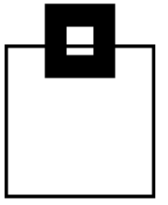
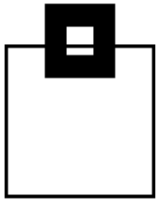


Space Troopers enter the ZOWE world

Db2 mission control for the next generation

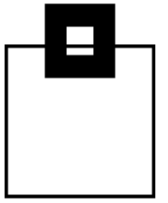
Roy Boxwell

Agenda



- Db2 limits
  - DSSIZE or PIECESIZE among other basic simple things
  - Linear Dataset limits (32 for segmented spaces, 254 for LOBs)
  - COPYPOOL
  - PBGs & MAXPARTITIONS
  - SEQUENCES and numeric Primary Key usage
  - Other limits of Db2 (DBATs etc.)
- Resource monitoring - Different types for different needs
  - Manual
  - Iterative
  - Real-time
- Zowe integration and exploitation
  - Pre-reqs you have to get right
  - Knowledge you must have
  - What we did and How we did it
  - How it looks today (Tomorrow will be different!)

Limits through the machine

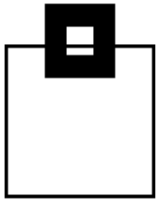


They say that space is infinite - up to a point

The same is true for Db2!

If you can imagine having infinite disk space and infinite memory, what limits could there possibly be?

Limits through the machine



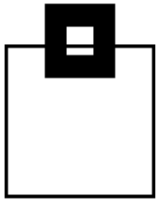
In the beginning was the VSAM Cluster...

Well before Db2 saw the light of day
Older than me!

Size limit of 4GB per dataset - that is HUGE!
We will never get that much data...

Remember Bill Gates and the apocryphal story about 640KB of RAM...

Limits through the machine

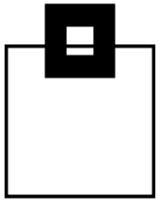


The VSAM limit is actually split into different limits that the DBA and Db2 must work with:

- Simple/Segmented VSAM Linear Dataset (LDS from now on) is limited to 2GB
- Partitioned Space 1GB, 2GB, 4GB, 8GB, 16GB, 32GB, 64GB

For larger than 4GB objects you must have a data class in SMS with the Extended Format and Extended Addressability set (If in Db2 12 then check out APAR PH10015 UI64089).

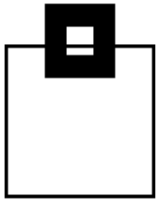
Limits through the machine



Index sizes are dependent on what they are and how they, and their table, are defined:

- Partitioned Indexes only have one dataset but their maximum size varies with usage of LARGE, DSSIZE and the number of PARTITIONs
- Non-partitioned Indexes can have PIECESIZE to enable multiple datasets which start at 256KB and then binary step all the way up to 268,435,456KB! The number of these is also dependent on LARGE, PIECESIZE and the number of PARTITIONS

Limits through the machine



Here you can see that a seemingly simple question:

How big can my partitioned index get?

Gets a rather complex answer:

MIN (Table space DSSIZE , 2^{32} / (Max No. Parts * IX PGSIZE))

Eg: 64GB DSSIZE with 32KB TS Page Size give 2048 as Max No. Parts and 4KB IX Page size gives you 8GB as a maximum dataset size.

All clear on that??

Limits through the machine

With NPIs you also see that another seemingly simple question:

How many pieces can my non-partitioned secondary index get?

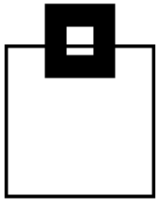
Gets another rather complex answer:

MIN (4096 , 2^{32} / (PIECESIZE / IX PGSIZE))

Eg: Tablespace with more than 64 partitions or DSSIZE used, IX with 8GB PIECESIZE and with a 4KB IX page size gives you a maximum of 2048 datasets.

All clear on that??

Limits through the machine



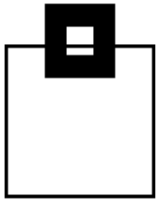
Hooray for Db2 12 FL500!

In Db2 12 we get the Relative Page Numbering (RPN) PBR with variable DSSIZE and *also* a variable DSSIZE for the partitioning indexes. Plus the DSSIZE GB does not have to be a binary number.

To do this the RID has increased to seven bytes but it completely decouples the number of partitions from the equation - which is a very good thing!

However, the rule is still true that for larger than 4GB objects you must have a data class in SMS with the EF and EA set.

Limits through the machine

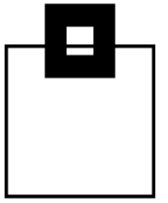


These dataset limits then go into the LDS arena:

- A simple or segmented space can have 32 LDSs
- A non-partitioning index can have a “number” of pieces

- A LOB space can have 254 LDSs
- A partition can only have one LDS of course!

Limits through the machine



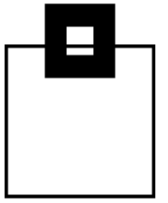
This is then the first set of limits for us.

You must monitor how many LDSs all of your simple, segmented and LOB objects have and you must get warned well before you hit the buffers! If you have 32 LDSs the REORG might take a while…

Partitioned objects are of course different…

Here you must monitor how ***full*** each partition is and how ***full*** each index is and how many pieces there are in your NPSI. Is that all?

Limits through the machine



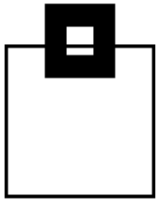
No! Of course not…

The Partition By Growth (PBG) Universal Table Space (UTS) construct brought with it some “new” problems:

1) No partitioning index allowed – only NPSIs on these
2) MAXPARTITIONS is the new LDS limit

So, for all your PBGs you must monitor not only ***how many*** partitions they currently have, but also how ***full*** is that very last partition when you have it allocated!

Limits through the machine



Even worse is what happens if your define a PBG space with MAXPARTITIONS 10, NUMPARTS 10, a CLUSTER YES index with all columns DESC?

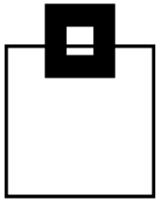
Can you guess?

