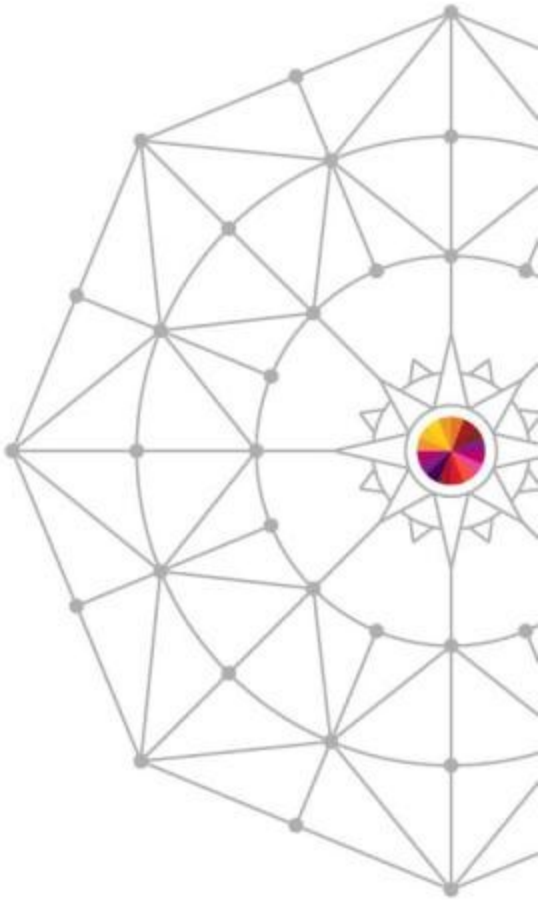
TPFUG Washington, D.C.

ADBI: An Object Oriented Interface to z/TPFDF

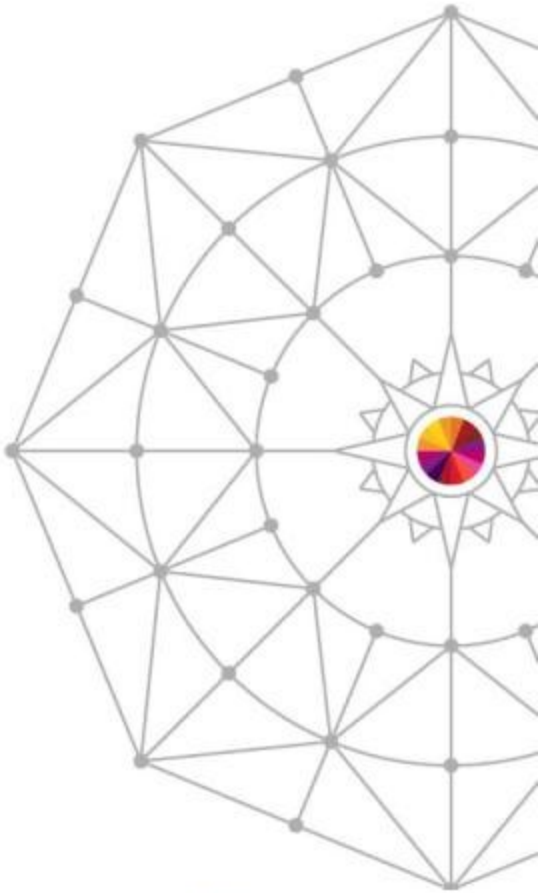
Josh Wisniewski
TPF Development Lab

Application Development Subcommittee
March 11, 2014



Disclaimer

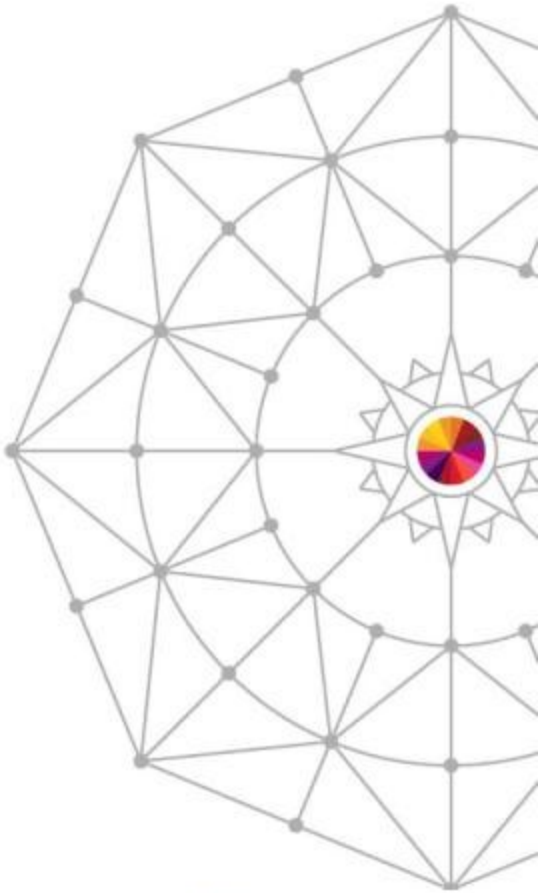
Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.



