z/OS
Version 2 Release 4

Authorized Code Scanner Guide

SC283123-40

Note

Before using this information and the product it supports, read the information in “Notices” on page 49.

This edition applies to Version 2 Release 4 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2020-12-13

Contents

Tables

Figures

About this Document

Who should use this document

This document is for general users of the IBM z/OS Authorized Code Scanner (zACS)

How to use this document

To use this document:

- Read Chapter 2, “Overview,” on page 3. It tells you how the IBM z/OS Authorized Code Scanner (zACS) is designed to discover vulnerabilities within Program Calls (PCs) and Supervisor Calls (SVCs) so that they can be subsequently remedied.
- Choose whether to run the Integrity Scanning Tool with either the panels or the commands:
  - If you want to use panels, read “Running using the Panel ” on page 19. This topic explains how to get help while using the panels.
  - If you want to use commands, “Running using the Command Line ” on page 17.
- Read Chapter 3, “Configuration,” on page 5 as it contains step-by-step procedures for you to follow.

How to send your comments to IBM

We invite you to submit comments about the z/OS® product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead “If you have a technical problem” on page xi.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM® Knowledge Center function
If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS Authorized Code Scanner Guide, SC283123-40
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Summary of changes for z/OS Version 2 Release 4 (V2R4)

The following changes are made to z/OS Version 2 Release 4 (V2R4).

December 2020

- For APAR OA60166, Post GA changes have been made. See “Configuration file” on page 10.

November 2020

- For APAR OA59928, Job Card Account Information is added. See Chapter 3, “Configuration,” on page 5.

Chapter 1. Introduction

System integrity is the inability of any program not authorized by a mechanism under the installation's control to circumvent or disable store or fetch protection, access a resource protected by a Security Server/Manager, or obtain control in an authorized state; that is, in supervisor state, with a protection key less than eight (8), or Authorized Program Facility (APF) authorized.

Program Call (PC) and Supervisor Call (SVC) routines provide a variety of critical services to z/OS. Some are created and maintained by IBM or IBM business partners as part of core z/OS subsystems and middleware products, for example. Others are created by vendor applications for z/OS. Some are implemented by system programmers to provide specialized in-house functionality for a client's enterprise.

PCs and SVCs have the architectural capability, depending on how they are defined, to allow an unauthorized program to invoke them, yet execute in an authorized state. The implementation of these PC and SVC routines must, therefore, handle this critical boundary carefully to ensure the system integrity of z/OS. Untrusted parameters, for example, need to be safely copied with architected instructions such as MVCSK and MVCDK so that an unauthorized program cannot fetch or update storage where it could not otherwise do so. For more information, see the z/Architecture Principles of Operation.

If a PC or SVC routine is implemented incorrectly, a security vulnerability may be introduced that compromises the system integrity of z/OS. The IBM z/OS Authorized Code Scanner (zACS) provides the ability to test PCs and SVCs on a given z/OS V2R4 instance to address this.

Chapter 2. Overview

The IBM z/OS Authorized Code Scanner (zACS) is designed to prevent unauthorized callers from being incorrectly granted an authorized state by detecting vulnerabilities within Program Calls (PCs) and Supervisor Calls (SVCs). The tool consists of a started task, a set of REXX execs, and associated data sets. The started task runs authorized while the tests generated by the REXX execs do not. By design, the zACS generates many ABENDs, and due to its volatile nature is meant to be run in a development and test environment. Both the input and output data sets contain sensitive data, as the input determines the scope of the scan and the output may describe potential vulnerabilities found on z/OS, and their respective locations. Therefore, these data sets, along with the REXX execs that help generate them, must be properly protected by an external security manager product such as RACF.

The challenge of finding programming errors in code integrity is that the code base largely functions normally in general forms of testing. The PC and SVC routines implement a critical boundary between unauthorized users and authorized code. Any z/OS instance contains hundreds, often thousands of these routines. They are built into applications, middleware, and the operating system itself. They are created by IBM, by the z/OS vendor community, and by clients' in-house applications tailored to the needs of their respective enterprises. The zACS provides the ability to scan these routines. As with any scanning tool, there may be programmer errors that cannot yet be detected and conversely the tool may produce some false positives. Fundamentally, the zACS provides a new, foundational cybersecurity analysis via patented algorithms that dynamically scan PC and SVC code on z/OS, supporting the growing movement of DevSecOps.

Chapter 3. Configuration

This topic assumes a high level qualifier of 'SYS1.BPN.**'. We recommend that your system be up to date on service before running the tool. Additionally, the REXX executables used in this tool assume the user is signed onto the user id ‘userid’.

Run the

```
SETPROG APF,ADD,DSNAME=<load-auth-dsn>,VOLUME=<volume-of-load-auth-dsn>
```

Figure 1. APF (Authorized Program Facility) load library command

command in order to add the load library to the dynamic authorized program facility (APF) list.

Note: This command does not persist through re-IPL. It is recommended to add this to your PARMLIB. https://www.ibm.com/support/knowledgecenter/en/SSQ2R2_9.5.1/com.ibm.guide.hostconfig.doc/topics/apf_auth_progxx.html>

Data Set Protection and Permissions

zACS is a tool that analyzes the SVCs/PCs of the current execution environment of a z/OS image. Since it will be reporting on security characteristics, access to the libraries for this tool is best kept to a known and trusted set of users. The executable library should not be placed in the LINKLIST or LPA.

Note: The following instructions are specific to z/OS Security Server (RACF). If another ESM is used then refer to the applicable vendor documentation for equivalent security configuration.

Example:

The following example is described with z/OS Security Server (RACF) commands where enhanced generic naming is allowed in the DATASET and the FACILITY classes, and list-of-groups access checking is active.

Step 1: Protect the data sets containing the tool

The base tool consists of the following data sets by default:

```
SYS1.BPN.SBPNCFG
SYS1.BPN.SBPNEXEC
SYS1.BPN.SBPNLOAD
SYS1.BPN.SBPNSAMP
```

1. Define a group of users who will be given access to run zACS and to examine its results.

   ```
   ADDGROUP BPNGRP
   ```

2. Define profile to protect data sets

   ```
   ADDSD ‘SYS1.BPN.**’ UACC(NONE)
   ```

3. Grant the group access to the profile

   ```
   PERMIT ‘SYS1.BPN.**’ CLASS(DATASET) ID(BPNGRP) ACCESS(READ)
   ```

4. Refresh the in-storage profiles so that the changes you have made to the FACILITY class take

   ```
   SETR GENERIC(DATASET) REFRESH
   ```

5. Add the necessary userids to the group

   ```
   CONNECT userid GROUP(BPNGRP)
   ```

Step 2: Protect the tool’s generated output and configuration.

As per the instructions found in the “Configuration file” on page 10 section, the configuration file is placed in the following sequential data set:

```
userid.ZACS.CONFIG
```

By default, the PC/SVC table are generated and placed in the respective data sets:

```
userid.ZACS.PCNUM
```

```
userid.ZACS.SVCNUM
```

The location of the PC log, SVC log, output data set, PC exclude list, and SVC exclude list are configurable in the configuration file and started task, however, we strongly suggest the following naming convention:

```
userid.ZACS.PCLOG
userid.ZACS.SVCLOG
userid.ZACS.OUTPUT
userid.ZACS.FILTER.MODNAME
userid.ZACS.FILTER.JOBNAME
```

Protect the tool’s generated output and configuration files and permit the users to have access to them. It is assumed all output and configuration files are under the high-level qualifier ‘*userid*.ZACS’

1. Define profile to protect the data sets

   ```
   ADDSD 'userid.ZACS.**' UACC(NONE)
   ```

2. Grant the group access to the profile

   ```
   PERMIT ‘userid.ZACS.**’ CLASS(DATASET) ID(BPNGRP) ACCESS(UPDATE)
   ```

   Note: If the set of users accessing the output from the tool is different from the users who can execute it, you can define a second group containing those users and permit that group to the profile.

Figure 2. Define profile and grant group access to protect data sets.

Step 3: Grant permissions to other resources if needed for your configuration

1. Permit the group update access to LPA

   ```
   PE CSVDYLPA.ADD.BPN.** CL(FACILITY) ID(BPNGRP) ACC(UPDATE)
   ```

2. If using a data set for logrec and the clearlogrec option in the configuration file is set to yes, permit the group update access to the logrec data set

   ```
   PE 'SYS1.LOGREC' ID(BPNGRP) ACC(UPDATE)
   ```

Figure 3. Grant permission to LPA and the logrec data set

Service Authorization

The user of zACS must have READ authority to the resource BPN.RUN in the XFACILIT class. Below is an example where resource-name would be replaced with BPN.RUN.

Note: The following instructions are specific to z/OS Security Server (RACF). If another ESM is used then refer to the applicable vendor documentation for equivalent security configuration.

The following example is described with z/OS Security Server (RACF) commands where enhanced generic naming is allowed in the DATASET and the FACILITY classes, and list-of-groups access checking is active.

An administrator with the appropriate RACF authority would:

1. Activate the XFACILIT class and RACLIST it

```
SETR CLASSACT(XFACILIT) RACLIST(XFACILIT)
```

2. Define the resource to the XFACILIT class

```
RDEFINE XFACILIT resource-name UACC(NONE) AUDIT(ALL)
```

3. Permit the user or group read access to the resource

```
PERMIT resource-name CLASS(XFACILIT) ID(user or group ID) ACCESS(READ)
```

4. Refresh the in-storage profiles so that the changes you have made to the XFACILIT class take

```
SETR RACLIST(XFACILIT) REFRESH
```

LOGREC

Note: A logstream may be used instead of a LOGREC data set. If a logstream is used this section may be skipped.

A minimum of 10 cylinders is recommended for your LOGREC data set to ensure complete results. To verify the number of cylinders, check the size of the data set in ISPF 3.4:

```
                          Data Set
Information
 Command ===>
                                                                More:     +
 Data Set Name . . . . : SYS1.ZATB.LOGREC

 General Data                          Current Allocation
  Management class . . : **None**       Allocated cylinders : 10
  Storage class  . . . : **None**       Allocated extents . : 1
   Volume serial . . . : SYSA00
   Device type . . . . : 3390
  Data class . . . . . : **None**
   Organization  . . . : PSU           Current Utilization
   Record format . . . : U              Used cylinders  . . : 10
   Record length . . . : 0              Used extents  . . . : 1
   Block size  . . . . : 1944
   1st extent cylinders: 10
   Secondary cylinders : 0             Dates
   Data set name type  :                Creation date . . . : 2018/06/14
                                        Referenced date . . : 2019/08/29
                                        Expiration date . . : ***None***


  F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
 F12=Cancel
```

Figure 4. Cylinder requirement for LOGREC data set.

Note: The number of test iterations, and thus logrec entries, will differ depending on the PC or SVC being tested. While 10 cylinders is sufficient for most services, in some cases, services will not complete testing without a larger logrec data set. Such cases will result in a Return Code C, Reason Code C09, followed by an INCOMPLETE summary status.

If you are not sure the name of your LOGREC data set, you can find it with the console command:

```
D LOGREC
```

If needed, refer to the following for instructions on how initialize: https://www.ibm.com/support/knowledgecenter/SSLTBW_2.4.0/com.ibm.zos.v2r4.ieav100/init.htm and switch to: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.4.0/com.ibm.zos.v2r4.ieag100/setlog.htm, a different, presumably larger, logrec data set.

Note: If running on a virtual machine, the SVC76 VM option must be set to 'VM' so that the program checks created by the tool are properly sent to the logrec data set: Information for setting SVC76

.

JES

To assure the submitted test jobs complete before starting the next test, the system must be set to not run jobs of the same name simultaneously. To do this, set your initiators to DUPL_JOB=DELAY if using JES2, and DUPJOBNM=NO if using JES3. These are the default settings.

Started task (BPNZACS)

Define a profile in the STARTED class to associate the user id with the zACS address space:

1. Authorize the started task

```
RDEFINE STARTED BPNZACS.** UACC(NONE) STDATA(USER(userid) GROUP(SYS1) TRUSTED(YES))
```

2. Refresh that STARTED class

```
SETR RACLIST(STARTED) REFRESH
```

The following JCL is contained in 'SYS1.BPN.SBPNSAMP(BPNZACS)' data set. This needs to be copied into the working PROCLIB. MYOUTDD must point to the potential vulnerability data set, which is used for the output of the tool. When this task is started, the output data set will be cleared of any data, including any vulnerability information from a previous time the task was running.

```
//BPNZACS PROC
//*********************************************************************
//* THIS EXECUTES THE BPNZACS PROGRAM FOR INTEGRITY TESTING.          *
//* PUT THIS JCL IN THE PROCLIB DATA SET USED FOR STARTED TASKS       *
//* AND MAKE THE CORRESPONDING RACF UPDATES, E.G.                     *
//*   RDEFINE STARTED BPNZACS.** UACC(NONE) STDATA(USER(user)         *
//*     GROUP(SYS1) TRUSTED(YES))                                     *
//*   SETR RACLIST(STARTED) REFRESH                                   *
//*                                                                   *
//* INSTRUCTIONS:                                                     *
//*   CHANGE loadlib-dataset TO THE LOAD LIB DATASET UNDER YOUR HLQ   *
//*   CHANGE output-dataset TO THE ALLOCATED DATASET SPECIFIED FOR    *
//*     OUTPUT                                                        *
//*********************************************************************
//GOSTEP EXEC PGM=BPNGMAIN,TIME=NOLIMIT
//STEPLIB  DD DSN=loadlib-dataset,DISP=SHR
//MYOUTDD  DD DSN=output-dataset,DISP=SHR
```

Figure 5. SYS1.BPN.SBPNSAMP(BPNZACS)

BPNPCNUM

The following JCL is contained in 'SYS1.BPN.SBPNSAMP(BPNPCNUM)' data set. This should be copied to a separate JCL data set. The output of BPNPCNUM goes to '*userid*.ZACS.PCNUM' containing the list of PC numbers for the system.

