



# IBM Training

## IBM System Automation for z/OS 4.1 Architecture

### Student Notebook

Course code SM917 ERC 1.2

May 2019



**z Systems software**

All files and material for this course are IBM copyright property covered by the following copyright notice.

© Copyright IBM Corp. 2019. All Rights Reserved.

US Government Users Restricted Rights: Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

IBM, the IBM logo, and ibm.com are 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" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this publication to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results.

<b>About this course .....</b>	<b>xiii</b>
About the student .....	xiv
Learning objectives .....	xv
Course agenda .....	xvi
<b>1 IBM System Automation for z/OS 4.1 Architecture .....</b>	<b>1</b>
Objectives .....	2
Lesson 1 Overview .....	3
Overview .....	3
Business issues .....	4
Automation evolution .....	6
System Automation for z/OS .....	8
System Automation for z/OS: Components .....	10
System Automation V4.1 (GA 3/17) .....	12
Continuous enhancements (post-GA service-level) .....	16
Lesson 2 Component Processor Operations .....	17
Component processor operations .....	17
Component processor operations .....	18
z Systems external automation and single point of control .....	20
ProcOps Hardware Management Console tasks .....	21
Do your own smart capping and load distribution .....	22
Lesson 3 Component System Operations .....	25
Component System Operations .....	25
Component System Operations overview .....	26
What can you automate? .....	28
Summary of key automation technologies .....	30
Architecture .....	32
Key automation features .....	33
Grouping concepts .....	36
Example: Relationships: Starting an application .....	37
Automation flow .....	38
Lesson 4 Goal driven automation .....	40
Goal driven automation .....	40
Lesson 5 Policy-based automation .....	42
Policy-based automation .....	42
What is the power of an automation policy? .....	43
>800 predefined, selectable messages .....	45
Best practices policy for 270 applications and application classes out of the box .....	47
Policy reports provide automation documentation .....	49
Lesson 6 Key operations features .....	50
Key operations features .....	50
Key operations features .....	51
User interfaces .....	53
Service Management Unite: User interface .....	54
Key capabilities .....	56
“Domain and Automation Health” dashboard .....	58
Domain page .....	59
Service Management Unite V1.1.5 integrates with Zowe and more .....	61

Single Point of Control and Single System Image .....	64
Operator interface: NetView dialogs .....	65
Operator user interface: Status Display Facility .....	67
Example of a system status summary .....	69
Operator user interface: Tivoli Enterprise Portal .....	71
Tivoli Enterprise Portal workspace .....	73
Tivoli Enterprise Portal resource topology .....	75
Lesson 7 Key integration capabilities .....	77
Key integration capabilities .....	77
System Automation Integration Points .....	79
Lesson 8 More Automation .....	80
More Automation .....	80
End-to-end automation .....	82
Application pacing .....	83
OMEGAMON looping address space suppression .....	84
Hey, your IPL is complete or takes longer .....	85
Sysplex automation .....	87
Lesson 9 Configuration assistant .....	89
Configuration assistant .....	89
Traditional approach .....	90
Configuration assistant .....	91
Configuration assistant: Concept .....	92
Summary .....	93
Unit Summary .....	95
<b>2 Architecture and concepts .....</b>	<b>96</b>
Objectives .....	97
Lesson 1 Architecture .....	98
Architecture .....	98
What is a software resource? .....	99
Architecture: Manager-agent design details .....	101
Automation manager basics .....	103
Important characteristics .....	105
Automation manager flags .....	106
Automation manager environment .....	108
Architecture details .....	110
The role of the automation agent .....	112
Automation agent overview .....	114
The automation agent structure: Simplified .....	116
Message flow .....	118
The role of the operator .....	120
Automation in action .....	121
Lesson 2 Goal driven automation .....	123
Goal driven automation .....	123
What is goal driven automation .....	124
Goal driven automation scenario .....	126
Defining and overriding goals .....	127
The path from request to order .....	131

Persistence rules for requests .....	132
The request process .....	133
Suspending resources .....	135
Suspending resources workflow .....	137
Suspended resources .....	138
Lesson 3 Policy-based automation .....	140
Policy-based automation .....	140
Automation policy features .....	141
Automation resources .....	143
Automation policy for applications .....	144
Starting applications .....	146
Stopping applications .....	147
Lesson 4 Monitoring and status .....	149
Monitoring and status .....	149
Passive monitoring .....	150
Active monitoring .....	151
Automation agent status .....	152
Agent statuses and transitions .....	154
Key status meanings .....	156
Automation manager resource status .....	157
Observed status values .....	158
Desired status values .....	160
Startability status values .....	162
Automation status values .....	163
Health status values .....	164
Suspended flag values .....	166
Compound status values .....	167
Mapping agent status to manager status values .....	168
Application lifecycle and related statuses .....	169
Lesson 5 Relationships .....	170
Relationships .....	170
How relationships work (1 of 2) .....	171
How relationships work (2 of 2) .....	172
Relationships example .....	174
List of relationships .....	175
List of conditions .....	176
Lesson 6 Service periods .....	178
Service periods .....	178
Service period example .....	180
Overriding service periods .....	181
Service period resource override example .....	182
Lesson 7 Application groups .....	183
Application groups .....	183
Application groups (1 of 2) .....	184
Application groups (2 of 2) .....	185
Group type and nature .....	186
Lesson 8 More automation policy .....	188
More automation policy .....	188

Defaults .....	189
Major and minor resources .....	191
Agent automation flags .....	193
Agent automation flags .....	195
Threshold processing .....	197
Threshold processing and minor resources .....	199
Message assignment .....	201
Notify operators .....	203
Message policy .....	204
Message policy supports .....	205
Message capturing .....	207
Finding captured messages .....	209
Value returned is severity .....	209
Runmodes .....	210
Runmode scenarios .....	212
Characteristics of runmodes .....	213
Pacing gates motivation .....	214
Application Pacing – how does it work? .....	215
Pacing Gate policy .....	216
Pacing Gate approach.....	217
Lesson 9 End-to-end automation .....	218
End-to-end automation .....	218
End-to-end automation functions .....	219
End-to-end automation architecture .....	220
End-to-end automation concepts .....	221
End-to-end automation concepts .....	223
End-to-end automation architecture details .....	225
Cross sysplex automation example .....	226
End-to-end adapter .....	227
End-to-end agent .....	229
REF resources on SMU .....	230
Summary .....	231
<b>3 Details .....</b>	<b>233</b>
Objectives .....	234
Lesson 1 Monitor resources .....	235
Monitor resources .....	235
Monitor resources overview .....	236
Monitor resources overview (continued) .....	237
Characteristics of active and passive monitoring .....	238
Monitor resources at a glance .....	239
Health status criteria example .....	240
Lesson 2 Triggers and events .....	241
Triggers and events .....	241
Events .....	242
One trigger and several events example .....	243
Unsetting events .....	244
UNSET=DOWN processing .....	245

Lesson 3 MVS and JES automation .....	246
MVS and JES automation .....	246
Managing z/OS logs, data sets, and console buffers .....	247
Spool management .....	248
Lesson 4 Dependency relationships details .....	249
Dependency relationships details .....	249
Relationships .....	250
How dependencies affect the request process .....	252
Characteristics of active and passive dependencies .....	254
MakeUnavailable example .....	256
HasParent relationship .....	257
Examples of external relationships .....	259
Combining dependency relationships .....	261
Lesson 5 The order process .....	262
The order process .....	262
Starting applications revisited .....	263
Starting applications: Prepare available order .....	264
Starting applications: Make available order .....	265
MakeAvailable order scenario .....	266
Stopping applications revisited .....	268
Stopping applications: Prepare unavailable order .....	270
Stopping applications: Make unavailable order .....	271
Actions after an IPL .....	272
MakeUnavailable scenario .....	274
Unexpected shutdown or ABEND scenario .....	276
Lesson 6 Factors that can influence the request process .....	277
Factors that can influence the request process .....	277
Where do votes come from? .....	278
Votes priority and life-span .....	279
The winning vote .....	280
MakeAvailable/WhenAvailable with Scope Only .....	281
MakeAvailable/WhenAvailable and HasParent with Scope All .....	282
How the winning vote is chosen .....	283
Request sources .....	284
Persistence rules for requests .....	285
Request priority scheme .....	286
Request priorities by source .....	287
MakeAvailable vote propagation example .....	288
Inhibitors .....	289
Possible inhibitors .....	290
Location of inhibitors in the request flow .....	293
Lesson 7 Application group details .....	294
Application group details .....	294
Characteristics of an application group .....	295
Application group members .....	297
Characteristics of system application groups .....	298
Characteristics of systplex application groups .....	299
Group behavior: Active or passive .....	300

Nature of groups .....	301
Behavior of groups with basic nature .....	302
Status values for basic groups .....	303
Vote propagation in basic groups .....	304
Factors that influence the behavior of server groups .....	305
Behavior of groups with server nature .....	307
Status values for server groups .....	308
Behavior of groups with move nature .....	309
Initiating application moves .....	310
Summary of group availability attributes .....	311
Preference values and their effect on group policy .....	312
Automation manager bonus values .....	314
Effective preference value tie breaker .....	315
Group modes .....	316
Example of a sticky move .....	317
<b>Lesson 8 Transient resources .....</b>	<b>319</b>
Transient resources .....	319
Startup for transient jobs .....	320
Summary .....	321
<b>4 Processor Operations .....</b>	<b>322</b>
Objectives .....	323
<b>Lesson 1 Processor Operations architecture .....</b>	<b>324</b>
Processor operations architecture .....	324
Component processor operations .....	325
z Systems external automation and single point of control .....	327
Processor operations building blocks .....	328
Processor operations architecture .....	331
Processor operations architected interfaces .....	333
Processor operations automation policy .....	336
Task structure .....	337
<b>Lesson 2 Details .....</b>	<b>338</b>
Details .....	338
Target system status (1 of 3) .....	339
Target system status (2 of 3) .....	340
Target system status (3 of 3) .....	341
Built-in automation: IPL .....	342
Target system grouping .....	344
Target system status and alerting .....	345
Priority messages (Synchronous WTORs) .....	346
<b>Lesson 3 Processor Operations scenarios .....</b>	<b>348</b>
Processor operations scenarios .....	348
Frequent processor operations scenarios .....	349
Processor operations usage areas .....	350
Status display facility: Main panel .....	351
Status display facility .....	352
Status display facility: Target hardware .....	353
ProcOps messages in netlog .....	354

Getting started . . . . .	355
Target system overview: ISQXDST . . . . .	356
Target system summary . . . . .	358
Hardware communication sessions . . . . .	359
Sample task: Issue operating system commands . . . . .	360
Sample task: Query / Update LPAR settings (1 of 2) . . . . .	361
Sample task: Query / Update LPAR settings (2 of 2) . . . . .	362
Sample task: Query target image information . . . . .	363
Sample task: Activation profile management (1 of 2) . . . . .	364
Sample task: Activation profile management (2 of 2) . . . . .	365
ProcOps Hardware Management Console tasks . . . . .	366
ProcOps: Hardware messages task . . . . .	367
ProcOps: Operating system messages task . . . . .	368
ProcOps: Hardware commands tasks . . . . .	369
ProcOps: Actions on groups . . . . .	370
ProcOps: Hardware recovery tasks . . . . .	371
ProcOps: Test task for z/OS . . . . .	372
ProcOps: Hardware LPAR control tasks . . . . .	373
ProcOps: Activation profiles tasks . . . . .	374
ProcOps: Hardware group capacity tasks . . . . .	375
ProcOps: Hardware exclusive control . . . . .	376
ProcOps: Hardware time-related tasks . . . . .	377
ProcOps: Hardware temporary capacity upgrade tasks . . . . .	378
ProcOps: Hardware energy control tasks . . . . .	379
ProcOps: Hardware query tasks . . . . .	380
ProcOps: Hardware security log monitoring . . . . .	381
ProcOps: Shutdown or restart HMC task . . . . .	382
Do your own smart capping and load distribution . . . . .	383
Automatic CPC and LPAR capacity changes . . . . .	386
Programming considerations: processor operations . . . . .	388
Summary . . . . .	389



# About this course

---



## IBM System Automation for z/OS 4.1 Architecture



© Copyright IBM Corporation 2018

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This course introduces and explains the System Automation for z/OS (SA z/OS) components, architecture, and concepts. Focus is on the system operations component with Processor Operations as an optional topic. It is a prerequisite for attending the operations SM927 or administrator SM937 classes.

This course does not include labs.

For information about other related courses, see the IBM education training paths website:

[ibm.com/training/](http://ibm.com/training/)

Details	
<b>Delivery method</b>	Classroom or instructor-led online (ILO)
<b>Course level</b>	ERC 1.2 This course level is an update. The original course 1.0 was new.
<b>Product and version</b>	IBM System Automation for z/OS 4.1 Including updates June 2018: APAR OA55386 and Service Management Unite 1.1.4
<b>Duration</b>	2.5 days
<b>Skill level</b>	Basic



**Note:** This course applies to IBM System Automation for z/OS (System Automation), but can be also used for IBM Automation Control for z/OS. Functions not provided by IBM Automation Control for z/OS are in *underlined italic*. The terms automation and the automation product are used when there are no differences. Alternatively the product name is used to describe its functions.

## About the student

This course is designed for users of IBM System Automation for z/OS, especially operators, administrators, and system programmers. It is the required prerequisite course of all other IBM System Automation for z/OS courses and should be attended first.

Before taking this course, make sure that you have the following skills:

- Basic z/OS skills
- Basic NetView skills

Recommended courses before this course are:

- *TZ203 - IBM Tivoli NetView for z/OS 6.1 Fundamentals*
- *TZ213 - IBM Tivoli NetView for z/OS 6.1 Automation*

# Learning objectives

The learning objectives are to get the required knowledge to attend the System Automation for z/OS operations (SM927) or administration (SM937) classes:

- Purpose and features
  - Major components
  - Architecture and design of the system operations component
  - Policy-based automation and concepts
  - Goal driven automation
- 

## Objectives

---

In this course, you learn to perform the following tasks:

- Describe the IBM System Automation for z/OS 4.1 architecture
- Describe the components of the product
- Describe the product's automation capabilities
- Describe policy-based and goal driven automation
- Explain the key automation concepts and automation policy for applications
- Explain operator interfaces including Service Management Unite
- Explain the request process, inhibitors, and orders
- Describe product details like resource relationships, groups, threshold processing, message automation...
- Describe end-to-end automation
- Describe Processor Operations architecture and implementation options

---

© Copyright IBM Corporation 2018

# Course agenda

The course contains the following units:

1. [IBM System Automation for z/OS 4.1 Architecture](#)

- Describe IBM System Automation for z/OS and IBM Automation Control for z/OS and their capabilities
- Describe the components of the product
- Describe policy-based and goal driven automation
- Describe the product's automation capabilities
- Describe its key operations features
- Describe its integration, additional automation, end-to-end automation, and the configuration assistant

2. [Architecture and concepts](#)

- Describe the architecture
- Describe the automation agent role and operation
- Describe the automation manager role and operation
- Explain the key automation concepts
- Describe goal driven automation
- Explain the automation statuses and their effect on automation
- Describe automation policy for applications
- Describe resource dependencies and relationships
- Provide an overview of application groups
- Explain automation flags, threshold processing, message policy, and notify operators
- Describe end-to-end automation architecture

### 3. Details

- Describe a monitor resource and its effect on the health status of linked resources
- Describe events and triggers
- List MVS automation
- Describe resource relationships details
- Explain the order process
- List and describe factors that can influence goal-driven automation
- List types and natures of application groups
- Explain behavior or attributes of application groups
- List transient resource automation

### 4. Processor Operations

- Describe architecture including implementation options
- Explain usage and operator interface
- Describe automation policy





# 1 IBM System Automation for z/OS 4.1 Architecture

---

IBM System Automation for z/OS 4.1



## Unit 1 IBM System Automation for z/OS Overview



© Copyright IBM Corporation 2019  
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

# Objectives

- When you complete this unit, you can perform the following tasks:
  - Describe IBM System Automation for z/OS and IBM Automation Control for z/OS and their capabilities
  - Describe the components of the product
  - Describe policy-based and goal driven automation
  - Describe the product's automation capabilities
  - Describe its key operations features
  - Describe its integration, additional automation, end-to-end automation, and the configuration assistant

- Note:

This course applies to IBM System Automation for z/OS (System Automation), but can be also used for IBM Automation Control for z/OS. Functions not provided by IBM Automation Control for z/OS are in *underlined italic*. The terms automation and the automation product are used when there are no differences. Alternatively the product name is used to describe its functions.

## Objectives

This unit covers the automation product overview, its components, policy-based and goal driven automation, key automation and operations features, integration, additional automation, end-to-end automation, and the configuration assistant.



**Note:** This course applies to IBM System Automation for z/OS (System Automation), but can be also used for IBM Automation Control for z/OS. Functions not provided by IBM Automation Control for z/OS are in *underlined italic*. The terms automation and the automation product are used when there are no differences. Alternatively the product name is used to describe its functions.

# Lesson 1 Overview

## Lesson 1: Overview

- Business issues and automation evolution
- IBM System Automation for z/OS overview
- IBM System Automation for z/OS components
  - Processor Operations
  - System Operations

© Copyright IBM Corporation 2019

1-3

### Overview

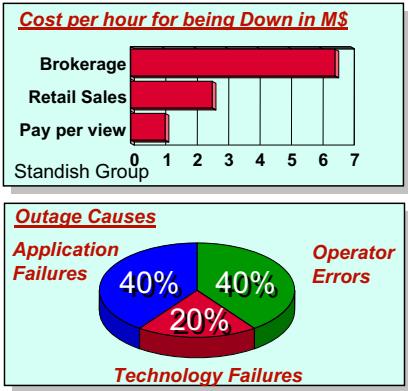
This lesson starts with the business issues that the automation product can help solving, followed by the automation evolution.

The key automation capabilities are explained.

The two components of the automation product, **processor operations** and **system operations** are introduced.

New features and enhancements in System Automation V4.1 plus the 3Q18 updates are also summarized.

# Business issues



Application downtime affected by:

- Application complexity
- Application dependencies
  - CICS, DB2, IMS, network, and other systems
- Heterogeneous environments

## IT Pressures

- Application availability
- Operations complexity and costs
- Skills and education requirements
- Automation implementation and maintenance costs
- Rapid change of IT infrastructure

- **Loss** of business
- **Loss** of customers – the competition is just a mouse click away
- **Loss** of credibility, brand image, and stock value



1-4

## Business issues

On this slide, you see how outages and complexity can affect your business. The pains are similar to many issues that IBM is trying to solve by its performance and availability products. The main topic is that IBM clients, especially in the online business environment, can no longer afford downtime. So, it is not just a question about losing business, it is a question about losing customers and losing reputation. What you see here is that the costs in brokerage houses are about \$6.5 million for one hour of being down.

Many IBM clients are starting to deploy new workloads, like SAP, like WebSphere, and similar customer relationship management tools. They have a need here to react quickly to business needs. So, they get changes very fast and they must react very fast to get them deployed on the systems.

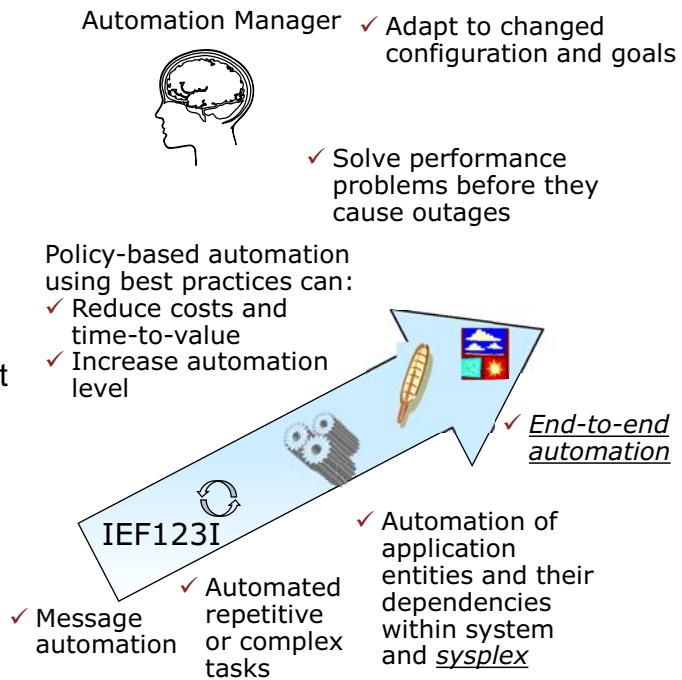
The new workloads are often complex by themselves, already. Complexity is further increased by prerequisites like messaging, networking, directory, and database access and by the need to make them web-based. Based on the tools and the processes clients use currently, it is very hard to figure out how the failure of a resource affects related applications. Let's assume that a certain component in an SAP system crashes or your database is no longer working. Does this failure impact your whole business transaction and what are the right steps to get back online.

So, do you just have to restart the component or do you have to bring down the whole application and start right then. That is all very complex. You face many situations where the people working on those problems don't have a real understanding of all of the relationships of such an application resource, how to repair it, and what it means to the business. On the left side, you see a pie chart and that 40% of all issues are caused by operator errors due to complexity.

So overall, it all comes down basically to a cost of ownership, speed to market, and competitiveness issue. So you either have to spend much money, or you lose customers against the competition, and lose business. Therefore, you must tackle this problem. Automation is the way to address some of those issues.

# Automation evolution

1. Message filtering
2. Reactive message automation
3. Command-driven resource management
  - Start, stop, recycle
  - Parent-child dependency
4. **Policy-based** automation (versus scripts)
5. **Goal driven** application management
  - Cluster-wide
  - Complex dependencies
6. Disaster recovery
7. **Pro-active**, health-based automation
8. End-to-end automation
9. Adaptive automation



© Copyright IBM Corporation 2019

1-5

## Automation evolution

The latest stages of evolution are:

- Understanding health of systems and applications to enable pro-active automation
- Composite application management across the enterprise
- Adaptive automation adapts to changed configuration and goals

Automation started in the late 80's with **message suppression** and simple **message automation**.

Both are still important as message rates increase with server size.

The next stage was **command-driven resource management** that has a resource model that allows to start, stop, and recycle resources according to simple parent-child dependencies between resources.

More advanced products, like System Automation for z/OS, reduce the coding effort by policy-based automation and enable management at the business application level. Operators are not required to remember application components and dependencies.

It offers a simple human interface for complex tasks, like system IPL or shutdown, that can be handled with a single command.

Applications can be started and stopped in a consistent way.

**Policy-based automation** using best practices can reduce costs and time-to-value and increase the automation degree.

**Goal driven automation** greatly simplifies operations: The operator just requests whether applications should be up or down, automation takes care of any dependencies and resolution of affected or even conflicting goals.

The next stage extended high availability into **disaster recovery** by using automation as a platform for disaster recovery and by synchronizing data, system, and application recovery.

The next stage of the evolution was understanding the health of system and applications to enable **pro-active automation** that can solve performance problems before they cause outages. A monitor concept allows to integrate with monitors to update the health state. Integration with OMEGAMON allows to send exceptions to automation, which in turn can retrieve more performance data for drilling down to the root cause.

With System Automation for z/OS V3, integration of z/OS into **end-to-end automation** of heterogeneous business applications became possible. This allows to have one enterprise-wide automation and operations team. Initially end-to-end automation required IBM Tivoli System Automation Application manager but System Automation for z/OS V4.1 now also provides end-to-end automation.

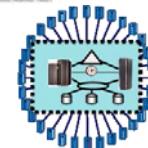
An operator can use a browser to manage composite applications that include z/OS applications.

Adaptive automation started already with goal driven and cluster-wide automation. It can adapt to changed configuration, like an offline system or application and to changed goals, like avoiding a system or requesting an application to be up.

**Adaptive automation** uses the cluster-wide status of systems and applications, the goals defined in the policy, the goals modified or created by the operator to adapt the application status so that the goals are met as much as possible.

# System Automation for z/OS

- High availability for z/OS, z/OS UNIX, and distributed resources
- **Policy-based automation** using best practices
- Automated and easier mainframe operation through **goal driven automation** and **reduced complexity**
- **State-of-the-art GUI: Service Management Unit**
- Basis for disaster recovery with Geographically Dispersed Parallel Sysplex (GDPS)
- Parallel Sysplex single system image and automation
- **Pro-active automation** through monitor and OMEGAMON integration
- Comprehensive integration
- **External automation** to initialize, configure, change capacity, recover, and shut down IBM z Systems



1-6

© Copyright IBM Corporation 2019

## System Automation for z/OS

Automation provides high availability for z/OS applications including UNIX System Services (USS) and cross sysplex and cross platform resources.

## Policy-based automation

**Policy-based automation** is a sophisticated methodology that allows to easily incorporate business goals into an automation framework. Policy-based automation can help reduce complexity, implementation time, coding, support effort, and can replace scripts.

IBM ships plug-and-play automation that includes **best practices** automation for many z/OS applications. This provides the following advantages:

- Reduce time and effort in creating a new policy or updating one
- Improve automation quality and reduce human error

## Goal driven automation and reduced complexity

**Goal driven automation** greatly simplifies operations. System operators just request what they want. Automation takes care of any dependencies and goals that are affected. It can resolve

conflicting goals and even can remember goals. Automation provides accelerated startup and shutdown, and correct recovery by managing relationships between resources.

**Grouping of resources** and definition of aggregate or business applications can greatly reduce the complexity of automation definition and operations.

## **Service Management Unite**

IBM Service Management Unite (SMU) is a customizable service management user interface that provides dashboards to monitor and operate z systems environments. It provides a single point of control to monitor and operate in your environment. Operators can quickly and confidently analyze, isolate and diagnose problems as all relevant data including important logs is provided in a single place. Service Management Unite Automation also enables operators to interact directly with the system by issuing commands and viewing results without going to a different console.

## **Parallel Sysplex and disaster recovery**

NetView and System Automation for z/OS are the basis for disaster recovery with Geographically Dispersed Parallel Sysplex (GDPS).

Automation is sysplex-wide. Operators can operate from a single point of control and the sysplex appears as a single system image.

IBM Automation Control for z/OS can only automate a single system.

The single point of control of IBM Automation Control for z/OS is limited to three systems.

## **Integration**

Monitor resources enable integration with any monitor and include easy to exploit OMEGAMON access to update the health status of monitor resources, which can trigger **pro-active automation**.

On top of the rich set of NetView interfaces, comprehensive **integration** exists with z/OS components, for example WLM and ARM, systems management products, for example IBM Workload Scheduler, System Automation for Integrated Operations Management, and Netcool/OMNIbus.

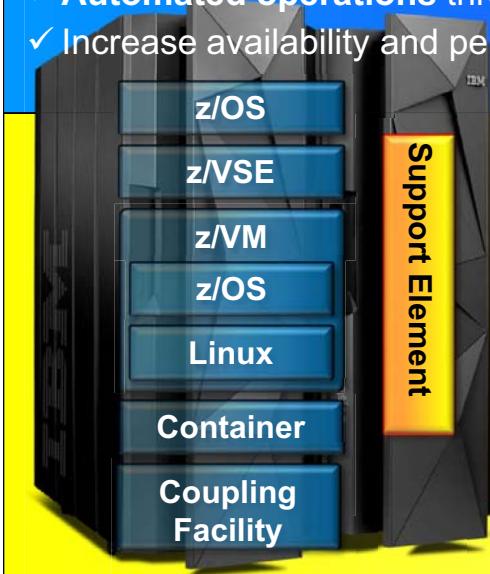
## **External automation**

**External automation** provides visualization, control, and automation of z/OS systems and IBM z Systems, z/VM, and its guests. Automation Control for z/OS is limited to the local system.

# System Automation for z/OS components

## System Operations

- ✓ Automate repetitive and complex tasks in **z/OS systems and end-to-end**
  - ✓ Helps operators to perform their tasks in a more robust way
  - ✓ Reduces z/OS specific skill requirements
- ✓ **Automated operations** through monitoring of applications and messages
- ✓ Increase availability and performance through pro-active automation



## Processor Operations

- ✓ z Systems external automation and single point of control
- ✓ **Faster LPAR startup**, recovery, shutdown
- ✓ **Save software costs** through LPAR capacity management
- ✓ **Higher availability**: Don't miss any hardware alerts
- ✓ **Save energy** with power management

### *System Automation for z/OS: Components*

System Automation for z/OS has two key components that are designed to automate processor and system operation. These components are called processor operations and system operations.

## ***The system operations component***

The **system operations** component and NetView as the prerequisite automation platform, can help operators by automating many system console operations and selected operator tasks such as startup, monitoring, recovery, and shutdown of z/OS subsystems, components, applications, UNIX (USS), and sysplex resources. The system operations component can also automate operator console messages, messages in the job log, initiate timer-based actions, and prevent shortages of critical z/OS resources.

