z/OS 3.2 IBM Education Assistant

Solution Name: SDSF

Solution Element(s): SDSF

July 2025

© 2025 IBM Corporation

Agenda

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

© 2025 IBM Corporation

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - Rocket Software

© 2025 IBM Corporation

Objectives

- Two new major functions introduced
  - pySDSF – Python interface to most common SDSF data
  - zFS profile store for SDSF settings and configuration
- New RACF log panel
- Other new SDSF panels
- Useability and other miscellaneous enhancements

© 2025 IBM Corporation

Overview

- Who (Audience)
  - System programmers, operators, administrators, general users
- What (Solution)
  - Systems management
- Wow (Benefit / Value, Need Addressed)
  - Simplify management of the system
  - Improve user experience

© 2025 IBM Corporation

Usage & Invocation

- ISPF dialog
- TSO command
- SDSF batch (no longer being enhanced, use REXX instead)
- SDSF/REXX (including sysrexx)
- SDSF/Java
- z/OSMF SDSF plug-in
- Python (New)

© 2025 IBM Corporation

Interactions & Dependencies

- Software Dependencies
  - Python 3.13 for new support
- Hardware Dependencies
  - None
- Exploiters
  - None

© 2025 IBM Corporation

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

© 2025 IBM Corporation

Installation & Configuration

- New ISFPRMxx keywords on the GROUP statement required to enabled the zFS profile store facility
  - PROFILE(FILESYS | ISPF)
  - Default path used :
    - $HOME/.isf/*userid*
- Python interface shipped in */usr/lpp/sdsf/python/dist/*

© 2025 IBM Corporation

pysdsf

Python interface to SDSF

© 2025 IBM Corporation

pysdsf – Python Interface to SDSF

- Easy to use python interface to SDSF
- Initial support for the following primary panels
  - AS, DA, H, I, O, ST
  - LOG (both SYSLOG and OPERLOG)
- Initial support for the following secondary panels
  - JDS (aka ‘?’), JS
- SYSOUT browse supported
- Operator commands also supported
  - Both generated by actions and freeform “/” style

© 2025 IBM Corporation

pysdsf – Installation And Setup

- pysdsf supplied as part of the standard SDSF SMP/E FMID
- Artifacts created via SMP/E APPLY
- Wheel file distributed in :

  /usr/lpp/sdsf/python/dist/
- Documentation distributed in :

  /usr/lpp/sdsf/python/doc

© 2025 IBM Corporation

pysdsf – Example Code

```python
from pysdsf import Connection
from pysdsf.isfexception import ISFException

if __name__ == '__main__':
    connection = Connection()
    # Get all the jobs with prefix SDSF*
    try:
        status_result = connection.status_of_jobs(prefix='SDSF*',
                                                  owner='*')
        # Any row returned?
        if len(status_result):
            print('Job_Name Job_ID     Queue')
            for row in status_result:
                print(
                    f'{row["JNAME"].ljust(8)} {row["JOBID"].ljust(8)} '
                    f'  {str(row["QUEUE"]).ljust(10)} ')
        else:
            print('Jobs with prefix SDSF* not found')
    except ISFException as e:
        print(f'Error occurred response is {e}')
    finally:
        connection.close()
```

Following slides will breakdown the statements to explain the general flow and look-and-feel of pysdsf

Multiple examples shipped with the product

© 2025 IBM Corporation

pysdsf – Breakdown Of Statements (1)

```
from pysdsf import Connection
from pysdsf.isfexception import ISFException
```

Import the pysdsf package and its allied custom exception

```
connection = Connection()
```

pysdsf uses the “connection” object to facilitate communication with the SDSF server

```
status_result = connection.status_of_jobs(prefix='SDSF*',
                                          owner='*')
```

Named SDSF functions can be passed parameters explicitly or set as a dictionary :

```
options = {
   ‘prefix’:’SDSF*’,
   ‘owner’:’*’
}

 status_result = connection.status_of_jobs(**options)
```

© 2025 IBM Corporation

pysdsf – Breakdown Of Statements (2)

```
if len(status_result):
    print('Job_Name Job_ID     Queue')
    for row in status_result:
        print(
            f'{row["JNAME"].ljust(8)} {row["JOBID"].ljust(8)} '
            f'  {str(row["QUEUE"]).ljust(10)} ')
else:
    print('Jobs with prefix SDSF* not found')
```

Process each row of the results and access column data using the internal column name

Internal column names match SDSF REXX

```
except ISFException as e:
    print(f'Error occurred response is {e}')
```

Handle any exceptions from SDSF

```
connection.close()
```

Close the connection with SDSF

© 2025 IBM Corporation

pysdsf – Browse SYSOUT For Active Job

```
from pysdsf import Connection
from pysdsf.isfexception import ISFException

if __name__ == '__main__':
    connection = Connection()
    # Get all the jobs with prefix SDSFAUX*
    options = {
        'prefix': 'SDSFAUX*',
        'owner': '*'
    }
    try:
        da_result = connection.active_users(**options)
        # Any row returned?
        if len(da_result):
            for da_row in da_result:
                try:
                    print(f'Browse result for {da_row["JNAME"]}({da_row["JOBID"]})')
                    browse_result = da_result.browse(da_row, **options)
                    if browse_result:
                        for line in browse_result:
                            print(line)
                    else:
                        print('Nothing to browse')
                except ISFException as e:
                    print(e)
        else:
            print('Jobs with prefix SDSFAUX* not found')
    except ISFException as e:
        print(f'Error occurred response is {e}')
    finally:
        connection.close()
```

connection.active_users = “DA”

Simple interface to browse SYSOUT for job.

© 2025 IBM Corporation

pysdsf – Issue Actions Against Jobname(s)

```python
def run(job_name, max_rc) -> None:
    """
    Function to perform SDSF commands and action
    @param job_name: Job name pattern
    @param max_rc: Max rc to filter on
    @return:
    """
    connection = Connection()
    try:
        status_result = connection.status_of_jobs(prefix=job_name,
                                                  cols=['JNAME', 'JOBID', 'RETCODE'],
                                                  filter=f"RETCODE  EQ 'CC {max_rc:04}'")
        # Any row returned?
        if len(status_result):
            for st_row in status_result:
                print(
                    f'Job {st_row["JNAME"]}({st_row["JOBID"]}) '
                    f'Max-RC {st_row["RETCODE"]} command issued')
                # Issue Purge action against row
                action_result = status_result.action('P', st_row)
                print('Command response below:')
                # Show any command response
                for response in action_result:
                    if response.find(' ISF0') == -1:
                        print('  ' + response)
        else:
            print(f'Job with name pattern "{job_name}" not found, no action taken')
    except ISFException:
        pass
    finally:
        connection.close()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Script to purge jobs for a given max cc')
    parser.add_argument('--jobname',
                        required=True,
                        type=str,
                        help='Job name pattern or job name')
    parser.add_argument('--maxrc',
                        required=True,
                        type=int,
                        help='Max return code to look at before purging job')
    args = parser.parse_args()
    run(args.jobname, args.maxrc)
```

Simple interface to issue action character against a row

Will result in z/OS operator command being issued and we can examine the response

© 2025 IBM Corporation

pysdsf – Scanning the SYSLOG/OPERLOG

```
import datetime
from pysdsf import Connection
from pysdsf.isfexception import ISFException

if __name__ == '__main__':
    # Connection for SDSF. Note: Does not invoke SDSF
    connection = Connection()
    # Get datetime for the time and date
    today: datetime = datetime.datetime.now()
    an_hour_ago: datetime = today - datetime.timedelta(hours=1)
    # Set the start and stop times to limit records
    # Set response line limit to limit number of line to return in response
    try:
        sys_log_result = connection.read_syslog(start_date=an_hour_ago.strftime("%m/%d/%Y"),
                                                start_time=an_hour_ago.strftime("%H:%M:%S"),
                                                stop_date=today.strftime("%m/%d/%Y"),
                                                stop_time=today.strftime("%H:%M:%S"),
                                                response_limit=100)

        # List any messages related to the read request

        for row in sys_log_result:
            print(row)
    except ISFException as e:
        print(e)
    finally:
        connection.close()
```

Simple interface to browse and process SYSLOG rows

© 2025 IBM Corporation

pysdsf – Initial Function List

• *active_users*	DA panel
• *address_space_memory*	AS panel
• *browse*	SYSOUT browse
• *held_output*	H panel
• *input_queue*	I panel
• *issue_command*	“/” facility
• *output_queue*	O panel
• *read_syslog/read_operlog*	LOG panel (SYSLOG/OPERLOG)
• *status_of_jobs*	ST panel

© 2025 IBM Corporation

SDSF Profile Data In zFS

© 2025 IBM Corporation

SDSF Profile Data in ISPF - Restrictions

- SDSF for z/OS 3.1 and prior releases use ISPF profile to store user settings, preferences including sort, arrange and filter criteria
- ISPF has a fixed limit of 64K for all data within a single xxxxPROF member of the ISPPROF data set
- SDSF users with large amounts of sort, arrange or filter criteria can exceed 64K in size causing errors when trying to VREPLACE/VPUT data into ISFPROF
- Using ISPF for profile data also means that other interfaces cannot retrieve stored user settings

© 2025 IBM Corporation

SDSF Profile Data in zFS

- SDSF for z/OS 3.2 introduces the ability to store user profile data in the z/OS UNIX file system
  - Removes the 64K data limit constraint
  - Makes profile data available to non-ISPF interfaces
- Profile data stored as JSON documents
- Positions SDSF to provide further enhancements in later releases based on the JSON document profile store

© 2025 IBM Corporation

SDSF Profile Data in zFS Implementation

- New SDSF group keyword “PROFILE”
  - ISPF Use legacy ISPF profile (default)
  - FILESYS Use z/OS UNIX file system
- Profile data stored in the “.isf” directory in user home directory
- New SDSF group keyword “PROFILEPATH”
  - Specifies any additional directory path appended after “.isf”
  - Default is null
- Invocation screen name appended to directory name if not equal to “SDSF”
  - Allows users to have different profile stores for fastpath SDSF invocations from places like the ISPF command table

© 2025 IBM Corporation

SDSF Profile Data in zFS Migration

- SDSF profile path will be created if does not exist and any existing ISPF profile settings will be automatically migrated to new JSON format
  - Users can re-migrate by deleting the “.isf” directory and re-invoking SDSF
- Once the SDSF profile path is created, any existing or updated ISPF profile settings will be ignored
  - Data will not be saved to ISPF profile members after migration
- Important to note if sharing SDSF ISFPROF members with downlevel systems
  - SDSF for z/OS 3.1 and below will have no knowledge of profile data in the profile path filesystem
  - Some customers might postpone using FILESYS profiles until all members in sysplex are at least z/OS 3.2

© 2025 IBM Corporation

SDSF Profile Data Considerations

- SDSF users should not edit the JSON documents in the “.isf” profile directory
- Any malformed or invalid JSON data encountered during initialization will cause SDSF to discard all profile path information and fall back to using default settings

