USING DDL IN IMS

Robert Recknagel IBM CHAMPION
Senior Architect Mainframe Databases/Middleware

Mail: Robert.Recknagel@f-i-ts.de
Phone: +49 89 94511 8789

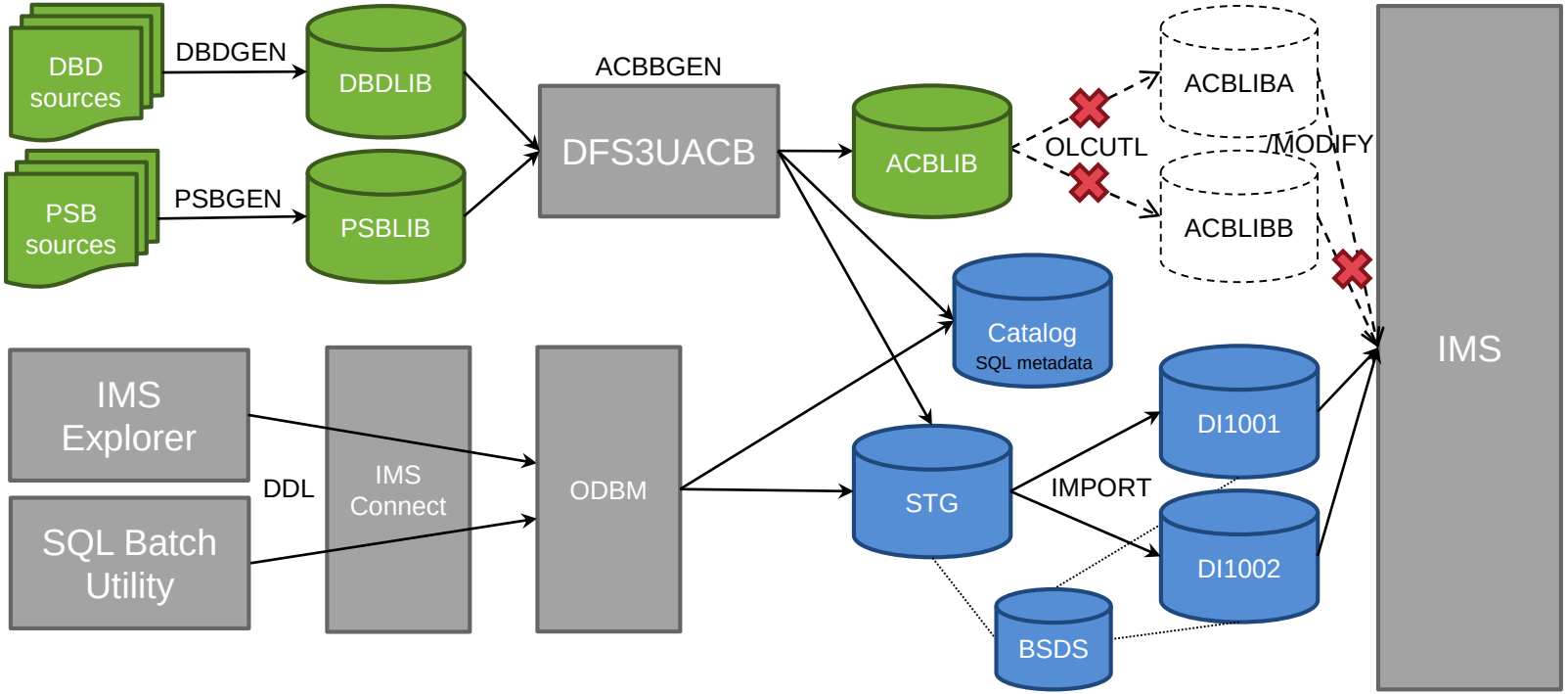
finanz informatik
technologie service

Agenda

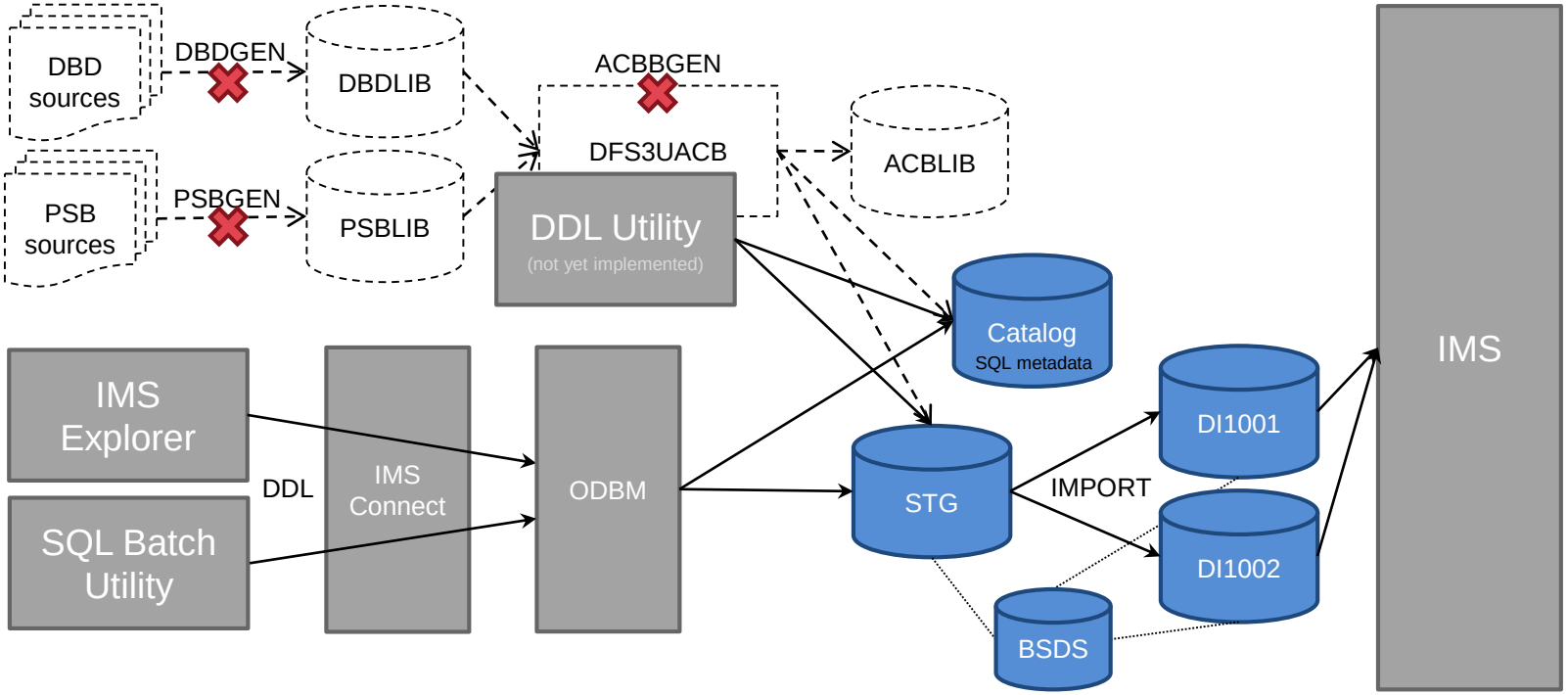
1. The DDL infrastructure of IMS
2. The DDL syntax of IMS
3. Ways to execute DDL in IMS
4. Challenges with mixed ACBGEN/DDL environments
5. Migrating to a DDL-only environment
6. Unresolved issues with DDL

The DDL infrastructure of IMS

Overview of an IMS managed ACBs environment



Overview of a DDL-only environment



Last preparations

- After your migration to IMS Managed ACBs DBRC, DLIBATCH and IMS database utilities still can continue to use DBDLIB and PSBLIB, but at the latest when starting to use DDL they all have to use the Directory instead
- DBRC can be configured to use the Directory instead of the DBDLIB by executing the DBRC command **`CHANGE.RECON CATALOG(DFSCD000)`**

  ⓘ Note, that DBRC only allows you to define one Catalog / Directory even if you may not share it in your data sharing environment.

- If you are still using DLIBATCH, you need to either modify your <u>all</u> your DLIBATCH jobs / procedures to reference the DFSDFxxx member or you need to add the Catalog Definition Exit DFS3CDX0 to your SDFSRESL

  ⓘ Note, that you need to modify the Catalog Definition Exit DFS3CDX0 if you are not using the default Catalog alias DFSC.

- Some IMS database utilities also can be configured to use the Directory, by referencing the DFSDFxxx member or by adding the Catalog Definition Exit DFS3CDX0 to your SDFSRESL, others have individual settings or still don‘t yet fully support IMS Managed ACBs

The DDL syntax of IMS

Create DBDs using DDL

```
CREATE DATABASE <database_name> ACCESS <database_access>
  [<additional_database_options>];
CREATE TABLESPACE <dataset_name> IN <database_name>
  [<additional_dataset_options>];
CREATE TABLE <external_segment_name>(
    [<field_definition>]
    [,<index_relationship_definition>]
  ) INTERNALNAME <internal_segment_name>
  IN DATABASE <database_name>
  [<additional_segment_options>];
```

View CREATE DATABASE in IBM Documention

View CREATE TABLESPACE in IBM Documention

View CREATE TABLE in IBM Documention

Note, that this is a massively simplified syntax diagram applicable to most of the database types. HALDBs and logical databases won‘t have a CREATE TABLESPACE statement, GSAM databases may not have a CREATE TABLE statement.