Note: The SYSTEM= keyword may be added to the job card to specify the current system on which zACS will be run.

```
//ZACSJP JOB  NOTIFY=&SYSUID,MSGCLASS=A,REGION=5M
//*********************************************************************
//* GENERATES THE PC TABLE FOR INTEGRITY TESTING.                     *
//*                                                                   *
//* INSTRUCTIONS:                                                     *
//*   CHANGE loadlib-dataset TO THE LOAD LIB DATASET UNDER YOUR HLQ   *
//*********************************************************************
//DELETE   EXEC PGM=IEFBR14
//DELDSN   DD DISP=(MOD,DELETE),DSN=&SYSUID..ZACS.PCNUM,
//            SPACE=(TRK,1),UNIT=SYSDA
//*
//CREATE   EXEC PGM=BPNGPCN
//STEPLIB  DD DSN=loadlib-dataset,DISP=SHR
//BPNPCOUT DD DISP=(NEW,CATLG),DSN=&SYSUID..ZACS.PCNUM,
//             SPACE=(TRK,(10,10)),UNIT=SYSDA,
//             DCB=(LRECL=133,BLKSIZE=0,RECFM=FBA)
//SYSOUT   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Figure 6. SYS1.BPN.SBPNSAMP(BPNPCNUM)

BPNSVCNM

The following JCL is contained in 'SYS1.BPN.SBPNSAMP(BPNSVCNM)' data set. This should be copied to a separate JCL data set. The output of BPNSVCNM goes to '*userid*.ZACS.SVCNUM' containing the list of SVC numbers for the system.

Note: The SYSTEM= keyword may be added to the job card to specify the current system on which zACS will be run.

```
//ZACSJS JOB  NOTIFY=&SYSUID,MSGCLASS=A,REGION=5M
//*********************************************************************
//* GENERATES THE SVC TABLE FOR INTEGRITY TESTING.                    *
//*                                                                   *
//* INSTRUCTIONS:                                                     *
//*   CHANGE loadlib-dataset TO THE LOAD LIB DATASET UNDER YOUR HLQ   *
//*********************************************************************
//DELETE   EXEC PGM=IEFBR14
//DELDSN   DD DISP=(MOD,DELETE),DSN=&SYSUID..ZACS.SVCNUM,
//            SPACE=(TRK,1),UNIT=SYSDA
//*
//CREATE   EXEC PGM=BPNGSVCN
//STEPLIB  DD DSN=loadlib-dataset,DISP=SHR
//BPNSVOUT DD DISP=(NEW,CATLG),DSN=&SYSUID..ZACS.SVCNUM,
//             SPACE=(TRK,(10,10)),UNIT=SYSDA,
//             DCB=(LRECL=133,BLKSIZE=0,RECFM=FBA)
//SYSOUT   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Figure 7. SYS1.BPN.SBPNSAMP(BPNSVCNM)

Registration

Update your IFAPRDxx parmlib member with the following text:

```
    PRODUCT OWNER('IBM CORP')
    NAME('z/OS')
    ID(5650-ZOS)
    VERSION(*) RELEASE(*) MOD(*)
    FEATURENAME('ZACS')
    STATE(ENABLED)
```

Figure 8. IFAPRDxx parmlib member example

You can update your active IFAPRDxx parmlib member with the SET PROD=xx console command.

Configuration file

The zACS sample configuration file can be found in 'SYS1.BPN.SBPNSAMP(BPNCFG)'. Copy this file into a sequential data sate named '*userid*.ZACS.CONFIG' for runtime use. The suggested attributes are as follows:

```
Data Set Name . . . . : userid.ZACS.CONFIG

General Data                            Current Allocation
 Management class . . : **None**         Allocated tracks  . : 1
 Storage class  . . . : **None**         Allocated extents . : 1
  Volume serial . . . : TEMP01
  Device type . . . . : 3380
 Data class . . . . . : **None**
  Organization  . . . : PS               Current Utilization
  Record format . . . : FB                Used tracks . . . . : 1
  Record length . . . : 133               Used extents  . . . : 1
  Block size  . . . . : 23408
  1st extent tracks . : 1
  Secondary tracks  . : 0                Dates
  Data set name type  :                   Creation date . . . : 2020/06/05
  Data set encryption : NO                Referenced date . . : 2020/06/05
                                          Expiration date . . : ***None***

  SMS Compressible  . : NO
```

Figure 9. userid.ZACS.CONFIG

The configuration file contains several keywords that will need to be set before running. Single quotes within the keyword's value may only be used in the keywords *jobAccount* and *jobProgrammer* and must be escaped by doubling them. Blank spaces within a keyword's value may only be used in the keyword *jobProgrammer*. A keyword's value may take up multiple lines, this can be done by ending the line you wish to continue with a comma. Example:

```
jobOtherKeywords    =
        'NOTIFY=ZACSUSER,',
        'PRTY=12'
```

Will result in the value NOTIFY=ZACSUSER,PRTY=12.

The keywords are as follows:

clearLogrec
: zACS creates many program checks that can quickly fill up the system's logrec data set. zACS will not be able to properly execute its testing if the logrec data set becomes full. When 'YES' is specified, a job to clear the contents of the logrec data set will be run after each PC or SVC is tested to avoid filling the logrec data set. Note that the logrec data set will be cleared completely, including information from other programs using it. If a logstream is used instead of a logrec data set, then this keyword is ignored.

logrecDSname
: The name of the logrec data set that will be cleared by the clear logrec job. If using a logstream instead of a logrec data set, specify 'LOGSTREAM'.

testcaseJobName
: The JCL job name to be used for all test jobs. Also used as the JCL job name for the clear logrec job, if clearlogrec is specified as 'YES'.

jobMsgClass
: The job message class for the test JCL jobs. Also used as the job message class for the clear logrec job, if clearlogrec is specified as 'YES'.

jobRegion
: Sets the job REGION parameter for the test JCL jobs. Valid units are 'M' and 'K'. If clearlogrec is specified as 'YES', this is also used as the job region for the clear logrec job.

jobTimeout
: Sets the job TIME parameter for the test JCL jobs. This limit is specified in minutes. If clearlogrec is specified as 'YES', this is also used as the job timeout for the clear logrec job.

jobSystem
Sets the job SYSTEM parameter for the test JCL jobs. If clearlogrec is specified as ‘YES’, this is also used as the job system for the clear logrec job.

printLogOutput
The display option for the list of PC or SVC numbers submitted during an zACS run. If ‘SCREEN’ is specified, the display is sent to the screen. If ‘LOG’ is specified, this display is redirected to the respective data sets specified by pcLogDSname and svcLogDSname.

Note: These data sets will be overwritten the next time the tool is run.

jclDSname
The dataset containing BPNPCNUM and BPNSVCNM. Used to submit BPNPCNUM and BPNSVCNM from the panel.

pcTableDSname
The dataset containing the PC table, generated from BPNPCNUM. The special keyword '&userid' can be used to substitute your userid. For example, '&userid.ZACS.PCNUM'. If omitted, the default is '&userid.ZACS.PCNUM'.

svcTableDSname
The dataset containing the SVC table, generated from BPNSVCNM. The special keyword '&userid' can be used to substitute your userid. For example, '&userid.ZACS.SVCNUM'. If omitted, the default is '&userid.ZACS.SVCNUM'.

useJobAccount
The yes/no switch on whether to use the account information parameter on the JCL job card. If clearLogrec is specified as ‘YES’, this switch also applies to the clear logrec job.

jobAccount
Sets the positional account information parameter for the test JCL jobs. If clearLogrec is specified as ‘YES’, it is also used as the account information for the clear logrec job. If single quotes are used, they must be escaped by doubling them. Example: to assign the account information D548-8686,’12/8/85’,PGMBIN use jobAccount = ‘D548-8686,’’12/8/85’’,PGMBIN’

useJobProgrammer
The yes/no switch on whether to use the programmer name parameter on the JCL job card. If clearLogrec is specified as ‘YES’, this switch also applies to the clear logrec job.

jobProgrammer
Sets the positional programmer name parameter for the test JCL jobs. . If clearLogrec is specified as ‘YES’, it is also used as the programmer name for the clear logrec job. If single quotes are used, they must be escaped by doubling them. Example: to assign the name Dave O’dell use jobProgrammer = ‘DAVE O’’DELL’

useOtherKeywords
The yes/no switch on whether to use any other keyword parameters on the JCL job card not already specified. If clearLogrec is specified as ‘YES’, this switch also applies to the clear logrec job.

jobOtherKeywords
Sets the any other keyword parameters desired for the test JCL jobs. . If clearLogrec is specified as ‘YES’, they are also applied to the clear logrec job. Full keyword parameter syntax must be used, with multiple keywords being separated by quotes. For example, jobOtherKeywords = 'NOTIFY=ZACSUSER,PRTY=12'

includeOrExclude
The switch between an inclusion filter and an exclusion filter. If ‘INCLUDE’ is specified, only services found in the jobFilterDSname and modFilterDSname will be run if filterByJobName and filterbyModName are specified as ‘YES’, respectively. If ‘EXCLUDE’ is specified, services found in the jobFilterDSname and modFilterDSname will be excluded if filterByJobName and filterbyModName are specified as ‘YES’, respectively.

filterbyModName
The yes/no switch on whether to use the include/exclude by module name list.

modFilterDSname

The sequential data set to contain a list of module names to be included or excluded from a full run. Applies to both PC and SVC runs. The suggested attributes are as follows:

```
Data Set Name . . . . : userid.ZACS.FILTER.MODNAME
General Data                          Current Allocation
  Management class . . : **None**       Allocated tracks  . : 5
  Storage class . . . : **None**        Allocated extents . : 1
   Volume serial . . . : TEMP01
   Device type . . . . : 3380
  Data class . . . . . : **None**
   Organization . . . : PS             Current Utilization
   Record format . . . : FB              Used tracks . . . . : 1
   Record length . . . : 133             Used extents  . . . : 1
   Block size . . . . : 23408
   1st extent tracks . : 5
   Secondary tracks . : 1              Dates
   Data set name type  :                 Creation date . . . : 2020/01/22
   Data set encryption : NO              Referenced date . . : 2020/01/22
                                         Expiration date . . : ***None***    SMS
Compressible  . : NO
```

Figure 10. userid.ZACS.FILTER.MODNAME

filterByJobName

The yes/no switch on whether to use the include/exclude by job name list.

jobFilterDSname

The sequential data set to contain a list of job names to be included or excluded from a full run of the tool. Only applies to PC run. The suggested attributes are as follows:

```
Data Set Name . . . . : userid.ZACS.FILTER.JOBNAME                          General
Data                            Current Allocation
  Management class . . : **None**       Allocated tracks  . : 5
  Storage class  . . . : **None**       Allocated extents . : 1
   Volume serial . . . : TEMP01
   Device type . . . . : 3380
  Data class . . . . . : **None**
   Organization  . . . : PS            Current Utilization
   Record format . . . : FB             Used tracks . . . . : 1
   Record length . . . : 133            Used extents  . . . : 1
   Block size  . . . . : 23408
   1st extent tracks . : 5
   Secondary tracks  . : 1             Dates
   Data set name type  :                Creation date . . . : 2020/01/22
   Data set encryption : NO             Referenced date . . : 2020/01/22
                                        Expiration date . . : ***None***    SMS
Compressible  . : NO
```

Figure 11. userid.ZACS.FILTER.JOBNAME

stOutDSname

The sequential data set to contain the found potential vulnerabilities. It must match the potential vulnerability data set specified by MYOUTDD in the started task. Data set must exist.

Example allocation:

```
alloc new da(<output-dataset>) dd(<myoutdd>) dsorg(ps) space(300,300) tracks lrecl(133)
blksize(0) recfm(f,b)
```

```
free dd(<myoutdd>)
```

Figure 12. Example allocation and deallocation of data set.

The suggested attributes are as follows:

```
Data Set Name . . . . : userid.ZACS.OUTPUT                                  General
Data                            Current Allocation
 Management class . . : **None**       Allocated tracks  . : 600
 Storage class  . . . : **None**       Allocated extents . : 2
  Volume serial . . . : TSOE01
```

```
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PS             Current Utilization
  Record format . . . : FB              Used tracks . . . . : 26
  Record length . . . : 133             Used extents  . . . : 1
  Block size  . . . . : 27930
  1st extent tracks . : 300
  Secondary tracks  . : 300            Dates
  Data set name type  :                 Creation date . . . : 2020/01/23
  Data set encryption : NO              Referenced date . . : 2020/01/27
                                        Expiration date . . : ***None***  SMS
Compressible  . : NO
```

Figure 13. userid.ZACS.OUTPUT

printStatus
The switch to determine if 'TEST IN PROGRESS' and 'TEST DONE' messages are to be printed to the potential vulnerability data set. If 'ALL' is specified, these messages will be printed to the potential vulnerability data set, always. If 'ERROR' is specified, 'TEST IN PROGRESS' messages will not be displayed, and 'TEST DONE' messages will be displayed only if a non-zero return code is detected. This option allows the user to suppress output from successful test cases. The default is 'ALL'.

printSummary
The switch to determine if summary information is to be displayed at the end of each service's tests in the potential vulnerability data set. If 'ALL' is specified, summary information is printed after testing completes for each service. If 'ERROR' is specified, summary information is only printed after testing completes for a service if a non-zero return code was detected. The default is 'ALL'.

svcLogDSname

The sequential data set to contain the log of the most recent SVC run. If the data set does not exist it will be created using the attributes specified by svcLogDSlrec, svcLogDSblksz, svcLogDSpriSp, svcLogDSsecSp. The suggested attributes are as follows:

```
Data Set Name . . . . : userid.ZACS.SVCLOG                              General
Data                          Current Allocation
 Management class . . : **None**       Allocated tracks  . : 5
 Storage class  . . . : **None**       Allocated extents . : 1
  Volume serial . . . : TEMP01
  Device type . . . . : 3380
 Data class . . . . . : **None**
  Organization  . . . : PS             Current Utilization
  Record format . . . : FB              Used tracks . . . . : 1
  Record length . . . : 133             Used extents  . . . : 1
  Block size  . . . . : 1330
  1st extent tracks . : 5
  Secondary tracks  . : 1              Dates
  Data set name type  :                 Creation date . . . : 2020/01/20
  Data set encryption : NO              Referenced date . . : 2020/01/21
                                        Expiration date . . : ***None***     SMS
Compressible  . : NO
```

Figure 14. userid.ZACS.SVCLOG

svcLogDSlrec
The record length to be used in allocating the svcLogDSname data set if it does not already exist.

svcLogDSblksz
The block size to be used in allocating the svcLogDSname data set if it does not already exist.

svcLogDSpriSp
The primary space quantity to be used in allocating the svcLogDSname data set if it does not already exist.

svcLogDSsecSp
The secondary space quantity to be used in allocating the svcLogDSname data set if it does not already exist.

pcLogDSname
The sequential data set to contain the log of the most recent PC run. If the data set does not exist it will be created using the attributes specified by pcLogDSlrec, pcLogDSblksz, pcLogDSpriSp, pcLogDSsecSp. The suggested attributes are as follows:

```
Data Set Name . . . . : userid.ZACS.PCLOG                                 General
Data                            Current Allocation
 Management class . . : **None**        Allocated tracks  . : 100
 Storage class  . . . : **None**        Allocated extents . : 1
  Volume serial . . . : TEMP01
  Device type . . . . : 3380
 Data class . . . . . : **None**
  Organization  . . . : PS              Current Utilization
  Record format . . . : FB               Used tracks . . . . : 3
  Record length . . . : 133              Used extents  . . . : 1
  Block size  . . . . : 1330
  1st extent tracks . : 100
  Secondary tracks  . : 40              Dates
  Data set name type  :                  Creation date . . . : 2020/01/20
  Data set encryption : NO               Referenced date . . : 2020/01/21
                                         Expiration date . . : ***None***         SMS
Compressible  . : NO
```

Figure 15. userid.ZACS.PCLOG

pcLogDSlrec
The record length to be used in allocating the pcLogDSname data set if it does not already exist.

pcLogDSblksz
The block size to be used in allocating the pcLogDSname data set if it does not already exist.

pcLogDSpriSp
The primary space quantity to be used in allocating the pcLogDSname data set if it does not already exist.

pcLogDSsecSp
The secondary space quantity to be used in allocating the pcLogDSname data set if it does not already exist.