TPFUG Washington, D.C.

Why have an object oriented interface to z/TPFDF?

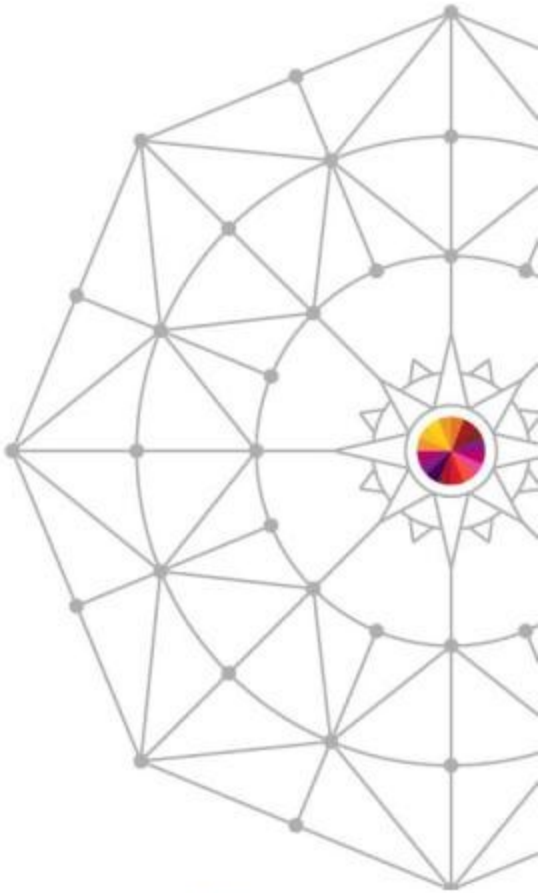
- Simplified access to existing and new z/TPFDF database content without the programmer having extensive knowledge of z/TPFDF.
- Reduced runway for new developers to become productive z/TPF developers (ie college new hires).
- Increased developer productivity due to simplified programming model.
- Object oriented coding model encourages and facilitates modularization and code reuse.
- Decreased time to market of new applications written on TPF.
- Builds upon our Big Data and Data Eventing infrastructure by further exposing data definition of logical records contained in z/TPFDF databases.



TPFUG Washington, D.C.

What is ADBI?

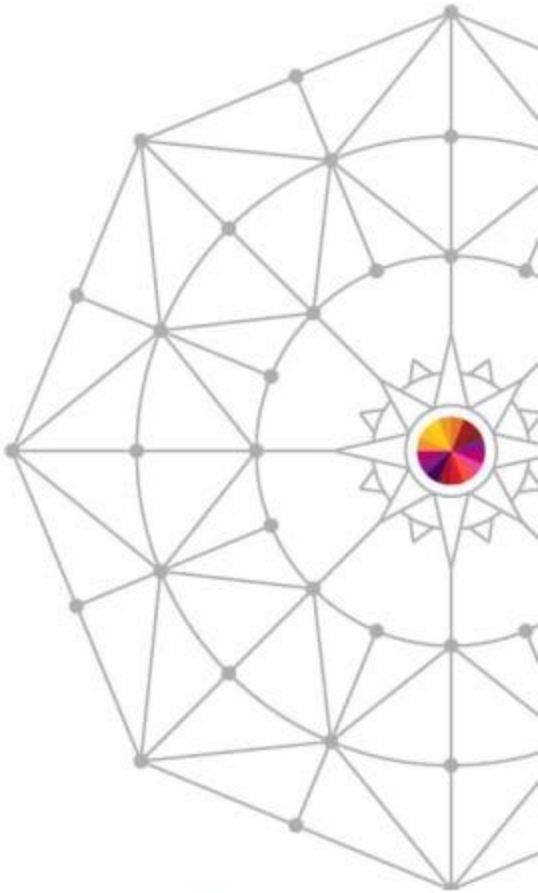
- Application Database Interface (ADBI) is an object oriented framework for accessing and manipulating data in a record based hierarchical or network model database.
- ADBI was developed as the programming model interface to WebSphere Transaction Cluster Facility (WTCF).
  - WTCF is a middleware solution for implementing record based hierarchical database applications on a distributed platform such as AIX® on IBM Power Systems.
  - The WTCF architecture is modeled after the fundamental architecture of TPF.



TPFUG Washington, D.C.

Why ADBI instead of other object oriented models?