First Db2 inserts to partition one, once it is full it then jumps the inserts to partition ten and when *that* is full it jumps backwards nine and then eight etc.

This is pretty horrible and even John Campbell hates it.

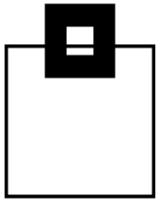
Luckily there is software that detects this and handles “gaps in PBGs” correctly.

Limits through the machine



Ok, so now you are checking how many LDSs of which size
for all of your different objects – Everything must be ok now??

Limits through the machine



No! Of course not…

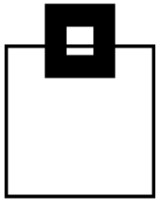
Think EXTENTS… many years ago the limit was 255 extents for a dataset but nowadays it is 7,257.

The sliding scale of extent allocation is active for all datasets with a SECQTY of -1 or a positive SECQTY and ZPARM MGEXTSZ = YES.

Note that implicitly created tablespaces always work as if MGEXTSZ = YES.

Also note that MGEXTSZ has been deprecated in Db2 12 (PH28280) and is now assumed always to be set to YES.

Limits through the machine

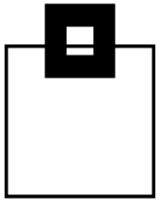


While thinking about EXTENTS remember that these all live on VOLUMES…

So you now have the 59 Volume limit to worry about (7,257 extents is 59 * 123 extents per volume which is the limit for a dataset on a volume).

You must watch out for message DSNP030I which is your “last volume allocated” warning. Running out of volumes is “sub-optimal”…

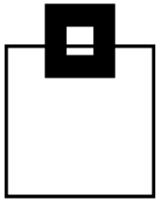
Limits through the machine



Ok, so now you are checking how many LDSs of which size with how many extents on how many volumes for all of your different objects

– Everything must be ok now??

Limits through the machine



No! Of course not...

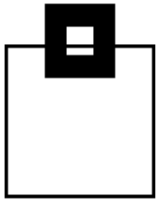
Think SMS Copy Pool sizes...When you are using flashcopy or just normal Image Copy you must guarantee that you have enough space to actually do all the copies you want to do...

Limits through the machine

This job is usually done by the storage team but I think it would be a good idea if the DBA also checked whether or not the copy pool lives up to its name

or is it only a puddle?

Limits through logic

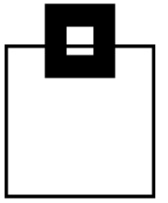


Now we get to the part where Db2 constructs start saying “no”

Think SEQUENCEs here… A quick look in the SQL guide will show you that when you create a SEQUENCE:

```
CREATE SEQUENCE ROY_TEST_SEQ_ASC
        START WITH 1
        INCREMENT BY 1
        MAXVALUE 9999
        CACHE 10;
```

Limits through logic



You actually get:

```
CREATE SEQUENCE ROY_TEST_SEQ_ASC
        START WITH 1
        INCREMENT BY 1
        MAXVALUE 9999
        NO CYCLE
        CACHE 10;
```


The “NO CYCLE” is the, by default, bad guy here…
This is a -904 waiting to happen…

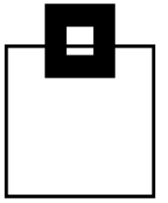
Limits through logic

For all of SEQUENCEs you must periodically see how close to the top or the bottom of your available range they are.
This is also true for Identity columns and XML DOC Ids:

```
SEQTYPE CHAR(1) NOT NULL
Type of sequence object:
     A Alias for a sequence
     I An identity column
     S A user-defined sequence
     X An implicitly created DOCID
       column for a base table that contains XML data.
```

You want to get these *way* before they hit the buffers!

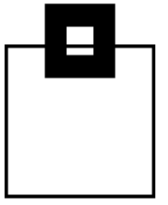
Limits through logic



Now in the time before SEQUENCEs lots of shops were using SMALLINT, INTEGER or DECIMAL defined fields in their tables'
Primary Keys doing exactly what a modern sequence does.

Naturally no-one has "updated" these pre-sequence sequences to use proper sequences and so you have another layer of danger lurking out there…

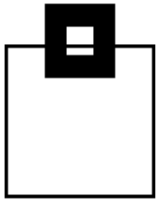
Limits through logic



You must find all your numerically defined primary key fields and see if they are approaching the Db2 numeric limits...

Item	Limit
Smallest SMALLINT value	-32768
Largest SMALLINT value	32767
Smallest INTEGER value	-2147483648
Largest INTEGER value	2147483647
Smallest BIGINT value	-9223372036854775808
Largest BIGINT value	9223372036854775807
Smallest DECIMAL value	$1 - 10^{31}$
Largest DECIMAL value	$10^{31} - 1$

Limits through logic



But what are you checking?

Are you looking at just the Db2 Catalog values after a RUNSTATS?

Or

Are you selecting all the columns dynamically from the User Data so that you get real values?

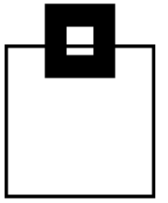
Horses for courses - as we say in England...

Limits through design

Design limits that require checking:

- How many columns in a table?
- How many columns in an index?
- How many bytes long is my index?
- Is there a Database limit?
- Is there an Object limit?
- Death by DBAT?

Limits through design

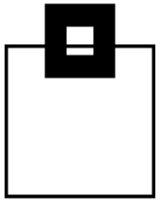


How many columns in a table?

From day one Db2 has allowed 750 columns as the absolute maximum.
Depending on View definitions you can actually be forced to have less...

Something to check just in case that ALTER ADD COLUMN is gonna fail horribly...

Limits through design



How many columns in an index?

From day one Db2 has allowed 64 columns as the absolute maximum. However the byte count varies depending on whether or not the index is a good old partitioning index (PI) or any other index.