Example Configuration File

Note: Full line comments proceeded by '/*' and ending with '*/' will be ignored.

```
/*********************************************************************/
/*********************************************************************/
/**                                                                 **/
/** zACS Configuration file                                         **/
/**                                                                 **/
/*********************************************************************/
/*********************************************************************/

/*********************************************************************/
/*                                                                   */
/* Runtime settings                                                  */
/*                                                                   */
/* Instructions:                                                     */
/* Configure whether you would like to clear the logrec database     */
/*   while running zACS, default is 'NO'.                            */
/* Specify your logrec data set name.                                */
/* Required Job Card Parameters:                                     */
/*   Choose the jobname you would like zACS tests to run under,      */
/*     default is 'ZACSJ'.                                           */
/*   Specify the MSGCLASS with which to submit testcase jobs, default*/
/*     is 'A'.                                                       */
/*   Specify the REGION with which to submit testcase jobs, default  */
/*     is '50' with unit 'M'.                                        */
/*   Specify the TIME with which to submit testcase jobs, default    */
/*     is '5'.                                                       */
/*   Specify the SYSTEM with which to submit testcase jobs, default  */
/*     is the special keyword 'DEFAULT' which will use the system on */
/*     which the job is submitted.                                   */
/* Specify 'LOG' to have the PC/SVC log data go to the log data sets */
/*   or 'SCREEN' to have it print to the screen, default is 'SCREEN'.*/
/*   Note: When using the panel, this value is ignored and output    */
```

```
/*   will go to the log.                                              */
/* Specify the dataset containing the JCL to generate the PC and     */
/*   SVC tables.                                                      */
/* Specify the dataset containing the PC table, '&userid' can be     */
/*   used to substitute your userid                                   */
/* Specify the dataset containing the SVC table, '&userid' can be    */
/*   used to substitute your userid                                   */
/*                                                                    */
/**********************************************************************/
clearLogrec         = 'NO'
logrecDSname        = 'SYS1.LOGREC'
testcaseJobName     = 'ZACSJ'
jobMsgClass         = 'A'
jobRegion           = '50M'
jobTimeout          = '5'
jobSystem           = 'DEFAULT'
printLogOutput      = 'SCREEN'
jclDSname           = 'jcl-dataset'
pcTableDSname       = '&userid.ZACS.PCNUM'
svcTableDSname      = '&userid.ZACS.SVCNUM'

/**********************************************************************/
/*                                                                    */
/* Optional Job Card Parameters                                       */
/*                                                                    */
/* Instructions:                                                      */
/* Configure the optional job card parameters to include on the      */
/*   generated zACS jobs:                                             */
/*   Configure whether you would like to include an account number,  */
/*     default is 'NO'.                                               */
/*   Specify your account number. Escape single quotes by doubling   */
/*      them. EX: A,''1/2/3'',B for A,'1/2/3',B                       */
/*   Configure whether you would like to include programmer name,     */
/*     default is 'NO'.                                               */
/*   Specify the programmers name. Escape single quotes by doubling  */
/*     them. EX: T.O''NEILL for T.O'NEILL                             */
/*   Configure whether you would like to include any other keyword    */
/*     parameters, default is 'NO'.                                   */
/*   Specify any other job card parameters you would like to include,*/
/*     if specifying other keywords, proper job card syntax must be  */
/*     used. EX: 'NOTIFY=ZACSUSER,PRTY=12'                            */
/*                                                                    */
/**********************************************************************/
useJobAccount       = 'NO'
jobAccount          = '<account-number>'
useJobProgrammer    = 'NO'
jobProgrammer       = '<programmer-name>'
useOtherKeywords    = 'NO'
jobOtherKeywords    = '<other-parameters>'

/**********************************************************************/
/*                                                                    */
/* Filter settings                                                    */
/*                                                                    */
/* Instructions:                                                      */
/* Specify 'EXCLUDE' to exclude modules and/or jobnames during your  */
/*   run or 'INCLUDE' to run a subset of modules and/or jobnames     */
/*   during your run. This value is ignored if both filterByModName  */
/*   and filterByJobname are 'NO'. Default is 'EXCLUDE'.              */
/* Change 'module-name-filter-list-dataset' to the name of the       */
/*   data set containing the list of modules you would like to       */
/*   include or exclude. Configure whether or not you would like to  */
/*   include or exclude modules during your run, default is 'NO'.    */
/* Change 'jobname-filter-list-dataset' to the name of the data set  */
/*   containing the list of jobnames you would like to include or    */
/*   exclude. Configure whether or not you would like to include or  */
/*   exclude jobnames during your run, default is 'NO'.              */
/*                                                                    */
/**********************************************************************/
includeOrExclude    = 'EXCLUDE'
filterByModName     = 'NO'
modFilterDSname     = 'module-name-filter-list-dataset'
filterByJobname     = 'NO'
jobFilterDSname     = 'jobname-filter-list-dataset'

/**********************************************************************/
/*                                                                    */
/* Vulnerability log setting                                          */
/*                                                                    */
/* Instructions:                                                      */
/* Change 'output-dataset' to the name of the data set to which you  */
/*   would like to send detected potential vulnerability output.     */
```

```
/*   This must match the data set name specified by MYOUTDD in the   */
/*   started task.                                                    */
/* Specify 'ERROR' to suppress test in progress messages printed to  */
/*   the potential vulnerability log and to print return codes only  */
/*   when an unsuccessful result is detected. Specify 'ALL' to print */
/*   test in progress messages and to print return codes to the      */
/*   potential vulnerability log always. Default is 'ALL'            */
/* Specify 'ERROR' to suppress the summary information printed at    */
/*   the end of each service in the potential vulnerability log when */
/*   a successful result is detected, or 'ALL' to print summary      */
/*   information always. Default is 'ALL'.                           */
/*                                                                   */
/*********************************************************************/
stOutDSname         = 'output-dataset'
printStatus         = 'ALL'
printSummary        = 'ALL'

/*********************************************************************/
/*                                                                   */
/* SVC log settings                                                  */
/*                                                                   */
/* Instructions:                                                     */
/* Change 'svc-log-dataset' to the name of the data set to which you */
/*   would like to send run SVCs log information.                    */
/*                                                                   */
/*********************************************************************/
svcLogDSname        = 'svc-log-dataset'
svcLogDSlrec        = '133'
svcLogDSblksz       = '1330'
svcLogDSpriSp       = '5'
svcLogDSsecSp       = '1'

/*********************************************************************/
/*                                                                   */
/* PC log settings                                                   */
/*                                                                   */
/* Instructions:                                                     */
/* Change 'pc-log-dataset' to the name of the data set to which you  */
/*   would like to send run PCs log information.                     */
/*                                                                   */
/*********************************************************************/
pcLogDSname         = 'pc-log-dataset'
pcLogDSlrec         = '133'
pcLogDSblksz        = '1330'
pcLogDSpriSp        = '100'
pcLogDSsecSp        = '40'

/*********************************************************************/
/*********************************************************************/
/**                                                                 **/
/** End zACS Configuration file                                     **/
/**                                                                 **/
/*********************************************************************/
/*********************************************************************/
```

Figure 16. Example Configuration file.

Chapter 4. Running

This chapter assumes a high level qualifier of 'SYS1.BPN.**'.

Note: Changes made to the configuration file or filter lists while a run is in progress will not be reflected until the subsequent run.

Setup

Start the tool with your started task using the console:

```
S BPNZACS
```

Warning: Starting this task will clear the output data set defined in the started task, including any vulnerability data that was found.

The tool sets the following slip when the task is started.

```
SLIP SET,ID=BPN1,ERRTYP=PROG,A=(RECORD,NODUMP),END
```

Note: This slip may override other slips set on the system.

During a test run, zACS intentionally generates many ABENDs. This slip will suppress all dump generation while the zACS task is started, unless a slip is set afterwards, overriding it. The slip will be removed when the task is terminated with the following command:

```
P BPNZACS
```

Figure 17. Starting tool, setting slip trap, and purging tool.

Note: When shutting down zACS, an IEF352I message may be seen, this message may be ignored.

Before testing PCs, run BPNPCNUM to generate a PC table in '*userid*.ZACS.PCNUM'. As PC numbers can change, this job should be run regularly to ensure the table is up to date:

```
submit SYS1.BPN.SBPNSAMP(BPNPCNUM)
```

Before testing SVCs, run BPNSVCNM to generate SVC entries in '*userid*.ZACS.SVCNUM'. The job to generate the SVC table only needs to run once for every IPL or any time the SVC table or ESR SVCs are updated.

```
submit SYS1.BPN.SBPNSAMP(BPNSVCNM)
```

Figure 18. Generate PC table and SVC entries.

Running using the Command Line

zACS can be run via the command line by executing the BPNKNSVC and BPNKNPCX REXX executables to run SVCs and PCs respectively.

Note: Custom modifications to the provided REXX are not supported.

To test all SVCs in the generated SVC table, run:

```
ex 'SYS1.BPN.SBPNEXEC(BPNKNSVC)'
```

To test all PCs in the generated PC table, run:

```
ex 'SYS1.BPN.SBPNEXEC(BPNKNPCX)'
```

Figure 19. Test SVC and PC tables.

Note: When kicking off a full run, potentially hundreds of JCL jobs will be submitted. Jobs from any subsequent runs will be queued up behind the initial run if it is not yet complete. It is strongly advised to wait until all jobs have completed before starting a new run. To limit the number of SVC or PCs run see "Filtering Run Services with Include Lists or Exclude Lists" on page 26. It is also strongly advised to not submit the JCL test jobs directly, bypassing the REXX execs.

Optional Parameters

To further limit the number of services tested in a run, optional parameters may be used when running the REXX executables.

Note: The exclude lists described in "Filtering Run Services with Include Lists or Exclude Lists" on page 26 will be ignored when optional parameters are specified.

- BPNKNSVC:
  - `ex 'SYS1.BPN.SBPNEXEC(BPNKNSVC)' '<svc#>,<esr#>,<module_name>'`
  - Single SVC number - Used to run just a single SVC, specified as hexadecimal. If ESR Routing number is also specified, then only the hexadecimal values 6D, 74, 7A or 89 are accepted.
  - ESR Routing Number - Routing number to run a single Extended SVC, the single SVC number must also be specified in hexadecimal.
  - Module name - Used to run all SVCs for a single module.
- BPNKNPCX:
  - `ex 'SYS1.BPN.SBPNEXEC(BPNKNPCX)' '<pc#>,<sequence#>,<module_name>'`
  - Single PC number - Used to run just a single PC, specified in hexadecimal.
  - Sequence number - Sequence number for a PC, single PC number must also be specified in hexadecimal. If a sequence number is not specified, the default is 0.
  - Module name - Used to run all PCs for a single module.

Note: When running with optional parameters, a module name may not be specified if a single PC or SVC number is already specified.

For example, run only PCs in module IEAVH607:

```
ex 'SYS1.BPN.SBPNEXEC(BPNKNPCX)' ',,IEAVH607'
```

Similarly, run only ESR 26 with routing number 109:

```
ex 'SYS1.BPN.SBPNEXEC(BPNKNSVC)' '6D,1A,'
```

Figure 20. Example PC and SVC run.

Running using the Panel

To start the panel, run the following command from the command line:

```
ex 'SYS1.BPN.SBPNEXEC(BPNISPFR)'
```

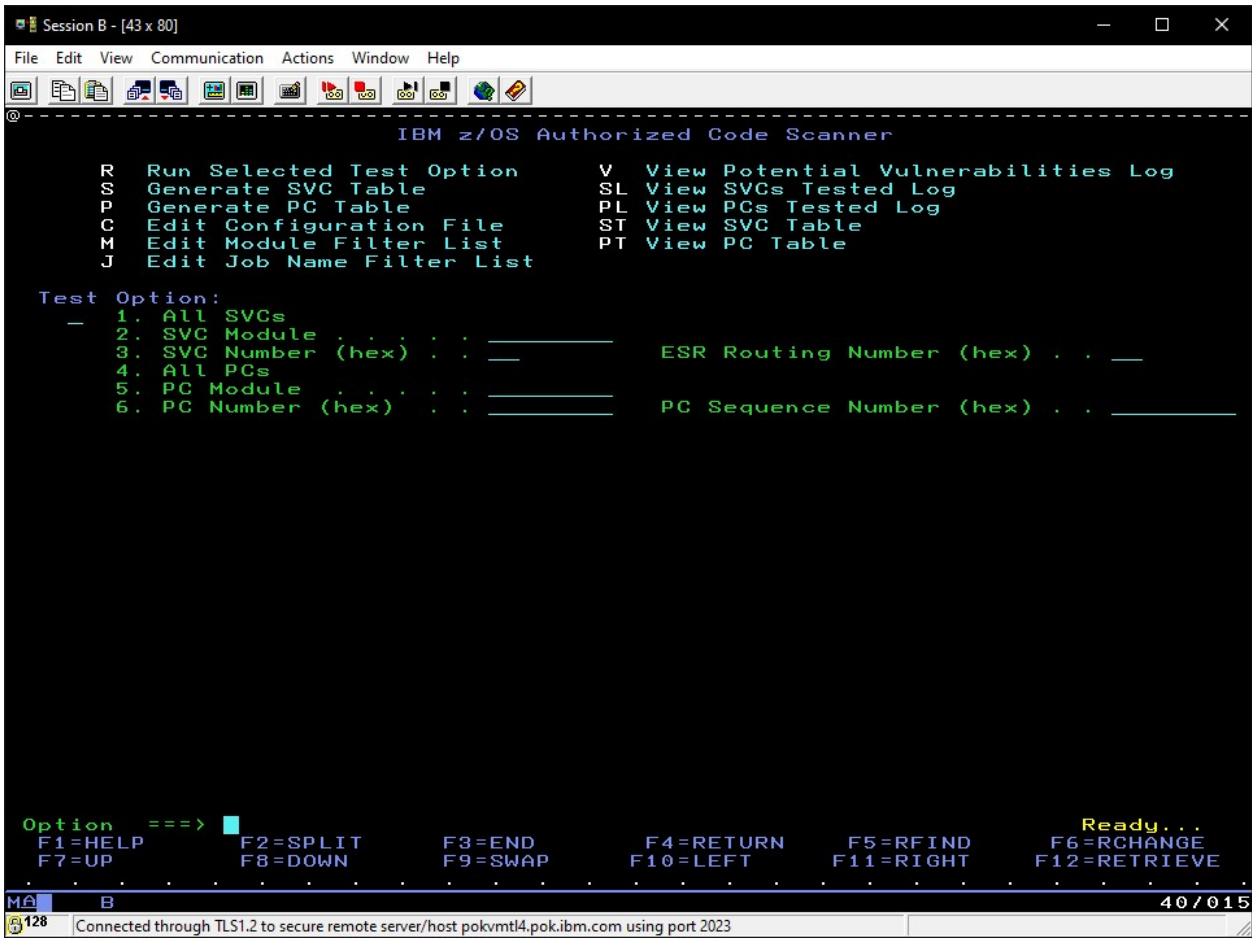

Figure 21. zACS Panel

zACS can be run via the panel by specifying the numbers 1-6 in the Test Option field, and the run option ‘R’ in the option field.