Sample on how to create a DBD using DDL

```
CREATE DATABASE IVPDB1 ACCESS HIDAM OSAM;
CREATE TABLESPACE DFSIVD1 IN IVPDB1
  SIZE PRIMARY 2048;
CREATE TABLE A1111111(
     A1111111 CHAR(10) START 1 TYPE C
       INTERNALNAME A1111111 PRIMARY KEY,
     LCHILD IVPDB1I.A1 INDEX
  ) INTERNALNAME
  IN DATABASE IVPDB1
  MAXBYTES 40 TWINBWD CTRYES;
<create_statements_for_primary_index_DBD_IVPDB1I>
COMMIT DDL;
```

Modify DBDs using DDL

```
ALTER DATABASE <database_name>
  <database_options_you_want_to_modify>;
```

View ALTER DATABASE in IBM Documention

```
ALTER TABLESPACE <dataset_name> IN <database_name>
  <dataset_options_you_want_to_modify>;
```

View ALTER TABLESPACE in IBM Documention

```
ALTER TABLE <external_segment_name>
  IN DATABASE <database_name>
  [ADD|ALTER|DROP COLUMN <field_name> [<field_options>]]
  [ADD|DROP LCHILD <field_name> [<lchild_options>]]
  [<segment_options_you_want_to_modify>];
```

View ALTER TABLE in IBM Documention

Note, that these are massively simplified syntax diagrams showing only a few options on how a DBD can be modified. You could also use for instance CREATE TABLE to add another segment to the DBD.

Sample on how to modify a DBD using DDL

```
ALTER TABLE STOKSTAT
  IN DATABASE DI21PART
  ADD COLUMN UNIT_PRICE DECIMAL(9,3) BYTES 9 START 21
    INTERNAL TYPECONVERTER ZONEDDECIMAL ISSIGNEDNO;
COMMIT DDL;
```

Note, that DDL calculates a lot of the values you need to specify in a DBD source on its own. BYTES is such a value. But because DDL treats DECIMAL values by default as packed decimal values it would incorrectly calculate 5 for BYTES in this case.

Delete DBDs using DDL

```
DROP DATABASE <database_name>;
```

View DROP DATABASE in IBM Documention

Create PSBs using DDL

```
CREATE PROGRAMVIEW <psb_name>(
    [CREATE SCHEMA TP <internal_pcb_name>
      [<additional_alternate_tp_pcb_options>]]
    [,CREATE SCHEMA DB <internal_pcb_name>
      USING <database_name>(
        CREATE SENSEGVIEW <segment_name> <add_options>,
        CREATE SENSEGVIEW <segment_name> <add_options>
      ) [<additional_db_pcb_options>]]
    [,CREATE SCHEMA GSAM <internal_pcb_name>
      USING <database_name>
      [<additional_gsam_pcb_options>]]
  ) [<additional_psb_options>];
```

View CREATE PROGRAMVIEW in IBM Documention

Note, that this is a massively simplified syntax diagram only showing some options, which are relevant for most of the PSBs

Sample on how to create a PSBs using DDL

```
CREATE PROGRAMVIEW DFSIVP1(
    CREATE SCHEMA TP PCB#1 MODIFYYES EXPRESSYES,
    CREATE SCHEMA DB PCB#2 USING IVPDB1(
        CREATE SENSEGVIEW A1111111
      ) PROCOPT 'A'
  ) LANGASSEM;
```

Delete PSBs using DDL

```
DROP PROGRAMVIEW <psb_name>;
```

View DROP PROGRAMVIEW in IBM Documention

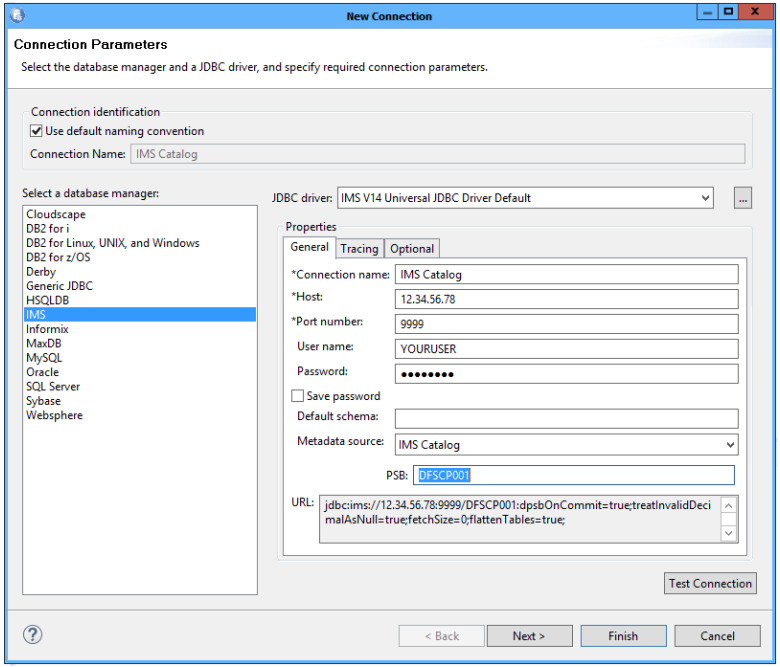
Note, that currently the only way to modify a PSB is to delete and re-create it.

Ways to execute DDL in IMS

Preparing to execute DDL with the IMS Explorer

- IMS Explorer and the SQL Batch Utility are currently the only ways to execute DDL in IMS
- Both require to have ODBM and IMS Connect in place
- As long as ODBM won‘t be used by distributed Java applications accessing both IMS and DB2 databases within the same commit scope, ODBM may be setup to run without RRS
- To limit the usage of ODBM to DDL and to restrict the usage of the DDL to specific users you may configure ODBMSECURE=R in the Common Service Layer section of the DFSDFxxx member and create the following profiles in the RACF class AIMS respectively A<*rclass*>:
  - * with access for nobody
  - DFSCP000 with access for everybody, who should be able to view Catalog metadata
  - DFSCP001 with access for everybody, who should be able to execute DDL
- Be aware, that IMS Connect needs to run with RACF=Y such that ODBM will be able to perform these checks with the appropriate RACF user ID

Executing DDL using the IMS Explorer



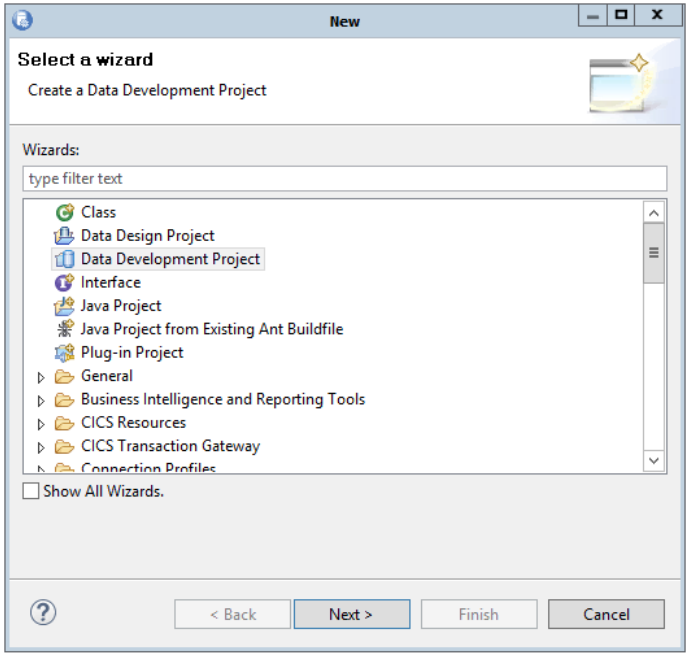
Start IMS Explorer and create a workspace.

After IMS Explorer has been fully launched, find the *Data Source Explorer* in the left lower edge.

Create a connection to the IMS Catalog by right-clicking in the *Data Source Explorer* and choose *New*. The new connection dialog will appear.

In the new connection dialog choose IMS on the left, provide a connection name (for instance *IMS Catalog*), your IMS Connect IP address or a corresponding DNS name, your IMS Connect ODBA port number, your RACF user ID and password as well as the Catalog update PSB name DFSCP001 and click on *Finish*.

