z/OS 3.1 IBM Education Assistant

Solution Name: NoSQL for z/OS – VSAMDB Modern API,
SMS RAS Enhancements,
Catalog and IDCAMS Enhancements

Solution Element(s): VSAM RLS, DFSMSdfp SMS, DFSMSdfp Catalog/IDCAMS

July 2023

Agenda

- Trademarks
- For each initiative:
  - Objectives
  - Overview
  - Usage & Invocation
  - Interactions & Dependencies
  - Upgrade & Coexistence Considerations
  - Installation & Configuration
  - Summary
  - Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - None

z/OS 3.1 IBM Education Assistant

Solution Name: EzNoSQL

Agenda

- Objectives
- Overview
- Content Solution Website
    - Getting Started
    - C API Technical documentation
        - Getting Started
            - Executables and Side Deck
            - Sample Program
- JSON Documents
    - Why JSON?
- Indexes
    - Index Examples
- Recoverable vs Non-Recoverable Databases
- C APIs
- API Examples
- Java APIs
- Performance Proof Points
- Futures

Objectives

- Discuss the capability of leveraging EzNoSQL as a highly available and scalable JSON NoSQL database on z/OS

Overview

- EzNoSQL is intended for application developers, ISVs, and/or IBM products, which may require a highly available and scalable JSON NoSQL database.

- EzNoSQL is a z/OS based JSON database exploiting Parallel Sysplex data sharing capabilities and accessible via C or Java based APIs.

- EzNoSQL provides the following benefits:
  - Ability to store data in a common UTF8 JSON format
  - Ability to access the data via modern programming languages
  - Ability to share the database via a single instance eliminating common programming challenges such as eventual consistency and sharding
  - Transactional consistency via commit or rollback APIs
  - Inherits z/OS data features such as System Managed Storage, encryption, compression, etc.

EzNoSQL Overview

EzNoSQL Overview

- EzNoSQL on z/OS provides a **comprehensive set of C and Java based Application Programmer Interfaces (APIs):**
  - enables applications to access JSON (UTF-8) documents
  - while utilizing the full data sharing capabilities of IBM's Parallel Sysplex Coupling Facility (CF) technology and System z operating system (z/OS).

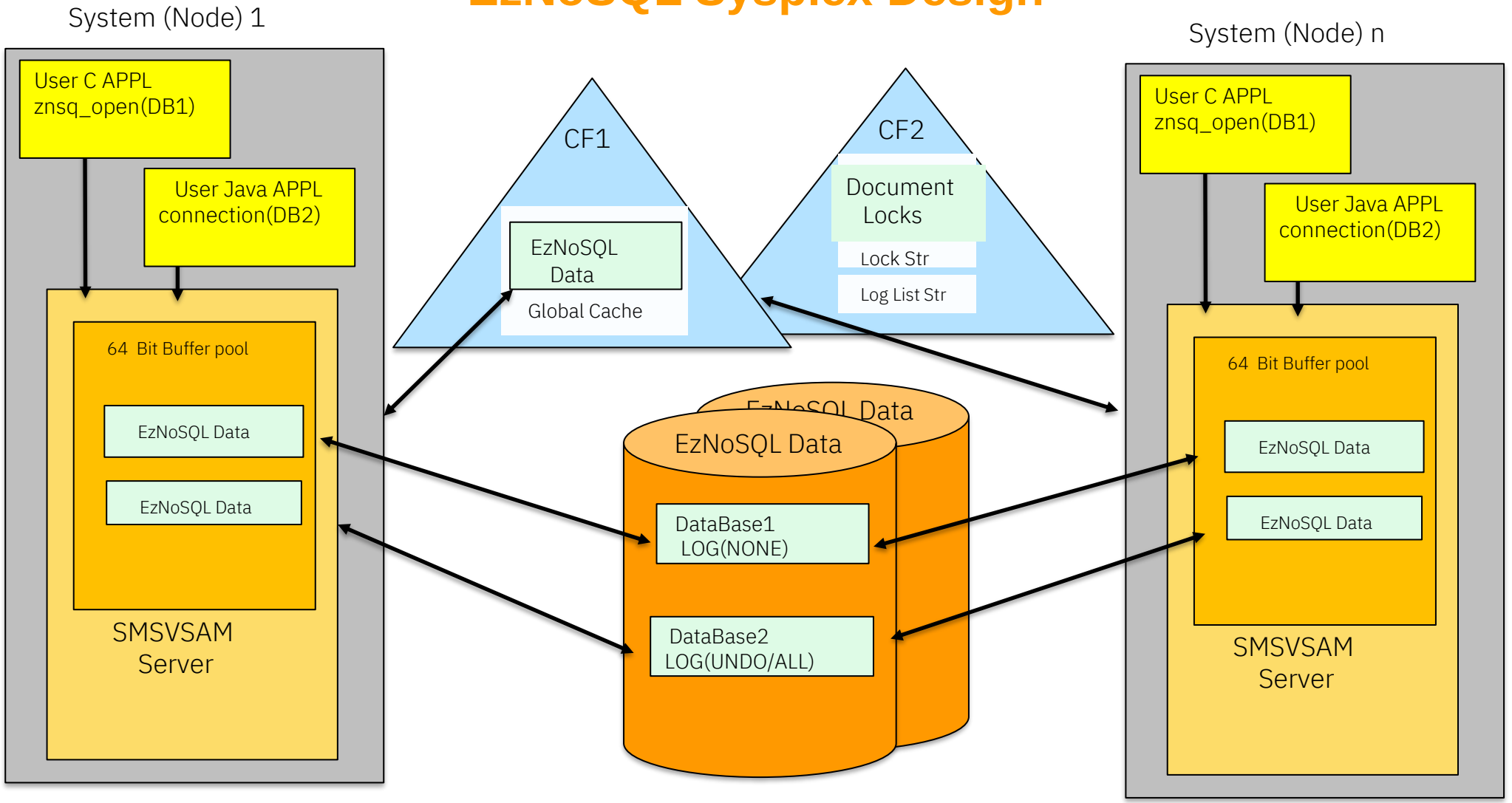
- IBM's CF technology provides shared memory between processors which
  - enables separate processors to **share a single instance of a data base** (collection of documents)
  - without the need for **data sharding, replicating** the updates, or programming for **eventual consistency**.