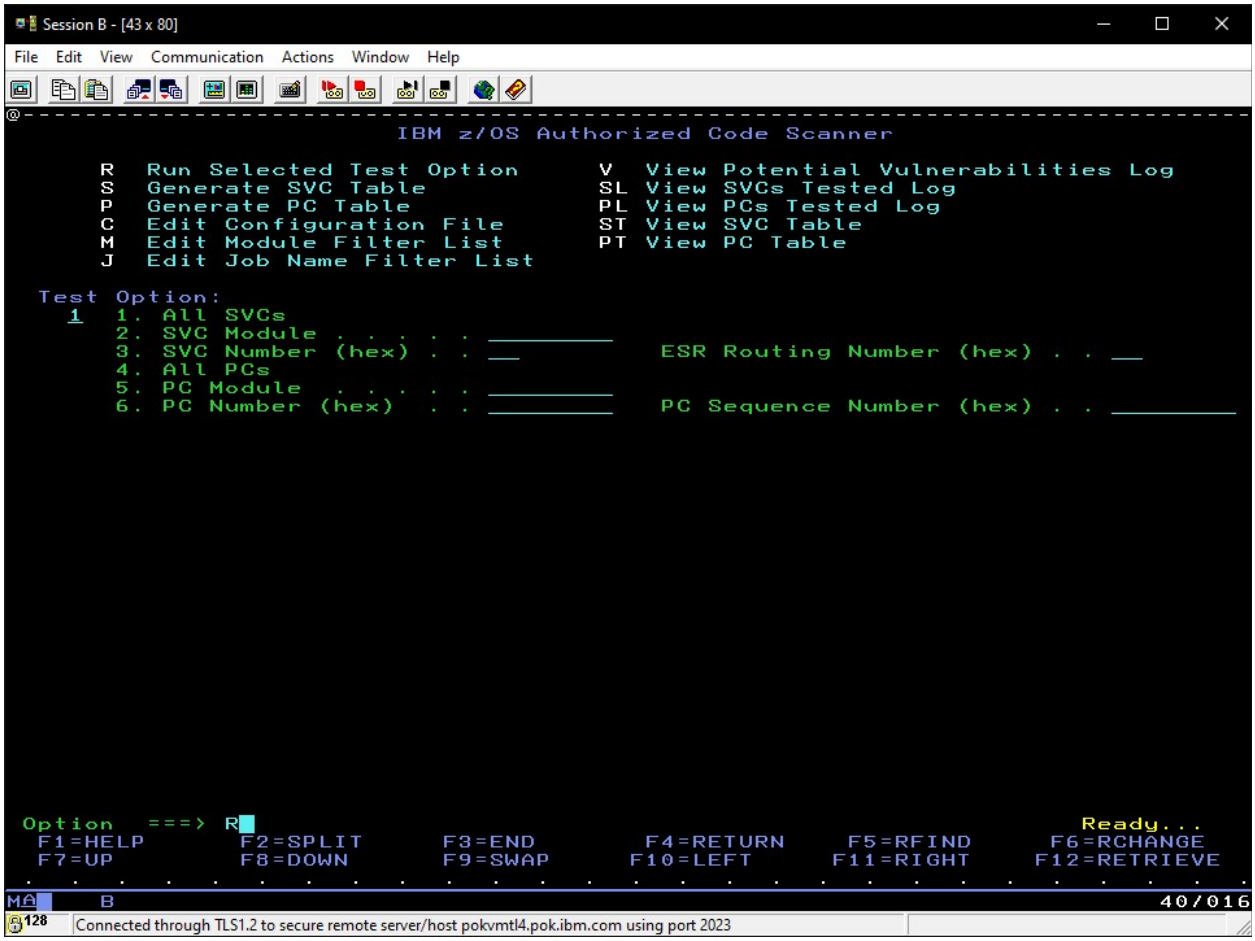

Figure 22. Running all SVCs from the panel.

To test all SVCs in the generated SVC table, specify test option 1 in the Test Option field, specify option ‘R’ in the Option field, and press enter.

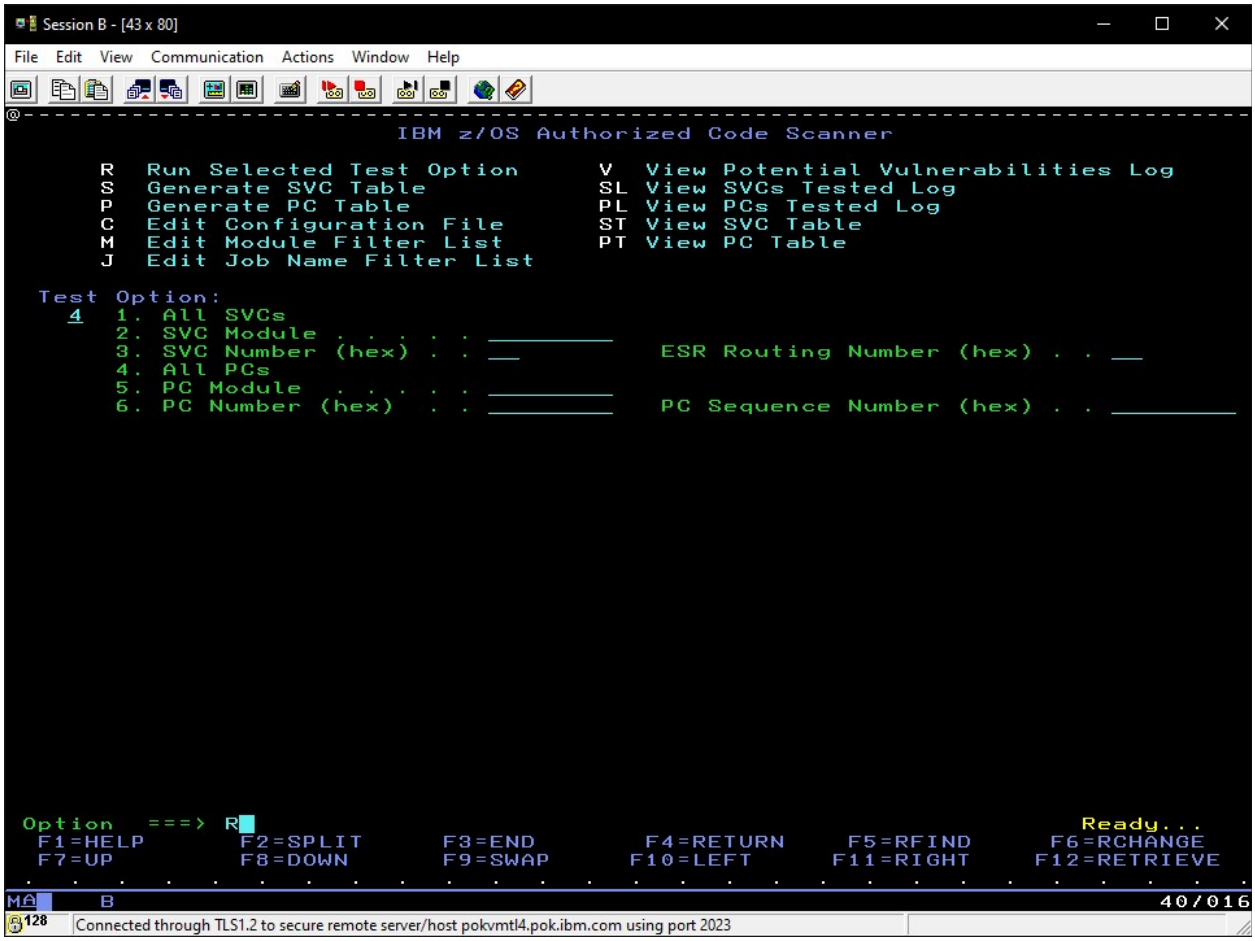

Figure 23. Running all PCs from the panel.

To test all PCs in the generated PC table, specify test option 4 in the Test Option field, specify option ‘R’ in the Option field, and press enter.

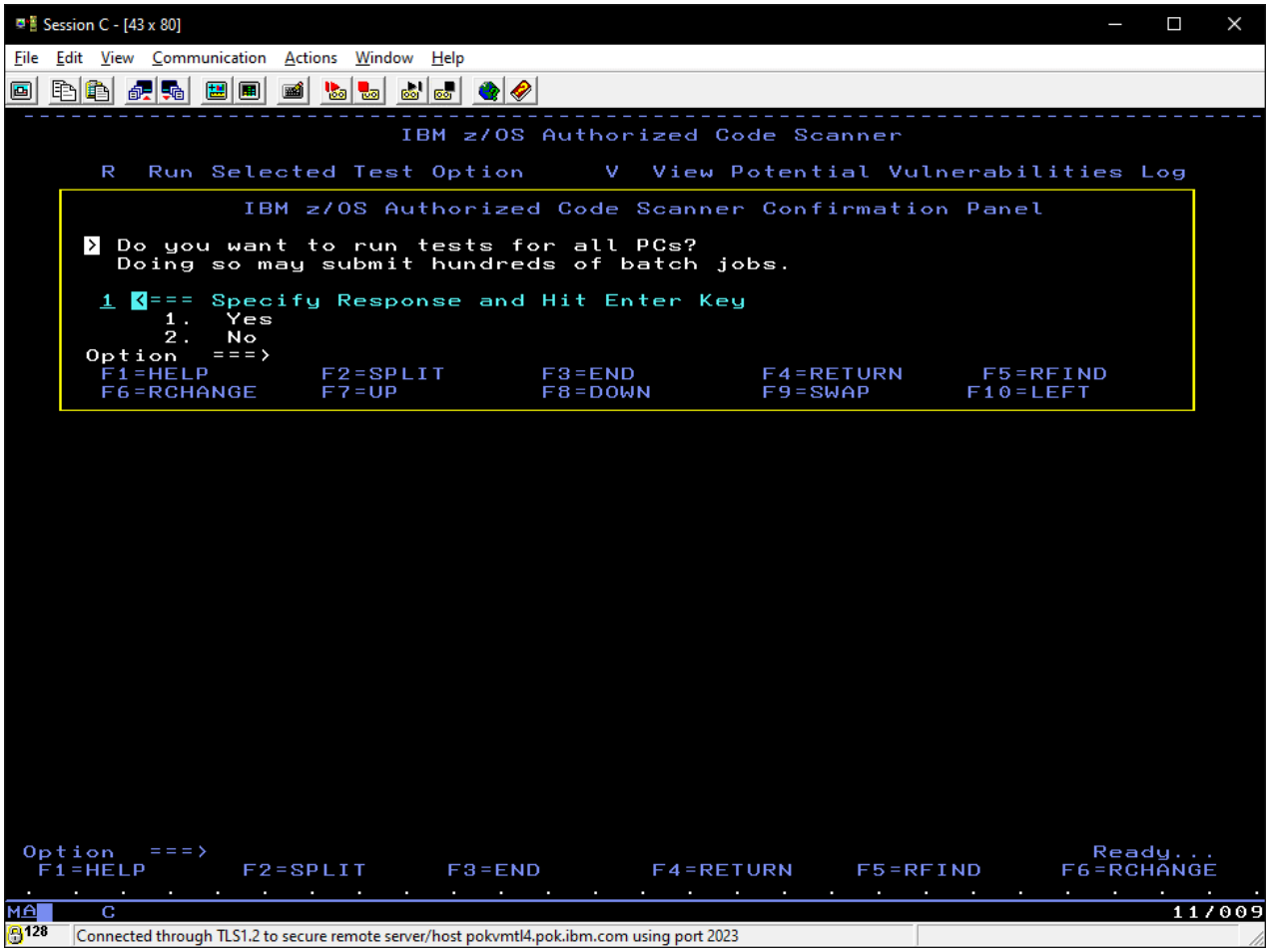

Figure 24. Confirmation pop-up for All PCs run.

When kicking off a full run, potentially hundreds of JCL jobs will be submitted. When the ‘R’ option is selected with test option 1 or 4, a confirmation window will pop up to prevent accidentally starting a large run. To continue with the run, select 1 and press enter, to prevent the run from starting select 2 and press enter.