Executing DDL using the IMS Explorer

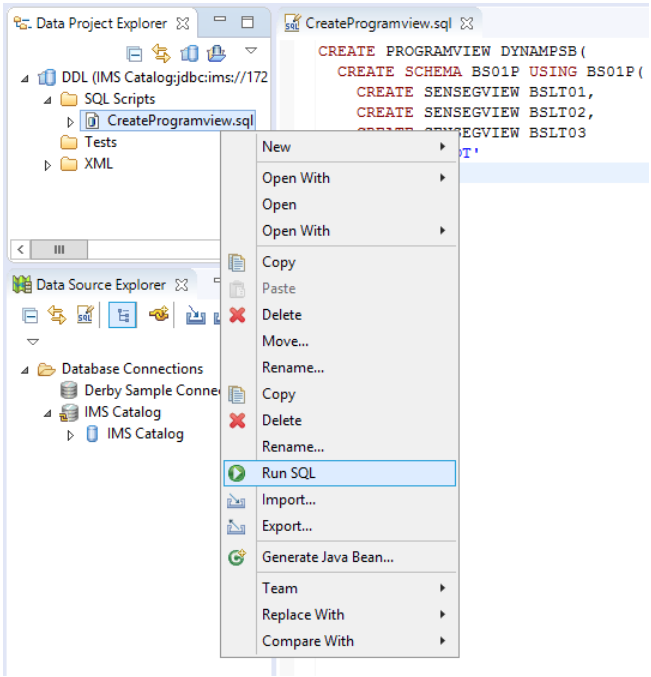


Now create a new *Data Development Project* by navigating through the menu *File >> New >> Other*.

Select *Data Development Project* in the upcoming dialog and click *Next*. Then enter a project name – for instance *DDL* – and click *Next* again. Now select your IMS Catalog connection profile and click *Finish*.

A message will appear and asks you, if you want to switch to the *Data perspective*. Click on *Yes*.

Executing DDL using the IMS Explorer



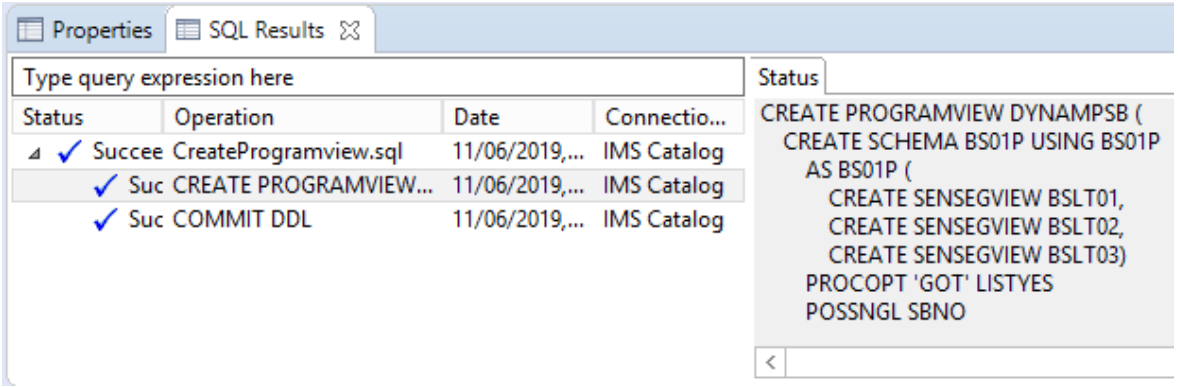
Create a new SQL script by opening the project you created in the *Data Project Explorer* view – which is on the left side above the *Data Source Explorer* view – and right-clicking on the *SQL Scripts* folder.

Provide a name for the new SQL script in the appearing dialog – for instance *CreateProgramview* – and click on *Finish*.

The SQL script is going to be opened on the right side. Enter your DDL statement there and save it using the keys *CTRL* + *S*. Don‘t forget the COMMIT DDL statement at the end! IMS Explorer currently doesn‘t automatically commit DDL.

You can now run your DDL statement by right-clicking on it in the *Data Project Explorer* view and select *Run SQL*.

Executing DDL using the IMS Explorer



The *SQL Results* view will be opened below the SQL script and show either success or an error. In the case of an error you will get a Java exception with an error message in the *Status* tab on the right side. The error messages usually contain SQL codes, AIB return and reason codes and / or DL/I status codes.

Preparing to execute DDL with the SQL Batch Utility

- You may prepare your system to run the SQL Batch Utility as it may not feasible to rollout IMS Explorer to work environment of every person, who should be able to create or modify DBDs or PSBs, and as it may not be acceptable from a regulatory standpoint to deploy DBD and PSB definitions using IMS Explorer in production
- To be able to run the SQL Batch Utility you need to setup the IMS Java on Demand feature (FMID JMK1506) and – in the case you haven‘t done it yet – the IBM Java SDK for z/OS as well as the JZOS Batch Launcher
- After the installation of the IMS Java on Demand feature you need to mount the IMSJAVA ZFS filesystem containing the IMS Universal JDBC Driver (imsudb.jar)
- The SQL Batch Utility is part of this Java archive
- You may also setup PassTicket support in IMS Connect to avoid needing to use cleartext passwords when running the SQL Batch Utility

Preparing to execute DDL with the SQL Batch Utility

- To do so you need to specify the APPL=<*your_appl_name*> in the ODACCESS statement of the HWSCFGxx member and create the following profiles in RACF:
  - Profile <*your_appl_name*> in class APPL with UPDATE access for everybody who should be able to query Catalog metadata or to execute DDL
  - Profile <*your_appl_name*> in class PTKTDATA with a SSIGNON key
  - Profile IRRPTAUTH.<*your_appl_name*>.* in class PTKTDATA with UPDATE access for everybody who should be able to query Catalog metadata or to execute DDL and with READ access for IMS Connect
- The activation of the IMS Connect PassTicket support may have an impact on existing distributed Java applications accessing IMS databases in the case you aren‘t using mixed-case passwords or passphrases and the password isn‘t being sent in capital letters, because after activating PassTicket support, IMS Connect doesn‘t translate the password into upper case anymore (this shouldn‘t affect IMS Connect clients invoking transactions, because there is a separate APPL setting for such clients in the DATASTORE statement of the HWSCFGxx member)

Executing DDL using the SQL Batch Utility

```
//SQLBATCH EXEC PGM=JVMLDM80,REGION=0M,
//            PARM='/ com.ibm.ims.jdbc.batch.BatchUtil'
//STDENV   DD *
export JAVA_HOME=/usr/lpp/java/J8.0
export PATH=/bin:"$JAVA_HOME"/bin
export LIBPATH=/lib:/usr/lib:"$JAVA_HOME"/bin:"$JAVA_HOME"/bin/j9vm
CLASSPATH=/usr/include/java_classes/IRRRacf.jar
export CLASSPATH="$CLASSPATH":/usr/lpp/ims/imsjava/imsudb.jar:
export IBM_JAVA_OPTIONS="-Xmx1G -Dfile.encoding=Cp1047"
/*
//MAINARGS DD DUMMY
//IMSSQL   DD *
--INSERT YOUR CONNECT, DDL, COMMIT AND DISCONNECT STATEMENTS HERE
/*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Java Interface to RACF used for PassTicket generation

Memory limit for the JVM

Your EBCDIC codepage

Note, that JVMLDM80 is the name of the 31bit Java 8.0 JZOS Batch Launcher. If you want to use the 64bit Java 8.0 JZOS Batch Launcher you need to specify JVMLDM86. You may need to modify this for future Java releases. Ensure that JAVA_HOME, PATH and LIBPATH settings point to corrensponding Java SDK. You will get errors if you are using the wrong Java SDK with the JZOS Batch Launcher you specified before.

Sample statements for the SQL Batch Utility