© 2025 IBM Corporation

RACF Log

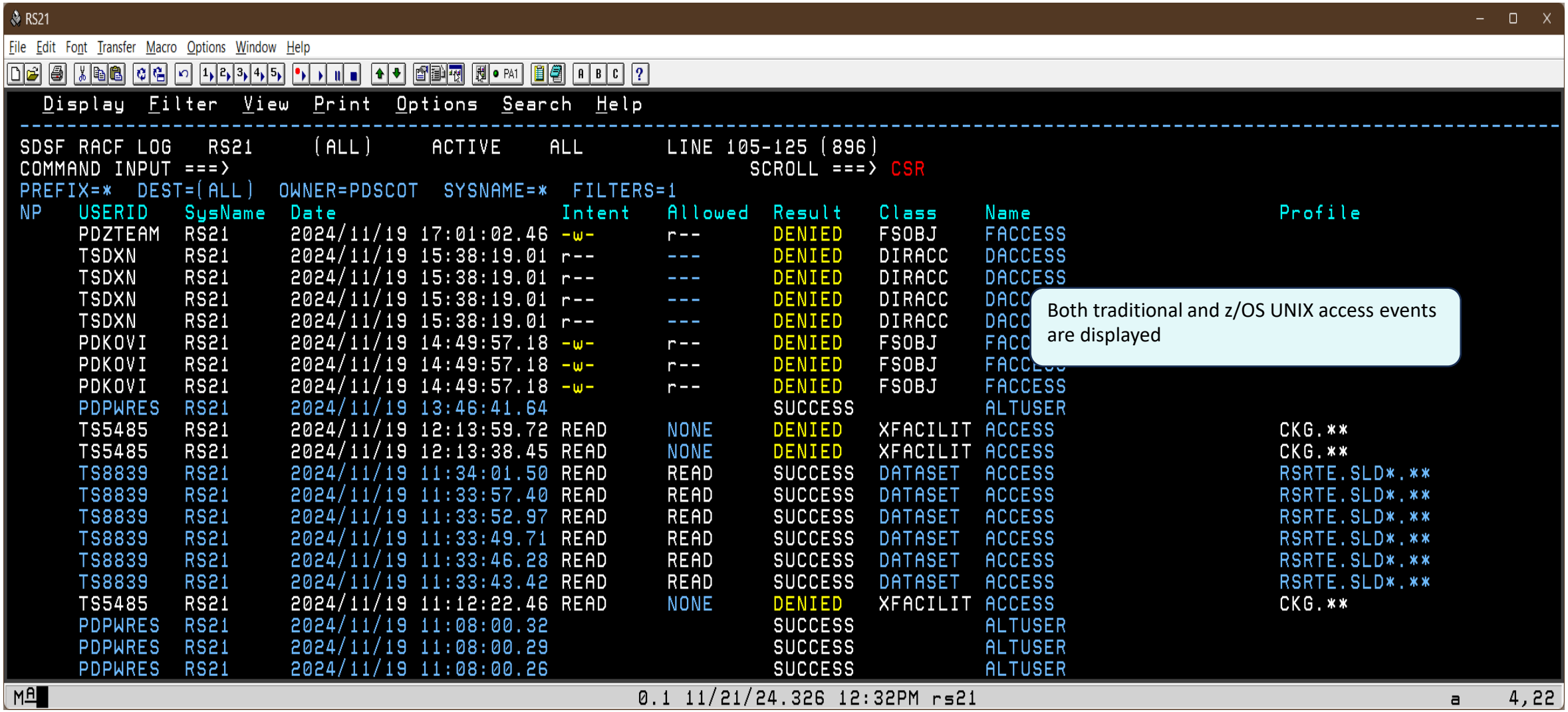
© 2025 IBM Corporation

RACF Log Panel (1)

- Provides a summary of recent logged RACF activity
  - Includes failures and logged access for traditional resources as well as certain z/OS UNIX information
- Uses the SDSF event log architecture to provide the data
  - SMF record type 80 contents used to populate panel entries
- Drill down to the OPERLOG from row on RLOG panel using “L” action
  - Saves hunting OPERLOG for ICH408I messages
- Direct access to profile information and access list using “LP” and “LA” actions
- RLOG panel can be filtered using FEATENT statements in similar fashion to ELOG
  - New categories of “SAF”, “SAF_CMD” and “SAF_UNIX”

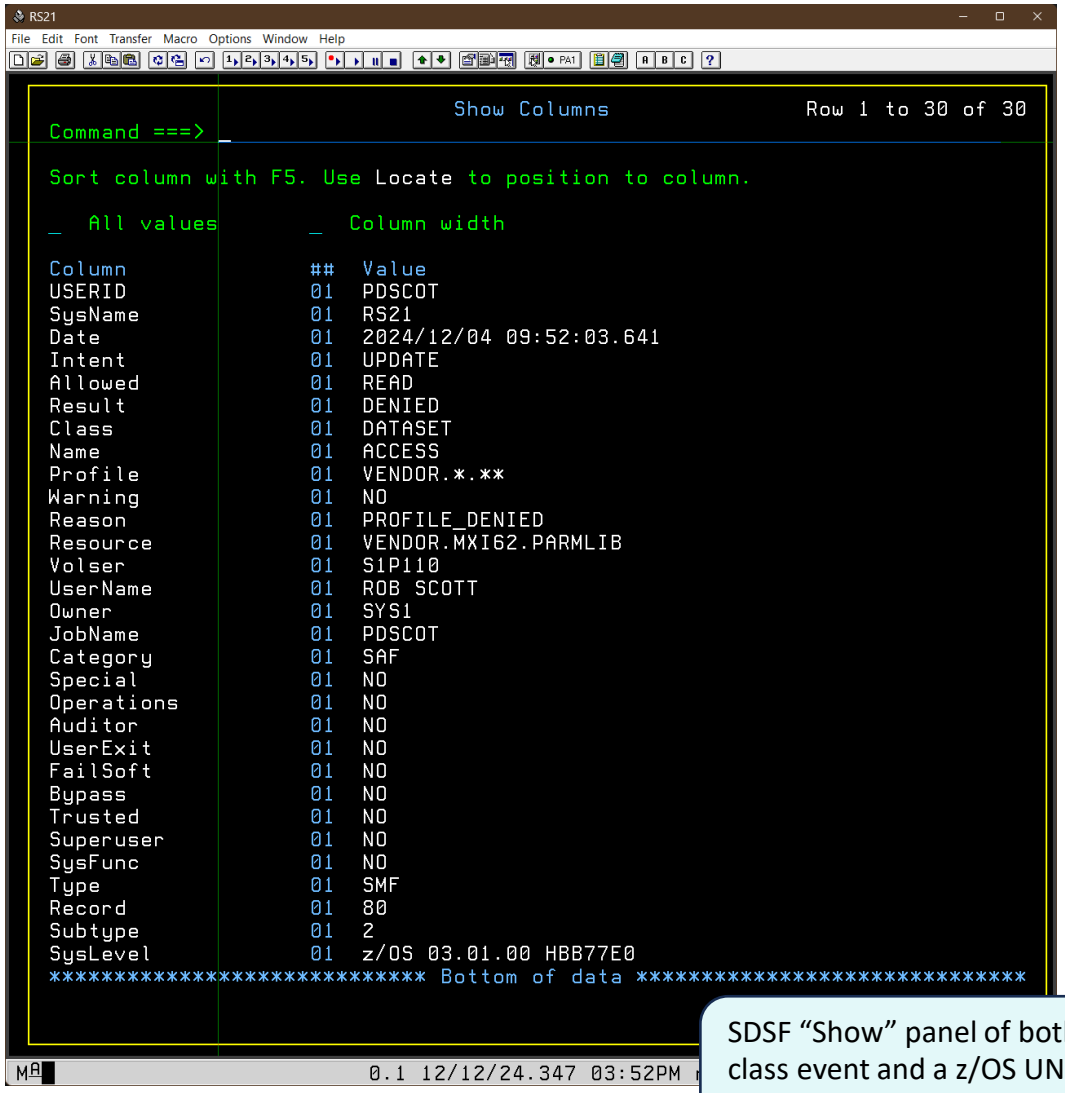
© 2025 IBM Corporation

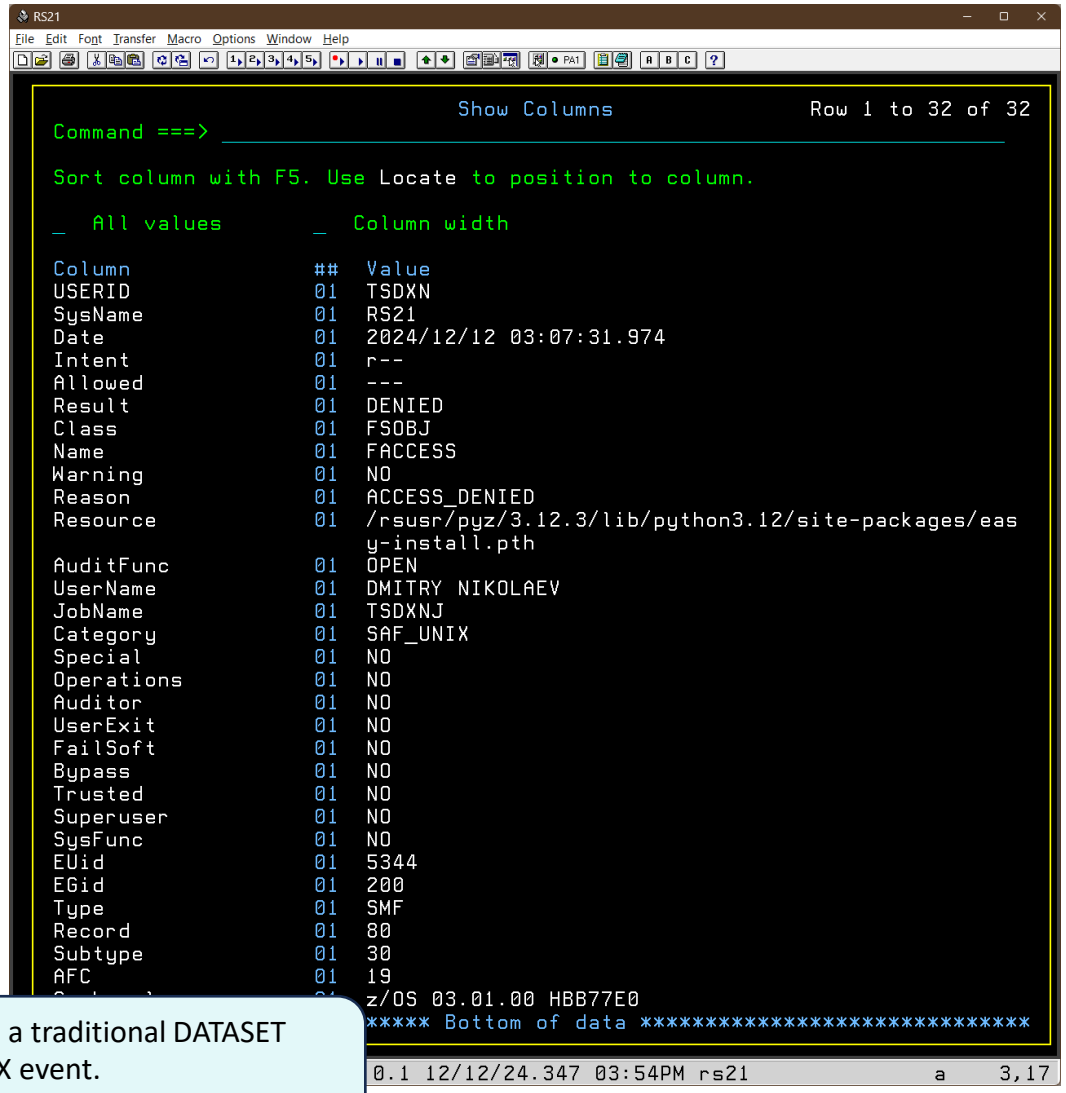
RACF Log Panel (2)



© 2025 IBM Corporation

RACF Log Panel - Detail





SDSF “Show” panel of both a traditional DATASET class event and a z/OS UNIX event.