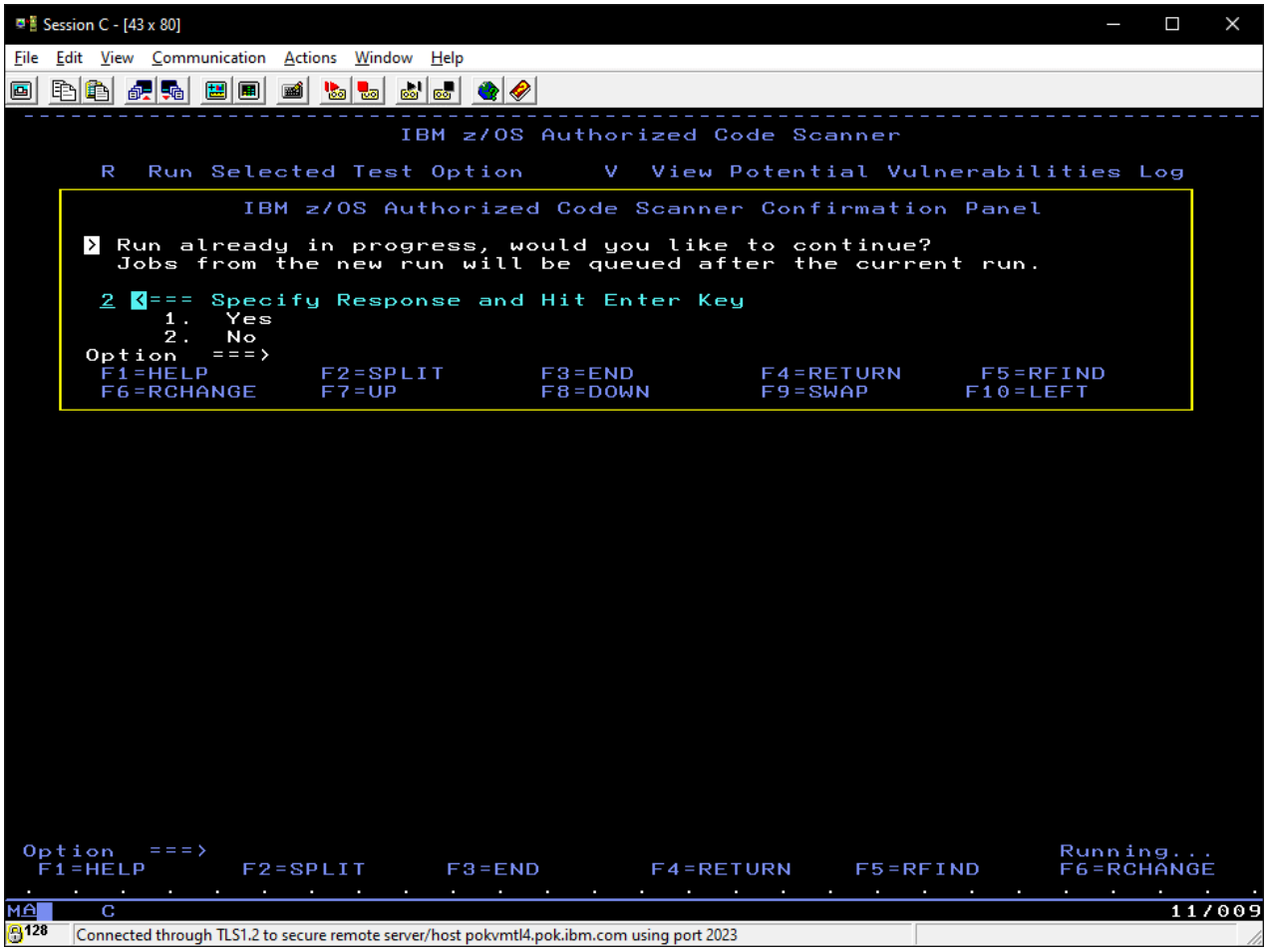

Figure 25. Confirmation pop-up for run already in progress.

Jobs from any subsequent runs will be queued up behind the initial run if it is not yet complete. It is strongly advised to wait until all jobs have completed before starting a new run. If a run is not yet complete, a confirmation window will pop up to prevent accidentally starting a new run. To continue with the run, select 1 and press enter, to prevent the run from starting select 2 and press enter.

Note: To limit the number of SVC or PCs run see “Filtering Run Services with Include Lists or Exclude Lists” on page 26.

Optional Parameters

To further limit the number of services tested in a run, optional parameters may be used when running the from the panels by specifying test options 2, 3, 5, or 6 in the Test Option field, filling in the necessary parameter fields, specifying option ‘R’ in the Option field, and pressing enter.

Note: The exclude lists described in “Filtering Run Services with Include Lists or Exclude Lists” on page 26 will be ignored when optional parameters are specified.

Test Option 2

Run all SVCs for a single module. Uses the following parameter field:

Module Name (Required)
The name of the module for which all SVCs are to be run.

Test Option 3
Run a single SVC. Uses the following parameter fields:

SVC Number (Required)
The hexadecimal SVC number to be run. If ESR Routing number is also specified, then only the hexadecimal values 6D, 74, 7A or 89 are accepted.

ESR Routing Number (Optional)
The hexadecimal routing number to run a single Extended SVC.

Test Option 5
Run all PCs for a single module. Uses the following parameter field:

Module Name (Required)
The name of the module for which all PCs are to be run.

Test Option 6
Run a single PC. Uses the following parameter fields:

PC Number (Required)
The hexadecimal PC number to be run.

PC Sequence Number (Optional)
The hexadecimal sequence number of the PC number to be run. If not specified, the default is 0.

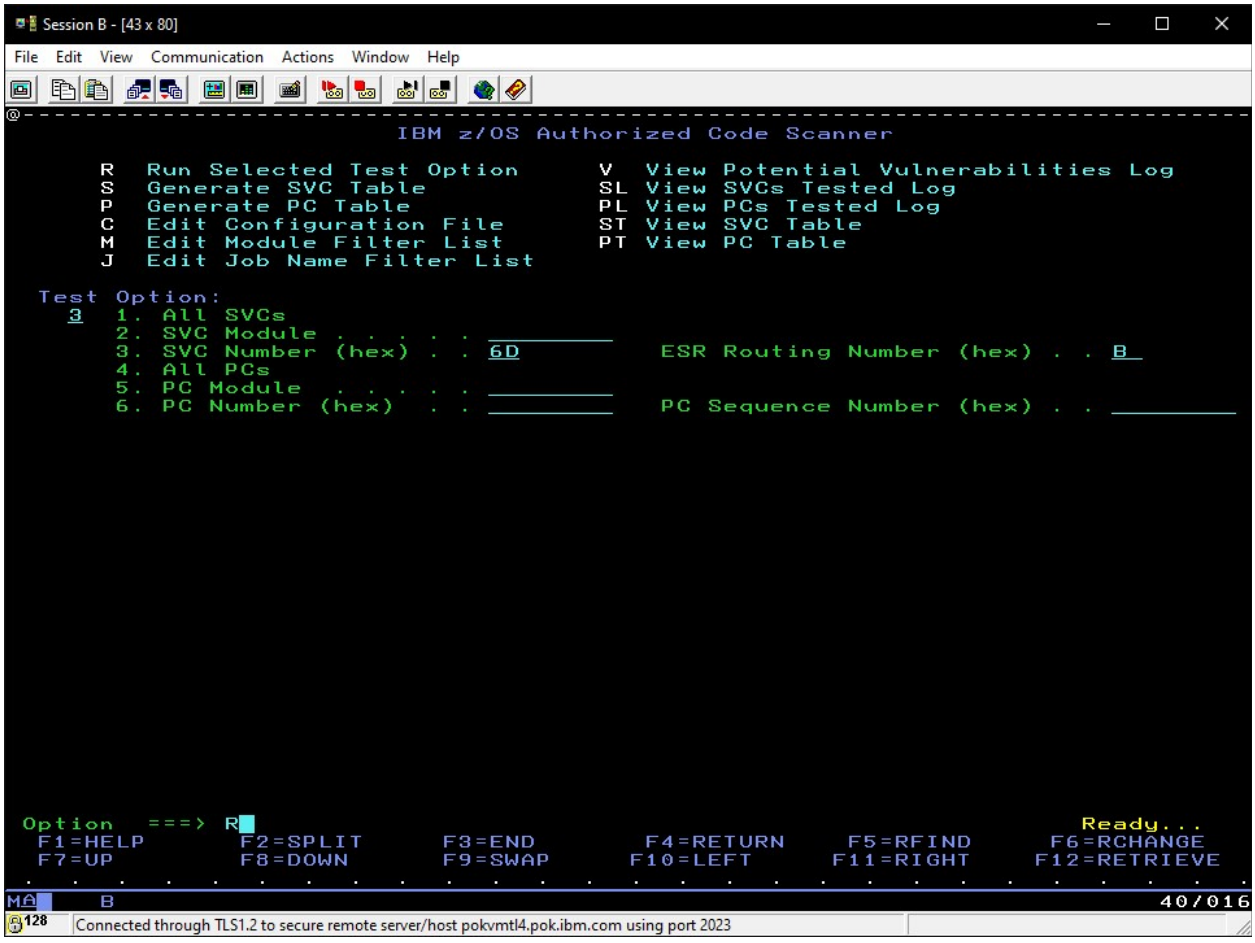

Figure 26. Running a single SVC from the panel with routing number 109 and ESR 11.

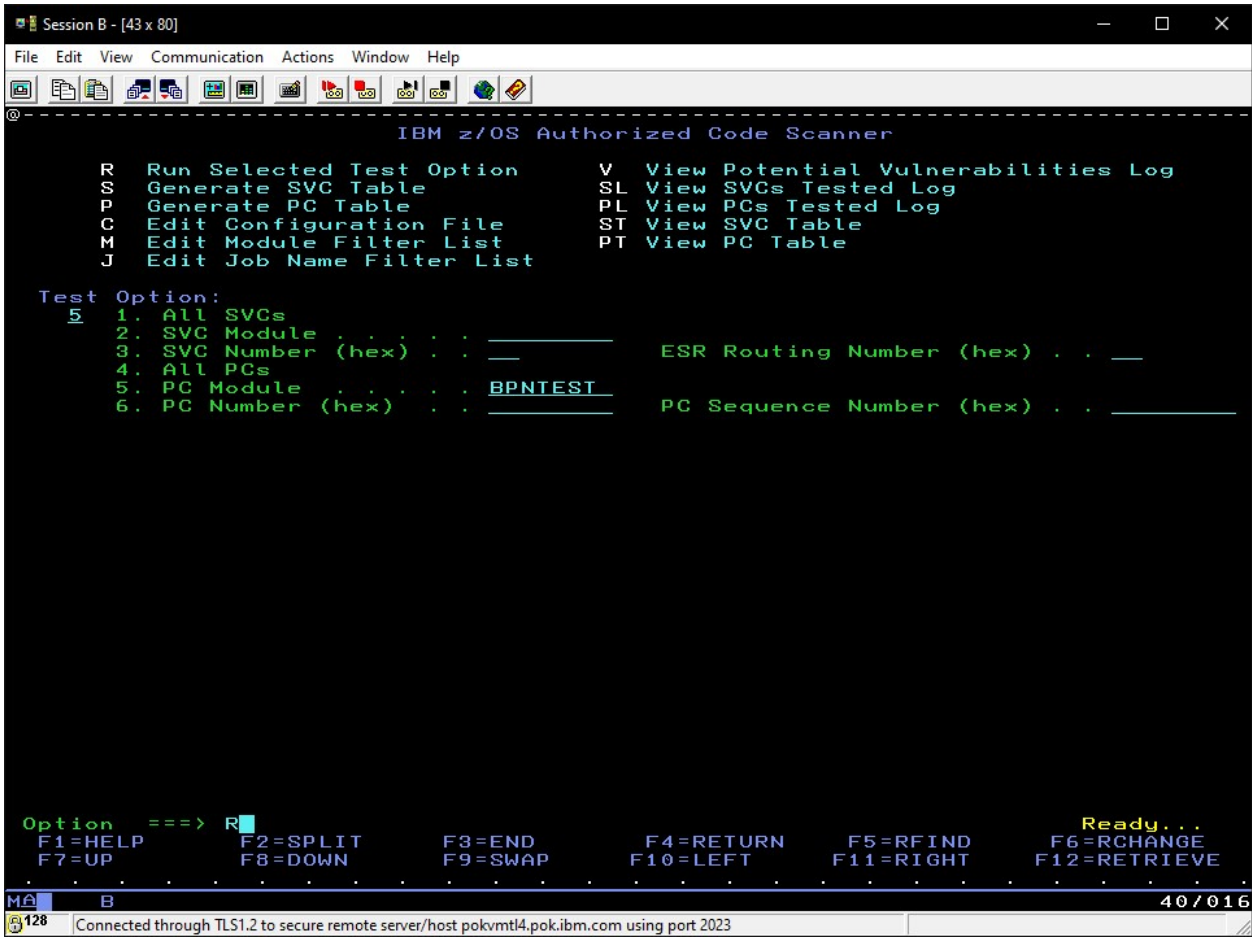

Figure 27. Running PCs by module name from the panel with module name BPNTEST.

Additional Panel Options

In addition to running zACS tests, other actions may be taken from the zACS Panel by specifying its respective option in the option field and pressing enter.

Option S

Run the JCL, the containing data set of which is specified in the configuration file, to generate the SVC Table.

Option P

Run the JCL, the containing data set of which is specified in the configuration file, to generate the PC Table.

Option C

Opens the configuration file, located at ‘userid.ZACS.CONFIG’, in edit mode, if it exists.

Option M

Opens the exclusion/inclusion by module name list, as specified by the keyword filterByModName in the configuration file, in edit mode, if it exists.

Option J

Opens the exclusion/inclusion by job name list, as specified by the keyword filterByJobName in the configuration file, in edit mode, if it exists.

Option V

Opens the vulnerabilities log, as specified by the keyword stOutDSname in the configuration file, in browse mode, if it exists.

Option SL
Opens the SVC log, as specified by the keyword svcLogDSname in the configuration file, in browse mode, if it exists.

Option PL
Opens the PC log, as specified by the keyword pcLogDSname in the configuration file, in browse mode, if it exists.

Option ST
Opens the SVC table, as specified by the keyword svcTableDSname in the configuration file, in browse mode, if it exists.

Option PT
Opens the PC table, as specified by the keyword pcTableDSname in the configuration file, in browse mode, if it exists.

Note: When using the panel, the option to send PC or SVC log output to the screen is disabled. Log output will be sent to the log data sets specified in the configuration file even if 'SCREEN' is specified for printLogOutput.

Filtering Run Services with Include Lists or Exclude Lists

During a full run, services may be excluded or included based on module name or job name. In the configuration file, modFilterDSname and jobFilterDSname point to the Filter by Module Name List and Filter by Job Name List respectively. Note that only PCs can be filtered by job name. The lists are formatted as one module name or job name per line. Blank lines and line comments surrounded with '/*' and '*/' will be ignored.

Note: The exclude and include lists determine which modules are individually tested or skipped. If a module in the exclude list is called by a separate module not on the exclude list, it may still appear in the output file if a potential vulnerability within it is hit during the calling module's test.

Exclude by module example, runs all services except those with the module names listed:

```
/*********************/
/* Excluded Modules */
/*********************/
IGVSLIST
IGVLOCP
IEAVRT04
ITVDIV
ASRSERVP
IEAVLSEX
IEAVESTA
IGVVSTOR
```

Figure 28. Example of excluded module.