The automation product integrates with **OMEGAMON XE** to help you to increase availability and performance through proactive automation.

## ***The processor operations component***

**Processor operations** helps operators manage more systems with greater efficiency. This means that one operator, even from a remote location, can configure, initialize, monitor, shut down, and

recover a central processor complex (CPC), Logical Partitions (LPAR), and multiple systems in parallel and respond to various detected conditions. An operator, using one standard interface, can do all that across multiple types of systems such as z/OS, z/VSE, z/VM, and Linux.

Clients can save software costs through LPAR capacity management, for instance by adjusting capacity across LPARs and WLM capacity groups automatically.

Hardware alerts and priority messages that must be responded to in a matter of seconds can be automated to increase availability.

Save energy with power management based on time of day or system load.



**Note:** The I/O Operations component is not covered as V3.5 was the last release supporting it.

# System Automation V4.1 (GA 3/17, \* updates)

- Service Management Unite (SMU): A new way to operate your mainframe
  - Highly customizable, including possibility to create custom dashboards
  - Integrates with many other UIs: SMU Performance Management, IBM Workload Scheduler, OMNIbus, APM, Zowe ...
  - Allows monitoring and controlling of distributed apps through Universal Automation Adapter
  - \* Highly available; installation with Docker; Ask Watson; ...
- Cross sysplex and end-to-end automation \*
  - Automate resources in another sysplex or on other platforms
- Suspend and resume resources initially \* and dynamically by operator
- State analysis: Why is the Desired Status not reached?
- Guaranteed delivery of EIF events
- New policies for z/OS Connect EE and OMEGAMON for JVM
- I/O Operations discontinued: Note: V3.5 is the last release supporting I/O Operations
- V4.1.0 is the last release to support IBM zEnterprise BladeCenter Extension
- APAR OA55386 June 2019 adds also: INGPLEX IPL, Parallel SDF updates, ProcOps dynamic target system names...



*System Automation V4.1 (GA 3/17)*

## New features and enhancements

Version 4.1.0 provides enhancements that are focused on a completely new user experience, with a modernized management console to manage applications running beyond Parallel Sysplex barriers. This offering is designed to minimize users' efforts to reduce the costs of IT operation, lower risks, and improve system and application availability. New enhancements include:

- The introduction of Service Management Unite, a modernized graphical user interface that enables you to better operate and automate your application and resource management.
- Enhanced scope to automate applications and resources on multiple sysplexes without the need to manage dependencies manually.
- Capability to manage and automate critical applications running on different sysplexes with transparent and easy-to-use Service Management Unite dashboards.
- A new System Automation for z/OS analyze function (INGWHY) that provides expert guidance to operators, when an unexpected status of an automated resource appears. The analysis is valuable in error situations when operators must quickly restore service.

A new and easy way to temporarily suspend automation for specific resources and their dependents without impacting the operations team by generating false alarms. New resources can be suspended initially using the suspend file.

## More enhancements

- JobLog Monitoring that provides the following features:
  - Is enhanced to detect when a data set is spun off by JES. As a result, it monitors the new spool data set.
  - Can resume monitoring where it was before when NetView must be recycled.
- Alerts that can be sent by System Automation for z/OS with the NetView confirmed message adapter function, which can be used to guarantee delivery of alerts to the target of Event Integration Facility (EIF) events.
- System Automation for z/OS, which supports conditional operations on automation workstations jointly with the IBM Workload Scheduler product.
- New configurable peer grouping for SA-BCPii processor connections. This feature enables the optimization of the number of initial auto-connections, the elapsed hardware connection startup time, and the reduction of the SA-BCPii-related network traffic in a processor LAN environment.
- Processor operations connection start performance environments. New internal processor operations event registration logic optimizes the z Systems API calls at processor operations startup time (ISQSTART) and individual target connection starts (ISQXIII).
- New policies for z/OS Connect EE and OMEGAMON for JVM; updated policies for all OMEGAMON products; and enhanced policies that include support for SDSF and dynamic VIPA management.
- Supervision of the defined connections between IMS and DB2, IMS and MQ, CICS and DB2, and CICS and MQ.
- Smart commands and options in the customization dialog box to help optimize user productivity by quickly locating an existing object or creating objects in the policy.
- Flexibility to set and receive the correct JES2 message prefix for use in customer scripts when it differs from the default.

System Automation for z/OS, V4.1.0 is the last release to support IBM zEnterprise BladeCenter Extension (zBX) Hardware Management. I/O operations and the support of NetView Management Console (NMC) are not supported by System Automation for z/OS, V4.1.0. Both functions are removed from System Automation for z/OS, V4.1.0.

## New function APAR OA55386

SA z/OS V4.1.0 new function APAR OA55386 is available since June 29th, 2018. This APAR provides the following enhancements:

### Service Management Unite (SMU) Automation V1.1.4

- The installation of SMU is simplified with a prebuilt Docker image.
- "Ask Watson" dashboard is added as an open beta feature to provide a cognitive documentation search.
- To ensure a reliable system with high performance and less downtime, you can set up SMU with high availability.
- With Universal Automation Adapters, SMU Automation can automate applications that run on non-z/OS systems.
- SA operations experience is enhanced:
  - For a stop, start, or suspend request, you can choose the new REMOVE=SYSGONE option to automatically remove the request when the system where the selected resource runs, leaves the sysplex.
  - The resource status of a system is now represented as the worst compound status of all top-level resources running on that system. This can be combined with a Resource name filter or Resource class filter as data set parameter. In this case, the worst resource state is derived by the worst compound state of all resources on the system that match the specified filter criteria.
  - A Hide operational tasks option is added into the automation domain topology and automation node list data sets. Choose this option if the context menu of nodes that are contained in these data sets should not include any operational tasks, such as excluding a node.

### End-to-end automation

SA z/OS V4.1 extends its cross-sysplex automation capabilities to true end-to-end cross platform automation. Resources on distributed systems, for instance running on Linux, can be managed by the Universal Automation Adapter that is a part of Service Management Unite Automation.

### Processor operations

Processor operations is enhanced to support dynamic creation of target system names. The new AOF\_AAO\_ISQ\_DYNTGT option is introduced to define the dynamic name pattern. The benefit is that you can define a backup LPAR for another processor entry, without the need to define a corresponding system entry and the need for a 'dummy' system name.

## Other enhancements

- SDF now supports the status update of the primary focal point and the backup focal point in parallel.
- The new DSN parameter is added to the INGPLEX IPL command. It allows you to specify a different IPL data set for displaying and comparing IPL information. It might be useful in case of a disaster recovery when you need IPL information of the sysplex that is down.
- The new MAXINT parameter is added to the INGRCHCK command to limit the maximum times of resource status check. You can use it to avoid infinite resource check, which is task blocking

## Continuous enhancements (post-GA service-level)

- OA56547 – New user exit AOFEXC27 for the INGAUTO command (April 2019)
- OA56909 – GDPS Toleration Support (March 2019)
- OA56629 – Executing NetView Commands from TSO REXX Programs (3/2019)
- Service Management Unite (SMU) Automation V1.1.6 (March 2019)
- OA55859 – Allow alternative usage of RMTCMD for TWS automation (March 2019)
- SMU Automation V1.1.5 (December 2018)
- OA54684 Enhancements (November 2018)
- OA55159 – Additional IBM z14 Exploitation Support (November 2018)
- OA55386 – Multiple Enhancements (June 2018)
  - Service Management Unite (SMU) Automation V1.1.4
  - End-to-end automation
  - Processor Operations
- OA53587 – IBM z14 Exploitation Support (March 2018)
- OA54030 – INGRDS Command Enhancements (December 2017)
- OA53366 – Small enhancements (November 2017)
- OA52610 – Multiple Enhancements (September 2017)
- OA52638 – IBM z14 Toleration Support (July 2017)
- OA52425 – Small enhancements (July 2017)

[https://www.ibm.com/support/knowledgecenter/SSWRCJ\\_4.1.0/com.ibm.safos.doc\\_4.1/Continuous\\_enhancements.html](https://www.ibm.com/support/knowledgecenter/SSWRCJ_4.1.0/com.ibm.safos.doc_4.1/Continuous_enhancements.html)

© Copyright IBM Corporation 2019

1-9

*Continuous enhancements (post-GA service-level)*

This slide lists all the new functions or enhancements that are incorporated into SA z/OS V4.1, since it was generally available (GA) in March 2017.

For details go to:

[https://www.ibm.com/support/knowledgecenter/SSWRCJ\\_4.1.0/com.ibm.safos.doc\\_4.1/Continuous\\_enhancements.html](https://www.ibm.com/support/knowledgecenter/SSWRCJ_4.1.0/com.ibm.safos.doc_4.1/Continuous_enhancements.html)

# Lesson 2 Component Processor Operations

## Lesson 2: Component Processor Operations

- Processor Operations provides external automation of many z Systems operating systems
- z Systems hardware operator tasks and automation
- Automatic CPC and LPAR capacity changes

© Copyright IBM Corporation 2019

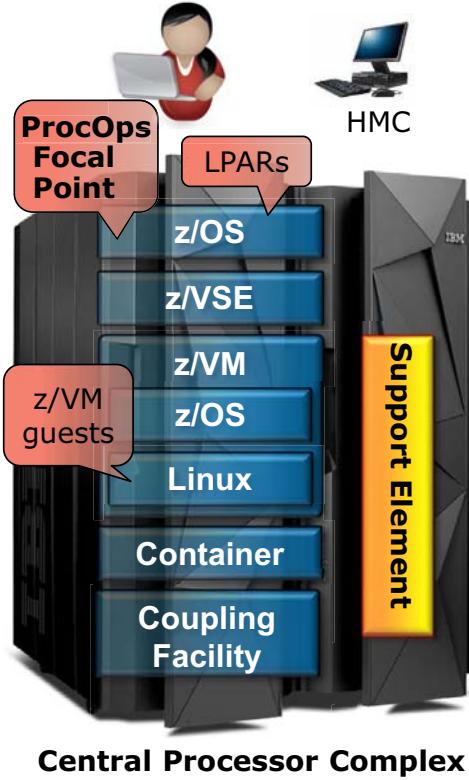
1-10

### *Component processor operations*

In this lesson component processor operations is described.

- Processor operations external automation capabilities that are important when internal automation is not available in z Systems operating systems
- Using processor operations for z Systems hardware operator tasks and automation
- Using processor operations for automatic CPC and LPAR capacity changes, a function that many clients are interested in

# Component Processor Operations



- Single point of control for z Systems
- LPAR startup, recovery, shutdown
- LPAR capacity and weight management
- Hardware status changes and alerts
- Power management
- z/VM guest system support
- NetView and SNMP based
- Secured through SNMP V3 encryption and SAF protected access

© Copyright IBM Corporation 2019

1-11

## Component processor operations

Component processor operations runs on the NetView platform and can monitor and control whole z Systems CPCs, logical partitions (LPARs), operating systems like z/OS, and virtual machines over the network.

Processor operations can help operators manage more systems with greater efficiency by providing z Systems **external automation** and a **single point of control**, which is easy to configure and easy to use. One operator in one location can initialize, configure, monitor, shutdown, and recover multiple systems, both local and remote, and respond to various detected conditions. Automation Control and System Automation for z/OS are the only products that provide one platform for internal and external automation from Power-On reset to activate, run time messages and commands at the system console and, for z/OS IPL, automation during the nucleus initialization program (NIP) stage when internal automation is not yet available.

Highlights include LPAR Capacity Management, power mode control, and support for Linux as a z/VM guest.

## SNMP connection

Management is done through SNMP connection either to the support element (SE) or the Hardware

Management Console (HMC) using the System z API.

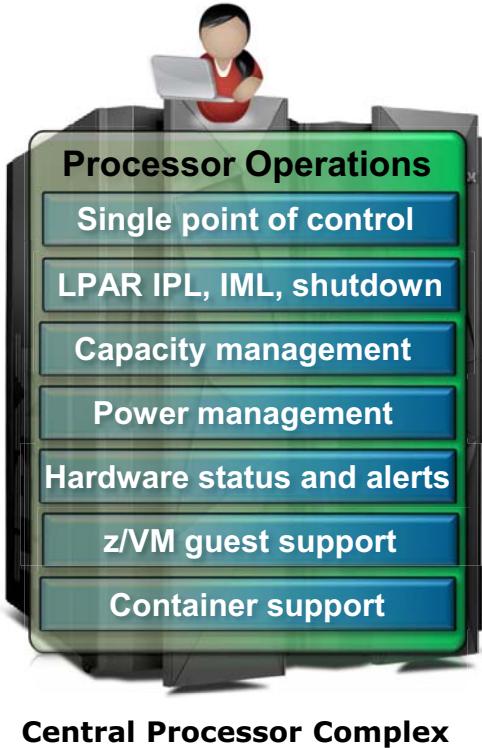
SNMP v2 (System Automation for z/OS 3.4 and older): Security has been the biggest weakness of SNMP since the beginning. Authentication in SNMP Versions 1 and 2 amounts to nothing more than a password (community string) sent in clear text between a manager and agent.

**Solution:**

SNMP v3 makes no changes to the protocol aside from the addition of cryptographic security:

- Confidentiality: Encryption of packets to prevent snooping by an unauthorized source.
- Integrity: Message integrity to ensure that a packet has not been tampered with in transit. It includes an optional packet replay protection mechanism.
- Authentication: To verify that the message is from a valid source.

## **z Systems external automation and single point of control**



### **External Automation**

- Power-on reset, activate, IPL (NIP time)...
- Respond to system or operator console messages including priority messages (synchronous WTOs)
- Detect and resolve wait states
- CPC capacity changes, powermode control
- Server Time Protocol and LPAR management

### **Ease to use common commands for:**

- CPCs, LPARs, systems, and z/VM guests
- z/OS, Linux, z/VM, z/VSE, containers

### **Easy to configure**

- Policy based

© Copyright IBM Corporation 2019

1-12

#### *z Systems external automation and single point of control*

You can perform Power-On reset, activate, **initial program load** (IPL), respond to hardware messages, monitor hardware status, and detect and resolve wait states.

Hardware alerts and **priority messages** (synchronous WTOs), some of which must be responded to in a matter of seconds can be automated to increase availability.

Manage , for instance by adjusting capacity across LPARs and WLM capacity groups automatically.

Save energy with **power management** based on time of day or system load.

Use the **Server Time Protocol** to improve time synchronization in a sysplex or non-sysplex configuration.

Using one standard interface, the operator can do all that across multiple types of systems. Automated routines for frequently used functions can speed the work of a skilled operator and help less-experienced operators become more productive.

Processor operations is easy to configure using automation policy that is integrated with system operations.

# ProcOps Hardware Management Console tasks

Daily	Recovery	Configuration	Timer	Console
Hardware Messages	Start All	Change LPAR Controls	System (Sysplex) Time	System Information
Operating System Messages	Stop All	Customize Activation Profiles		View Security Logs
Activate	Reset Normal	Change LPAR Group Controls	Temporary Capacity	
Reset Normal	PSW Restart	Perform Model Conversion		Shutdown or Restart
Deactivate	Reset Clear	Special Control	Energy Control	
Grouping	Load	Customize Console Services	Set Power Saving	
	Test			
	Interrupt			

Same console tasks supported as on the Hardware Management Console

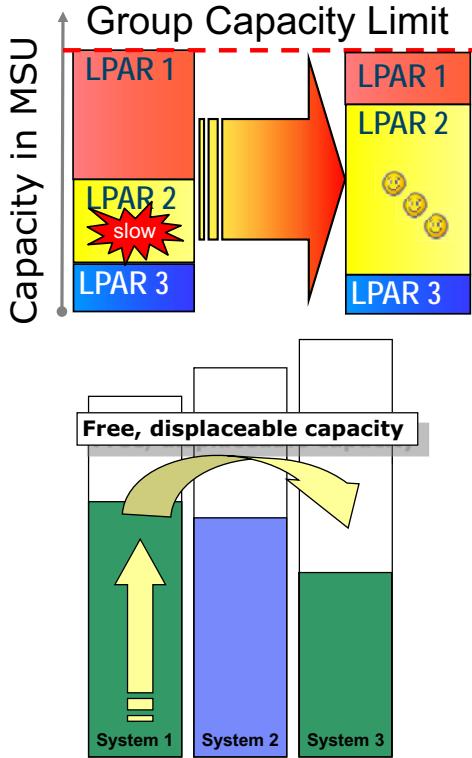
## ProcOps Hardware Management Console tasks

Processor operations supports the same console tasks as the Hardware Management Console or the support element including hardware messages, actions against groups, hardware commands, LPAR control, capacity value, activation profiles, exclusive control, Server Time Protocol, temporary capacity upgrade actions, query and set the CPC power mode, system information, viewing security logs, and restart or shutdown a SE or HMC.

Here are some examples what you can do:

- SYSRESET and ACTIVATE an LPAR
- LOAD image (might also be initiated by the Activation Profile)
- Manage ACTIVATION PROFILES
- Manage temporary capacity using either Capacity Backup (CBU) or On/Off Capacity on Demand (OOCoD) – a corresponding contract needs to be in place

## Do your own smart capping and load distribution



### Software cost challenges:

- Charges based on peak rolling 4-hour average or the LPAR Defined Capacity whichever is lower
- Capping might cap priority workload

### Solutions using SysOps:

- Move workload to under-utilized system using server or move groups based on
  - Predicted free capacity and WLM data
  - Using OMEGAMON metrics
- Policy-based looping jobs resolution

### Solution using ProcOps

- Adjust capacity across LPARs and WLM capacity groups automatically
  - Absolute capping supported

© Copyright IBM Corporation 2019

1-14

#### Do your own smart capping and load distribution

Many clients face software cost challenges that they try to resolve with subcapacity pricing for Variable Workload License Charges (VWLC) software that is based on monthly peak *rolling 4-hour* average (R4HA) usage measured in Millions of Service Units (MSUs) and LPAR size definitions like the *LPAR Defined Capacity*. When the R4HA becomes higher or equal to the Defined Capacity then the LPAR is capped by the *z/OS Workload Manager* (WLM) which slows down the system and priority workload.

Automation software can measure the four-hour rolling average, know when it's approaching the capping level, recognize peaks before they become peaks, and take appropriate action before the usage becomes a problem. These actions can smooth out usage spikes and, in fact, allow you to run at a lower capping level than otherwise practical, without sacrificing your service levels.

Therefore, you get the full benefits of subcapacity pricing.

## **Solutions using SysOps:**

- Moving workload to under-utilized system based on the predicted free capacity and WLM data or based on OMEGAMON metrics
  - Load situation of z/OS systems considered when moving applications when more than 1 system is eligible with the same preference
  - AM queries periodically WLM capacity of all systems
  - SA z/OS tries to estimate how the number of free MSUs will change after each start operation
- A policy-based solution for active suppression of looping address spaces

## **Solutions using ProcOps:**

Based on schedule, workload, or application, you can use processor operations to complete the following tasks:

- Adjust LPAR Weights of systems that are in the same capacity group
- Change LPAR defined capacity or group defined capacity
- Adjusting capacity across LPARs and WLM capacity groups automatically

## **Background:**

A System z LPAR can be assigned resources like CPU, memory, and weight for WLM LPAR CPU management.

The problem is that often applications are licensed by capacity, sometimes calculated as average. Clients often need temporary higher capacity, for example, for quarter end processing or for a marketing campaign. If they exceed their allowed capacity, they must either pay extra or reduce to a lower capacity to meet the average capacity they are licensed for.

## **4-hour rolling average (4HRA)**

The 4-hour rolling average is calculated by WLM. The 4-hour rolling average (SMF70LAC) is the average of the 48 last measures of instantaneous MSU (IMSU or ACTMSU, the real consumption).

## **Defined capacity**

Defined capacity is the size of an LPAR in MSUs that you set indicating the capacity of an LPAR. This is also referred to as a soft cap. The defined capacity specifies a maximum rolling 4-hour average consumption for an LPAR. This means that it is possible for the actual capacity being used by the LPAR at a given time to be higher than the defined capacity.

## Soft capping

'Soft capping' is a technique that is primarily intended to help you control your software costs. The process of limiting the amount of capacity that can be consumed by an LPAR (and as a result, controlling your software bills) is often called 'soft capping' or 'defined capacity' or capacity group limit. Capping starts when the rolling 4-hour average reaches the defined capacity. At that point, the LPAR is capped to the defined capacity. The WLM Capping% (SMF70NSW) represents the percentage of time during which the LPAR was soft-capped during the interval.

## Capacity groups

Capacity groups are an extension of the concept of defined capacities. However, whereas a defined capacity only applies to a single LPAR, a group capacity limit (as its name implies) specifies a limit on the rolling 4-hour average utilization for a group of LPARs. The share of the group capacity limit that each member of the group is entitled to is determined by that LPAR's relative weight compared to the other members of the capacity group, which is combined with a share of any capacity that other group members might not be using, also called unused capacity (UC). Capacity groups and defined capacities can be combined.

Most clients use capacity groups as they allow that a member can consume unused capacity of other members without extra charges. An LPAR can only be a member of one capacity group at a time, but it can be moved to another capacity group. All the members of the capacity group must be on the same CPC, but they do not need to be in the same sysplex.

# Lesson 3 Component System Operations

## Lesson 3: Component System Operations

- Architecture
- Summary of key automation technologies and features
- Relationships example: Starting an application
- Grouping concepts
- Automation in action

© Copyright IBM Corporation 2019

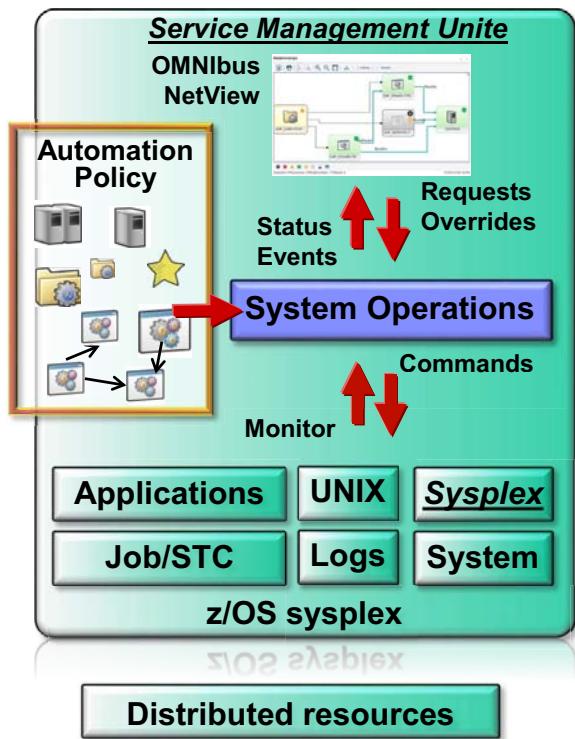
1-15

### *Component System Operations*

This lesson covers an overview of component **system operations**, its architecture, key automation technologies and features, and its grouping concepts.

Examples explain relationships and starting an application and automation in action.

# Component System Operations overview



## Policy-based and goal driven automation

- Start, recover and shutdown z/OS and distributed applications and systems using:
    - Timers, events, triggers, service periods
    - Dependencies and groups
    - Thresholds, active and external monitors
  - JES, CICS, IMS, DB2, WebSphere, OMEGAMON, UNIX, SAP, GDPS...
  - **Message monitoring and response**
    - WTO, WTOR, joblog, NetView, hardware
  - Escalation using OMNIBus, email, SMS...
  - **Prevent outages of critical resources:**
    - WTO and AMRF buffers, spool, sysplex
    - SYSLOG, LOGREC, SMF, dump data sets
- Easier operations at the application level**
- 3270 or **Service Management Unit**
  - Single point of control, single system image

© Copyright IBM Corporation 2019

1-16

## Component System Operations overview

The system operations component, which exploits NetView, automates many system console operations and selected operator tasks such as startup, monitoring, recovery, and shutdown of z/OS subsystems, components, applications, UNIX (USS), and sysplex resources and entire systems and sysplexes and even distributed resources (SA z/OS only).

The system operations component provides comprehensive, Plug and Play automation for mission-critical software like JES, CICS, IMS, DB2, IBM Workload Scheduler, WebSphere, OMEGAMON, UNIX, SAP, and GDPS (SA z/OS only), and many more.

Applications can be started or stopped with **timers, events, triggers**, or **service periods**.

**Dependencies** can be automatically established, like starting TSO after VTAM.

Resources can be members of **groups**, which can greatly reduce the complexity of automation definition and operations.

**Server groups** control which and how many group members are started. Group members can represent, for instance, application servers.

**Thresholds** can be defined to alert operators or stop recovery.

Various **active monitors** including a fast control block scanner and a monitor for UNIX resources are available.

Clients can specify their own monitor and interval. Monitor resources can determine the **health status** using external monitors or OMEGAMON.

**Messages** like write-to-operator (WTO) or WTOR, or even messages from job logs or internal CICS and IMS messages can be automated without writing scripts. Automation can be different depending on codes in the message or depending on how often the message appeared.

**Escalation** to various notification targets can be set up easily using pre-defined or user-defined alert points on a resource level to complete the following tasks:

- Create incidents in IBM Tivoli Service Request Manager
- Create events in IBM Tivoli Netcool OMNIbus
- Alarm staff using email, pager, or SMS using IBM System Automation for Integrated Operations Management (SA IOM)

The system operations component provides automation procedures that enable recovery of the following **z/OS components** and data sets:

- SYSLOG data sets
- LOGREC data sets
- System Management Facility (SMF) data sets
- Write-to-operator (WTO) buffers
- JES spool
- z/OS dump data sets
- Action Message Retention Facility (AMRF) buffers
- Sysplex resources, for example, coupling facility (SA z/OS only)

All that you have to do is specify to system operations in an ISPF dialog what resources you want to automate and monitor, and what your policies are for automation and monitoring. The automation engine, which is identical on all systems, starts, monitors, recovers, and shuts down resources and complex applications according to your configuration and goals.

At any time, the operator can monitor and control from an enterprise-wide single point of control at the application level. You have the choice of using a 3270-based NetView console or the graphical user interfaces Service Management Unite (part of SA z/OS) or the Tivoli Enterprise Portal, which is monitoring only. When exceptions occur or goals must be changed, an operator just issues requests or overrides. SA z/OS offers a sysplex single system image, system boundaries are removed.

**Goal driven automation** greatly simplifies operations. Operators just request what they want, and automation takes care of any dependencies and resolution of affected or even conflicting goals.

## What can you automate?

- Automate messages from z/OS, applications, logs, hardware
- Prevent outages of critical resources
- Start, recover, and stop z/OS and cross platform resources
  - Started tasks (STCs) and jobs independent of scheduler
  - UNIX System Services (USS) resources
  - Cross sysplex and cross platform
- IPL, startup, and shutdown pacing
- Change system configuration like day or night shift
- Automatic or manual application move or switch in sysplex
- DB2, CICS, and IMS automation
- Escalation of problems
- Pro-active automation using integration with monitors and OMEGAMON
- Integration with scheduling
- Problem determination

© Copyright IBM Corporation 2019

1-17

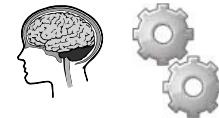
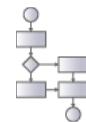
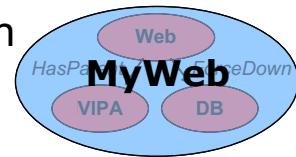
### What can you automate?

- Basic automation scenarios like automating messages from many sources, starting, recovering, and stopping of z/OS, UNIX System Services (USS) resources and even any resources across sysplexes and platforms
- Prevent outages of critical resources like MVS consoles, JES spool, logrec, SMF, page or dump data sets, couple data sets, or syslog
- Advanced automation scenarios like:
  - IPL, startup, and shutdown pacing
  - Different system configurations
  - Pro-active automation using integration with monitors
  - Automatic or manual application move or switch inside SAplex
- Escalation of problems to a status console or to an alert console
- DB2 automation
- CICS and IMS automation, even console access and internal messages and transactions
- Integration with scheduling: IBM Workload Scheduler has an automation workstation that allows

to send any command to automation, synchronously and asynchronously

## Summary of key automation technologies

- **Policy-based automation** can replace scripts
- Plug and play automation using best practices
- Powerful sysplex-wide automation and operation
- **Goal driven automation** to keep applications in line with business goals
- **Grouping** of resources for reduced complexity and management at business application level
- **Relationships** between resources for accelerated startup and shutdown, and correct recovery
- Manage by resource state, not by message
  - Status change triggers automation
- Monitor resources with health status and actions



Animated

© Copyright IBM Corporation 2019

1-18

### Summary of key automation technologies

**Policy-based automation** is a sophisticated methodology that allows to easily incorporate business goals into an automation framework. As the term implies, policy-based automation uses simple policy definitions specified in fill in the blanks ISPF-based panels, with no program scripting or special education required.