- Sysplex allows for easy **horizontal scalability** by adding additional processors (or z/OS instances) as required.

- Implementing EzNoSQL on z/OS will inherit many of the desired functions provided by z/OS such as **system managed storage, data encryption and compression.**

- Available on z/OS 2.4 and above with APAR **OA62553 (C APIs)** and **OA64018 (Java APIs).**

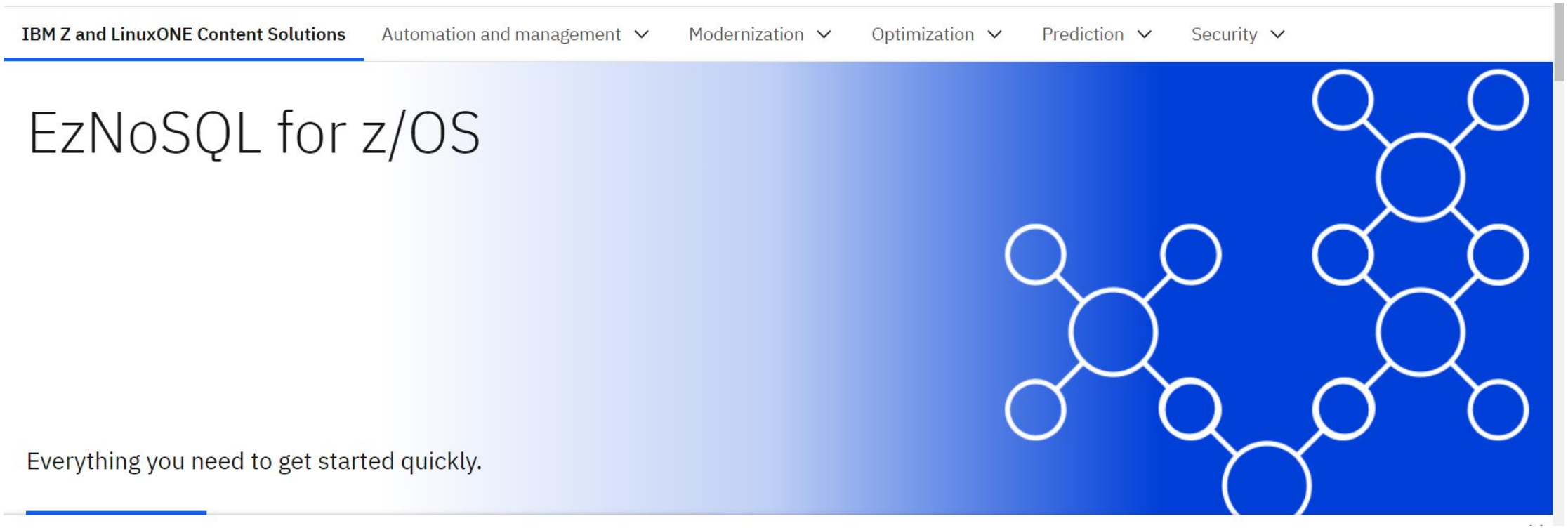
EzNoSQL Sysplex Design



Content Solution Web Site

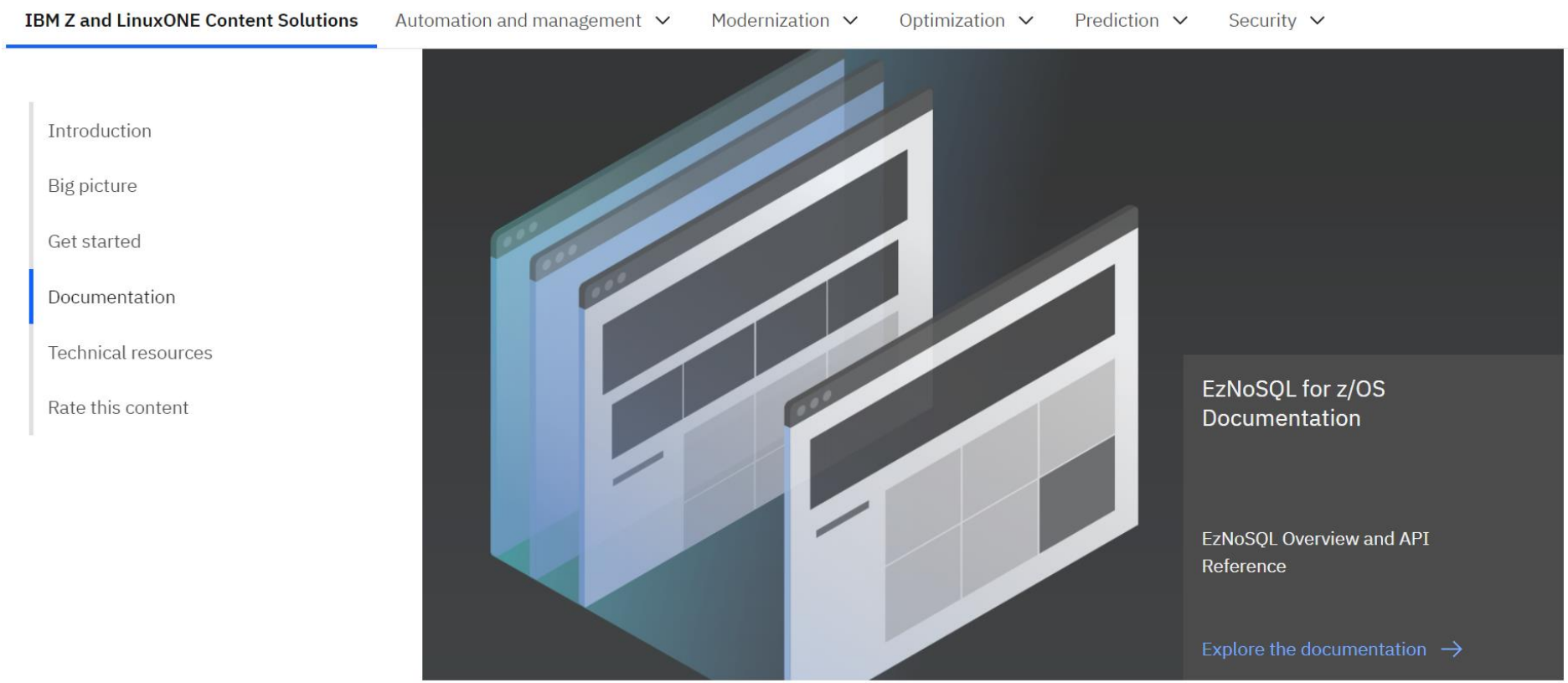
Getting Started…

- Website (will be updated when the Java APAR closes OA64018):
  https://www.ibm.com/support/z-content-solutions/eznosql



