z/TPFDF V1.1

TPF Users Group Spring 2008

Application Development using SDODF

Name: Glenn Katzen
Venue: Applications Development Subcommittee

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

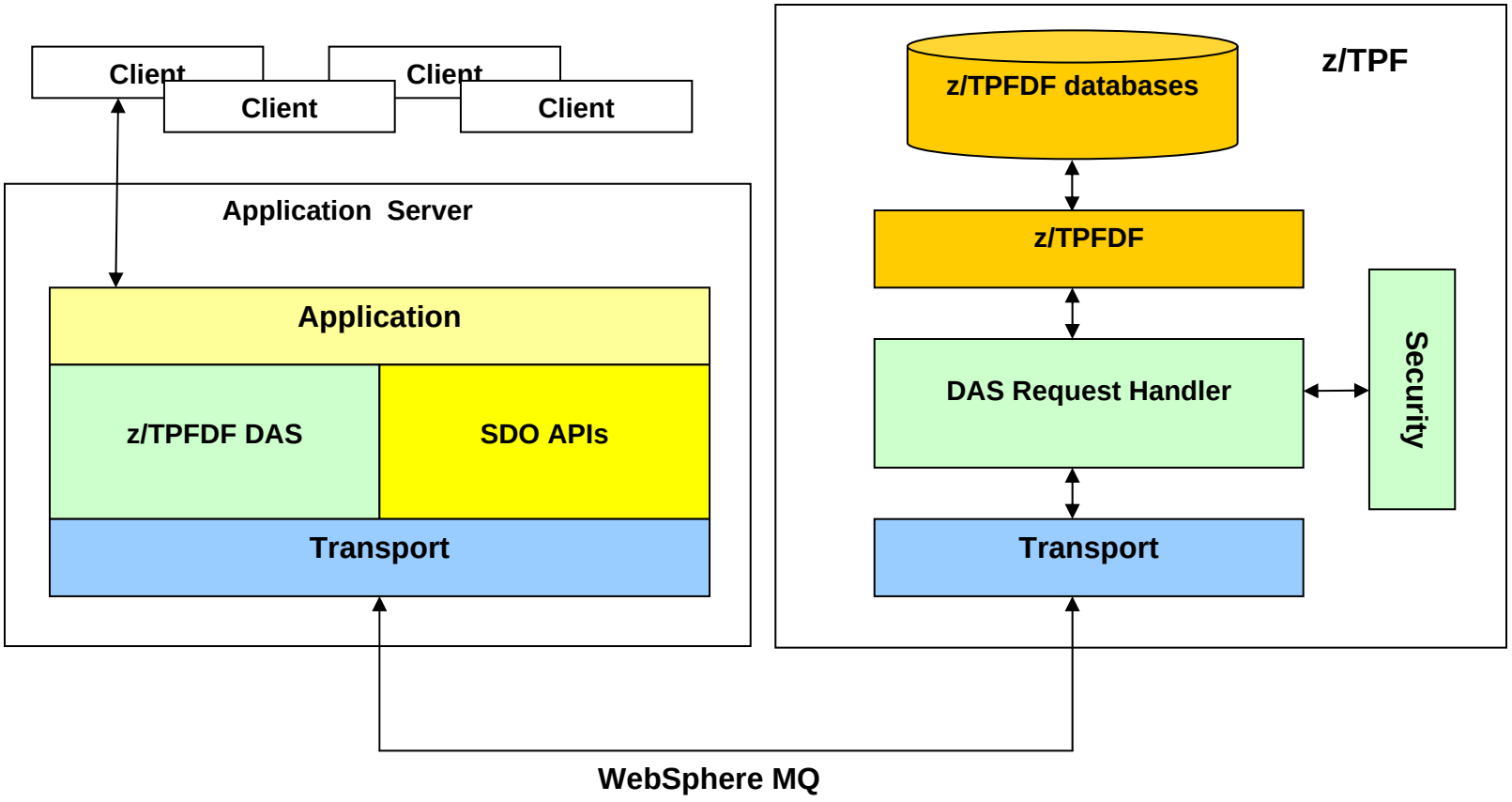
Agenda

- **SDODF Overview**
- **Service Data Objects (SDO)**
- **Mapping z/TPFDF to SDO**
- **Programming with SDODF**
- **SDODF Considerations**

SDODF Overview

- **SDODF provides access to z/TPFDF databases using Service Data Objects.**
- **Remote client applications written in Java can use SDODF to retrieve data and make updates.**

