

2012

IBM System z Technical University

Enabling the infrastructure for smarter computing

Introduction to new Features in z/VSE Connectors

zDG08

Ingo Franzki – ifranzki@de.ibm.com



Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*, AS/400®, e business(logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

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.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* All other products may be trademarks or registered trademarks of their respective companies.

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.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

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.

CONNECTOR DR

STOP

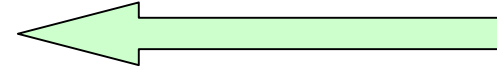
BOBBY LO



Agenda

§ **z/VSE V5.1 + PTFs Connector Enhancements**

§ z/VSE Database Call Level Interface



§ **z/VSE V5.1 Connector Enhancements**

§ VSE Script Connector: SYSIPT Variables Support

§ VSE Script Connector: New functions

§ VSE Script Connector: Logging of script input and output

§ VSAM Redirector: MapperConfigGUI Enhancements

§ VSE Connector Client & Server: LDAP signon support

§ VSE Connector Client & Server: LIBR DATA=YES

§ **z/VSE V4.3 Connector Enhancements**

§ POWER Output Generation Messages and exploitation in Java-based Connector

§ Decimal Position support for Java-based Connector

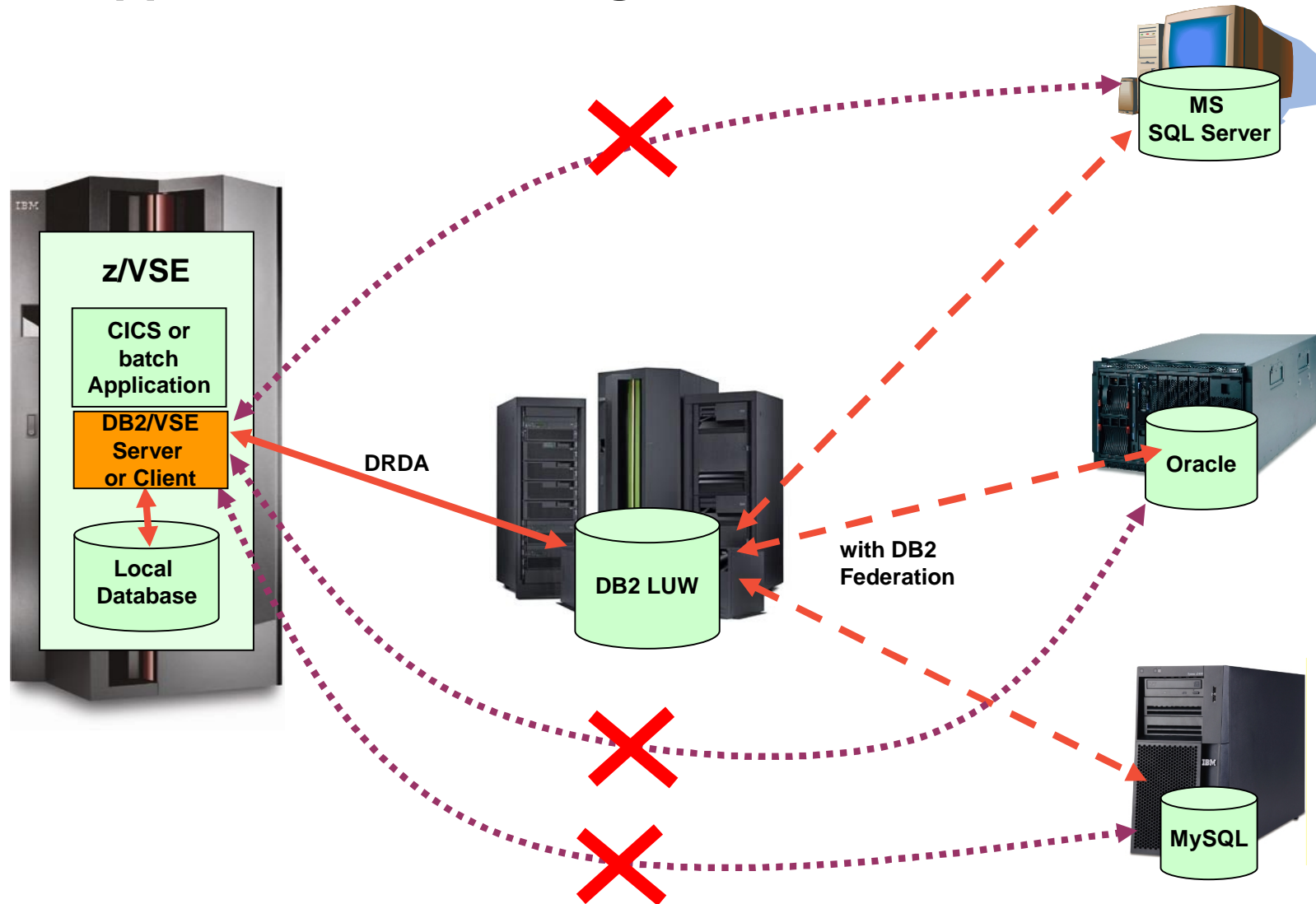
§ EXCPAD for VSAM Redirector

§ Redirector Trace activation via VSAM SNAP

§ **New and updated Tools**

§ New Tool: Virtual z/VSE FTP Daemon

z/VSE applications accessing Databases



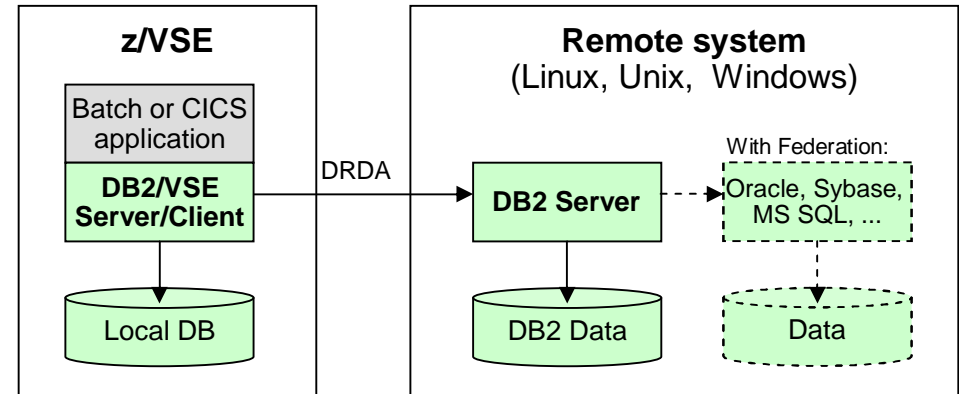
Options for using Databases with z/VSE applications

§ DB2/VSE or DB2/VM Server

- § Local database residing in z/VSE or z/VM
- § Lacks support of modern SQL functionality
- § Only quite old SQL level supported

§ DB2/VSE Client Edition

- § Remote database (on Linux, Windows, Unix)
- § Communication via DRDA protocol
- § Same old SQL level supported as DB2/VSE Server
- § Can not use modern SQL functionality provided by DB2 LUW
- § Can only access remote DB2 databases
 - ü Other databases (e.g. MS SQL Server, Oracle, etc) can only be accessed through IBM InfoSphere Federation Server



§ VSAM Redirector

- § Primarily used to keep Databases in sync with VSAM data
- § Also allows migration from VSAM to database

§ **New:** z/VSE Database Call Level Interface

- § Allows z/VSE applications to access a relational database on any suitable database server
 - ü IBM DB2, IBM Informix, Oracle, MS SQL Server, MySQL, etc.
- § Utilize advanced database functions and use SQL statements provided by modern database products



z/VSE V5.1 + PTFs: z/VSE Database Call Level Interface (DBCLI)

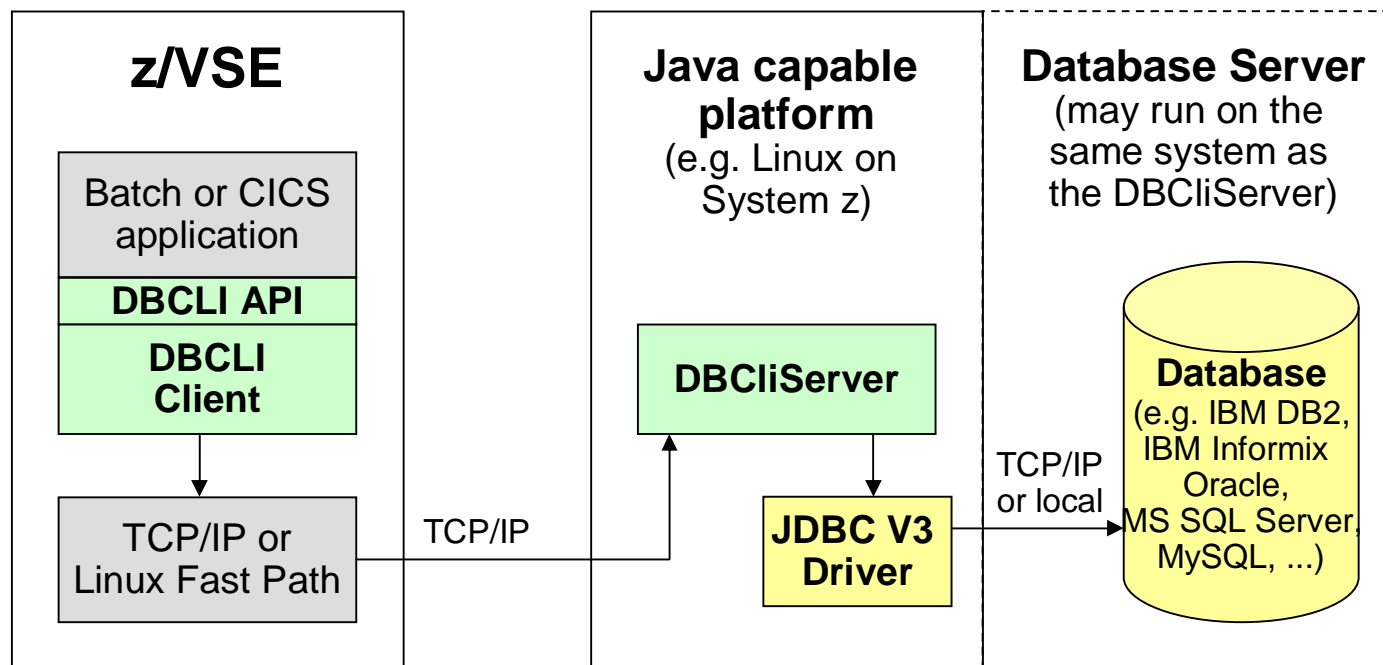
§ Allows z/VSE applications to access a relational database on any suitable database server



§ IBM DB2, IBM Informix, Oracle, MS SQL Server, MySQL, etc.