Policy-based automation includes resource information, groups of resources, and relationships in the decision-making process before taking action. Resource information defines resource class and name; how to start, stop, and monitor the resource; and what the preferred systems are (SA z/OS only). Resources can be members of cluster-wide (SA z/OS only) groups and can have relationships.

IBM ships **Plug and Play automation** that includes best practices automation for many z/OS applications. This can help to complete the following tasks:

- Reduce time and effort in creating a policy or updating one
- Improve automation quality

**Goal driven automation** greatly simplifies operations. Goal driven automation tries to keep applications in line with business goals specified in the policy or by the operator. Automation can

take care of any dependencies by issuing start or stop commands to put other resources into the required status. It can resolve conflicting goals and even can remember goals.

**Grouping of resources** and definition of aggregate or business applications can greatly reduce the complexity of automation definition and operations.

Exploiting application groups is beneficial in many ways:

- Automation definition and operations can be greatly simplified through the grouping of resources and even business-application definitions.
- Application groups let you monitor important business applications and help verify that everything they require is available.
- Groups can make operations easier by showing the aggregated status of resources and by group actions such as startup or shutdown.
- Through groups, operators are freed from knowing the various pieces that make up an application, their dependencies, how to start or stop them, and so on.

**Relationships** between resources to achieve accelerated startup and shutdown, and correct recovery. Resources can have complex dependencies of different kinds to other resources inside and outside of the application (group) they belong to. This gives you the power to define these dependencies, so that resources get what they need, are started in the right order as quickly as possible and are shut down fast without interference.

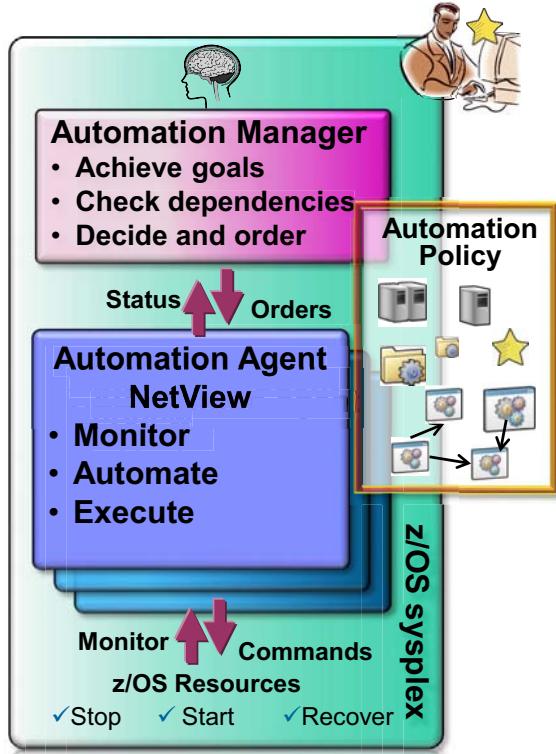
### Manage by state, not by message

Automation uses monitoring to keep track of the resource status. Although automation can use messages to update the status and also can automate messages, it is the status change that triggers automation. For example, child resources are started after the parent is up or a dependent resource is forced down when the resource it depends on terminates.

Also, messages are forgotten, but the status is preserved even across restarts and IPLs.

**Monitor resources** enable integration with any monitor and include easy to exploit OMEGAMON access to update the health status of monitor resources, which can trigger **pro-active automation**. The **health status** of monitor resources is propagated along monitor relationships to applications and is aggregated into the overall resource status.

# Architecture



- The automation policy defines the resource automation and goals ★
- Automation manager (AM)
  - Keep resource status in line with goals and dependencies by sending start and stop orders to agent
- NetView-based automation agent role:
  - Monitor and automate resources
  - Inform AM and execute AM orders
  - Operator console
- Operators
  - Control automation
  - Detect and manage problem statuses
  - Override goals

© Copyright IBM Corporation 2019

1-19

## Architecture

Elements of the automation engine are separated. Those that observe, react, and take action remain within the IBM Tivoli NetView address space. Known as the **automation agent**, this portion must be present on every system to be automated.

The coordinating, decision-making, and controlling elements are grouped into a single address space known as the **automation manager**. The automation manager tries to keep resource status in line with goals and dependencies by sending start and stop orders to agent

The automation agent runs in the NetView address space which provides the operator console. The automation agent monitors and automates resources. Status changes are sent to the automation manager. The automation agent receives orders from the automation manager and issues commands that are based on defined automation policy.

The operator role is to control automation, to detect and react on problem statuses, messages, and alerts as well as problem analysis, execution of automation change requests by managing automation, policy, and resources.

# Key automation features

- Comprehensive automation for z/OS and UNIX applications, for example:
  - User definable **start types** and three **stop types**
  - Desired available status, startup and restart options
  - Warning and alert **thresholds** to stop recovery
  - Exception and **captured messages**
  - Built-in ASCB scan monitor reduces CPU overhead for monitoring
  - Monitoring of **UNIX** processes, files, and ports
- **Server group** with *availability* and *satisfactory targets* and member *preferences*
- Operator notification and **alerting** to SDF, SA IOM, OMNIbus, problem management, and so on
- **Runmodes** for selective startup like base, online, databases, and so on
- **Pacing gates** can prevent startup or stop of too many applications
- Non-disruptive cluster-wide policy activation
  - Synchronized with NetView automation table load
- **Automation reports** and availability and recovery time reporting
- Comprehensive **automation infrastructure** makes extension easy

## Key automation features

On this slide are the key automation features.

At the core is comprehensive automation for z/OS and UNIX applications.

Highlights include:

- User definable **start types** with multiple passes and stages that also affect message automation as well as prestart and poststart policy
- Three **stop types** with multiple passes and an INIT and FINAL phase
- **Desired available status** to initially start or stop an application or to accept its current observed status
- **Startup and restart options** like whether the application is started after any failure or only for specific ABEND codes.
- **Thresholds** can be set that let the operator know if certain errors are occurring infrequently, frequently, or have reached a critical level where the recovery process must end to avoid endless loops. This is done by specifying how many times an error must happen in a certain time period for each error situation.
- **Exception and captured messages**

Special support exists for **UNIX resources** running in z/OS UNIX System Services: Monitoring of UNIX processes including UNIX user ID, command path, and filter as well as files, and ports.

Automatic or manual move of an entire application inside a system or Parallel Sysplex (SA z/OS only) is provided by **move groups** that have member preference values.

**Server groups** can represent, for instance, application servers, and control which and how many group members are started through preferences and availability and satisfactory targets. System Automation supports sysplex-wide group members.

Move groups and server groups allow to define preferred locations of a resource and what should happen when a system is unavailable or available again.

Various active monitors including a fast control block scanner and a monitor for UNIX resources are available.

**Escalation** to various notification targets can be set up easily using pre-defined or user-defined alert points on a resource level to:

- Create incidents in IBM Tivoli Service Request Manager
- Create events in IBM Tivoli Netcool OMNIbus
- Alarm staff using email, pager, or SMS using IBM System Automation for Integrated Operations Management (SA IOM)

**Runmodes** can be defined to partially start the system or to start backup applications that are normally started on another system.

The policy management can be incremental and is non-disruptive during policy activation. Policy activation is synchronized with the NetView automation table and message revision table load.

As it was covered previously, it is not mentioned here is that the system operations component provides automation procedures that enable recovery of many z/OS components and data sets.

If the resource consumption of too many applications during the start and stop phases becomes a problem, **pacing gates** can be defined to limit the number of applications of a kind that can be started or stopped at the same time.

**Automation reports** display statistical information about the automation agent and some basic information about the automation manager.

For availability and recovery time reporting automation collects and records job-related information, and writes **System Management Facility** (SMF) records at specific events in the lifetime of a resource.

Comprehensive **automation infrastructure** makes extension easy. Automation supplies commands that provide your automation procedures with a simple, standard way of interfacing with the automation control file, the automation status file, and the NetView log file.

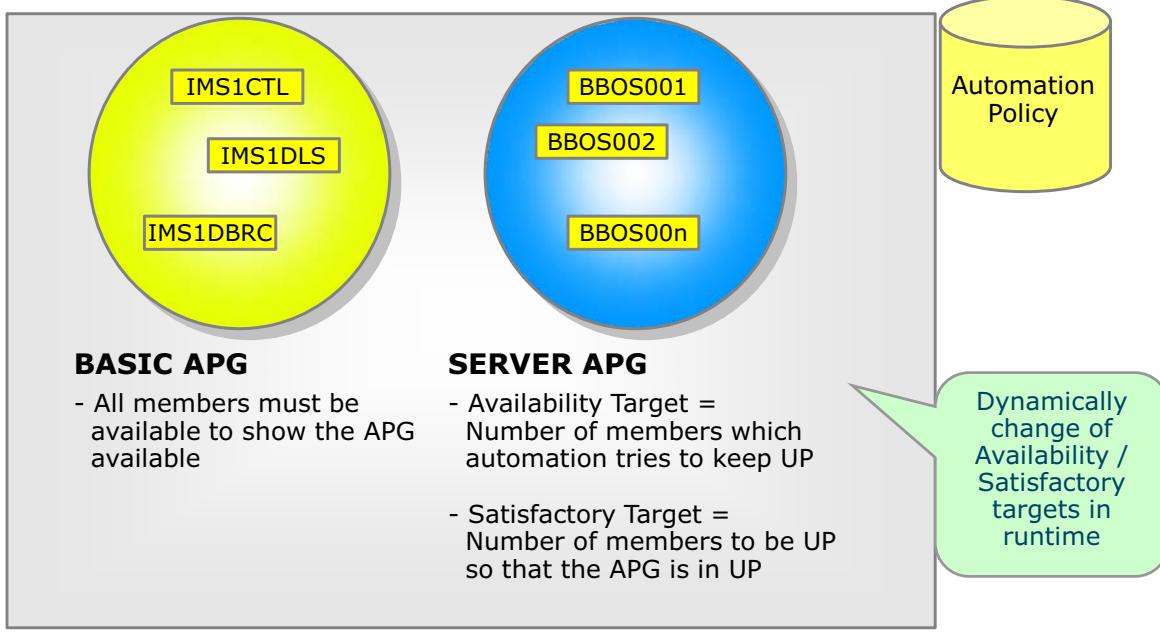
Using these commands in automation procedures provides you with the following advantages:

- Reduced development time: Less code must be written.
- Portable code: Automation policy information that is unique to an enterprise can be kept in the automation control file rather than distributed among many automation procedures. The automation procedures implement a number of different rules for handling a situation and the automation control file is used to select which rules are applicable to the current situation.
- A consistent, documented interface. Messages like Write-to-operator (WTO) or WTOR or even from job logs or UNIX can be automated without writing scripts. Automation can be different depending on codes in the message or depending on how often the message appeared.

## Grouping concepts

An Application Group (APG) is a manageable resource. Requests are propagated to its members.

The APG shows an aggregated state derived from the statuses of its members.



© Copyright IBM Corporation 2019

1-21

### Grouping concepts

In general, the application groups are manageable resources as any other resource like TSO or RRS.

This means you can easily start or stop application groups and these start-stop requests are propagated to the group members. A **basic application group** is available when all of the members are available.

In the Plug and Play automation policy, all the base z/OS components together with all network components are contained in a basic application group.

A **server application group** is available when some of its members are available.

This number is specified with the *availability target* value in the policy.

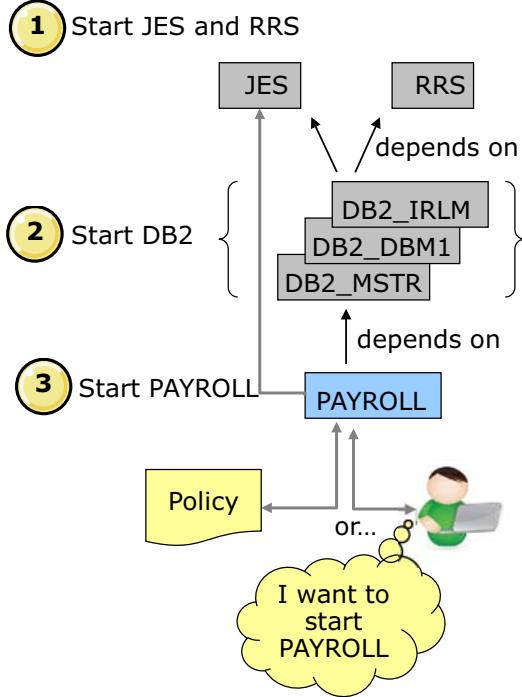
You can change this value dynamically in the runtime environment.

With the *satisfactory target*, you define the number of members to be available so that the application group is available. The satisfactory target number can be equal or less to the availability target number.

In addition, there are **move groups**, which are basically a server group with an availability target of one.

## Example: Relationships: Starting an application

Automation will:



- Application PAYROLL uses a DB2 database to access employee data
- DB2 itself consists of multiple address spaces. Required are:
  - MSTR, master address space
  - DBM1, services
  - IRLM, lock manager
- For transaction processing, the Resource Recovery Services (RRS) address space is needed
- Most address spaces depend on the JES subsystem
- To ensure proper function of PAYROLL, these dependencies must be considered when PAYROLL is started

© Copyright IBM Corporation 2019

1-22

### Example: Relationships: Starting an application

To get another view on the relationships, let's assume you have an application that is called PAYROLL. This application uses DB2 to access employee data. Now the scenario is either that of a system IPL where the application should be brought up by automation automatically, or where the operator issues a request to start this application.

So payroll depends on DB2. DB2 itself consists of a set of address spaces. The MSTR address space, the DBM1 address space with the database services and the IRLM, the lock manager that is necessary to serialize access to the database.

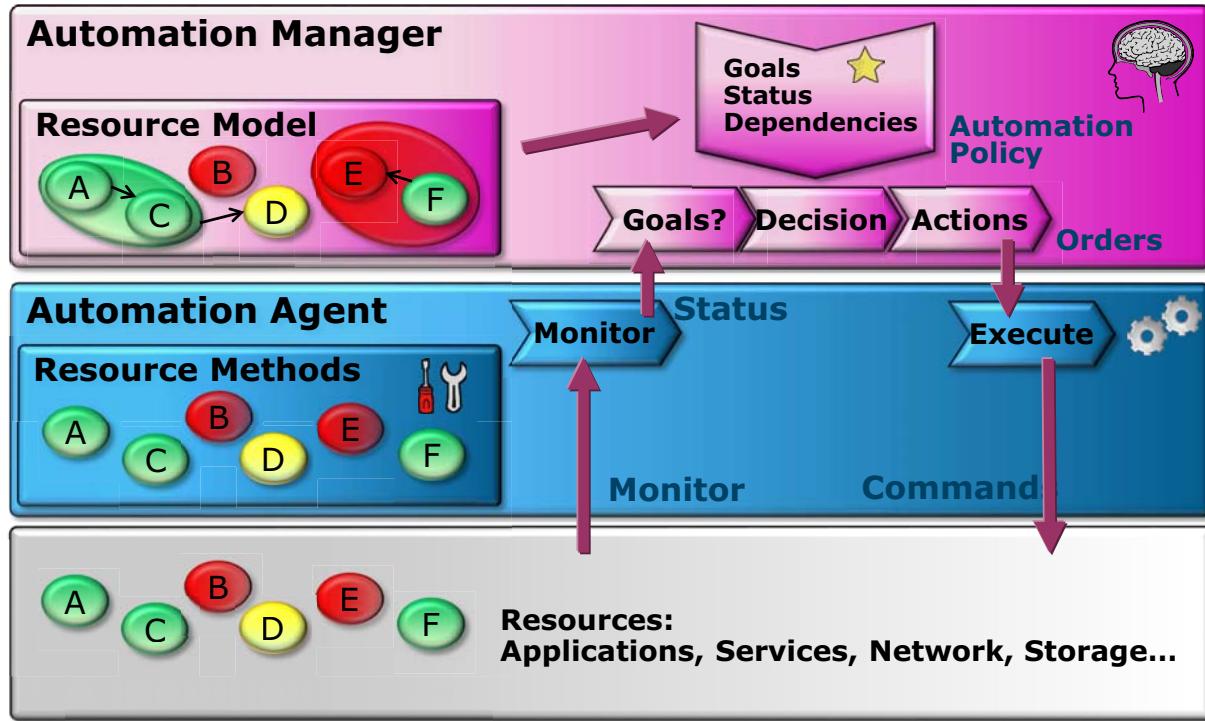
Looking more closely at DB2 you find that DB2 depends on the Resource Recovery Services for two-phase commit transaction processing and, like most other address spaces, it also depends on JES.

So what is the task of automation in this case?

Well, automation ensures that when the desired status of PAYROLL is AVAILABLE, that it follows the dependency graph to ensure that all supporting resources are started and up and running before PAYROLL is made available.

Here, it therefore must ensure that JES and RRS are started, then the DB2 address spaces must be started and finally, if all the prerequisites are fulfilled, PAYROLL can be started.

# Automation flow



© Copyright IBM Corporation 2019

1-23

## Automation flow

Applications, Services, Network, Storage, and other resources are modeled in automation policy.

The automation policy specifies resource attributes, groups of resources, and relationships. Resource attributes include resource category and name, resource methods to start, stop, and monitor the resource, goals, and a lot more. Resources can be members of groups and can have relationships.

The automation manager uses the automation policy to create a resource model that defines resources, groups, and their relationships as well as goals.

The automation manager maintains resource and system status and any changes to goals.

The mission of the automation manager is to keep the system in line with the goals.

1. The automation agent notifies the automation manager about a status change.
2. This status change can affect other resources, for example when dependent resources need to be stopped or were waiting for another resource's status change.
3. Then, the automation manager uses the goals, resource model, relationships, location, and status to decide the recovery actions. If the goal can only be achieved when the dependencies are fulfilled, the goal might be propagated along the dependencies and might cause more actions.

4. The automation manager delegates the actual executions that resolve the situation to the automation agent(s).
5. The automation agent monitors, starts, and stops a resource. It executes the orders by issuing the commands that are defined in the policy.

# Lesson 4 Goal driven automation

## Lesson 4: What is goal driven automation

- Availability goal
  - Suspended goal
  - Goals come from a **source**
  - Operator can create (request) or override goals
  - Goals have a **priority**
  - Goals are **persistent**, even when they cannot be achieved
    - Unlike commands that are entered and forgotten
    - Cancel requests when you no longer need them
  - Automation manager tries to achieve goals while taking care of dependencies
  - Easier and safer operations with a single action at application group level
    - **Goals can be propagated** to group members and along active dependencies putting all required resources in the desired status
    - **Conflicting goals are resolved** by priorities
  - Resource status adjusted to goals, dependencies, configuration, and status
- 
- Priority ↑
- |           |             |
|-----------|-------------|
| available | unavailable |
| resumed   | suspended   |
- |            |
|------------|
| Source     |
| Operator   |
| Automation |
| ExtSched   |
- |        |
|--------|
| Force  |
| High   |
| Low    |
| Policy |

© Copyright IBM Corporation 2019

1-24

### Goal driven automation

Goal driven automation is very different from the command driven automation of other products:

- Goal driven automation can keep applications in line with business goals, dependencies, configuration, and status
- Goals are defined in the policy, can be overridden or created by the operator
- Goals are persistent
- Goals can be propagated

The administrator defines the goals for the application according to business requirements in the policy. Goals are either available (up) or unavailable (down). Application group goals control, which and how many members are started.

The operator can change or override goals and create start or stop goals by entering requests. Operator requests can have a higher priority than the Desired Available state from the policy.

Goals are persistent. They are not like commands that are run and forgotten. You must cancel requests when you no longer need them.

Operations at the application level means starting an application having several components and dependencies with one request.

- The request is propagated to group members and along the dependency tree
- When multiple dependencies exist, conflicting goals can be the result. Conflicting goals are resolved. Requests have a **priority** and have a **source**. Automation uses priorities and sources when determining which request to accept. Sources might be, for example, automation scripts (**AUTOOPS**), human operators (**OPERATOR**), or **EXTERNAL**, like IBM Workload Scheduler

This process can prevent operations errors, like shutting down a resource that is needed by another application. Operators can use goal driven automation to simplify operations by entering or removing a request at the application level. They can use the automation product to manage dependencies and goals that are affected and to resolve conflicting goals.

The automation product automatically adjusts resource status to business goals, dependencies, configuration, and status.

# Lesson 5 Policy-based automation

## Lesson 5: Policy-based automation

- The power of an automation policy
- Policy available out of the box
- Automation policy features
- Automation policy for applications
- Policy reports



*Policy-based automation*

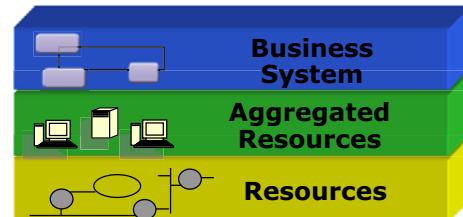
Lesson 5 is about policy-based automation.

It explains:

- The power of an automation policy
- Policy available out of the box
- Automation policy features
- Automation policy for applications
- Policy reports

# What is the power of an automation policy?

- Faster time-to-value through:
  - Plug-and-play automation using best practices for many applications
  - Easier definition through ‘fill in the blanks’ panels
- Less programming required – less coding errors
- Consistent and reliable automation actions
- Management of entire business applications, rather than individual resources
- Enterprise-wide shareable and clone-able automation policy
- Documentation in reports
- Activity log for policy changes



© Copyright IBM Corporation 2019

1-26

*What is the power of an automation policy?*

Policy-based automation focuses on separating your business and operational policies from the mechanics of performing the automation according to the policies. Policy-based automation focuses on setting your policies and automation deals with implementing them.

Policy-based automation includes resource information, groups of resources, and relationships into the decision-making process before acting.

Resource information defines resource class and name, as well as how to start, stop, and monitor the resource. Resources can be members of system-wide groups and can have relationships.

**What is the power of a policy?**

- Easier definition of automation requirements. In a policy, you can define which resources belong together and should be managed as one business entity. For example, a DB2 system consists of many resources. With the automation product, you can group and aggregate resources to more meaningful manageable entities, for example, My-HumanResource-Application. You can monitor on this level, issue commands on this level, and manage at the business level rather than at the single IT resource level.
- In policies, you can specify how resources are dependent and related on each other. For example, which of the other resources must be available before a certain resource can be

started. In another example, my\_database must be up and running before my\_application is started.

- Policy definitions can be reused, copied, and cloned for similar applications elsewhere in the enterprise.
- Because the underlying technology is responsible for the detailed actions, these actions are performed in a consistent and reliable manner. With traditional programming solutions, the testing of abnormal conditions is difficult and prone to be incomplete. The action of automation under these abnormal conditions is, however, critical to the entire automation solution.

Less programming and related special education is required. With programmed scripts, high effort is required to take care of every resource status, relationship, and cluster configuration. You can never be sure whether you missed or did not test something.

## > 800 predefined, selectable messages

Component	Description	Message prefixes
ZFS	Unix File System for zSeries	(IOE)
APA	IBM Application Performance Analyzer	(CAZ)
APPC	Advanced Program to Program Communication	(ASB ATB)
BMC	BMC multiple Software Products	(BMC BBM)
CA	CA multiple Software Products	(CAS ESF PXM)
CICS	CICS transaction server	(CTG DFH BBM)
DB2	DB2 database server for z/OS	(DSN)
DFHSM	Hierarchical Storage Manager	(ARC)
DFRMM	Removable Media Manager	(EDG)
GDPS	GDPS and z/OS Hyperswap Services	(GEO IOS)
GMFHS	NetView Graphical Monitor Facility Host Subsystem	(DUI)
InfoSphere	DB2 and IMS Queue Replication	(ASN CAC)
IMS	IMS database and supporting framework	(DFS CQS CSL DSNM)
IWS	IBM Workload Scheduler	(EQQ)
JES2	Job Entry Subsystem 2	(HASP)
JES3	Job Entry Subsystem 3	(IAT)
Lifeline	Multi-Site Workload Lifeline	(AQS)
LDAP	Lightweight Directory Access Protocol	(GLD)
MIM	CA MultiSystem Integrity Manager	(MIM)
MQ	IBM MQ - Message Oriented Middleware	(MQ)
NFS	z/OS Network File System	
OMEGAMON	z/OS Omegamon Perform. Monitors (CN)	
OMVS/USS	z/OS Unix System Services (Open MVS)	
OSA/SF	zSeries Open Systems Adapter Support	
QFETCH	Unicenter CA-QuickFetch	
RACF	Resource Access Control Facility	
RMF	Resource Measurement Facility	
RODM	NetView Resource Object Data Manager	
RRS	Resource Recovery Services	
SDSF	System Display and Search Facility	
SLS	SUN Microsystem Tape System Host Sc	
TBSM	Tivoli Business Service Manager	
TPX	CA Terminal Productivity Executive (VTAM appl)	(TPX)
TSM	Tivoli Storage Manager	(ANR)
VPS	LRS VTAM Printer Support	(VPS)
VTAM	z/OS Communication Server	(IST)
WebServer	z/OS HTTP Server	(IMW)
WebSphere	z/OS WebSphere components	(BBO)

- Selectable components
- Serviced
- Service updates report
- Changeable
- All messages policy panel:

Make a selection ( S ) or use shortcuts ( WHU AC AS AO MS MD )		
Cmd	Message ID	Description
	IEF378I	CATALOG disposition error
	IEF402I	Job failed
	IEF403I	Job is starting
	IEF404I	Job has terminated
	IEF450I	Job abended
	IEF451I	Job cancelled by cond code
	IEF452I	JCL error
	IEF453I	JCL error
	IEF743I	Job abend
	IFB040I	LOGREC recovery
	IFB060E	LOGREC recovery

© Copyright IBM Corporation 2019

1-27

>800 predefined, selectable messages

More than 800 messages are predefined. These messages are either part of a selectable component as the left screen shot shows, or cover special areas as well as basic system messages, for example IEF\* messages, which do not belong to an application.

**Each component can be deselected** per system to reduce the number of messages in the NetView automation table.

### How to apply service updates

Whenever maintenance updates that affect the generation of NetView automation tables are available, the AT/MRT APAR Apply Options panel is displayed.

It allows to complete the steps provided here:

- Apply the changes.
- Create a detailed service updates report about the differences.
- Skip the changes.

The service updates report provides the following details:

- New messages together with the generated NetView automation table entry.
- Deleted messages together with the generated NetView automation table entry.
- Messages that have a modified NetView automation table entry. It includes the old and new generated NetView automation table entry.

Furthermore, for all messages the user-defined NetView automation table entry is shown, if it exists in the policy database.

## **All messages policy panel**

Messages policy is available in the messages entry type option 21. It shows a list of all messages defined in the policy database together with the messages from MVC entry '+SA\_PREDEFINED\_MSGS'.

This list offers alternative access to the messages definitions with the advantage that it is not necessary to know the entry defined for the message.

This policy is the main location for all AT, MRT, and MPF related changes.

# Best practices policy for 270 applications and application classes out of the box

- ✓ Minimal effort, high quality
- ✓ Messages for the following components are provided and serviced

• z/OS ++	• NetView +	• OMVS	• MQ +
• DLF, LLA, VLF	• Automation Manager +	• TCP/IP	• Db2 ++
• JES2 & JES3	• SysOps ++	• DVIPA	• IRLM
• VTAM	• ProcOps	• OMPROUTE	• IMS +++
• TSO	• E2E	• CRON	• IMS FDR
• RACF	• GDPS +	• HTTP Server	• CICS +++
• HSM	• Lifeline	• RESOLVER	• CICSplex SM
• RMM	• OMEGAMON +++	• inetd	• SAP +++
• RRS	• IWS ++	• Portmapper	• BMC products +
• SDSF	• TSM	• LDAP	• CA products +
• FFST	• TBSM +	• NFS	• SUN Tape System
• RMF +		• ZFS	• WebSphere +
• APPC & ASCH		• FTP	• CIM
• ...		• SSH	• Infosphere
		• Syslogd	• z/OS Connect
		• TN3270 +	
		• OSA	

More messages and components are added permanently  
(Development / APARs)

## ++ Size of policy

Animated

© Copyright IBM Corporation 2019

1-28

Best practices policy for 270 applications and application classes out of the box

+ means multiple resources and components.

More messages and components are added permanently by development and APARs.