© 2025 IBM Corporation

RACF Log Panel - Columns

- Columns include
  - Class, resource and profile
  - Jobname, User name, application name and owner
  - Access intent and allowed
  - Warning, special, operations, auditor, failsoft, bypass, trusted attributes in effect
  - Any z/OS UNIX EUID and EGID
  - Any log string data
  - Any SECLABEL
  - Names for any attempted RACF commands

© 2025 IBM Corporation

RACF Log Panel - Actions

- L
  - Invoke LOG (OPERLOG) panel and position to the date-stamp of the logged event
- LA
  - Show the RACF access list for the profile covering the resource in shown in the row
- LP
  - Browse the RACF profile covering the resource in shown in the row

© 2025 IBM Corporation

Other New SDSF Panels

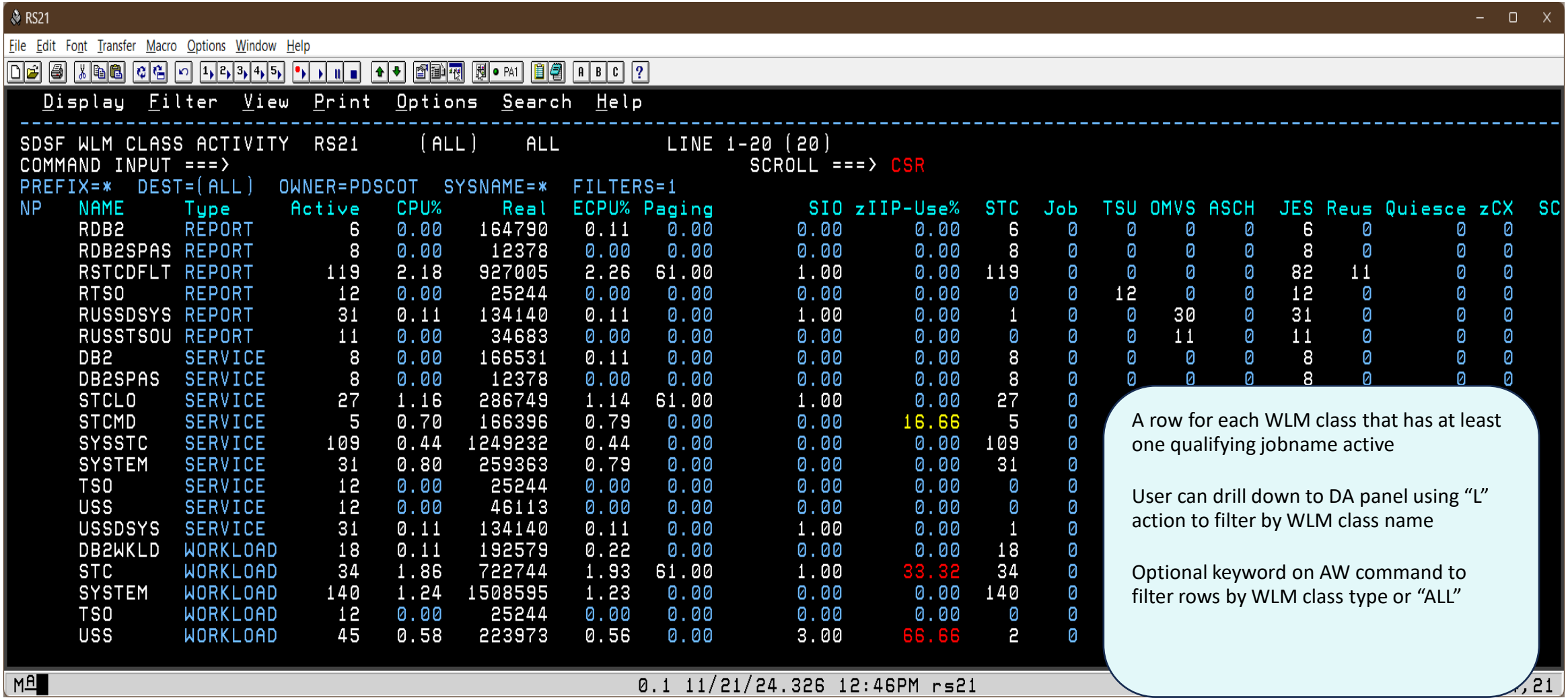
© 2025 IBM Corporation

New SDSF Panels

AW	Active Job Summary By WLM Class
CAT	Master And User Catalogs
CMO	64-bit Common Memory Objects
DEVS	Device Space
FXE	z/OS Function Registry
JRU	JES2 Resource Usage by Userid
MFP	z/OS UNIX Path Module Fetch
NAP	TCP/IP Activity Summary By Port
PST	z/OS UNIX Threads
RACD	RACF Data Sets
RACF	General RACF Information
RACR	RACF RRSF Nodes
SMFL	SMF Logstreams
SMFR	SMF Real Time Resources
VTOC	DASD Volume Table Of Contents