```
//IMSSQL   DD *
CONNECT jdbc:ims://12.34.56.78:9999/DFSCP001:applName=YOURAPPL;;
CREATE PROGRAMVIEW DYNAMPSB(
  CREATE SCHEMA YOURDB USING YOURDB(
    CREATE SENSEGVIEW YOURDBS1,
    CREATE SENSEGVIEW YOURDBS2,
    CREATE SENSEGVIEW YOURDBS3
  ) PROCOPT 'GP'
) LANGCOBOL;
COMMIT DDL;
DISCONNECT;
/*
```

Note, that the PARM value and the STDENV in-stream dataset shown on the slide before and the IMSSQL in-stream dataset shown here are case-sensitive. Ensure to not have line numbers in column 73 to 80 of both in-stream datasets.

How IMS currently processes DDL

- No matter whether you are using IMS Explorer or the SQL Batch Utility you first need to connect to IMS Connect‘s ODBA port to be able to execute any DDL statement
- During the connection setup the IMS Universal JDBC Driver will query metadata about the Catalog resources
- When you then execute a DDL statement, ODBM will start a Database Metadata Update (DMU) region to process the DDL statement
- Within the DMU region the DDL statement first will be parsed,
  - then a new DBD / PSB instance will be inserted into the Catalog and finally the Block Builder will create a new Directory control block and write it into the staging Directory
  - or (in the case of DROP statement) all existing instances of the DBD / PSB will be deleted in the Catalog and Auto Import will remove the corresponding control block from the Directory
- After commit the DMU region will be terminated
- Any error during the DDL processing will lead into an automic rollback of both Catalog and Directory (except in a few still unresolved error scenarios)

Manual vs. automatic import of DDL changes

- As long as you haven‘t coded AUTOIMPORT(CREATE) in the CATALOG section of the DFSDFxxx member you will need to manually import new DBDs and PSBs created using DDL
- DBDs and PSBs you have updated using DDL always need to be manually imported at the moment, because there is currently no automatic import for updated DBDs and PSBs (metadata-only updates to DBDs currently don‘t require an import)
- DBDs and PSBs you have deleted using DDL are always immediately removed from both Catalog and Directory
- For newly created DBDs and PSBs you can also request IMS to automatically create DB respectively PROGRAM resource definitions for them by specifying AUTOCREATE=YES in the CATALOG section of the DFSDFxxx member* (IMS will use the default resource descriptors)
- For newly created DEDBs you can also request IMS to automatically pre-allocate the database datasets by specifying PREALLOC=YES in the DDL section of the DFSDFxxx member
- Additional settings in this section will tell IMS how to pre-allocate these database datasets

**this requires Dynamic Resource Definition (DRD) to be activated*

Challenges with mixed ACBGEN/DDL environments

Challenges with mixed ACBGEN/DDL environments

- At least in the development stage you won‘t switch from traditional ACBGEN to DDL at one point in time, because you will need to
  - add DDL statements to your source control management system,
  - modify your deployment processes to run DDL instead of DBDGEN / PSBGEN / ACBGEN,
  - train DBAs and developers to use DDL instead of traditional DBD and PSB macros
- During the transition time both traditional ACBGEN and DDL will be used and this will cause additional challenges, because DDL doesn‘t insert, update or delete anything in the DBDLIB / PSBLIB / ACBLIB and because ACBGEN isn‘t aware of any modification done to the Catalog / to the Directory using DDL due to the fact, that for DFS3UACB the ACBLIB is still the trusted source
- This can cause issues like
  - IMPORT to fail after a DBD modification done with DBDGEN / ACBGEN, because DDL-built PSBs referencing the DBD haven‘t been rebuilt,
  - DBD or PSB modifications done with DDL to be overwritten by ACBGEN and vice versa,
  - PSBs referencing a DBD which has been dropped using DDL to be imported after an ACBGEN

Challenges with mixed ACBGEN/DDL environments

- To prevent issues like this you may
  - only use DDL in a dedicated DDL test environment and don‘t use traditional gens in this environment (which could be challenging at the moment, because current issues of DDL may still force you to use traditional gens for several kinds of DBDs and PSBs),
  - always automatically run the Catalog Library Builder Utility DFS3LU00 after any DDL statement that has been processed and the changes have been imported (which is currently not possible in the case you have any GSAM or logical DBDs in your systems and even in the case you don‘t have such DBDs it is still no complete solution because this won‘t help you to remove DBDs and PSBs being dropped by DDL from your DBDLIB / PSBLIB / ACBLIB)
- We discussed these and other issues in mixed ACBGEN/DDL environments in several meetings with the IMS lab and proposed potential solutions to this, but the IMS lab hasn‘t decided yet on how to help clients to better handle this transition time

Catalog Library Builder Utility DFS3LU00

- The Catalog Library Builder Utility DFS3LU00 can be used to create DBD sources, PSB sources, DBDs, PSBs and ACBs from the Directory
- Like lots of vendor utilities this utility is based on the Catalog API, which is in real not a Catalog API but a Directory API, because it processes the control blocks in the Directory and not the Catalog instances
- If you intend to use this utility you should have APAR PH14927 applied, but even with this APAR the utility is still unable to handle
  - GSAM DBDs and PSBs with one or more GSAM PCBs,
  - Logical DBDs
- In addition to this you should be aware, that you will lose any additional metadata you added for SQL access to an IMS database
  - using DDL or
  - after you have recovered your Directory using the Directory Recovery Utility DFS3RU00 no matter whether the metadata has been added using traditional gens or using DDL

Catalog Library Builder Utility DFS3LU00