SDODF Overview



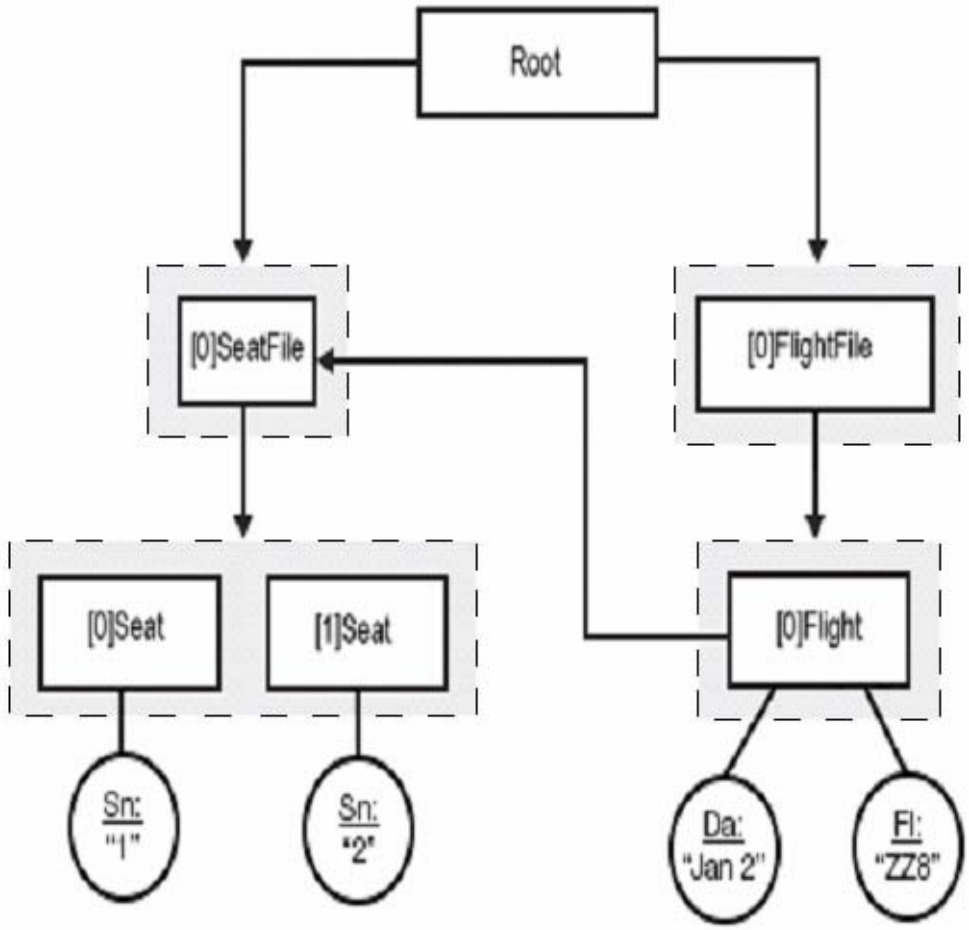
Service Data Objects

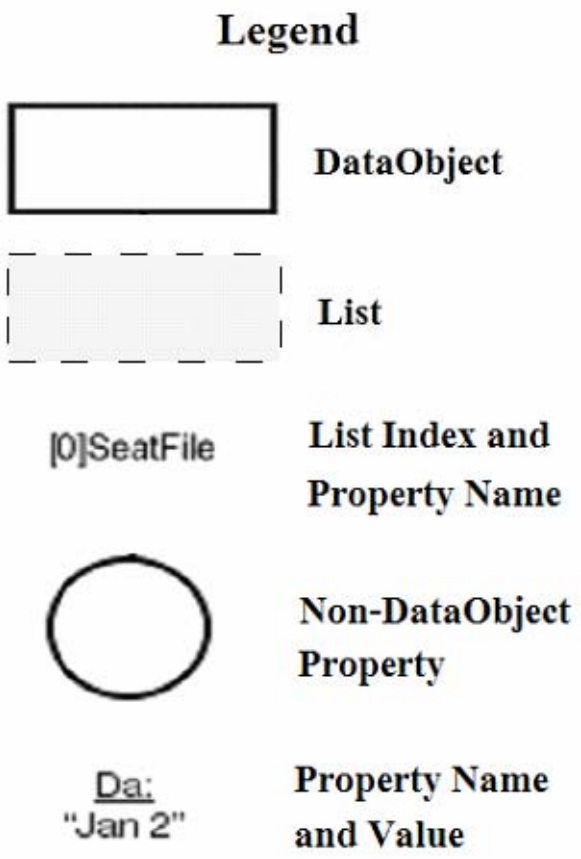
- **DataGraph**
  - A collection of one or more DataObjects
  - Always has a single root DataObject
- **DataObject**
  - A collection of one or more Properties
- **Property**
  - Has a unique name and a type
  - May be single-valued or many-valued

Mapping z/TPFDF to SDO

- **The root DataObject has a many-valued property for each file retrieved. The values of these properties are lists of DataObjects representing subfiles.**
- **A subfile DataObject has a many-valued property for each LREC type retrieved. The values of these properties are lists of DataObjects representing LRECs.**
- **An LREC DataObject has a single-valued property for each field within the LREC.**

Mapping z/TPFDF to SDO





Programming with SDODF

- **z/TPFDF DAS**
  - readData – method to obtain a DataGraph
  - applyChanges – method to apply updates made to a DataGraph
- **SDO**
  - Methods to traverse and make changes to a DataGraph
  - These methods are not tied to z/TPFDF and are applicable for any data source.

