

Implementing the IMS Catalog using Ansible

Agenda

Ansible Overview

- Concepts
- Benefits
- Ansible Automation Platform
- Ansible for IBM Z Content
- Ansible with IMS
 - Brief overview of IMS Catalog
 - Overview of Ansible IMS Collection
 - Demo
 - Configure IMS Catalog with Ansible
 - Walkthrough of IMS Provisioning Playbooks



ANSIBLE

Benefits

Simplicity

Uses a simple, human-readable YAML syntax for playbooks, making it easy to learn and use.

Agentless

No need to install special software on managed nodes as it relies on SSH to communicate with the nodes, reducing overhead and simplifying setup.

Idempotent

Ansible playbooks are designed to be idempotent, meaning that running a playbook multiple times will always result in the same state.

Community

There is a large and active community of users and contributors. This community provides a wealth of knowledge, pre-built roles, modules, and playbooks through Ansible Galaxy.

Basic concepts

Inventory

This is a list of managed nodes (hosts) that Ansible can interact with.

Can be a simple text file or a dynamic inventory.

Playbooks

These are YAML files that define a series of tasks to be executed on managed nodes.

They are Ansible's configuration, deployment, and orchestration language and can include variables, tasks, handlers and roles.

Modules

Units of code that Ansible executes on managed nodes.

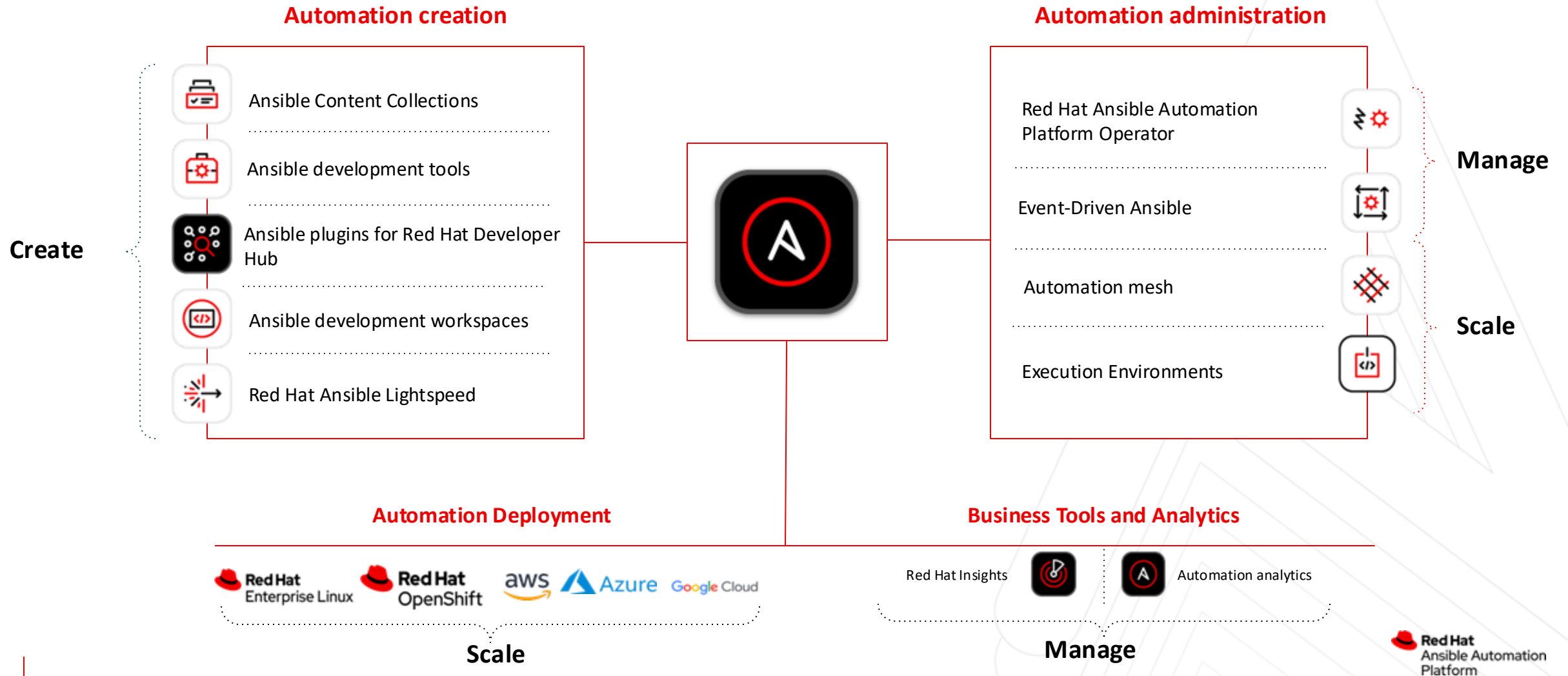
Each module is designed for a specific purpose.

