z/OS 3.2 IBM Education Assistant

Solution Name: z/OS Parmlib Syntax Validation REST APIs

Solution Element(s): zOSMF

July 2025

Agenda

- Trademarks
- 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

Overview

- Who (Audience)
  - As a system programmer, I want to validate parmlib changes so that system IPL won’t encounter any issues.
- What (Solution)
  - z/OSMF Parmlib Management plugin that provides REST API for parmlib validation
- Wow (Benefit / Value, Need Addressed)
  - System programmer can confidently validate most commonly used parmlib members using consistent validation approach and common programming interface

New z/OSMF plugin for Parmlib Management (1)

- Background – As-Is

Parmlib Management today requires deep learning curve and repeated manual effort, sometimes could be error-prone

- Parmlib management actions <u>require unique skills</u>
  - One Parmlib member may have several commands for query and change actions
  - Different Parmlib members have different commands
  - There are 80+ parmlib members and all of them have different syntax

- Parmlib management is very flexible which could cause <u>efficiency issue and error-prone</u>
  - IPL settings VS. Dynamic settings
  - Suffix concatenation
  - Inconsistent concatenation rules across different Parmlib members
  - Use of symbols

Syntax format of IKJTSOxx

```
ALLOCATE DEFAULT{(OLD)} | {(SHR)}
AUTHCMD NAMES(cmd1,cmd2...)
AUTHPGM NAMES(pgm1,pgm2...)
AUTHTSF NAMES(name1,name2...)
NOTBKGND NAMES(cmd1,cmd2...)
HELP language(dsname1[,dsname2,...])[,language(dsname1[,dsname2,...])]
LOGON
  LOGONHERE(ON|OFF)
  PASSWORDPREPROMPT(ON|OFF)
  PASSPHRASE(ON|OFF)
  TIMEOUT(n)
  USERIDMAX{(7)} | {(8)}
  VERIFYAPPL(ON|OFF)
CONSOLE
  INITUNUM(nnnn)
  INITSNUM(nnnn)
  MAXUNUM(nnnnn)
  MAXSNUM(nnnnn)
PLATCMD{NAMES(cmd1,cmd2...)} | {NONE}
PLATPGM{NAMES(pgm1,pgm2...)} | {NONE }
TEST TSOCMD(cmd1,cmd2,cmd3.....)
     SUBCMD((scmd1,load1),(scmd2,load2)...)
TRANSREC
  NODESMF{((nodename1,smfid1),(nodename2,smfid2),...)}
          {((*,*))
  SPOOLCL(spoolclass)
  CIPHER{(ALWAYS) | (YES) | (NO)}
  OUTWARN(n1,n2)
  OUTLIM(n1)
  VIO(unitname)
  LOGSEL(logselector)
  LOGNAME(lognamesuffix)
  USRCTL(name)
  SYSCTL(datasetname)
  SYSOUT(sysoutclass | *)
  DAPREFIX(TUPREFIX | USERID)
SEND
  OPERSEND(ON| OFF),
  USERSEND(ON | OFF),
```

Syntax format of BPXPRMxx

```
{IPCSHMSPAGES(nnnnn)}
{FORKCOPY(COPY)}
{SUPERUSER(user_name)}
{TTYGROUP(group_name)}
{CTRACE(parmlib_member_name)}
{STEPLIBLIST('/etc/steplib')}
{USERIDALIASTABLE('/etc/tablename')}
{SERV _LPALIB('dsname', 'volser')}
{SERV _LINKLIB('dsname', 'volser')}
{FILESYSTYPE TYPE(type_name)
        ENTRYPOINT(entry_name)
        PARM('parm')}
        ASNAME(proc_name,'start_parms')
{SYSPLEX(YES|NO)}
{VERSION('nnnn'[,UNMOUNT|NOUNMOUNT])}

{ROOT  FILESYSTEM('fsname') or DDNAME(ddname)
        TYPE(type_name)
        MODE(access)
        PARM('parameter')
        SETUID|NOSETUID
        SYSNAME(sysname)
        TAG(NOTEXT|TEXT,ccsid)
        AUTOMOVE|NOAUTOMOVE
        MKDIR('pathname')}

{MOUNT FILESYSTEM('fsname') or DDNAME(ddname)
        TYPE(type_name)
        MOUNTPOINT('pathname')
        MODE(access)
        PARM('parameter')
        SETUID|NOSETUID
        SECURITY|NOSECURITY
        SYSNAME(sysname)
        TAG(NOTEXT|TEXT,ccsid)
        AUTOMOVE[(INCLUDE,sysname1,sysname2,...,[sysnameN|*])]
        |[(EXCLUDE,sysname1,sysname2,...,sysnameN)]
        |NOAUTOMOVE|UNMOUNT
        MKDIR('pathname')}

{NETWORK DOMAINNAME(sockets_domain_name)
        DOMAINNUMBER(sockets_domain_number)
        MAXSOCKETS(nnnnn)
        TYPE(type_name)
        INADDRANYPORT(starting_port_number)
        INADDRANYCOUNT(number_of_ports_to_reserve)}

{SUBFILESYSTYPE NAME(transport_name)
        TYPE(type_name)
        ENTRYPOINT(entry_name)
        PARM('parameter')
        DEFAULT}
```