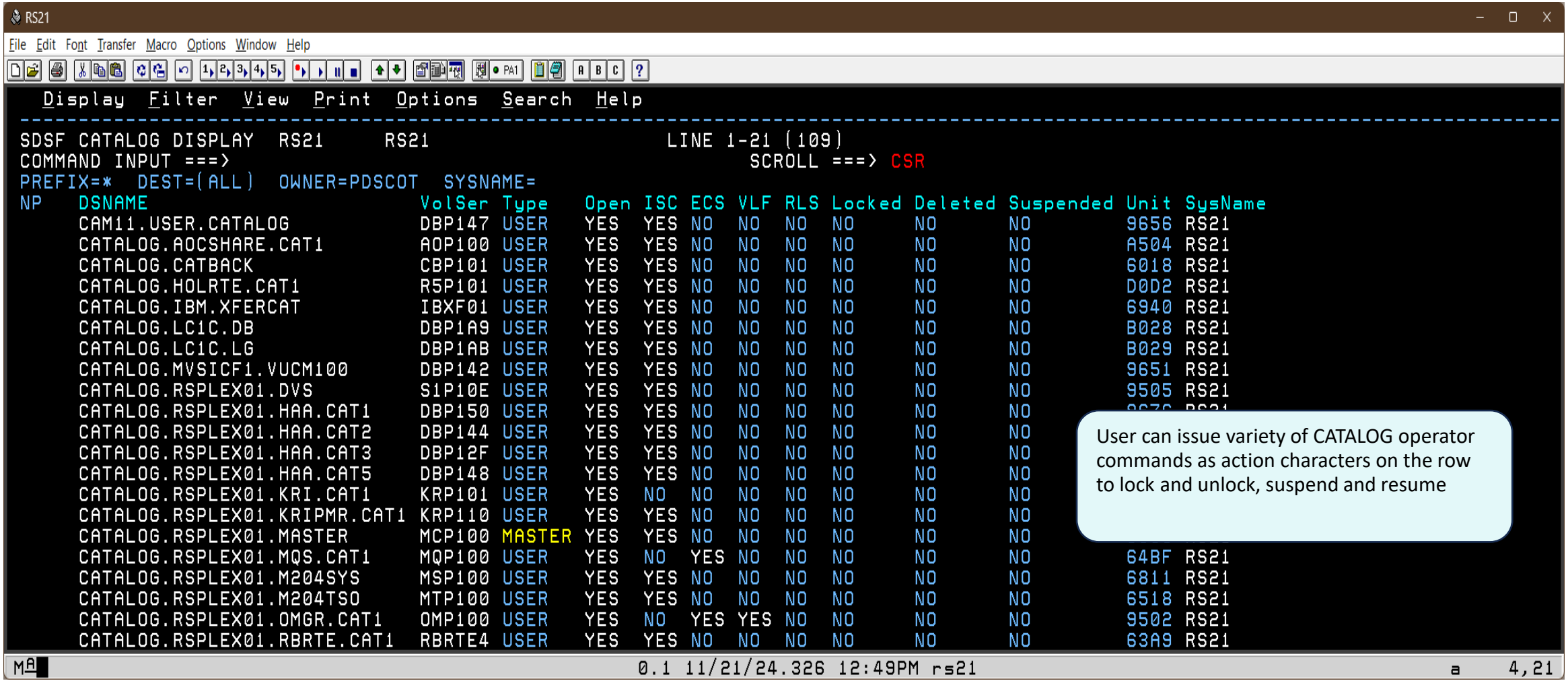
© 2025 IBM Corporation

AW Panel – Address Space Summary By WLM Class



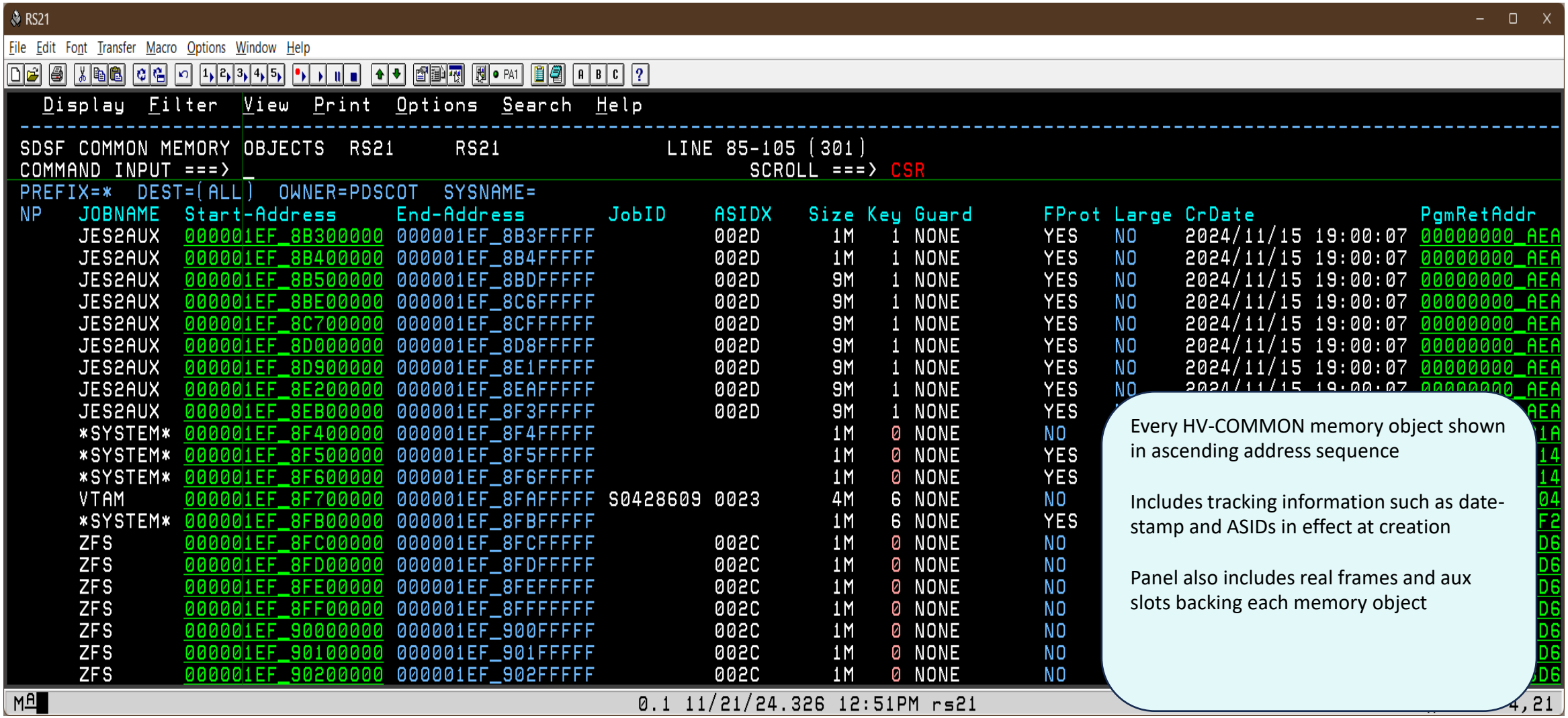
© 2025 IBM Corporation

CAT Panel – Master and User Catalogs



© 2025 IBM Corporation

CMO Panel – 64-Bit Common Memory Objects



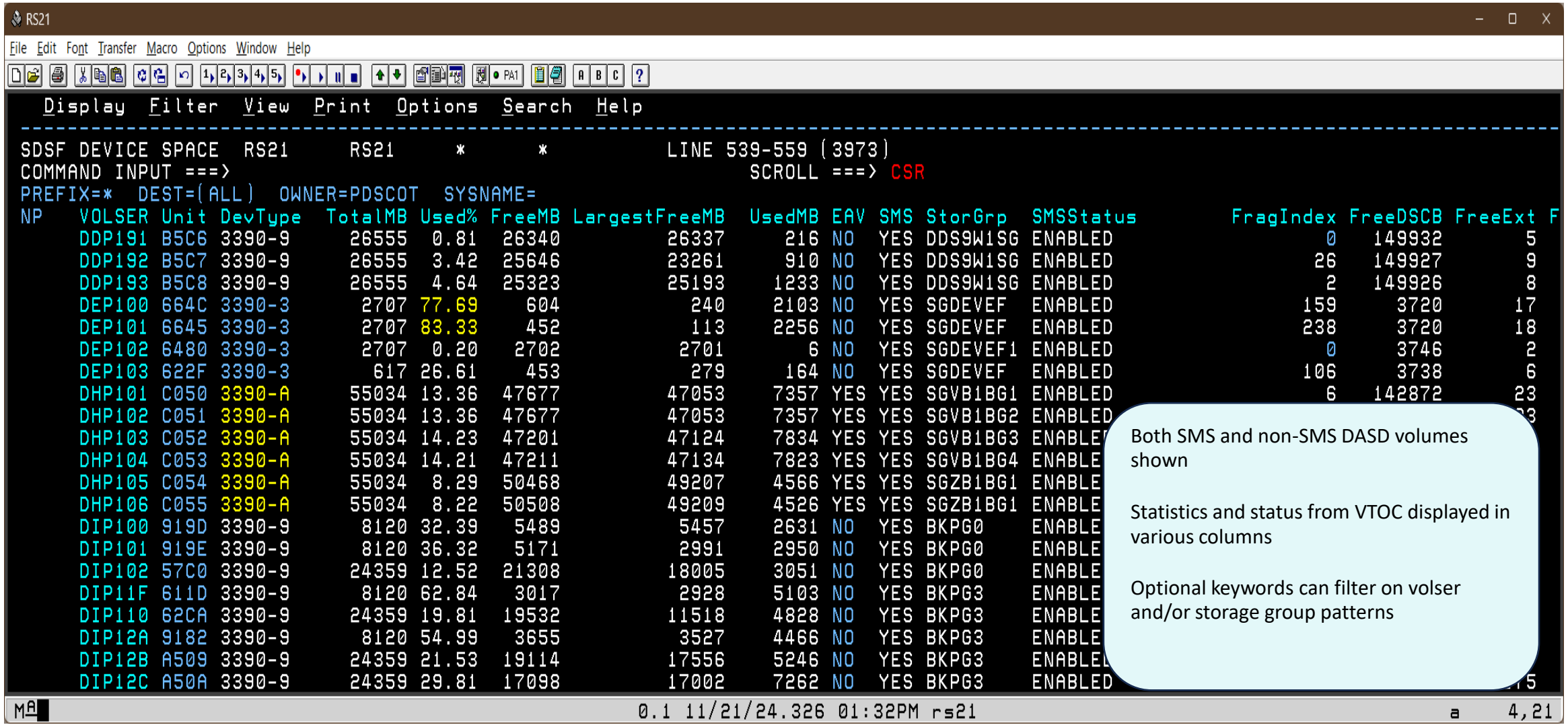
Every HV-COMMON memory object shown in ascending address sequence