Roles

Roles are a way to organize playbooks and other related files to facilitate sharing and reuse.

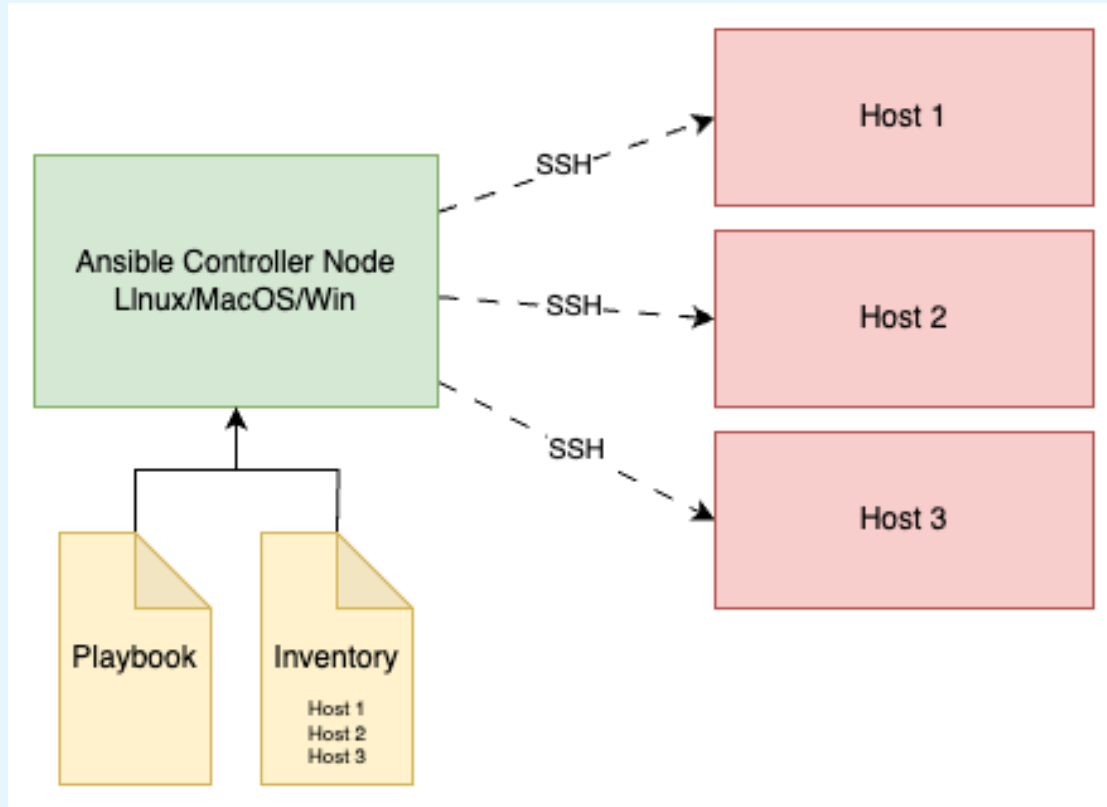
A role can include tasks and variables just as a playbook.