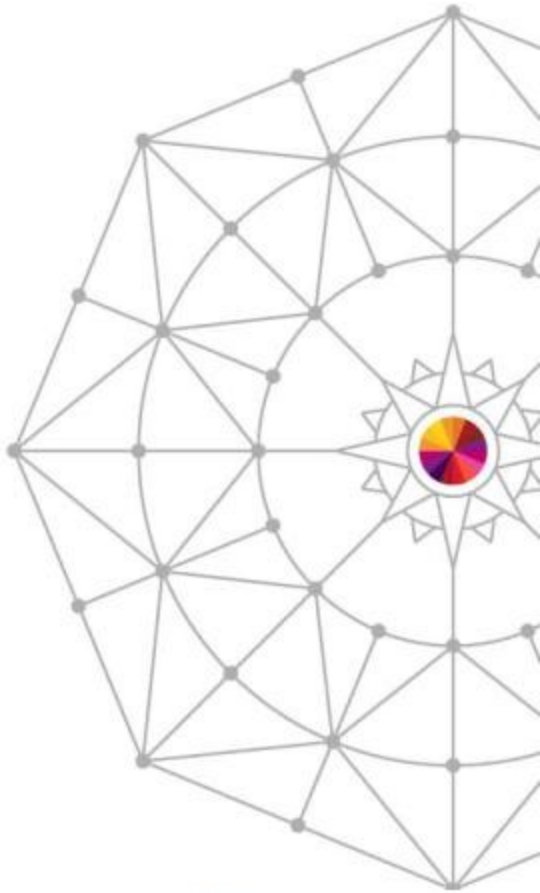
- ADBI provides a logically simplified interface to z/TPFDF data over z/TPFDF APIs and etc.
- ADBI maintains the extensive benefits of the z/TPFDF database architecture and provides access to existing databases without any change to those databases.
- A variety of WTCF POCs, speed team (college students) and other efforts have proven the viability and ease of development of the ADBI programming model.



TPFUG Washington, D.C.

What ADBI is not?

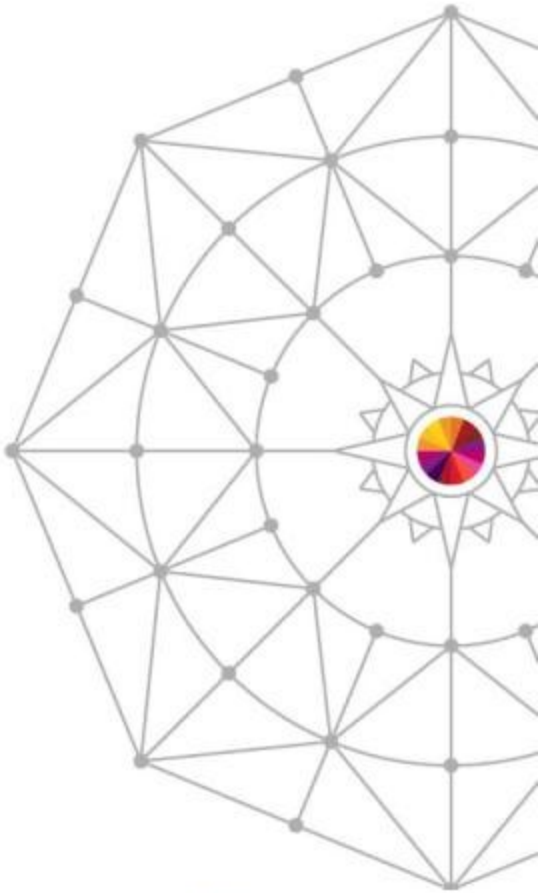
- ADBI is not a repackaging of the z/TPFDF API. ADBI will use z/TPFDF under the covers while hiding the involved details.
- ADBI will support most all z/TPFDF database types and APIs.
  - Limited to R type databases.
  - Work files will not be supported.



TPFUG Washington, D.C.

ADBI development direction

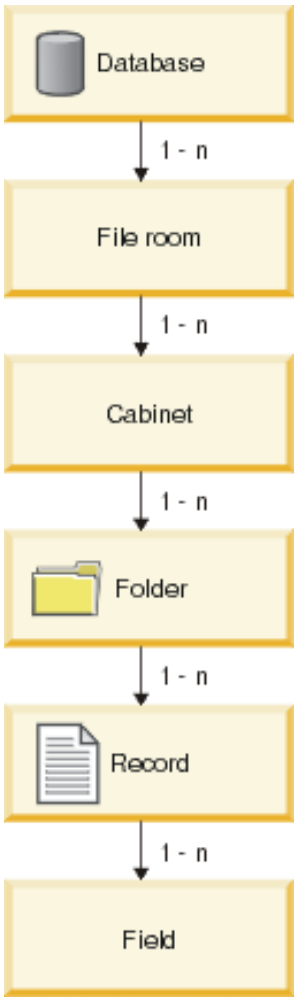
- ADBI developed applications are written native to z/TPF.
  - ADBI does not provide remote access APIs.
- Primary ADBI development focus on providing access and ability to modify existing/new z/TPFDF data.
- Potential future ADBI development will focus on simplifying the creation of new databases.



TPFUG Washington, D.C.