à The database product must provide a JDBC driver that supports JDBC V3.0 or later

à Utilize advanced database functions and use SQL statements provided by modern database products



Requires z/VSE 5.1 plus PTFs (UK78892 and UK78893)

z/VSE V5.1 + PTFs: z/VSE Database Call Level Interface (DBCLI)

§ The z/VSE Database Call Level Interface provides a programming interface (API)

- § Call interface for use with COBOL, PL/1, Assembler, C and REXX
- § Can be used in Batch as well as in CICS TS applications
- § Supports LE enabled as well as non-LE environments (Assembler, REXX)

§ It provides callable functions for

- § Initializing and Terminating the API Environment
- § Connecting and Disconnecting to/from the DBCLI Server and the Database
- § Executing SQL Statements
- § Retrieving query results through cursors
- § Handling of Logical Units of Work (Transactions)
- § Retrieving Database Meta Data



§ The API is not compatible with DB2/VSE's EXEC DB2 preprocessor interface

- § But it provides similar functions
- § The API is similar to the ODBC programming interface

§ A COBOL example is provided to show how DBCLI can be used in your applications

Using the DBCLI API in your applications

§ Using DBCLI in COBOL:

§ The COBOL copybook IESDBCOB contains common declarations

```
CALL 'IESDBCLI' USING FUNCTION ENV-HANDLE parm1 parm2 ... parmN RETCODE.
```

§ Using DBCLI in PL/I

§ The PL/I copybook IESDBPL1 contains common declarations

```
CALL IESDBCLI (FUNCTION, ENV_HANDLE, parm1, parm2, ..., parmN, RETCODE);
```

§ Using DBCLI in C

§ The C header file IESDBC.h contains common declarations

```
IESDBCLI (function, &env_handle, &parm1, &parm2, ..., &parmN, &retcode);
```

§ Using DBCLI in Assembler

§ The Assembler macro IESDBASM contains common declarations

```
CALL IESDBCLI, (FUNCTION, ENV_HANDLE, parm1, parm2, ..., parmN, RETCODE), VL
```

§ The following register conventions apply:

- ü Register 0, 1, 14, and 15 are used by the interface and must be saved prior to invocation
- ü Register 13 must point to a 72-byte save area provided by the caller

§ Using DBCLI in REXX

```
ADDRESS LINKPGM "IESDBCLA FUNCTION ENV_HANDLE parm1 parm2 ... parmN RETCODE"
```

§ All parameters must be initialized with a value of the appropriate length before calling the DBCLI API. This is especially true for output parameters.

§ Fullword binary variables must be initialized to contain 4 bytes (for example, VARIABLE = D2C(0,4))

§ Since the variable is expected to contain a value in binary representation, you must convert the value from the REXX string representation into the binary representation and vice versa using the REXX functions C2S and D2C



DBCLI Concepts: Initializing and terminating the environment

When using the API provided by the DBCLI client, you must:

§ **Initialize** the API environment by calling the **INITENV** function before calling any other function

§ The INITENV function allocates an **environment handle** that you must pass to all subsequent functions

§ You can have only one active environment at a time in your program

§ **Terminate** the API environment (at the end of your program) by calling the **TERMENV** function

§ The TERMENV function frees all resources allocated by the DBCLI code

§ The TERMENV function will also close any "left over" connections or statements

§ After the TERMENV function, the environment handle is no longer valid

§ You can set and get various **attributes** on the **environment level**

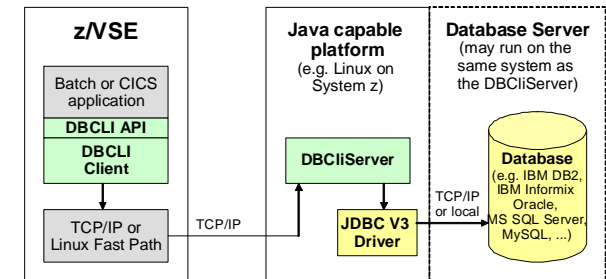
§ You do so by calling the **SETENVATTR** or **GETENVATTR** function



DBCLI Concepts: Connect to the DBCLI Server and Database

To access a Database, you must connect to the DBCLI server and the Vendor database

- § You connect to the DBCLI server (DBCliServer) and the database by calling the **CONNECT** function
- § You must supply the:
 - § IP address or hostname of DBCliServer
 - § Alias name of the database or the JDBC URL to which you wish to connect
 - § User-ID and Password to authenticate with the database
- § The **CONNECT** function allocates a **connection handle** that you must pass to all subsequent functions that require a connection
 - § You can have multiple connections to the same or different DBCLI servers and databases at a time
 - § Each connection is represented by its own connection handle
- § When you are finished working with a database, you must disconnect from the database and the DBCLI server (DBCliServer) by calling the **DISCONNECT** function
 - § The DISCONNECT function frees the connection handle and all left over statements (if any) that you have allocated using this connection



DBCLI Concepts: Logical Units of Work (Transactions)



Per default, a connection operates in **transaction mode**:

- § Any database updates that you perform are contained in a **logical unit of work**
- § You can **end a logical unit of work** by calling the COMMIT or ROLLBACK functions:
 - § The **COMMIT** function commits all changes done since the beginning of the logical unit of work and starts a new logical unit of work
 - § The **ROLLBACK** function rolls back (reverts) all changes since the beginning of the logical unit of work or up to a savepoint

- § Usually, you should **explicitly call the COMMIT function at the end of the program**.
- § If you do not call the COMMIT function, DBCLIServer will **automatically commit** all changes
 - § **if you gracefully close the connection** by calling the **DISCONNECT** function
- § If the **connection is dropped** (for example, because the program abends), the DBCLI server **rolls back all changes** done since the beginning of the last logical unit of work

- § You can set a connection into **auto-commit mode**
 - § In auto-commit mode, every SQL statement is treated as **its own logical unit of work** and is **committed automatically** when the statement execution is complete.
 - û Therefore, you do not have to call the COMMIT or ROLLBACK functions.
 - § You set a connection into auto-commit mode by calling the **SETCONNATTR** function to set the **CONNATTR-AUTO-COMMIT** attribute to TRUE

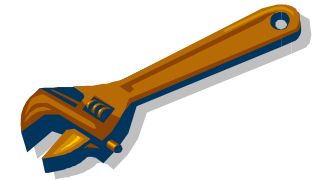
DBCLI Concepts: Preparing SQL Statements

In order to execute an SQL statement, you must first **prepare the SQL statement**

§ During preparation, the database will **pre-compile the SQL statement** and create an **access plan** for the statement

§ The access plan is kept as long as the statement exists

§ You can then **execute** the statement **as many times** as you want



§ The **PREPARESTATEMENT** function prepares an SQL statement for execution

§ It allocates a **statement handle** that represents the statement

§ An application can have multiple prepared statements at a time

§ The **PREPARECALL** function prepares a **stored procedure call** statement for execution

§ SQL statements may contain **parameters** that are **evaluated at execution time**

§ Parameters are marked by a **question mark (?)** within the SQL statement

§ The parameters are **numbered in order of appearance**, starting with 1

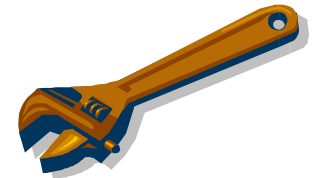
§ After preparing, the application can **bind host variables to the parameters** using the **BINDPARAMETER** function

§ When the statement is later **executed**, the **content of the host variables is used** and sent to the database.