An integrated solution **for the enterprise.**



	Traditional Mainframe Management	Ansible
Automation Approach	<ul style="list-style-type: none"> - Often relies on JCL and custom scripts (REXX). - Procedural, step-by-step instructions. - Typically mainframe specific. 	<ul style="list-style-type: none"> - Declarative, describing desired state. - Uses YAML for playbooks, easy to read and write. - Cross-platform, can manage mainframe and distributed systems.
Configuration Management	<ul style="list-style-type: none"> - Often manual processes or homegrown tools. - Configuration spread across multiple libraries or data sets. 	<ul style="list-style-type: none"> - Centralized configuration as code. - Version control integration. - Easy to audit and track changes.
Learning Curve and Skill Transfer	<ul style="list-style-type: none"> - Steep learning curve for mainframe-specific tools. - Limited pool of skilled professionals. 	<ul style="list-style-type: none"> - Easier learning curve, especially for those familiar with modern IT tools and practices. - Skills transferable across platforms.

Requirements



Ansible installed on control node

The control node is the machine where Ansible commands and playbooks are executed.

SSH Access to managed nodes

Ansible uses SSH for connecting to and managing remote hosts.

The control node must have access to all hosts.

Python and ZOAU installed on managed nodes

Ansible modules are typically written in Python and executed on managed nodes.

No charge PID.
PAX and SMP/E editions.

Valid inventory file or hosts defined

The inventory file contains the list of hosts being managed.

Hosts can be divided in different groups or could even be retrieved dynamically with a script.

Similarities and Comparisons
JCL and Ansible

Purpose	Structure	Job Steps / Tasks
<p>JCL: Defines a series of jobs to be executed on a mainframe system.</p> <p>Ansible Playbooks: Define a series of tasks to be executed on multiple servers or systems.</p>	<p>JCL: Uses a structured language with specific statements (e.g., JOB, EXEC, DD).</p> <p>Ansible: Uses YAML format with defined structures (e.g., hosts, tasks, vars).</p>	<p>JCL: Each EXEC statement typically represents a job step.</p> <p>Ansible: Each task in a playbook is analogous to a job step.</p>
Conditional Execution	Error Handling	Output
<p>JCL: Uses IF/THEN/ELSE constructs and condition codes.</p> <p>Ansible: Uses 'when' statements and registered variables for conditional execution.</p>	<p>JCL: Uses condition codes and COND parameters to handle errors.</p> <p>Ansible: Uses 'ignore_errors', 'failed_when', and 'block/rescue' for error handling.</p>	<p>SDSF: Provides a way to view and control job output and system resources.</p> <p>Ansible: Provides real-time output of task execution and supports various output formats (JSON, YAML, etc.).</p>

What does automation on z/OS look like?

```
//CREATPDS JOB (123),CLASS=C,MSGCLASS=S,MSGLEVEL=(1,1),
//  NOTIFY=&SYSUID
//*
//STEP001 EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSDUMP DD SYSOUT=*
//DD01 DD DSN=USERID.DATASET1,
//      DISP=(NEW,CATLG,DELETE),
//      SPACE=(TRK,(12,33),RLSE),UNIT=SYSDA,
//      DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=800)
//*
//STEP002 EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=A
//OUT1 DD DSNAME=USERID.DATASET1,UNIT=disk,VOL=SER=111112,
//      DISP=(OLD,KEEP)
//IN6 DD DSNAME=USERID.DATASET6,UNIT=disk,VOL=SER=111115,
//      DISP=OLD
//SYSUT3 DD UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4 DD UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN DD *
COPYOPER COPY OUTDD=OUT1
         INDD=IN6
         SELECT MEMBER=(B,A)
```