Includes tracking information such as date-stamp and ASIDs in effect at creation

Panel also includes real frames and aux slots backing each memory object

© 2025 IBM Corporation

DEVS Panel – Device Space



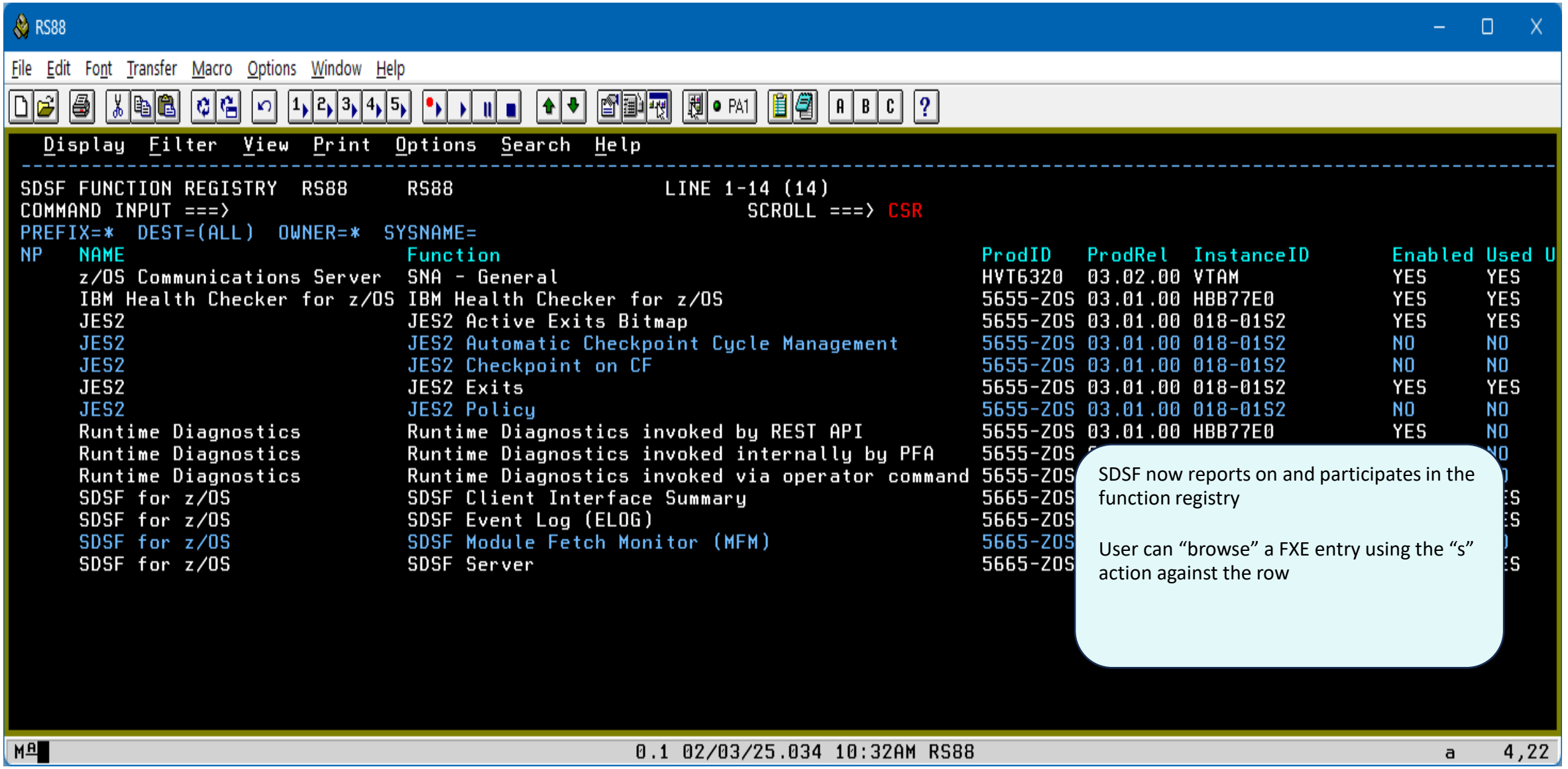
Both SMS and non-SMS DASD volumes shown

Statistics and status from VTOC displayed in various columns

Optional keywords can filter on volser and/or storage group patterns

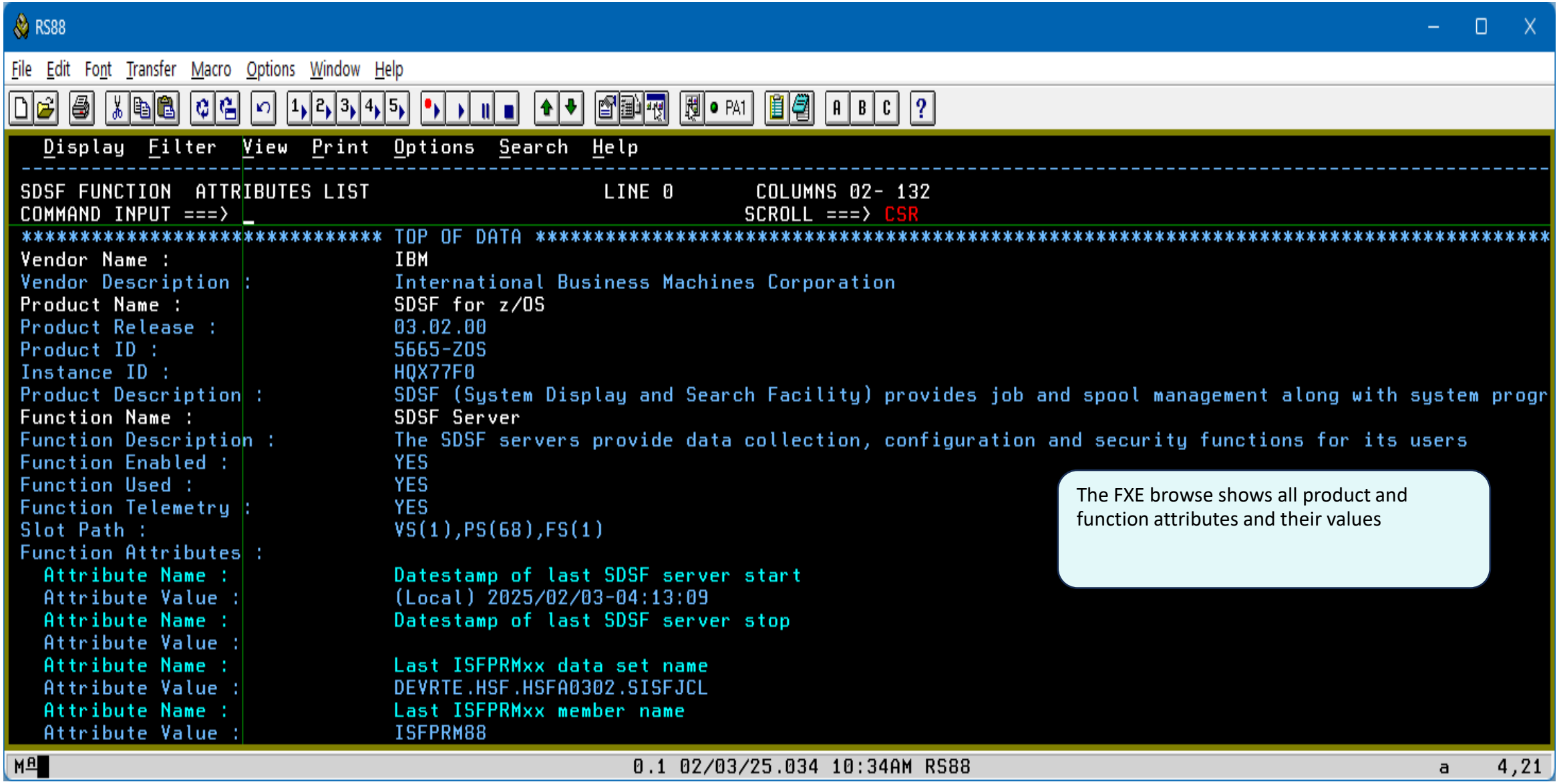
© 2025 IBM Corporation

FXE Panel – z/OS Function Registry



© 2025 IBM Corporation

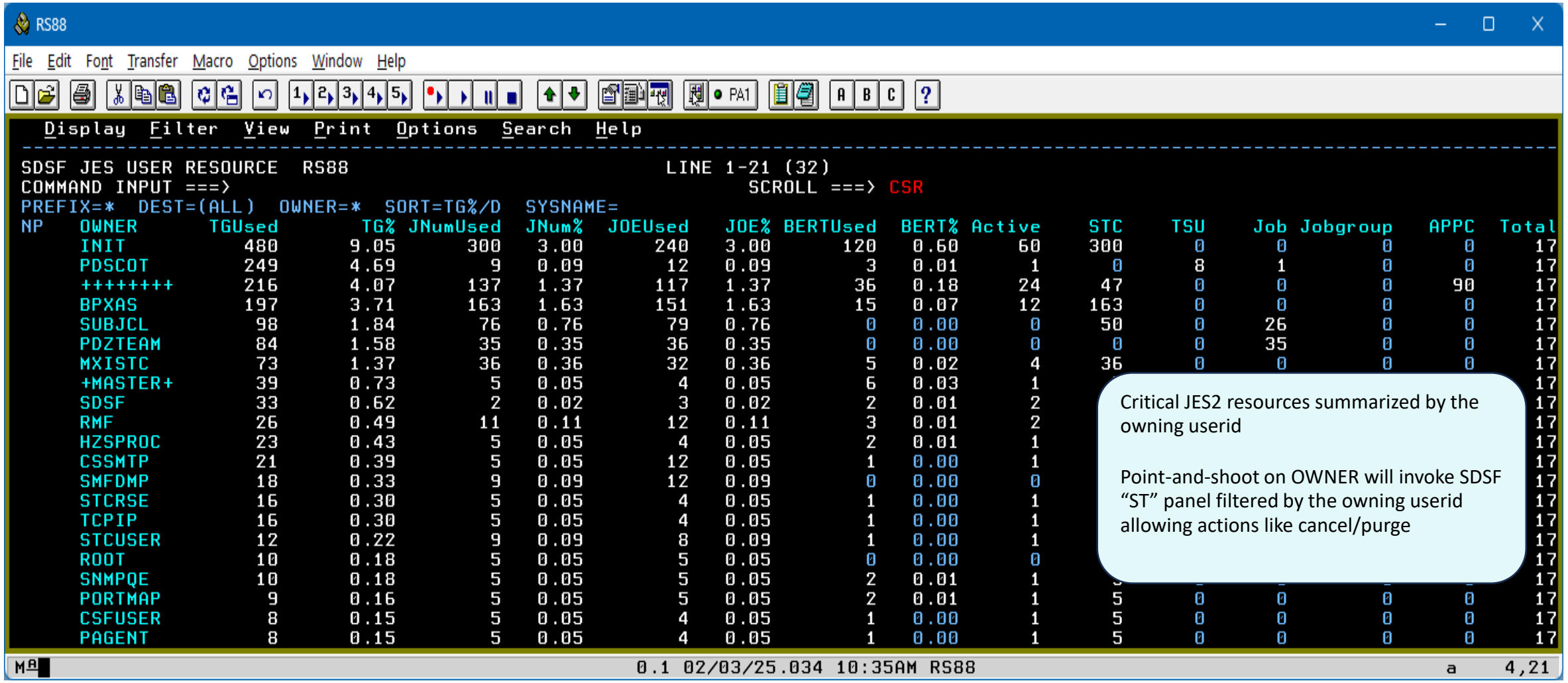