DBCLI Concepts: Preparing SQL Statements

In order to execute an SQL statement, you must first **prepare the SQL statement**

- § During preparation, the database will **pre-compile the SQL statement** and create an **access plan** for the statement
 - § The access plan is kept as long as the statement exists
 - § You can then **execute** the statement **as many times** as you want
- § The **PREPARESTATEMENT** function prepares an SQL statement for execution
 - § It allocates a **statement handle** that represents the statement



SQL statements may contain parameters that are evaluated at execution time

- § Parameters are marked by a **question mark (?)** within the SQL statement

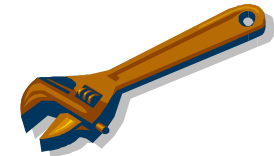
```
SELECT * FROM EMPLOYEE WHERE EMPNO>? AND SALARY>?
                                Parameter 1  Parameter 2
```

- § The parameters are **numbered in order of appearance**, starting with 1
- § When using DB2/VSE preprocessor, above statement would look like:


```
SELECT * FROM EMPLOYEE WHERE EMPNO>:empno AND SALARY>:salary
```

- § The application **binds host variables to the parameters** using the **BINDPARAMETER** function
 - § When the statement is later executed, the **content of the host variables is used** and sent to the database
 - § You also specify the **data type** and **length** of the variable with the **BINDPARAMETER** call
 - § **Indicator variables** are used to determine if the parameter value is **NULL**

DBCLI Concepts: Executing statements



To execute a statement, you must call the **EXECUTE** function

- § If the statement was an SQL **update statement**, you can retrieve the number of rows updated using the **GETUPDATECOUNT** function or the **UPDATE-COUNT** parameter at the EXECUTE function

- § If the statement was a SQL **query statement**, you can **use a cursor** to retrieve (fetch) the result rows and columns
 - § A statement can provide multiple results (mostly stored procedures)
 - § To retrieve the additional results you must call the **GETMORERESULTS** function
 - § The GETMORERESULTS function will move to the next available cursor or update count

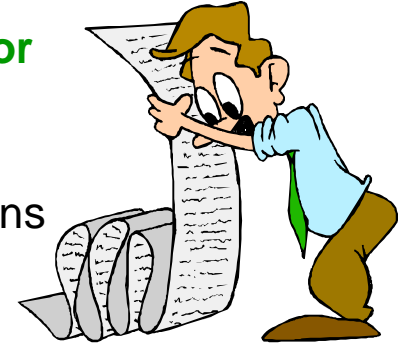
- § If the statement was a stored procedure call, **output parameters** are updated with the data passed back by the stored procedure

- § When you no longer need a statement, you must close it by calling the **CLOSESTATEMENT** function:
 - § The CLOSESTATEMENT function frees the statement handle and closes all cursors (if any) that may still be open from the last statement execution
- § The statement handle is no longer valid after the CLOSESTATEMENT function

DBCLI Concepts: Result sets and Cursors

The execution on an **SQL query** returns a result in form of a **cursor**

- § A cursor allows you to retrieve (fetch) the result rows and columns
 - § You can use the **GETNUMCOLUMNS** and **GETCOLUMNINFO** functions to obtain detailed information about the cursor's columns
 - § The **columns are numbered** in order of appearance, starting at 1



- § To fetch the result rows using the cursor, you must first **bind host variables to the columns** of interest
 - § You bind host variables to the columns of interest by calling the **BINDCOLUMN** function
 - § If the **FETCH** function is called later on, the host variables will be updated with the contents of the column in the row that has been fetched
- § Per default, the **FETCH** function processes the cursor **from the beginning to the end**
 - § You may **reposition with a cursor**
 - ü Providing the database supports this and you have created the statement using the appropriate type
- § Repositioning can be performed using either the:
 - § **FETCH** function with operations **FETCH-PREVIOUS**, **FETCH-FIRST**, **FETCH-LAST**, **FETCH-ABSOLUTE** or **FETCH-RELATIVE**.
 - § **SETPOS** function

DBCLI Concepts: Database Meta Data

The DBCLI interface allows you to retrieve **meta data** from the database

§ This includes functions to get a **list of tables**, indexes, keys, **columns of a table**, and so on

§ This information is typically stored in system catalog tables in the database.

§ You can also execute regular SELECT statements against the system catalog tables, but this requires that you know which database system and vendor you are using

§ System catalog tables are vendor- and database-specific

§ The DBCLI interface provides a **set of database independent functions** to retrieve meta data information.

§ These functions are prefixed with 'DB'

§ The function DBTABLES for example retrieves a list of tables available in the database

§ Please note that some databases may not support all of the meta data functions



COBOL Example

```
PROCEDURE DIVISION.  
MAIN-PROGRAM.  
    DISPLAY 'COBSAMPL STARTED'.  
*  
* Perform the INITENV call  
*  
    MOVE 'SOCKET00' TO TCPNAME.  
    MOVE 'EZASOH99' TO ADSNAME.  
    CALL 'IESDBCLI' USING FUNC-INITENV ENV-HANDLE  
        TCPNAME ADSNAME RETCODE.  
    DISPLAY 'RETCODE OF INITENV IS ' RETCODE.  
    IF RETCODE > EOK THEN  
        PERFORM CHECK-ERROR  
    END-IF.
```

← Initialize the environment

COBOL Example

```

PROCEDURE DIVISION.
MAIN-PROGRAM.
  DISPLAY 'COBSAMPL STARTED'.
*
*
* * Connect to the DBCLI server and the database
*
  MOVE '9.152.2.70' TO SERVER.
  MOVE 10 TO SERVER-LEN.
  MOVE 16178 TO PORT.
  MOVE 'SAMPLE' TO DBNAME.
  MOVE 6 TO DBNAME-LEN.
  MOVE 'dbuserid' TO USERID.
  MOVE 8 TO USERID-LEN.
  MOVE 'password' TO PASSWD.
  MOVE 8 TO PASSWD-LEN.
  CALL 'IESDBCLI' USING FUNC-CONNECT ENV-HANDLE CON-HANDLE
    SERVER SERVER-LEN PORT DBNAME DBNAME-LEN
    USERID USERID-LEN PASSWD PASSWD-LEN
    RETCODE.
  DISPLAY 'RETCODE OF CONNECT IS ' RETCODE.
  IF RETCODE > EOK THEN
    PERFORM CHECK-ERROR
  END-IF.

```

IP or hostname of
DBCLI Server

Database alias name
User-ID & Password

Connect to the
DBCLI Server
and the Database

COBOL Example

```

PROCEDURE DIVISION.
MAIN-PROGRAM.
    DISPLAY 'COBSAMPL STARTED'.
*
*
* * Connect to the DBCLI server and the database
*
    MOVE '9.152.2.70' TO SERVER.
    MOVE 10 TO SERVER-LEN.
    MOVE 16178 TO PORT.
    MOVE 'SAMPLE' TO DBNAME.
    MOVE 6 TO DBNAME-LEN.
*
* Prepare an SQL statement for later execution
*
    MOVE 'SELECT * FROM EMPLOYEE WHERE EMPNO>? AND SALARY>?'
      TO SQL.
    MOVE LENGTH OF SQL TO SQL-LEN.
    CALL 'IESDBCLI' USING FUNC-PREPARESTATEMENT ENV-HANDLE
      CON-HANDLE STMT-HANDLE SQL SQL-LEN
      CURSOR-TYPE-SCROLL-INSENSITIVE CURSOR-CONCUR-READ-ONLY
      HOLD-CURSORS-OVER-COMMIT RETCODE.
    DISPLAY 'RETCODE OF PREPARESTATEMENT IS ' RETCODE.
    IF RETCODE > EOK THEN
      PERFORM CHECK-ERROR
    END-IF.
  
```

SQL Statement
Containing Parameter
Markers ('?')

Prepare an
SQL Statement
for later execution

COBOL Example

```

PROCEDURE DIVISION.
MAIN-PROGRAM.
  DISPLAY 'COBSAMPL STARTED'.
*
*
* * Connect to the DBCLI server and the database
*
  MOVE '9.152.2.70' TO SERVER.
  MOVE 10 TO SERVER-LEN.
  MOVE 16178 TO PORT.
  MOVE 'SAMPLE' TO DBNAME.
  MOVE 6 TO DBNAME-LEN.
*
* Prepare SQL statement for later execution
*
* Bind the EMPNO host variable (Text) to parameter 1.
* Here we specify the optional codepage parameter to
* send the text data in the desired codepage.
*
  MOVE 1 TO PARM-IDX.
  MOVE LENGTH OF EMPNO TO EMPNO-LEN.
  MOVE 'CP1047' TO CODEPAGE.
  MOVE LENGTH OF CODEPAGE TO CODEPAGE-LEN.
  CALL 'IESDBCLI' USING FUNC-BINDPARAMETER ENV-HANDLE
    STMT-HANDLE PARM-IDX NATIVE-TYPE-STRING
    EMPNO EMPNO-LEN EMPNO-IND
    CODEPAGE CODEPAGE-LEN RETCODE.
  DISPLAY 'RETCODE OF BINDPARAMETER IS ' RETCODE.
  IF RETCODE > EOK THEN
    PERFORM CHECK-ERROR
  END-IF.

```

Bind host variable
"EMPNO"
to parameter
number 1
as STRING

COBOL Example

```

PROCEDURE DIVISION.
MAIN-PROGRAM.
    DISPLAY 'COBSAMPL STARTED'.
*
*
* * Connect to the DBCLI server and the database
*
    MOVE '9.152.2.70' TO SERVER.
    MOVE 10 TO SERVER-LEN.
    MOVE 16178 TO PORT.
    MOVE 'SAMPLE' TO DBNAME.
    MOVE 6 TO DBNAME-LEN.
*
* Prepare SQL statement for later execution
*
* Bind the EMPNO host variable (Text) to parameter 1.
* Here we specify the optional codepage parameter to
* send the text data in the desired codepage.
*
* Bind the SALARY host variable (packed decimal) to parameter 2.
* Here we specify the decpos parameter to indicate that we
* want to send the numeric data with 2 implied decimal places.
*
    MOVE 2 TO PARM-IDX.
    MOVE LENGTH OF SALARY TO SALARY-LEN.
    MOVE 2 TO DECPOS.
    CALL 'IESDBCLI' USING FUNC-BINDPARAMETER ENV-HANDLE
        STMT-HANDLE PARM-IDX NATIVE-TYPE-PACKED-SIGNED
        SALARY SALARY-LEN SALARY-IND
        DECPOS RETCODE.
    DISPLAY 'RETCODE OF BINDPARAMETER IS ' RETCODE.
    IF RETCODE > EOK THEN
        PERFORM CHECK-ERROR
    END-IF.