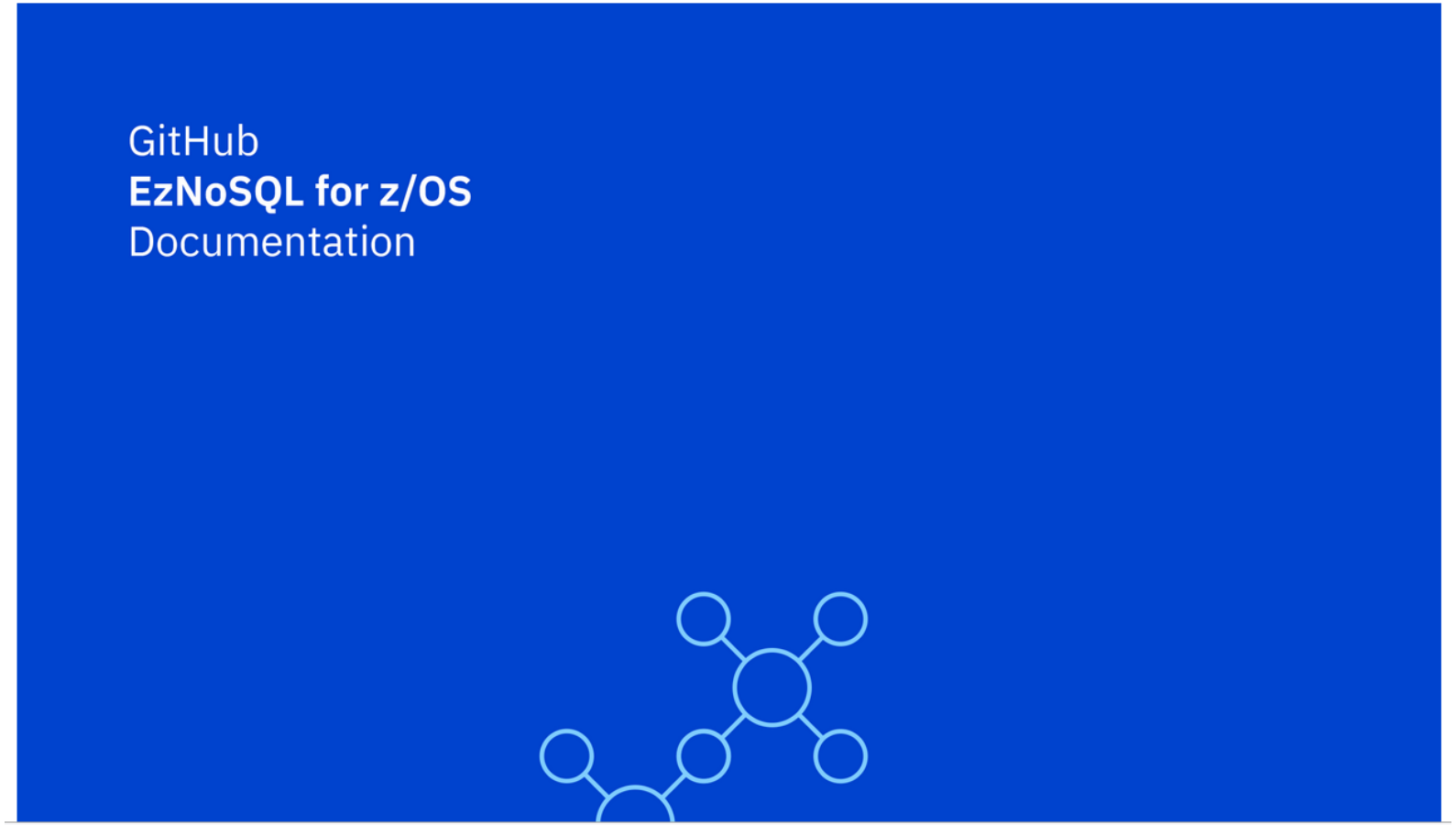
Getting Started…

- Documentation:



EzNoSQL Technical Documentation

- Documentation in GitHub:



EzNoSQL C API Technical Documentation

Table of Contents

Getting Started…

Executables and Side Decks

The following table shows the names and locations of the EzNoSQL executables, side decks, and sample program:

Member	Location	Description
`libigwznsqd31.so`	`/usr/lib/`	31-bit API Library DLL
`libigwznsqd31.x`	`/usr/lib/`	31-bit x side deck
`libigwznsqd64.so`	`/usr/lib/`	64-bit API Library DLL
`libigwznsqd64.x`	`/usr/lib/`	64-bit APIs
`igwznsqdb.h`	`/usr/include/zos/`	EzNoSQL Header File
`igwznsqsamp1.c`	`/samples/`	Sample 31-bit application program

*** There will be .so files and a similar sample program for the Java APIs

Getting Started...

