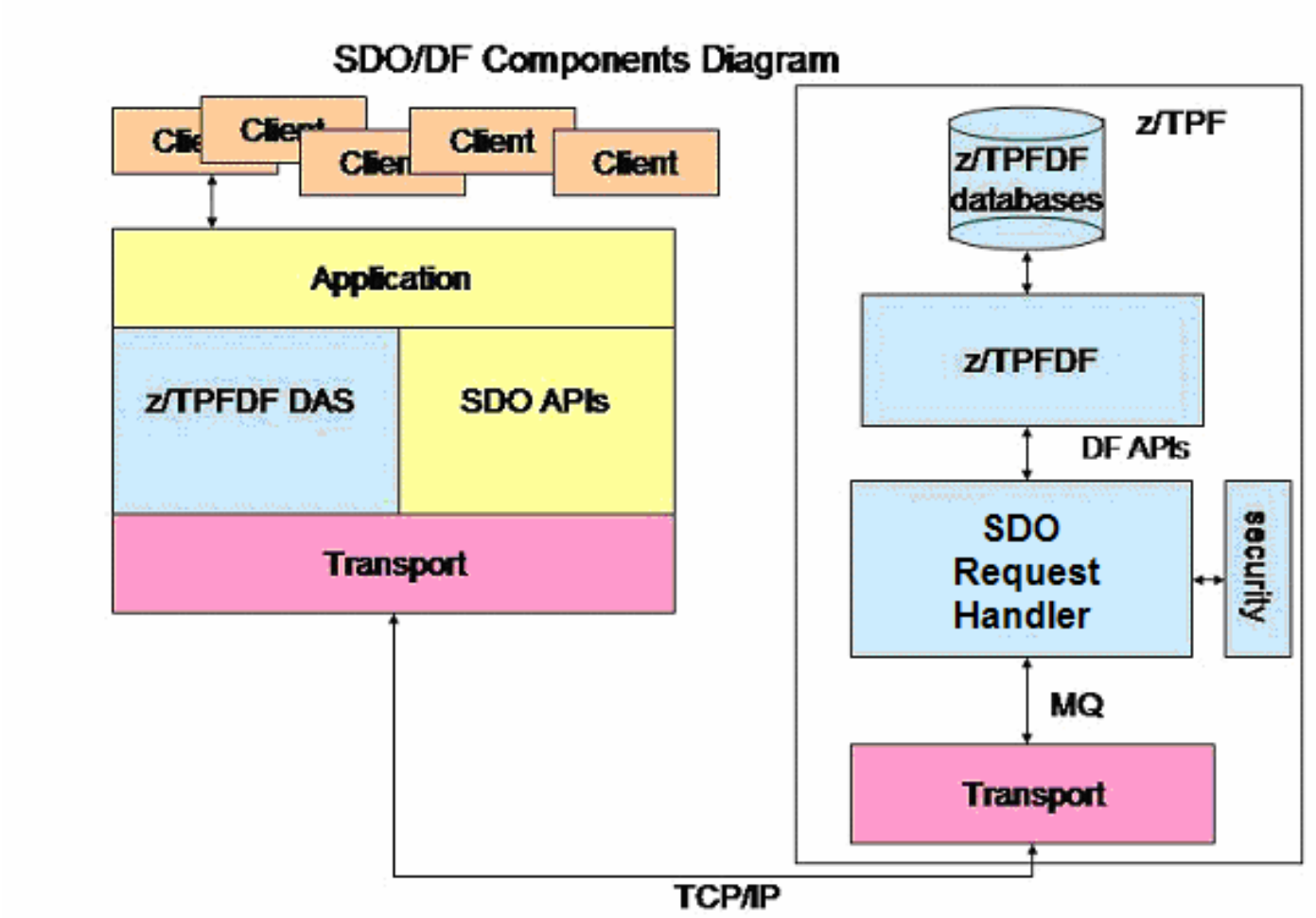
- Data Access Service (DAS)
  - Specific form of SCA (Service Component Architecture) service
  - Load DataGraph from a data source or service
  - Propagate changes back into the data source
  - Disconnected model
  - XML – XML file DAS, Relational – JDBC DAS



SDO in SOA



z/TPFDF and SDO Direction



z/TPFDF and SDO (cont.)
Direction

- Metadata
  - New ZUDFM MLS METADATA command
  - Fill in blanks
    - Aliases
    - Java data types (String, Date, Boolean etc.)
- Security
  - User exits on both sides