New z/OSMF plugin for Parmlib Management (2)

- Background – As-Is

Parmlib Management today requires deep learning curve and repeated manual effort, sometimes could be error-prone

- Parmlib management may introduce repeatable manual effort. For instance, query value of a parmlib option may require
  1. Figure out what is the active suffix concatenation
  2. Figure out what is the active parmlib data set list
  3. Find member with specific suffix from parmlib data set list
  4. Find the right option from the parmlib content
  5. Look for symbol value if any symbols are used
  6. Repeat #3 to #5 depends on how many members are concatenated.
  7. #1 to #6 may be repeated if the active suffix in step 1 is determined by other parmlib members

- No common way for verification which sometimes causes errors.
- No common programmatic way to work with Parmlib values, hence, hard to integrate with modern solutions.
- No good visualization for cross sysplex Parmlib settings

New z/OSMF plugin for Parmlib Management (3)

A new plugin is introduced (via APAR PH56207) to simplify Parmlib Management

Foundation:

The core part of Parmlib Management plugin is a common parser that can understand syntax format description of Parmlibs no matter how different the syntax format is.

Stage 1 (REST API only):

User can validate syntax of most z/OS Parmlibs* with one single REST API without need to understand syntax format of Parmlib and deal with various flexibility at all.

Use cases include:

1. Validate syntax of a specific member for a Parmlib type.
    - Content of Parmlib member can be attached directly in the request body or specified via data set & member parameters
2. Validate active members of a specific Parmlib based on specified LOADxx.
3. Validate active members of all supported Parmlib based on specified LOADxx.

* 38 z/OS Parmlibs are supported in the first stage

New z/OSMF plugin for Parmlib Management (4)

Use case 1 – Validate syntax of a single specified member

https://zosmfHost:zosmfPort/zosmf/parmlib/v1/ParmlibType/validate?member=<membername>&dataset=<datasetname>

Example1: Validate SYS1.PARMLIB.POK(BPXPRMA4)

Request URL: PUT https://zosmfHost/zosmf/parmlib/v1/BPXPRM/validate?member=BPXPRMA4&dataset=SYS1.PARMLIB.POK

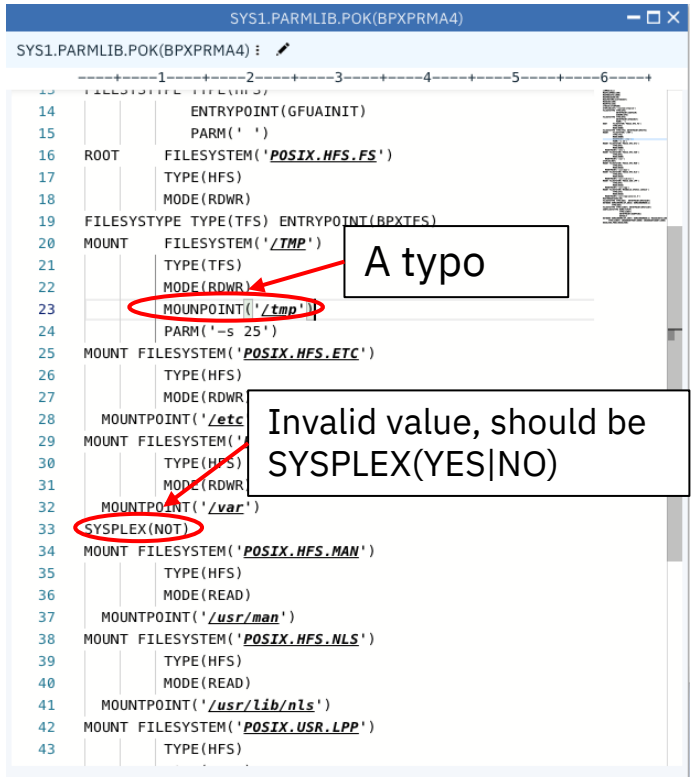
Response:

Case 1: validation succeeds

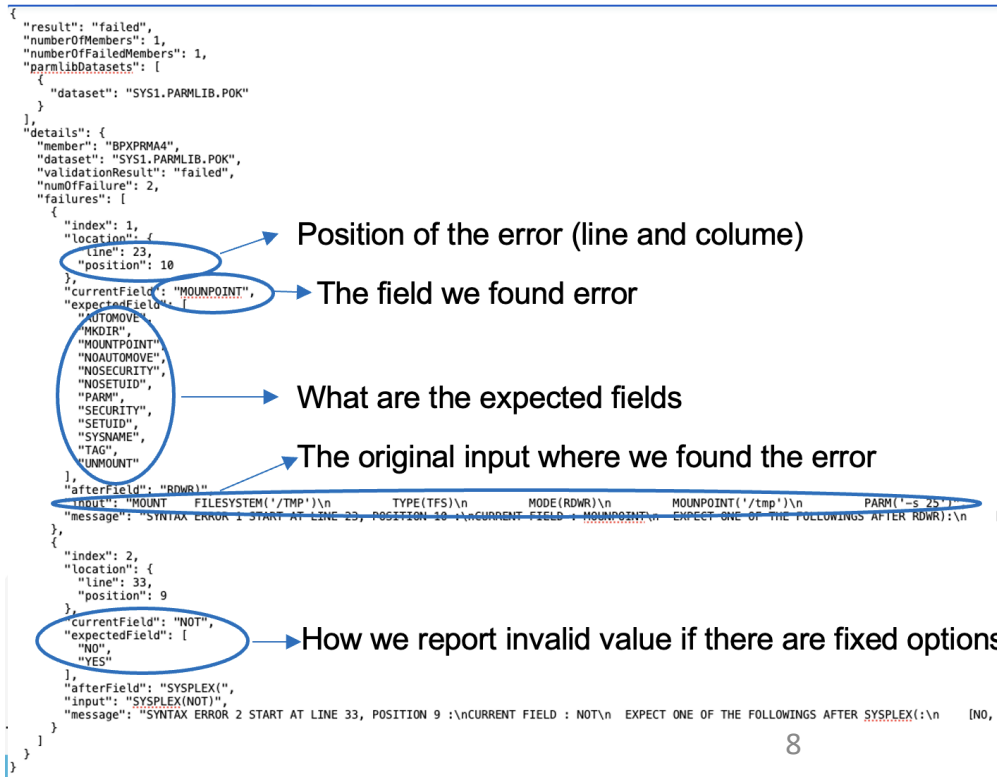

```
{
  "result": "success",
  "numberOfMembers": 1,
  "numberOfFailedMembers": 0,
  "parmlibDatasets": [
    {
      "dataset": "SYS1.PARMLIB.POK"
    }
  ],
  "details": {
    "member": "BPXPRMA4",
    "dataset": "SYS1.PARMLIB.POK",
    "validationResult": "success"
  }
}
```


Case 2: validation is failed

Content of BPXPRMA4



Validation result returned by REST API



New z/OSMF plugin for Parmlib Management (5)

Use case 2 – Validate implemented members of a specific Parmlib type

https://zosmfHost:zosmfPort/zosmf/parmlib/v1/ParmlibType/validate?memLOAD=<member>&dsnLOAD=<dataset>&volsLOAD=<volumes>

Example1: Validate active members of BPXPRMxx based on active LOADxx

Request URL: PUT https://zosmfHost/zosmf/parmlib/v1/BPXPRM/validate

Response:


```
{
  "result": "failed",
  "numberOfMembers": 5,
  "numberOfFailedMembers": 1,
  "parmlibDatasets": [
    {
      "volser": "CMNST1",
      "dataset": "USER.PLX6.TFIX.PARMLIB"
    },
    {
      "volser": "CMNST1",
      "dataset": "USER.V3R1.PARMLIB"
    },
    {
      "volser": "CMNST1",
      "dataset": "USER.PLX6.PARMLIB"
    },
    {
      "volser": "CMNST1",
      "dataset": "SVT.COMMON.PARMLIB"
    },
    {
      "volser": "CMNSTC",
      "dataset": "SYS1.PARMLIB"
    }
  ],
  "details": {
    "BPXPRMxx": [
      {
        "member": "BPXPRMST",
        "dataset": "SVT.COMMON.PARMLIB",
        "specifiedVia": [
          "SYS0.IPLPARM(LOAD16)",
          "USER.PLX6.PARMLIB(IEASYMX6)",
          "USER.PLX6.PARMLIB(IEASYSX6)"
        ],
        "validationResult": "success"
      },
```