ADBI programming model

- ADBI defines a database hierarchy that is easily relatable.



z/TPFDF equivalent

-- All z/TPFDF files for a subsystem or SSU

-- Logical grouping of z/TPFDF files

-- z/TPFDF File

-- z/TPFDF subfile

-- Logical Record (LREC)

-- Label in LREC DSECT

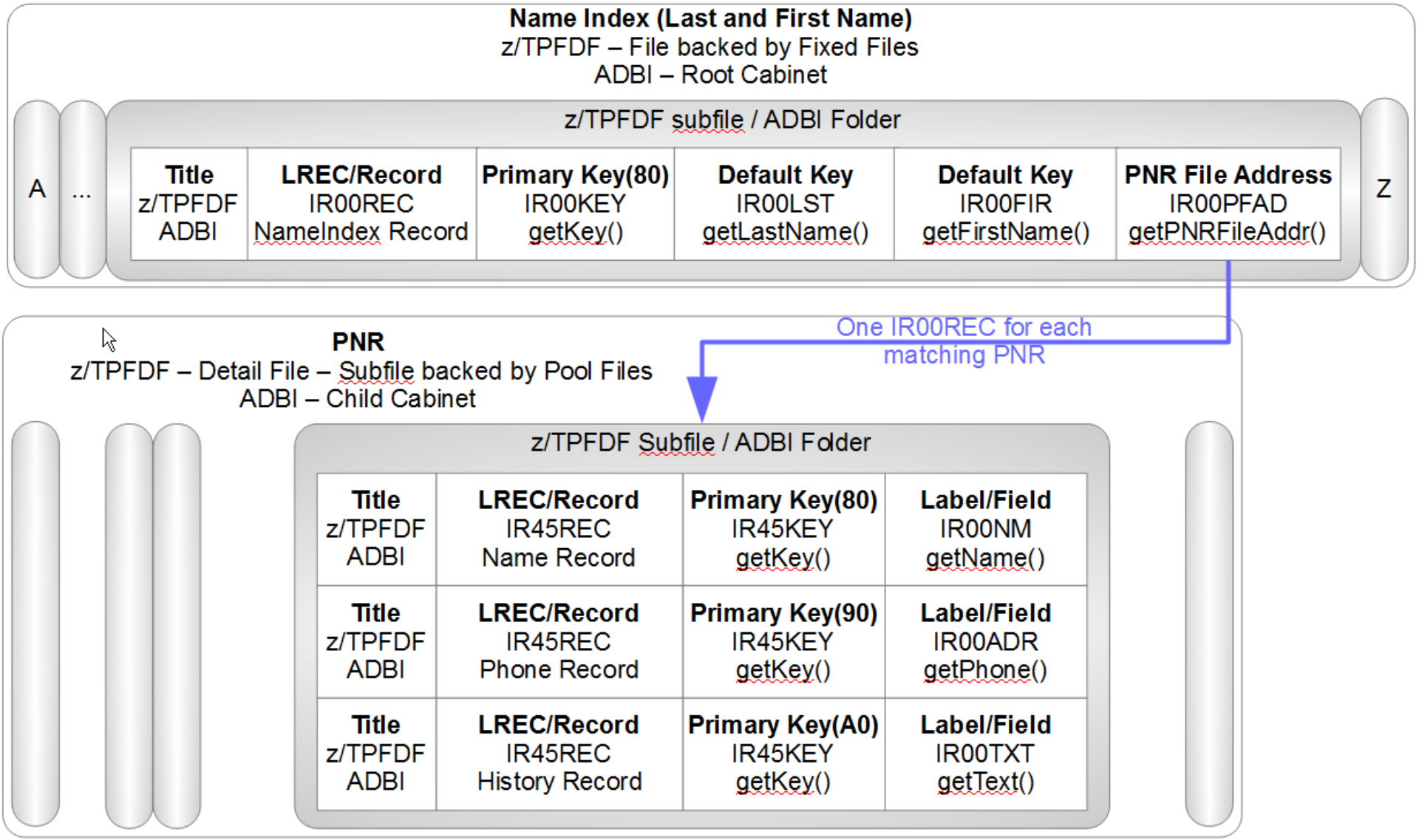
TPFUG Washington, D.C.

ADBI programming model

- ADBI defines class APIs for each level of the hierarchy and provides additional helper classes.
- Application specific record and field classes are created using generated code based on the DFDL definition of the z/TPFDF file.
- Field definitions will include the ability to define objects within objects, arrays, variable length fields and etc.

ADBI programming model example

- Modify phone number for every PNR with matching last and first name.



ADBI programming model example

```
1  ConnectionPtr conn = Database::connnect("Res1");
2  conn->startMessage("PNR", "ChangePhone");
3  FileRoomPtr fr = ADBIManager::getFileRoom(conn, "Reservations");
4  CabinetPtr pnrCab = fr->getCabinet("PNR");
5  PNRByLastNameFirstName floc(lastName, firstName);
6  FolderIteratorPtr folderIt = pnrCab->createFolderIterator(floc);
7  try
8  {
9     while (folderIt->goToNext())
10    {
11       FolderPtr pnr = pnrCab->openFolder(folderIt, ADBI_LOCK);
12       RecordLocator recLoc = RecordLocator(pnrCab, phoneRecord::RECORD_ID);
13       RecordIteratorPtr recIterPtr = pnr->createRecordIterator(recLoc);
14       while (recIterPtr->goToNext())
15       {
16          PhoneRecordPtr phonRecPtr = PhoneRecord::readRecord(pnr, recIterPtr);
17          phonRecPtr->setNumber(newNumber);
18          pnr->updateRecord(phonRecPtr);
19       }
20       pnr->close();
21    }
22 }
23 catch (ExceptionBase &e)
24    Handle Errors;
25 conn->endMessage();
26 Database::disconnect(conn);
```

TPFUG Washington, D.C.