```
//LIBBUILD EXEC PGM=DFS3LU00,REGION=0M
//STEPLIB  DD  DISP=SHR,DSN=IMS.SDFSRESL
//         DD  DISP=SHR,DSN=IMS.IMSDALIB
//DFSRESLB DD  DISP=SHR,DSN=IMS.SDFSRESL
//PROCLIB  DD  DISP=SHR,DSN=IMS.PROCLIB
//SYSLIB   DD  DISP=SHR,DSN=IMS.SDFSMAC
//SYSIN    DD  *
 PSBLIB
 MBR=YOURPSB
 IMSCATHLQ=DFSCHLQ
/*
//SYSAIN   DD  SPACE=(80,(2500,2500)),RECFM=F,LRECL=80,BLKSIZE=80
//SYSLIN   DD  SPACE=(80,(2500,2500)),RECFM=F,LRECL=80,BLKSIZE=80
//DBDSOR   DD  DISP=OLD,DSN=IMS.DBDLIB.SOURCE
//DBDLIB   DD  DISP=OLD,DSN=IMS.DBDLIB
//PSBSOR   DD  DISP=OLD,DSN=IMS.PSBLIB.SOURCE
//PSBLIB   DD  DISP=OLD,DSN=IMS.PSBLIB
//ACBLIB   DD  DISP=OLD,DSN=IMS.ACBLIB
//SYSPRINT DD  SYSOUT=*
//LUSYSPRT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
```

Note, that you can only specify DBDSOR, PSBSOR, DBDLIB, PSBLIB or ACBLIB in the SYSIN control statements. In the case you only want to rebuild single sources or members you can specify the name of the DBD or PSB using the MBR parameter (there is no way to specify more than one DBD or PSB in a single execution of this utility using the MBR parameter).

For the IMSCATHLQ parameter you need to either provide the Directory HLQ or the name of the corresponding DFSMDA member (replace DFSC if you are using a different Catalog alias).

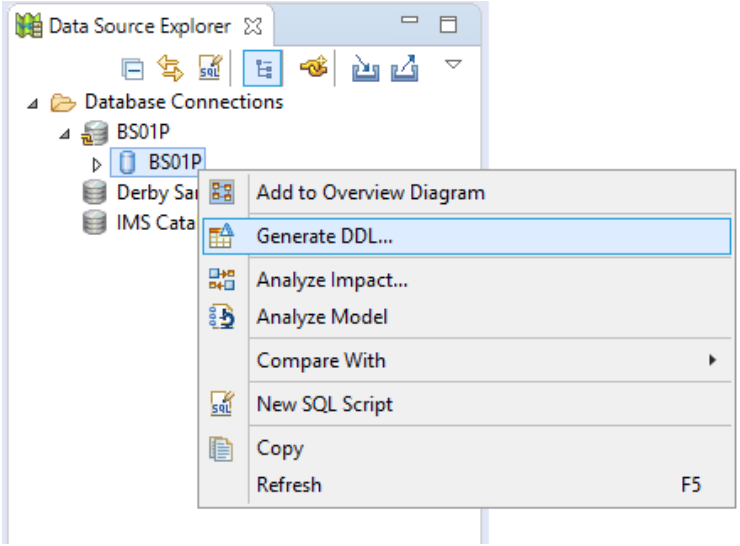
finanz informatik technologie service

Migration to a DDL-only environment

Preparing your SCM for DDL

- Currently DDL statements can be generated either using IMS Explorer or using the DDL Generation Batch Utility and either using DBD and PSB sources or the Catalog as input
- If you are using a source control management (SCM) system like CA Endevor or Serena ChangeMan to manage your DBD and PSBs sources and to generate DBDs / PSBs today, you need to import the DDL statements into your SCM and enable it to execute DDL statements
- Keep in mind, that when using DDL to create DBDs with indexes or logical relationships the DDL statements for the DBD and all of their indexes and /or logically related DBDs needs to be executed at once (there cannot be a commit in between)
- Before importing the DDL statements into your SCM you need to decide on whether to concatenate the DDL statements for DBD and all of their indexes and/or logically related DBDs and treat them as one resource or whether to model these kinds of relationships in your SCM and enable the SCM to concatenate the DDL statements of related DBDs before the execution
- Keep also in mind, that you need to manage both DBD and PSB sources and traditional gen processes as well as DDL statements and DDL execution in the transition time

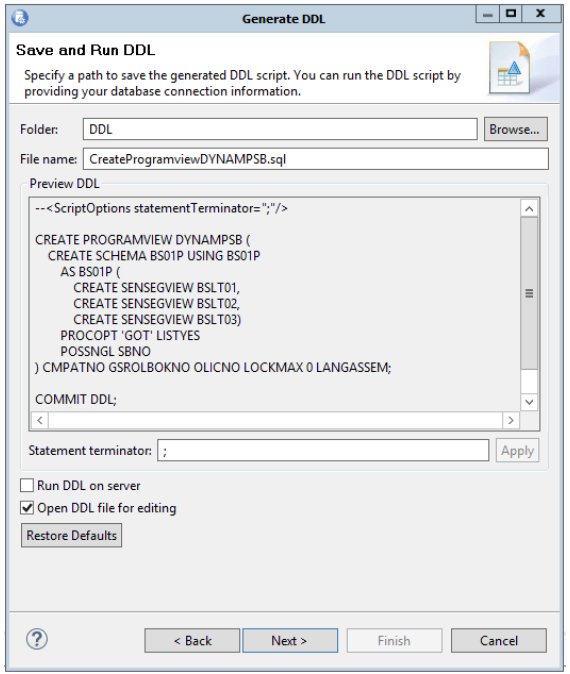
Generating DDL using the IMS Explorer



Create a new database connection in the *Data Source Explorer View* like you did it before for the Catalog. This time use the PSB you want to generate DDL for or a PSB containing a DB PCB referencing the DBD you want to generate DDL for. Name the connection profile like the PSB or the DBD (in the screenshot the DBD name *BS01P* is being used as the connection profile name). Be aware, that the DB PCBs must have PCBNAMEs or labels – otherwise IMS Explorer cannot use them, because ODBM uses the AIB interface.

After creating the database connection right-click on the entry with the database symbol below the connection profile and select *Generate DDL*.

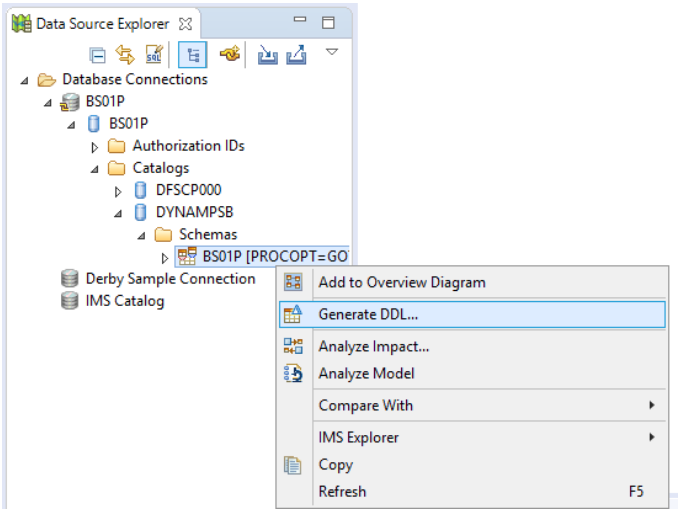
Generating DDL using the IMS Explorer



The *Generate DDL* dialog appears. Click *Next* twice. You will now get an informational message you can close by clicking on *OK*. Afterwards you will see a dialog window like in the screenshot.

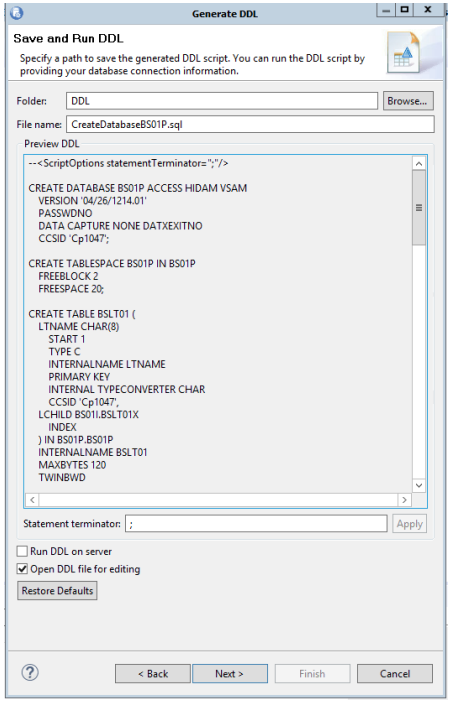
The DDL statement generated for the PSB is being previewed in this dialog window. To save it as a SQL script you need to provide a file name for the SQL script. If you check *Open DDL file for editing*, then click *Next* and finally on *Finish*, the SQL script will be opened on the right side.

Generating DDL using the IMS Explorer



If you want to generate DDL statements for a DBD open the items of the database connection until you reach the schemas. Select the schema named like the DB PCB that grants you access to the database for which you like to generate DDL, right-click on it and select *Generate DDL*.

Generating DDL using the IMS Explorer

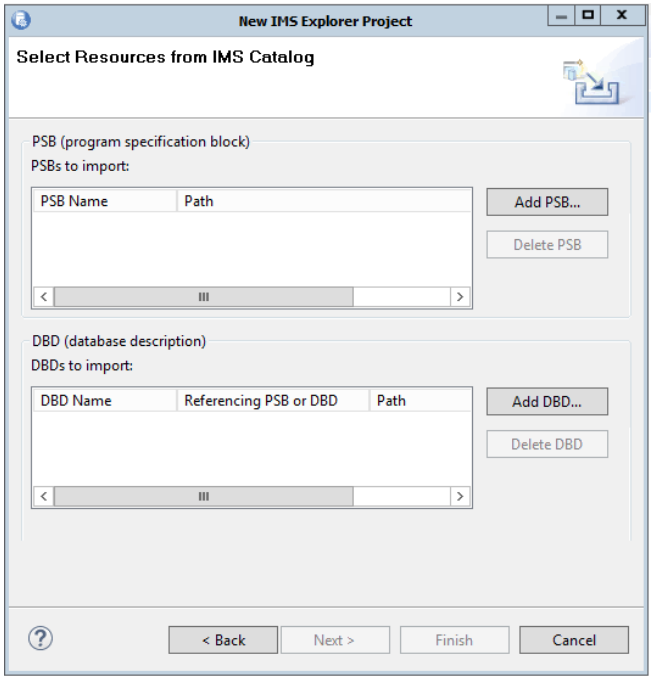


In the upcoming dialog click *Next* once, then deselect *Program view* (you already generated the DDL for the PSB) and click *Next* again. Again an informational message appears. Close it by clicking on *OK*.

You will now see again a dialog window like in the screenshot. Provide a name for the SQL script, select *Open DDL file for editing*, click on *Next* and finally on *Finish*. The SQL script with the generated DDL will be opened on the right side.

You now have the DDL for both PSB and DBD in your *Data Development Project*.

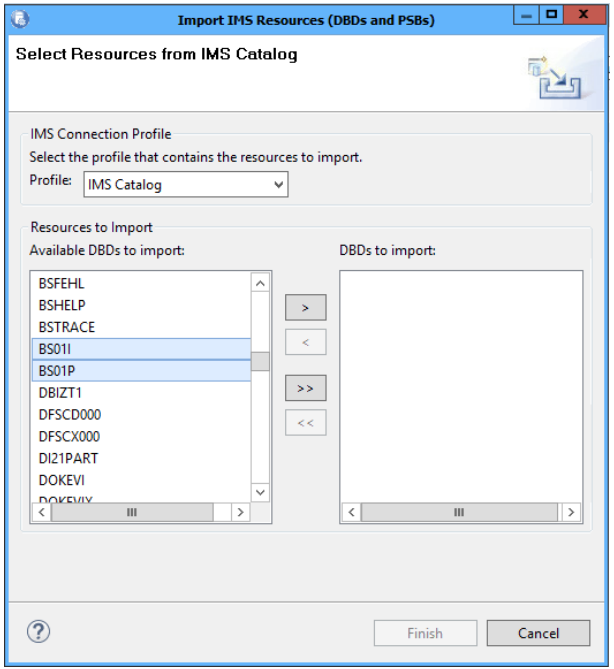
Generating DDL using the IMS Explorer



A different way to get DDL statements for your DBDs and PSBs generated in IMS Explorer is to import your DBDs and PSBs into an IMS *Explorer Project*. Therefore switch back to the *IMS Explorer* perspective in which you started before switching to the *Data* perspective by clicking on the IMS Explorer symbol in the right upper edge.

Now create a new *IMS Explorer Project* by navigating through the menu *File >> New >> IMS Explorer Project*. Provide a name for the project in the upcoming dialog – for instance *DBDs and PSBs* – and click *Next*. Then select the *IMS Catalog* in the *Import from* dropdown menu and click *Next* again. A dialog window like in the screenshot appears. Click on *Add DBD* to add the DBDs which should be imported.

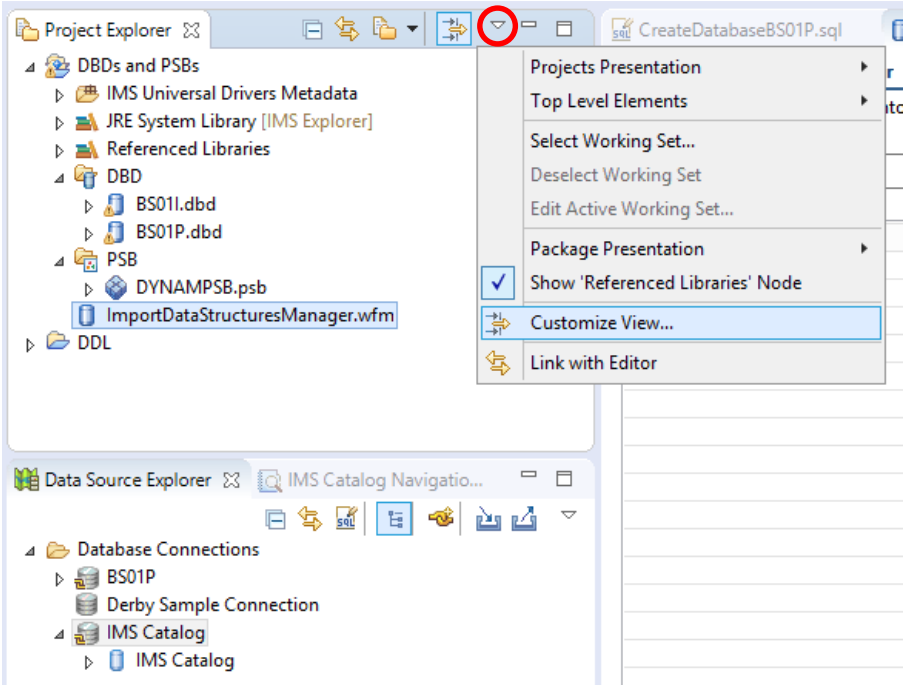
Generating DDL using the IMS Explorer



Select the IMS Catalog connection profile for the import. IMS Explorer now retrieves a list of all DBDs which are in the IMS Catalog. After the list has been retrieved, select the DBDs you want to import and click on the arrow. If you want to import all, click on the double arrow. When every DBD you want to import is in the right list, click on *Finish*.

Repeat the same for the PSBs you want to import by clicking on *Add PSB* in the dialog window from the previously shown screenshot, selecting again the IMS Catalog connection profile and the PSBs you want to import after the PSB list has been retrieved. Then click on *Finish*. Finally click on *Finish* again in the previously shown dialog window. The DBD and PSB import now starts. You may be prompted that your DBDs are missing structural information from your COBOL copybooks or PL/I includes. Click on *OK*.

Generating DDL using the IMS Explorer

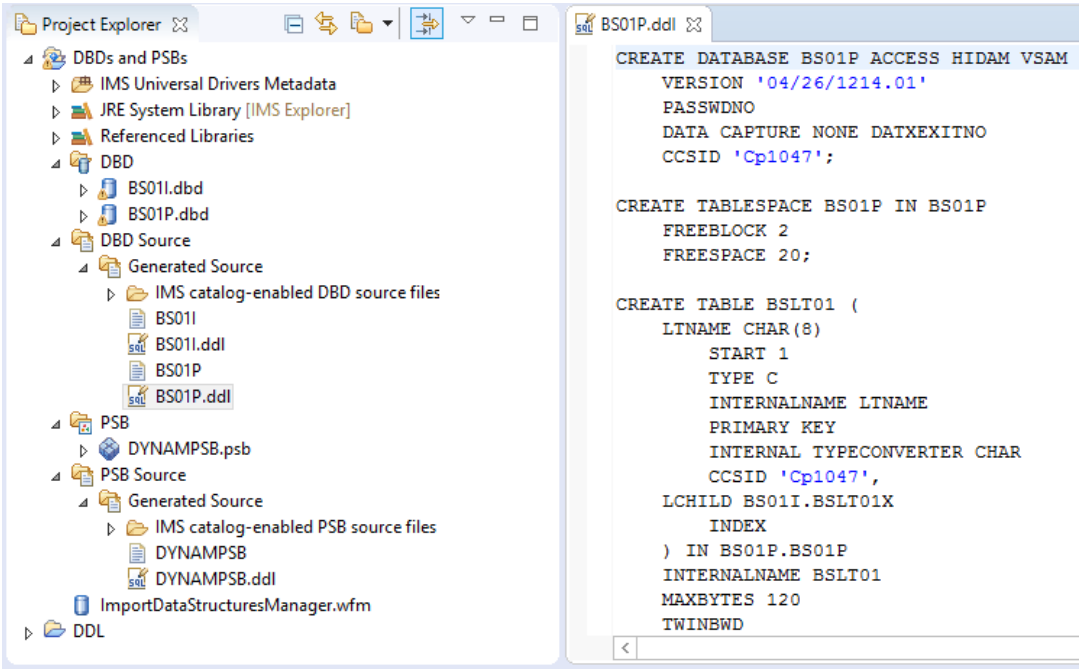


DBDs and PSBs now have been imported and can be visualized, their sources can be shown and they can be edited (for instance you can add your COBOL copybooks or PL/I includes to the DBDs to SQL-enable them).

Even if you can‘t see it, the DDL for every DBD and PSB you have imported has been generated in the background. To make the DDL visible, you need to click on the arrow pointing down in the *Project Explorer View* and select *Customize View*.

In the upcoming dialog uncheck *Hides DBD Source folder* and *Hides PSB Source folder* and click on *OK*.

Generating DDL using the IMS Explorer



You can now see and open the DBD and PSB source folders which are used for IMS Explorer-internal processing. They contain DDL files. You can open them by double-clicking on it. In contrast to the SQL scripts in the *Data Development Project* you can‘t execute them. You would need to copy the DDL statements into SQL scripts in such a project to execute them.

Navigate with a FTP client into your workspace if you want to upload the DDL onto your LPAR.

Generating DDL using the DDL Generation Utility

- While IMS Explorer may work for DDL test purposes, it isn‘t the most comfortable way to migrate all your DBD and PSB sources in your SCM to DDL statements
- The easiest way to do this is the DDL Generation Batch Utility, which is part of the same Java archive as the SQL Batch Utility presented before
- It allows you to either generate DDL from the IMS Catalog or from DBD and PSB source libraries
- You can use it to generate DDL for all or for specific DBDs respectively PSBs
- If you aren‘t sure that the definitions in your SCM still match what is in your IMS systems, we recommend you to use the IMS Catalog as the source for the DDL generation (we experienced a lot of differences between what is defined in our customer‘s IMS systems and what they have in their SCMs – especially for DBDs)
- You need to have IMS Connect and ODBM in place to be able to use the IMS Catalog as the source for this utility
- We strongly recommend you to wait for APAR PH18077 before using this utility, because this APAR solves lots of issues in it (the APAR will probably be shipped within the next two months)

Generating DDL using the DDL Generation Utility