DataGraph and z/TPFDF data Direction

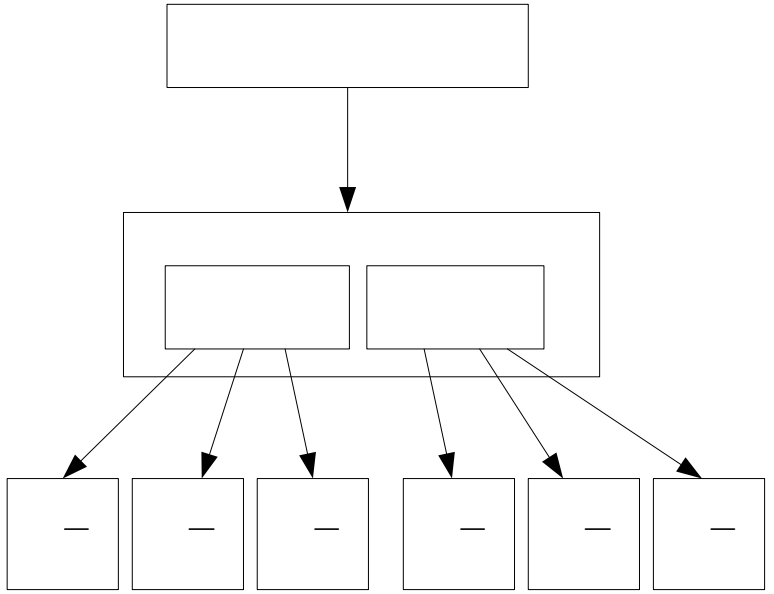
- z/TPFDF <=> SDO mapping:
  - LREC – Data Object
  - Subfile – Data Object
  - Field – Property
- Access Data
  - Recursively traverse references, e.g.,

```
List a = a.getList("Aircraft80");
String At = a[0].getString("At");
```

  - XPath, e.g.,

```
String At =
a.getString("Aircraft80.0/At");
a.setString("Aircraft80[Sr='25']/Cl",
            "New Eco")
```

IR25DF			
Aircraft File			
Aircraft80	**Aircraft Type (At)**	**Seat Number (Sr)**	**Seat Class (Cl)**
80	747	24	Eco
80	747	25	Eco



Generic z/TPFDF-SDO application
Direction

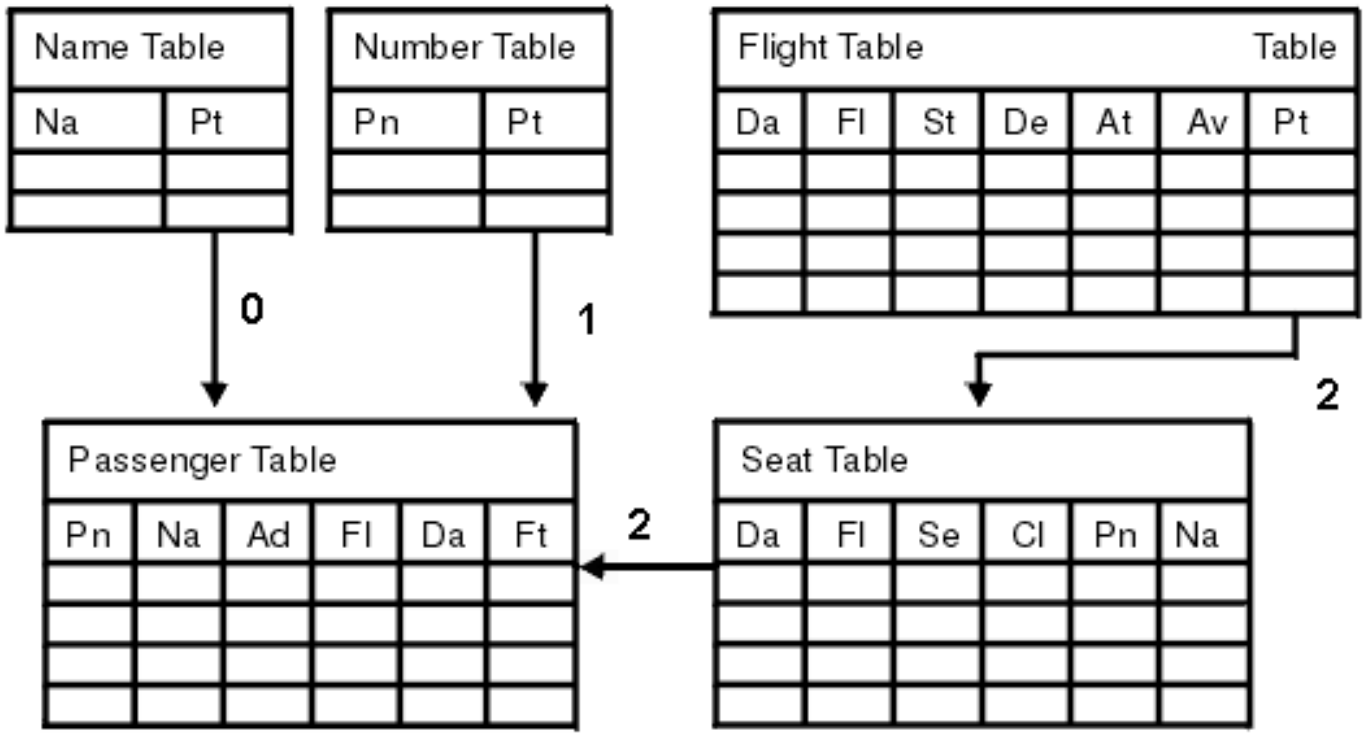
- Generic code structure:

```
/* Instantiate DAS */
TPFDF_DAS das = new TPFDF_DAS("/etc/db.cfg", "DBName");
/* Get data from z/TPFDF */
DataGraph dg = das.readData(…);
DataObject rootDO = dg.getRootObject();
…
/* DataGraph manipulation - read and modify data */
String name = rootDO.get("XPath expression/Name");
if (name.equals("Old Name"))
      rootDO.set("XPath expression/Name", "New Name");
…
das.applyChanges(dg);   /* Make updates on z/TPFDF */
```