The complete best practices policy covers:

1. 270 applications and application classes
2. 127 Application groups
3. 28 Monitor resources
4. More than 800 messages
5. 19 systems including z/OS, GDPS, Linux, Coupling facility, z/VM, z/VSE, zAware
6. 7 sysplexes, clusters, and system groups
7. 5 Processors
8. Automation for MVS Components

The policy is based on best practices and includes:

- Applications and application classes with:
  - Startup procedures
  - Shutdown procedures
  - Relationships
  - Status messages
  - Automated messages as well as captured messages
  - Special policy like JES2 Drain or USS Control specifications
- Application groups
  - Basic for linking and complex applications
  - Move and server groups for failover and proactive automation
- Monitor resources to determine health status like for CICS, IMS; JES...
- MVS Components automation and defaults
- Defaults for applications, systems, sysplexes
- Automation Infrastructure like Automation Operators
- Overall Policy Database documentation

# Policy reports provide automation documentation

## System Automation for z/OS Policy database report

**User Id:** INGC102  
**PolicyDB Name:** IBM\_ALL  
**PolicyDB data set:** 'SAZOS.PDB.IBMA  
**Creation Date:** Monday, 4 Mar 2018  
**Creation Time:** 15:19:47

Specifications for DFHSI1580D		
Ps>Select	Ct	Reply Text
<a href="#">DFHSM0102</a>		GO
<a href="#">DFHTS1310</a>		CRITICAL message captured
<a href="#">DFHTS1311</a>		CRITICAL message captured
<a href="#">DFHZC3482</a>		CRITICAL message captured
<a href="#">DFHZC5966</a>		CRITICAL message captured

### Selected Entry Type(s)

[ENT: Enterprise, AT, MRT and MPF Specifications](#)  
[GRP: Groups](#)  
[SBG: SubGroups](#)  
[SYS: Systems](#)  
[APG: ApplicationGroups](#)  
[API: Applications](#)  
[EVT: Events](#)  
[SVP: Service Periods](#)  
[TRG: Triggers](#)  
[PRO: Processors](#)  
[MTR: Monitor Resources](#)  
[ENS: zEnterprise Ensembles](#)  
[TMR: Timers](#)  
[TPA: Tape Attendance](#)  
[MVC: MVS Components](#)  
[PAC: Pacing Gates](#)  
[MDF: MVSCOMP Defaults](#)  
[SDF: System Defaults](#)  
[ADF: Application Defaults](#)  
[AOP: Automation Operators](#)  
[NFY: Notify Operators](#)  
[NTW: Networks](#)  
[XDF: Sysplex Defaults](#)  
[RES: Resident CLISTS](#)  
[SCR: Status Displays](#)  
[UET: User E-T Pairs](#)  
[DMN: Remote Domains](#)  
[REF: Resource References](#)  
[OMN: OMNI System Details](#)

hyperlinked

### Startup procedures

Phase	Description		
STARTUP	Executed to initiate the startup		
Type	AutoFn/*	Command Text	
AUTO		MVS S &SUBS/OB,PARM='SI,START=AUTO&APPLPARMS'	
COLD		MVS S &SUBS/OB,PARM='SI,START=COLD&APPLPARMS'	
INITIAL		MVS S &SUBS/OB,PARM='SI,START=INITIAL&APPLPARMS'	
NORM		MVS S &SUBS/OB,PARM='SI,START=AUTO&APPLPARMS'	

### Threshold Specifications

Minor Resource	Critical Number	Critical Interval	Frequent Number	Frequent Interval	Infrequent Number
TRAN	2	00:05:00	4	01:00:00	5

### Linked Classes and Instances

Class / Instance	Type	Description
<a href="#">C_CICS_CMAS</a>	CLASS	Class for CICS CPSMCMAS
<a href="#">C_CICS_WUI</a>	CLASS	Class for CICS CPSMWUI
<a href="#">C_CICS_XACTIONSERVER</a>	CLASS	Class for CICS Transaction Server

### Example of Policy Database report

© Copyright IBM Corporation 2019

Animated 1-29

Policy reports provide automation documentation

The policy database stores a huge number of definitions that can be used also to provide automation documentation:

The Customization Dialog has implemented a function for generating policy database reports. Two report formats are available.

1. You can generate a flat file output that is written to a PDS data set
2. The second possibility is to generate an HTML format and write it to a file system directory to share it easily with operators.

This version of the report creates hyperlinks for all entry names and resource names that take you to the start of the data for the selected entry or resource. These links also work if the HTML report is split into several files.

# Lesson 6 Key operations features

## Lesson 6: Key operations features

- Key operations features
- User interfaces
  - 3270-based operator dialogs
  - Status Display Facility
  - **Service Management Unite**
  - Tivoli Enterprise Portal

### *Key operations features*

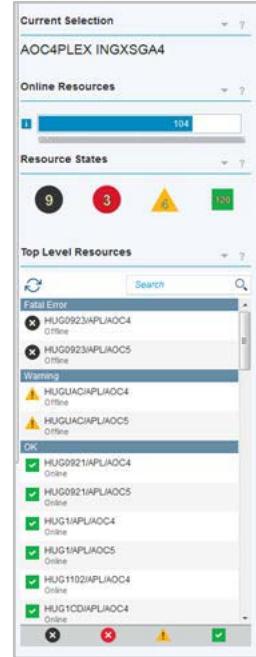
Lesson 6 introduces the key operations features and the user interfaces available in NetView with 3270-based operator dialogs and the dynamic, highly customizable Status Display Facility.

The following graphical user interfaces are available:

- **Service Management Unite (SMU)**: A new way to operate your mainframe.
- The Tivoli Enterprise Portal to monitor the status of automation on z/OS systems from a browser-based interface.

# Key operations features

- **Operations at the application level** lowers complexity
  - Automation takes care of dependencies and components
  - Status aggregation that includes health status
- **Goal driven automation** can reduce errors
  - Goals are not changed by IPL
- Replace consoles with a **single NetView console**
  - Dynamic **Status Display Facility** and powerful operator commands and dialogs
  - Event and status history and logs
  - **Single point of control**
  - Parallel sysplex-wide **single system image**
  - Convenient panel tailoring, sorting, and filtering capabilities
  - Automation flags to switch automation on or off
    - Scope can be system-wide down to single messages or transactions
- *Suspend and resume automation of applications*
- *System IPL complete notification*
- Web-based user interfaces
  - **Service Management Unite** (SA z/OS only)
  - Tivoli Enterprise Portal



© Copyright IBM Corporation 2019

1-31

## Key operations features

On this slide is a selection of key operations features in the automation product.

**Operations at the application level** means starting an application consisting of several components with one request.

**Goal driven automation** simplifies operations. System operators enter or remove a request at the application level. The automation product manages any dependencies and goals that are affected. It can resolve conflicting goals and handle goals even over restarts and IPLs.

## Replace your consoles with a single NetView console:

At any time, the operator can monitor and control from an enterprise-wide (SA z/OS only) single point of control at the application level. You have the choice of using a 3270-based NetView console or the graphical user interface of the Tivoli Enterprise Portal. Both are easy to use, powerful, and provide a common interface for all types of resources. When exceptions occur or goals must be changed, an operator just issues requests and overrides. SA z/OS offers a sysplex single system image, system boundaries are removed.

The single point of control provided by Automation Control for z/OS is limited to three systems in the sysplex.

The 3270-based NetView console includes the dynamic Status Display Facility (SDF) and powerful operator commands and dialogs as well as the netlog that shows all messages and WTOs that are sent to NetView.

Operators can tailor the layout of most display panels that are horizontally scrollable to pin or hide to determine order of columns and to sort by key in different directions and columns.

**Automation flags** allow to switch automation on or off, from a system-wide scope down to single messages or transactions.

**System IPL complete** notification.

**Suspend and resume resources** is a new and easy way to temporarily suspend automation for specific resources and their dependents without impacting the operations team by generating false alarms.

Operators can take resource in and out of automation easily.

# User interfaces

- **Service Management Unite**
- 3270 based operator dialogs
- Status display facility (SDF)
- Tivoli Enterprise Portal

The first screenshot shows the **MVSA - SYSTEM STATUS SUMMARY** interface. It has three main sections: Resources (with links to APPS, GROUPS, and MONITORS), Messages (with links to WTOR and MESSAGES), and Special Items (with a link to GATEWAY). Below these is a table for the INGKYST0 command, showing resource details like Type, System, Sus, Compound, Desired, and Observed status. The second screenshot is a Relationships diagram showing dependencies between various automation jobs (EAP\_3IAPO/TEST, EAP\_3CH/APL/TE, EAP\_3IAP/PUT, EAP\_3IAP/PUT/TEST, VTAM/APL/TESTM, TESTMVS) using HasParent and RunsOn relationships. The third screenshot is a Resource Topology window showing dependencies for UDOAPG\_S6/APG/KYC, including UDOAPG\_S5/APG/KEYC and UDOAPG\_S4/APG/KEYC.

## User interfaces

With system operations commands, you can control and maintain all of the resources sysplex-wide from a single point of control.

**IBM Service Management Unite** at the top is a new customizable dashboard user interface that can increase productivity of day to day operations even for expert users.

**SDF** to the left is the abbreviation of ***Status Display Facility***. It consists of a set of hierarchical panels in NetView which are delivered as samples. The system administrator can also define user-specific panel layouts. If the status of an object changes, it is dynamically reflected on the panels and is propagated up in the hierarchy.

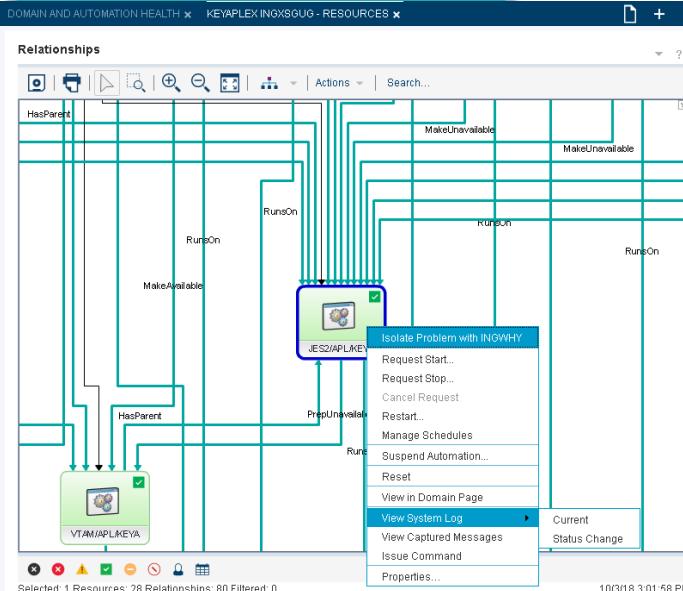
The ***INGLIST command*** at the bottom is available in NetView and provides details about all resources from the point of view of the automation manager. It displays information about a resource, such as statuses, flags, and schedules.

The ***Tivoli Enterprise Portal*** support allows you to monitor the status of automation on z/OS systems and z/OS sysplexes (SA z/OS only) using a TEP client. On the TEP workspaces, you get an overview on the resources with all their different states. The graphical summaries are combined with the detailed tabular views.

# Service Management Unite (SMU): User interface

**Modern Dashboard UI**

**Control business applications not components**



**Customizable Dashboards**

**Monitor and control your datacenters from smart devices**

- Topology
- Operations
  - Requests
  - Suspend and resume
  - Reset status
- Service periods
- Manage groups
- Netlog and syslog (CANZLOG)
- Command entry and command responses
- Captured messages
- Also used by OMEGAMON

1-33

## Service Management Unite: User interface

IBM Service Management Unite is a new customizable dashboard user interface that is available with IBM Service Management Suite for z/OS and IBM System Automation for z/OS 4.1. Service Management Unite provides system programmers, operators, and administrators with a transparent view of the system health status and allows for easy problem identification. When you are entitled for the Service Management Unite that comes with IBM Service Management Suite for z/OS, the dashboard enables operators to see both monitoring and automation exception events together so that they can identify critical problems. Operators can quickly and confidently analyze, isolate, and diagnose problems as all relevant data including important logs is provided in a single place. Service Management Unite also enables operators to interact directly with the system by issuing commands and viewing results without going to a different console.

Service Management Unite also provides access to automation functions to start, stop, or recycle business applications running on z/OS or Linux on z Systems and supported distributed platforms, even from mobile devices.

With Service Management Unite, you can perform automation activities in less time with fewer errors as compared to a conventional 3270 panel, by using a built-in scheduler. The dialog on the dashboard lets you easily start and stop resources at any defined time.  
Groups can be also easily managed.

Service Management Unite provides **widgets** and **dashboards**.

“Widgets” are Standard UI pieces which can be used to visualize the data provided by “data provider” of different products.

A compilation of widgets is called a “dashboard”. Products, like SA, provide standard dashboards, but a user is free to compile their own dashboard according to their needs. You can easily create dashboards that show widgets and data from different data providers or even from text files.

# Key capabilities

- Highly customizable
  - Create your own dashboards based on DASH capabilities
  - Customize pre-defined pages (for example, Configure displayed columns or data)
  - Role-based views
- Flexible views
  - Overviews about CICSplex, DB2, IMS, LPAR, MQ, and WebSphere Application Server
  - Mobile system status and health dashboard
  - Graphical view of System Automation resource relationships
  - System Automation Node/System centric view
  - Visualization of different resource types and state information
  - Expandable/collapsible tree views for resource hierarchies
- Integration
  - Problem isolation dashboards combining monitoring information and automation capabilities
  - Create linked views showing data from other IBM products
  - Create views integrating your own data
- Other capabilities
  - Enhanced search and filter capabilities
  - Allow user requests against multiple resources at a time
  - Server Group support
  - Allow priorities on start/stop requests

© Copyright IBM Corporation 2019

1-34

## Key capabilities

Service Management Unit is highly customizable. You can complete the following tasks:

- Create your own dashboards based on DASH capabilities
- Customize pre-defined pages (for example, Configure displayed columns/data)
- Use role-based views

Flexible views provide the following information:

- Overviews about CICSplex, DB2, IMS, LPAR, MQ, and WebSphere Application Server
- Mobile system status and health dashboard
- Graphical view of System Automation resource relationships
- System Automation system centric view
- Visualization of different resource types and state information
- Expandable/collapsible tree views for resource hierarchies

Integration capabilities:

- Problem isolation dashboards combining monitoring information and automation capabilities
- Create linked views showing data from other IBM products
- Create views integrating your own data

Other capabilities

- Enhanced search and filter capabilities
- Allow user requests against multiple resources at a time
- Server Group support
- Allow priorities on start/stop requests

# “Domain and Automation Health” dashboard

**Scenario:** Get an overview of connected automation domains and a summary of the domain and application status, with the possibility to drill-down into a displayed domain or application to get more information.

**Online Resources:** Number of top-level resources that are available

**Resource States:** How many top-level resources are in which Compound State?

**Top-Level Resources:**

- Shows all top-level resources of the selected domain sorted by Compound State
- Icon displays compound state
- Text below resource name shows Observed State

The dashboard interface includes a navigation bar with tabs like 'DOMAIN AND AUTOMATION HEALTH', 'Domains and Nodes', and 'Messages'. The main area shows a 'Topology View of Domains and Nodes' with nodes like KEYC, KEYB, KEYA, and cluster1. A callout for 'click to update' points to a node. Below this is a 'Messages' section listing recent events such as 'Domain cluster1 joined successfully' and 'Policy changed event has been received from domain cluster1'. To the right are three callout boxes explaining 'Online Resources', 'Resource States', and 'Top-Level Resources' with their respective definitions and icons.

### “Domain and Automation Health” dashboard

The Domain and Automation Health dashboard displays your domain and node topology. You can see statistics about the health of top-level resources of a selected domain or node. From here, you can also drill down to operational views of displayed domains, nodes, and resources. You can search for domains and follow up on your recent actions in the messages widget.

#### Domains and nodes widget:

Use this widget to get a high-level overview of the health status of your topology. You can see all your domains, nodes, their relationship towards each other, and a visual indicator for the health status of each domain and node in the topology. Expand to see their nodes

Domains and nodes have status indicators to show the domain health.

**TreeTable View of Resources:**

- Shows the full resource hierarchy of the current domain
- A predefined set of attributes are displayed in columns
- Displayed columns can be customized
- Expand the resource groups to display the nested resources

**Domain Info Bar:**

Shows the domain name, allows to switch the domain, shows communication status

**Relationships:**

Topology View of Relationships: Displays the relationship graph for the resource which is selected in the Resources widget

**Table of Requests:**

Shows all requests and votes for the currently selected resource

**Table of Nodes:**

Lists all Nodes belonging to the current domain

## Domain page

The Domain page shows one system, also known as domain. It includes the following widgets:

### Resources widget

The Resources widget displays the top-level resources of the selected domain. The resource table displays a predefined set of attributes for each resource. They include the state and request-related attributes that are most important from an automation perspective.

### Nodes widget:

The Nodes widget displays all known nodes of all known domains within System Automation Application Manager. The nodes are displayed in a table with columns. By clicking the column header, you can sort the column content. You can exclude and include nodes using this widget. To see which resources are hosted by a particular node, select the node. The associated resources are displayed in the Resources and Relationships widgets.

### Relationships widget:

You can select a resource in the Resources widget. The Relationships widget displays the relationships to all resources from and to the selected resource.

The Relationships widget offers filter possibilities for large relationship graphs. Select a state icon at the bottom of the widget to display only the desired resources.

**Requests widget:**

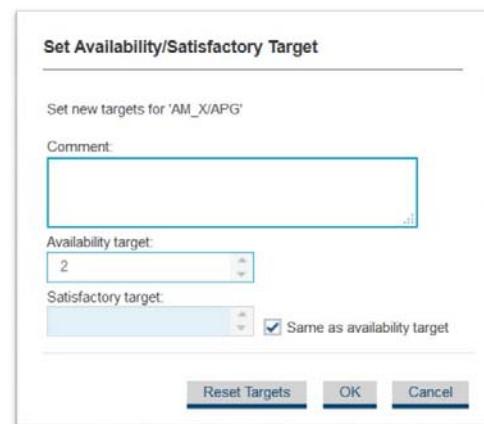
The Requests widget displays the list of requests that exist for the resource that is selected in the Resources widget. Use this list to understand how operator requests and their propagation determine the resulting desired state of the selected resource. From the menu of a selected request, you can review the complete set of properties including the comment. You can also cancel operator requests directly from the list.

# Service Management Unite V1.1.5 integrates with Zowe and more



- An extensible framework for connecting applications and tools to the mainframe
- Aims to make the mainframe an integrated and agile platform
- First open source project on z/OS <https://zowe.org>

- More:
- New Web Configuration Tool
  - Set targets for Server Groups:



© Copyright IBM Corporation 2019

1-37

Service Management Unite V1.1.5 integrates with Zowe and more

## What is Zowe?

Zowe is an open source project created to host technologies that benefit the Z platform from all members of the Z community (Integrated Software Vendors, System Integrators and z/OS consumers). Zowe, like Mac or Windows, comes with a set of APIs and OS capabilities that applications build on and also includes some applications out of the box.

Zowe offers modern interfaces to interact with z/OS and allows you to work with z/OS in a way that is similar to what you experience on cloud platforms today. You can use these interfaces as delivered or through plug-ins and extensions that are created by clients or third-party vendors.

Zowe consists of the following main components.

### Zowe App Framework

A web user interface (UI) that provides a virtual desktop containing a number of apps allowing access to z/OS function. Base Zowe includes apps for traditional access such as a 3270 terminal

and a VT Terminal, as well as an editor and explorers for working with JES, MVS Data Sets and Unix System Services.

## API Mediation Layer

Provides a gateway that acts as a reverse proxy for z/OS services, together with a catalog of REST APIs and a dynamic discovery capability. Base Zowe provides core services for working with MVS Data Sets, JES, as well as working with z/OSMF REST APIs. The API Mediation Layer also provides a framework for Single Sign On (SSO).

## Zowe CLI

Provides a command-line interface that lets you interact with the mainframe remotely and use common tools such as Integrated Development Environments (IDEs), shell commands, bash scripts, and build tools for mainframe development. It provides a set of utilities and services for application developers that want to become efficient in supporting and building z/OS applications quickly. The CLI provides a core set of commands for working with data sets, USS, JES, as well as issuing TSO and console commands.

# SMU Automation V1.1.5 (December 2018)

SMU Automation V1.1.5 is integrated with Zowe:

- A Zowe application plug-in is provided for SMU Automation to allow you to use SMU directly on Zowe Desktop and leverage free and commercial APIs in Zowe Application Framework.
- A new JES Explorer dashboard is provided to allow you to view job content and job output to isolate environmental issues. The JES Explorer dashboard can be started from SA APL resources.

See SMU exploitation and integration with Zowe for more information.

A web-based configuration tool is added as a modern alternative of the configuration dialogue cfgsmu. See [Use Web Config Tool] Configuring the SMU server and [Use Web Config Tool] Configuring the Universal Automation Adapter.

The installation is simplified and consolidated to provide minimal time-to-value:

- The tool to install SMU Automation is replaced by Installation Manager, which requires fewer user inputs and provides a consolidated installation experience. See Installing SMU Automation.
- The SMU Docker image and the eezdocker.sh script are enhanced to deploy and configure SMU in a Docker environment more easily. See Installing Service Management Unite with Docker/Installing and uninstalling SMU Automation with Docker.

Enhancements to SA Server Groups (with APAR OA54684 installed in System Automation for z/OS V4.1): You can now modify the satisfactory target and availability target of a server group in an easy-to-use dialogue from an SA dashboard.

# Service Management Unite (SMU) Automation V1.1.6 (March 2019)

## Enhancements to the Docker scripts:

The SMU Automation Docker image and the eezdocker.sh script are enhanced to deploy, configure, and upgrade SMU Automation in a Docker environment more easily. See [Installing Service Management Unite Automation with Docker](#).

## Enhancements to the Web Configuration Tool:

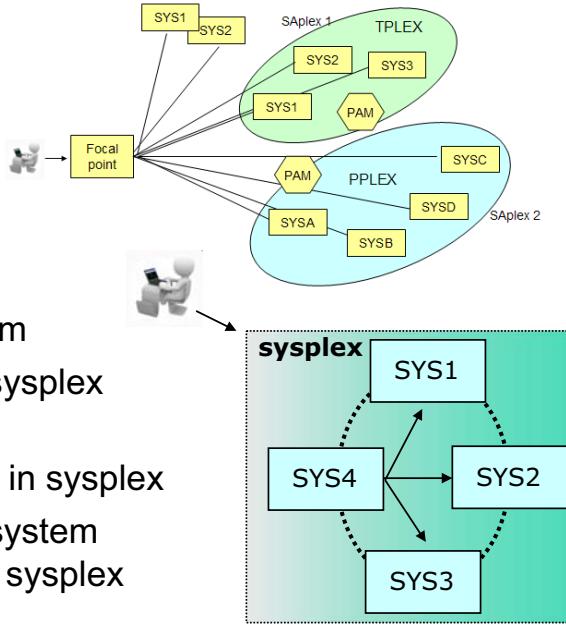
With the enhanced web configuration tool, you can configure properties to enable and establish connection with Zowe in the SMU dashboard.

## Enhancements to SMU exploitation and integration with Zowe:

- Zowe V1.0.1, V1.1.0, and V1.2.0 are supported.
- The SMU plug-in is packaged as a .tar file for easier decompression.
- In the JES Explorer dashboard, a dialogue is provided to assist you in easily accepting the certificate to avoid security issues when accessing Zowe micro-services.

# Single Point of Control and Single System Image

- Automation commands can be sent to any connected system in the sysplex or enterprise
  - Responses automatically routed back to user



## Single System Image

- Sysplex acts like a large single system
- Allows actions on subsystem within sysplex without specifying system name
- Host system is known on all systems in sysplex
- Selection panel displayed when subsystem runs on more than one system in the sysplex

© Copyright IBM Corporation 2019

1-38

### *Single Point of Control and Single System Image*

You can logon to any NetView and send automation commands to any connected system in the sysplex or enterprise by specifying the TARGET parameter. The responses are automatically routed back to the user. The INGLIST and DISPSTAT commands can even display resources from different systems in one panel.

**Single System Image** means that a sysplex acts like a large single system. It allows actions on subsystems within the sysplex without specifying the system name. Its host system is known on all systems in sysplex.

A selection panel is displayed when subsystem runs on more than one system in the sysplex.

# Operator interface: NetView 3270 dialogs

SA z/OS - Command Dialogs							Line 1 of 42
Domain Id . . . : AOFDA	----- INGLIST -----			Date . . . : 09/20/18			
Operator Id : INGC102	Sysplex = SYSPLEX1			Time . . . : 19:08:23			
A Update	B Start	C Stop	D INGRELS	E INGVOTE	F INGINFO	G Members	
H DISPTRG	I INGSCHED	J INGGROUP	K INGCICS	L INGIMS	M DISPMTR	P INGPAC	
R Resume	S Suspend	T INGTWS	U User	X INGWHY	/ scroll		
CMD Name	Type	System	Sus	Compound	Desired	Observed	Nature
AM_X	APG			SATISFACTORY	AVAILABLE	AVAILABLE	SERVER
AUTOMGR	APL	MVSA		PROBLEM	UNAVAILABLE	PROBLEM	
AUTOMGR2	APL	MVSA		SATISFACTORY	AVAILABLE	AVAILABLE	
AUTONETV	APL	MVSA		DEGRADED	AVAILABLE	AVAILABLE	
AUTOSSI	APL	MVSA		SATISFACTORY	AVAILABLE	AVAILABLE	
A01B	APL	MVSA		PROBLEM	AVAILABLE	HARDDOWN	
A02G	APL	MVSA		INHIBITED	AVAILABLE	SOFTDOWN	
A03B	APL	MVSA		SATISFACTORY	AVAILABLE	AVAILABLE	
A04G	APG	MVSA		SATISFACTORY	AVAILABLE	AVAILABLE	BASIC
A05B	APL	MVSA		SATISFACTORY	AVAILABLE	AVAILABLE	
A06B	APL	MVSA		SATISFACTORY	AVAILABLE	AVAILABLE	
B01B	APL	MVSA		INHIBITED	AVAILABLE	SOFTDOWN	
B02B	APG	MVSA		AWAITING	AVAILABLE	SOFTDOWN	BASIC
B03B	APL	MVSA		SATISFACTORY	UNAVAILABLE	SOFTDOWN	
C01B	APL	MVSA		SATISFACTORY	UNAVAILABLE	SOFTDOWN	
C02G	APL	MVSA		SATISFACTORY	UNAVAILABLE	SOFTDOWN	
C03G	APG	MVSA		SATISFACTORY	UNAVAILABLE	SOFTDOWN	BASIC
C04B	APG	MVSA		SATISFACTORY	UNAVAILABLE	SOFTDOWN	SERVER
C05B	APL	MVSA		SATISFACTORY	UNAVAILABLE	SOFTDOWN	
C06G	APL	MVSA		SATISFACTORY	UNAVAILABLE	SOFTDOWN	
DONT#TOUCH	APG	MVSA		DEGRADED	AVAILABLE	AVAILABLE	BASIC
D01B	APL	MVSA		SATISFACTORY	UNAVAILABLE	SOFTDOWN	
D02G	APL	MVSA		SATISFACTORY	UNAVAILABLE	SOFTDOWN	
D03B	MTR	MVSA		DEGRADED	AVAILABLE	AVAILABLE	
D04G	APL	MVSA		AWAITING	AVAILABLE	SOFTDOWN	
E00G	APG	MVSA		DEGRADED	AVAILABLE	DEGRADED	
E01G	APL	MVSA		SATISFACTORY	AVAILABLE	AVAILABLE	
E02G	APL	MVSA		PROBLEM	UNAVAILABLE	HARDDOWN	
E03G	APL	MVSA		SATISFACTORY	AVAILABLE	AVAILABLE	
E04G	APL	MVSA		SATISFACTORY	AVAILABLE	AVAILABLE	
E05G	APL	MVSA		SATISFACTORY	UNAVAILABLE	SOFTDOWN	

Command ==> ■

F1=Help F2=End F3=Return F4=DISPSTAT F5=Filters F6=Roll  
F8=Forward F9=Refresh F10=Previous F11=Next F12=Retrieve

© Copyright IBM Corporation 2019

1-39

## Operator interface: NetView dialogs

Different sets of commands are available from NetView host sessions:

- Automation manager commands
- System operations commands (automation agent commands)
- Processor operations commands

With system operations commands, you can control and maintain all of the resources sysplex-wide from a single point of control (*for more than 3 systems this is a SA z/OS only function*). They operate in two modes:

- Full-screen mode: If it is a command to display information and you enter the command name, a panel is displayed showing all available resources. On this panel, you can specify further actions, commands, or both for a special resource. If you enter a command to maintain a resource without further parameters, a full-screen panel prompts you for more information.
- Line mode: From a command line, you can enter the complete syntax of a command to receive either the desired output directly or to manipulate the resource you wanted in the way you wanted. The system operations commands with an OUTMODE parameter can be used in NetView pipes or REXX execs, however INGLIST should not be used as an API. Use INGDATA instead.

The INGLIST command provides details about all resources from the point of view of the automation manager. It displays information about a resource, such as statuses, flags, and schedules.