Compile and Link the Sample Program:

Compile and Link Procedure

To compile and link the sample program `/samples/ibm/igwznsqsamp1.c`:

```
xlc -c -qDLL -qcpluscmt -qLSEARCH="//'SYS1.SCUNHF'" igwznsqsamp1.c
xlc -o igwznsqsamp1 igwznsqsamp1.o -W l,DLL /usr/lib/libigwznsqd31.x
```

Example C Program:

Executable example located in USS: /samples/igwznsqsamp1.c or /samples/IBM/igwvrivp

(Minor customization required (HLQ and STORCLAS) documented in prolog)

```
//
// Program Name: igwznsqsamp1.c
//
// Function: Verify the EzNoSQL API's provided in z/OS 2.4 by:
//           Use the API's to create a JSON database and index,
//           write, position and read three JSON documents using an
//           alternate key, close and destroy the JSON database.
//
//           Note: Before running this program, be sure the PTF for
//                 the EzNoSQL enabling APAR OA62553 has been
//                 installed on your system.
//
// To compile and link this sample program:
// xlc -c -qDLL -qcpluscmt -qLSEARCH="//'SYS1.SCUNHF'" igwznsqsamp1.c
// xlc -o igwznsqsamp1 igwznsqsamp1.o -W l,DLL /usr/lib/libigwznsqd31.x
//
```

Example C Program:

```
# igwznsqsamp1
Database HL1.NOSQLDB created
Index HL1.NOSQLDB.AIX created
Database opened
Index for "Author" added
Document 1 written
Document 2 written
Document 3 written
Position to first document
Return code received from next_result was: X'34'
Verify first document
Return code received from next_result was: X'34'
Verify second document
Return code received from next_result was: X'0'
Verify third document
Database closed
Database destroyed
#
```

JSON Documents

JSON (UTF-8) Documents:

Document:
{ element, element,... }

- One document (object) = one VSAM record
- Elements can vary in location, number, contents, and size.

Where an element is:
"key" : value

- Keys are character strings,
  Values can be strings, numbers, arrays,
  etc. JSON supports 5 types.

JSON Example:
{ "_id" : "00000001" ,
"Name" : "John Smith" }

JSON UTF-8 representation:
7B225F6964223A22303030303030303122 2C224E616D6522 3A224A6F686E20536D69746822 7D
{ " _ i d " : " 0 0 0 0 0 0 0 1 " , " N a m e " : " J o h n S m i t h " }

For compete specification on JSON:
www.json.org

Why JSON?

NoSQL (e.g. unstructured) key:value data bases have the following characteristics:

- The data (objects) may have widely varying formats:
  - {”Name” : “John Smith”, “Address” : “24 First St”, “Email” : “JohnSmith@gmail.com”}
  - {“Address” : “1 North St”, “Name” : “Joe Jones” , “Gender” : “M” , “Married” : “Y”}
- Data format not directly related to the file definition. Allows for rapid application changes.
- No need for a DBA to manage the data base.

VSAM data bases (e.g. “semi-structured”) have the following characteristics:

- The data (records) are partly structured and partly varying:
  - #1256789 John Smith (flags) 24 First St JohnSmith@gmail.com
  - #3763521 Joe Jones (flags) M Y
- Data format must have **primary** and alternate keys in the same location and length.
- Changes to the application related to the key’s location or length require the data base to be redefined.

EzNoSQL Indexes

EzNoSQL Indexes

- Indexes:
  - All key names must be <256 characters
  - Indexes can be read forward or backward
  - Primary Indexes:
    - User supplied key values must be unique
    - If a key name is not supplied on the create, a primary key of “znsq_id” and unique key values will be auto-generated and pre-appended to the user document
    - No length restriction for the key value (unordered index)
    - Sequential inserts not allowed (unordered index)
    - Sequential reads will not maintain order (unordered index)
  - Secondary Indexes:
    - Optionally added by the application
    - Dynamically enabled/disabled however physically created only when the database is closed.
    - Key values maybe unique or nonunique, 4 gig duplicate keys allowed.
    - No length restriction, however, only the first 251 bytes are used as the value, otherwise treated as a non-unique key value
    - Sequential inserts allowed
    - Sequential reads will maintain order
  - Multikeys:
    - Can be used as a primary or secondary key
    - Multiple key names are concatenated via a reverse solidus character: \
    - Allows key values to be used within imbedded documents or arrays

EzNoSQL Index Examples

```
{
  "Customer_id":"4084",
  "Address": { "Street":"1 Main Street", "City":"New York", "State":"NY" }
  "Accounts": ["Checking","Savings"]
 }
```

- Unique primary key **“Customer_id”**:
  - Key Value: “4084”
  - “4084” is encrypted and randomized into an internal “derived key”
  - Document can be retrieved by with the argument “Customer_id” and “4084”
- Secondary key **“Address”:**
  - Key value: { "Street":"1 Main Street", "City":"New York", "State":"NY" }
  - Document can be retrieved with the argument “Address” and { "Street":"1 Main Street", "City":"New York", "State":"NY" }
- Secondary key **“Accounts”:**
  - Key values: “Checking” and “Savings”
  - Document can be retrieved by “Address” and either “Checking” or “Savings”
- Secondary multikey **“Address\Street”**:
  - Key Value: “1 Main Street”
  - Document can be retrieved with the argument “Address\Street” and “1 Main Street”

EzNoSQL Recoverable vs Non-Recoverable Databases

Recoverable vs Non-Recoverable Databases

- Determined during create with **the znsq_log_options** parameter:
  - **NONE** – represents a non-recoverable data base
  - **UNDO** or ALL* – represents a recoverable data base
- The recoverability of a database determines the duration of the locking and the transactional (atomic) capabilities:
  - Non-recoverable:
    - Exclusive document level locks (obtained for all writes) are **held only for the duration of the write** request
    - Shared locks (optionally obtained for reads) are held only for the duration of the read request
  - Recoverable:
    - **Exclusive document level locks** (obtained for all write requests) are **held for the duration of the transaction.**
    - Shared locks (optionally obtained for reads) have two options:
      - Consistent Reads (CR) are held for the duration of the read request
      - Consistent Read Extended (CRE) are held for the duration of the transaction
    - A transaction ends following a commit or backout.
    - Auto commits are issued by default after every write request. May be disabled with the znsq_set_autocomit

* CICSVR currently does not support EzNoSQL forward recoveries.