FXE Panel – z/OS Function Registry Browse



The FXE browse shows all product and function attributes and their values

© 2025 IBM Corporation

JRU Panel – JES2 Resource By Userid (Owner)

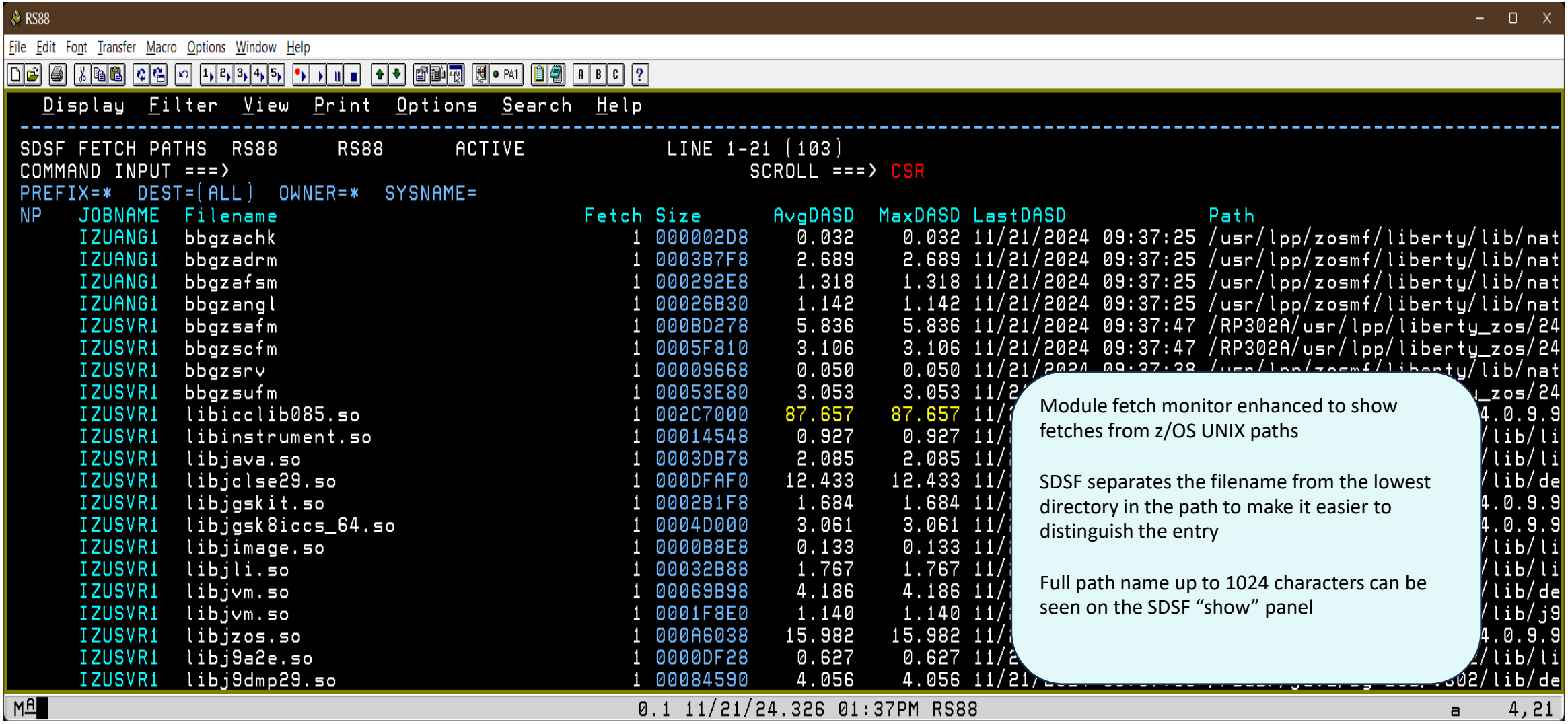


Critical JES2 resources summarized by the owning userid

Point-and-shoot on OWNER will invoke SDSF “ST” panel filtered by the owning userid allowing actions like cancel/purge

© 2025 IBM Corporation

MFP Panel – Module Fetch UNIX Path



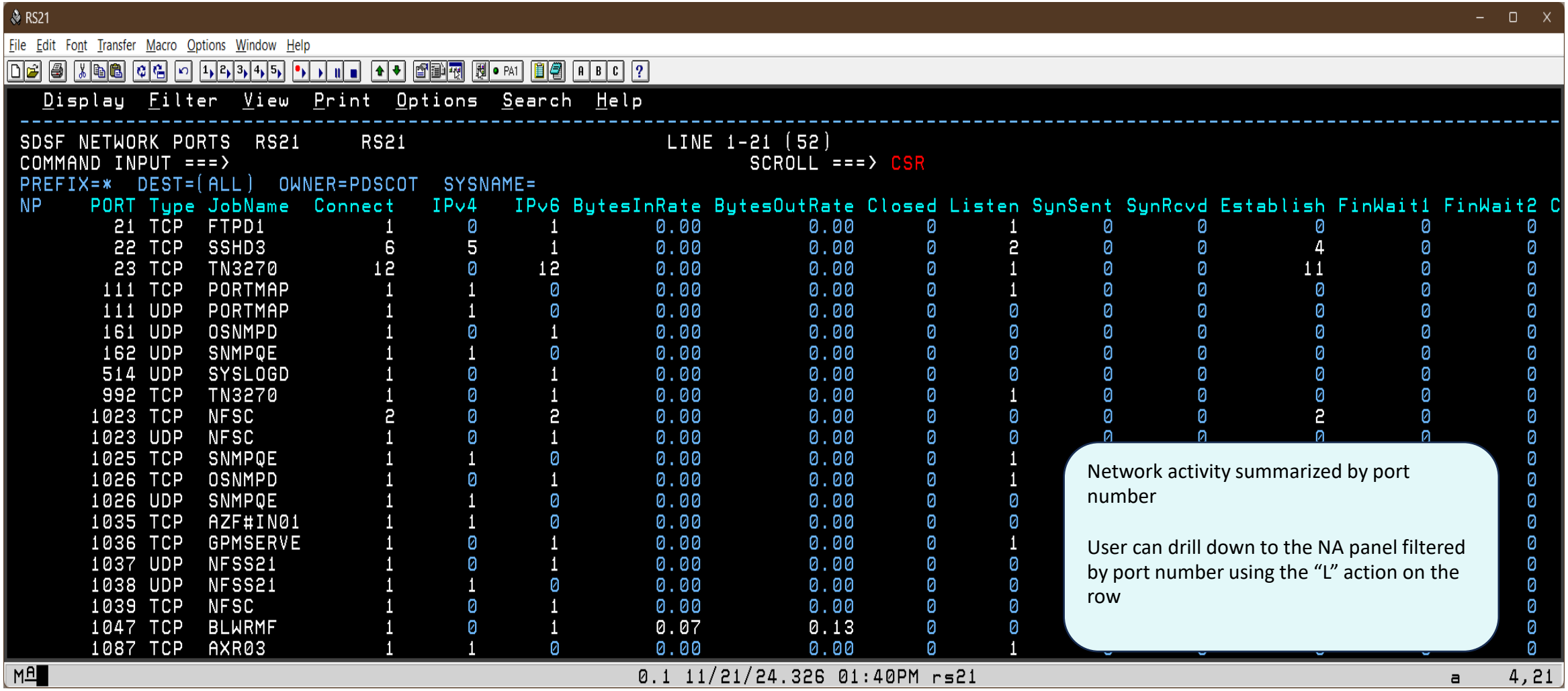
Module fetch monitor enhanced to show fetches from z/OS UNIX paths