On the INGLIST panel, you can:

- See more information about the resources by scrolling horizontally using PF11. INGLIST can use large 3270 screen sizes.
- Use PF9 (Refresh) to obtain a new set of data for the displayed subsystems.
- Use PF5 to invoke a filter dialog.
- Launch other actions to maintain and control resources or groups.
- Use the CMD column on the left of the panel to issue various commands against any of the resources that are displayed. The available commands are indicated at the top of the INGLIST panel.
- Use PF4 to toggle to the DISPSTAT command dialog, which gives you the automation agent view of the resources.

Description of the fields:

- The Name field shows the name of the resource.
- The Type field shows the type of the resource.
- The System field shows where the resource is defined.
- The Sus field shows whether the resource is suspended or not. No automation is done for the resource if it is suspended.
- The Compound field shows the compound status of the resource. This is a summary of all statuses of the resource and provides a single value to check the status of a resource.
- The Desired field shows the status that the automation manager is trying to move the resource to. It can either be available or unavailable.
- The Observed field shows the automation manager observed status which indicates the current status of the resource, as reported by the automation agent.
- The Nature field applies to group resources only and defines the type of the group: BASIC, MOVE, or SERVER.

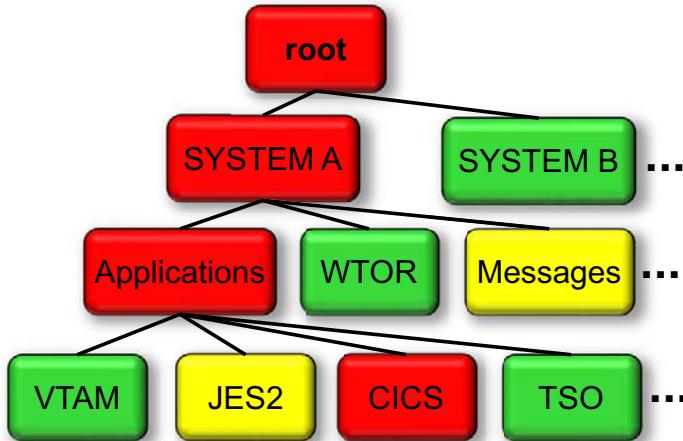
Using PF11, you can display more information like automation, startability, and health status, description, start and stop types, triggers, schedules, category, subtype, jobname, the current runmode, Inform, pacing gates, and more.

Use PF1 HELP to get a more detailed description.

The SORT, FIND, and RFIND subcommands are supported.

## Operator user interface: Status Display Facility

- Status Display Facility (SDF)
- Set of hierarchical panels with pre-defined or user-defined content
- Status changes are dynamically reflected on the panel and propagated up in the hierarchy
- SDF can be used to monitor status of
  - Application software
  - Outstanding operator replies (WTORs) or
  - User items
- Easy to setup
  - Support to facilitate customization and generation of panels



© Copyright IBM Corporation 2019

1-40

### Operator user interface: Status Display Facility

Next topic is a NetView based operator user interface of the automation product that is called SDF. SDF is the abbreviation of Status Display Facility. SDF displays the status of various resources of a z/OS system and of other z/OS systems that specify this system as an automation focal point.

With SDF you can monitor the status of applications, application groups, monitor resources, WTOR messages, captured messages, user items, hardware resources, like processors, and even IBM Workload Scheduler applications and tapes from a NetView console.

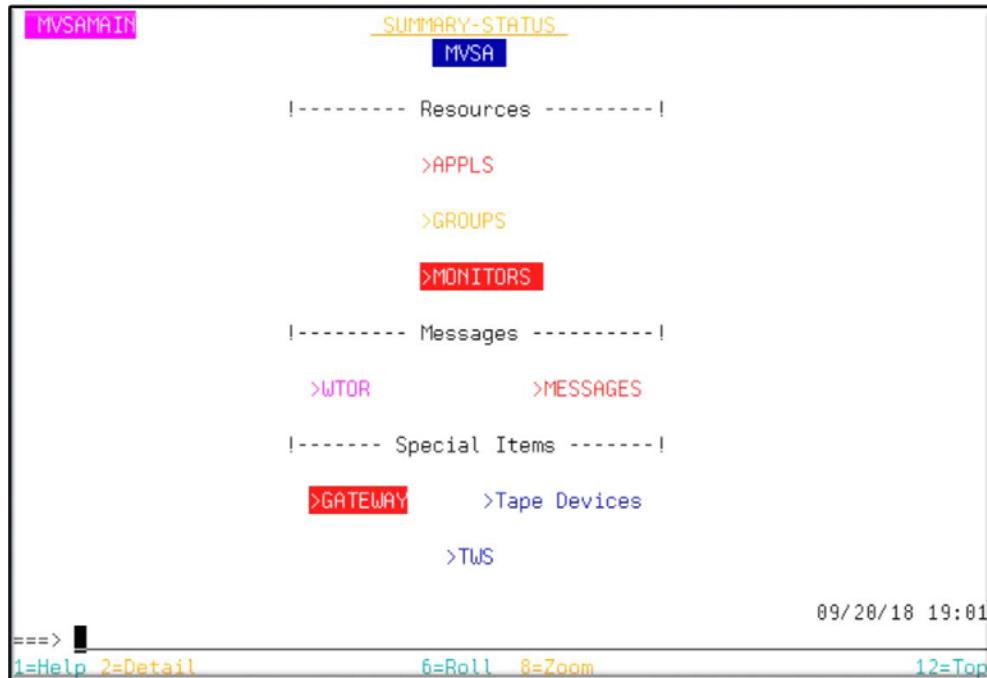
It consists of a set of hierarchical panels, which are delivered as samples. It is set up by your system programmer during the customization of the automation product. SDF is highly customizable. The system administrator can also define user-specific panel layouts. The Status Display Facility has dynamically updated panels showing color-coded status conditions. If the status of an object changes, it is dynamically reflected on the panels and is propagated up the hierarchy.

The graphic shows an example of the highest level panel (also known as the root panel or system summary panel). It runs on the SDF focal point system and can be customized to show systems and hardware resources, enterprise wide.

On the root panel, each system is displayed in a color that reflects the highest priority status of the resources in that system. If a resource in a system changes status, the system changes color to reflect the new status. By observing this panel, you can see status changes in any of your systems.

## Example of a system status summary

3270 based Status display facility (SDF)



© Copyright IBM Corporation 2019

1-41

*Example of a system status summary*

This panel is an example of a system status summary that shows multiple status fields like 'APPLS', 'GROUPS', and so on. Here you see a red status in the status field of MESSAGES. This indicates that there are problematic messages on the system.

The setup of SDF is very easy based on the delivered samples and generation of panels. SDF is highly customizable. It is set up by your system programmer during the customization of SA z/OS.

SDF displays the status of various resources in a z/OS system, and in other z/OS systems that specify this system as an automation focal point.

You use SDF to monitor the status of application software, WTORs, and gateways on your systems from a NetView or operator console.

The status conditions that can be displayed by SDF include those for:

- Applications and subsystems
- Monitor resources
- WTORs
- Gateways (SA z/OS only): A gateway is a combination of a NetView-NetView task session and two automation operators (one on each of two systems) that allows communication of messages, commands, and responses between the two systems.
- Application groups
- Spool usage problems from z/OS subcomponents
- Exceptional messages
- Processors
- Ensembles

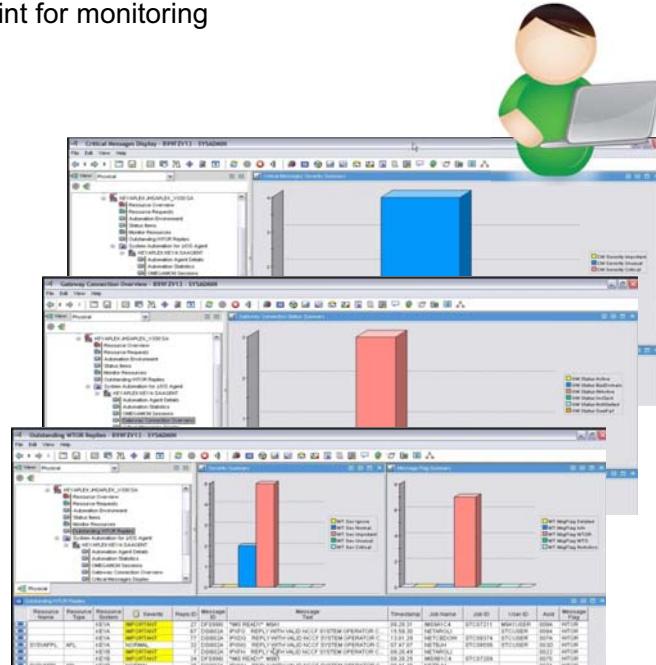
## **How to Check Resource Status Conditions**

Normally when you use SDF to monitor resources, you observe the highest level panel until you see a system change color. This color change indicates a change in status in one of the resources in the system. To determine which resource, and to see more information on the status change:

1. Select the system (or subsystem) that shows a status change by moving the cursor to it using the Tab key or the mouse.
2. Press PF2. This takes you to the Detail Status Display panel for the resource that caused the system to change color. If more than one resource changes status at the same time, SDF shows the information for the system with the highest priority status first. You can press PF8 to page through the Detail Status Display panels for other resources on the system.
3. With the F8 key, you can zoom to the next lower level

# Operator user interface: Tivoli Enterprise Portal

- Tivoli Enterprise Portal is the integration point for monitoring
  - OMEGAMON XE
  - Tivoli Monitoring
  - Tivoli composite application monitors
- Automation adds its operational views to the TEP presented side by side with availability data
- Capabilities
  - Resource overview, health, and status item workspaces
  - Situation monitoring and visualization
  - Graphical summaries combined with detailed tabular views
  - Context-sensitive linking between workspaces
  - Display critical messages and WTORS
  - Display topology view



© Copyright IBM Corporation 2019

1-42

## Operator user interface: Tivoli Enterprise Portal

The Tivoli Enterprise Portal (TEP) support allows you to monitor the status of automation on z/OS systems and z/OS sysplexes (SA z/OS only) using a TEP client. The client is the user interface for the monitoring agents. The monitoring agent uses the Tivoli Monitoring Services infrastructure, which provides security, data transfer and storage, notification mechanisms, user interface presentation, and communication services for products in the IBM Tivoli Monitoring and OMEGAMON XE suites in an agent-server-client architecture.

The monitoring agent is installed on the systems that you want to monitor and passes data to a hub Tivoli Enterprise Monitoring Server (monitoring server), which can be installed on z/OS, Windows, and some UNIX operating systems. The monitoring server communicates with the Tivoli Enterprise Portal Server (portal server), which then communicates with the portal client.

The TEP is also the integration point for monitoring OMEGAMON XE, and other monitors including monitors created by clients.

With the automation product, the operational views are added to the TEP across several workspaces.

On these workspaces, you get an overview on the resources with all their different states.

The graphical summaries are combined with the detailed tabular views.

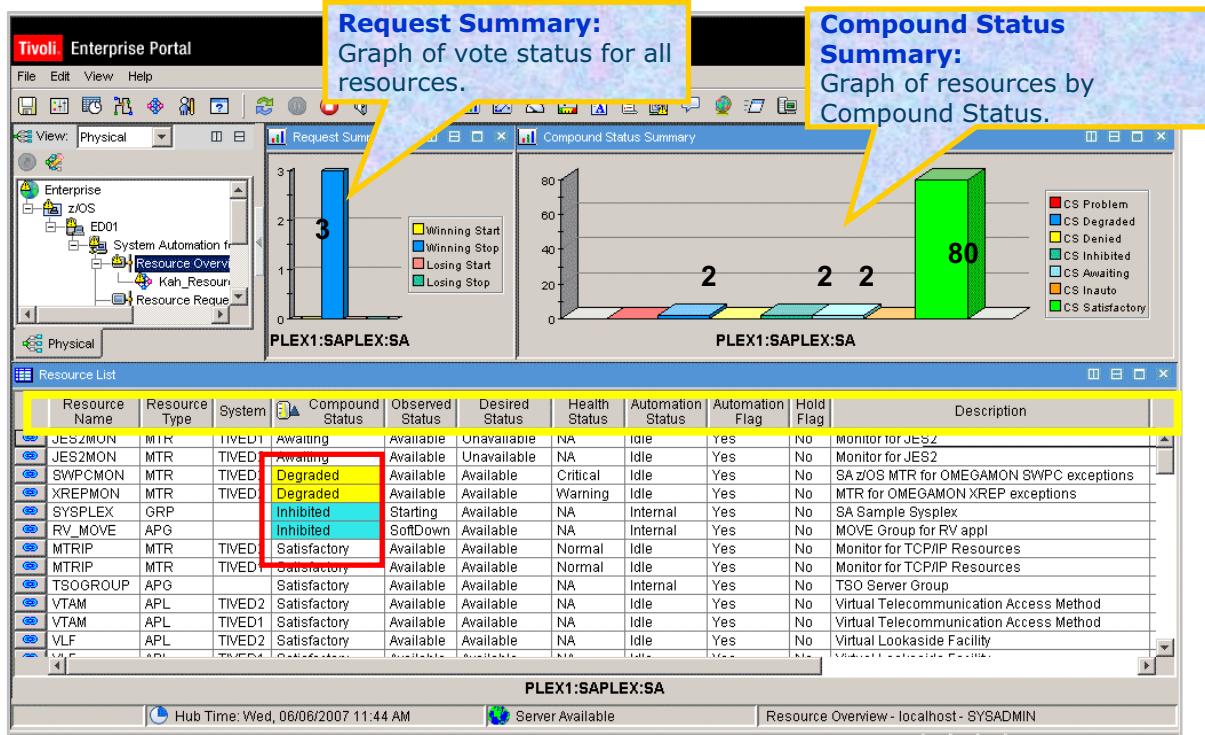
You can jump directly from one workspace to another using the context-sensitive links.

Additionally you see information like the outstanding WTORs and the relationships with the topology view.

The automation workspaces are *display only* but are extendable by *take actions* for start, stop, and so on.

All automation objects and attributes are available for OMEGAMON situations. When the specified condition consisting of specific attributes of specific objects becomes true, the situation is visualized on the Tivoli Enterprise Portal to alert operators. Expert advice and simple actions can be defined too.

# Tivoli Enterprise Portal workspace



© Copyright IBM Corporation 2019

1-43

## Tivoli Enterprise Portal workspace

The Resource Overview workspace provides an overview of resources, their state and requests to them.

The predefined workspace contains the following views:

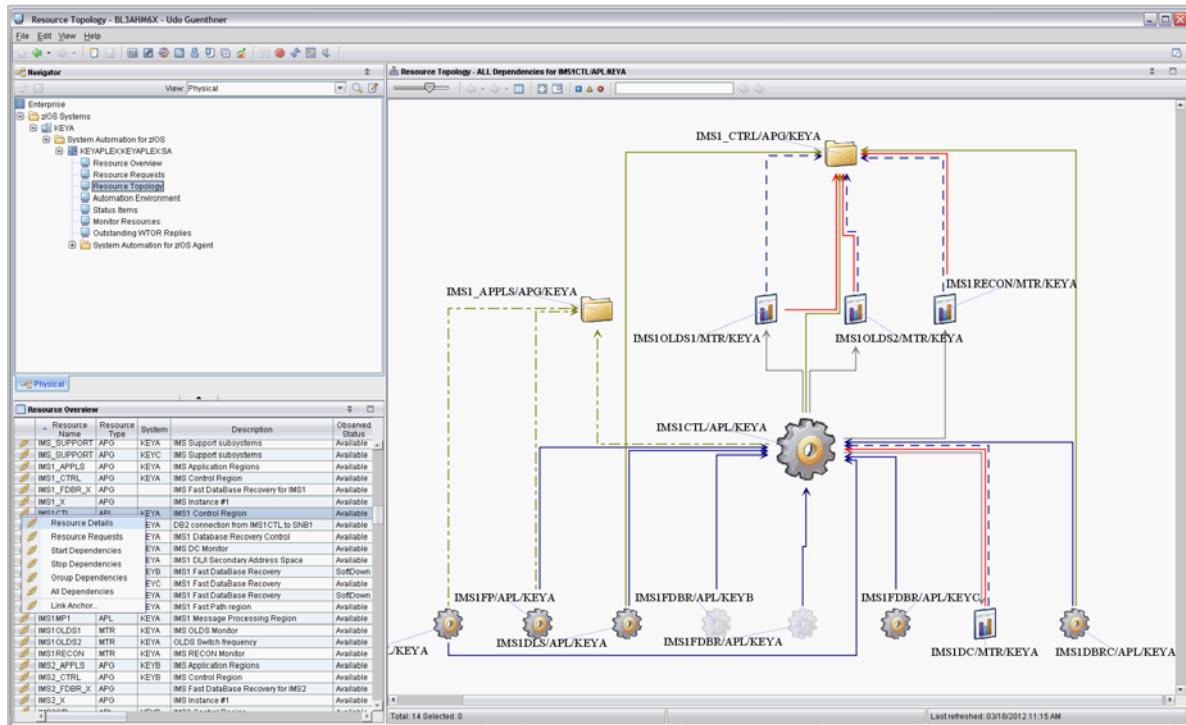
- The Request Summary bar chart shows the distribution of request types (start versus stop, winning versus losing) across all requests. A link is provided to navigate to the Resource Requests workspace for detailed information about the requests.
- The Compound Status Summary bar chart shows the distribution of compound states across all resources.
- The Resource List table lists all resources, the various automation states, and other descriptive information. Links are provided to navigate to the Resource Details workspace for further details about a particular resource or to navigate to the Resource Requests workspace for detailed information about the requests.

## Filter support

Special filter support exists for the Resource Overview workspace because this list can be large. Although all filtering is done on the Tivoli Enterprise Portal (TEP) side, some filters for this workspace are also implemented on the NetView side. This can help reduce the amount of data that must be transferred from the automation manager. The following filters are supported:

- Operand may be equal to or contained in the resource name.
- Operand may be equal to or contained in the resource type.
- Operand may be equal to or contained in the resource system.
- Status may be equal to the compound, observed, or health status and a multiple of such expressions are logically ORed.
- Status may be not equal to the compound, observed, or health status and a multiple of such expressions are logically ANDed.
- Combinations of different attributes are only allowed within a single filter row, that is, the attributes are logically ANDed.

# Tivoli Enterprise Portal resource topology



© Copyright IBM Corporation 2019

1-44

## Tivoli Enterprise Portal resource topology

The Tivoli Enterprise Portal topology view that combines the dependency graph with the resource status and other important runtime data about the resource that is managed and automated by the automation product allows operators to notice odd system behavior. The view can also be used by the automation administrator to spot misconfiguration but also to verify the correct relationship definitions previously made in the customization dialog.

The Resource Topology workspace displays dependencies between the resource in focus and other resources. Four modes are available:

- Start Dependencies: Displays all resources that must be up so that the resource in focus can be started
- Stop Dependencies: Displays all resources that must be down so that the resource in focus can be stopped
- Group Dependencies: Displays all groups where the resource in focus is a member of and, if this resource is a group, all of its members and submembers
- All Dependencies: Displays all resources that have direct dependencies with the resource in focus

The predefined workspace contains the following views:

- The Resource List lists all resources as in the Resource Overview workspace.
- The Resource Topology shows a graphical view of resource dependencies.

# Lesson 7 Key integration capabilities

## Lesson 7: Key integration capabilities

- NetView is an excellent integration platform: Many interfaces, great connectivity
- Geographically Dispersed Parallel Sysplex
- Alerting to SDF, System Automation for Integrated Operations Management, OMNIbus, Event Integration Facility, problem management
- Workload automation products can create requests to start or stop applications
  - Comprehensive integration with IBM Workload Scheduler
- Pro-active automation with OMEGAMON XE
- Monitoring of CICSplex System Manager events
- Reporting: Application state changes written to System Management Facility
  - System Automation Application Manager availability, startup, and shutdown reports
- End-to-end automation of applications running in:
  - Multiple sysplexes or cross platform in z/OS, Linux, AIX, and Windows

### *Key integration capabilities*

On this slide are key integration capabilities in the automation product.

**NetView** is an excellent integration platform: It has many interfaces and great connectivity. In IBM Automation Control for z/OS, not all of the interfaces are available.

The service offering Geographically Dispersed Parallel Sysplex provides disaster recovery, not only for z/OS but also for Linux on System z and even non-System z platforms. Geographically Dispersed Parallel Sysplex requires NetView and System Automation for z/OS.

**Escalation** to various notification targets can be set up easily using pre-defined or user-defined alert points on a resource level to perform the following functions:

- Create incidents in IBM Tivoli Service Request Manager or other products. IBM Tivoli Directory Integrator is used as an interface.
- Create events in IBM Tivoli Netcool/OMNibus through Event Integration Facility (EIF).
- Alarm staff using email, pager, or SMS using IBM System Automation for Integrated Operations Management.
- Display exceptional messages and status for resources that have SDF as notification target on the Status Display Facility (SDF).

**Workload automation** products can create requests to start or stop applications by using the request driven interface.

Comprehensive integration with IBM Workload Scheduler exists. IBM Workload Scheduler has an automation workstation that allows to send any command to automation, synchronously and asynchronously. Automation can display the status of the current plan and also query whether today is marked as a holiday in the IBM Workload Scheduler calendar.

### Pro-active automation with OMEGAMON XE

To overcome the islands of monitoring and automation, automation must incorporate monitoring. The IBM Monitoring products OMEGAMON XE but also NetView provide a wealth of hardware and software performance and availability data.

System Automation for z/OS and IBM Automation Control for z/OS integrate with OMEGAMON XE. System Automation for z/OS and IBM Automation Control for z/OS can use OMEGAMON classic exceptions and OMEGAMON XE situations for exception-based automation.

The **monitor resources** allow to define active and passive monitors, thresholds, and also actions to resolve performance problems.

The monitor result is translated into an application health status that is propagated to monitored resources and aggregated into their compound status to allow easy detection of performance problems.

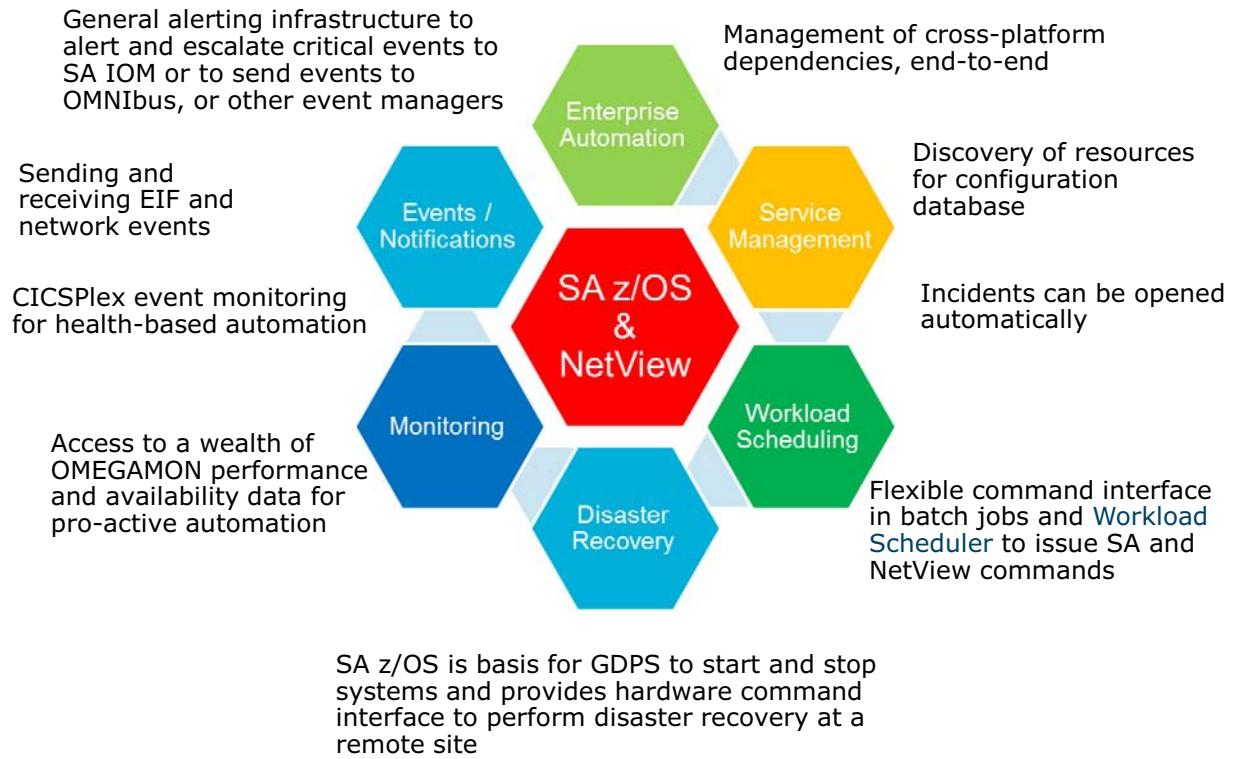
Pro-active automation is able to detect performance problems and to act before a failure occurs.

**CICS Automation** allows the monitoring of events that are issued by **CICSPlex System Manager** (CICSPlex SM). When an event is received, the event severity is mapped to the health status of a related monitor resource and, in addition, defined recovery commands are issued.

The automation product collects and records job-related information, and writes System Management Facility (SMF) records at specific events in the lifetime of a resource. The INGPUSMF batch utility produces a report file that you can import into a spreadsheet. You can also convert and write the report into DB2 tables that are provided and used by the IBM System Automation Application Manager.

Integration into end-to-end automation that is provided by IBM System Automation Application Manager, is only available with System Automation for z/OS, however -to-end automation is now provided by SA z/OS itself.

# System Automation integration points



© Copyright IBM Corporation 2019

1-46

## System Automation Integration Points

This is an overview of the System Automation integration points.

In addition to the already described integration points, the resources managed by System Automation can be exported to a discovery library adapter (DLA) file that can be imported in XML format by configuration management databases.

System Automation commands can be also entered from TSO and batch jobs.

# Lesson 8 More Automation

## Lesson 8: More automation

- **End-to-end automation**
- Application pacing
- Looping address space suppression
- IPL complete notification
- **Sysplex automation**
- Job log monitoring
- Availability and recovery time reporting
- Relational Data Services (RDS)

### More Automation

This lesson provides an overview of:

- **End-to-end automation**
- Application pacing
- Looping address space suppression
- IPL complete notification
- **Sysplex automation**

The other automation items are only discussed briefly below:

- Job log monitoring is designed to monitor JES spool files only. For SA z/OS controlled jobs, you have to define the monitoring interval, the messages and data sets to be monitored for each job in the customization dialog. Message INGY1300I wraps the parsed messages.  
job log monitoring is stopped for suspended resources. To enable job log monitoring again, you

must do it manually using INGJLM, or make sure that the resource is first resumed and a start request or vote exists for automation to start it.

- Availability and Recovery Time Reporting

Automation collects and records job-related information, and writes System Management Facility (SMF) records at specific events in the lifetime of a resource. The INGPUSMF batch utility produces a report file that you can import into a spreadsheet.

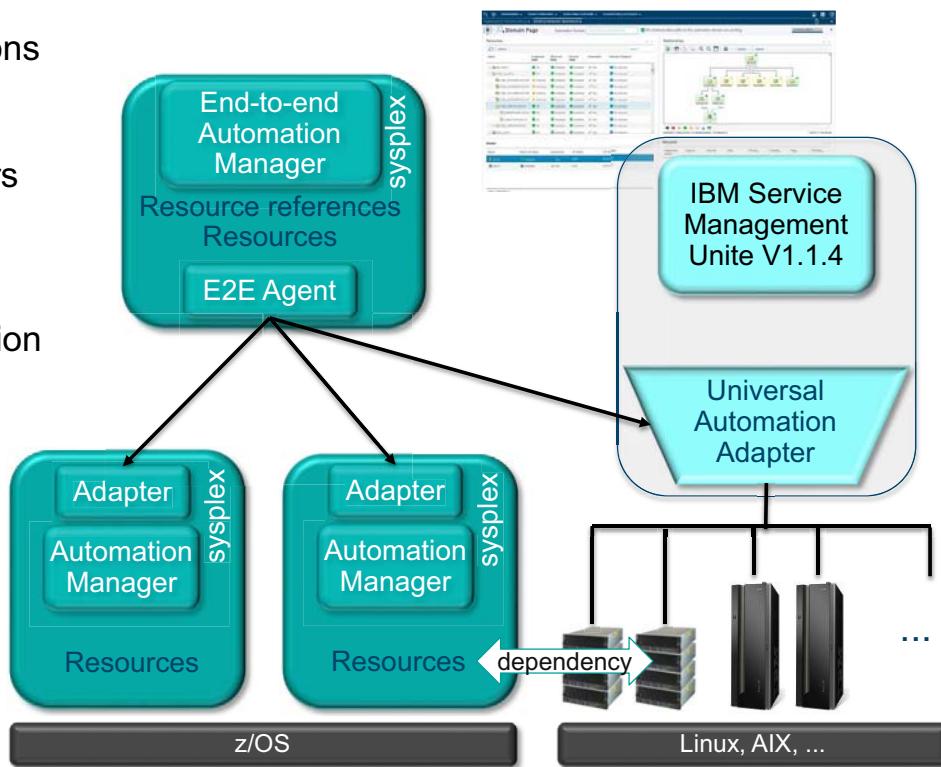
- Relational Data Services (RDS)

The INGRDS command provides basic access methods for built-in relational data tables.