```
//DDLGEN   EXEC PGM=JVMLDM86,REGION=0M,
//            PARM='com.ibm.ims.jdbc.batch.DdlGenerator'
//STDENV   DD *
export JAVA_HOME=/usr/lpp/java/J8.0
export PATH=/bin:"$JAVA_HOME"/bin
export LIBPATH=/lib:/usr/lib:"$JAVA_HOME"/bin:"$JAVA_HOME"/bin/j9vm
CLASSPATH=/usr/include/java_classes/IRRRa...jar
export CLASSPATH="$CLASSPATH":/usr/lpp/ims/imsjava/imsudb.jar:
export IBM_JAVA_OPTIONS="-Xmx1G -Dfile.encoding=Cp1047"
/*
//MAINARGS DD DUMMY
//SYSIN    DD *
CONNECT jdbc:ims://12.34.56.78:9999/DFSCP000:applName=YOURAPPL;;
DBDCAT(*);
PSBCAT(*);
/*
//DDLDBD   DD DISP=SHR,DSN=IMS.DBDLIB.DDL     PRE-ALLOCATED DDL STATEMENT
//DDLPSB   DD DISP=SHR,DSN=IMS.PSBLIB.DDL     LIBRARIES WITH RECFM=FB,LRECL=80
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Java Interface to RACF used for PassTicket generation

Memory limit for the JVM

Your EBCDIC codepage

Note, that JVMLDM80 is the name of the 31bit Java 8.0 JZOS Batch Launcher. If you want to use the 64bit Java 8.0 JZOS Batch Launcher you need to specify JVMLDM86. You may need to modify this for future Java releases. Ensure that JAVA_HOME, PATH and LIBPATH settings point to corrensponding Java SDK. You will get errors if you are using the wrong Java SDK with the JZOS Batch Launcher you specified before.