How many active members we found

Parmlib data sets

Active member & its Data set

Parmlib chain


```
{
  "member": "BPXPRMR1",
  "dataset": "USER.V3R1.PARMLIB",
  "specifiedVia": [
    "SYS0.IPLPARM(LOAD16)",
    "USER.PLX6.PARMLIB(IEASYMX6)",
    "USER.PLX6.PARMLIB(IEASYSX6)"
  ],
  "validationResult": "success"
},
{
  "member": "BPXPRMX6",
  "dataset": "USER.PLX6.PARMLIB",
  "specifiedVia": [
    "SYS0.IPLPARM(LOAD16)",
    "USER.PLX6.PARMLIB(IEASYMX6)",
    "USER.PLX6.PARMLIB(IEASYSX6)"
  ],
  "validationResult": "failed",
  "numOfFailure": 1,
  "failures": [
    {
      ......
    }
  ]
},
```


Result of the 2nd member

Result of the 3rd member

Number of failures found in this member

New z/OSMF plugin for Parmlib Management (6)

Use case 3 – Validate implemented members of all supported Parmlib types

https://zosmfHost:zosmfPort/zosmf/parmlib/v1/LOAD/validate?deep=true&memLOAD=<member>&dsnLOAD=<dataset>&volsLOAD=<volumes>

Example1: Validate active members of all supported Parmlib types based on active LOADxx

Request URL: PUT https://zosmfHost/zosmf/parmlib/v1/LOAD/validate?deep=true

Response:


```
JSON
  result : "failed"
  numberOfMembers : 53
  numberOfFailedMembers : 4
  parmlibDatasets
  details
    CEEPRMxx
    DEVSUPxx
      0
        member : "DEVSUP00"
        dataset : "USER.PLX6.PARMLIB"
        specifiedVia
          0 : "SYS0.IPLPARM(LOAD16)"
          1 : "USER.PLX6.PARMLIB(IEASYMX6)"
          2 : "SVT.COMMON.PARMLIB(IEASYSST)"
        validationResult : "success"
    IEAPAKxx
      0
        member : "IEAPAK00"
        dataset : "SYS1.PARMLIB"
        specifiedVia
          0 : "DEFAULT"
        validationResult : "success"
    IEASYSxx
      0
        member : "IEASYS00"
        dataset : "SVT.COMMON.PARMLIB"
        specifiedVia
          0 : "DEFAULT"
        validationResult : "success"
      1
        member : "IEASYSST"
        dataset : "SVT.COMMON.PARMLIB"
        specifiedVia
        validationResult : "success"
```


Total num of active members we validated

Parmlib chain

Default member is supported


```
COMMNDxx
IQPPRMxx
VATLSTxx
CLOCKxx
CONSOLxx
AUTORxx
AXRxx
DIAGxx
IZUPRMxx
SCHEDxx
IEASYMxx
IRRPRMxx
LOADxx
COUPLExx
IEASVCxx
IXGCNFxx
GRSCNFxx
IEFOPZxx
IEAFIXxx
IFAPRDxx
IGDSMSxx
LPALSTxx
EXITxx
IGGCATxx
IEFSSNxx
ALLOCxx
GRSRNLxx
PROGxx
CUNUNIxx
BPXPRMxx
CEAPRMxx
FXEPRMxx
IEALPAxx
IKJTSOxx
```

New z/OSMF plugin for Parmlib Management (7)

Support remote systems in the same sysplex

Just use “system” parameter to specify target system which you would like to validate syntax for.

For example:

Assume z/OSMF is running system1 and system2 is in the same sysplex with system1. To validate all active members of system2, here is the URL:

https://zosmfinSystem1/zosmf/parmlib/v1/LOAD/validate?deep=true&system=system2

Interactions & Dependencies

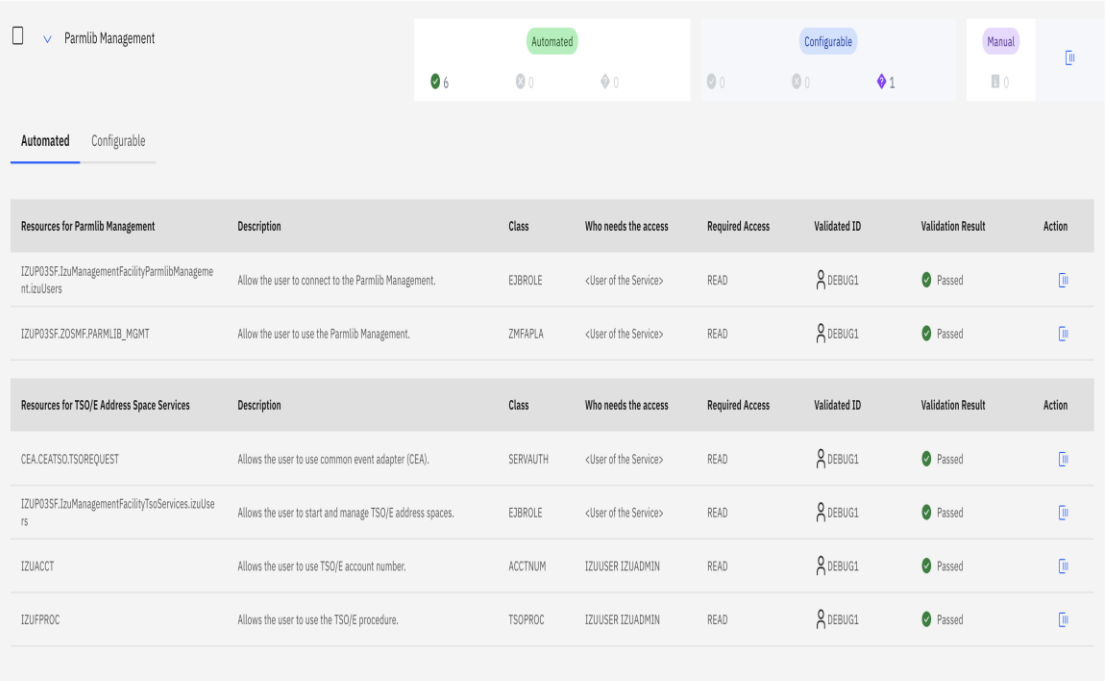
- Software Dependencies
  - No dependencies.
- Hardware Dependencies
  - None
- Exploiters
  - None

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. **N/A**
- List anything that doesn't work the same anymore. **N/A**
- Upgrade involves only those actions required to make the new system behave as the old one did.
- Coexistence applies to lower level systems which coexist (share resources) with latest z/OS systems.

Installation & Configuration

- zOSMF needs to be installed and configured
- Parmlib Management is an internal plugin which is installed as part of zOSMF
- Security setup
  - Set up zOSMF security
  - Parmlib Management requires additional SAF resources.
    - Sample in SAMPLIB(IZUPMSEC)
    - Use z/OSMF Security Assistant to validate user access
  - In addition to access to the plugin, access to the parmlib datasets is also required
- Parmlib Management is disabled by default. Enable via z/OSMF General Settings task and Restart z/OSMF



Resources for Parmlib Management	Description	Class	Who needs the access	Required Access	Validated ID	Validation Result	Action
IZUP03SF.IzuManagementFacilityParmlibManagement.izuUsers	Allow the user to connect to the Parmlib Management.	EJBROLE	<User of the Service>	READ	DEBUG1	Passed	
IZUP03SF.ZOSMF.PARMLIB_MGMT	Allow the user to use the Parmlib Management.	ZMFAPLA	<User of the Service>	READ	DEBUG1	Passed	

Resources for TSO/E Address Space Services	Description	Class	Who needs the access	Required Access	Validated ID	Validation Result	Action
CEA.CEATSO.TSOREQUEST	Allows the user to use common event adapter (CEA).	SERVAUTH	<User of the Service>	READ	DEBUG1	Passed	
IZUP03SF.IzuManagementFacilityTsoServices.izuUsers	Allows the user to start and manage TSO/E address spaces.	EJBROLE	<User of the Service>	READ	DEBUG1	Passed	
IZUACCT	Allows the user to use TSO/E account number.	ACCTNUM	IZUUSER IZUADMIN	READ	DEBUG1	Passed	
IZUFPROC	Allows the user to use the TSO/E procedure.	TSOPROC	IZUUSER IZUADMIN	READ	DEBUG1	Passed	

Summary

- New z/OSMF Parmlib Management Plugin that provides REST API to validate most commonly used parmlib member types
- Addressing complexity:
  - Learning curve of Parmlib Syntax
  - Flexibility of Parmlib data set concatenation
  - Flexibility of Parmlib member concatenation
  - Flexibility of Parmlib statement override
  - Usage of System Symbols
  - Manual effort
  - No consistent way to validate z/OS Parmlibs

Appendix

- Publications
  - zOSMF Configuration Guide
  - zOSMF Programming Guide
  - Online help within zOSMF
  - Sample Python code
    - https://github.com/IBM/IBM-Z-zOS/blob/main/zOSMF/Zosmf-Python/ParmlibAPITest.py