ADBI programming model example

- **Outline of our example:**

```
SETUP
LOOP THROUGH PNR FOLDERS WITH MATCHING NAME
{
    LOOP THROUGH PHONE RECORDS
    {
        UPDATE PHONE NUMBER RECORD
    }
}
ERROR HANDLING
END
```

ADBI programming model example

- Database class provides access to the database data. ADBI applications start and end with a connect and disconnect.
- Connection objects provide the link to the database for future calls.
- The Connection startMessage and endMessage serve as bookends of a message. They are used to ensure integrity and provide debugging aids.
  - Multiple startMessage/endMessage pairs can occur during the life of an ECB to implement a long running server model.
- Database and Connection are primarily maintained for portability with WTCF and do not have a corresponding z/TPFDF equivalent.

```
 1   ConnectionPtr conn = Database::connnect("Res1");
 2   conn->startMessage();
     ...
25   conn->endMessage();
26   Database::disconnect(conn);
```

TPFUG Washington, D.C.

ADBI programming model example

- ADBIManager objects provide access to the FileRoom (a logical group of Cabinets).
- FileRoom objects provide access to the individual Cabinets (z/TPFDF File equivalent).
- Cabinet objects encapsulate a collection of Folders (z/TPFDF subfile equivalent). Cabinet objects provide the means of accessing a Folder by way of the OpenFolder API (dfopn/DBOPN equivalent).

```
3   FileRoomPtr fr = ADBIManager::
                     getFileRoom(conn, "Reservations");
4   CabinetPtr pnrCab = fr->getCabinet("PNR");
```

ADBI programming model example

- FolderIterator and FolderLocator objects are used together to traverse the cabinet hierarchy.
- FolderLocator objects describe how the cabinet hierarchy is to be traversed in descriptive terms. In our example, we are locating all PNRs that have a matching last and first name from a name index cabinet (FolderLocators are the equivalent of z/TPFDF Path, algorithm strings and etc).
- FolderIterator objects provide the ability to iterate through the Folder (z/TPFDF subfile) matches for the FolderLocator. The destination cabinet is used to create the FolderIterator.

```
5  PNRByLastNameFirstName floc(lastName, firstName);
6  FolderIteratorPtr folderIt =
       pnrCab->createFolderIterator(floc);
```

TPFUG Washington, D.C.

ADBI programming model example

- The try, catch, throw programming model is employed for error handling.

```
 7   try
 8   {
         ...
22   }
23   catch (ExceptionBase &e)
24      Handle Errors();
```

ADBI programming model example

- As previously noted, FolderIterator objects are used to traverse matches for the FolderLocator. This is accomplished by a simple test of the goToFirst or goToNext member function.
- The Folder object (z/TPFDF subfile) is then opened based on the current position of the FolderIterator. Folders can be opened for read, lock, or optimistic lock (lock only if a change occurs). The folder must be closed before the end of your message (Connection::endMessage). Locks are released when the close is issued.

```
 9  while (folderIt->goToNext())
10  {
11      FolderPtr pnr = pnrCab->openFolder(folderIt,
                                           ADBI_LOCK);
        ...
20      pnr->close();
21  }
```

ADBI programming model example

- RecordLocator and RecordIterator objects provide the ability to traverse the Records in a Folder (z/TPFDF LRECs in a subfile).
- RecordLocator objects describe how the Records will be searched. This search can be accomplished by the Record Id (z/TPFDF primary key) or other fields in the Record (z/TPFDF keys or other labels in the DSECT).
- RecordIterator objects provide the ability to iterate through the Records (z/TPFDF LREC) matches for the RecordLocator. This is accomplished by a simple test of the goToFirst or goToNext member function.

```
12  RecordLocator recLoc = RecordLocator(pnrCab,
                                         phoneRecord::RECORD_ID);
13  RecordIteratorPtr recIterPtr = pnr->
                                    createRecordIterator(recLoc);
14  while (recIterPtr->goToNext())
15  {
       ...
19  }
```

TPFUG Washington, D.C.

ADBI programming model example

- The Record object (z/TPFDF LREC) is then opened based on the current position of the RecordIterator. The Record class is created by using a code generator on the DFDL description of the LREC layout in the DBDEF DSECT. Getters and setters for each field are generated as appropriate.
- The Record object is then written back into the Folder using the updateRecord member function. Record objects can also be created and inserted into a Folder.

```
16  PhoneRecordPtr phonRecPtr =
                PhoneRecord::readRecord(pnr, recIterPtr);
17  phonRecPtr->setNumber(newNumber);
18  pnr->updateRecord(phonRecPtr);
```