EzNoSQL C APIs

C Level APIs **

Four Elements

DB Management	Connection Management	Document Management	Document Retrieval
Create DB / Destroy DB Add Index / List Indicies	Open / Alt Open Close	Add Document Delete Document	Search Next Result Close Result Set
Primitives focused on Managing VSAM Datasets and indexes of entries	Primitives focused on Open and closing of VSAM datasets and managing indices	Primitives focused on Adding new entries to and removing from VSAM datasets	Primitives focused on Retrieving entries matching a criteria

C Level APIs **

Four Elements

DB Management	Connection Management	Document Management	Document Retrieval
Create DB / Destroy DB Add Index / List Indicies	Open / Alt Open Close	Add Document Delete Document	Search Next Result Close Result Set
znsq_create znsq_create_index znsq_add_index znsq_drop_index znsq_destroy znsq_report_stats	znsq_open znsq_close	znsq_update_result znsq_delete_result znsq_delete znsq_update znsq_write znsq_commit znsq_abort znsq_last_result znsq_set_autocomit	znsq_position znsq_read znsq_next_result znsq_close_result

EzNoSQL Java APIs

Java Level APIs **

Three Java Classes

Database

Define, Delete, Create Index, Add Index, Drop Index

Connection

Open, Close, Read, Search, Insert, Replace, Delete, Commit, Abort, Report

ResultSet

Search Next, Replace, Delete, Close Search

Java Level APIs **

Three Java Classes

Database

Define, Delete, Create Index, Add Index, Drop Index

fromDbParameters()
createIndex()
dropIndex()
addIndex()
delete()

Connection

Open, Close, Read, Search, Insert, Replace, Delete, Commit, Abort, Report

fromDataSetName()
deleteOne()
readOne()
read()
readUpdate()
readGeneric()
insert()
replace()
commit()
setAutoCommit()
abort()
close()
getStatistics()

ResultSet

Search Next, Replace, Delete, Close Search

fromSearchParmeters()
next()
delete()
update()
close()

Example: **znsq_last_result C** API – Returned by Java APIs for Exceptions

```
znsq.last.result Report  2022.042 15:48:52
API Name: znsq_create          RC: 0000000C  RS: 82030004
Diagnostic Data:
IDCAMS  SYSTEM SERVICES                                    TIME: 15:48:51        03/03/22

DEFINE CLUSTER                        -
   (NAME(MY.JSON.DATA) -
   CYLINDERS(1000000 1)      -
   RECORDSIZE(500000 500000) -
   SHAREOPTIONS(2 3)        -
   STORCLAS(SXPXXS01)        -
   DATACLAS(KSX00002)        -
   LOG(NONE)                 -
   SPANNED                   -
   DATABASE(JSON)            -
   KEYNAMEU(_id)             -
   VOLUME (XP0201)           -
   FREESPACE (50 5))         -
   DATA(NAME(MY.JSON.DATA.D) -
   CONTROLINTERVALSIZE (32768)) -
   INDEX(NAME(MY.JSON.DATA.I) -
   CONTROLINTERVALSIZE (32768))
   IGD01007I DATACLAS ACSRTN VRS 02/19/92-1 ENTERED
```

Example: **znsq_last_result** API (cont.)

```
IGD01007I DATACLAS ACS EXECUTOR TEST ROUTINE IS ENTERED
IGD01007I DATACLAS DEFAULTING TO JCL OR TSO BATCH ALLOCATE
IGD01008I THE STORCLX ACSRTN VRS 07/06/99-1 ENTERED FOR SYSTEM X
IGD01008I STORCLAS ACS EXECUTOR TEST ROUTINE IS ENTERED
IGD01008I STORCLAS DEFAULTED TO JCL OR TSO BATCH ALLOCATE
IGD01009I MGMTCLAS ACSRTN VRS 07/28/88-1 ENTERED FOR SYSTEM X
IGD01009I   MGMTCLAS BEING DEFAULTED VIA JCL
IGD01010I STORGRP ACSRTN VRS 08/06/98-1 ENTERED FOR SYSPLEX
IGD01010I STORGRP BEING SET VIA STORCLAS UNLESS MULTIPLE GRPS SET
IGD01010I STORGRP BEING SET VIA STORCLAS UNLESS MULTIPLE GRPS SET
IGD17226I THERE IS AN INSUFFICIENT NUMBER OF VOLUMES IN THE ELIGIBLE
STORAGE GROUP(S) TO SATISFY THIS REQUEST FOR DATA SET
HL1.JSONWS01.BASE.KSDS1
IGD17290I THERE WERE 2 CANDIDATE STORAGE GROUPS OF WHICH THE FIRST 2
WERE ELIGIBLE FOR VOLUME SELECTION.
THE CANDIDATE STORAGE GROUPS WERE:SXP01 SXP02
IGD17279I 1 VOLUMES WERE REJECTED BECAUSE THEY WERE NOT ONLINE
IGD17279I 1 VOLUMES WERE REJECTED BECAUSE THE UCB WAS NOT AVAILABLE
IGD17279I 2 VOLUMES WERE REJECTED BECAUSE OF INSUFF TOTAL SPACE
IGD17219I UNABLE TO CONTINUE DEFINE OF DATA SET
HL1.JSONWS01.BASE.KSDS1
IDC3003I FUNCTION TERMINATED. CONDITION CODE IS 12

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 12
```

Example: **znsq_report_stats** API


```
{"name":"HL1.JSON.KSDS1",
                "version":1,
                "documentFormat":"JSON",
                "keyname":"_id",
                "logOptions":"UNDO",
                "readIntegrity":"NRI",
                "readOnly":false,
                "writeForce":true,
                "autoCommit":false,
                "descendingKeys":false,
                "timeout":5,
                "avgDocumentSize":1000,
                "blockSize":26624,
                "avgElapseTime:150,
                "avgCPUTime:8,
                "statistics":
                 {"numberBlocksAllocated":1234,
                  "numberBlocksUsed":124,
                  "numberExtents":1,
                  "numberRecords":2,
                  "numberDeletes":5,
                  "numberInserts":10,
                  "numberUpdates":4,
                  "numberRetrieves":13},
                "numberIndices":1,
"indices":
                [{"name":"HL1.JSON.AIX.KSDS1",
                 "keyname":"Firstname",
                 "pathname":"HL1.JSON.PATH1",
                 "active":true,
                 "unique":true,
                 "descendingKeys":false,
                 "blockSize":26624,
                 "statistics":
                   {"numberBlocksAllocated":1234,
                    "numberBlocksUsed":124,
                    "numberExtents":1,
                    "numberRecords":2,
                    "numberDeletes":5,
                    "numberInserts":10,
                    "numberUpdates":4,
                    "numberRetrieves":13},
                 "numberCompoundKeys":1,
                 "compoundKeys":
                   [{"descendingKeys":T/F,name":"altkeyname"},
                    {}...]
                 }]
               }
```