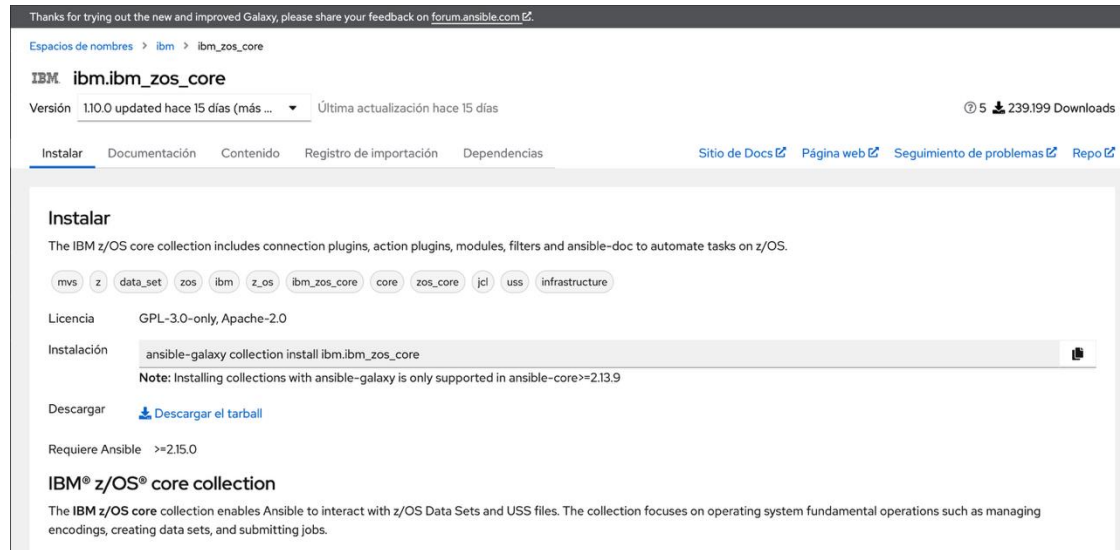
How are we simplifying automation on z/OS with Ansible?

```
- name: Copy a PDS to a new customized PDS.  
  ibm.ibm_zos_core.zos_copy:  
    src: "USERID.DATASET6{{ item }}"  
    dest: "USERID.DATASET1{{ item }}"  
    remote_src: true  
    volume: 111112  
  loop:  
    - A  
    - B
```

Ansible for IBM Z Collections

A collection is a distribution format for Ansible content.

It can be seen as as a package that bundles playbooks, roles, modules and plugins together.



ibm.ibm_zos_core

This collections enables Ansible users to manage batch jobs, program authorizations and operator operations.

- zos_copy
- zos_dataset
- zos_job_submit
- zos_operator
- zos_apf

+227k







ibm.ibm_zos_ims

The ibm_zos_ims collection support tasks such as generating DBDs, PSBs and ACBs as well as running type-1 and type-2 commands.

- ims_acbgen
- ims_catalog_populate
- ims_command
- ims_dbrc
- ims_ddl

+47k

Red Hat® Ansible® Certified Content for IBM Z®: Collections

Available on Ansible Galaxy here and on Automation Hub				
IBM z/OS® Core		ibm_zos_core Provided by ibm 23 Modules 0 Roles 26 Plugins 0 Dependencies mvs z data_set 9 more	354,000	Downloads
IBM z/OS® CICS®		ibm_zos_cics Provided by ibm 15 Modules 0 Roles 35 Plugins 0 Dependencies z zos cics 4 more	20,000	Downloads
IBM z/OS® IMS™		ibm_zos_ims Provided by ibm 8 Modules 0 Roles 9 Plugins 1 Dependency mvs zos_ims z 11 more	52,000	Downloads
IBM z/OS® Management Facility (z/OSMF)		ibm_zosmf Provided by ibm 3 Modules 13 Roles 4 Plugins 0 Dependencies mvs liberty_provisioning_template z	9,000	Downloads
IBM Z® Hardware Management Console		ibm_zhmc Provided by ibm 28 Modules 0 Roles 1 Plugin 0 Dependencies z hmc ibm 2 more	108,000	Downloads
IBM Z® System Automation		ibm_zos_sysauto Provided by ibm 0 Modules 2 Roles 0 Plugins 0 Dependencies z zos ibm 6 more	3,950	Downloads