# End-to-end automation (E2E)

Control applications running on z/OS sysplexes and on distributed servers

Manage cross platform automation dependencies



© Copyright IBM Corporation 2019

1-48

## End-to-end automation

With SA z/OS V4.1, the automation manager provides cross sysplex automation capabilities. With APAR OA55386, SA z/OS V4.1 extends its automation capabilities to cross platform resources.

Prior to V4.1, SA z/OS automation manager automated resources in the same sysplex (or SAplex).

Now the SA z/OS automation manager is able to automate resources and their dependencies

between multiple sysplexes (or SAplexes) and cross platform resources using **resource references**.

In this case, the primary automation manager (PAM) is also called E2E manager.

This cross sysplex/platform automation is also called **end-to-end automation**.

The SA z/OS **end-to-end automation adapter** connects the automation domain with the SMU server or with the **E2E agent** running in the sysplex of the E2E manager.

The **Universal Automation Adapter** (UAA) enables Service Management Unite (SMU) automation to monitor, operate, and automate resources running on non-z/OS systems. It connects to the remote non-z/OS systems using Secure Shell (SSH). The Universal Automation Adapter is part of SMU. It's installed with SMU and runs on the same Linux.

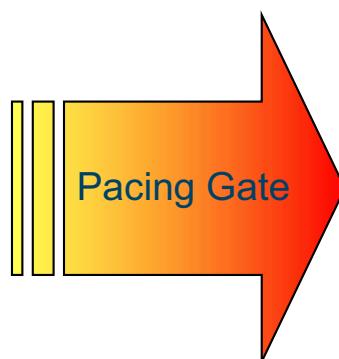
# Application pacing

## Problem:

CPU peaks occur when starting or stopping bulk applications e.g. at IPL

## Solution:

“Only n workloads can pass the pacing gate at a time”  
“Others have to queue up behind those passing the gate.”



© Copyright IBM Corporation 2019

1-49

## *Application pacing*

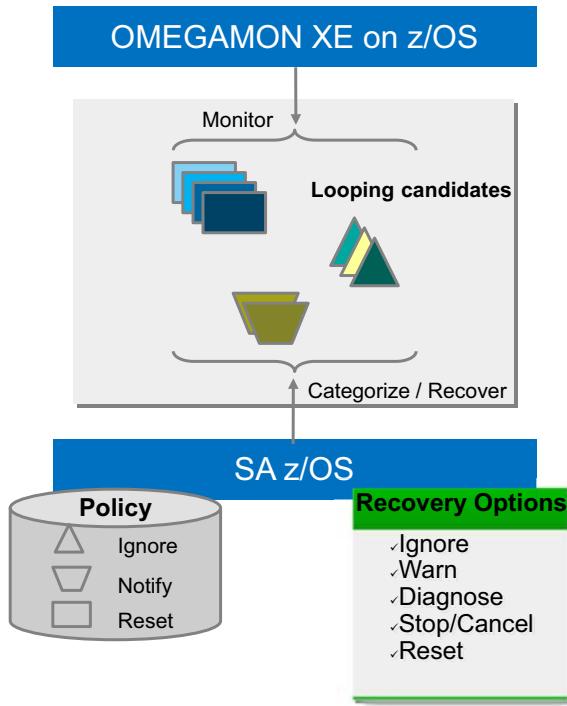
The resource consumption of applications (that is, of type APL) during the start and stop phases can vary a lot. While many applications are started or stopped very quickly, other applications may consume a lot of CPU resources and may even tend to dominate how the system assigns CPU resources among the started tasks in need.

To facilitate the management of those applications aiming for efficient resource utilization while avoiding customization efforts, the application pacing capability can be used to control how many applications of a kind can be started or stopped at the same time. For this, any application that is eligible to receive a start or stop order from the automation manager has to 'transit' through a defined Pacing Gate.

If the number of applications that are currently in transition reaches the maximum concurrency level, additional applications are held back (waiting) until another application finished this transition and reached the final desired status or terminated during that transition.

In order to use application pacing, you need to define one or more Pacing Gate entries using the Customization Dialog, option 13 (PAC) and specify the concurrency level of starting and stopping applications. Next, you assign those applications that you want to more tightly control to a Pacing Gate. This is done by selecting a pacing gate using the PACING GATE policy.

# OMEGAMON looping address space suppression



## Situation

The overall z/OS system utilization and also the utilization of individual started tasks / jobs is understood for normal and peak hours

## Problem

Detect when started tasks / jobs show abnormally high CPU utilization.  
Prevent that these types of work can dominate the system



## Solution

OMEGAMON XE on z/OS data is analyzed for high CPU utilization and categorized into different types of work to allow specific recovery actions through policy – not programming!

© Copyright IBM Corporation 2019

1-50

### *OMEGAMON looping address space suppression*

This function combines OMEGAMON's ability to provide deep insight into the workings of a z/OS system with System Automation's pragmatic approach to automation to detect, highlight and automatically manage address spaces that have gone into CPU intensive loops.

This extends System Automation's ability to identify and recover failing jobs, enabling it to handle a class of failures that have previously been very difficult to manage.

OMEGAMON XE on z/OS data is analyzed for high CPU utilization and categorized into different types of work to allow specific recovery actions through policy – not programming!

You need:

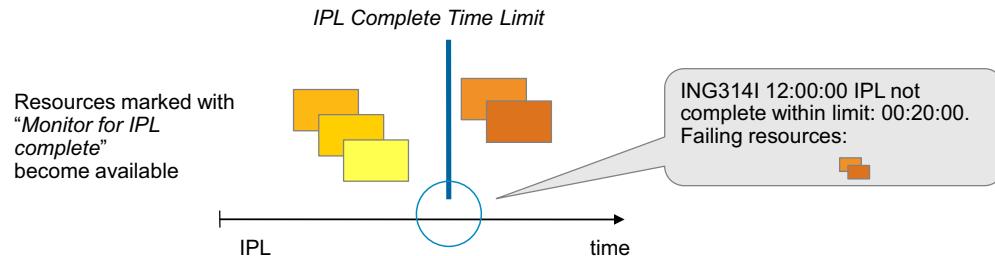
- OMEGAMON for z/OS installed and running on the system where you want to run the procedure
- A TEMS server up and running that is in communication with OMEGAMON for z/OS

Recovery categories and actions are defined using the following pseudo messages:

- INGCATEGORY: category rule for WLM Service Class, address space job type, and jobname
- INGRECOVERY: Recovery Category and pass: warning, diagnostic, stop, quiesce, reset

## Hey, your IPL is complete or takes longer

- Operators often don't know when an IPL is finished
- With SA z/OS, the administrator can define the automation resources of interest (APL, APG and MTR) and an IPL time limit
- SA z/OS monitors the progress of these resources during IPL and reports status to operators via messages
  - If all completed in advance (ING313I)
  - If not all completed within the time limit (ING314I)
  - When the last resource finally completed after the time limit (ING315I)
  - Example:



© Copyright IBM Corporation 2019

1-51

*Hey, your IPL is complete or takes longer*

Operators often don't know when an IPL is finished.

The administrator can define the automation resources of interest (APL, APG and MTR) and an IPL time limit.

SA z/OS monitors the progress of these resources during IPL and reports status to operators via messages:

- If all completed in advance (ING313I)
- If not all completed within the time limit (ING314I)
- When the last resource finally completed after the time limit (ING315I)

Suspended resources are not exempted from IPL-complete monitoring

**Implementation:**

- A time period where the IPL is expected to be complete is specified on a per system basis via an attribute for the SYS resource defined in entry type 'System Defaults (SDF)'.
- On APLs, APG, and MTRs there is a flag 'Monitor for IPL complete', which can be customized to YES or NO.
- Whether or not ING315I is given is customizable via an advanced automation option:  
`AOF_AAO_IPL_COMPLETE_MSG`  
COND Specifies whether only message ING313I is given if all resources reached the AVAILABLE status within the expected time limit and ING314I if this is not the case.  
ALWAYS Specifies that if ING314I has been given and the resources become AVAILABLE later (at any point in time) ING315I will be issued additionally.

## **Sysplex automation**

- Continuous availability for couple data sets (CDS)
  - Automatic allocation of alternate couple data sets
- System logger: Reformat primary and alternate LOGR CDS
- Managing coupling facilities (CF)
  - Manage CF structures
  - Drain, remove, or reintegrate CF...
- Automate long running ENQs or hung commands
  - Define hung commands, allowed execution time and action
- Auxiliary storage shortage recovery
  - Dynamically allocating spare local page data sets
- System removal from sysplex
  - Automating IXC102A and IXC402D messages

### *Sysplex automation*

- Continuous availability for couple data sets (CDS) is provided by automatic allocation of alternate couple data sets
- System logger: Reformat primary and alternate LOGR CDS  
Two problems that can arise in connection with the log stream data sets are a shortage of

directory space in the LOGR CDS and incorrect share options for the log stream data sets. SA z/OS provides the following recovery actions for these problems:

- The primary and alternate LOGR CDSs are automatically re-sized if there is a directory shortage
- The operator is notified if the share options for log stream data sets are not defined correctly
- Managing coupling facilities (CF)
  - Manage CF structures
  - Drain, remove, or reintegrate CF...
- Automate long running ENQs or hung commands
  - Define hung commands, allowed execution time and action
- Auxiliary storage shortage recovery:  
Dynamically allocating spare local page data sets
- System removal from sysplex:  
Automating IXC102A and IXC402D messages

# Lesson 9 Configuration assistant

## Lesson 9: Configuration assistant

Replace traditional customization with

- Complete options file
- Run configuration assistant batch job
- Use generated jobs to allocate runtime data sets
- PARMLIB, PROCLIB, VTAMLST members generated
- SAF definitions generated

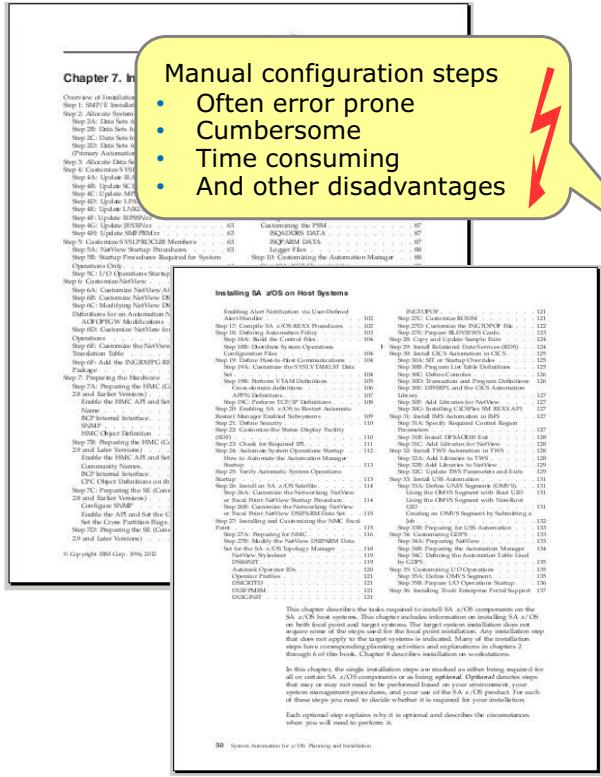
### Configuration assistant

A new configuration assistant function is available.

Instead of manually adapting configuration jobs, start procedures, and initialization files to your environment, this assistant generates these files for you.

- Faster time to value through the Configuration Assistant for the runtime environment.
- Automation comes with five predefined operator roles. It avoids user exits to provide granular security and to exploit z/OS's security access facility and thus leverages existing security governance that is already in place in an enterprise's mainframe environment.

# Traditional approach



Read the Installation manual:

- Decide which of the many installation steps apply to your z/OS environment

Perform those steps by:

- Adapting all the identified sample files
- Completing your environmental data at multiple places spread across the sample files

And do all this in a consistent way

## Traditional approach

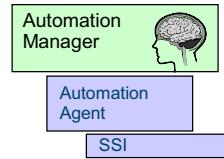
The traditional approach for installing and configuring a product is to read the manuals and then you have to decide which of the many steps apply to your local installation. You perform these step-by-step as described in the thick installation manuals. This means that you adapt the delivered samples at multiple places spread often over several sample files.

The challenge for the system administrator is to do this all in a consistent way. So, normally the system administrator crosses their fingers that this works the right way.

# Configuration assistant

## Overview

- Simple configuration to get base automation components running
  - Automation manager
  - Automation agent
  - Subsystem interface
- Use of **configuration assistant** that automates majority of configuration process
- Keep number of configuration variables down at the necessary minimum
- Benefit from lab **experience** using “standard” option set (stylesheet)
- Post install/**configuration verification** to ensure all necessary steps (in particular steps that must be done by other persona) have been completed

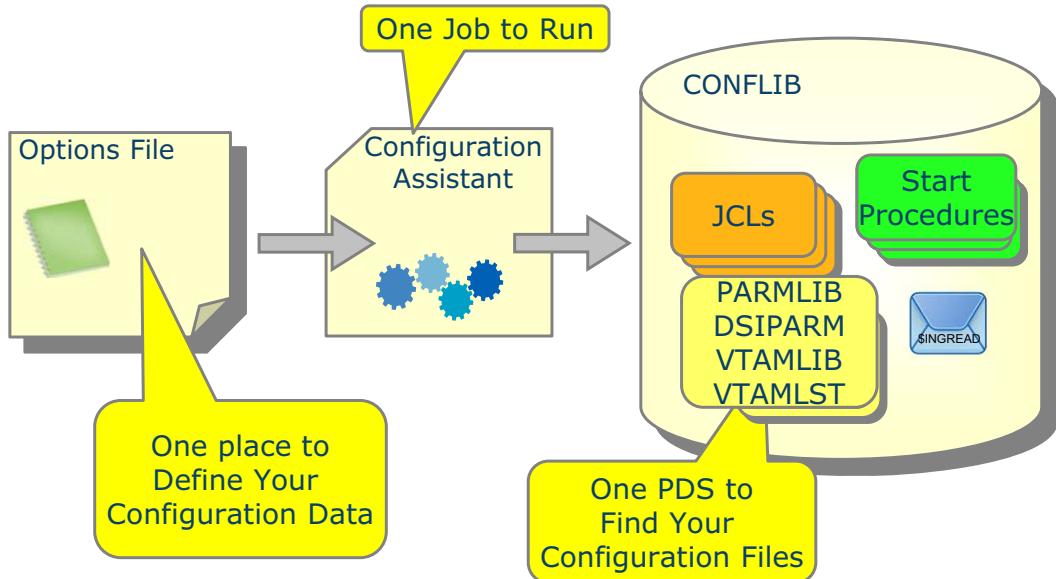


### Configuration assistant

After you installed the product with SMP/E, the configuration assistant comes into play. It covers the actions to be done when the product target libraries are available to your system environment. The configuration assistant sets up an environment so that the base automation components are running. These are the automation manager, the automation agent, and the subsystem interface.

The configuration assistant automates most of the configuration steps. The concept is that the number of configuration variables is as low as absolutely necessary (approximately 30). When using the configuration assistant, you benefit from the experience of the IBM Böblingen lab. After the configuration assistant has completed, a verification of the generated setup can be done against your system environment.

# Configuration assistant: Concept



Generating Configuration Files supporting system symbols.

© Copyright IBM Corporation 2019

1-56

## Configuration assistant: Concept

The concept of configuration assistant is to have one place to store your configuration data. Then, have one job to process them. Then, have one output PDS where all your JCLs, start procedures, and parameter files are stored. Now the SYSPROG can inspect the members and, if it is required, can adapt them to the local needs.

The CONFLIB data set contains these items:

- Jobs to allocate all data sets and USS paths that are required by SA z/OS during runtime
- Procedures to start the components of SA z/OS to be copied to your target SYS1.PROCLIB
- Runtime configuration members for both Automation Manager and Automation Agent
- Parameter files that are ready to be copied to your target SYS1.PARMLIB
- VTAM definitions that are files ready to be copied to your target VTAMLST
- Jobs to delete data set files and USS paths in case you must reconfigure or delete SA z/OS again
- A job to verify the success of the installation and configuration process.

# Summary

- Business applications require high availability
  - Downtime can cause lost sales and customer shifts to competitors
- Automation provides **high availability** for z/OS applications including UNIX
- **Policy-based plug and play automation** using best practices can replace scripts
  - Faster time-to-value
  - Much reduced maintenance burden
  - Less human error
- **Goal driven automation** to keep applications in line with business goals
- **Grouping** of resources for reduced complexity and management at business application level
  - Status aggregation
  - Server and move groups help achieving availability targets
- **Relationships** between resources for accelerated startup and shutdown, and correct recovery
- **Manage by state**, not by message
- Integration with monitors and OMEGAMON to increase automation level with **pro-active automation**



© Copyright IBM Corporation 2019

1-57

## Summary

Business applications require **high availability** as downtime can cause lost sales and customer shifts to competitors.

Automation provides high availability for z/OS applications including USS.

**Policy-based automation** is a sophisticated methodology that allows to easily incorporate business goals into an automation framework.

IBM ships **Plug and Play automation** that includes best practices automation for many z/OS applications. This can help to:

- Reduce time and effort in creating a new policy or updating one
- Improve automation quality and reduce human error

**Goal driven automation** greatly simplifies operations. System operators just request what they want. Automation takes care of any dependencies and goals that are affected. It can resolve conflicting goals and even can remember goals.

**Grouping of resources** and definition of aggregate or business applications can greatly reduce the complexity of automation definition and operations.

Exploiting groups is beneficial in many ways:

- Automation definition and operations can be greatly simplified through the grouping of resources and even business application definitions
- Groups let you monitor important business applications and help verify that everything they require is available
- Groups can make operations easier by showing the aggregated status of resources and by group actions such as startup or shutdown
- Through groups, operators are freed from knowing the various pieces that make up an application, their dependencies, how to start or stop them, and so on

Server groups control that and how many group members are started. Group members can represent, for instance, application servers.

**Relationships** between resources for accelerated startup and shutdown, and correct recovery.

Resources can have complex dependencies of different kinds inside and outside of an application. The automation product gives you the power to define these dependencies, so that applications get what they need, are started in the right order as quickly as possible and are shut down fast without interference.

**Manage by state**, not by message.

Automation knows about more than 20 different states and therefore knows exactly what's going on and what the options are.

Although automation uses messages to update the status and also can automate messages, it is the status change that triggers automation. For example, dependent resources are started after the parent is up or a dependent resource is forced down when the resource it depends on terminates. Also, messages are forgotten, but the status is preserved even across restarts and IPLs.

**Monitor resources** enable integration with any monitor and include easy to exploit OMEGAMON access to update the health status of monitor resources that can trigger **pro-active automation**.

The **health status** of monitor resources is propagated along monitor relationships to applications and is aggregated into the **compound status** which reflects the overall resource status.

## Unit Summary

Now that you have completed this unit, you can perform the following tasks:

- Describe IBM System Automation for z/OS and IBM Automation Control for z/OS and their capabilities
- Describe the components of the product
- Describe policy-based and goal driven automation
- Describe the product's automation capabilities
- Describe its key operations features
- Describe its integration, additional automation, end-to-end automation, and the configuration assistant

### *Unit Summary*

You should now be able to describe the automation product overview, its components, policy-based and goal driven automation, key automation and operations features, integration, additional automation, end-to-end automation, and the configuration assistant.



## 2 Architecture and concepts

IBM System Automation for z/OS 4.1



## Unit 2: Architecture and concepts



© Copyright IBM Corporation 2019

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

# Objectives

When you complete this unit, you can perform the following tasks:

- Describe the architecture
- Describe the automation agent role and operation
- Describe the automation manager role and operation
- Explain the key automation concepts
- Describe goal driven automation
- Explain the automation statuses and their effect on automation
- Describe automation policy for applications
- Describe resource dependencies and relationships
- Provide an overview of application groups
- Explain automation flags, threshold processing, message policy, and notify operators
- Describe end-to-end automation architecture

## *Objectives*

When you complete this unit, you can perform the following tasks:

- Describe the architecture
- Describe the automation agent role and operation
- Describe the automation manager role and operation
- Explain the key automation concepts
- Describe goal driven automation
- Explain the automation statuses and their effect on automation
- Describe automation policy for applications
- Describe resource dependencies and relationships
- Provide an overview of application groups
- Explain automation flags, threshold processing, and notify operators

# Lesson 1 Architecture

## Lesson 1: Architecture

- What is a software resource?
- Manager-agent design
- Automation manager role and environment
  - Managed resources, status and flags
- Automation agent role and overview
- z/OS message flow
- The role of the operator

© Copyright IBM Corporation 2019

2-3

### Architecture

Lesson 1 covers the architecture, especially the manager and agent roles, important characteristics, configuration including data sets, message and automation flow. Also covered are the role of the operator, the managed resources, their manager status and flags.

## What is a software resource?

- Also known as application or subsystem
- Attributes of a resource:
  - Can be started, stopped, and monitored
  - Can have dependencies to other resources
  - Can be combined (grouped) to application groups
- Defined in a policy database
  - Applications
  - Subsystems
  - Started tasks
  - VTAM nodes
  - Databases
  - UNIX System Services resources
  - Monitors (cannot be a member of a group)
- End-to-end automation uses resource references

*What is a software resource?*

**Resources or applications** are automated items that are defined in the policy database. The automation agents start, stop, and monitor automated resources. On some automation panels they are called resources; on others they are called applications or subsystems. Neither term is precise, and a better description might be *automated resource*.

A software resource:

- Can be started, stopped, and monitored
- Can have dependencies to other resources
- Can be combined (grouped) to application groups

Resources are defined in a policy database, examples are:

- Applications
- Subsystems
- Started tasks
- VTAM nodes
- Databases
- UNIX System Services resources including processes, ports, and files
- Anything the user wants to automate including remote, dummy, or pseudo resources
- Monitors (cannot be a member of an application group)
- There are also hardware resources managed by the ProcOps component. Hardware resources are also defined in a policy database. Hardware resources are not controlled by the automation manager.

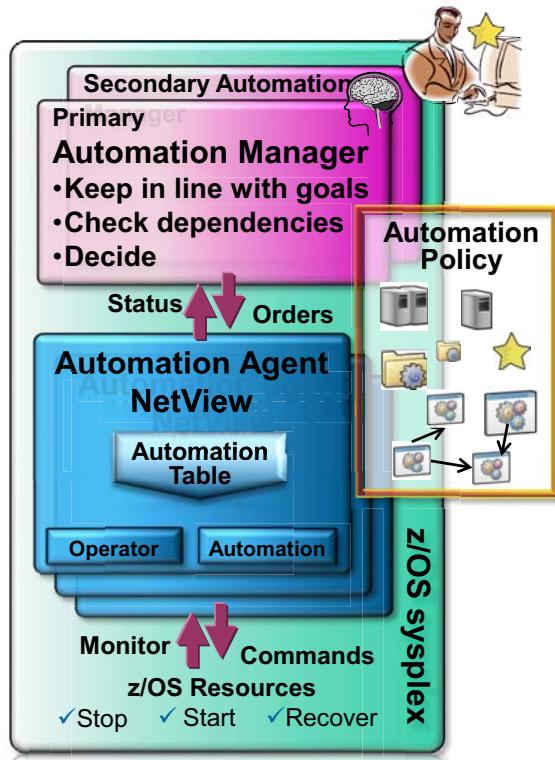


**Note:** There are also hardware resources managed by the ProcOps component. Hardware resources are also defined in a policy database. Hardware resources are not controlled by the automation manager.



**Note:** End-to-end automation uses resource references to real resources in other sysplexes or distributed systems which adds more resource types

# Architecture: Manager-agent design details



- The automation policy defines the resource automation and goals 
- Automation manager (AM)
  - Loads policy and maintains resource status
  - While taking care of dependencies the AM tries to achieve goals by sending start and stop orders to agent
- NetView-based automation agent role:
  - Loads policy as ordered by AM
  - Status monitoring to inform AM
  - Executing AM start and stop orders
  - Operator console
- Operators
  - Control automation
  - Detect and manage problem statuses
  - Override goals using requests and more

© Copyright IBM Corporation 2019

2-5

*Architecture: Manager-agent design details*

## The automation manager

The automation manager orders the agent to load the same policy.

In the automation manager's sphere of control, it maintains status information about each resource, it controls resources, their dependencies to other resources and their desired status (goal). The automation manager uses all of this information to determine which applications to start and stop, and when to do so. The manager does not issue start and stop commands; it sends orders to automation agents which issue the start and stop commands.

## The automation agent

The automation agent runs in the NetView address space which provides the operator console. Access to the agent is through the NetView command facility. Agent commands display and allow some modification of the resource status, automation goals and status, and policy.

The automation agent uses the NetView automation table message-processing function for automation purposes. Messages are generated as the result of some event or situation. Many automation policy definitions use messages to trigger actions. A NetView automation table is the

detection mechanism for messages. The automation product creates entries in the NetView automation table for specific messages. The messages are assigned to a NetView **Autotask** to distribute workload and to ensure serialization of the messages as they are automated. Based on the type of message an action is routed to an automation operator. The action can include update of the agent status for an application or z/OS commands.

The automation agent receives orders from the automation manager and issues commands that are based on defined automation policy.

The agent sends the status changes for each application to the primary automation manager. The primary automation manager sends orders to the automation agent, requesting the start or stop of applications.

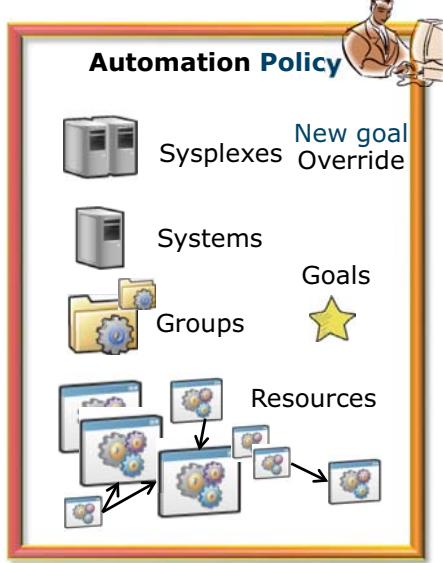
## ***The operator***

The operator role is to control automation, to detect and react on problem statuses, messages, and alerts as well as problem analysis, execution of automation change requests by managing automation, policy and resources.

Operators can use requests to override any goals.

# Automation manager basics

## Automation manager



- Scope is SAplex: System, sysplex, or a subset
- Has central awareness of all resources and their statuses, relationships, and location
- It implements sysplex, system, application group, monitor resources, domains and resource references
- Provides suspendable, goal driven automation including service periods, events and triggers, runmodes, and pacing gates
- Is the single decision-maker for the entire SAplex
- Sends orders to agents to initiate resource starts or stops
- One primary and multiple secondary automation managers with lossless takeover
- Informs the notify operators about important events

© Copyright IBM Corporation 2019

2-6

### Automation manager basics

In addition to the already provided automation manager description:

The automation manager is the central decision-maker in the SAplex.



**Important:** SAplex or "SA z/OS Subplex" is a term used in conjunction with a sysplex. In fact, a SAplex is a subset of a sysplex. However, it can also be a sysplex. For IBM Automation Control for z/OS a SAplex is a system.

The automation manager is aware of all automated resources in the SAplex and implements and manages these resources and policy items:

- Sysplexes and subgroups (GRP and SBG)
- Systems (SYS)
- Applications (APLs)
- Application Groups (APGs)
- Monitor resources (MTRs)
- Remote domains (DMN)
- Resource references (REF)

The automation manager implements suspendable, goal driven automation including:

- Service Periods (SVP)
- Triggers and Events (TRG, EVT)
- Runmodes
- Pacing gates (PAC)

Goals, requests, votes, dependencies, groups, system groups, service periods, triggers and events, monitor resources, automation manager flags, and all states except for the agent status are not known to the automation agent.

This split is transparent for the operator as above mentioned manager data can be displayed and managed from the agent.

There is one active primary automation manager for each defined SAplex. There can be one or more secondary automation managers. Takeover by a secondary automation manager is automatic and lossless

The primary automation manager informs the notify operators, logged on to the agent, about important events.