z/TPFDF DAS Instantiation
Direction

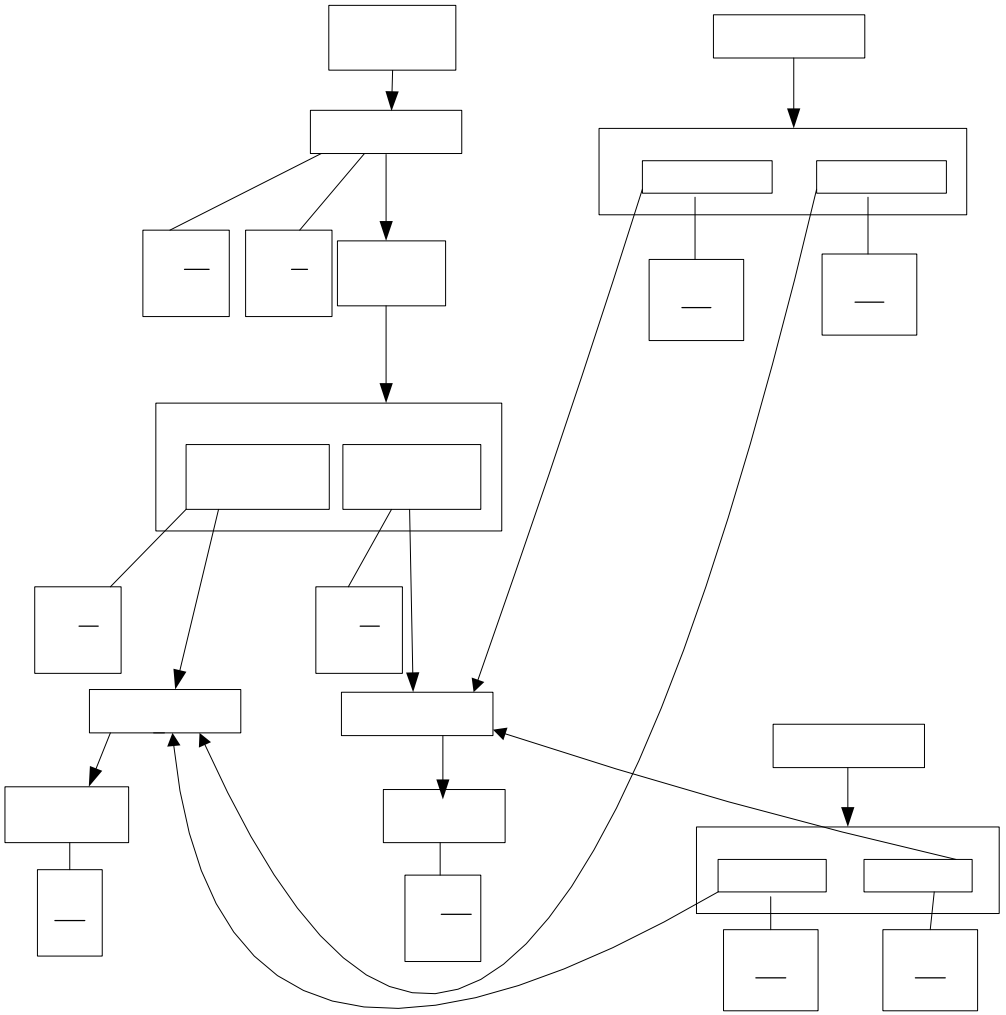
- Configuration file:
  - General parameters
    - Administration parameters (timeout value, max data size)
  - Database
    - Database name (new DBDEF parameter!)
    - Subsystem (SS)
    - Subsystem user (SSU)
    - How to access this database/SS/SSU combination
      - MQ parameters (Queue Manager, Queue Name)