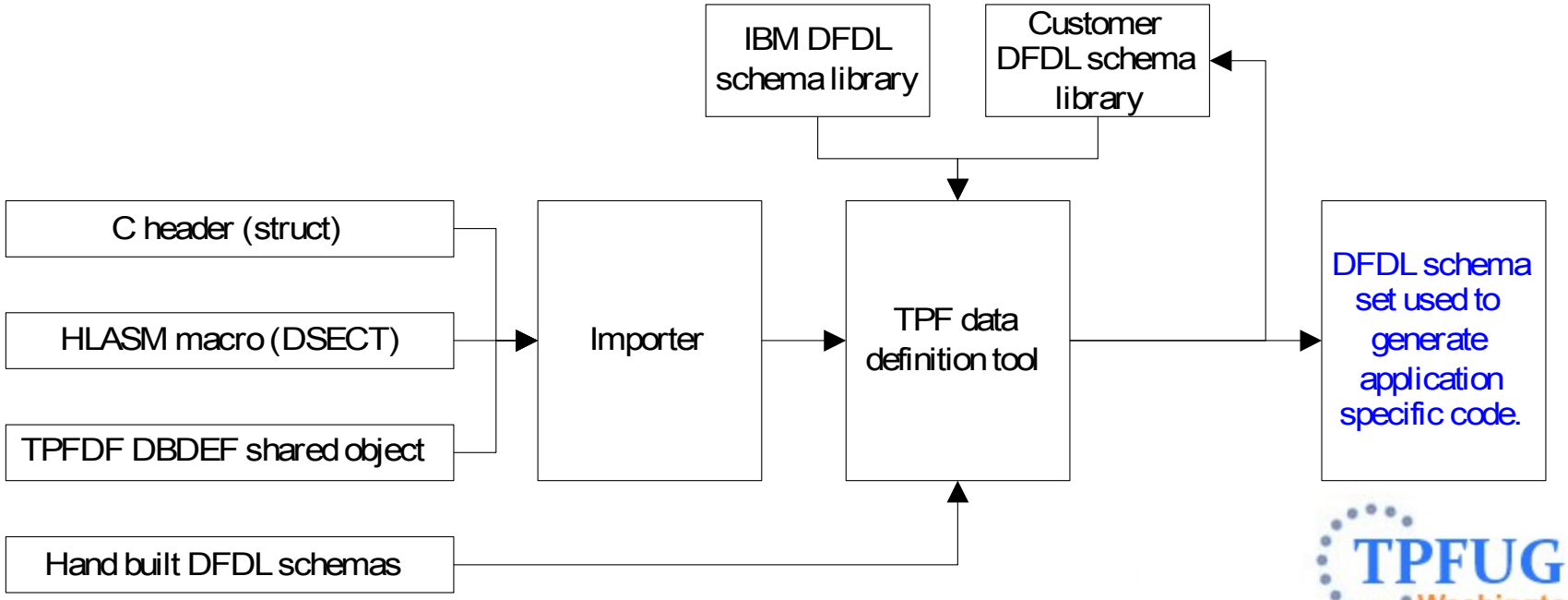
TPFUG Washington, D.C.