finanz informatik technologie service

Generating DDL using the DDL Generation Utility

```
//DDLGEN   EXEC PGM=JVMLDM86,REGION=0M,
//            PARM='com.ibm.ims.jdbc.batch.DdlGenerator'
//STDENV   DD *
export JAVA_HOME=/usr/lpp/java/J8.0
export PATH=/bin:"$JAVA_HOME"/bin
export LIBPATH=/lib:/usr/lib:"$JAVA_HOME"/bin:"$JAVA_HOME"/bin/j9vm
export CLASSPATH=/usr/lpp/ims/imsjava/imsudb.jar
export IBM_JAVA_OPTIONS="-Xmx1G -Dfile.encoding=Cp1047"
/*
//MAINARGS DD DUMMY
//SYSIN    DD *                          DDL GENERATION FROM DBD AND PSB SOURCES
DBDSRC(*);
PSBSRC(*);
/*
//DBDSRC   DD DISP=SHR,DSN=IMS.DBDLIB.SOURCE  DBD SOURCE LIBRARY
//PSBSRC   DD DISP=SHR,DSN=IMS.PSBLIB.SOURCE  PSB SOURCE LIBRARY
//DDLDBD   DD DISP=SHR,DSN=IMS.DBDLIB.DDL     PRE-ALLOCATED DDL STATEMENT
//DDLPSB   DD DISP=SHR,DSN=IMS.PSBLIB.DDL     LIBRARIES WITH RECFM=FB,LRECL=80
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Note, that the PARM value as well as the STDENV and the SYSIN in-stream datasets shown on the slide before and on this slide are case-sensitive. Ensure to not have line numbers in column 73 to 80 of both in-stream datasets.

Deciding for a leading source

- Because you probably won‘t fully migrate all stages to DDL at once, you need to decide whether DBD and PSB sources or DDL statements will be the leading source in SCM after you migrated the first stage to DDL
- If you decide to let DBD and PSB sources be the leading source until production is being migrated to DDL, you need to do all modification in DBD and PSB sources and automatically trigger the generation of DDL statements after any modification to a DBD or PSB source
- If you decide that DDL statements will be the leading source, you need to do all modifications using DDL statements and automatically trigger the generation of DBD and PSB sources after any modification using DDL statements (this is currently harder to implement, because at the moment there is no utility to generate DBD and PSB sources directly from DDL statements)
- No matter what you decide to be the leading source, don‘t forget about the relationships between DBDs when designing the process to get the other type of source
- We don‘t recommend you to manually update both sources, because it is very easy to accidently forget one or make different changes in both

Altering vs. dropping and re-creating DBDs / PSBs

- When modifying a DBD currently the DBD source is being modified, a DBDGEN and an ACBGEN is being performed then the old DBD will be replaced by using an ACB online change while the database is being stopped
- With DDL you have two options:
  - You can use an ALTER statement to modify the DBD
  - Or you can a DROP statements followed by the modified CREATE statements within the same commit scope
- The first option may be more comfortable to use for DBAs / sysprogs, but the second may be easier to manage with the SCM
- No matter what you decide to do, the modified control block then will need to be activated using the IMPORT DEFN SOURCE(CATALOG) command while the database is being stopped (metadata-only modifications done using DDL currently don‘t even require an activation)
- Because there is currently no ALTER PROGRAMVIEW statement, PSBs at the moment can be only modified using DDL by dropping and re-creating them within the same commit scope

Fallback considerations

- The fallback from DDL is mostly something you will have to handle in your SCM, because you won‘t need to fallback from all the preparations you have done in IMS to prepare for the execution of DDL unless you don‘t also need to fallback from IMS managed ACBs
- To ensure, that such a combined fallback won‘t happen, you shouldn‘t use DDL right after the migration to IMS managed ACBs
- In the case you are still using DLIBATCH and it still has been using the DBDLIB and PSBLIB after you migrated to IMS managed ACBs, you may first modify your DLIBATCH to use the Directory and not use DDL before you aren‘t sure, that DLIBATCH still works fine when using the Directory
- The same may apply to your database utilities and other kinds of IMS tools
- For being able to fallback from DDL in your SCM you still need to have up-to-date DBD and PSB sources in your SCM and not removed the generation processes
- You need to automatically trigger the generation of DBD and PSB sources whenever a DDL statement has been executed to be able to fallback from DDL or you need to generate up-to-date DBD and PSB sources for all DBDs and PSBs from the IMS Catalog when falling back

Fallback considerations

- We don‘t believe, that you will need to completely fallback from DDL (assuming that you will wait with the migration to DDL until IBM declares DDL to be production-ready)
- Most probably you will have issues with either building specific DBDs or PSBs or with the applications using specific DDL-built DBDs or PSBs respectively their Catalog metadata
- In such a case you should be able to switch back to traditional gens for those DBDs and PSBs
- The largest risk we see in the usage of DDL are potential differences in the Directory control blocks compared to the ones the combined ACBGEN and Catalog Populate Utility DFS3UACB builds (we already ran into such issues – for instance with PSBs for JMP/JBP applications, which have become non-Java PSBs)
- Such a difference may not necessarily immediately lead into an abend, it could also cause a database to be corrupted
- Because of this we are very thoroughly testing DDL
- Before we migrate a cutomer‘s system we will definitely backup all IMS databases as well as DBDLIB, PSBLIB and Staging ACBLIB

Fallback considerations

- During the migration of at least the customer‘s development systems we will rebuild all DBDs and PSBs using DDL
- After this we will ask them to very thoroughly test their applications to find such potentially possible differences in the control blocks, which could cause all kinds of issues
- We aren‘t sure yet whether we will do this in all of our customer‘s systems including production or not
- But doing this at least in the development systems and leaving enough time until the migration will be done in the production systems, will provide more security that the DDL-built control blocks for the customer‘s DBDs and PSBs are okay
- Not testing this with all DBDs and PSBs at once but testing this with later modifications to existing DBDs and PSBs is an alternative way, but we are sure, that the tests will get less and less thoroughly over time when DDL becomes business as usual
- When you are using SQL access to IMS databases you may also thoroughly check the Catalog metadata DDL builds (we also found differences in it)

Unresolved issues with DDL

Complex, unintuitive and partially inconsistent syntax

- The IMS DDL syntax is very complex, the way things are being defined are different from statement to statement and so unintuitive
- The IMS lab wanted to simplify the IMS database administration and make it more flexible with DDL, but they at least failed the first goal
- There are many inconsistencies in how parameter values are being set in the DDL statements: sometimes as keyword literals, sometimes as keyword with a value in paranthesis, sometimes within quotes and sometimes not, sometimes with commas in between and sometimes not
- While DBDs are defined with multiple DDL statements, PSBs are defined with a single DDL statement including all PCB definitions for the PSB in paranthesis, the same applies to segment and field sensitivity definitions within DB PCBs
- IMS-specific terms have been only partially replaced by SQL terms and even those SQL terms sometimes are inconsistently being in DDL and SQL access to IMS databases
- There are also still too many abbreviations instead of meaningful terms

No native utilities for DDL

- At the moment DDL statements can only be executed, if you have IMS Connect, ODBM and Java setup and configured in your environment
- This makes it very hard to adopt DDL
- IBM may create a native DDL utility in the future to lower the barriers of adopting DDL and ship it as part of continuous delivery
- Currently we don‘t know about any plans to also provide a native utility for the DDL generation, but maybe vendors will help with this
- If not, you at least need to setup and configure IMS Connect, ODBM and / or Java in one environment and generate all your DDL there or you need to import all DBD and PSB sources into IMS Explorer and then export the generated DDL from there

ODBM doesn‘t free storage used for DDL processing

- During our first intensive DDL tests about two years ago we ran into an U3062 abend of ODBM
- Until this point we executed several hundred DDL statements, which caused the amount of main storage ODBM was using to increase to several GB and finally lead into this abend
- During further investigations we found out, that ODBM doesn‘t free at least parts of the storage it is using for DDL processing
- Even when reguarly performing commits during DDL processing the amount of main storage ODBM uses will noticeable grow over time
- IBM will fix this with APAR PH34557

Maximum DDL statement length

- We have a few very complex databases that are accessed by our customers using SQL
- When trying to define one of them using DDL, IMS Explorer respectively the SQL Batch Utility fails to send the DDL statements to ODBM, because they are exceeding the current maximum DDL statement length of 65,535 bytes
- Just to get an idea on which statement size IBM would need to support, we used the DDL Generation Batch Utility to generate DDL for our most complex database and it ended up in DDL statements with a total size of 3,216,880 bytes (the CREATE TABLE statement for the largest segment in this database is already more than half of this size large)
- IBM will fix this issue with APAR PH23749

CREATE TABLE

- If you are trying to create a segment with more than 913 fields using a CREATE TABLE statement, DLISAS will terminate with a S0C4 abend
- In our most complex database we currently have more than 3,000 fields in a dozen mapping cases on one segment and more than 4,000 fields in a few hundred mapping cases on a second segment
- There is currently no APAR open yet for this issue
- DDL incorrectly fills INTERNALTYPECONVERTER settings into Catalog MAR/CMAR segments for FIELDs a USERTYPECONVERTER has been defined for
- It may also overwrite the correct field length by an incorrect field length, because it incorrectly applies defaults for fields INTERNALTYPECONVERTER are used for
- This may cause lots of different issues with Java applications, JDBC-based tools, the SQL Batch Utility, the Directory Recovery Utility, the Catalog Library Builder Utility and the DDL Generation Batch Utility
- This issue also applies to ALTER TABLE statements
- IBM will fix this issue with APAR PH30239

ALTER DATABASE | TABLESPACE | TABLE

- DDL currently doesn‘t rebuild referencing PSBs when you perform a structural change to DBDs using an ALTER DATABASE, ALTER TABLESPACE or ALTER TABLE statement
- IBM will fix this issue with APAR PH25758
- Adding a new dependent segment to a database using a CREATE TABLE statement may lead into a loop in ODBM
- IBM will fix this issue with APAR PH34098
- Trying to modify the relative start position of a subfield of a STRUCT or ARRAY with an ALTER TABLE statement will lead into SQL error code -199
- IBM told us, that this issue will be fixed with APAR PH25235, but it doesn‘t resolve the problem
- Currently there is no way to modify mapping cases or fields within a mapping case using an ALTER TABLE statement
- There is no APAR open yet for this issue

DROP DATABASE

- In an IMSPlex with a shared Catalog and Directory, deleting a GSAM DBD with a DROP DATABASE statement will lead into a S0C4 abend of the IMS Control Region
- All IMS systems except the one, which processed the DDL statement will abend with this S0C4
- IBM will fix this with APAR PH32934
- Under very special, but not yet finally clarified circumstances a DROP DATABASE statement can lead into a loop in DLISAS
- We expect an APAR to be opened on this soon after the circumstances have been finally analyzed by IBM

CREATE PROGRAMVIEW

- When you try to create a PSB without any alternate TP PCB, database PCB or GSAM PCB with DDL, you will receive SQL error code -104
- IBM will fix this issue with APAR PH23481
- Creating PSBs with DB PCBs for a logical DBD may fail with SQL error code -9055 because DDL is unable to find a dependent segment in the logical DBD
- As a workaround you can use the REFERENCES attribute to define the parent segment of the segment in the DB PCB (which usually isn‘t necessary anymore in DDL)
- IBM has confirmed that DDL will be improved to correctly handle this without needing to use the REFERENCES attribute, but hasn‘t opened an APAR yet for this
- DDL causes too many DBD/PSB cross-reference segments to be written into the IMS Catalog
- There is no APAR open yet for this issue

ALTER PROGRAMVIEW

- Currently there is no ALTER PROGRAMVIEW statement to modify PSBs
- You need to delete the PSB using the DROP PROGRAMVIEW statement and create the modified PSB again using the CREATE PROGRAMVIEW statement
- When this is being done within the same commit scope, it behaves like an ALTER but it may appear to be less comfortable / less intuitive to developers / DBAs / sysprogs

Automatic deletion of Directory control blocks

- Whenever you execute DROP PROGRAMVIEW or DROP DATABASE statements and commit these changes, subsequently executed DDL statements may run into a conflict with the automatic deletion of the Directory control blocks IMS performs in the background
- IBM will fix this with APAR PH32897

Staging Directory not backed out

- When you execute multiple DDL statements within the same commit scope and the DDL Parser successfully parsed the DDL statements, but the Block Builder fails to build the second or any later Directory control block it is requested to build, all changes to the IMS Catalog will be backed out but control blocks which previously have been successfully built and written to the Staging Directory won‘t be backed out – which means Catalog and Directory are out of sync
- There is no APAR open yet for this issue

DDL security

- The only way to secure DDL execution you currently have is to protect the PSB DFSCP001 using the RACF class AIMS (or A<rclass>) or protect the program DFSCP001 using the RACF class IIMS (or I<rclass>)
- By defining ODBMSECURE=R in the Common Service Layer section of the DFSDFxxx member, the AIMS class will be used to protect the usage of a PSB with ODBM while the IIMS class will be used to protect the IMS-internal usage of the PSB
- This allows you to differentiate between users who should be able to execute DDL and those who should be able to perform ACBGENs using DFS3UACB
- But there is no way to allow only IMS DBAs to create, alter and drop IMS databases when you on the other hand want your developers to create their PSBs on their own
- There is also no way to differentiate between CREATE, ALTER and DROP permissions
- You can also not prevent users who have permission to use PSB DFSCP001 from doing any Catalog manipulation by using SQL UPDATE or DELETE statements instead of DDL statements

DDL error code and error message handling

- We experienced several cases, in which the DDL processing fails or didn‘t came to a successful end, but the SQL Batch Utility ended with RC=0 or IMS Explorer showed a successful execution
- IBM will fix some of these issues with APAR PH10681
- If you forget to commit the DDL statements, you may also get the impression that the execution of the DDL statements was successful, but in the end you won‘t have created any new DBD or PSB instance in the Catalog or any new/updated member in the Directory (which won‘t be fixed by this APAR)
- But this APAR will also remove the message