```

Bind host variable
“SALARY”
to parameter
number 2
as PACKED decimal

COBOL Example

```

PROCEDURE DIVISION.
MAIN-PROGRAM.
    DISPLAY 'COBSAMPL STARTED'.
*
*
* * Connect to the DBCLI server and th
*
    MOVE '9.152.2.70' TO SERVER.
    MOVE 10 TO SERVER-LEN.
    MOVE 16178 TO PORT.
    MOVE 'SAMPLE' TO DBNAME.
    MOVE 6 TO DBNAME-LEN.
*
* Prepare SQL statement.
*
* Bind the EMPNO host variable
* Here we specify the option
* send the text data in the
*
* Bind the SALARY host
* Here we specify the d
* want to send the nume
*
    MOVE 2 TO PARM-IDX
    MOVE LENGTH OF SALARY TO SALARY-LEN.
    MOVE 2 TO DECPOS.
    CALL 'IESDBCLI' USING FUNC-BINDPARAMETER ENV-HANDLE
        STMT-HANDLE PARM-IDX NATIVE-TYPE-PACKED-SIGNED
        SALARY SALARY-LEN SALARY-IND
        DECPOS RETCODE.
    DISPLAY 'RETCODE OF BINDPARAMETER IS ' RETCODE.
    IF RETCODE > EOK THEN
        PERFORM CHECK-ERROR
    END-IF.
*
* Set the host variables values and corresponding indicator
* variables:
*
    MOVE '000030' TO EMPNO.
    MOVE INDICATE-NOTNULL TO EMPNO-IND.
    MOVE 01000.00 TO SALARY.
    MOVE INDICATE-NOTNULL TO SALARY-IND.
*
* Execute the statement. This will use the values of the
* host variables for the parameters.
*
    CALL 'IESDBCLI' USING FUNC-EXECUTE ENV-HANDLE
        STMT-HANDLE RETCODE.
    IF RETCODE > EOK THEN
        PERFORM CHECK-ERROR
    END-IF.
    DISPLAY 'RETCODE OF EXECUTE IS ' RETCODE.
    IF RETCODE > EOK THEN
        PERFORM CHECK-ERROR
    END-IF.

```

Assign values

Execute the statement

COBOL Example

```

PROCEDURE DIVISION.
MAIN-PROGRAM.
    DISPLAY 'COBSAMPL STARTED'.
*
*
* * Connect to the DBCLI server and th
*
    MOVE '9.152.2.70' TO SERVER.
    MOVE 10 TO SERVER-LEN.
    MOVE 16178 TO PORT.
    MOVE 'SAMPLE' TO DBNAME.
    MOVE 6 TO DBNAME-LEN.
*
* Pre
*
* Bind the EMPNO host variab
* Here we specify the optio
* send the text data in the
*
* Bind the SALARY host
* Here we specify the d
* want to send the nume
*
    MOVE 2 TO PARM-IDX
    MOVE LENGTH OF SALARY TO SALARY-LEN.
    MOVE 2 TO DECPOS.
    CALL 'IESDBCLI' USING FUNC-BINDPARAMETER ENV-HANDLE
        STMT-HANDLE PARM-IDX NATIVE-TYPE-PACKED-SIGNED
        SALARY SALARY-LEN SALARY-IND
        DECPOS RETCODE.
    DISPLAY 'RETCODE OF BINDPARAMETER IS ' RETCODE.
    IF RETCODE > EOK THEN
        PERFORM CHECK-ERROR
    END-IF.
*
* Set the host variables values and coresponding indicator
* variables:
*
* Bind host variable FIRSTNAME (text) to the column 2.
* Here we do not specify the codepage parameter so we
* receive the text data in the default codepage.
*
    MOVE 2 TO COL-IDX.
    MOVE LENGTH OF FIRSTNAME TO FIRSTNAME-LEN.
    CALL 'IESDBCLI' USING FUNC-BINDCOLUMN ENV-HANDLE
        STMT-HANDLE COL-IDX NATIVE-TYPE-STRING
        FIRSTNAME FIRSTNAME-LEN FIRSTNAME-IND
        RETCODE.
    DISPLAY 'RETCODE OF BINDCOLUMN IS ' RETCODE.
    IF RETCODE > EOK THEN
        PERFORM CHECK-ERROR
    END-IF.
    IF RETCODE > EOK THEN
        PERFORM CHECK-ERROR
    END-IF.

```

Bind host variable
"FIRSTNAME" to
result set column
number 2

COBOL Example

```

PROCEDURE DIVISION.
MAIN-PROGRAM.
    DISPLAY 'COBSAMPL STARTED'.
*
* * Connect to the DBCLI server and th
*
    MOVE '9.152.2.70' TO SERVER.
    MOVE 10 TO SERVER-LEN.
    MOVE 16178 TO PORT.
*
*
*
*
*
*
*
*
*
* Bind the EMPNO host variab
* Here we specify the optio
* send the text data in the
*
* Bind the SALARY host
* Here we specify the d
* want to send the nume
*
    MOVE 2 TO PARM-IDX
    MOVE LENGTH OF SALARY TO
    MOVE 2 TO DECPOS.
    CALL 'IESDBCLI' USING FUNC
    STMT-HANDLE PARM-IDX
    SALARY SALARY-LEN SA
    DECPOS RETCODE.
    DISPLAY 'RETCODE OF BINDE
    IF RETCODE > EOK THEN
        PERFORM CHECK-ERROR
    END-IF.
*
*
* Set the host variables values and coresponding indicator
* variables:
*
*
* Bind host variable FIRSTNAME (text) to the column 2.
* He
*
* Fetch all available rows and display the data.
* Since columns may be NULL we check the indicator variables.
* FETCH without an operation argument means FETCH NEXT.
* E
*
    PERFORM WITH TEST AFTER UNTIL RETCODE > EOK
        CALL 'IESDBCLI' USING FUNC-FETCH ENV-HANDLE
            STMT-HANDLE RETCODE
        DISPLAY 'RETCODE OF FETCH IS ' RETCODE
        IF RETCODE > EOK AND RETCODE NOT = ENOMOREDATA THEN
            PERFORM CHECK-ERROR
        END-IF
        IF RETCODE = EOK THEN
            DISPLAY 'ROW DATA INFO FOR ROW NUMBER ' ROW-NUMBER
            IF EMPNO-IND = INDICATE-NULL THEN
                DISPLAY ' EMPNO IS NULL'
            ELSE
                DISPLAY ' EMPNO IS EMPNO'
            END-IF
            IF FIRSTNAME-IND = INDICATE-NULL THEN
                DISPLAY ' FIRSTNAME IS NULL'
            ELSE
                DISPLAY ' FIRSTNAME IS FIRSTNAME'
            END-IF
        END-IF
    END-IF.

```

Fetch all rows

DBCLI Concepts: Hints & Tips



- § The DBCLI code is **CICS-aware**
 - § If running under CICS, any memory allocations are performed using EXEC CICS GETMAIN instead of using the GETVIS macro

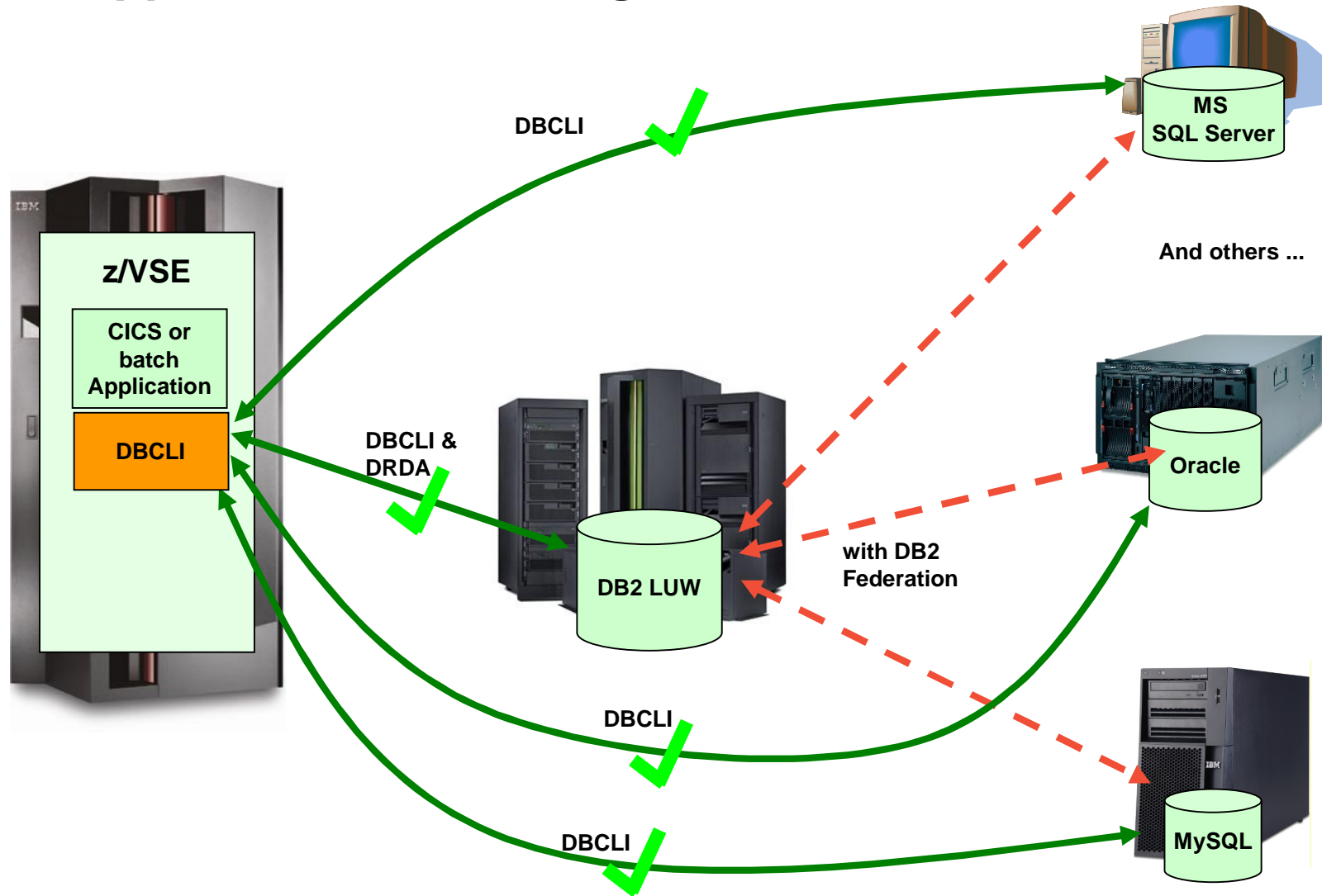
- § When using the DBCLI API in CICS transactions while CICS operates with storage protection, **all programs using the DBCLI API need to be defined with EXECKEY(CICS)**
 - § This is also true for those programs that link to these programs
 - § TASKDATAKEY(CICS) for the transaction definition is NOT required.