Collectively

- 83 Modules
- 15 Roles

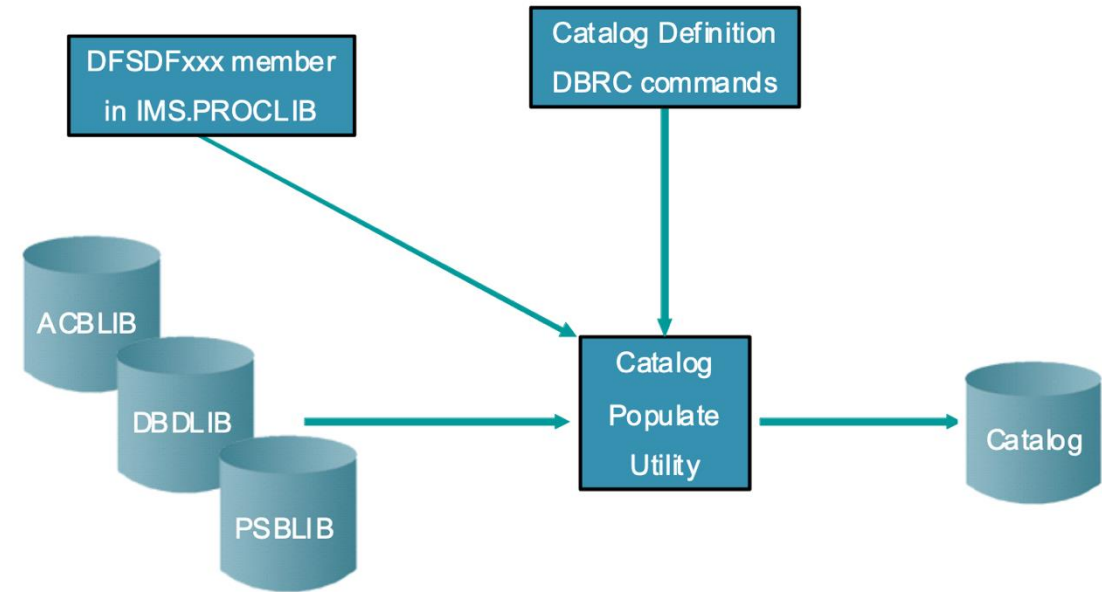
Implementing the IMS Catalog

- Add catalog DBDs and PSBs to the DBDLIB, PSBLIB and ACBLIB.
- ACBGEN the catalog DBD and PSB resources into the ACBLIB.
- Modify DFSDFxxx PROCLIB member.
- Define the catalog database into DBRC.
- Create HALDB structure, minimum of 1 partition.

```
*-----* 00010000
* CATALOG SECTION * 00020000
*-----* 00030000
<SECTION=CATALOG> 00040000
CATALOG=Y /* CATALOG IS ON */ 00050000
ALIAS=DFSC 00060000
RETENTION=(VERSIONS=5,DAYS=365) 00070000
STORCLAS=BASE 00080000
MGMTCLAS=STANDARD 00090000
SMSVOLCT=1 00100000
SPACEALLOC=(PRIMARY=500 SECONDARY=50) 00110000
GURCACHE=5 00120000
```