  ```
  Disconnecting before commiting, rollback was issued.
  ```

  the SQL Batch Utility currently incorrectly prints even after a successful DDL execution
- Besides this there are lots of issues with SQL error codes not matching the error message, SQL error code and error message not matching the documentation, incomplete and misleading error messages
- IBM will fix a lot of these issues with APAR PH31690

DDL error code and error message handling

- In addition to this APAR PH31768 will provide an unique SQL error code and a meaningful error message when using a reserved SQL keyword as a name for a DBD, dataset, segment, field, XDFLD, mapping, mapping case, PSB or PCB
- If you have such a name you can surround it by double quotation marks
- DDL then will treat it as a name and not as a SQL keyword
- There are also several issues with messages DDL prints into the job log of the IMS Control Region - for instance the PSB name is printed as reason into message DFS4774E
- IBM will fix these issues with APAR PH27321

DDL Generation Batch Utility

- When generating using the DDL Generation Batch Utility you may run into the following problems:
  - No DDL statements will be generated for GSAM DBDs and DLIBATCH PSBs when there are multiple instances in the Catalog
  - The values for different database dataset attributes are not or incorrectly processed
  - Long EXTERNALNAMES and remarks cause the DDL generation to fail
  - A comma may be missing in front of mapping definitions within CREATE TABLE statements
  - COMMENT ON statements for fields with different NAME and EXTERNALNAME are incorrect
- IBM will fix these issues in APAR PH18077
- Some of these issues may also appear when using the current release of IMS Explorer to generate DDL statements, but the next release probably includes the fixes from APAR PH18077
- You may experience very long runtimes when generating DDL statements from IMS Catalog even after applying APAR PH18077
- This will be addressed in a subsequent APAR, which hasn‘t been opened yet

DDL Generation Batch Utility

- The DDL Generation Batch Utility may also fail to create DDL statements for PSBs containing remarks, because the GUR call returns an invalid XML in this case
- The DBD-wide encoding won‘t be correctly set into CREATE DATABASE statements, because the GUR call doesn‘t return the DBD-wide encoding
- IBM will fix both issues with APAR PH23490
- Message DFS552I informing about the stop of a DMU region is incorrectly being printed to the job log of DLISAS instead of to the job log of the IMS Control Region
- There is no separate APAR open for this issue, IBM informed us that this minor issue will be fixed within one of APARs opened for a different DDL issue but not yet in which of those

Catalog Library Builder Utility

- The Catalog Library Builder Utility is very important for both the time in which not all stages have been migrated to DDL and for the fallback from DDL, but it still has several larger problems:
  - It isn‘t able to create DBD sources or DBD control blocks for GSAM databases or logical databases
  - It skips to create GSAM PCBs into PSB sources or PSB control blocks
  - SQL metadata will get lost when creating DBD or PSB sources respectively DBD, PSB or ACB control blocks if the corresponding Directory control block has been either recoverd by the Directory Recovery Utility DFS3RU00 or created/modified using DDL
- If you haven‘t yet applied APAR PH14297 you may encounter even more issues with this utility
- IBM will fix the GSAM-related issues with APAR PH30248
- There are no APARs open yet for the other issues

Catalog secondary index

- We ran into a still unclear issue with very extreme runtimes when accessing the IMS Catalog through its secondary index to get DBD/PSB cross-reference information
- It took IMS up to 45 minutes and millions of I/Os to get the PSBs referencing some of the DBDs
- For a comparison we ran the same kind of query without using the secondary index and even though the whole Catalog needs to be scanned in this case it tooks only a few seconds
- This only happens in a system in which we built the whole IMS Catalog using DDL (except for the resources DDL is still unable to process)
- IBM found out, that this doesn‘t happen anymore when IMS managed ACBs will be turned off (without changing anything in the IMS Catalog database or its secondary index)
- Nevertheless they haven‘t yet been able to determine the cause of this issue
- The issue may not necessarily be a DDL issue, but we neither encountered this during the five years we are now operating Catalog-only environments nor during the intensive IMS managed ACBs test we have done and the two years we are now operating several IMS managed ACBs environments

Catalog secondary index

- When you have APAR PH30666 applied and perform a DROP DATABASE statement on a GSAM DBD, DDL performs the checks for referencing PSBs using the Catalog secondary index
- Due to this issue the execution of such a DROP DATABASE statement may take much longer than expected and use much more resources than expected or may even fail when an ODBM timeout occurs while IMS Explorer / the SQL Batch Utility waits for a response from IMS
- Currently this can‘t happen when dropping other types of DBDs, but this may change in the future, because there are plans to harmonize the currently different behaviors of DDL regarding the drop of normal DBDs and GSAM DBDs