SDSF separates the filename from the lowest directory in the path to make it easier to distinguish the entry

Full path name up to 1024 characters can be seen on the SDSF “show” panel

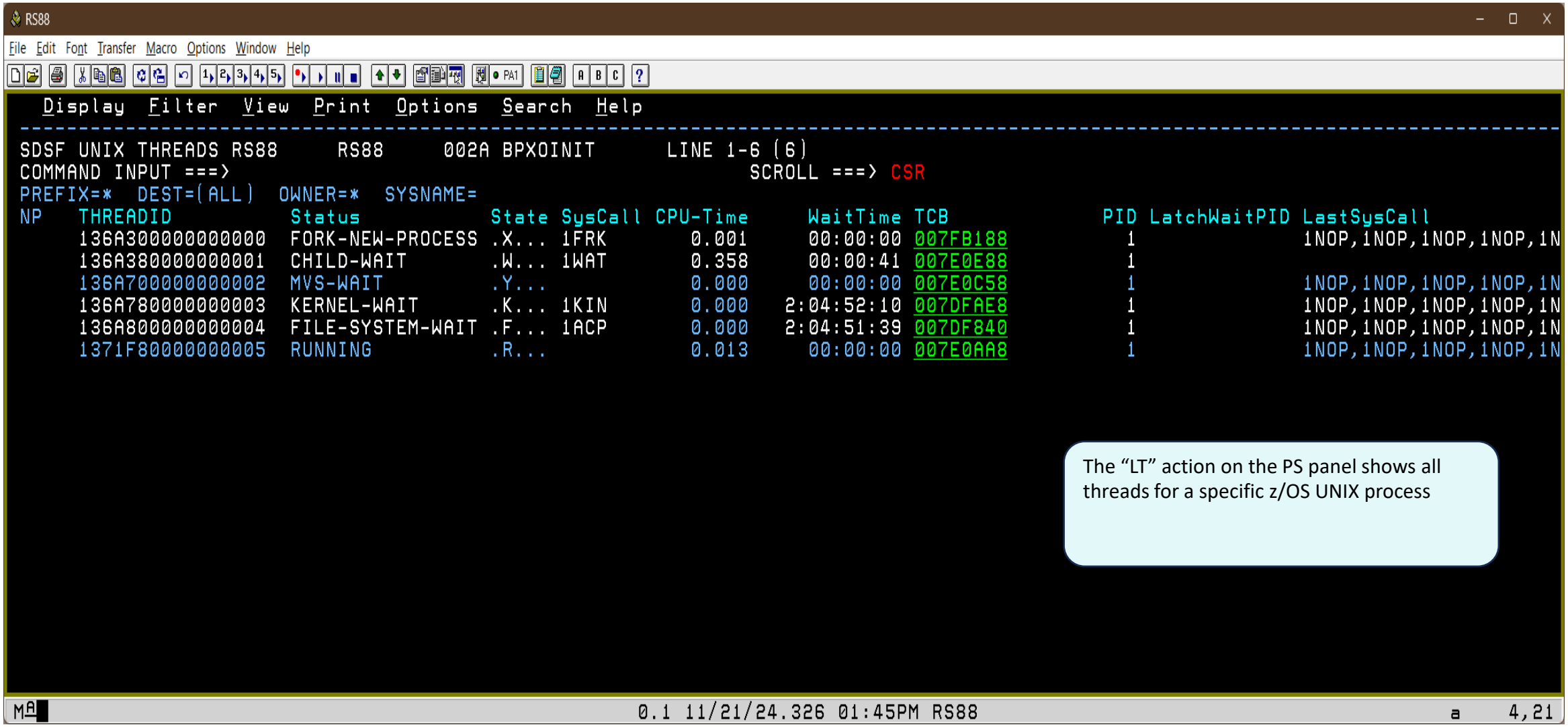
© 2025 IBM Corporation

NAP Panel – Network Activity By Port Number



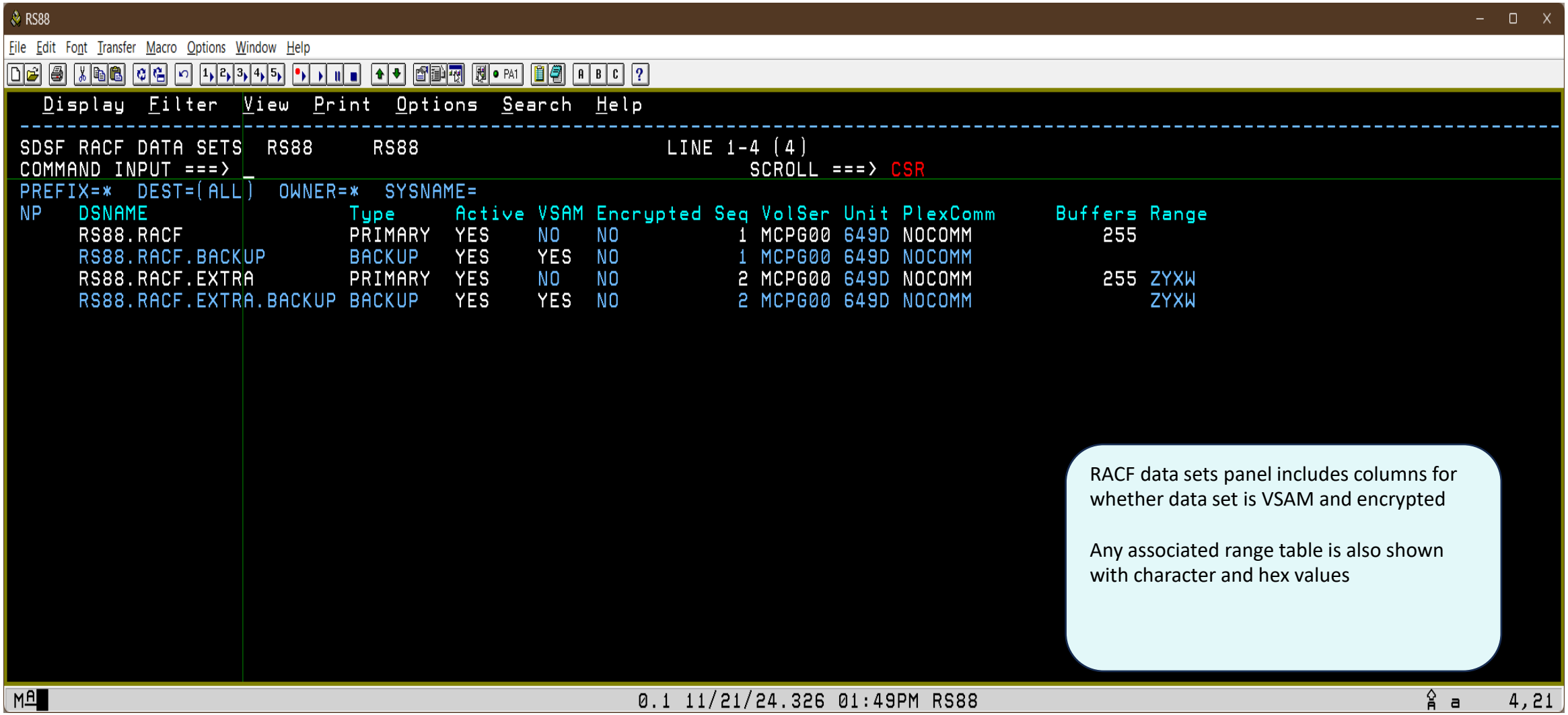
© 2025 IBM Corporation

PST Panel – z/OS UNIX Process Threads



© 2025 IBM Corporation

RACD Panel – RACF Data Sets

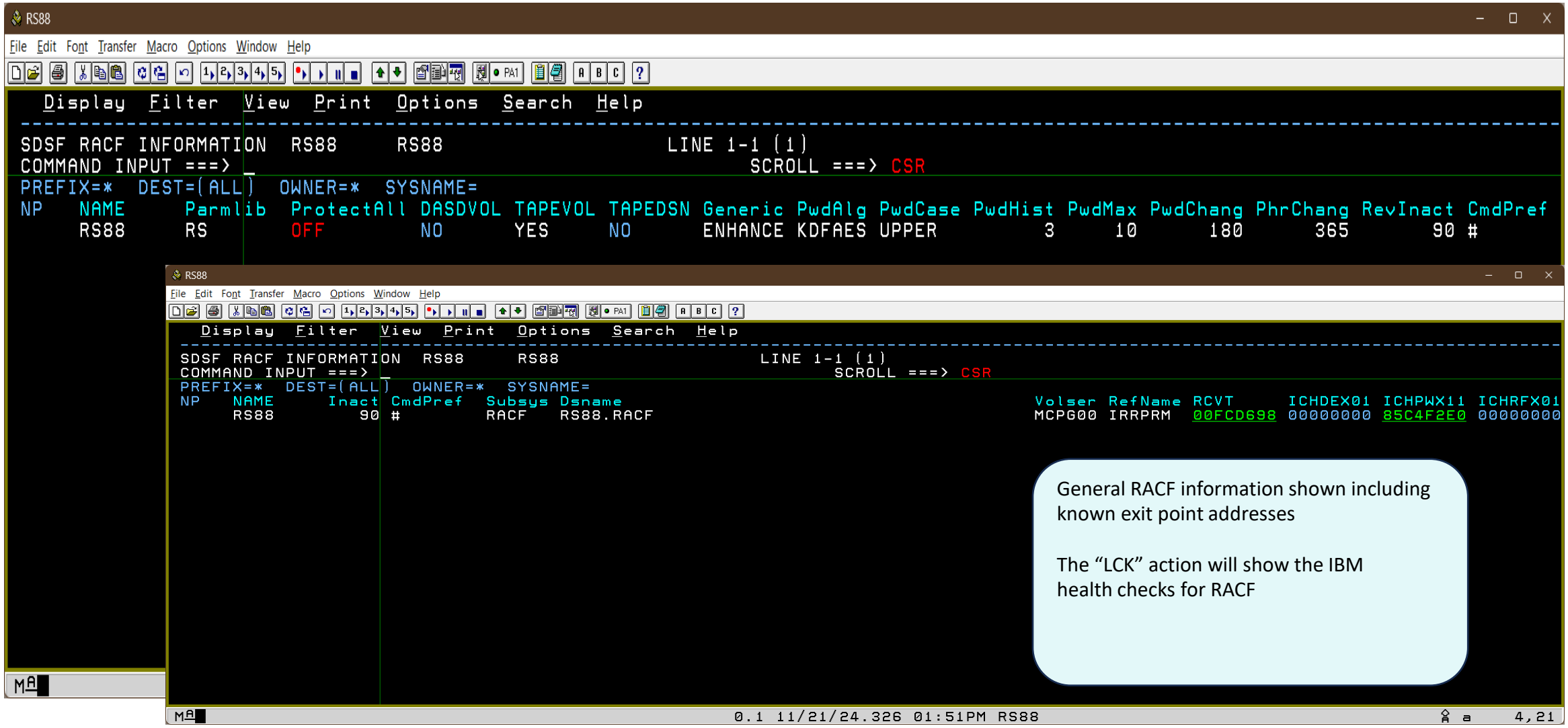


RACF data sets panel includes columns for whether data set is VSAM and encrypted

Any associated range table is also shown with character and hex values

© 2025 IBM Corporation

RACF Panel – General RACF Information

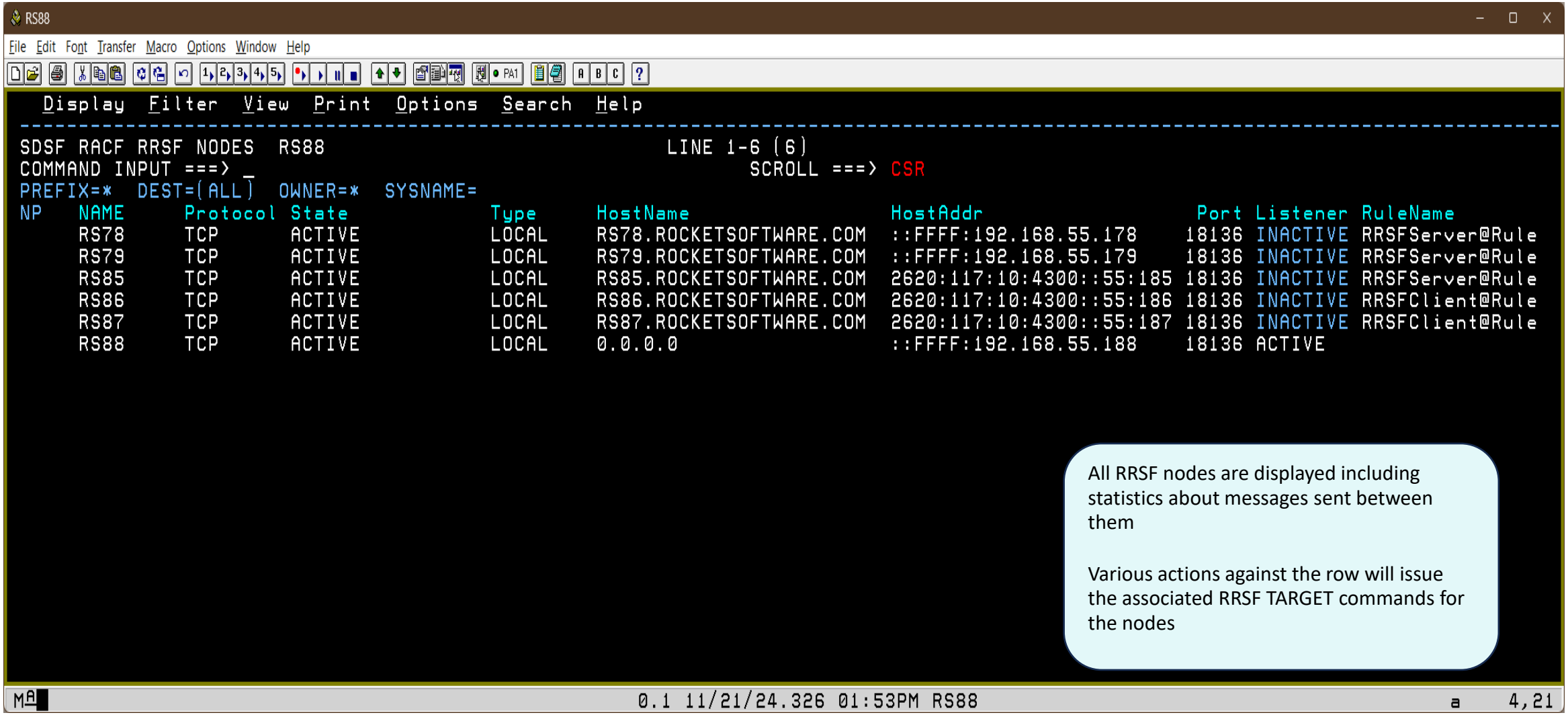


General RACF information shown including known exit point addresses

The “LCK” action will show the IBM health checks for RACF

© 2025 IBM Corporation

RACR Panel – RACF RRSF Nodes

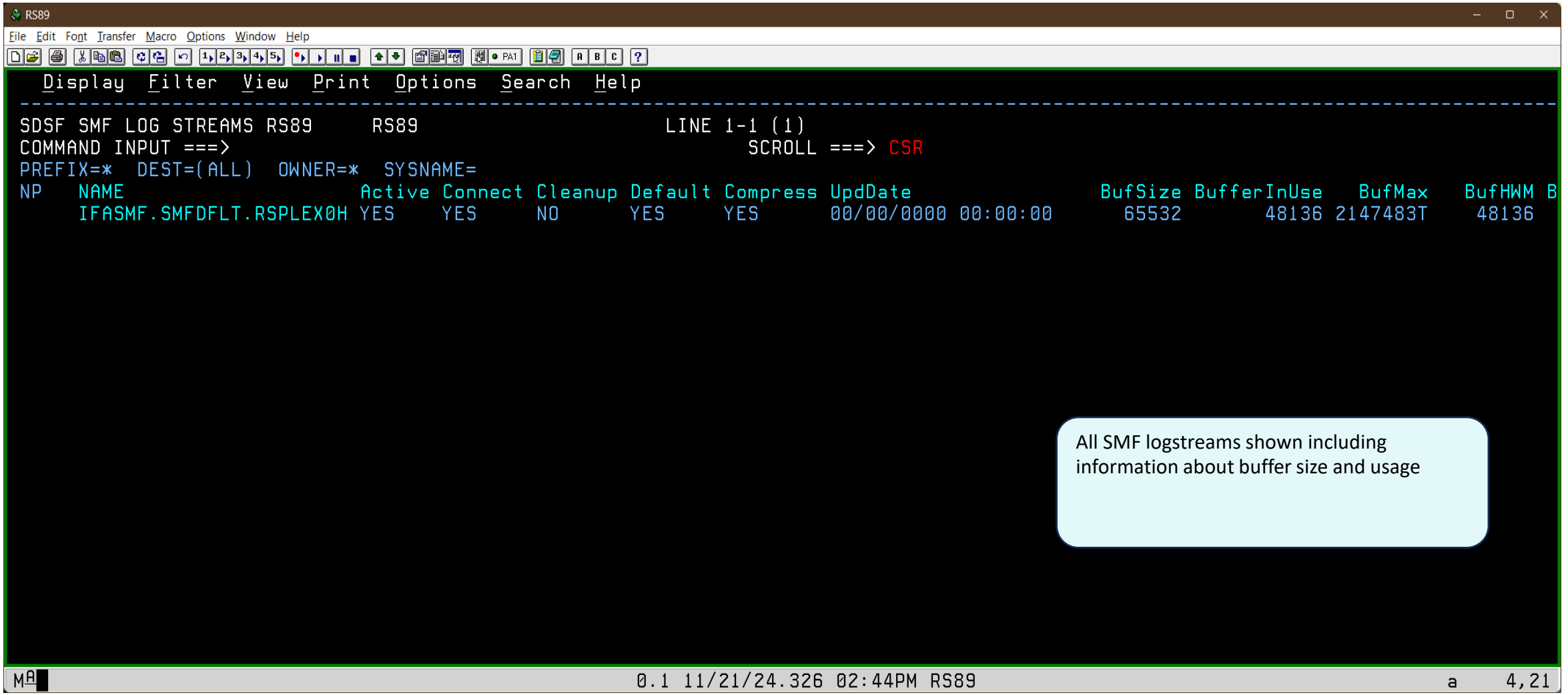


All RRSF nodes are displayed including statistics about messages sent between them

Various actions against the row will issue the associated RRSF TARGET commands for the nodes