- § When using the DBCLI API in CICS transactions, the **EZA "task-related-user-exit" (TRUE) has to be activated** before these transactions can be run
 - § For details on how to activate this TRUE, refer to "CICS Considerations for the EZA Interfaces" in the z/VSE TCP/IP Support, SC34-2640

- § Most JDBC drivers will only accept **pure SQL statements**
 - § They will not accept SQL preprocessor statements that are used for DB2 Server for VSE applications

- § The call to the IESDBCLI function must be a **static CALL** in COBOL
 - § Do not use the DYNAM compiler option

z/VSE applications accessing Databases



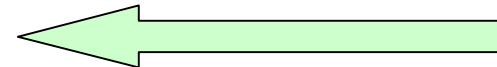
Agenda

§ **z/VSE V5.1 + PTFs Connector Enhancements**

- § z/VSE Database Call Level Interface

§ **z/VSE V5.1 Connector Enhancements**

- § VSE Script Connector: SYSIPT Variables Support
- § VSE Script Connector: New functions
- § VSE Script Connector: Logging of script input and output
- § VSAM Redirector: MapperConfigGUI Enhancements
- § VSE Connector Client & Server: LDAP signon support
- § VSE Connector Client & Server: LIBR DATA=YES



§ **z/VSE V4.3 Connector Enhancements**

- § POWER Output Generation Messages and exploitation in Java-based Connector
- § Decimal Position support for Java-based Connector
- § EXCPAD for VSAM Redirector
- § Redirector Trace activation via VSAM SNAP

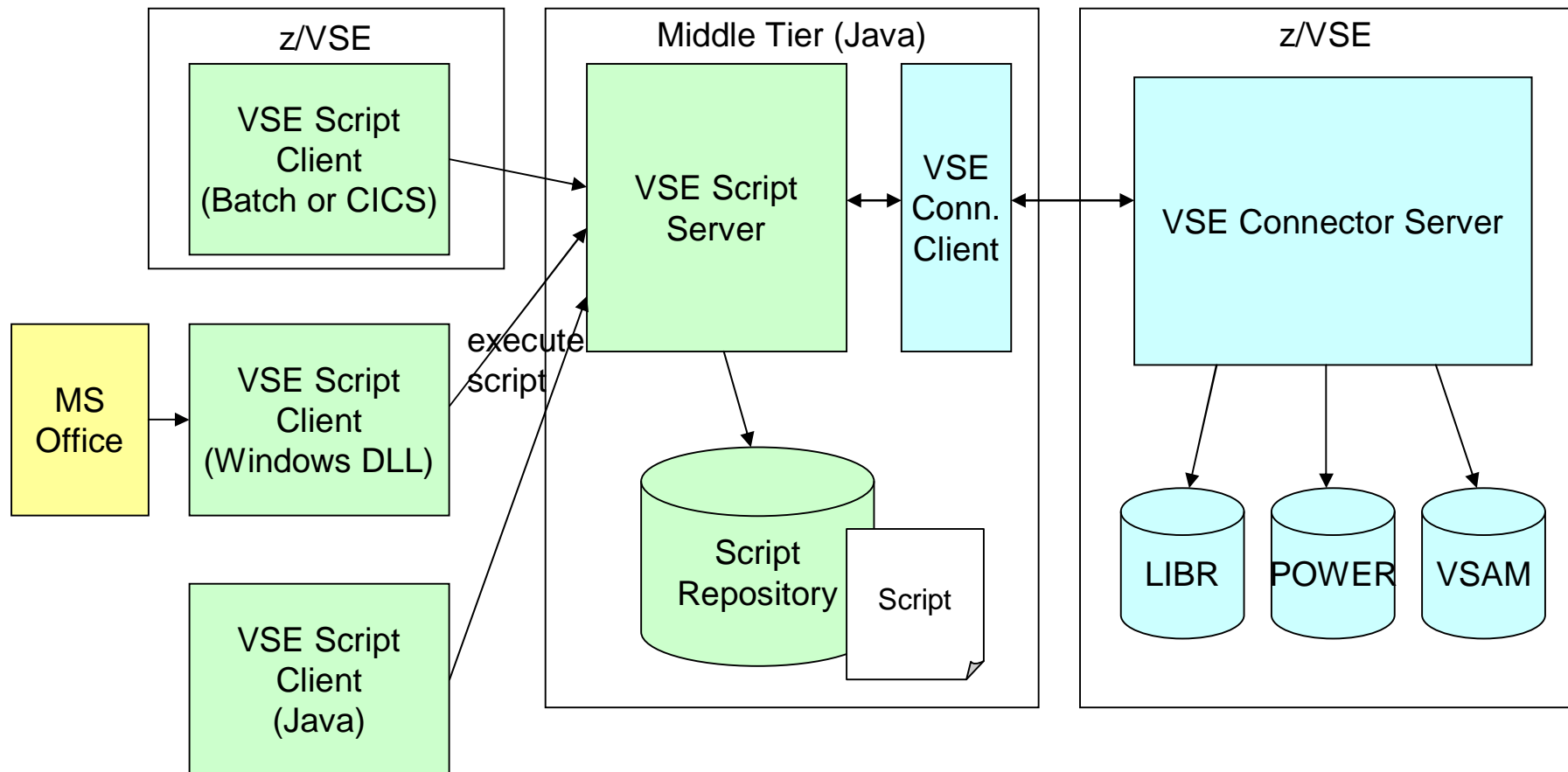
§ **New and updated Tools**

- § New Tool: Virtual z/VSE FTP Daemon

z/VSE V5.1: VSE Script Connector Overview

§ Part of the z/VSE Connectors since z/VSE V3.1

§ Allows remote access to z/VSE resources and data from non-Java platforms



z/VSE V5.1: VSE Script Connector: SYSIPT Variables Support

- § The SYSIPT variables support extends the VSE Script BATCH client programs by adding support for **symbolic variables**
- § Customers can assign the variables dynamically in JCL before they invoke the VSE Script batch client
- § Usage examples:
 - § Feed in data from previous job steps
 - § Centralize often used settings, such as IP address
- § Example: sets the target host and the script to execute using variables:

```

* $$ JOB JNM=START, DISP=L, CLASS=A
// JOB START
// LIBDEF *, SEARCH=(PRD1. BASE, PRD2. SCEEBASE, PRD2. DBASE)
// SETPARM DESTIP=' 10. 31. 0. 1'
// SETPARM SCRIPT=' testscript. src'
// SETPARM HELLO=' HELLO '
// SETPARM WORLD=' WORLD'
// EXEC IESSCBAT, PARM=' CODEPAGE=CP1047 SHOWERROR=YES SYMBOLS=YES'
&DESTIP: 4711
&SCRIPT
Script input ...
&HELLO&WORLD. !
/*
/&
* $$ EOJ

```

z/VSE V5.1: VSE Script Connector: SYSIPT Variables Support

§ The support must be enabled by setting the new PARMs parameter
SYMBOLS=YES

§ The default for this new parameter is SYMBOLS=NO to ensure backward compatibility.

§ The defined format of the variables specified in SYSIPT will be the **same format** that is described in “System Control Statements” manual, **Job Controls 'Symbolic Parameters'** chapter, available here:

http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IESSOE51/3.7?SHELF=IESVSE71&DT=20090403085040

§ A symbolic variable starts with '&'

§ When a '&' is needed in the input, write it as '&&'

§ Symbolic variable name contains of characters [0-9][A-Z] (yes, uppercase!) (this is not checked by the library, but the symbol would be not found)

§ Any other character beside [0-9][A-Z] marks the end of the current symbol

§ A '.' after the symbol name marks the end of the symbol without printing a character, for example '&SYMBOL.ALL' where SYMBOL='HELLO' will result in 'HELLOALL' without a '.'