Performance Proof Points

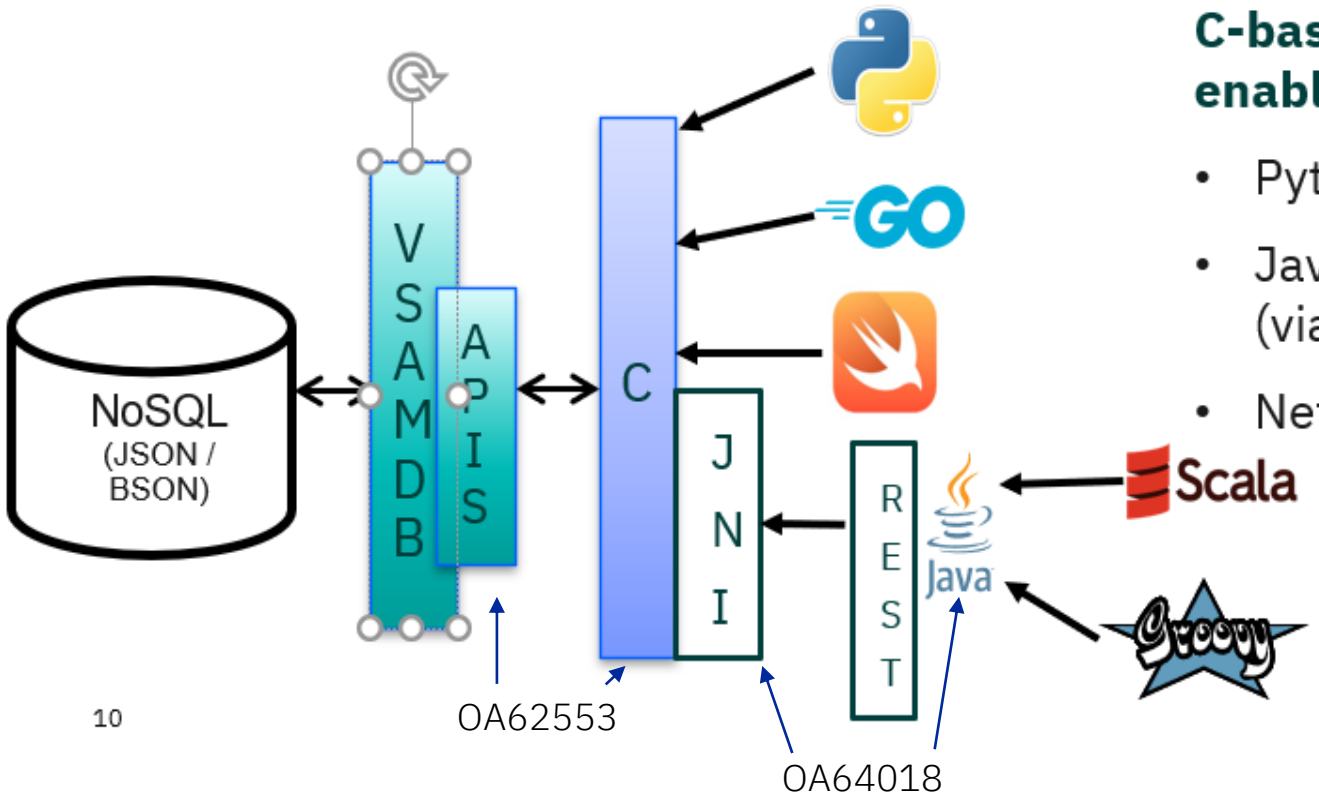
Claim:
With EzNoSQL for z/OS, process up to 294,000 read transactions per second or up to 30,000 insert transactions per second to a NoSQL database on an IBM z16 z/OS LPAR with 8 CPs.

DISCLAIMER: Performance results are based on IBM internal tests running a 64-bit application designed to only drive I/O requests executing read or insert requests to a NoSQL Database on z/OS using EzNoSQL. The measurement environment consisted of a single IBM z16 LPAR with 8 CPs running z/OS 2.4 with APAR OA63680 applied. Eight copies of the application were run concurrently accessing the same database. The database contains 10 million documents, each of which is 121 bytes long, and uses a 100 GB buffer pool. Read tests had a 100% buffer hit ratio. Your results may vary.

EzNoSQL Futures

Futures

FUTURE Options**: Data stored in a platform independent format with full data sharing capabilities



C-based key-value interface to a NoSQL database enables higher level languages and interfaces

- Python, Go, Swift (interfaces to call directly)
- Java-based languages, like Groovy and Scala (via JNI)
- Network NoSQL interfaces

Interactions & Dependencies

- Software Dependencies
  - z/OS 2.4 and above
  - APARs OA62553 (C APIs) and OA64018 (Java APIs).
- Hardware Dependencies
  - Parallel Sysplex with at least one LPAR and one (internal or external) CF
- Exploiters
  - AI Framework
  - zWLM
  - zOSMF

Summary

- EzNoSQL:
  - Ability to store and share data in a common UTF8 JSON format
  - Ability to access the data via modern programming languages (C and Java)
  - Ability to share the database via a single instance eliminating common programming challenges such as eventual consistency and sharding
  - Transactional consistency via commit or rollback APIs
  - Inherits z/OS data features such as System Managed Storage, encryption, compression, etc.

Appendix

- https://www.ibm.com/support/z-content-solutions/eznosql

z/OS 3.1 IBM Education Assistant

Solution Name: SMS RAS Enhancements – ACS Installation Exit Dynamic Activation

Solution Element(s): DFSMSdfp SMS

Objectives

- Advertise the SMS ACS installation exit dynamic activation.

Overview

- Who (Audience)
  - As a storage administrator, I wish to have the SMS ACS installation exits (IGDACSDC, IGDACSMC, and IGDACSSC) converted to use dynamic exit services, so that I can install a new version or replace an existing version without scheduling an IPL
- What (Solution)
  - Use dynamic exit services, CSVDYNEX, to invoke the SMS ACS installation exits (IGDACSDC, IGDACSMC, and IGDACSSC)
- Wow (Benefit / Value, Need Addressed)
  - Any modification and control on the SMS ACS installation exits will not require an IPL of the system

Usage & Invocation

- SMS provides three dynamic ACS installation exits
  - For Data class ACS routine: **IGDACSDC**
  - For Storage class ACS routine: **IGDACSSC**
  - For Management class ACS routine: **IGDACSMC**
- Use any of the three standard methods of adding existing routines to these exits
  - An EXIT statement in SYS1.PARMLIB(PROGxx) member
  - An EXIT parameter in SETPROG console command
  - The CSVDYNEX macro
- SMS calls all the exit routines for a given exit
  - Non-zero return code from any exit routine: The system will fail the request
  - Zero return code from all the exit routines: The system will use the (cumulative) data in the parameter areas that were provided to the exit routine(s).