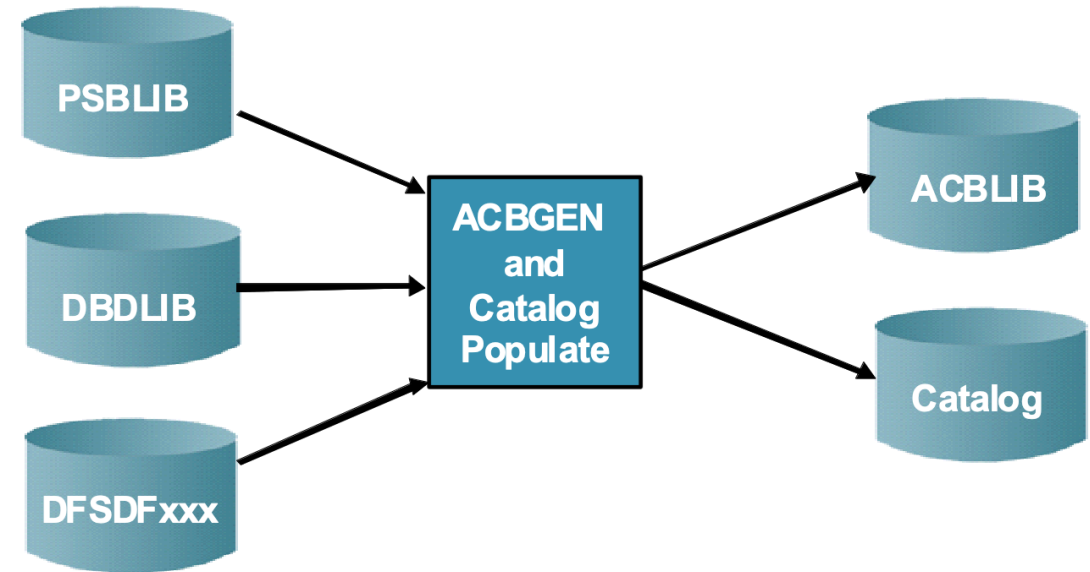
Loading the IMS catalog

- IMS Catalog Populate utility (DFS3PU00).
- Load, insert, or update DBD and PSB instances into the database data sets of the IMS catalog from ACB library data sets.
- Load mode DFSCPL00.
- Update mode DFSCP001.
- Analysis-only (read-only) mode DFSCP000.



Keeping the catalog and ACB libraries in sync

- ACB Generation and IMS Catalog Populate utility (DFS3UACB).
- Performs both the generation of ACB members in an IMS.ACBLIB data set and the creation of the corresponding metadata records in the IMS catalog.



IBM **ibm.ibm_zos_ims**

Version 1.3.0 updated 1 year ago (latest) Last updated 1 year ago

5 56,407 Downloads

Install Documentation Contents Import log Dependencies

[Docs site](#) [Website](#) [Issue tracker](#) [Repo](#)

Find content

Documentation (1)

Readme

Modules (8)

- ims_acb_gen
- ims_catalog_populate
- ims_catalog_purge
- ims_command
- ims_dbd_gen
- ims_dbrc
- ims_ddl
- ims_psb_gen

IBM z/OS IMS collection

The **IBM z/OS IMS collection**, also represented as **ibm_zos_ims** in this document, is part of the broader offering **Red Hat® Ansible Certified Content for IBM Z**. The IBM z/OS IMS collection supports tasks such as generating IMS Database Descriptors (DBD), Program Specification Blocks (PSB), Application Control Blocks (ACB), and running IMS type-1 & type-2 commands.

The **IBM z/OS IMS collection** works closely with offerings such as the [IBM z/OS core collection](#) to deliver a solution that will enable you to automate tasks on z/OS.

Red Hat Ansible Certified Content for IBM Z

Red Hat® Ansible Certified Content for IBM Z provides the ability to connect IBM Z® to clients\' wider enterprise automation strategy through the Ansible Automation Platform ecosystem. This enables development and operations automation on Z through a seamless, unified workflow orchestration with configuration management, provisioning, and application deployment in one easy-to-use platform.

The **IBM z/OS IMS collection**, as part of the broader offering **Red Hat® Ansible Certified Content for IBM Z**, is available on Galaxy as community supported.

For **guides** and **reference**, please visit [the documentation site](#).

Features

The IBM IMS collection includes [modules](#), and ansible-doc to automate tasks on IMS.

Ansible version compatibility

This collection has been tested against the following Ansible versions: >=2.14.0,<2.17.0.

Copyright

Let's implement
the IMS Catalog in
minutes...



Relevant links

- Z Samples repository: https://github.com/IBM/z_ansible_collections_samples
- Ansible for Z Intro repo, includes the demo playbooks: <https://github.com/IBM/z-ansible-workshops>
- Unified documentation for the collections: https://ibm.github.io/z_ansible_collections_doc/index.html