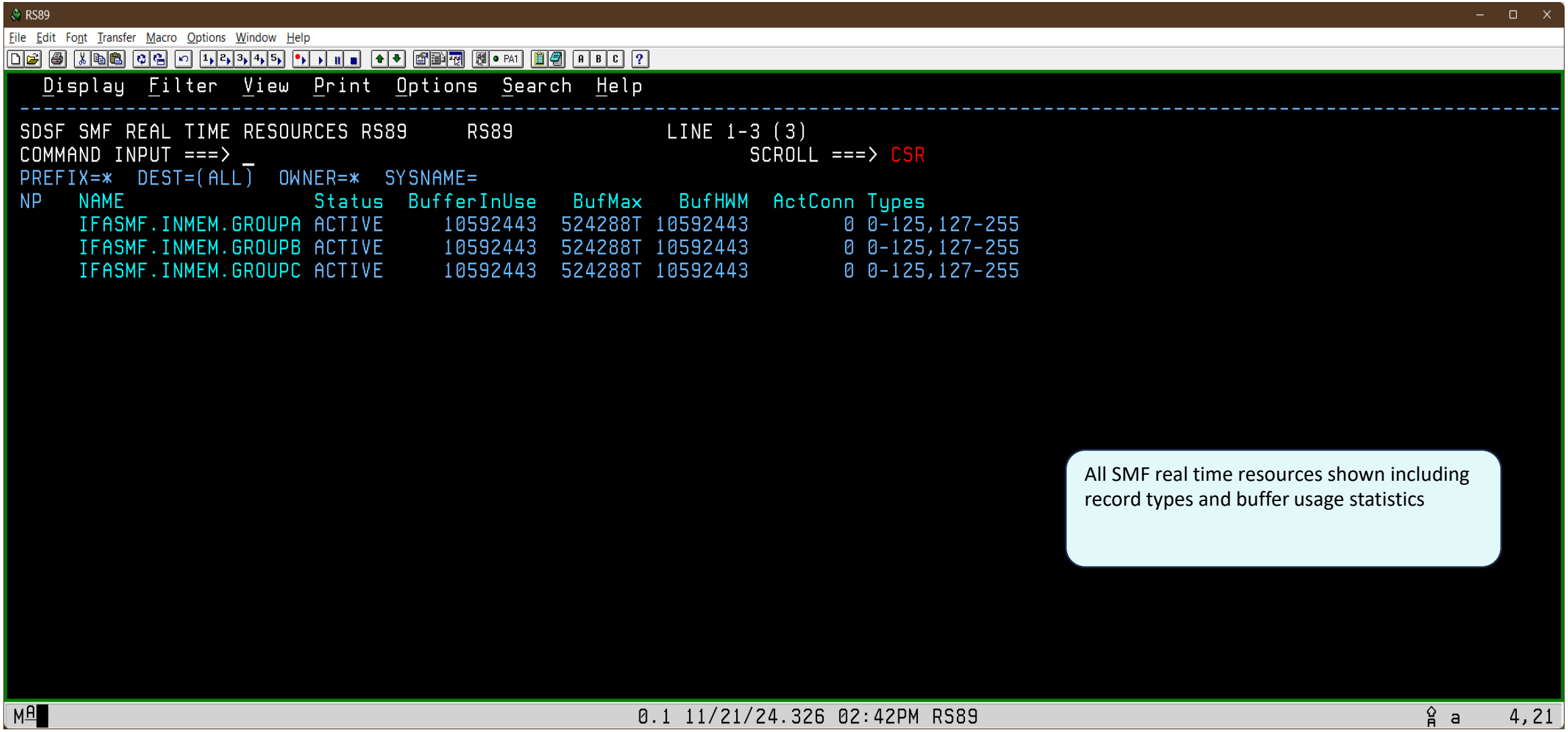
© 2025 IBM Corporation

SMFL Panel – SMF Logstreams



© 2025 IBM Corporation

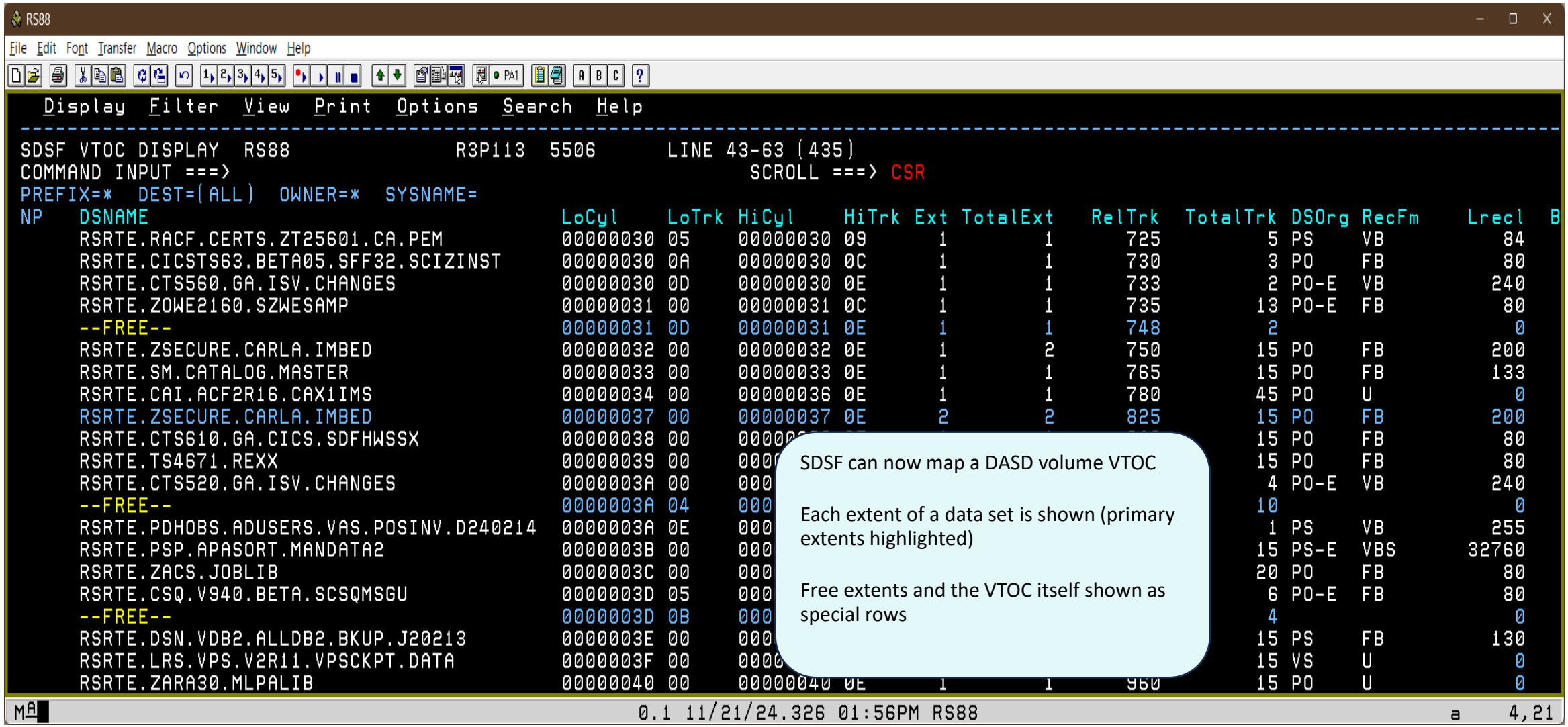
SMFR Panel – SMF Real Time Resources



All SMF real time resources shown including record types and buffer usage statistics

© 2025 IBM Corporation

VTOC Panel – DASD Volume Table Of Contents



© 2025 IBM Corporation

Useability Enhancements

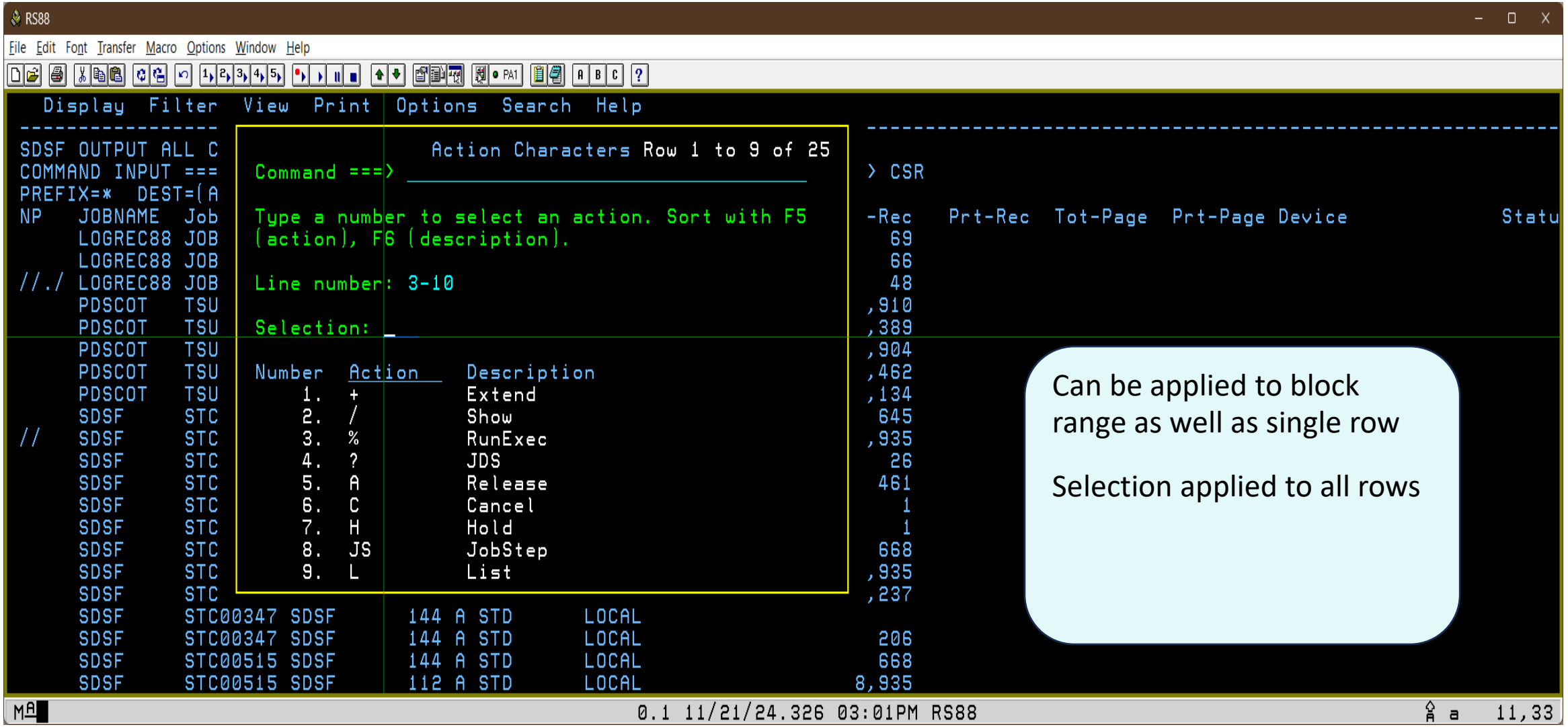
© 2025 IBM Corporation

Action Prompt Panel (1)

- New concept of SDSF “control” action char of a single dot “.”
  - Used in a couple of new enhancements
  - Common action character across all tabular panels
- Using the characters “./” in ISPF against a row will now cause a pop-up panel showing all actions defined for the panel in a scrollable list
- User can select entry to apply the action to the row(s)
- No longer need SET ACTION ON to see available actions

© 2025 IBM Corporation

Action Prompt Panel (2)



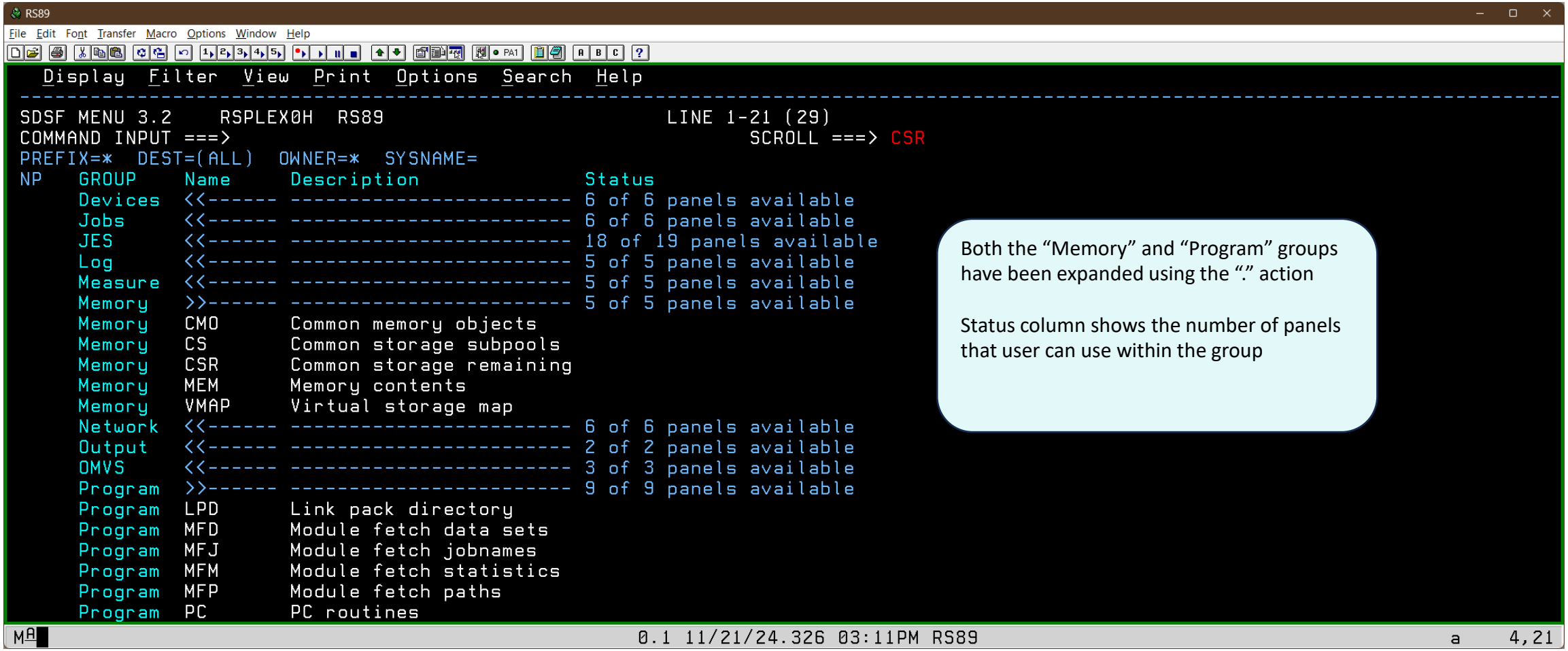
© 2025 IBM Corporation

MGRP Panel - Main Menu Group (1)

- Groups SDSF commands by their associated group name
- Allows user to expand or collapse individual groups to show associated commands
- Expand/collapse status of each group saved in user profile
  - Use “.” to expand and “..” to collapse
- MGRP command can be issued anywhere within product
- Use SET MAIN GROUP command to have this as the default SDSF main menu

© 2025 IBM Corporation

MGRP Panel - Main Menu Group (2)



Both the “Memory” and “Program” groups have been expanded using the “.” action

Status column shows the number of panels that user can use within the group

© 2025 IBM Corporation

Summary

- pySDSF
- SDSF Profile Data In zFS
- RACF Log Display
- New SDSF Panels
- Useability Enhancements

© 2025 IBM Corporation