Usage & Invocation (cont'd)

- Note:
  - The system presents the parameter areas to each exit routine as the areas were when the previous exit routine returned
  - Exit routines for an exit are not called in a predictable order
  - The ACS routine result can be other than what you want (or, possibly worse, inconsistent) when multiple exit routines attempt to set different combinations of ACS read-only variables and/or ACS read-write variables in the parameter areas that are passed to the exit routine(s).
  - ISVs providing exit routines for any of the ACS installation exits are responsible for ensuring that the exit routines do not conflict with the customer's system configuration policy
  - By default, the system does not disable the exits if the exit routines generate any unexpected errors.

Interactions & Dependencies

- Software Dependencies
  - The exit routine(s) for a given exit belong to storage administrators or ISVs.
- Hardware Dependencies
  - None.
- Exploiters
  - Storage administrators and ISVs.

Upgrade & Coexistence Considerations

- To exploit this solution, all systems in the Plex must be at the new z/OS level: No
- List any toleration/coexistence APARs/PTFs: None

- SMS will automatically add an ACS installation exit into the correspondent dynamic ACS installation exit
  - If SMS detects that an ACS installation exit is linked in SYS1.LPALIB as before the support
  - The exit routine name is the same with the ACS installation exit name
- The storage administrator probably will want to install the same exit routines on each system, but they do not have to do that.
- If an exit routine proves to be a problem, the storage administrator can easily remove it without an IPL or disabling the exit.

Installation & Configuration

- User exit routine(s) can be linked in any libraries such as SYS1.LPALIB, SYS1.LINKLIB, or a personal PDS.
- Use the existing SETPROG,EXIT command to add an exit routine to one of SMS ACS Installation dynamic exits.
- For example:

```
SETPROG EXIT,ADD,EXITNAME=IGDACSDC,MODNAME=MYMOD,DSNAME=MY.DSN,STATE=ACTIVE
SETPROG EXIT,ADD,EXITNAME=IGDACSSC,MODNAME=IGDACSSC

SETPROG EXIT,DELETE,EXITNAME=IGDACSDC,MODNAME=MYMOD

DISPLAY PROG,EXIT,EXITNAME=IGDACSSC,DIAG
```

Summary

- Inform that the SMS ACS installation exits (IGDACSDC, IGDACSMC, and IGDACSSC) have been converted to use dynamic exit services.

Appendix

- Publication references
  - z/OS DFSMS Installation Exits

z/OS 3.1 IBM Education Assistant

Solution Name: Catalog and IDCAMS Enhancements

Solution Element(s): DFSMSdfp Catalog/IDCAMS

Objectives

- Discuss and present:
  - 16-Byte Timestamp for SMF Record types 61, 65, and 66 and the IDCAMS RECOVER command
  - Simplify CAS Startup
  - Catalog Modify Command Enhancements
    - Active Tasks Value in the REPORT command
    - Catalog filter key for the ALLOCATED command

Overview

- Who (Audience)
  - System Programmers and Catalog Administrators
    - who need more accurate catalog forward recovery with an easier method of executing catalog forward recovery,
    - who would like to have better Catalog Address Space (CAS) startup performance,
    - and who need more information about the Catalog Address Space and the catalogs on their systems
- What (Solution)
  - Add a 16-byte time stamp to SMF type 61, 65 and 66 records and IDCAMS RECOVER command
  - Simplify CAS startup by eliminating the limited function address space
  - Enhance the MODIFY CATALOG,REPORT and ALLOCATED console commands
- Wow (Benefit / Value, Need Addressed)
  - More accurate timing for use in catalog forward recovery and an easy-to-use IDCAMS RECOVER command
  - Simpler, faster CAS startup
  - Add Active Task count to MODIFY CATALOG,REPORT and catalog name filtering to MODIFY CATALOG,ALLOCATED

Usage & Invocation 16-Byte Timestamp

- 16-Byte Timestamp for SMF record types 61 (DEFINE), 65 (DELETE) and 66 (ALTER)
  - To enable, add new parameter SMF16BYTETIMESTAMP(YES) to SYS1.PARMLIB(IGGCATxx)
    - Must do MODIFY CATALOG,RESTART to activate
    - Enablement can be checked by issuing MODIFY CATALOG,REPORT

```
*  ENABLED FEATURES          = DSNCHECK SYMREC UPDTFAIL          *
*  ENABLED FEATURES          = GDGEXTENDED                       *
*  ENABLED FEATURES          = SMF16BYTETIMESTAMP                *
*  DISABLED FEATURES         = DELFORCEWNG VVRCHECK              *
```

  - Enablement can also be checked programmatically by testing if IHADFA flag (bit) **DFACatSMF16TS** is on
  - The timestamp is added to the end of the SMF records. This means that if used for ICFRU for catalog forward recovery, the SORT commands must be changed to

```
INCLUDE COND=(6,1,BI,EQ,61,OR,6,1,BI,EQ,65,OR,'6,1,BI,EQ,66)
INREC IFTHEN=(WHEN=INIT,BUILD=(01,0004,5,258,263,32486,JFY=(SHIFT=RIGHT)))
SORT FIELDS=(0218,44,CH,A,0262,01,BI,A,32733,16,BI,D),EQUALS
OUTREC IFTHEN=(WHEN=INIT,BUILD=(01,0004,5,258,263,32486,JFY=(SHIFT=LEFT)))
OUTFIL VLTRIM=C' '
```

  - Or…

Usage & Invocation IDCAMS Recover Command

- IDCAMS RECOVER command
  - Description

Command	**Parameters**
RECOVER	(catname) STARTDATE STARTTIME STOPDATE STOPTIME GAPTIME [CLOCKDIFFERENCE]