ADBI programming model continued

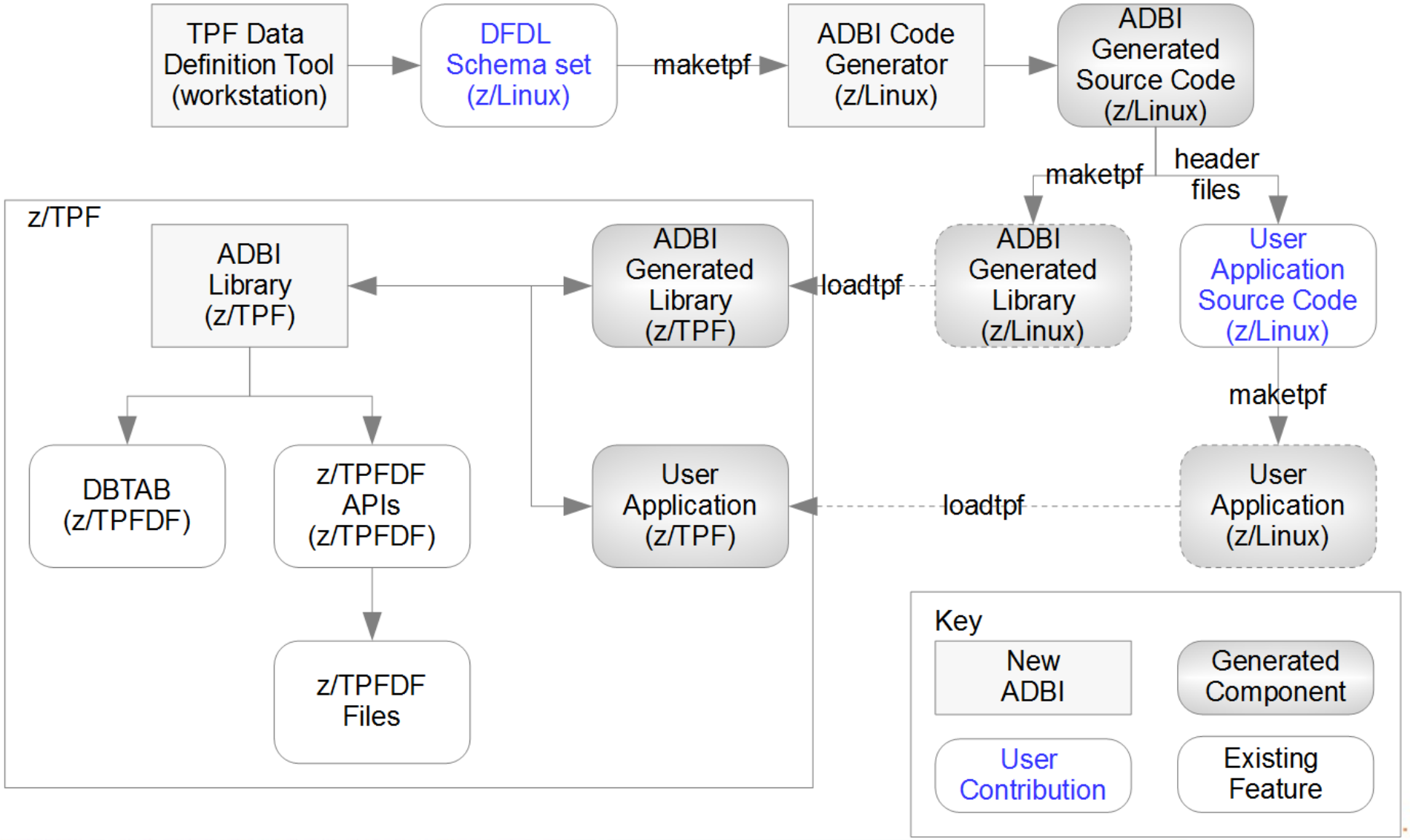
- Notice that the programming model makes extensive use of C++ namespaces to ensure name conflicts do not occur.
- Notice we did not delete any of our objects. Smart pointers are used such that after the last reference to an object is used, the object is automatically deleted.
- Notice that the programming model is built upon an object oriented exception error handling model. The ADBI ExceptionBase class can be used to catch and test for all ADBI exceptions. z/TPFDF errors will be caught by ADBI and thrown as exceptions.
- Transaction scopes will be supported.

Overall ADBI Architecture

- Bob Dryfoos' presentation "Unlocking Data on z/TPF" introduced Data Format Description Language (DFDL) and the TPF data definition tool.
- DFDL and the TPF data definition tool will be used to provide a description of the layout of z/TPFDF logical records.
- ADBI will use that description to generate application specific code.