Programming with SDODF – Scenario

- **Assume a simple database exists on z/TPFDF containing flight information and that metadata has been created.**
- **We would like to develop an SDODF application which can add, delete, and update LRECs in the top-level file.**
- **We will need to take the following steps:**
  1. Instantiate a z/TPFDF DAS.
  2. Use readData to obtain a DataGraph.
  3. Traverse the DataGraph and perform the add, update, or delete operation.
  4. Use applyChanges to make the change on z/TPFDF.

Programming with SDODF – Instantiating a DAS

- **Instantiating a z/TPFDF DAS requires:**
  - Location of an XML configuration file
  - Database name
  - Subsystem name (optional)
  - Subsystem user name (optional)
  - Username
  - Password (optional)
  - Encryption specification for password (optional)
  - User parameters (optional)

Programming with SDODF – Instantiating a DAS

The Java code below instantiates a z/TPFDF DAS for the “Flight” database with username “User1”, password “Pass”, encryption type “Unencrypted”, and no user parameters. A configuration file in the current directory named “ConfigFile.xml” is used.

```
DatabaseParameter databaseParam =
  new DatabaseParameter(  “Flight” );

AuthorizationModule authModule =
  new AuthorizationModule( “User1”, “Pass”, “Unencrypted”, null );

ZTPFDFDAS das =
  new ZTPFDFDAS( “ConfigFile.xml”, databaseParam, authModule );
```

Programming with SDODF – Using readData

- **Retrieving a DataGraph requires:**
  - z/TPFDF file to retrieve
  - Path to retrieve (optional)
  - Top-level filter (optional)
  - Interleave/Partition value (optional)
  - Search conditions (optional)
  - Properties to retrieve

Programming with SDODF – Using readData

The Java code below retrieves a DataGraph containing the Flight file. A top-level filter of 15 is specified. This top-level filter represents the algorithm string for the top-level file. A search condition is established to retrieve all LRECs with a flight number of 123. A wildcard (“*”) is used for properties so that all fields in the resulting LRECs will be returned.

```
PathParameter pathParam = new PathParameter( 15 );

SearchParameter searchParam = new SearchParameter();
searchParam.addCondition( “FlightFile”, “Fl”,
                                  SearchOperator.EQUAL, “123” );

PropertiesParameter propertiesParam = new PropertiesParameter( "*" );

DataGraph dataGraph = das.readData( “FlightFile”, pathParam,
                                                searchParam, propertiesParam );
```

Programming with SDODF – Working with a DataGraph

- **After readData is used to obtain a DataGraph, you can:**
  - Traverse the DataGraph
  - Add DataObjects
  - Delete DataObjects
  - Update DataObjects
- **Metadata defines which LREC fields are visible and which fields can be altered.**

Programming with SDODF – Traversing a DataGraph

The DataObjects within a DataGraph are accessed starting at the root DataObject. DataObject property values are retrieved using "get" methods.

```
DataObject root = dataGraph.getRootObject();

List<DataObject> subfiles = root.getList( "FlightFile" );
DataObject flightSubfile = subfiles.get( 0 );

List<DataObject> lrecs = flightSubfile.getList( "Flight" );
DataObject flightLREC = lrecs.get( 0 );
```

Programming with SDODF – Traversing a DataGraph

A DataGraph may also be traversed using XPath syntax. The code below retrieves the same flight LREC as the code on the previous slide.

```
DataObject root = dataGraph.getRootObject();

DataObject flightLREC =
  root.getDataObject( “FlightFile.0/Flight.0” );
```

Programming with SDODF – Creating an LREC

A new LREC is created by calling the createDataObject method on a subfile DataObject. The fields of the new LREC can then be set.

```
DataObject newFlightLREC =
  flightSubfile.createDataObject( “Flight” );

newFlightLREC.setString( “Fl”, “234” );
```

Programming with SDODF – Deleting an LREC

An LREC is deleted by calling the delete method on the LREC DataObject.

```
flightLREC.delete();
```

Programming with SDODF – Updating an LREC

LREC fields can be updated using the appropriate "set" methods on an LREC DataObject.

flightLREC.setString("Fl", "345");

Programming with SDODF – Using applyChanges

Once the desired changes have been made to the DataGraph, applyChanges is called to make the corresponding updates on z/TPFDF.

```
try {

  das.applyChanges( dataGraph );

} catch ( UpdatesNotAllowedException e ) {

} catch ( DataChangedException e ) {

}
```

SDODF Considerations

- **Optimistic locking is used**
- **Only one subfile can be modified per applyChanges call**
- **No support for transactions (commit/rollback)**
- **Message size is limited to 4 MB**
- **z/TPFDF DAS is not thread-safe**

Follow-up Information

- **PK60030**
  - z/TPFDF APAR (will be available)
- **PJ32720**
  - Co-requisite z/TPF APAR (will be available)
- **Java z/TPFDF DAS code will be available for download on the TPF web site as a JAR file.**
- **SDO libraries are freely available:**
  - http://www.eclipse.org/modeling/emf
  - http://incubator.apache.org/tuscany/

Trademarks

- IBM is a trademark 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.
- Notes
- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- 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.