For a PI you get a maximum size for:

- PADDED indexes of 255 – n – 3d bytes
- NOT PADDED indexes of 255 – n – 2m – 3d bytes

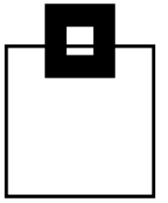
Where:

n is the number of columns which are NULLable

m is the number of varying length columns

d is the number of DECFLOAT columns

Limits through design



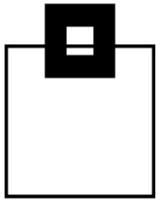
For any other indexes you get a maximum size for :

- PADDED indexes of 2000 – n – 3d bytes
- NOT PADDED indexes of 2000 – n – 2m – 3d bytes

Where:

n is the number of columns which are NULLable
m is the number of varying length columns
d is the number of DECFLOAT columns

Limits through design

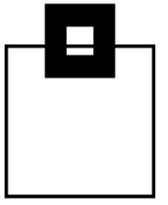


How many bytes long is my index?

In Db2 12 IBM came up with the Fast Traversal Block (FTB) which most people call Fast Index Traversal (FIT) and this comes with a bunch of limits:

1) Must be a unique index
2) Must have a total length <= 64 bytes
3) No TIMEZONE usage
4) No active versioning
5) Index partitions must have less than 2,000,000 leaf pages

Limits through design



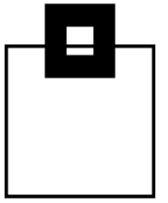
So you can see that you might well be using a FIT and then do e.g.

```
ALTER INDEX x.y ADD INCLUDE COLUMN ( col1 )
```

and *__boom__* the index is no longer eligible for FIT usage…

Check before *every* ALTER INDEX whether or not the index is used by FIT and whether or not your ALTER will tip it over the edge!

Limits through design

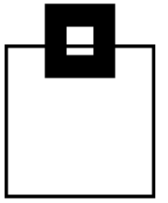


Is there a Database limit?

Yes indeed there is, but nowadays at 65,217 (Back in DB2 V5.1 it was 32,511) it is pretty hard to reach!

However, if you have a lot of implicit Tables/Tablespaces where no-one tidies up and DROPs the implicitly created databases they can sum up quickly! REORG mapping DBs are quite popular here...

Limits through design



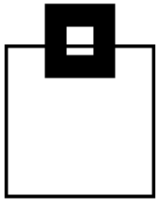
Is there an Object limit?

Actually yes there is!

- In a Database you can have a maximum of 32,767 OBIDs (Object Ids). Now remember the OBID is not 1:1 for any and all objects.
- Each tablespace, index or referential relationship takes two, whereas each table, check constraint, aux for LOB, XML for XML, trigger or view with INSTEAD OF takes one.

So make sure you check these counts on a regular basis as well.

Limits through design



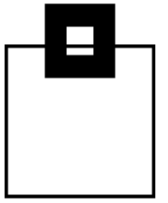
Death by DBAT?

You all know what happens when you run out of DBATs right?

It is not pretty…

Just a –DISPLAY DDF DETAIL is all you need…

Limits through design



Sadly not...

The problem here is that the DBAT counts that are output when you do a -DISPLAY are, of course, only the *local* counts...

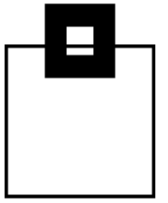
With Datasharing you only see the data of the Member that you are directly connected to which is pretty useless...

Limits through design

Here is an example output from a datasharing system:

```
DSNL080I  -SB11 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION           LUNAME            GENERICLU
DSNL083I xxxxxxxx           xxxxxxxx.xxxxxxxx -NONE
DSNL084I TCPPORT=xxxx  SECPORT=0     RESPORT=xxxx  IPNAME=-NONE
DSNL085I IPADDR=::xxx.xxx.x.xx
DSNL086I SQL    DOMAIN=xxxx.fritz.box
DSNL086I RESYNC DOMAIN=xxxx.fritz.box
DSNL087I ALIAS              PORT  SECPORT STATUS
DSNL088I TEST110            0     0       STOPD
DSNL089I MEMBER IPADDR=::xxx.xxx.x.xx
DSNL090I DT=A  CONDBAT=  10000 MDBAT=  200
DSNL091I MCONQN=    0 MCONQW=   0
DSNL092I ADBAT=    0 QUEDBAT=      0 INADBAT=      0 CONQUED=      0
DSNL093I DSCDBAT=      0 INACONN=      0
DSNL094I WLMHEALTH=100 CLSDCONQN=      0 CLSDCONQW=      0
DSNL100I LOCATION SERVER LIST:
DSNL101I WT IPADDR            IPADDR
DSNL102I 32 ::xxx.xxx.x.xx
DSNL102I 32 ::xxx.xxx.x.xx
DSNL105I CURRENT DDF OPTIONS ARE:
DSNL106I PKGREL = COMMIT
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Limits through design



So you have two choices…

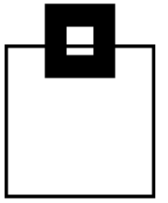
- Write a little REXX that runs a round robin style of –DISPLAYs
- Get all warm and cuddly with IFI processing to do it all in one call…

However you get the data, the interesting numbers are in the following two lines of output:


```
DSNL090I DT=A  CONDBAT=  10000 MDBAT=  200
DSNL092I ADBAT=    0 QUEDBAT=    0 INADBAT=    0
```

The MDBAT is your MAXDBAT value and ADBAT is the current number of DBATs. You *must* monitor this all the time to see if any member is running out of DBATs!

Future directions



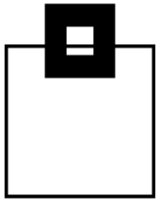
IBM keep lifting the limits of Db2.

Now with RPN the last big bottleneck has been broken – You still must do a TS Level REORG with TP level inline image copies to migrate to it and you need a new Mapping table but when you are there it is a much better green than where you are standing now!

Will they ever raise the other limits?

I do not know of course, but I hope they raise the eligibility of FITs soon!

Future directions



IBM keep lifting the limits of Db2.

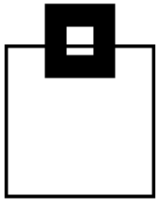
Now with RPN the last big bottleneck has been broken – You still must do a TS Level REORG with TP level inline image copies to migrate to it and you need a new Mapping table but when you are there it is a much better green than where you are standing now!

Will they ever raise the other limits?

I do not know of course, but I hope they raise the eligibility of FITs soon!

They are working on it! Check out PH30978

Monitoring

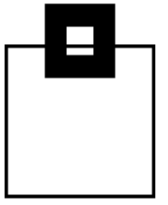


- Monitoring usually implies a realtime observation. However, if you don‘t look at the data yourself, it requires some alerting rules, otherwise you can just manually trigger a report.
- Of course realtime is always preferred, but some data does not change unless a change is initiated (e.g. schema), so realtime is sometimes as good as my last report created, but more expensive to gather!
- Some data is more expensive to gather than other (SQL, IFCID SMF vs. OPX, …)

→ Choosing the best data source and the right „monitoring“ strategy saves resources **and** protects from discovering problems too late

Update only if required, but make sure you are as current as needed!

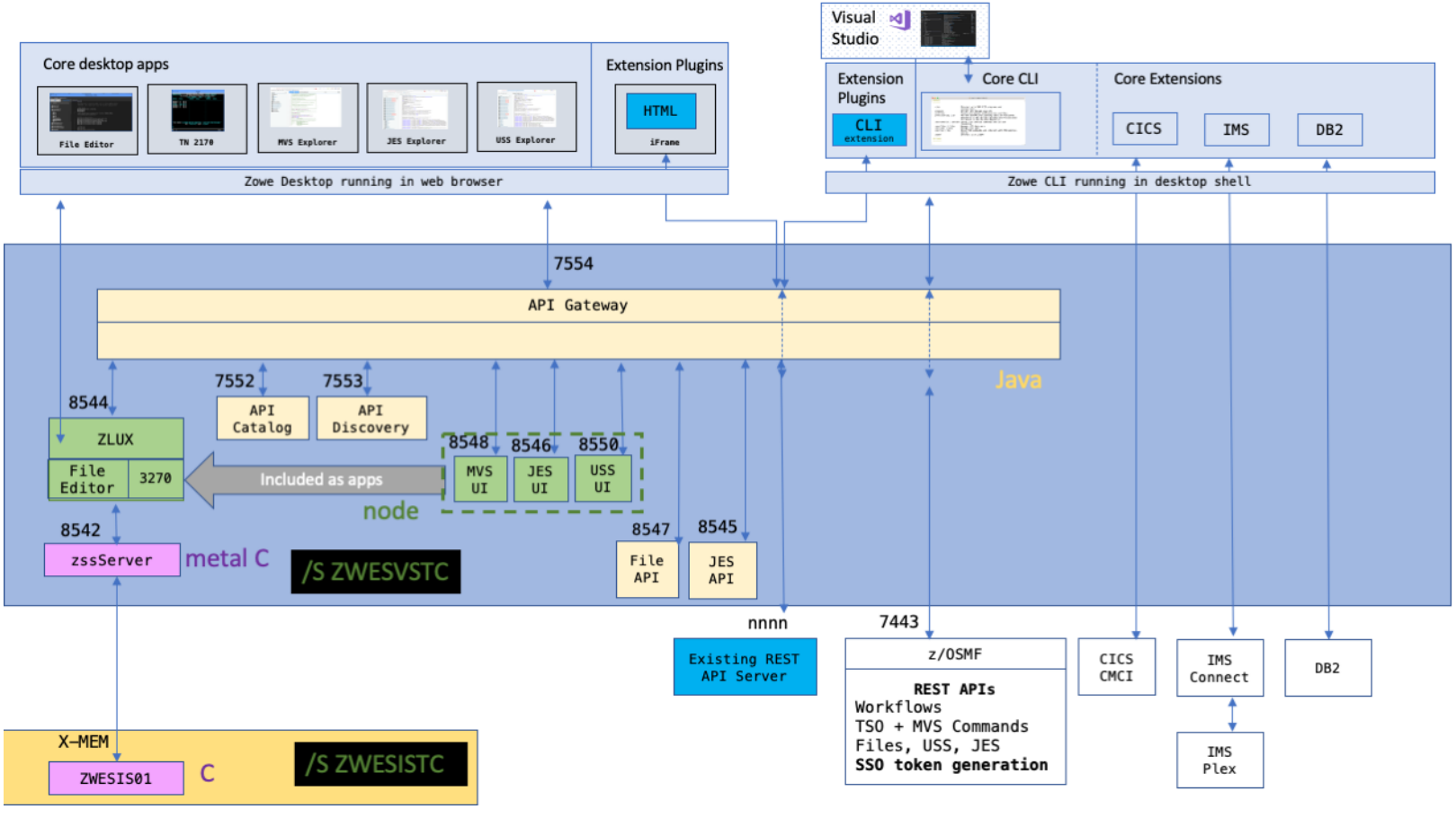
Monitoring



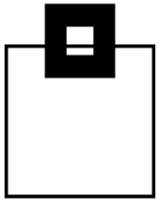
- Your needs may vary, but capacity planning is usually less realtime critical, as space assurance when it comes to extents, or LPS limits.
- Depending on your applications, data growth may be different
- Current data is only benefitial if someone notices and takes care of it

→ Are you as agile as your data?

Zowe components



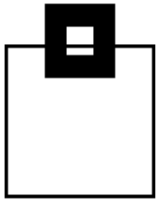
Zowe components



Zowe Application framework is four major components

1. Desktop
   Browser based web desktop
2. Application Server
   Web services framework plus proxy applications that communicates with z/OS services and components
3. ZSS Server
   REST services to support the Application Server
4. Application plug-ins
   Included and addable applications to access the mainframe and to perform various tasks, e.g.
   - Dataset editor and browser (z/OS and USS)
   - Workflows
   - z/OS subsystem browser (JES, CICS, Db2, IMS, …)
   - …

Zowe components



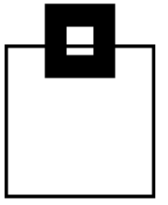
Zowe z/OS services contain the following core components

1. **z/OS dataset services**
   list, browse, edit, create, delete, … datasets and members
2. **z/OS job services**
   list, browse, submit jobs

A full list of capabilities of the RESTful API can be listed via the API catalog

- The Open API Specification describes the APIs and allows to use any standard-based REST API developer tool, or API management process
- APIs can be used by any application
- z/OS services are running as microservices with a Spring Boot embedded Tomcat stack

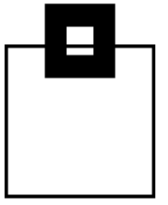
Zowe components



Zowe CLI comes with the following capabilities

- **Interact with files:**
  Create, edit, download, and upload data sets
- **Submit jobs:**
  Submit JCL from data sets or local storage, monitor the status, and view/download the output
- **Execute commands:**
  Issue TSO, or z/OS console commands
- **Integrated scripts:**
  Define scripts that do both mainframe and local tasks
- **Return JSON documents:**
  Return the data in JSON format to be used in other programming languages

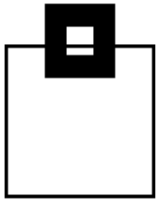
Zowe components



Zowe API mediation layer consists of the following components:

- API gateway
  - Clients interact with microservices behind a reverse proxy forwarding requests to the appropriate service
  - The gateway is built on Netflix Zuul and Spring Boot technology
- Discovery services
  - Accepts the REST service announcements and serves active ones
  - The service is built on Netflix Eureka and Spring Boot technology
- API catalog
  - UI catalog of published APIs along with their documentation (Swagger) and status
  - Services can be implemented by multiple instances for high-availability or scalability
- ESM microservices
  - Authenticates and authorizes users with mainframe credentials

ZOWE Our Journey (1:5)

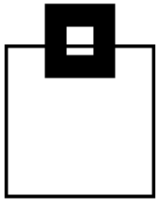


- ZOWE for SE started with version 0.9.5 and node.js on the PC – That‘s all you need!
  - Why did we choose ZOWE?
    - Modernize the Mainframe – The crowd is greying out there…
    - Multi-Factor Authentication (MFA) required – Nearly all of our customers require MFA these days and Eclipse based support is not being delivered by IBM
    - Much better GUI – HTML5 is way better than Eclipse with Jasper

ZOWE Our Journey (2:5)

- We chose our WorkloadExpert as the first of our Eclipse based plug-ins to be migrated to ZOWE
  - Largest of our GUIs with the most extensive use of graphics and reports
  - For this the API mediation layer and Application Framework from ZOWE were important
  - The Application Framework is what we call ZOWE desktop
    - It provides a virtual GUI but it can be accessed by browser
    - It serves as a starting point for preinstalled and external apps
    - It runs as a webserver on the mainframe

ZOWE Our Journey (3:5)



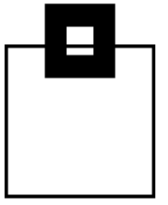
- To add your app to the ZOWE desktop you must use the ZOWE desktop technology stack
  - It consists of HTML, CSS and JavaScript for the GUI, Node.js for the data services and Java for the API Mediation layer

- ZOWE supports three frameworks for apps:
  - Angular
  - React
  - Iframe

- Angular and React are JavaScript Frameworks which allow you to develop highly interactive web apps also allowing the use of Typescript. This is then compiled into JavaScript.

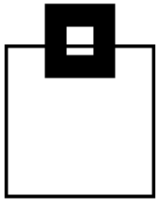
- The ZOWE desktop allows you to include Angular and React apps directly

ZOWE Our Journey (4:5)



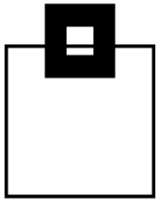
- Iframe is basically a website within a website
  - If you configure your app in ZOWE as iframe, it just takes the HTML and associated files (JavaScript, CSS, Images and other files) and serves as a website
  - If you take this approach you can include apps in ZOWE which have been built without even having ZOWE in mind
  - You can also still access functions provided by ZOWE because they are provided as a global variable
    - Eg URIBroker to avoid hard coding URLs or preference storage

ZOWE Our Journey (5:5)



- What we did and Why we did it!
  - Front-end: The GUI – user interaction and graphics. We chose Angular and we compile it with the angular-cli
  - Back-end: Mainframe with Db2 database. We chose to use an API via the API Mediation Layer instead of a data service
    - Why? The APIs have a wider applicability than data services as they are not only for the ZOWE desktop but for all ZOWE services
    - This could be seen as „overkill“ for our WLX App but the fact that the API can be written in Java allows us to re-use code from the original Eclipse plug-in
      - For this we use the Spring Boot framework

ZOWE Gotcha's



- We are a mainframe shop and so we had a little bit to learn…
  - z/OS Unix System Services: This must be installed and running well as it is an absolutely critical core requirement for ZOWE
  - Configuration and Profile files:
    - In Language Environment we found out that we had to raise the HEAP64 quite a bit:

      HEAP64(512M,4M,KEEP,256M,4M,KEEP,0K,0K,FREE)
    - The MEMLIMIT of the OMVS users must also be raised
    - In the profile datasets the important thing is to set up your ASCII:

```
#ASCII support the environment variables
export _BPXK_AUTOCVT=ON
export _CEE_RUNOPTS='FILETAG(AUTOCVT,AUTOTAG) POSIX(ON)'
export _TAG_REDIR_ERR=txt
export _TAG_REDIR_IN=txt
export _TAG_REDIR_OUT=txt
```

How it looks

- Here‘s how the json Iframe link to ZOWE looks:

```
VIEW       /Z23A/usr/lpp/seg/vnext/wlx/0.0.1/pluginDefinition.json
Command ===>
****** ***************************************************** Top
000001 ä
000002   "identifier": "de.seg.wlx.gui",
000003   "apiVersion": "0.0.1",
000004   "pluginVersion": "0.0.1",
000005   "pluginType": "application",
000006   "webContent": ä
000007     "framework": "iframe",
000008     "launchDefinition": ä
000009       "pluginShortNameKey": "WLXgui",
000010       "pluginShortNameDefault": "WLX GUI",
000011       "imageSrc": "assets/icon.png"
000012     ü,
000013     "descriptionKey": "WLX GUI",
000014     "descriptionDefault": "WLX GUI",
000015     "startingPage": "index.html",
000016     "isSingleWindowApp": true,
000017     "defaultWindowStyle": ä
000018       "width": 1028,
000019       "height": 768,
000020       "x": 20,
000021       "y": 20
```

How it looks

- Here‘s the contents of the assets directory where our SEG logo lives:

```
Command ===>

Pathname . : /Z23A/usr/lpp/seg/vnext/wlx/0.0.1/web/assets
EUID . . . : 3
Command  Filename        Message          Type Permission
---------------------------------------------------------
_________ .                               Dir  rwxr-x---
_________ ..                              Dir  rwxrwx--x
_________ _variables.scss                 File rw-r-----
_________ icon.png                        File rw-r-----
_________ i18n                            Dir  rwxr-x---
_________ veil-loaderrota                 File rw-r-----
*********************************************************
```

How it looks

- Here‘s how the index.html looks:

```
VIEW        /Z23A/usr/lpp/seg/vnext/wlx/0.0.1/web/index.html
Command ===>
****** ***************************************************** Top of Data ********
000001 <Üdoctype html>
000002 <html lang="en">
000003 <head>
000004   <meta charset="utf-8">
000005   <title>SQL WorkloadExpert for Db2 z/OS</title>
000006   <base href="/ZLUX/plugins/de.seg.wlx.gui/web/">  <meta name="viewport" con
000007   <link rel="icon" type="image/x-icon" href="favicon.ico">
000008 <link rel="stylesheet" href="styles.c3f4c4c5a52bb89c8ce1.css"></head>
000009 <body>
000010   <app-root></app-root>
000011 <script src="runtime-es2015.40d02d0fde87355ce70a.js" type="module"></script>
000012 </html>
****** ***************************************************** Bottom of Data ******
```

How it looks

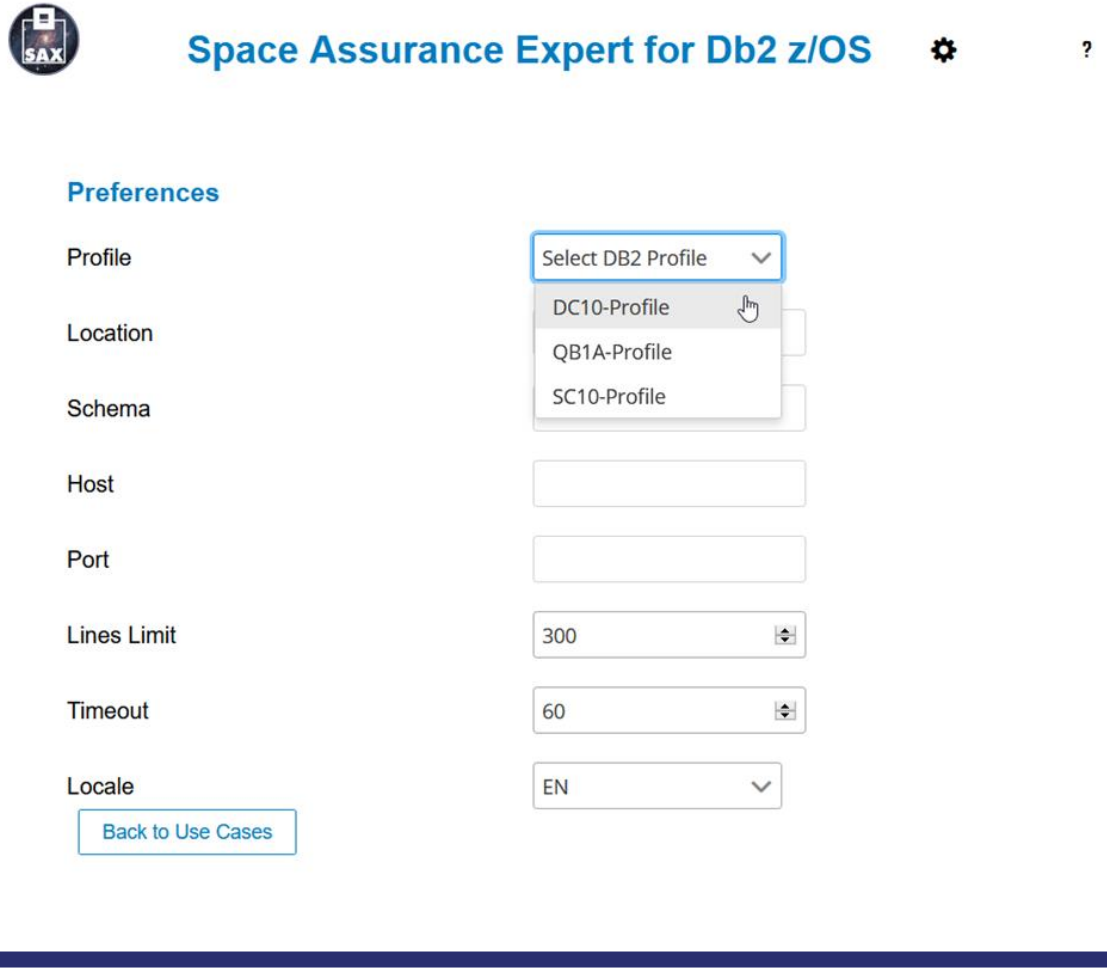
- And here's the beginning of that runtime:

```
VIEW       /Z23A/usr/lpp/seg/vnext/wlx/0.0.1/web/runtime-es2015.40d02d0fde87355ce70a.js
Command ===>
****** ********************************************************* Top of Data **********
000001 Üfunction(e)äfunction r(r)äfor(var n,c,u=rÝ0",i=rÝ1",f=rÝ2",d=0,p=Ý";d<u.length;
****** ********************************************************* Bottom of Data *******
```