§ The maximum final line length is not limited

§ A symbolic variable can be defined in JCL using

```
// SETPARM [SYSTEM] VARIABLE='VALUE'
```

z/VSE V5.1: VSE Script Connector: New functions

§ New LIBR functions

- § List libraries, sub libraries, members
- § Create/delete sub library
- § Copy/move member
- § Delete/rename member
- § Download member (binary and text)
- § Upload member (binary and text)
- § Put member on POWER queue
- § Get member from POWER queue

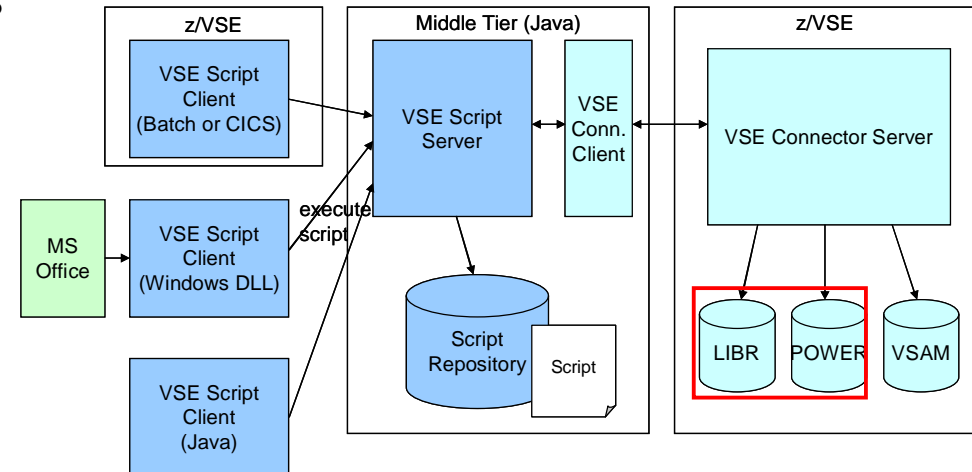
§ New POWER functions

- § Get entry in binary
- § Put entry in binary

§ Codepage related functions

- § Convert a string to binary and vice versa, using a specific codepage
- § Write/read a local file in binary

- à **Support for Binary data and Codepage tools allow to use VSE Script Connector with Double Byte Characters Set (DBCS) and Unicode data**



z/VSE V5.1: VSE Script Connector: Logging script input/output

- § The VSE Script Server now optionally prints all input and output data into the server log
- § This new feature can be used for audit purposes
- § The logging can be enabled using the new optional configuration parameters
 - § `logscriptinputparams`
 - § `logscriptoutput`
- § Additionally a script function was added to print directly into the server log:
 - § `PRINTLOG()`
- § This function can be exploited by user scripts to print audit-relevant messages to the server log

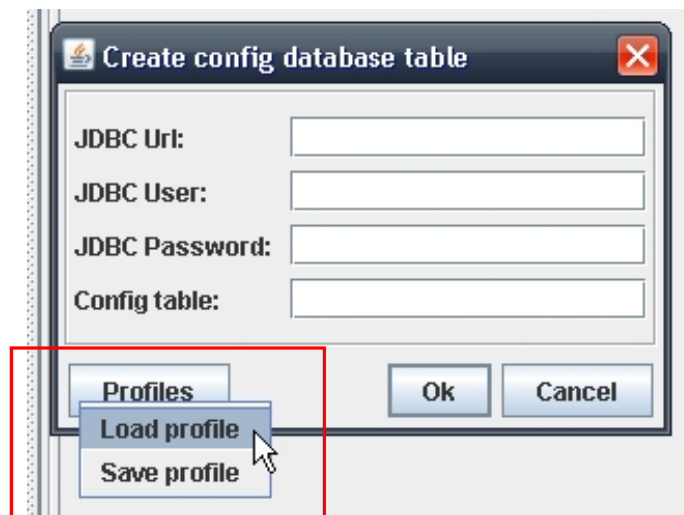
```

04.11.2010 08:56:30 (8) - Client connection request from 127.0.0.1
04.11.2010 08:56:30 (11) - Client has been accepted.
04.11.2010 08:56:30 (11) - Connection has been accepted from 127.0.0.1
04.11.2010 08:56:30 (11) - Using default system codepage.
04.11.2010 08:56:30 (11) - Executing script 'samples/qosub.src'
04.11.2010 08:56:30 (11) - Script receives 3 input parameter(s):
04.11.2010 08:56:30 (11) -   argv[1]='2'
04.11.2010 08:56:30 (11) -   argv[2]='test'
04.11.2010 08:56:30 (11) -   argv[3]='parameters'
04.11.2010 08:56:30 (11) - Script output follows:
04.11.2010 08:56:30 (11) -   'start'
04.11.2010 08:56:30 (11) -   'sub'
04.11.2010 08:56:30 (11) -   'end'
04.11.2010 08:56:30 (11) - PRINTLOG: 'New log output'
04.11.2010 08:56:30 (11) - Connection has been terminated from 127.0.0.1
04.11.2010 08:56:30 (11) - Client has been disconnected.

```

z/VSE V5.1: VSAM Redirector: MapperConfigGUI Enhancements

- § MapperConfigGui is part of VSE VSAM Redirectors DBHandler
- § The MapperConfigGui now allows to save profiles which contain the information needed to access a database target
- § The user don't have to enter them again and again when he switches between different JDBC targets, e.g. a test database system and the production database system
- § For security reasons the password is never saved in the profile



z/VSE V5.1: VSE Java-based Connector: LDAP signon support

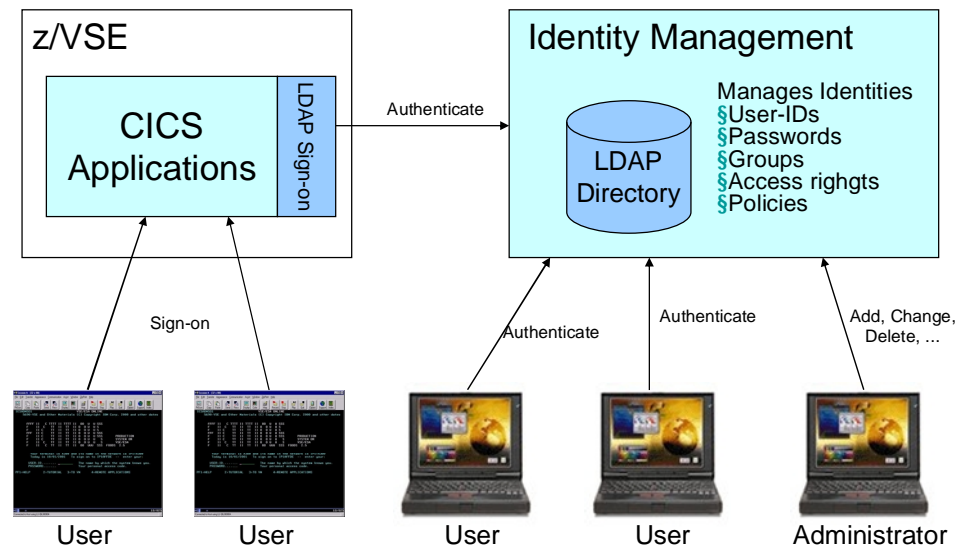
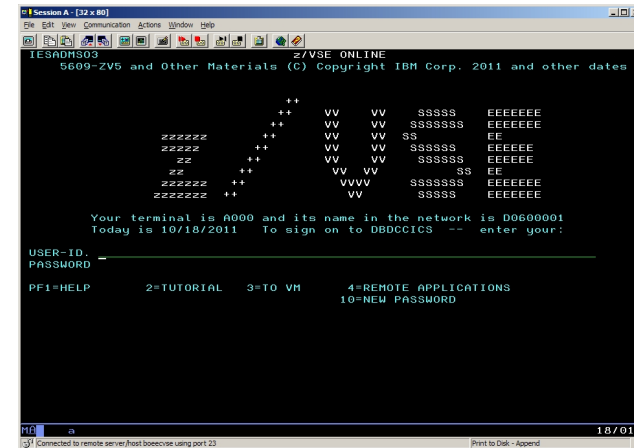
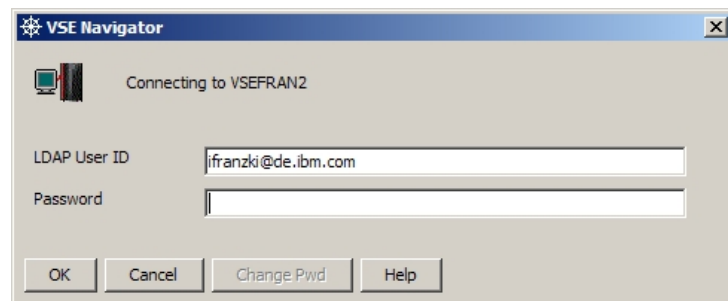
§ z/VSE V4.2 added support for LDAP Signon

- § Authenticate against a corporate wide Identity Management System (using LDAP)
- § Single Signon/Simplified Signon by using the same user-ID and password
- § User-ID & passwords up to 64 characters

§ z/VSE V5.1 adds LDAP signon support for the VSE Connector Client & Server

- § A Java based application can now use the same corporate user-ID and password as for IUI signon

Example: VSE Navigator



z/VSE V5.1: VSE Connector Client & Server: LIBR DATA=YES

§ VSE Connector Client & Server supports **access to LIBR members** since VSE/ESA 2.5

§ Download & Upload of LIBR members

§ Also access of .PROC members (procedures) is possible

§ Procedures may be cataloged with the DATA=YES attribute, if they contain SYSIPT data

```
// EXEC LIBR
ACCESS S=lib.sublib
CATALOG member.type DATA=YES
...
/*
```

§ Prior to z/VSE V5.1, any members created by the VSE Connector Server used DATA=NO

§ You could damage an procedure that was previously cataloged with DATA=YES

§ Since z/VSE V5.1, the VSE Connector Client & Server support the DATA=YES attribute

§ You can store a member with DATA=YES

§ Use method

```
VSELibraryMember.setSYSIPTDataInProcedure(boolean sysiptdata)
```

§ Example: VSE Navigator

§ Double-click on a member to edit it

§ Member automatically retains its DATA=YES attribute

Agenda

§ **z/VSE V5.1 + PTFs Connector Enhancements**

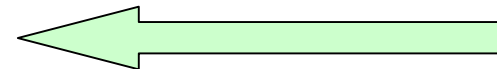
- § z/VSE Database Call Level Interface

§ **z/VSE V5.1 Connector Enhancements**

- § VSE Script Connector: SYSIPT Variables Support
- § VSE Script Connector: New functions
- § VSE Script Connector: Logging of script input and output
- § VSAM Redirector: MapperConfigGUI Enhancements
- § VSE Connector Client & Server: LDAP signon support
- § VSE Connector Client & Server: LIBR DATA=YES

§ **z/VSE V4.3 Connector Enhancements**

- § POWER Output Generation Messages and exploitation in Java-based Connector
- § Decimal Position support for Java-based Connector
- § EXCPAD for VSAM Redirector
- § Redirector Trace activation via VSAM SNAP