  - Performs all 3 steps of ICFRU (CRURRSV, SORT, and CRURRAP) and chooses the best SORT method depending on the 16-byte timestamp.

```
IDC5001I RECOVER PROCESS BEGINS
IDC5002I RECOVER SMF RECORD SELECTION AND VALIDATION BEGINS
IDC5011I ALL SMF RECORDS CONTAINED THE 16-BYTE TIMESTAMP.
IDC5003I RECOVER SMF RECORD SORT BEGINS
IDC5004I RECOVER SMF RECORD ANALYSIS AND PROCESSING BEGINS
IDC5005I RECOVER PROCESS ENDED, RETURN CODE = 0
```

Usage & Invocation CAS Startup

- Simplify CAS Startup
  - Changes Catalog Address Space (CAS) start-up to use only 1 address space instead of 2.
  - CAS no longer connects to the ECS structure during limited function (LF) processing.
  - CAS no longer attaches the Modify task during LF processing.

Usage & Invocation Modify Catalog,Report

- Active Tasks Value in MODIFY CATALOG,REPORT
  - A new line displaying the number of active service tasks is added to message IEC359I which is issued in response to the MODIFY CATALOG,REPORT command
    - This value is a very dynamic number and is only accurate for a moment in time.
    - No change to the format of the {MODIFY|F} CATALOG,REPORT command

```
*CAS*****************************************************************
*  CATALOG COMPONENT LEVEL   = HDZ3310                                *
*  CATALOG ADDRESS SPACE ASN = 0019                                   *
*  SERVICE TASK UPPER LIMIT  =  180                                   *
*  SERVICE TASK LOWER LIMIT  =   60                                   *
*  HIGHEST # SERVICE TASKS   =    2                                   *
*  # ATTACHED SERVICE TASKS  =    2                                   *
*  # ACTIVE SERVICE TASKS    =    0                                   *
*  MAXIMUM # OPEN CATALOGS   = 1,024                                  *
*  ALIAS TABLE AVAILABLE     = YES                                    *
*  ALIAS LEVELS SPECIFIED    = 1                                      *
*  SYS% TO SYS1 CONVERSION   = OFF                                    *
*  CAS MOTHER TASK           = 008FB1A0                               *
*  CAS MODIFY TASK           = 008FA9E8                               *
*  CAS ANALYSIS TASK         = 008FA588                               *
```

Usage & Invocation MODIFY CATALOG,ALLOCATED

- Catalog Name Filtering on the MODIFY CATALOG,ALLOCATED command
  - Add a new NAME parameter on the MODIFY CATALOG,ALLOCATED command
  - Format: {MODIFY|F} CATALOG,ALLOCATED,NAME(ucat-filter-name)
  - Ucat-filter-key supports special filter characters: *, **, % and %%.
  - Message IEC353I CATALOG ADDRESS SPACE MODIFY UNSUCCESSFUL will include a new reason “INVALID FILTER KEY USED”
  - Message IEC348I will display “NO CATALOGS MATCH” if the filter key did not match any allocated catalogs

```
14.41.23 SYSTEM1          f catalog,allocated,name(**.ucat%)
14.41.23 SYSTEM1          IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND
 ACTIVE
14.41.23 SYSTEM1          IEC348I ALLOCATED CATALOGS
*CAS*************************************************************
* FLAGS -VOLSER-USER-CATALOG NAME                              % *
* YSI-R- 1P0302 0001 TEST.SMS.UCAT2                            1 *
* YSI-R- 1P0302 0001 TEST.SMS.UCAT1                            1 *
* YSI-R- 1P0302 0001 PROD.SMS.UCAT2                            1 *
* YSI-R- 1P0302 0001 PROD.SMS.UCAT1                            1 *
*************************************************************
```

Interactions & Dependencies

- Software Dependencies
  - None
- Hardware Dependencies
  - None
- Exploiters
  - None

Upgrade & Coexistence Considerations

- To exploit this solution, all systems in the Plex must be at the new z/OS level: No
- List any toleration/coexistence APARs/PTFs. None
- List anything that doesn't work the same anymore. Nothing
- Coexistence with down level systems. None

Installation & Configuration

- To enable, the 16-Byte SMF timestamp, add new parameter SMF16BYTETIMESTAMP(YES) to SYS1.PARMLIB(IGGCATxx)
  - Must issue MODIFY CATALOG,RESTART to activate

```
EDIT       SYS1.PARMLIB(IGGCAT00) - 01.00
Command ===>
****** ******************************** Top
000001 ALLOCLCK(10,N)
000002 SMF16BYTETIMESTAMP(YES)
000003 SYSIGGV2(10,N)
000004 SYSZTIOT(10,N)
000005 SYSZVVDS(10,N)
000006 ALIASLEVEL(1)
000007 AUTOADD(OFF)
000008 DELFORCEWNG(NO)
000009 DELRECOVWNG(NO)
000010 DSNCHECK(YES)
000011 DUMP(OFF)
000012 EXTENDEDALIAS(NO)
000013 GDGEXTENDED(YES)
000014 GDGLIMITMAX(0)
000015 GDGFIFOENABLE(NO)
000016 NOTIFYEXTENT(80)
000017 SYMREC(YES)
000018 SYS%(OFF)
000019 TAPEHLQ(SYS1)
```

Summary

- 3 new items for Catalog/AMS
  - 16-Byte Timestamp for SMF Record types 61, 65, and 66 and the IDCAMS RECOVER command
  - Simplify CAS Startup by not having the limited function catalog address space
  - Catalog Modify Command Enhancements
    - Active Tasks Value in the REPORT command
    - Catalog filter key for the ALLOCATED command

Appendix

- *DFSMS Access Method Services Commands* **SC23-6846-xx**
- *MVS System Management Facilities (SMF)* **SA38-0667-xx**
- *DFSMS Managing Catalogs* **SC23-6853-xx**
- *MVS System Messages Volume 6 (GOS - IEA)* **SA38-0673-xx**
- *MVS System Messages Volume 7 (IEB-IEE)* **SA38-0674-xx**
- *DFSMSdfp Advanced Services* **SC23-6861-xx**
- *MVS Initialization and Tuning Reference* **SA23-1380-xx**