```
  Menu  Utilities  Compilers  Help
 BROWSE     /Z23A/usr/lpp/seg/vnext/wlx/0.0.1/web/8- Line 0000000000 Col 001 080
 Command ===>                                          Scroll ===> CSR
*************************** Top of Data ***********************************
(window.webpackJsonp=window.webpackJsonp!!Ý").push(ÝÝ8",ä"+FGM":function(e,t)äe.
exports=function(e)ävar t=e.COMMENT("--","$"),a="BIGINT INT8 BIGSERIAL SERIAL8 B
IT VARYING VARBIT BOOLEAN BOOL BOX BYTEA CHARACTER CHAR VARCHAR CIDR CIRCLE DATE
 DOUBLE PRECISION FLOAT8 FLOAT INET INTEGER INT INT4 INTERVAL JSON JSONB LINE LS
EG!10 MACADDR MACADDR8 MONEY NUMERIC DEC DECIMAL PATH POINT POLYGON REAL FLOAT4
SMALLINT INT2 SMALLSERIAL!10 SERIAL2!10 SERIAL!10 SERIAL4!10 TEXT TIME ZONE TIME
```

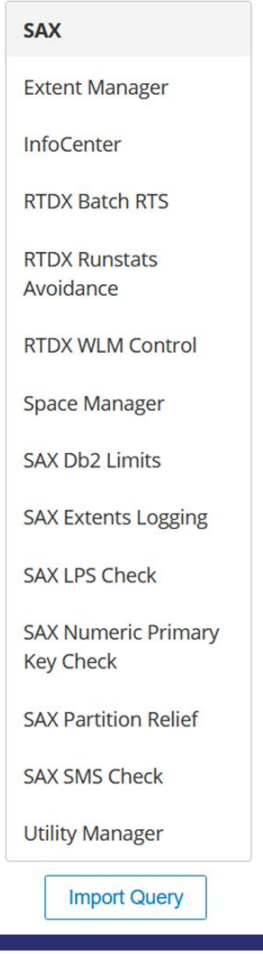
How it looks – In ZOWE

How it looks – In ZOWE



How it looks – In ZOWE

This is how the available Use Cases are displayed

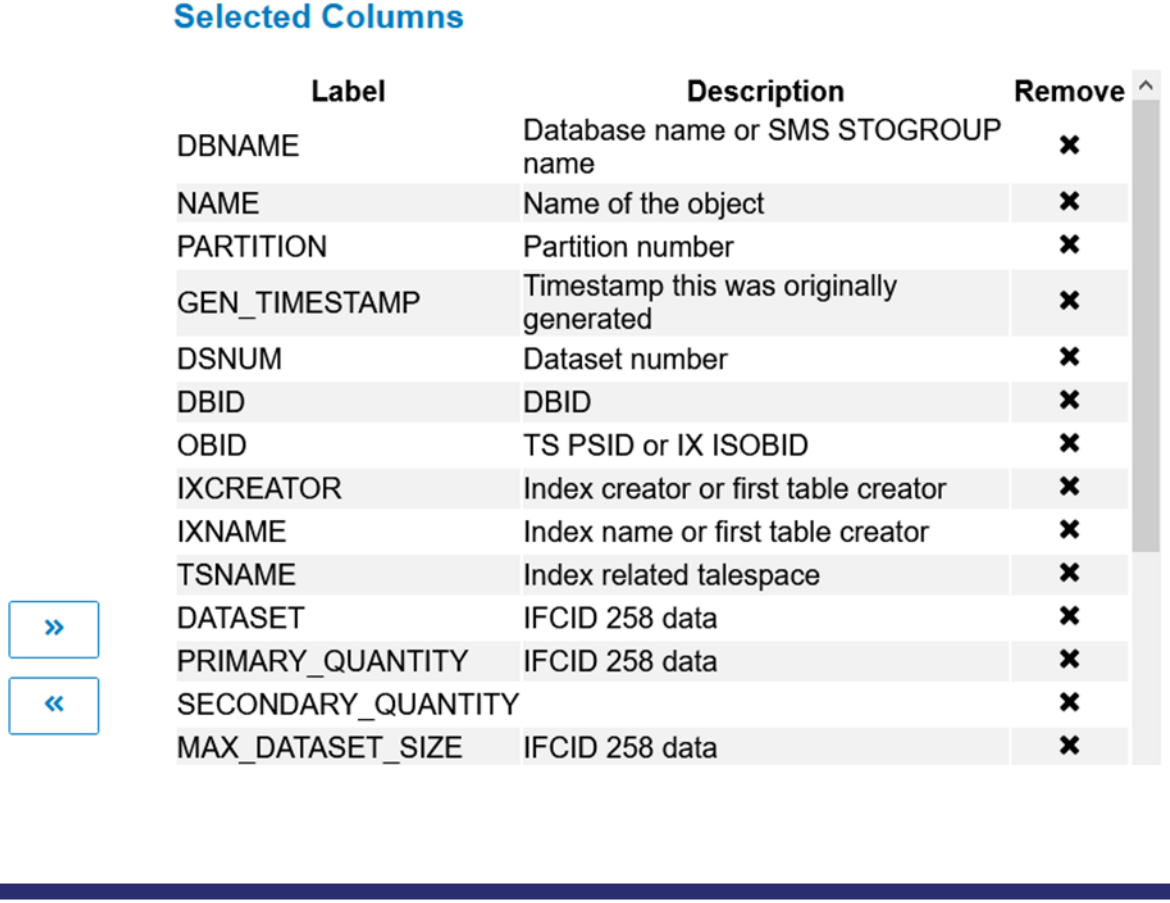


How it looks – In ZOWE

All of the Use Cases start with the same four tabs:

How it looks – In ZOWE

Each of the Use Cases has different columns that can be selected, sorted and filtered on:



How it looks – In ZOWE

The result is displayed in tabular form ready for new sorting or selecting the SQL text that was used or exporting the result to a CSV file:

Extent Manager

DBNAME	Name	Partition	Gen. Timestamp	DSNUM	DBID
DSN00420	SYSLGRNX	1	2021-03-26-08.36.18.873962	1	815
DSN00420	SYSLGRNX	1	2021-03-26-08.36.20.838877	1	815
DSN00420	SYSLGRNX	1	2021-03-26-08.36.22.237898	1	815
DSN00420	SYSLGRNX	1	2021-03-26-08.36.24.620797	1	815
DSN00420	SYSLGRNX	1	2021-03-26-08.36.27.161530	1	815
DSN00420	SYSLGRNX	1	2021-03-26-08.36.30.144645	1	815
DSN00420	SYSLGRNX	1	2021-03-26-08.36.33.911796	1	815
IQAD0610	IQASW0A6	0	2021-03-09-10.45.17.405788	1	280

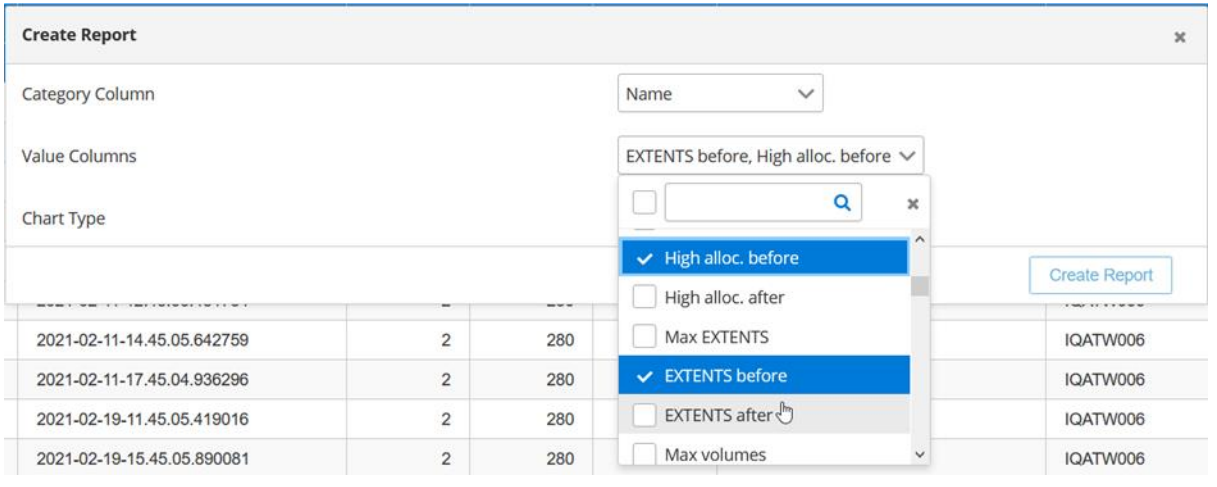
How it looks – In ZOWE

The result is displayed in tabular form ready for new sorting or selecting the SQL text that was used or exporting the result to a CSV file:

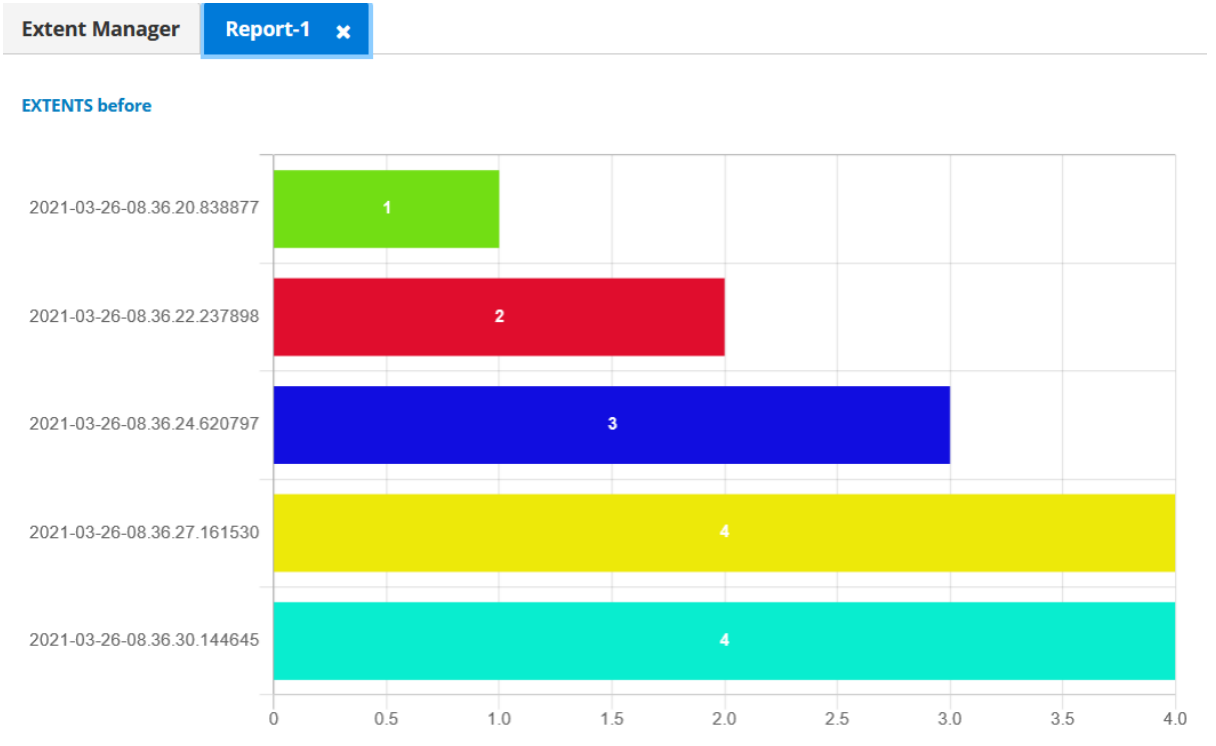
SECONDARY_QUANTITY	Max DATASET size	High alloc. before	High alloc. after	Max EXTENTS	EXTENTS before
1440	4194304	720	2160	7257	1
2160	4194304	2160	4320	7257	1
2880	4194304	4320	7200	7257	2
3600	4194304	7200	10800	7257	3
4320	4194304	10800	15120	7257	4
5040	4194304	15120	20160	7257	4
5760	4194304	20160	25920	7257	4
736	4193280	17600	19008	251	10

How it looks – In ZOWE

Naturally with HTML5 the embedded reporting is fully available:



How it looks – In ZOWE



Questions???

Many thanks for your attention and now....