§ **New and updated Tools**

- § New Tool: Virtual z/VSE FTP Daemon

z/VSE V4.3: POWER Output Generation Messages Support

§ As of **z/VSE 4.2**, VSE/POWER can generate the following notification messages for a SAS (Spool Access Support) application

§ **Job Generation message 1Q5HI (JGM):**

Informs that the job, submitted via SAS interface, has generated another job as punch output with DISP=I

§ **Job Completion message 1Q5DI (JCM):**

Informs that the job, submitted via SAS interface, has completed

§ **With z/VSE 4.3, a new notification message has been added:**

§ **Output Generation message 1Q5RI (OGM):**

Is generated each time when the job, submitted via SAS interface, has created LST or PUN entry, and this entry became ready for processing

§ For details about how to use the VSE/POWER Spool Access Support programming interface, please see Manual “VSE/POWER Application Programming”

z/VSE V4.3: POWER Output Generation Messages

§ **With the new OGM support, a Job Scheduler application can now control the whole lifetime of a job:**

§ Job **Submission**

§ Job **Generation** (DISP=I)

§ Job **Completion**

§ **Output** Generation

§ **Without OGMs, its hard to find all outputs generated by a job**

§ A job may produce various outputs

ú Multiple LST/PUN cards in the job

ú Output segmentation

§ Outputs may have different names than the generating job (JNM=nnn in LST/PUN card)

§ Outputs may have different numbers than the generating job

ú Segmentation overflow (more than 127 segments)

ú Multiple LST/PUN cards in the job

§ **OGMs now provide a save way to retrieve all outputs generated by a Job**

z/VSE V4.3: POWER Output Generation Messages

§ The VSE Connector Client & Server now support OGMs

§ When submitting a Job via VSE Connector Client, an application can request to queue OGMs for the job:

```
VSEPowerEntry entry = new VSEPowerEntry(system,QUEUE_RDR,"MYJOB");
entry.setQueueComplMsgs(true); // request job completion messages
entry.setQueueOutputMsgs(true); // request output generation messages
entry.put(jobfile); // submit the job
```

§ The application can check if a job has completed:

```
if(entry.isCompleted()) // check if the job execution is complete
```

§ When the job has completed, the application can retrieve a list of outputs generated by the job:

```
entry.addVSEResourceListener(this); // register as resource listener
entry.getOutputList(); // retrieve the list of output
entries
entry.removeVSEResourceListener(this); // de-register resource listener
```

§ The application can then process the list of received VSEPowerEntry objects

§ Example: `com/ibm/vse/samples/SubmitJob.java` in the samples directory

z/VSE V4.3: Decimal Positions

- § The VSE Connector supports decimal data types like **PACKED** or **ZONED**
 - § in both signed and unsigned variants
- § Those data types are often used by customer applications to store monetary type of information
 - § Monetary information usually has at least 2 decimal places, e.g. \$123.45
 - § COBOL or PL/1 applications usually store such decimal numbers as packed decimal (COMP-3) or zoned decimal data types, with **implied decimal position**
 - § The implied decimal position (as the name implies) is not really stored as part of the decimal number, but it is implied when reading or updating the number
- § **Example:**
 - § The decimal value of **123.45** is stored as packed decimal: **x'12345C'**
 - § The implied decimal position is 2 in this case (2 digits from the right).
- § **Since the decimal position is not stored as part of the numerical data, that information needs to be stored as part of the mapping information together with the field name, offset, length and type**
- § **With z/VSE 4.3, the VSE Connectors has been enhanced to support (implied) decimal positions**

z/VSE V4.3: Decimal Positions

§ Decimal positions apply to the following data types:

- § PACKED Packed Decimal (COBOL COMP-3)
- § UNPACKED Unsigned Packed Decimal
- § ZONED Zoned Decimal (COBOL PIC 9(n))
- § UZONED Unsigned Zoned Decimal

§ The decimal position can be:

- § Zero No decimal position (e.g. 12345)
- § Positive Specifies the number of decimal digits from the right
(e.g. 123.45 has a decimal position of 2)
- § Negative Specifies the number of implied zero digits right to the number
(e.g. 1234500 has decimal position of -2 if stored as 12345
as un-scaled value)

§ The decimal position is interpreted by the VSE Connector Client when passing such numerical data to the calling application

- § The implied decimal position as stored in the mapping is applied to the (un-scaled) number, before passing it to the user application
- § Any number passed from user application to the VSE Connector Client is converted to its un-scaled value based on the implied decimal position

§ Decimal numbers with a non-zero decimal position are represented as Java **java.math.BigDecimal** object by the VSE Connector Client

z/VSE V4.3: Decimal Positions

§ The following components have been updated to support decimal positions

- § The mapping file (IESMAPD) to store the decimal position
- § VSE Connector Client to handle decimal positions and java.math.BigDecimals
- § VSAM JDBC Driver to support Decimal Positions
- § VSE Script Server to support Decimal Positions
- § IDCAMS RECMAP command to support Decimal Positions
- § VSAM Maptool to support Decimal Positions
- § VSE Navigator to support Decimal Positions

§ Any existing mapping stored in the mapping file IESMAPD can be used unchanged with z/VSE 4.3 or later

- § Any migrated decimal field will have a zero decimal position, which is what they implicitly had when no decimal position support was existing.

§ Any existing application that did work with an older version of the VSE Connector Client will work unchanged with the z/VSE 4.3 version of VSE Connector Client

- § As long as the mapping is not changed to use decimal positions other than zero
- § Mappings migrated or copied over from previous versions will automatically have a zero decimal position, as stated above
- § User applications may have to be adapted if non-zero decimal positions are used

z/VSE V4.3: Redirector EXCPAD

§ Prior to z/VSE 4.3 the VSAM Redirector was executed **in the same subtask as VSAM and the application (caller)**

§ Redirector activities may be time consuming (network transfers, database operations, ...)

ü During this time, no other activities are possible for this subtask

§ Under CICS, VSAM normally returns back via EXCPAD exit when waiting for an I/O

ü Allows CICS to perform other activities concurrently

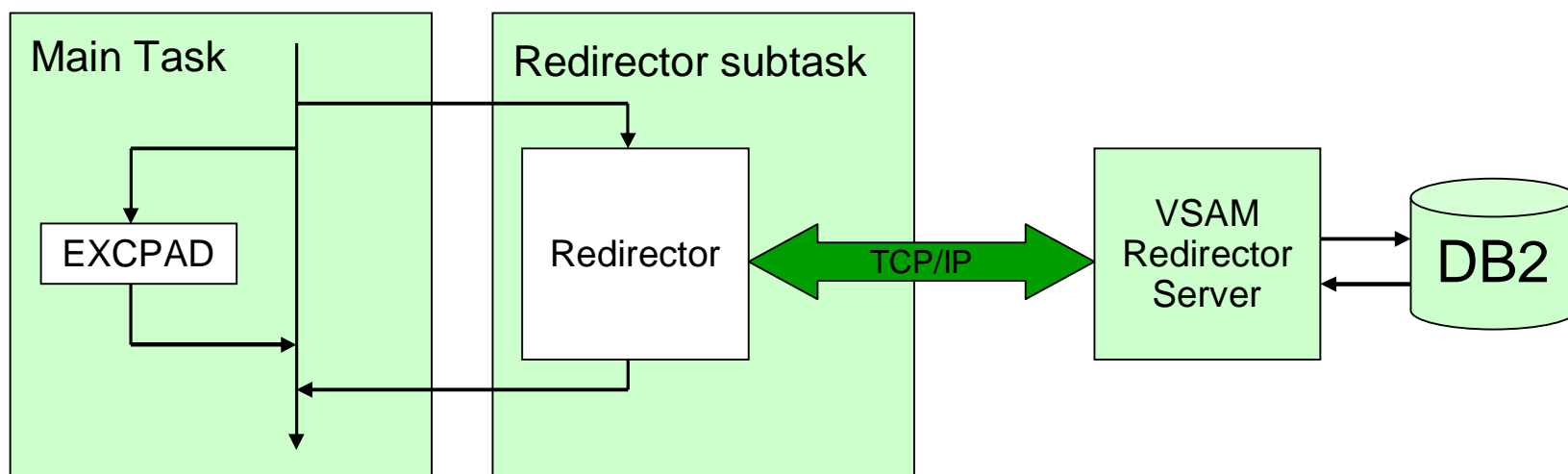
§ **Since z/VSE 4.3 VSAM executes the Redirector under a separate subtask**

§ VSAM now also returns back to CICS via EXCPAD when waiting for Redirector

ü Allows CICS to perform other activities concurrently

§ This capability is primarily implemented for CICS TS transactions.