Include by job name example, runs only services with the job names listed:

```

/***********************/
/* Included Job Names */
/***********************/
PCAUTH
GRS
IOSAS
```

Figure 29. Example of included module.

Locating Module Names and Job Names

Module names can be found in the 13th column of the '*userid*.ZACS.SVCNUM' file or the 9th column of the '*userid*.ZACS.PCNUM' file.

```
SVC   00  -- 31  1 N N N N N YNNNN 00FF1A70 IECVEXCP 80FF19E0 00000090
SVC   01  -- 31  1 N N N N N YNNNN 00FEF158 IEAVEWAT 80FEEF10 00000248
SVC   02  -- 31  1 N N N N N YNNNN 0133CC18 IEAVEPST 8133CC00 00000018
SVC   03  -- 31  1 N N N N Y YNNNN 01337AA0 IGC003   81337AA0 00000000
```

Figure 30. Module names in 13th column of userid.ZACS.SVCNUM

```
PCNUM 00000122 0000 0007 06FFD630 S 0 00000000 ISGNLPA  06FFD000 00000630 GRS
PCNUM 00000123 0000 0007 08F3C958 S 0 00000000 ISGNPVT  08F00000 0003C958 GRS
PCNUM 00000200 C000 0012 08F07710 S 0 00000000 IEFHB410 08F07710 00000000 ALLOCA
PCNUM 00000201 C000 0012 08F07990 S 0 00000000 IEFHB420 08F07990 00000000 ALLOCA
```

Figure 31. Module names in 9th column of userid.ZACS.PCNUM

Job names can be found in the 12th column of the '*userid*.ZACS.PCNUM' file.

```
PCNUM 00000122 0000 0007 06FFD630 S 0 00000000 ISGNLPA  06FFD000 00000630 GRS
PCNUM 00000123 0000 0007 08F3C958 S 0 00000000 ISGNPVT  08F00000 0003C958 GRS
PCNUM 00000200 C000 0012 08F07710 S 0 00000000 IEFHB410 08F07710 00000000 ALLOCA
PCNUM 00000201 C000 0012 08F07990 S 0 00000000 IEFHB420 08F07990 00000000 ALLOCA
```

Figure 32. Job names in 12th column of userid.ZACS.PCNUM

Note: Not all PCs and SVCs have associated module names.

Chapter 5. Diagnosis

After your test run is complete, you can review the results in the output file you specified for the MYOUTDD statement in the started task.

Note: Multiple tests per service are common.

This chapter provides examples of zACS output and explains how to interpret the results. Examples of code vulnerabilities are provided with instructions explaining how to fix them. Instructions for setting a SLIP to collect a dump for further diagnosis are also provided.

Interpreting the Potential Vulnerability Output File

Note: The following potential vulnerabilities were deliberately injected to help illustrate zACS output. The example below has a number inserted before each line for reference purposes, that do not actually appear in the generated output.

```
01 *** POTENTIAL VULNERABILITY FOUND IN PC 00180600 ***
02 ABEND COMPLETION CODE: 0C4000  REASON CODE: 00000011
03 PSW: 070C6000 88F101B2  MODULE: PVTMOD=(BPNTEST,000011B2)
04 INSTR LEN: 06  FAILING INSTR: B048 9A33 B04C D203 2000 3000
05 TRANSLATED INSTR: MVC     0(4,R2),0(R3)
06 TARGET ADDRESS CAUSING TRANSLATION EXCEPTION: 00000000_7FFFF401
07 HOME ASID: 0029 PRIMARY ASID: 001A SECONDARY ASID: 0029
08 CVSS: 8.8 (CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
09 SLIP SAMPLE FOR PC 00180600:
10 SLIP SET,COMP=0C4,P=(BPNTEST,000011B2),SDATA=(trt,rgn,sum,csa),end
11 +-----------------------------------------------+
12 |General Registers before the service           |
13 +-----------------------------------------------+
14 |  R0:00000000_7FFFF7FF   R1:00000000_00048000  |
15 |  R2:00000000_7FFFF7FF   R3:00000000_7FFFF7FF  |
16 |  R4:00000000_7FFFF7FF   R5:00000000_7FFFF7FF  |
17 |  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
18 |  R8:FFFFFFFF_00000068   R9:D7BFC9E8_005C6E00  |
19 |  RA:FFFFFFFF_005C6E08   RB:00000000_0001E0B0  |
20 |  RC:00000000_00034CB0   RD:00000000_7FFFF7FF  |
21 |  RE:00000000_7FFFF7FF   RF:00000000_7FFFF7FF  |
22 +-----------------------------------------------+
23 |General Registers at time of error             |
24 +-----------------------------------------------+
25 |  R0:00000000_00000000   R1:00000000_00048000  |
26 |  R2:00000000_7FFFF7FF   R3:00000000_08F57AF8  |
27 |  R4:00000000_02460360   R5:00000000_7FFFF7FF  |
28 |  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
29 |  R8:FFFFFFFF_00000068   R9:D7BFC9E8_00FC36A8  |
30 |  RA:FFFFFFFF_88F10158   RB:00000000_08F57AA8  |
31 |  RC:00000000_08F101F8   RD:00000000_08F57AA8  |
32 |  RE:00000000_7FFFF7FF   RF:00000000_00000002  |
33 +-----------------------------------------------+
34 |Access Registers at time of error              |
35 +-----------------------------------------------+
36 |R0:00000000 R1:00000000 R2:00000002 R3:00000000|
37 |R4:FFFFFFFF R5:FFFFFFFF R6:FFFFFFFF R7:FFFFFFFF|
38 |R8:FFFFFFFF R9:FFFFFFFF RA:00000000 RB:00000000|
39 |RC:00000000 RD:00000000 RE:00000000 RF:00000000|
40 +-----------------------------------------------+
41 *** END OF POTENTIAL VULNERABILITY REPORT ***
```

Figure 33. Potential Vulnerability Output Example 0C4

Line 01
Indicates the service number where the potential vulnerability was detected.

Line 02
Indicates the six-digit ABEND code, which allows for user ABENDs when applicable. The line also includes the ABEND's reason code.

Line 03
Indicates the failing Program Status Word (PSW) as well as the name of the module where the potential vulnerability was found, if known, and the offset where the ABEND occurred.

Line 04
Indicates the failing instruction and the instructions length from the System Diagnostic Work Area (SDWA). Information on the SDWA can be found in the SDWA Mapping section of z/OS MVS Data Areas Volume 4 (RRP - XTL). Information on the op code can be found in IBM z/Architecture Principles of Operation.

Line 05
Indicates the failing instruction translated into assembler format.

Line 06
Indicates the source or target address that caused the translation exception.

Line 07
The home, primary, and secondary address space identifiers.

Line 08
Indicates the probable CVSS score and vector of the potential vulnerability. CVSS scores are calculated using CVSS version 3.0. More information can be found here https://www.first.org/cvss/calculator/3.0.

Line 09
Indicates the header for the sample slip command.

Line 10
Provides an example slip command when one can be automatically generated.

Lines 14-21
Indicates the contents of the general purpose registers immediately before the service was invoked.

Lines 25-32
Indicates the contents of the general purpose registers at the time of error.

Lines 36-39
Indicates the contents of the access purpose registers at the time of error. These lines only appear in AR mode.

Additional output is provided if an overlay is detected.

```
01 *** POTENTIAL VULNERABILITY FOUND IN PC 00180607 ***
02 ABEND COMPLETION CODE: 0C1000  REASON CODE: 00000001
03 PSW: 070C2000 80000012
04 INSTR LEN: 02  FAILING INSTR: 5040 4143 00FD 5078 5040 4143
05 BREAKING EVENT ADDR: 08F10774  MODULE: PVTMOD=(BPNTEST,00001774)
06 BREAKING EVENT INSTR: 0513
07 TRANSLATED INSTR: BALR    R1,R3
08 HOME ASID: 0029 PRIMARY ASID: 001A SECONDARY ASID: 0029
09 CVSS: 8.8 (CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
10 SLIP SAMPLE FOR PC 00180607:
11 SLIP SET,IF,RANGE=(08F10774),SDATA=(trt,rgn,lpa,nuc,sum,csa),end
12 +-----------------------------------------------+
13 |General Registers before the service           |
14 +-----------------------------------------------+
15 |  R0:00000000_7FFFF7FF   R1:00000000_00061000  |
16 |  R2:00000000_7FFFF7FF   R3:00000000_7FFFF7FF  |
17 |  R4:00000000_7FFFF7FF   R5:00000000_7FFFF7FF  |
18 |  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
19 |  R8:FFFFFFFF_00000068   R9:D7BFCA00_005C6E00  |
20 |  RA:FFFFFFFF_005C6E08   RB:00000000_000370B0  |
21 |  RC:00000000_0004DCB0   RD:00000000_7FFFF7FF  |
22 |  RE:00000000_7FFFF7FF   RF:00000000_7FFFF7FF  |
23 +-----------------------------------------------+
24 |General Registers at time of error             |
25 +-----------------------------------------------+
26 |  R0:00000000_7FFFF7FF   R1:00000000_88F10776  |
27 |  R2:00000000_7FFFF7FF   R3:00000000_00000000  |
28 |  R4:00000000_024603D0   R5:00000000_7FFFF7FF  |
29 |  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
30 |  R8:FFFFFFFF_00000068   R9:D7BFCA00_00FC36A8  |
31 |  RA:FFFFFFFF_88F1073C   RB:00000000_08F5B598  |
32 |  RC:00000000_08F107A8   RD:00000000_08F5B598  |
33 |  RE:00000000_7FFFF7FF   RF:00000000_7FFFF7FF  |
34 +-----------------------------------------------+
35 *** END OF POTENTIAL VULNERABILITY REPORT ***
```

Figure 34. Potential Vulnerability Output Example 0C1.

Line 01
Indicates the service number where the potential vulnerability was detected.

Line 02
Indicates the six-digit ABEND code, which allows for user ABENDs when applicable. The line also includes the ABEND's reason code.

Line 03
Indicates the failing Program Status Word (PSW).

Line 04
Indicates the failing instruction and the instructions length from the System Diagnostic Work Area (SDWA). Information on the SDWA can be found in the SDWA Mapping section of z/OS MVS Data Areas Volume 4 (RRP - XTL). Information on the op code can be found in IBM z/Architecture Principles of Operation.

Line 05
Indicates the breaking event address as well as the name of the module where the potential vulnerability was found, if known, and the offset where the ABEND occurred.

Line 06
Indicates instruction at the breaking event address.

Line 07
Indicates the breaking event address translated into assembler format.

Line 08
The home, primary and secondary address space identifiers.

Line 09
Indicates the probable CVSS score and vector of the potential vulnerability. CVSS scores are calculated using CVSS version 3.0. More information can be found here https://www.first.org/cvss/calculator/3.0.

Line 10

Indicates the header for the sample slip command

Line 11

Provides an example slip command when one can be automatically generated.

Lines 15-22

Indicates the contents of the general purpose registers immediately before the service was invoked.

Lines 26-33

Indicates the contents of the general purpose registers at the time of error.

```
01 *** POTENTIAL VULNERABILITY FOUND IN PC 00180609 ***
02 ABEND COMPLETION CODE: 0E0000  REASON CODE: 00000029
03 PSW: 070C6000 88F108FE  MODULE: PVTMOD=(BPNTEST,000018FE)
04 INSTR LEN: 06  FAILING INSTR: B048 9A33 B04C D203 2000 3000
05 TRANSLATED INSTR: MVC     0(4,R2),0(R3)
06 ACCESS REGISTER CAUSING TRANSLATION EXCEPTION: 03
07 HOME ASID: 0029 PRIMARY ASID: 001A SECONDARY ASID: 0029
08 CVSS SCORE CANNOT BE DETERMINED FROM AVAILABLE INFORMATION
09 SLIP SAMPLE FOR PC 00180609:
10 SLIP SET,COMP=0E0,P=(BPNTEST,000018FE),SDATA=(trt,rgn,sum,csa),end
11 +-----------------------------------------------+
12 |General Registers before the service           |
13 +-----------------------------------------------+
14 |  R0:00000000_7FFFF7FF   R1:00000000_0003A000  |
15 |  R2:00000000_7FFFF7FF   R3:00000000_7FFFF7FF  |
16 |  R4:00000000_7FFFF7FF   R5:00000000_7FFFF7FF  |
17 |  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
18 |  R8:FFFFFFFF_00000068   R9:D7BFCA04_005C6E00  |
19 |  RA:FFFFFFFF_005C6E08   RB:00000000_000100B0  |
20 |  RC:00000000_00026CB0   RD:00000000_7FFFF7FF  |
21 |  RE:00000000_7FFFF7FF   RF:00000000_7FFFF7FF  |
22 +-----------------------------------------------+
23 |General Registers at time of error             |
24 +-----------------------------------------------+
25 |  R0:00000000_7FFFF7FF   R1:00000000_0003A000  |
26 |  R2:00000000_00000020   R3:00000000_08F5B9A8  |
27 |  R4:00000000_024603F0   R5:00000000_7FFFF7FF  |
28 |  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
29 |  R8:FFFFFFFF_00000068   R9:D7BFCA04_00FC36A8  |
30 |  RA:FFFFFFFF_88F108A8   RB:00000000_08F5B958  |
31 |  RC:00000000_08F10940   RD:00000000_08F5B958  |
32 |  RE:00000000_7FFFF7FF   RF:00000000_00000002  |
33 +-----------------------------------------------+
34 |Access Registers at time of error              |
35 +-----------------------------------------------+
36 |R0:005FE7EC R1:00000000 R2:00000002 R3:00000020|
37 |R4:FFFFFFFF R5:FFFFFFFF R6:FFFFFFFF R7:FFFFFFFF|
38 |R8:FFFFFFFF R9:FFFFFFFF RA:00000000 RB:00000000|
39 |RC:00000000 RD:00000000 RE:00000000 RF:07000000|
40 +-----------------------------------------------+
41 *** END OF POTENTIAL VULNERABILITY REPORT ***
```

Figure 35. Potential Vulnerability Output Example 0E0.

Line 01

Indicates the service number where the potential vulnerability was detected.

Line 02

Indicates the six-digit ABEND code, which allows for user ABENDs when applicable. The line also includes the ABEND's reason code.

Line 03

Indicates the failing Program Status Word (PSW) as well as the name of the module where the potential vulnerability was found, if known, and the offset where the ABEND occurred.

Line 04

Indicates the failing instruction and the instructions length from the System Diagnostic Work Area (SDWA). Information on the SDWA can be found in the SDWA Mapping section of z/OS MVS Data Areas Volume 4 (RRP - XTL). Information on the op code can be found in IBM z/Architecture Principles of Operation.