# Important characteristics

- Controls all automated resources in the SAplex
  - Naming convention is *resource\_name/resource\_type/system*
    - Resource types include APL, APG, SYG, SYS, MTR
    - Examples: TSO/APL/MVSA  
CICS/APG
- Maintains six statuses for each resource
  - Observed status as reported by the automation agent
  - Desired status the goal
  - Automation status what automation is doing with the resource
  - Startability status is it possible to start the resource?
  - Health status from “linked” monitor resource
  - Compound status aggregation of all other statuses and other values
- If *observed status* is not equal *desired status* actions are taken

## *Important characteristics*

For many commands, only applications, application groups, and monitor resources are valid resources. Except for sysplex application groups which have a two-part resource name, resource names are composed of an automation name, a type, and a system name. The format of a resource name is *resource\_name/resource\_type/system*. The automation manager maintains six statuses for each resource that it controls.

1. Observed status as reported by the automation agent
2. Desired status: the goal
3. Automation status: what automation is doing with the resource
4. Startability status: is it possible to start the resource?
5. Health status from “linked” monitor resource
6. Compound status as aggregation of all other states and other values

If the observed status is not equal the desired status automation manager tries to take actions to reach the goal.

## Automation manager flags

- Automation flag (**YES** or **NO**)
  - If set to NO then, these actions occur:
    - No orders are sent from the automation manager until the flag is turned on again
    - Actions from the automation table are still performed
      - Recovery actions, for example
      - The automation manager does not control resource recovery
- Hold flag (**YES** or **NO**)
  - If set to YES, then these actions occur:
    - Resources cannot be started until an operator issues an INGREQ command
    - Also disables the implied MakeAvailable vote
- Suspend flag: whether and how resource is suspended

© Copyright IBM Corporation 2019

2-8

### Automation manager flags

The two automation manager flags are automation and hold:

- The **Automation flag** of the automation manager applies to each resource that is defined to the automation manager. If set to NO, it inhibits the sending of all automation orders for that resource. If set to YES (the normal, default setting), orders are sent freely.
- The **Hold flag** is used to allow the NOSTART option to be selected at agent initialization. If you reply NOSTART to the AOF603D WTOR message during initialization, you are telling the automation manager to ignore any initial start requests for inactive applications.  
If NOSTART is requested, the Hold flag for each inactive applications in that system is set to YES. Setting this flag to YES inhibits start orders only. Any application that is normally set to a status of DOWN during agent initialization is placed in a status of AUTODOWN.  
If NOSTART is not requested at agent initialization, all hold flags for the system are set to NO. The first INGREQ command that is issued against a resource automatically changes the hold flag to NO.

The INGSET command can be used to change the settings of above flags. One possible use is to set the automation flag of an application before shutdown so that it is not started after another IPL.

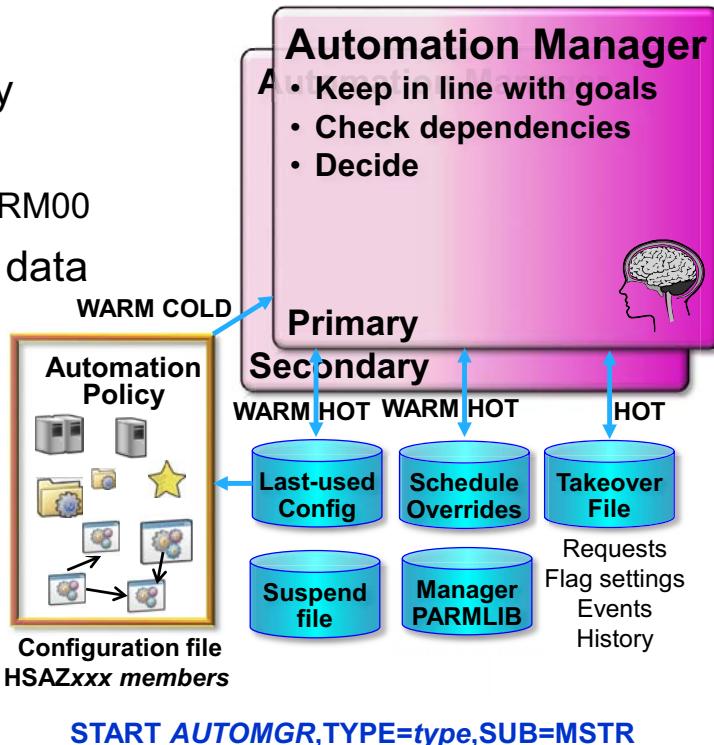
You can dynamically change the hold flag. However, that disables the implied MakeAvailable vote from the automation manager also. Automation manager flag settings are part of the persistent data that is restored on a manager HOT start.

- Implied MakeAvailable votes are also inhibited if the hold flag is set to YES. For start requests after initialization is complete, the Hold flag is set to NO for targeted applications with an agent status of AUTODOWN. An application starts if there are no other votes or inhibitors.
- **Suspend flag:** whether and how resource is suspended.  
automation allows the operator or administrator to suspend a resource. While it is suspended, automation does not attempt to start or stop this resource. Similarly, it does not react on messages that would normally trigger status commands or other commands that are defined in the policy for a message. Most importantly, it does not alert the operations team by exposing an unusual status on any status display (for instance INGLIST or SDF), which operators are normally sensitive to.

# The automation manager environment

Shared data sets:

- Configuration file = Policy
- Manager PARMLIB
  - Initialization member HSAPRM00
- Takeover File: persistent data
- Schedule Overrides
- Name of last-used configuration file
- Suspend file



© Copyright IBM Corporation 2019

2-9

Automation manager environment

## Data sets for all automation managers in a SAplex or a standalone system

**Configuration file:** This is the output data set for the customization dialog when building the automation configuration. The automation policy is stored in this data set named system operations Control File with extension SOCNTL.

It is accessed dynamically by both automation manager and automation agents.

**Manager or HSA PARMLIB:** DD name HSAPLIB points to a partitioned data set which contains information required for the initialization of the automation manager and default values for other operational parameters.

**Takeover file:** The takeover file is a VSAM data set which contains persistent data for the manager to use during HOT starts of the automation environment. Persistent data includes information such as votes, events, automation manager flag changes, history, and application group changes.

**Override file:** The override file is a VSAM data set which contains schedule and resource overrides to defined service periods. Information in the HSAOVR data set is used for WARM or HOT starts of the automation manager.

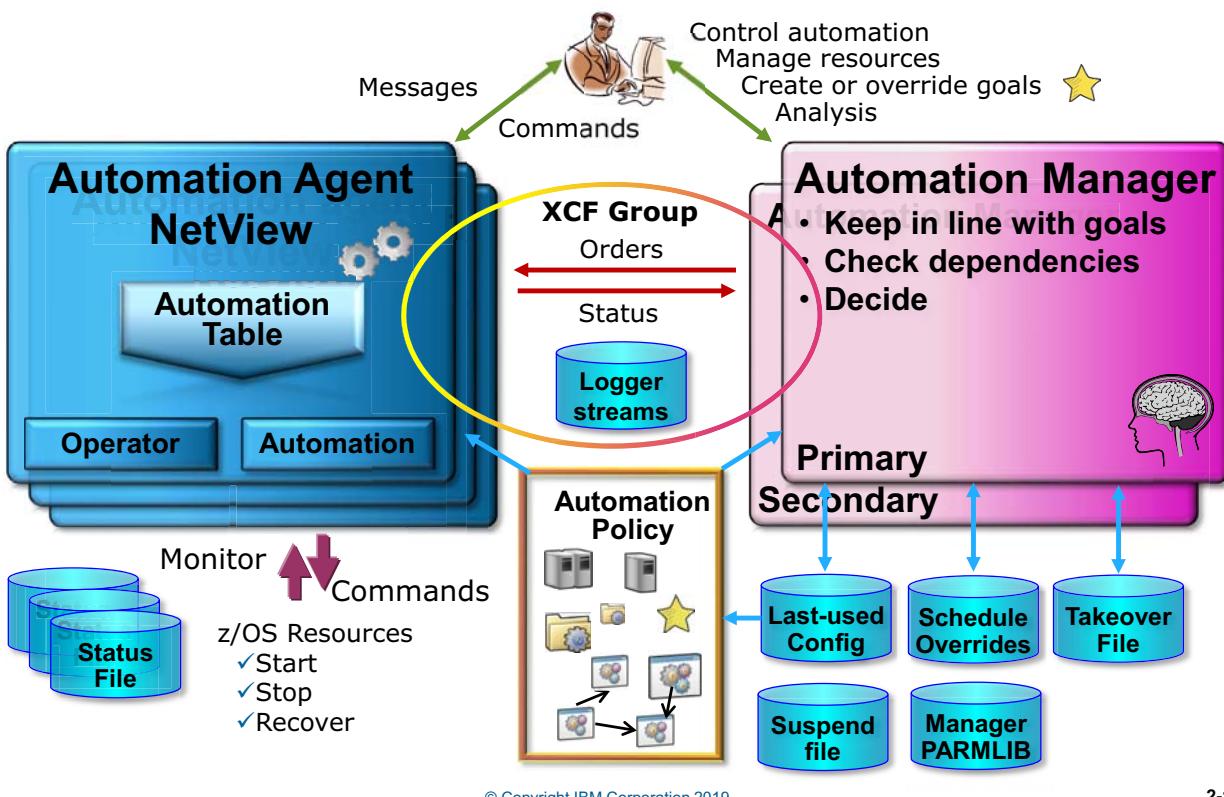
**Configuration information data sets:** DD name HSACFGIN points to a sequential data set which contains the name of the last-used automation manager configuration file. During WARM and HOT starts, the automation manager uses information in the HSACFGIN data set to reconnect to previously used configuration files. The last-used name is updated whenever the automation manager is started COLD and when the configuration files are refreshed.

**Suspend file:** Resources that are already automated can be suspended and resumed during runtime using the INGSUSPD command. But there might be situations, where it is useful or necessary to "plan" resources in the policy, which are not ready to be used. For instance, an SA administrator adds an application to the policy and the application is not yet installed. If a policy with such "planned" resources is loaded, they are automated (started and stopped) and may produce error situations until somebody suspends them manually with the INGSUSPD command. To prevent such error situations, it is possible to leverage the planned suspend capabilities of so called suspend file.

This suspend file is processed during a manager COLD/WARM start or an automation configuration REFRESH. A suspend request is generated for all resources listed in the suspend file, so they are not automated after the load of a new or changed policy.

The automation manager runs in a z/OS address space of its own. Normally, you would add this start command to the COMMNDxx PARMLIB member so that the automation manager is automatically started at IPL time as described in IBM System Automation for z/OS Planning and Installation. Alternatively, you can start it with an MVS start command that calls a module that initializes the automation manager. In the start command you can override several parameters of the start procedure. Most important is the start type that you specify with the TYPE parameter.

## Architecture details



2-10

### Architecture details

This chart covers the manager, agent, and operator roles, the configuration, including data sets, and the communication.

## The automation manager

The automation manager runs in its own z/OS address space. Users communicate with the automation manager through the automation agent. The automation manager communicates with the automation agent on each system through Cross-System Coupling Facility (XCF). It receives updates about the status of resources in its automation model and sends orders to the agent when various conditions within the model are encountered.

The IBM Automation Control automation manager communicates only with the automation agent in the same system whereas the System Automation automation manager communicates with all automation agents in the SAplex. At initialization, the automation manager reads start parameters from its **PARMLIB** data set and the automation policy from the **last-used configuration file** or a new one specified by the PARMLIB or operator.

**OS loggers streams.** These streams are dedicated to System Automation for z/OS use, such as work item history and automation manager detail messages.

## The automation agent

Resource status is maintained internally, with important status values written to the **automation status file**. The automation status file is read at initialization time to help in making restart decisions. Messages (alerts) can be sent to specific operators, called **Notify Operators**, allowing remedial actions to be taken.

When an agent initializes, it connects with its primary automation manager and receives the name of the manager configuration file and a token. The agent uses this information to locate its automation control files in the configuration data set, and to ensure that it is consistent with the manager files. The automation agent and manager communicate through z/OS cross-system coupling facility (XCF). To be able to communicate in certain situations, the automation manager instances and the automation agents belonging to one SAplex must be members of the same XCF group.

## The role of the automation agent

Handle the start, stop, and monitoring of resources on behalf of the manager

- Receive orders from the automation manager
- Performs passive and active monitoring
- Maintain agent status as events and actions occur
- Send status updates to the automation manager
- Determine whether recovery thresholds are met or exceeded
- Maintains its own automation flags

Operator console and interface to automation manager

Automation platform

- Message automation
- Automation interface for client extensions

*Monitoring* in this context refers to automation product processes that check the status of resources and **not** to monitor the resources

### *The role of the automation agent*

The automation agent starts, stops, and monitors automated resources on behalf of the manager.

The automation agent receives orders from the automation manager, performs passive and active monitoring, maintains agent status as events and actions occur, and sends status updates to the automation manager.

The automation agent maintains **thresholds** that can be set to let the operator know if certain errors are occurring infrequently, frequently or have reached a critical level where the recovery process must end to avoid endless loops. This is done by specifying how many times an error must happen in a certain time period for each error situation.

**Agent automation flags** can be set within the automation configuration file and modified dynamically by operator commands. If an event occurs that triggers automation, automation checks the appropriate flag to determine whether automation is currently on.

The automation agent provides the following functions:

- Operator console and interface to automation manager
- Automation platform
- Message automation
- Automation interface for client extensions

## Automation agent overview

The automation agent runs in a NetView address space and includes the following components:

- Automation policy read in from the automation control file
  - Stored in NetView common global variables
- Automation tables with automation table statements
- Optional message revision table to change messages
- REXX and other programs
- Automation operators (autotasks)
- Human operators
- Operator dialogs and commands
- Automation status file
- NetView NETLOG and CANZLOG

### Automation agent overview

The automation agent runs in the NetView address space. It operates on the automation policy read in from the automation control file and stored in **NetView common global (CGLOBAL) variables**.

### NetView for z/OS automation table

The automation control file also can include the NetView for z/OS automation table which is used for message-processing purposes. Messages are generated by the automated resources. Most automation policy definitions use messages to trigger actions.

The messages are assigned to a NetView autotask to distribute workload and to ensure serialization of the messages as they are automated.

Automation table actions can be MVS commands or calling REXX and other programs provided by the agent or by the user.

Defined automation table actions and timer events drive the three major areas of agent automation routines:

- Resource management
- z/OS automation
- Monitoring

All three automation areas set status internally within the agent automation status file. They communicate the status to the SDF component, which is based on the Inform List policy. The resource status is propagated to the automation manager, and the automation status is also kept in NetView common global variables.

The automation agent logs status and events for its resources in an automation status file. The status in the automation status file is used when checking thresholds. During a restart, the automation status file is read and the status is passed to the manager.

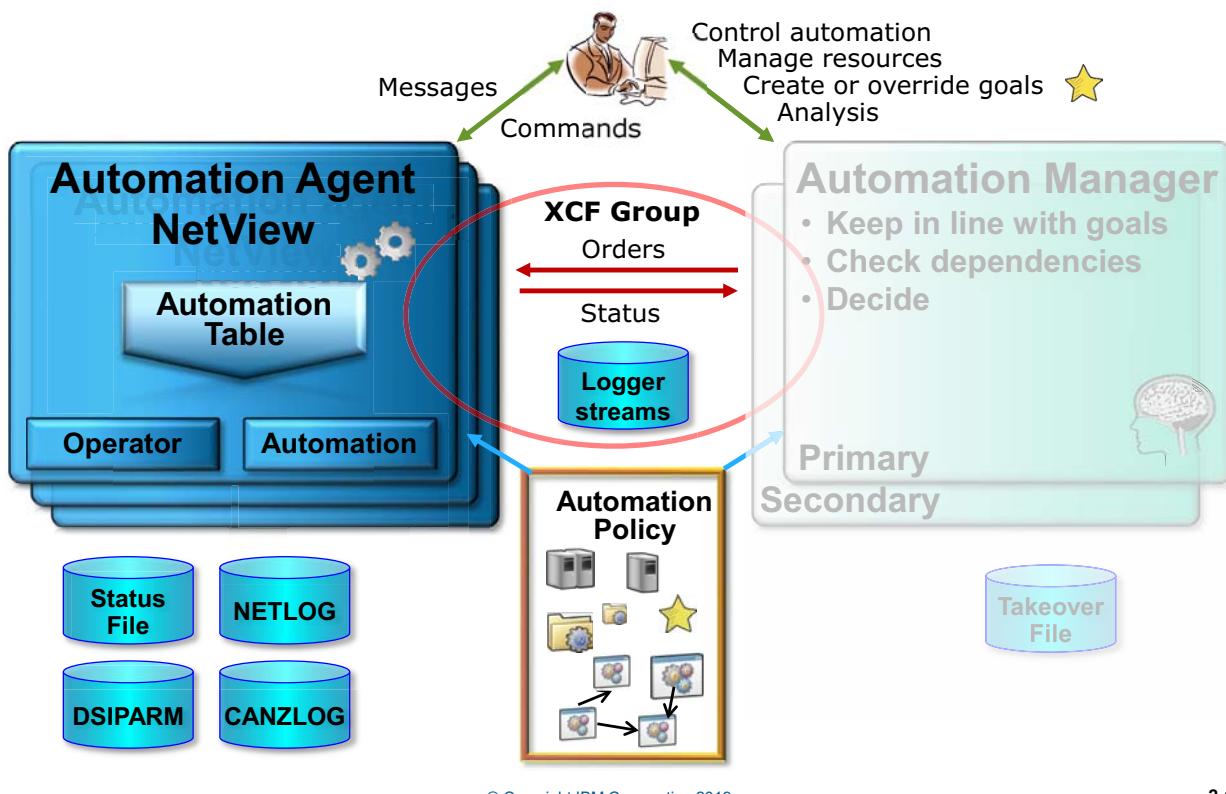
### **Message revision table**

The message revision table allows modification and even deletion of messages.

### **Operator console**

The operator uses NetView as an operations console. The operator receives, views, and deletes messages that are routed from the NetView automation table or generated by automation. From NetView the operator can enter z/OS, NetView and automation agent and automation manager commands, see responses, display automation and resource status in full screen panels and request actions, and browse the NetView NETLOG or CANZLOG for problem determination.

## The automation agent structure: Simplified



© Copyright IBM Corporation 2019

2-13

*The automation agent structure: Simplified*

More features in addition to the already provided automation agent description:

### ***The automation agent is a NetView application***

The automation agent runs in the NetView address space on the system that it is managing. The agent uses many of the standard NetView automation facilities. Think of the agent as an automation application that runs under the control of NetView.

### ***NetView parmlib (DSIPARM)***

The communication data services task (DST) initialization processing reads a NetView parmlib (DSIPARM) member /NGXINIT. The name of the XCF group to which the agent belongs is specified in INGXINIT. With this information available, the agent attempts to contact the primary automation manager. The manager and agent initialization is described in the Operations course.

When the agent initializes, it reads the policy information and sets NetView common global variables for defined automation agent policies.

DSIPARM also contains:

- SDF initialization parameters, trees, and panels
- Automation Network member AOFOPFGW
- NetView Stylesheet
- DSIOPF member for definition of users
- User automation tables

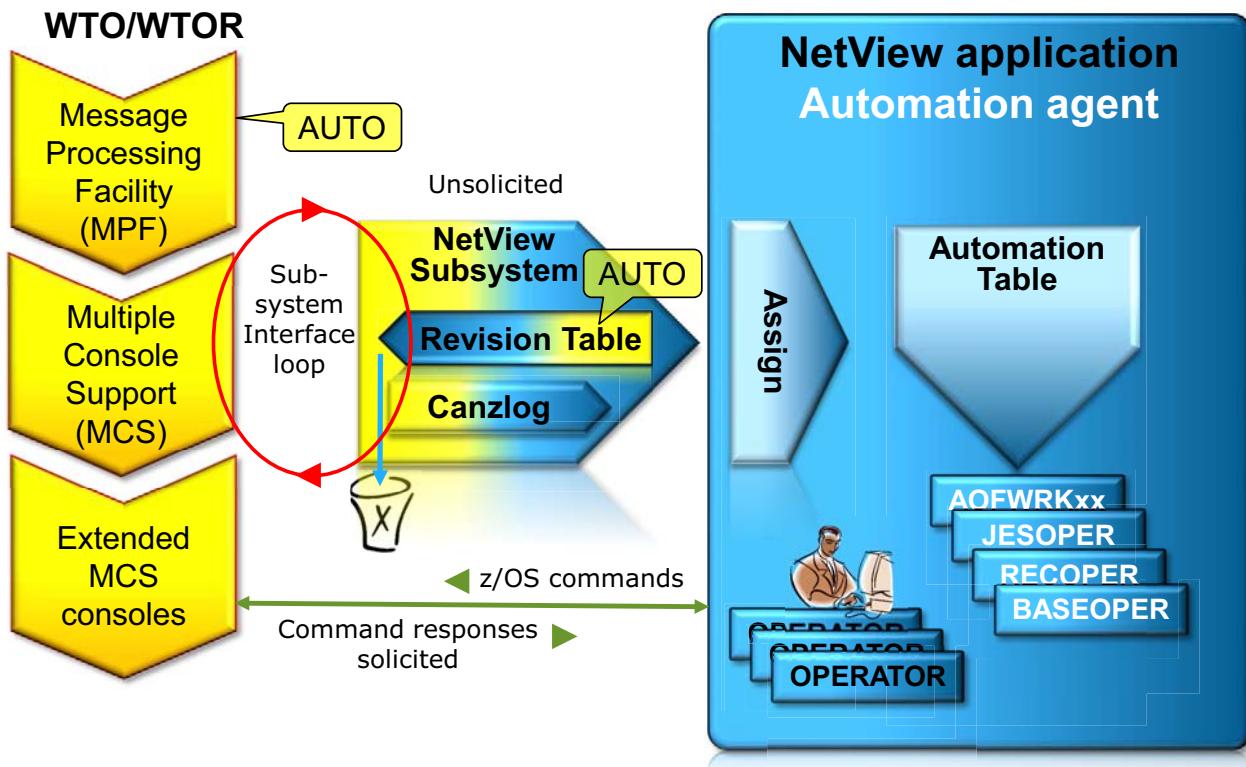
## Logs

All agent components log messages to the standard NetView log (NETLOG).

CANZLOG contains the NETLOG and the syslog, can be filtered, browsed, and searched as well as archived.

The agent also reads information that is placed in OS logger streams. These streams are dedicated to automation use, such as work item history and automation manager detail messages. Status is maintained internally, with important status values written to the automation status file. The automation status file is read at initialization time to help in making restart decisions. Messages (alerts) can be sent to specific operators, called Notify Operators, allowing remedial actions to be taken.

## Message flow



© Copyright IBM Corporation 2019

2-14

### Message flow

As **write-to-operator (WTO)** and **write-to-operator-with reply (WTOR)** messages are generated, they first flow into the **Message Processing Facility (MPF)** of z/OS. The MPF is a means of filtering messages from automation by coding AUTO=NO. An MPF exit routine does processing that is specific to a certain type of message or a particular message ID. After MPF, the WTO flows to IEAVMXIT which can be specified as the general-purpose exit routine that does processing that is common to many messages (WTOs). **Message Flood Automation** processing occurs before a message is placed onto the **Subsystem Interface (SSI)**. Automation products such as NetView, which can obtain messages from the subsystem interface, see messages after Message Flood Automation has seen and potentially modified them. All requests to delete, log, or queue messages are processed after return from the subsystem interface. Therefore, NetView and other automation products that sit on the subsystem interface can see the message and potentially copy or modify the message (possibly overriding Message Flood Automation decisions) before z/OS deletes, logs, or queues the message.

The **message revision table (MRT)** is an optional set of statements that are in DSIPARM (sample member name is CNMSMRT1). The MRT is loaded from the NetView application address space with the REVISE command. The MRT is loaded into storage within the **NetView SSI** address space. The MRT can change or even delete messages. Clients can decide to use MPF, MRT, or both, however, using MRT might override Message Flood Automation decisions. It provides

statistics and usage information and a test mode, and is active even when the NetView program is not. The message revision table is controlled by the NetView system programmer rather than by the z/OS system programmer. The automation product creates an MRT which might be loaded dynamically.

After MPF processing, the original z/OS message is sent to Multiple Console Support (MCS) and then to the Subsystem Interface loop where the NetView SSI listens for *unsolicited messages*. The message can be suppressed or attributes of the message can be changed (color, route codes, descriptor codes, for example). If the message is suppressed with MPF, then it is not displayed on consoles. A message copy can also be sent to NetView for automation by coding AUTO(Y) in the MPF member or REVISE ('Y' automate) in the MRT. Messages that are passed to automation flow into NetView and drive its automation table.

The automation product creates entries in the **NetView automation table** for specific messages. The messages are assigned to a NetView task to distribute workload and to ensure serialization of the messages as they are automated. Based on the type of message, an action is routed to an automation operator. The action can include update of the agent status for an application or z/OS commands.

Operator definitions are customized in the dialogs by the system administrator. Automation operators are described later in this unit.

**Automation operators** or **Autotasks** and human operators can enter z/OS commands by using Extended MCS consoles. Command responses are sent back as *solicited messages*. Autotasks for automation can be started during NetView initialization. The automation table, command lists, command processors, and timer commands can all issue commands under the autotasks. Autotasks can receive messages and present them to the automation table or to installation-exit routines.

Some messages issued by application programs (such as CICS and IMS programs) to their consoles are not available through the subsystem interface or extended multiple console support (EMCS) consoles. To automate responses to such messages, you can use the NetView program's terminal access facility or automation product exits to automate these messages.



## The role of the operator

- Control automation
  - Load a configuration file
  - Manage automation managers
  - Set a runmode
- React on problem statuses, messages, and alerts
- Manage resources
  - Set statuses, for example to return resource to automation
  - Suspend and resume resources
  - Set triggers, events
- Create or override goals
  - Create, delete, or modify requests, override service periods,
  - Set availability targets
- Analysis of
  - Automation decisions
  - Resource policy, statuses, history...
  - Automation log and syslog



HardDown



© Copyright IBM Corporation 2019

2-15

### *The role of the operator*

The operator role is to control automation like loading or refreshing a configuration file, switching roles of automation managers, setting runmodes...

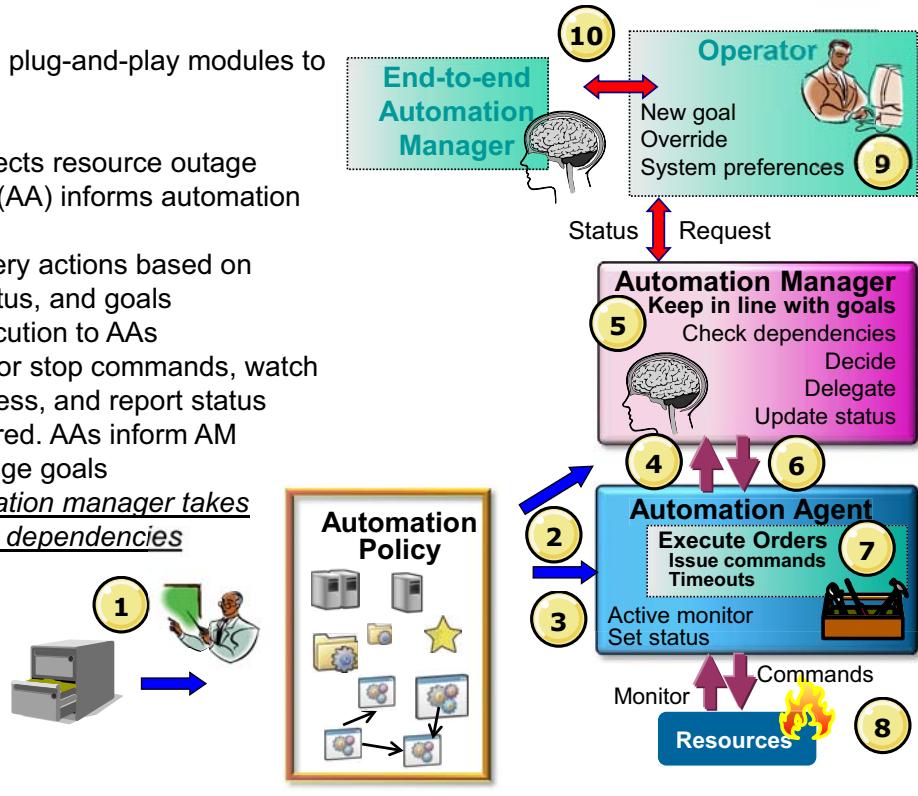
Operators should use their user interface of choice to detect and react on problem statuses, messages, and alerts. This can involve analysis of automation decisions, resource policy, statuses, history, and messages as well as of the automation log and syslog.

For example, when a resource problem is fixed after a harddown status, the operator can set the status to a softdown status to return the resource to automation.

Execution of change requests also can require to suspend or resume resources, to create or override goals using requests or service periods, or group availability targets.