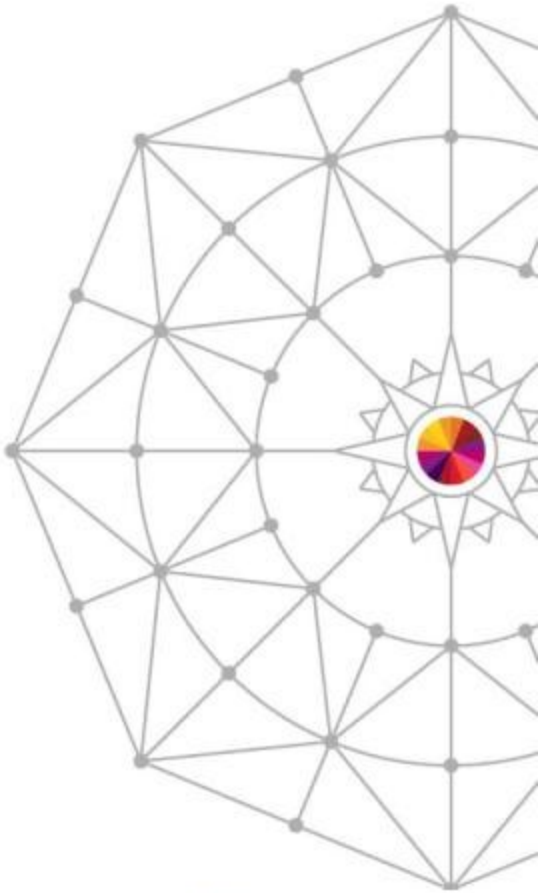


Overall ADBI Architecture



Overall ADBI Architecture

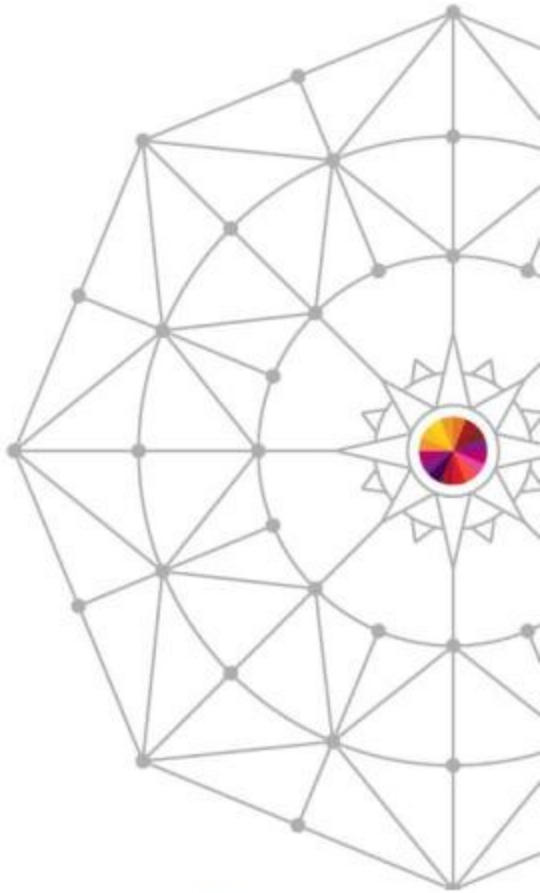
- In the initial releases of ADBI, the z/TPFDF DBDEF and DSECT definitions will serve as the master definition of the database. Any changes to the database cabinet or record layout should primarily start by making the changes there, using the importer and resolving the differences in the TPF Data Definition Tooling.
- In the future, the option may be provided for the DFDL and ADBI metadata to serve as the master definition of the database. An option would be provided to generate the required z/TPFDF DBDEF and DSECT definitions.



TPFUG Washington, D.C.

Anticipated ADBI Delivery Schedule

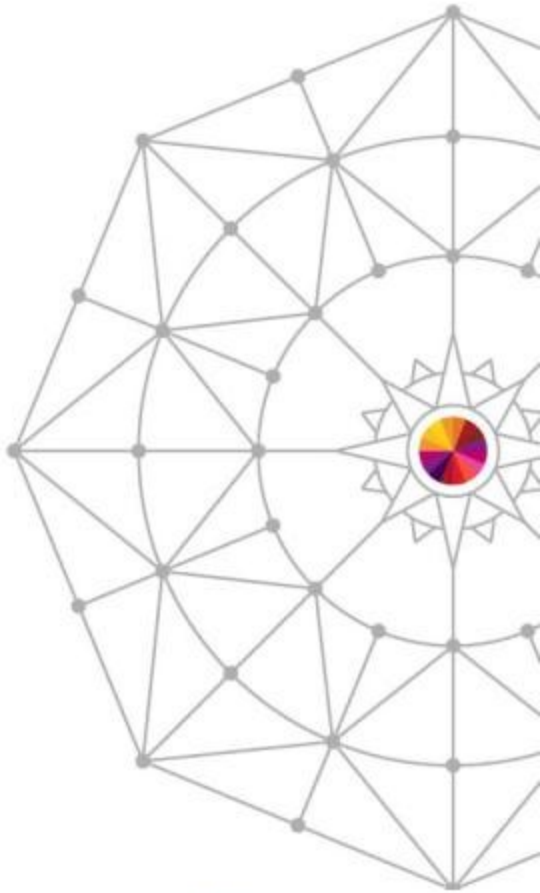
- TPF Data Definition tool initial release will be on PUT 11 in support of the Data Events project.
- ADBI access to existing/new z/TPFDF data is planned for deliver in the early PUT 12 time frame.



TPFUG Washington, D.C.

Continuing the ADBI discussion

- Questions?
- Suggestions?
- Please see myself, Bob Dryfoos, Mark Gambino or Colette Manoni if you'd like to discuss this topic in further depth. Or request a hot topic.



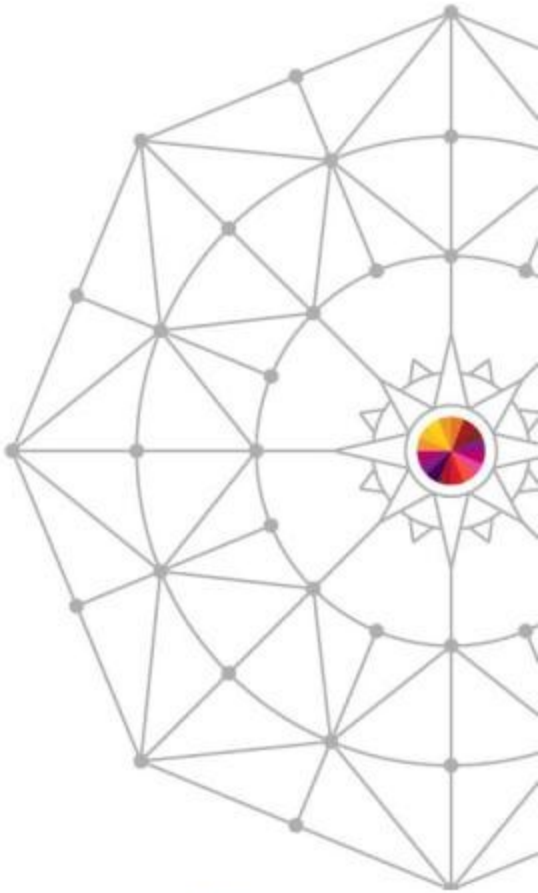
TPFUG Washington, D.C.

Trademarks

- IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at " Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

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.



TPFUG Washington, D.C.