Sample z/TPFDF database



Sample DataGraph

Direction



Get Data from z/TPFDF
Direction

```
readData(String fileToRetrieve,
         String mainDFPath,
         String[] searchKeys,
         String[] propertiesToReturn,
         String[] indexPathsToRetrieve)
```

- Information provided by application:
  - Which file to retrieve? (Passenger Table)
  - Which path to use to access the file? (Name? Number?)
  - Which LRECs to retrieve? (Na = “John Smith”)
  - Which fields in the LRECs to retrieve? (Na, Ad)
  - Which additional higher-level files to return? (Name Table, Seat Table, Flight Table)
- Example:
  - readData(“Passenger”, “0”, {“Sr=‘1’”, “Na=‘Ann’”}, {“Na”, “Ad”}, {“1”, “2”})

DataGraph Manipulation Direction

- Getters and Setters (generic and type-specific)
  - Data field property:
    - `String name  = dataobj.getString("Name");`
    - `dataobj.setString("Name", "newname");`
  - Index property:
    - `DataObject seatTable = (DataObject)flightTable.get("SeatTable");`
    - `flightLREC.set("SeatTable", newSeatTable);`
- Create
  - New subfile (off root)
    - `DataObject passenger = rootDO.createDataObject("Passenger");`
  - New LREC: (off subfile):
    - `DataObject lrec = passenger.createDataObject("LREC80");`
- Delete
  - Subfile or LREC:
    - `passenger.delete();`
  - Index:
    - `flightLREC.unset("SeatTable");`

Updating a z/TPFDF Database
Direction

`applyChanges(DataGraph datagraph)`

- Application supplies the updated `DataGraph` to the z/TPFDF DAS
- z/TPFDF DAS interrogates `ChangeSummary` and builds an update request in XML format to be sent to z/TPF
- applyChanges() handles changes to one subfile at a time. The check is performed prior to building an XML request to z/TPF

Important Points
Direction

- Optimistic concurrency
- z/TPF does not support two-phase commit, therefore updates performed via SDO and DAS cannot participate as part of an application commit scope
- z/TPFDF applications running locally control the locking protocol of files/subfiles (e.g., always lock *file A* before locking *file B*)

Requirements
Direction

- No specific remote platform required
- Will run anywhere WebSphere or equivalent runs

Notes

- Work in progress
- Your input is welcome!

Example (traditional vs. SDO)

TPFDF code

```
static void delete_spname()
{
  file_ptr95 = dfopn_acc("GR95SR
     ",_GR95SRI,DFOPN_ALG,DFOPN_HOLD, &ecbptr()->ebw040);
  dfopt1(file_ptr95,DFOPT_PATH,0);
  gr95pky = _GR95K90;
  df_nbrkeys(&keys,1);
  df_setkey(&keys,1,offsetof(struct gr95sr,gr95key),1,DF_EQ,
     &gr95pky,0,DF_NOORG,DF_CHAR);
  dfkey(file_ptr95,&keys);
  lrec95 = dfred(file_ptr95,0);
  if (DF_TEST(file_ptr95,DFC_ALG)) {
    sprintf(msg,msgd1a,&ecbptr()->ebw040);
    outmsg = msg;
  }
  else if (DF_ER(file_ptr95)) readerr(file_ptr95);
  else   {
    dfdix_alg_pth(file_ptr95,0,lrec95->gr95num,1);
    if (DF_ER(file_ptr95)) deindexerr(file_ptr95);
    qt08_return_employee_number(lrec95);
    dfopt1(file_ptr95,DFOPT_PATH,0);
    dfdel_acc(file_ptr95,DFDEL_ALG,DFDEL_ALL|DFDEL_NOKEY,
              &ecbptr()->ebw040);
    if (DF_ER(file_ptr95)) deleteerr();
    outmsg = msgd3;
  }
  dfcls(NULL,DFCLS_ALL);
  return;
}
```

SDO-based code

```
public void delete_spname(String input_string) {
    TPFDF_DAS das = new TPFDF_DAS("/etc/job1.cfg","JOB1");
    try {
        DataGraph dg = das.readData("NameJobDetail", "0",
        NULL, {"NameTable.*", "NameJobDetail.*"}, {"1"});
    } catch (Exception e) {
        System.out.println("readData() error: " +
               e.getMessage());
        System.exit(-1);
    }
    DataObject rootDO = dg.getRootObject();
    String xpath = "NameTable/Name80[Name='" +
                   input_string + "']/NameJob";
    DataObject subfile = rootDO.getDataObject(xpath);
    subfile.delete();
    try {
        das.applyChanges(dg);
    } catch (Exception e) {
        System.out.println("applyChanges() error: " +
                           e.getMessage());
        System.exit(-1);
    }
}
```

Trademarks

IBM and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries
BEA and WebLogic are registered trademarks of BEA Systems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Notes

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.