Line 05
Indicates the failing instruction translated into assembler format.

Line 06
Indicates the access register that caused the translation exception.

Line 07
The home, primary and secondary address space identifiers.

Line 08
Indicates the probable CVSS score and vector of the potential vulnerability. CVSS scores are calculated using CVSS version 3.0. More information can be found here https://www.first.org/cvss/calculator/3.0.

Line 09
Indicates the header for the sample slip command

Line 10
Provides an example slip command when one can be automatically generated.

Lines 14-21
Indicates the contents of the general purpose registers immediately before the service was invoked.

Lines 25-32
Indicates the contents of the general purpose registers at the time of error.

Lines 36-39
Indicates the contents of the access registers at the time of error.

Additional output is provided if an overlay is detected.

```
01 *** STORAGE OVERLAY FOUND IN PC 00180700 00000001 ***
02 CVSS: 8.8 (CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
03 +-----------------------------------------------+
04 |General Registers before the service           |
05 +-----------------------------------------------+
06 |  R0:00000000_7FFFF7FF   R1:00000000_7FFFF7FF  |
07 |  R2:00000000_00058200   R3:00000000_7FFFF7FF  |
08 |  R4:00000000_7FFFF7FF   R5:00000000_7FFFF7FF  |
09 |  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
10 |  R8:FFFFFFFF_00000068   R9:D7BFCA06_005C6E00  |
11 |  RA:FFFFFFFF_005C6E08   RB:00000000_000300B0  |
12 |  RC:00000000_00046CB0   RD:00000000_7FFFF7FF  |
13 |  RE:00000000_7FFFF7FF   RF:00000000_7FFFF7FF  |
14 +-----------------------------------------------+
15 |Key 0 Parameter data before the service        |
16 +-----------------------------------------------+
17 | 00058200: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
18 | 00058210: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
19 | 00058220: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
20 | 00058230: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
21 | 00058240: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
22 +-----------------------------------------------+
23 |Key 0 Parameter data after the service         |
24 +-----------------------------------------------+
25 | 00058200: 00000020 7FFFF7FF 7FFFF7FF 7FFFF7FF |
26 | 00058210: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
27 | 00058220: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
28 | 00058230: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
29 | 00058240: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
30 +-----------------------------------------------+
31 *** END OF STORAGE OVERLAY REPORT ***
```

Figure 36. Storage overlay output.

Line 01
Indicates that a storage overlay was detected in the indicated service.

Line 02
Indicates the probable CVSS score and vector of the potential vulnerability. CVSS scores are calculated using CVSS version 3.0. More information can be found here https://www.first.org/cvss/calculator/3.0.

Lines 06-13
Indicates the contents of the general purpose registers immediately before the service was invoked.

Lines 17-21
Indicates the contents of the storage where the overlay was detected before the service was invoked.

Lines 25-29
Indicates the contents of the storage where the overlay was detected after the service was invoked.

At the bottom of each service's output a summary is given.

```
01 +-----------------------------------------------+
02 | Summary for PC: 00180703           VULNERABLE |
03 +-----------------------------------------------+
04 | Testcase Templates Run:             00000020x |
05 | Test Iterations:                    00000045x |
06 +-----------------------------------------------+
07 | Overlay Count:                      00000006x |
08 | Potential Vulnerability Count:      0000001Ax |
09 +-----------------------------------------------+
```

Figure 37. Summary output.

Line 02
Indicates the service that was run and its overall status. The possibilities are as follows:

SUCCESSFUL
All tests ran with no potential vulnerabilities detected.

VULNERABLE
One or more potential vulnerabilities were detected.

INCOMPLETE
Testing was unable to complete entirely for this service. This may be caused, for example, by the logrec data set becoming full or by the test timing out if the service being tested does not return within a few seconds of being invoked.

UNDEFINED
The specified service is not defined to the system or cannot be invoked from this address space.

PROTECTED
The service cannot be invoked by unauthorized users and therefor does not apply to this tool's testing

Line 04
Indicates the total number of tests run on the service.

Line 05
Indicates the number of iterations run on the service. The number of iterations performed on any given service may vary.

Line 07
Indicates the number of storage overlays detected.

Line 08
Indicates the total number of potential vulnerabilities detected.

Note: The same potential vulnerability may be hit by multiple tests which can affect the counts in lines 7 and 8.

Serviceability Note: There can be false positives in the resulting output of this scanner as part of its normal operation. This is not a flaw in the tool but rather an indication of possible vulnerability scenarios that may or may not apply in particular instances. System integrity within a PC or SVC routine is the responsibility of the respective product owner of that routine, whether it be IBM-, vendor-, or in-house-provided code. The three-character prefix within the routine name provides a likely indication as to the product and its owner for purposes of confirming problem diagnosis and providing a fix as needed.

Setting a SLIP to Capture a Dump

When a vulnerability is found that you would like to set a skip for, the SLIP command syntax will vary depending what abend code *<xyz>* observed and if it was in PVT, LPA, or NUC.

Note: The following example slip commands use ACTION=SVCD and MATCHLIM=1 by default.

PVT Example:

```
SLIP SET,COMP=<xyz>,P=(<modname>,<offset>),SDATA=(trt,rgn,sum,nuc,csa,psa,lpa,sqa,lsqa),end
```

Figure 38. PVT Example.

LPA Example:

```
SLIP SET,COMP=<xyz>,L=(<modname>,<offset>),SDATA=(trt,rgn,sum,nuc,csa,psa,lpa,sqa,lsqa),end
```

Figure 39. LPA Example.

NUC Example:

```
SLIP SET,COMP=<xyz>,N=(<modname>,<offset>),SDATA=(trt,rgn,sum,nuc,csa,psa,lpa,sqa,lsqa),end
```

IF Example:

```
SLIP SET,IF,RANGE=(<failing address>),SDATA=(trt,rgn,lpa,nuc,sum,csa),end
```

After setting the SLIP, rerun the failing service number of module.

Note: PVTMOD SLIPs will not run if the local lock is held. If no module has been provided, or a PVTMOD COMP SLIP will not work, set a Instruction Fetch SLIP (IF) using the address. Add the job name (default is ZACSJ) to the slip, and if needed a data parameter containing the value desired in a specific register at the point you want to capture the dump may be added as well.

Vulnerability Examples and Fixes

The following examples are samples of potential vulnerability reports from zACS. These were generated using a test program called BPNTEST and are not real vulnerabilities on the system. Descriptions of the vulnerabilities and possible fixes for them will follow.

Fetch Vulnerability Example

```
TEST IN PROGRESS ON 2020/04/30 AT 11:07:31 FOR PC 00180601
*** POTENTIAL VULNERABILITY FOUND IN PC 00180601 ***
ABEND COMPLETION CODE: 0C4000  REASON CODE: 00000011
PSW: 070C6000 890EC342  MODULE: PVTMOD=(BPNTEST,00001342)
INSTR LEN: 06  FAILING INSTR: B048 9A33 B04C D203 3000 2000
TRANSLATED INSTR: MVC     0(4,R3),0(R2)
SOURCE ADDRESS CAUSING TRANSLATION EXCEPTION: 00000000_7FFFF801
HOME ASID: 0029 PRIMARY ASID: 001A SECONDARY ASID: 0029
CVSS: 6.5 (CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N)
SLIP SAMPLE FOR PC 00180601:
SLIP SET,COMP=0C4,P=(BPNTEST,00001342),SDATA=(TRT,RGN,SUM,CSA),END
+-----------------------------------------------+
|General Registers before the service           |
+-----------------------------------------------+
|  R0:00000000_7FFFF7FF   R1:00000000_0004E000  |
|  R2:00000000_7FFFF7FF   R3:00000000_7FFFF7FF  |
|  R4:00000000_7FFFF7FF   R5:00000000_7FFFF7FF  |
|  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
|  R8:FFFFFFFF_00000068   R9:D7D905AE_005C6E00  |
|  RA:FFFFFFFF_005C6E08   RB:00000000_000220B8  |
|  RC:00000000_0003AD40   RD:00000000_7FFFF7FF  |
|  RE:00000000_7FFFF7FF   RF:00000000_7FFFF7FF  |
+-----------------------------------------------+
|General Registers at time of error             |
+-----------------------------------------------+
|  R0:00000000_00000000   R1:00000000_0004E000  |
|  R2:00000000_7FFFF7FF   R3:00000000_090F3A00  |
|  R4:00000000_029F0110   R5:00000000_7FFFF7FF  |
|  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
|  R8:FFFFFFFF_00000068   R9:D7D905AE_00FC9610  |
|  RA:FFFFFFFF_890EC2F0   RB:00000000_090F39B0  |
|  RC:00000000_090EC388   RD:00000000_090F39B0  |
|  RE:00000000_7FFFF7FF   RF:00000000_00000002  |
+-----------------------------------------------+
*** END OF POTENTIAL VULNERABILITY REPORT ***
```

Figure 40. Fetch Vulnerability example output.

From the report, the CVSS score indicates the probable type of error that was detected. This CVSS score of 6.5 indicates a probable fetch violation. The report indicates that the PC is suspected of fetching storage that the caller was not authorized to view. The failing instruction information as well as the general registers can be used to understand what is occurring. In this case, the op code of the failing instruction is ‘D2’x which indicates an MVC instruction. The module name of BPNTEST and offset of ‘D02’x are also provided for additional diagnostics. A dump can also be captured of the module at that offset using the sample SLIP command provided in the report.

This vulnerability was caused by the following code sample:

```
   LA R3,copyparms
   MVC 0(4,R3),0(R2)
```

Figure 41. Fetch Vulnerability code example.

This code is vulnerable because it blindly copies information from the caller. General register 2 is being used as an input in this case, but the value provided in general register 2 is an address to which the caller does not necessarily have access. The PC routine is not accounting for the key of the caller.

The following is an alternative implementation:

```
    LHI R3,1
    ESTA R0,R3
    SRDL R0,48
    LHI R0,3
    LA R3,copyparms
    MVCSK 0(R3),0(R2)
```

Figure 42. Fetch Vulnerability alternative implementation.

The fix for this vulnerability is to utilize the instruction, MVCSK. The instruction will only succeed when the caller has access to the parameter provided, maintaining system integrity.

A break down of the assembler instructions in the fixed code are as follows:

LHI R3,1
LHI (Load Halfword Immediate (32)) loads general register 3 with a 1 indicating PSW for the ESTA instruction.

ESTA R0,R3
ESTA (Extract Stacked State) extracts the PSW's value and put it into general register 0.

SRDL R0,48
SRDL(Shift Right Double Logical) shifts the PSW value into the general register 1 such that the portion with the caller's keys gets set in general register 1.

LHI R0,3
LHI (Load Halfword Immediate (32)) loads the length that the copy should be into general register 0.

LA R3,copyparms
LA (Load Address) loads the address where the copy is to the program's storage (the destination or the target address).

MVCSK 0(R3),0(R2)
MVCSK (Move with Source Key) will move data from the address in general register 2 to the address in general register 3 using the key that is set in general register 1 and the length that is set in general register 0.

Store Vulnerability Example

```
TEST IN PROGRESS ON 2020/04/30 AT 11:04:29 FOR PC 00180600
*** POTENTIAL VULNERABILITY FOUND IN PC 00180600 ***
ABEND COMPLETION CODE: 0C4000  REASON CODE: 00000011
PSW: 070C6000 890EC25A  MODULE: PVTMOD=(BPNTEST,0000125A)
INSTR LEN: 06  FAILING INSTR: B048 9A33 B04C D203 2000 3000
TRANSLATED INSTR: MVC     0(4,R2),0(R3)
TARGET ADDRESS CAUSING TRANSLATION EXCEPTION: 00000000_7FFFF401
HOME ASID: 0029 PRIMARY ASID: 001A SECONDARY ASID: 0029
CVSS: 8.8 (CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
SLIP SAMPLE FOR PC 00180600:
SLIP SET,COMP=0C4,P=(BPNTEST,0000125A),SDATA=(TRT,RGN,SUM,CSA),END
+-----------------------------------------------+
|General Registers before the service           |
+-----------------------------------------------+
|  R0:00000000_7FFFF7FF   R1:00000000_00053000  |
|  R2:00000000_7FFFF7FF   R3:00000000_7FFFF7FF  |
|  R4:00000000_7FFFF7FF   R5:00000000_7FFFF7FF  |
|  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
|  R8:FFFFFFFF_00000068   R9:D7D90500_005C6E00  |
|  RA:FFFFFFFF_005C6E08   RB:00000000_000270B8  |
|  RC:00000000_0003FD40   RD:00000000_7FFFF7FF  |
|  RE:00000000_7FFFF7FF   RF:00000000_7FFFF7FF  |
+-----------------------------------------------+
|General Registers at time of error             |
+-----------------------------------------------+
|  R0:00000000_00000000   R1:00000000_00053000  |
|  R2:00000000_7FFFF7FF   R3:00000000_090F2FB0  |
|  R4:00000000_029F0100   R5:00000000_7FFFF7FF  |
|  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
|  R8:FFFFFFFF_00000068   R9:D7D90500_00FC9610  |
|  RA:FFFFFFFF_890EC200   RB:00000000_090F2F60  |
|  RC:00000000_090EC2A0   RD:00000000_090F2F60  |
|  RE:00000000_7FFFF7FF   RF:00000000_00000002  |
+-----------------------------------------------+
*** END OF POTENTIAL VULNERABILITY REPORT ***
```

Figure 43. Store Vulnerability example output.

From the report, the CVSS score indicates the probable type of error that was detected. This CVSS score of 8.8 indicates a probable store violation. The report indicates that the PC is suspected of storing to memory into which the caller was not authorized to store. The failing instruction information as well as the general registers can be used to understand what is occurring. In this case, the op code of the failing instruction is ‘D2’x which indicates an MVC instruction. The module name of BPNTEST and offset of ‘D02’x are also provided for additional diagnostics. A dump can also be captured of the module at that offset using the sample SLIP command provided in the report.

This vulnerability was caused by the following code sample:

```
LA R3,outputdata
MVC 0(4,R2),0(R3)
```

Figure 44. Store Vulnerability code example.

This code is vulnerable because it blindly copies information to an address provided by the caller. General register 2 is being used as an output in this case, but the value provided in general register 2 is an address to which the caller does not necessarily have access. The PC routine is not accounting for the key of the caller.

The following is an alternative implementation:

```
LHI R3,1
ESTA R0,R3
SRDL R0,48
LHI R0,3
LA R3,outputdata
MVCDK 0(R2),0(R3)
```

Figure 45. Store Vulnerability alternative implementation.