ü The Redirector EXCPAD is not used for VSAM files opened by CICS/VSE.



z/VSE V4.3: Redirector EXCPAD

§ Prior to z/VSE 4.3 heavy use of VSAM Redirector could slow down transaction processing in CICS

§ Due to VSAM requests block the CICS I/O task when Redirector is active

§ With the new subtask the VSAM Redirector handling no longer blocks the CICS I/O task

§ Allowing other transactions to do its work

§ Multiple redirected requests will be queued up for processing in the new subtask

§ The EXCPAD user exit is enabled automatically under the following conditions:

§ a VSE/VSAM cluster is enabled for the Redirector

§ the EXCPAD exit is defined during the OPEN request

§ VSAM will attach only one Redirector subtask per partition even if multiple redirected files are opened in the partition with an active EXCPAD

§ Support is transparent

§ No need to configure or setup anything

§ All types of Redirector activities are processed in subtask (except OPEN/CLOSE)

ú VSAM Redirector OWNER=VSAM or REDIRECTOR

ú VSAM Capture Exit

ú Customer/Vendor implemented Redirector Exit

z/VSE V4.3: Redirector Trace activation via SNAP

§ The VSAM Redirector host parts consist of

- § IESVEX01 (will be renamed to IKQVEX01 when activating redirection)
- § IESREDIR – VSAM Redirector Client
- § IESVSCAP – VSAM Capture Exit

§ All 3 parts have an internal trace facility

- § Prior to z/VSE 4.3, the trace could only be activated through a MSHP PATCH
 - ú Trace was written to SYSLOG (console) only
- § Since z/VSE 4.3, the trace can now be dynamically enabled (and disabled) via the VSAM SNAP trace
 - ú Trace is now written to SYSLST (listing) of job

§ Trace activation is done via IKQVEDA:

```
// EXEC IKQVEDA,PARM='SYSIPT'  
ENABLE SNAP=0010,PART=F2  
END  
/*
```

z/VSE V4.3: VSAM SNAP Trace assignments

Type:	Enables:
0001	Catalog management error code trace
0002	Buffer manager trace
0003	OPEN control block dump (when OPEN processing is complete) OPEN error trace (prints control blocks if an error occurs during OPEN processing) CLOSE control block dump (at the beginning of CLOSE processing)
0004	VSE/VSAM I/O trace
0005	I/O error trace
0008	Catalog management I/O trace (prints all I/O operations done by VSE/VSAM catalog management)
0009	Record management error trace (prints control blocks for any error detected by VSE/VSAM record management)
0010	Redirector Trace
0013	In-core wrap trace for trace points within VSE/VSAM Record Management
0014	Level2 SNAP013 Trace (I/O, EXCPAD and z/VSE Lock Activity)
0015	Level3 SNAP013 Trace (Buffer Management)
0016	Produce a printout (PDUMP) each time the SNAP013 Trace Table wraps.

Agenda

§ **z/VSE V5.1 + PTFs Connector Enhancements**

- § z/VSE Database Call Level Interface

§ **z/VSE V5.1 Connector Enhancements**

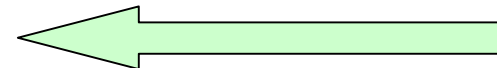
- § VSE Script Connector: SYSIPT Variables Support
- § VSE Script Connector: New functions
- § VSE Script Connector: Logging of script input and output
- § VSAM Redirector: MapperConfigGUI Enhancements
- § VSE Connector Client & Server: LDAP signon support
- § VSE Connector Client & Server: LIBR DATA=YES

§ **z/VSE V4.3 Connector Enhancements**

- § POWER Output Generation Messages and exploitation in Java-based Connector
- § Decimal Position support for Java-based Connector
- § EXCPAD for VSAM Redirector
- § Redirector Trace activation via VSAM SNAP

§ **New and updated Tools**

- § New Tool: Virtual z/VSE FTP Daemon



z/VSE Tools - Overview

§ IBM offers are a huge set of tools available on the z/VSE Homepage
<http://ibm.com/zvse/downloads>

§ Most tools are 'as is', at no additional charge.

§ Connector components (part of z/VSE and officially supported) are also available here

The screenshot shows the IBM z/VSE Downloads page in a Windows Internet Explorer browser. The browser's address bar shows the URL <http://www-03.ibm.com/systems/z/os/zvse/>. The page features a navigation menu on the left with the following items: About z/VSE, How to buy, News & announcements, Events, Solutions, Products & components, Documentation, Service & support, Downloads (highlighted with a red box), Education, Partners, FAQ, and Contact z/VSE. The main content area is titled 'Downloads' and has three tabs: Connectors, Tools, and Samples. Under the 'Connectors' tab, the following tools are listed: VSE Connector Client, VSE Script Server, Linux Fast Path Daemon, Virtual z/VSE FTP Daemon, VSE Health Checker, and CICS2WS Toolkit. Under the 'Tools' tab, the following tools are listed: VSAM Redirector Server, VSE Navigator, VSEPrint utility, VSE System class library, and VSE Security class library. Under the 'Samples' tab, the following tools are listed: VSE Virtual Tape Server, VSAM Maptool, Keyman/VSE, and WebSphere MQ Client for VSE. A 'Recent additions and updates' section lists the following tools and their update dates: VSE Script Server (updated 03/2012 for APAR PM58755), VSE Virtual Tape Server (updated 01/2012 for APAR PM55958), VSE Connector Client (updated 11/2011 for z/VSE V5.1 GA), Linux Fast Path Daemon (updated 11/2011 for z/VSE V5.1 GA), VSAM Redirector Server (updated 11/2011 for z/VSE V5.1 GA), VSE Navigator (updated 11/2011 for z/VSE V5.1 GA), VSAM Maptool (updated 11/2011 for z/VSE V5.1 GA), Keyman/VSE (updated 11/2011 for z/VSE V5.1 GA), VSE Health Checker (updated 11/2011 for z/VSE V5.1 GA), VSE System class library (updated 11/2011 for z/VSE V5.1 GA), VSE Security class library (updated 11/2011 for z/VSE V5.1 GA), Virtual z/VSE FTP Daemon (new 12/2010), and CICS2WS Toolkit Version 2.5 (updated 06/2010). The page also includes a 'We're here to help' section with an 'E-mail us' button, a 'Stay informed' section with a 'Get the latest news about z/VSE through Twitter' button, and a 'Downloads' section with a 'Get Java from IBM developerworks' button and an 'Order PTFs' button. There is also an 'Acrobat' section with a 'Get Adobe Reader' button and a 'Rate zEnterprise product' section with a star rating and a 'Rate zEnterprise product' button.

Virtual z/VSE FTP Daemon

§ The Virtual z/VSE FTP Daemon can be installed on any Java-enabled platform and **emulates an FTP server**

§ The actual access to z/VSE resources is done using the VSE Connector Server.

§ **Download:** <http://ibm.com/zvse/download>

à **Fits perfectly to Linux Fast Path**

§ **The Virtual z/VSE FTP Daemon:**

§ Handles all incoming FTP clients.

§ Connects to one or multiple VSE Connector Servers.

§ Is responsible for connection-handling.

§ Is responsible for data translation (ASCII-EBCDIC).

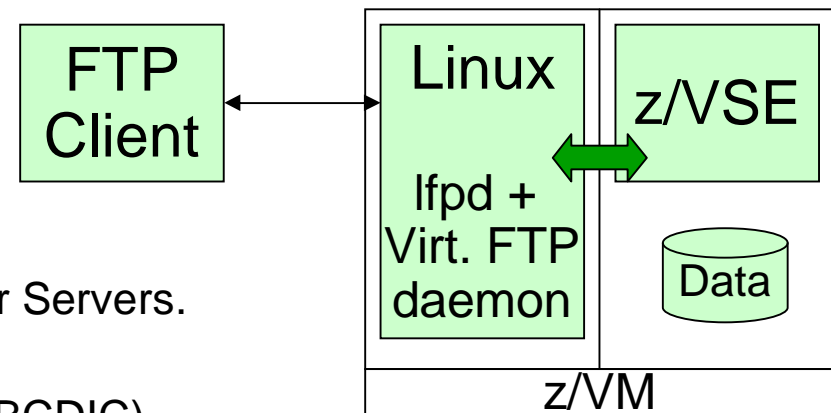
§ Is IPv6 ready

ú You can connect FTP clients using IPv6, the Virtual z/VSE FTP Daemon connects to the VSE Connector Server using IPv4.

§ Supports SSL

ú both for the FTP connection (between FTP client and Virtual z/VSE FTP Daemon, using implicit SSL (FTPS)),

ú and for the connection to the VSE Connector Server (between Virtual z/VSE FTP Daemon and z/VSE host).



LDAP Query Callable Module

§ The z/VSE LDAP Query Callable Module allows you to **programmatically query an LDAP server** from within your programs to retrieve attributes of an LDAP user

§ You can either call the z/VSE LDAP Query Callable Module directly (i.e. via an COBOL external call), or via EXEC CICS LINK when running under CICS

§ The z/VSE LDAP Query Callable Module can be used on **z/VSE 4.2 or later**


§ Extends the z/VSE LDAP Sign-on Support

```
01 LDGA-AREA.
   03 AREA-LENGTH      PIC S9(9) BINARY. <-- In: Length of the Area in Bytes
   03 USER-ID          PIC X(64).        <-- In: LDAP user ID to get attributes for
   03 SEARCH-FILTER    PIC X(128).       <-- In: Additional Search filter or blanks
   03 RET-CODE         PIC S9(9) BINARY. <-- Out: Return code
   03 LDAP-CODE        PIC S9(9) BINARY. <-- Out: LDAP Return code
   03 ATTR-COUNT       PIC S9(4) BINARY. <-- In: Number of attr entries following
   03 ATTR-ENTRY OCCURS x TIMES.
       05 ATTR-NAME     PIC X(64).        <-- In: Name of Attribute to get
       05 VALUE-LENGTH PIC S9(4) BINARY. <-- In: Length of ATTR-VALUE
       05 VALUE-COUNT   PIC S9(4) BINARY. <-- In/out: Number of Values following
       05 VALUE-ENTRY OCCURS y TIMES.
           07 ATTR-VALUE PIC X(n).        <-- Out: Attribute Values(s).
                                           Length (n) must match the VALUE-LENGTH

01 IESLDGAB PIC X(8) VALUE 'IESLDGAB'
...
Fill the parameter area here
...
CALL IESLDGAB USING BY REFERENCE LDGA-AREA.
```

Questions ?





IBM Systems Lab Services and Training

Helping you gain the IBM Systems skills
needed for smarter computing



- § Comprehensive education, training and service offerings
- § Expert instructors and consultants, world-class content and skills
- § Multiple delivery options for training and services
- § Conferences explore emerging trends and product strategies

Special Programs:

- § IBM Systems 'Guaranteed to Run' Classes -- ***Make your education plans for classes with confidence!***
- § Instructor-led online (ILO) training ***The classroom comes to you.***
- § Customized, private training
- § Lab-based services assisting in high tech solutions

www.ibm.com/training