The fix for this vulnerability is to utilize the instruction, MVCDK. The instruction will only succeed when the caller has access to the parameter provided, maintaining system integrity.

A break down of the assembler instructions in the fixed code are as follows:

LHI R3,1
LHI (Load Halfword Immediate (32)) loads the general register 3 with a 1 indicating PSW for the ESTA instruction.

ESTA R0,R3
ESTA (Extract Stacked State) extracts the PSW's value and put it into general register 0.

SRDL R0,48
SRDL (Shift Right Double Logical) shifts the PSW value into general register 1 such that the portion with the caller's key gets set in general register 1.

LHI R0,3
LHI (Load Halfword Immediate (32)) loads the length that the copy should be into general register 0.

LA R3,outputdata
LA (Load Address) loads the source address of where the data is to be copied.

MVCDK 0(R2),0(R3)
MVCDK (Move with Destination Key) will move data at the address in general register 3 to the address in general register 2 using the key that is set in general register 1 and the length that is set in general register 0.

Indirect Parameter and Storage Overlay Vulnerability Example

```
TEST IN PROGRESS ON 2020/12/09 AT 10:41:27 FOR PC 0018110C
*** POTENTIAL VULNERABILITY FOUND IN PC 0018110C ***
ABEND COMPLETION CODE: 0C4000  REASON CODE: 00000004
PSW: 070C1000 88B5E09C  MODULE: LPAMOD=(BPNTPCC,0000009C)
INSTR LEN: 04  FAILING INSTR: AAAA 5034 0000 5800 C00C 41F0
TRANSLATED INSTR: ST      R3,0(R4)
TARGET ADDRESS CAUSING TRANSLATION EXCEPTION: 00000000_00038404
HOME ASID: 001E  PRIMARY ASID: 001E  SECONDARY ASID: 001E
CVSS: 8.8 (CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
SLIP SAMPLE FOR PC 0018080C:
SLIP SET,COMP=0C4,L=(BPNTPCC,0000009C),SDATA=(TRT,LPA,SUM,CSA),END
+-----------------------------------------------+
|General Registers before the service           |
+-----------------------------------------------+
|  R0:00000000_7FFFF7FF   R1:00000000_00033000  |
|  R2:00000000_7FFFF7FF   R3:00000000_7FFFF7FF  |
|  R4:00000000_7FFFF7FF   R5:00000000_7FFFF7FF  |
|  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
|  R8:FFFFFFFF_00000000   R9:FFFFFFFF_007C8E08  |
|  RA:FFFFFFFF_007C8E00   RB:00000000_00006E58  |
|  RC:00000000_0003A650   RD:00000000_7FFFF7FF  |
|  RE:00000000_7FFFF7FF   RF:00000000_7FFFF7FF  |
+-----------------------------------------------+
|General Registers at time of error             |
+-----------------------------------------------+
|  R0:00000000_00000003   R1:00000000_0000078D  |
|  R2:00000000_7FFFF7FF   R3:00000000_AAAAAAAA  |
|  R4:00000000_00038000   R5:00000000_7FFFF7FF  |
|  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
|  R8:FFFFFFFF_00000000   R9:00000000_007C8E00  |
|  RA:FFFFFFFF_88B5E02C   RB:00000000_0A602860  |
|  RC:00000000_08B5E0D0   RD:00000000_0A602860  |
|  RE:00000000_0A6028D4   RF:00000000_00036000  |
+-----------------------------------------------+
*** END OF POTENTIAL VULNERABILITY REPORT ***
*** STORAGE OVERLAY FOUND IN PC 0018080C ***
CVSS: 8.8 (CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
+-----------------------------------------------+
|General Registers before the service           |
+-----------------------------------------------+
|  R0:00000000_7FFFF7FF   R1:00000000_00033000  |
|  R2:00000000_7FFFF7FF   R3:00000000_7FFFF7FF  |
|  R4:00000000_7FFFF7FF   R5:00000000_7FFFF7FF  |
|  R6:00000000_7FFFF7FF   R7:00000000_7FFFF7FF  |
|  R8:FFFFFFFF_00000000   R9:FFFFFFFF_007C8E08  |
|  RA:FFFFFFFF_007C8E00   RB:00000000_00006E58  |
|  RC:00000000_0003A650   RD:00000000_7FFFF7FF  |
|  RE:00000000_7FFFF7FF   RF:00000000_7FFFF7FF  |
+-----------------------------------------------+
|Key 0 Parameter data before the service        |
+-----------------------------------------------+
| 0002F490: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
| 0002F4A0: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
| 0002F4B0: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
| 0002F4C0: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
| 0002F4D0: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
+-----------------------------------------------+
|Key 0 Parameter data after the service         |
+-----------------------------------------------+
| 0002F490: AAAAAAAA 7FFFF7FF 7FFFF7FF 7FFFF7FF |
| 0002F4A0: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
| 0002F4B0: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
| 0002F4C0: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
| 0002F4D0: 7FFFF7FF 7FFFF7FF 7FFFF7FF 7FFFF7FF |
+-----------------------------------------------+
*** END OF STORAGE OVERLAY REPORT ***
```

Figure 46. Storage Overlay example output.

From this report, the error information points to a store protection violation. The CVSS score indicates this as well as the translated instruction at the time of error. The CVSS of 8.8 indicating a likely store violation occurred, and the translated instruction is a store instruction. As with the previous examples, the module name of BPNTPCC and the offset of ‘9C’x are provided for diagnostic purposes such as setting a slip command at the module and offset provided to get a dump of the potential vulnerability. In a separate test, zACS flagged that an overlay of storage has occurred. The storage overlay that occurred can be noted

in the output report by comparing the parameter data before and after the service. The first 4 bytes of storage have been overwritten after the service was called.

This example shows a routine that does several things right, but it does one thing wrong. The zACS tool gets far enough into the routine to expose the vulnerability and demonstrate the storage overlay.

```
    LA R14,myLB1
    LR R15,R1
    LHI R3,1
    ESTA R0,R3
    SRDL R0,48
    LHI R0,3
    MVCSK 0(R14),0(R15)
    L R15,myLB2ptr
    LA R14,myLB2
    MVCSK 0(R14),0(R15)
    LHI R0,4
    L R4,myLB3ptr
    L R3,myConst
    ST R3,0(R4)
```

Figure 47. Storage overlay code example.

The main issue in this code is the store instruction at the end. It is storing to the provided storage without considering the key of the caller. In the above example, this routine did properly copy an input from the caller as well as another input that was pointed to by the first input using MVCSK instructions. However, the input that was copied is then used as a target for a store instruction using key 0. This is an example of an indirect parameter causing a vulnerability after the initial parameters have been copied safely by the routine.

The following is an alternative implementation:

```
    LA R14,myLB1
    LR R15,R1
    LHI R3,1
    ESTA R0,R3
    SRDL R0,48
    LHI R0,3
    MVCSK 0(R14),0(R15)
    L R15,myLB2ptr
    LA R14,myLB2
    MVCSK 0(R14),0(R15)
    LHI R0,4
    L R4,myLB3ptr
    LA R3,myConst
    MVCDK 0(R4),0(R3)
```

Figure 48. Storage Overlay alternative implementation.

This fixes the vulnerability from the store because it considers the key of the caller when attempting to write to the storage provided by the caller. If the caller does not have access to the provided storage, the attempt to write to storage will fail with an ABEND 0C4. This will prevent the vulnerability caused by the original store instruction.

A break down of the assembler instructions in the fixed code are as follows:

LA R14,myLB1
LA (Load Address) loads the address of the local storage for the caller’s parameter to be copied into.

LR R15,R1

LR (Load Register) loads the value from register 1 which contains a parameter from the caller into register 15.

LHI R3,1
LHI (Load Halfword Immediate (32)) loads general register 3 with a 1 indicating PSW for the ESTA instruction.

ESTA R0,R3
ESTA (Extract Stacked State) extracts the PSW's value and put it into general register 0.

SRDL R0,48

SRDL(Shift Right Double Logical) shifts the PSW value into the general register 1 such that the portion with the caller's keys gets set in general register 1.

LHI R0,3

LHI (Load Halfword Immediate (32)) loads the length that the copy should be into general register 0.

MVCSK 0(R14),0(R15)

MVCSK (Move with Source Key) will move data from the address in general register 15 to the address in general register 14 using the key that is set in general register 1 and the length that is set in general register 0. This is copying the first parameter from the caller into local storage.

L R15,myLB2ptr

L(Load) load the value of the pointer passed as part of the caller’s first parameter.

LA R14,myLB2

LA (Load Address) loads the address of the local storage for the caller’s second parameter to be copied into.

MVCSK 0(R14),0(R15)

MVCSK (Move with Source Key) will move data from the address in general register 15 to the address in general register 14 using the key that is set in general register 1 and the length that is set in general register 0. This is copying the first parameter from the caller into local storage.

LHI R0,3

LHI (Load Halfword Immediate (32)) loads the length that the copy should be into general register 0.

L R4,myLB3ptr

L(Load) load the value of the pointer passed as part of the caller’s second parameter. This is where the output is copied into.

LA R3,myConst

LA (Load Address) loads the address of the local storage for the caller’s second parameter which to be copied into as part of the output.

MVCDK 0(R4),0(R3)

MVCDK (Move with Destination Key) will move data at the address in general register 3 to the address in general register 4 using the key that is set in general register 1 and the length that is set in general register 0. Safely copying out to the caller provided storage.

Chapter 6. System Codes

Return and Reason codes

This topic covers the return and reason codes of the output logs.

Table 1. Return and Reason codes for output logs.

Return code	Description
RC: 0	Success.
RC: 4	Warning. • RSN: 405 - potential vulnerability • RSN: 407 - PC or SVC cannot be run by unauthorized users • RSN: 408 - PC or SVC undefined • RSN: 409 - service timed out
RC: 8	Internal Error.
RC: C	Environmental Error. • RSN: C09 logrec error

Abend 2600 Reason Code Analysis

This topic covers the reason codes that occur from the 2600 abend code.

Table 2. Reason codes

Reason code	Description	Explanation
01	Start failed in BPNGMAIN	Check your zACS set up and configuration for any errors.
02	QEDIT failed in BPNGMAIN	Check your zACS set up and configuration for any errors.
03	LOCASCB failed in BPNGPC1	Check your zACS set up and configuration for any errors.
04	IEANTRT failed in BPNGPC1	Check your zACS set up and configuration for any errors.
05	IEANTCR failed in BPNGPC1	Check your zACS set up and configuration for any errors.
06	IEANTRT failed in BPNGLNK	Unexpected component error. Contact z/OS Support for diagnosis.
07	CSVDYLPA failed in BPNGMAIN	Unexpected component error. Contact z/OS Support for diagnosis.
08	ENFREQ failed in BPNGMAIN	Unexpected component error. Contact z/OS Support for diagnosis.
09	IEANTCR failed in BPNGMAIN	Unexpected component error. Contact z/OS Support for diagnosis.
0A	CSVDYLPA failed in BPNGMAIN	Unexpected component error. Contact z/OS Support for diagnosis.

Table 2. Reason codes (continued)

Reason code	Description	Explanation
0B	IEANTRT failed in BPNGENFL	Unexpected component error. Contact z/OS Support for diagnosis.
0C	OPEN failed in BPNGPCN	Unexpected component error. Contact z/OS Support for diagnosis.
0D	Already started	An attempt to start the tool was made when the tool is already started.
0E	Registration failed	The tool is not registered.
0F	CSVDYLPA Failed	Unexpected component error. Contact z/OS Support for diagnosis.
10	LOCASCB Failed	Unexpected component error. Contact z/OS Support for diagnosis.
11	OPEN failed in BPNGSVCN	Unexpected component error. Contact z/OS Support for diagnosis.
12	zACS Not Started	An attempt was made to run the tool, however, the tool is not started.
13	Incompatible Release	An attempt was made to start the tool on an incompatible version of z/OS. z/OS 2.4 and above are supported.
14	Task busy in BPNGPC1	An attempt was made to run a test while testing is already in progress.
15	Task busy in BPNKPCX	An attempt was made to run a PC test while testing is already in progress.
16	Task busy in BPNKSVC	An attempt was made to run a SVC test while testing is already in progress.
17	Task busy in BPNKEXT	An attempt was made to run a ESR test while testing is already in progress.
18	ASMADOP Failed	Unexpected component error. Contact z/OS Support for diagnosis.
19	Emulator Disallowed	zACS is not permitted to run on an emulator.
1A	Security Error	The user running a zACS test case must have read access to the BPN.RUN resource in the XFACILIT class. See message ICH408I for more details.
1B	PC Number Max Exceeded	More than 256 PCs supplied for a single LX.

Appendix A. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact the z/OS team web page (www.ibm.com/systems/campaignmail/z/zos/contact_z) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1)

are mutually exclusive alternatives. If you hear the lines `3.1 USERID` and `3.1 SYSTEMID`, your syntax can include either `USERID` or `SYSTEMID`, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The `*` symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element `*FILE` with dotted decimal number 3 is given the format `3 \* FILE`. Format `3* FILE` indicates that syntax element `FILE` repeats. Format `3* \* FILE` indicates that syntax element `* FILE` repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines `5.1*, 5.1 LASTRUN, and 5.1 DELETE` mean that if you use more than one of the `LASTRUN` and `DELETE` syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line `2.1 %OP1` means that you must refer to separate syntax fragment `OP1`.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example `5? NOTIFY`). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines `5 ?, 5 NOTIFY`, and `5 UPDATE`, you know that the syntax elements `NOTIFY` and `UPDATE` are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the `!` symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines `2? FILE, 2.1! (KEEP), and 2.1 (DELETE)`, you know that `(KEEP)` is the default option for the `FILE` keyword. In the example, if you include the `FILE` keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the `FILE` keyword is omitted, the default `FILE(KEEP)` is used. However, if you hear the lines `2? FILE, 2.1, 2.1.1! (KEEP)`, and `2.1.1 (DELETE)`, the default option KEEP applies only to the next higher dotted decimal number, `2.1` (which does not have an associated keyword), and does not apply to `2? FILE`. Nothing is used if the keyword `FILE` is omitted.

*** indicates an optional syntax element that is repeatable**
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line `5.1* data area`, you know that you can include one data area, more than one data area, or no data area. If you hear the lines `3* , 3 HOST, 3 STATE`, you know that you can include `HOST, STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for the Knowledge Centers. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Site Counsel
2455 South Road

Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, JES3, and MVS™, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those

products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and Trademark information (www.ibm.com/legal/copytrade.shtml).

Index

T

U

W

Product Number: 5650-ZOS

SC283123-40