## Automation in action

1. Administrator uses plug-and-play modules to create policy
2. Policy activated
3. Active monitor detects resource outage
4. Automation agent (AA) informs automation manager (AM)
5. AM decides recovery actions based on dependencies, status, and goals
6. AM delegates execution to AAs
7. AAs execute start or stop commands, watch timeouts and success, and report status
8. Resources recovered. AAs inform AM
9. Operator can change goals
10. End-to-end automation manager takes care of end-to-end dependencies



© Copyright IBM Corporation 2019

2-16

### Automation in action

This diagram shows the automation product in action.

1. The administrator uses plug-and-play modules to create the automation policy. The slide also covers the following areas:
  - How automation detects a resource outage
  - The roles and actions of the automation agent and the automation manager
  - The operator role
  - *The role of the end-to-end automation manager (SA z/OS only) provided by SA z/OS or IBM Tivoli System Automation Application Manager*

On the slide, you see the automation agent and automation manager actions when a resource must be recovered. The mission of the automation manager is to keep the system in line with the goals.

The automation agent notifies the automation manager about a status change (3,4). Then, the automation manager uses the configuration, goals, relationships, locations, and resource status to decide the recovery actions (5). Relationships can affect other resources across the SApex.

The automation manager delegates the actual execution that resolves the situation to the affected automation agent or agents.

The automation agent monitors, starts, and stops a resource. It also checks thresholds and timeouts. It executes the orders by issuing the commands that are defined in the policy.

As shown here, the automation product has an automation manager. It consists of coordinating, decision-making, and controlling elements which are grouped into a single z/OS address space. The automation manager also has a model of the automated resources.

The automation manager:

- Receives input from various sources like events and operators
- Analyzes and correlates this input as it relates to the status of resources in the automation model
- Decides what recovery actions must be taken in what order, possibly including dependent resources
- Sends requests for actions to the automation agent, which is the element that observes, reacts, and takes action. Actions to recover from failure situations include:
  - Restarting resources in place, or, if necessary,
  - Failing over to a backup system (SA z/OS only).

An agent must be running on every system to be automated. The automation manager communicates with agents through Cross-System Coupling Facility (XCF). In addition, if the primary automation manager fails, a secondary manager can take over. No loss of automation occurs, since the automation state is saved in a takeover file.

*IBM Tivoli System Automation Application Manager and SA z/OS provide cross cluster and cross platform automation. They can create requests against the SA z/OS automation manager and provide a DASH-based user interface.*

# Lesson 2 Goal driven automation

## Lesson 2: goal driven automation

- What is goal driven automation
- Goal driven automation scenario
- Defining and overriding goals
- From requests to votes
- Request priorities
- Request persistency
- The request process
- Suspending resources

© Copyright IBM Corporation 2019

2-17

### *Goal driven automation*

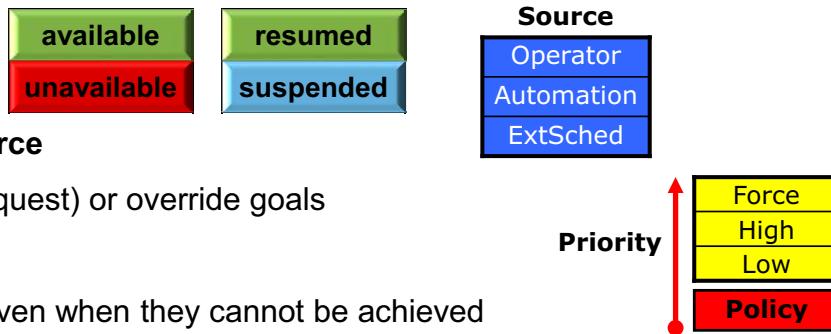
This lesson explains the goal driven automation concepts. What goals are, where they can be defined, where they come from, what priority they have, when they expire and how they can be overridden using requests.

You learn that:

- Goals can be conflicting
- Goals sometimes can not be achieved due to automation inhibitors
- Goal driven automation can be suspended

# What is goal driven automation

- Availability goal
- Suspended goal
- Goals come from a source
- Operator can create (request) or override goals
- Goals have a priority
- Goals are persistent, even when they cannot be achieved
  - Unlike commands that are entered and forgotten
  - Cancel requests when you no longer need them
- Automation manager tries to achieve goals while taking care of dependencies
- Easier and safer operations with a single action at application group level
  - Goals can be propagated to group members and along active dependencies as votes putting all required resources in the desired status
  - Conflicting votes are resolved by priorities and request sources
- Resource status adjusted to goals, dependencies, configuration, and status



© Copyright IBM Corporation 2019

2-18

What is goal driven automation

What is goal driven automation?

- Goal driven automation is very different from the command driven automation of other products.
- Goal driven automation can keep applications in line with business goals, dependencies, configuration, and status.
- Goals can be defined in the policy, overridden or created by the operator.
- Goals are persistent, can be propagated.

The administrator defines the goals for the application according to business requirements in the policy. Goals are simply either *available* (up) or *unavailable* (down). Application group goals in addition control, which and how many members are started.

The *suspend goals* are handled completely separately from the desired status goals.

The operator or other “sources”, like a scheduler or automation script, a change or override goals and create start or stop goals by entering *requests*. Operator requests can have a higher priority than the Desired Available setting in the policy.

Goals are persistent. They are not like commands that are run and forgotten. You must cancel requests when you no longer need them.

Operations at the application group level means starting an application having several components with one request.

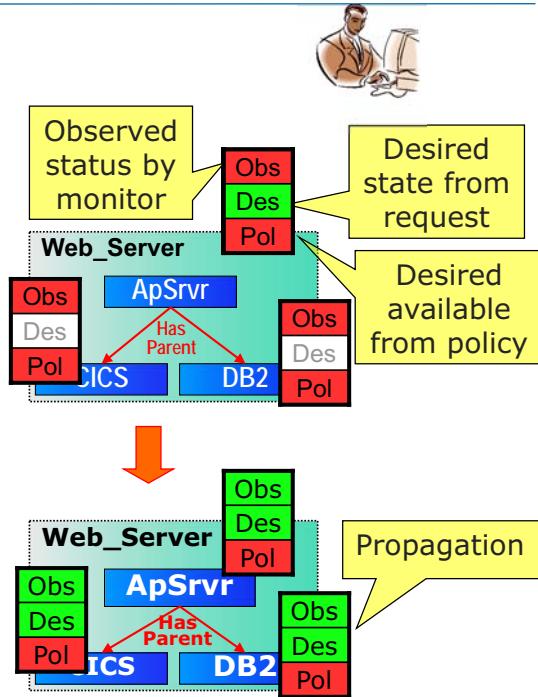
- *The request is propagated as votes to group members and along **active dependencies**.*
- When multiple dependencies exist, conflicting goals can be the result. Conflicting goals are resolved. Requests have a *priority* and have a *source*. Automation uses priorities and sources when determining the winning vote. Sources might be, for example, automation, operator, or external.

This process can prevent operations errors, like shutting down a resource that is needed by another application. Operators can use goal driven automation to simplify operations by entering or removing a request at the application level. They can use the automation product to manage dependencies and goals that are affected and to resolve conflicting goals.

A single policy can describe all valid configurations. If dependencies and sysplex groups are setup correctly, there is no need to define backup configurations to take care of a resource or system outage. The automation product automatically starts or stops resources to achieve business goals while taking care of dependencies, configuration, and status.

## Goal driven automation scenario

- Policy defines goal for group Web\_Server
  - Desired Available status is unavailable
- Operator creates start request
  - Beats Desired Available status
- Goal is propagated to group members and along dependencies
- Automation starts required CICS and DB2 first
- ApSrvr started after CICS and DB2 are up



© Copyright IBM Corporation 2019

2-19

### Goal driven automation scenario

Operations at the application level means starting an application having several components with one request. The Desired Available setting in the policy is unavailable (specified as OnDemand). Initially, application group Web\_Server is not started.

In this scenario, the operator enters a start request against application group Web\_Server. An operator request has a higher priority than the Desired Available setting in the policy. The desired state changes to available if no other requests have higher priority.

The request is propagated as votes to group members and along the dependency tree.

Here, you see that the state changes when an operator starts a resource. The operator changes the desired state, which has a higher priority than the Desired Available setting in the policy.

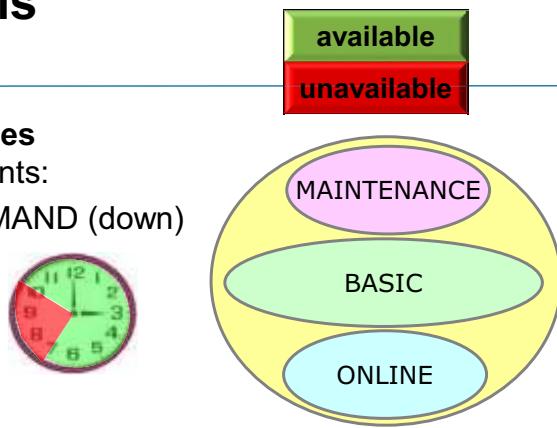
Because of the **HasParent** relationship, resource ApSrvr can only be started when resources CICS and DB2 are available.

The automation manager propagates the goal to the resources that ApSrvr depends on. The result is a start of resources CICS and DB2. After resources CICS and DB2 are available, resource ApSrvr is started also.

Operators must cancel requests when they no longer need them. So, to stop application group Web\_Server, just cancel the start request.

# Defining and overriding goals

- Administrator defines **goals** and **dependencies** in the policy according to business requirements:
  - Desired Available:** ALWAYS (up), ONDEMAND (down)
  - Runmodes**
  - Service periods**
  - Application groups**
    - Goal propagation to members as votes
    - Member with highest **preference value** are selected
    - Availability** and **satisfactory targets**
- Easy, exception-oriented operation**
  - Operator can "overrule" the policy goals by (service period) overrides or requests
- Responsibility moves from operator to automation administrator
  - However, operator is responsible for force requests and request removal
- Problem states stop goal driven automation and require manual intervention



© Copyright IBM Corporation 2019

2-20

## Defining and overriding goals

Administrator defines the **goals** for the application according to business requirements in the policy:

- The **desired state** is simply either available (up) or unavailable (down).
- Runmodes** can be activated which create unavailability goals for all resources NOT belonging to a runmode. **Runmodes** can be used to partially start or stop the system or to start backup applications that are normally started on another system. Examples on the slide show MAINTENANCE, BASIC, and ONLINE.
- Availability schedules**, called service periods, can be linked to resources which creates a goal.
- Application groups:
  - Goals are propagated to group members as votes
  - The member or members with the highest preference value are selected
  - Availability and satisfactory targets** determine how many members of a server group are started

The automation manager tries to keep the system in line with goals. Problem states, switched off automation flags, and any policy to not start a resource stop goal driven automation and require manual intervention.

Easy, exception-oriented operation:

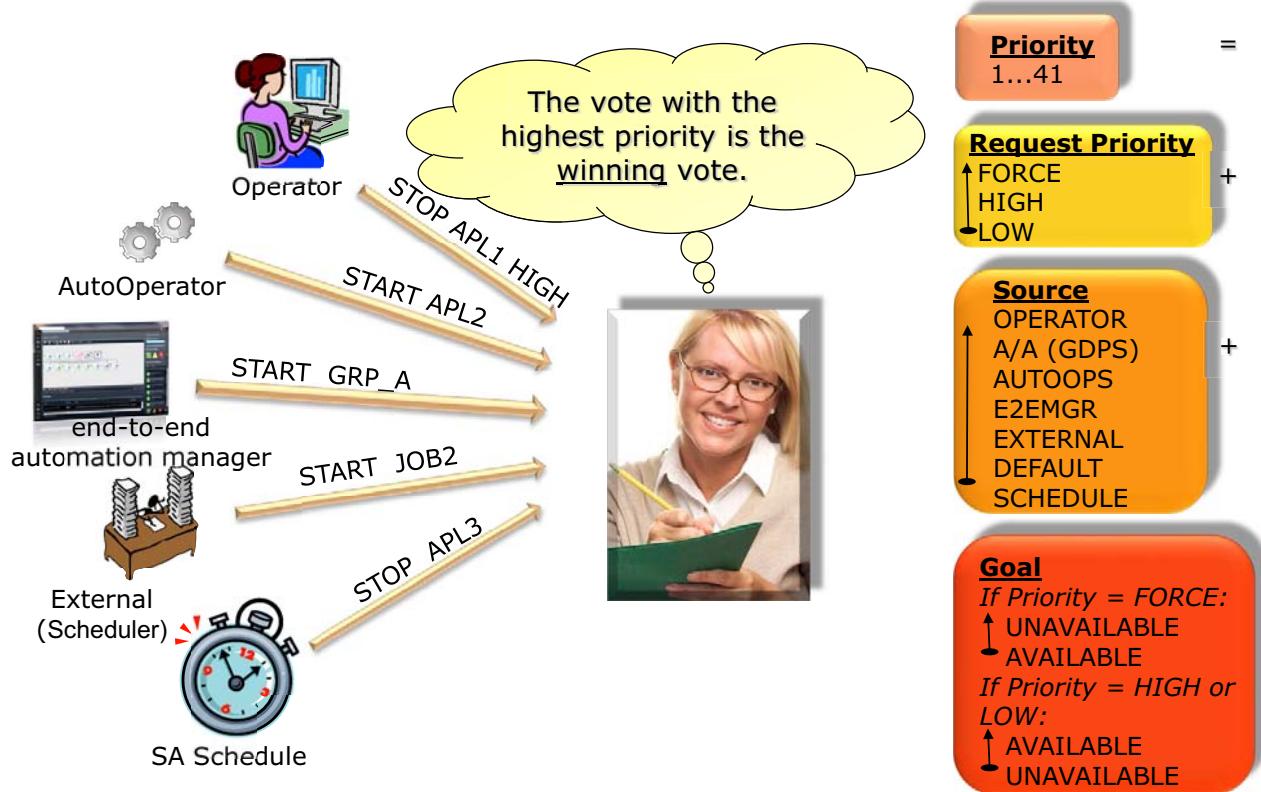
Operators can overrule the policy goals by service period overrides, changing group preferences, *excluding systems for groups (SA z/OS only)*, or start/stop requests.

Start/stop requests allow to complete the following tasks:

- Set the priority, which helps to become the *winning vote*.
- Define the scope of the request, like whether the dependent resources should be affected too.
- Specify expiration to, for example, automatically remove a request after an IPL.
- Ignore automation flag settings, trigger states, and the state of dependencies. However, this should be used only in emergencies.

The responsibility moves from operation to automation administrator, however, operation should use the highest priority only in emergencies, and also should remove requests when they are no longer required.

## Request priority scheme



© Copyright IBM Corporation 2019

2-21

### Request priority scheme

The request priority indicates three specifications in the request to the automation manager:

- Where the request originates
- What the external priority of the request is
- What action is requested, start or stop

The vote priority, which can be confused with the request priority, and which is a number, is set according to the source, priority and action of the initial request from which the vote originated. The manager generates one or more votes for each request.

When there is more than one vote for a resource, the highest priority vote wins. The winning vote determines the state that the automation manager applies as the desired status for that resource. If the winning vote is a MakeAvailable vote, the desired status is set to AVAILABLE and the resource is started. If the winning vote is a MakeUnavailable vote, the desired status is set to UNAVAILABLE and the resource is stopped.

If the current observed status of an application is inconsistent with its desired status, the automation manager can send orders to the agent to start or stop the application. If there are two votes with identical priority, one of them is arbitrarily chosen as the winning vote. The decision does not affect

any action because identical priority votes are requesting the same action. It does affect what the operator sees in INGVOTE displays.

The sources in descending priority order are:

- OPERATOR
- A/A (GDPS Active/Active )
- AUTOOPS
- E2EMGR
- EXTERNAL
- DEFAULT
- SCHEDULE

The source is often described as *originator* and can include the operator or task name that created the request.

For all priorities except FORCE, the request or goal to make a resource available has a higher priority than to make it unavailable. However, a FORCE UnAvailable request that is issued by an operator has the highest priority.

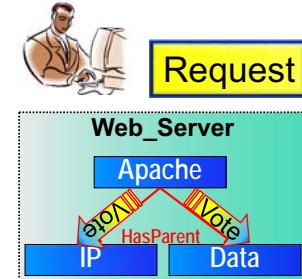
This is really the only method to beat automation attempts to make a resource available if for any reason you want to have this resource down.

If you add all possible priorities from sources, actions and priorities you can count 41 priority levels.

## The path from request to order



- Request entered with INGREQ command with type, scope, priority, source, override, expiration, comment...
- Request is propagated as votes
  - To group members
  - To supporting resources along active dependencies
  - To dependent resources or children for scope=all
- The winning vote changes the resource's desired status
- If desired status is not the same as the observed status: Order is sent when no inhibitors are present and resource is not suspended
- Display votes with the INGVOTE command
  - Display and cancel requests



© Copyright IBM Corporation 2019

2-22

### The path from request to order

Requests are entered with type, scope, priority, source, override, expiration, comment...

Requests are propagated as votes:

- To group members
- To supporting resources along active dependencies
- To dependent resources, often called *children*, for scope=*all*

The winning vote changes the resource's desired status.

If desired status is not the same as the observed status an order is sent when no inhibitors are present and the resource is not suspended.

Use the INGVOTE command to display votes and cancel requests. Votes can not be canceled directly.

## Persistence rules for requests

- **Requests remain active** until removed (canceled or expired), replaced, or automatically removed at IPL
- **One request from one source for one resource at a time**
  - New request replaces previous request independent from priority
  - For example, a request from operator OPER1 can replace a request from OPER2 for the same resource
- The default (implied) is a MakeAvailable vote from the automation manager
  - Can be changed by setting the Desired Available in the policy to OnDemand or ASIS

© Copyright IBM Corporation 2019

2-23

### Persistence rules for requests

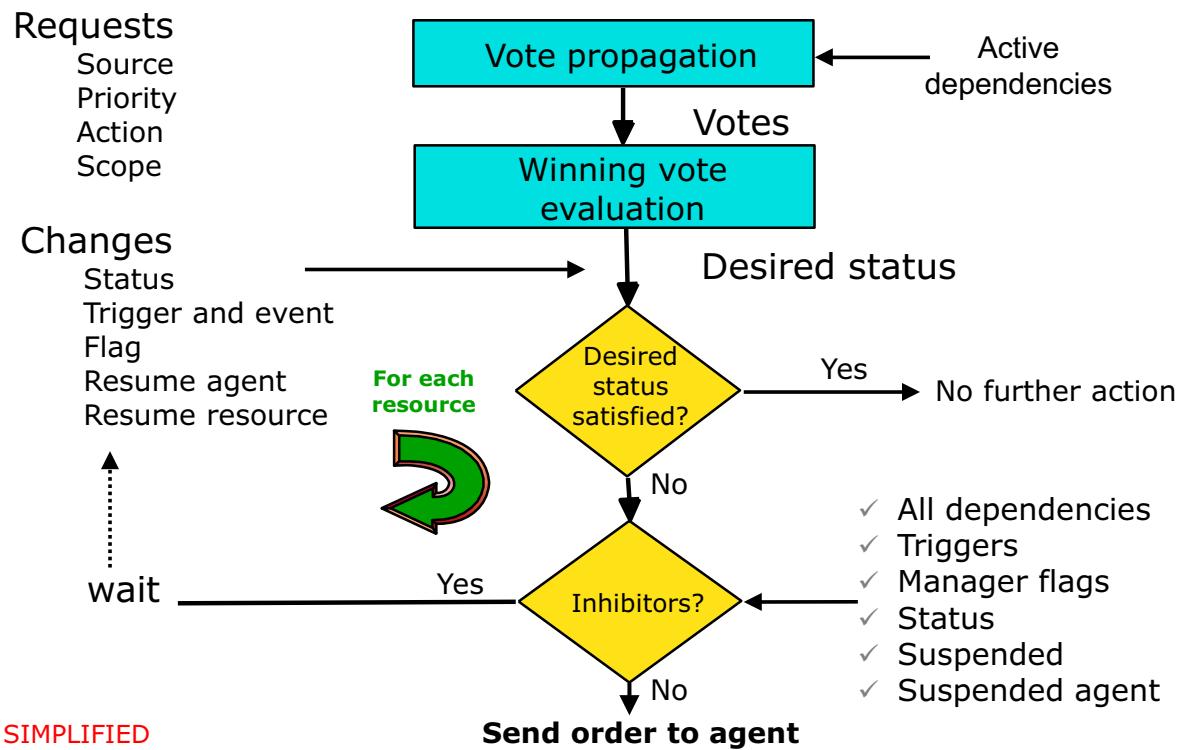
Requests are persistent and can even survive IPLs. However, a WARM or COLD start of the automation manager removes all requests and resulting votes. For each targeted resource, the automation manager keeps only one request from each source. Only the latest request is kept.



**Note:** Using the Advanced Automation Option (AAO) INGREQ\_Originator, you can have multiple requests from operators. For more information, see *System Automation for z/OS Customization and Programming Guide*.

When the term *source* is used, it has a specific meaning for operators and auto operators. Operator means any human NetView operator. A request from one operator replaces an existing request from a different operator. The operators might even be logged on to different systems within the SAplex. Similarly, for auto operators (*autotask*), a request from one autotask replaces an existing request from another autotask. A request can also be explicitly canceled, with one of the following commands: INGVOTE, or INGREQ REQ=CANCEL.

## The request process



© Copyright IBM Corporation 2019

2-24

### The request process

Requests are translated into votes which are persistent. Each request, because of its scope and active dependencies, creates internal requests called votes. A request can remain active, but never satisfied, because other higher priority requests exist or because passive dependencies are not fulfilled. The process of creating votes from the basic requests is called vote propagation.

The automation product provides *goal driven* automation. The automation manager is critical in the process. Each vote represents the *desired state* for the resource to which it applies. As the automation manager processes votes, it works towards reaching the goal which is the desired state. When there are several votes for a single resource, the manager chooses a winner. The desired status of the winning vote becomes the desired status of the resource.

If the actual status of the resource and its desired status do not match, the automation manager attempts to send an order to the agent to start or stop the resource. Before the manager sends an order, a number of possible *inhibitors* are checked. These inhibitors can delay the sending of the order until certain conditions are satisfied, or overridden. When all inhibitors are satisfied, the manager sends the order to the agent.



**Note:** This diagram is used several times in this unit to explain the automation manager and the request process.

# Suspending resources

- Suspended resources
  - Are not automated
  - Can not be started or stopped by automation
  - Are still monitored, but status values are only active or inactive status values for agent status and observed status
  - Job Log Monitoring is stopped for suspended resources
- Resource can be suspended by:
  - Using the INGSUSPD command which produces a **suspend request**
  - Entering resource in **suspend file** before warm or cold start or refresh
- Suspend requests
  - One per resource for all sources, one priority, scope can be specified
  - Resource must complete its start or stop before suspend request succeeds
- Requests using INGREQ:
  - OVERRIDE = SUS is possible to ignore the suspension
  - OVERRIDE = SUS is used by default for a system shutdown
- RESUME automation restores the normal behavior (**except Job Log Monitoring**)

## Suspending resources

Normally, automation automates resources that are defined in the policy based on their desired status goals. In some situations, for instance, when maintenance activities require manual startup or shutdown of a resource, the involvement of automation is not wanted. In fact, it would be even counterproductive if automation "corrected" that. In such situations, automation allows the operator or administrator to suspend a resource. While it is suspended, automation does not attempt to start or stop this resource. Similarly, it does not react on messages that would normally trigger status commands or other commands that are defined in the policy for a message. Most importantly, it does not alert the operations team by exposing an unusual Automation Agent or observed status on any status display (for instance INGLIST or SDF), which operators are normally sensitive to.

- The resources are still monitored, but status values are only active or inactive status values for agent status and observed status
- Job Log Monitoring is stopped for suspended resources

## Resource can be suspended by:

- Using the INGSUSPD command which produces a suspend request. There is only one suspend request per resource for all sources, only one priority. The scope of the suspend

request can be specified. For a SCOPE=ALL request, suspend votes are propagated along the dependency chain to all dependents

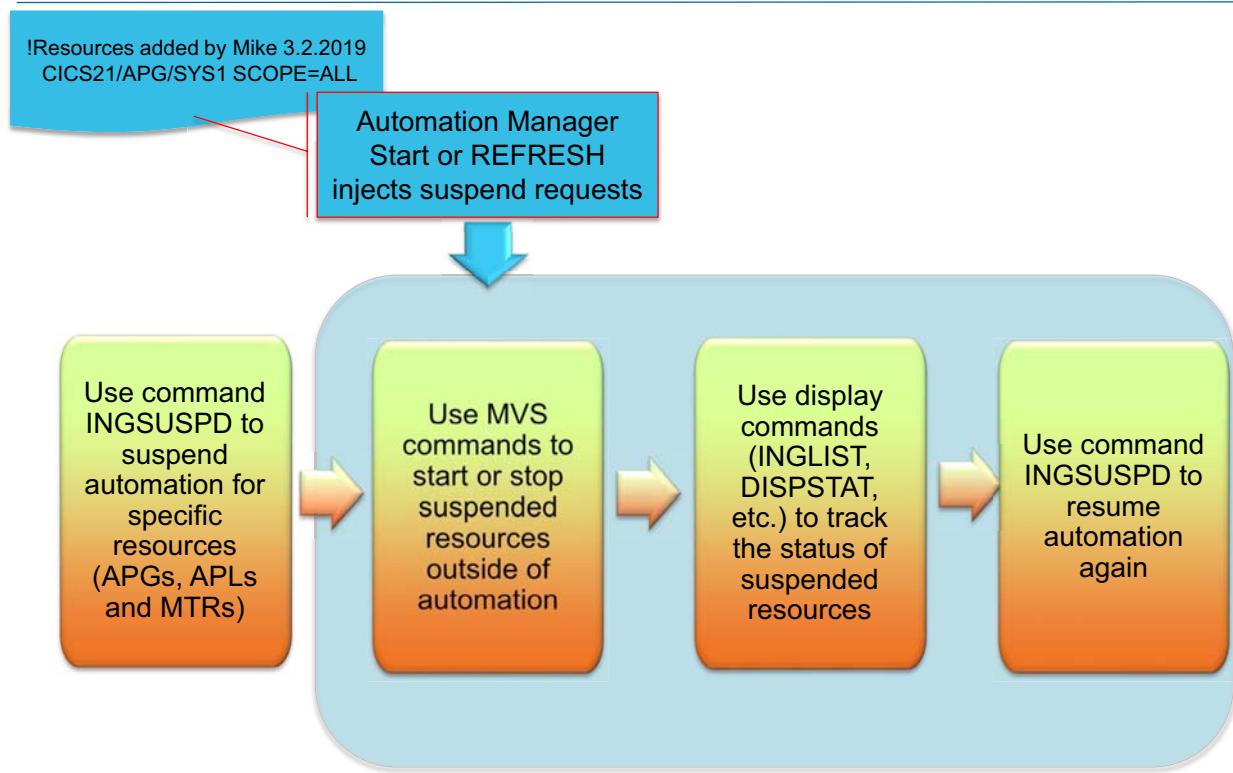
- Entering resource in suspend file before warm or cold start or refresh

Requests using INGREQ:

- OVERRIDE = SUS is possible to ignore the suspension
- OVERRIDE = SUS is used by default for a system shutdown

RESUME automation, which can be done with INGSUSPD or by killing the suspend vote in INGVOTE, restores the normal behavior (except Job Log Monitoring).

## Suspending resources workflow



© Copyright IBM Corporation 2019

2-26

### Suspending resources workflow

- Mike prepares suspend file:
  - You can add descriptive information in the file with "!" at the beginning of each line
  - Beginning in column 2 he defines resources to be suspended in the format name[/type[/system]]
  - He can specify optionally the scope and remove parameters
  - An automation manager Start or REFRESH injects suspend requests
- Alternatively
  - Use the INGSUSPD command which produces suspend requests
- Use MVS commands to start or stop suspended resources outside of automation
- Use display commands (INGLIST, DISPSTAT, etc.) to track the status of suspended resources
- Use command INGSUSPD to resume automation again

## Suspended resources

If suspend request succeeds and automation status is idle:

- Automation manager suspend flag is set to suspended
- Agent automation flag turned off

Implications:

- INGSET/INGAUTO can not be used for suspended resources
  - Except to set automation status to IDLE
- Suspended group member behaves like a passive member
  - It can not be selected by the group
- Suspended group behaves like a passive group
  - No votes are generated
  - No members can be selected
- A suspended MTR does not monitor anymore (Timer is cancelled)
- Job Log Monitoring is stopped for suspended resources
- Still performed: OUTREP, Status Observer, special resources, message capturing, threshold handling, WLM resources management, IPL complete notification

### Suspended resources

If suspend request succeeds and automation status is idle:

- Automation manager suspend flag is set to suspended
- Agent automation flag turned off. To differentiate between the turned off automation agent flag or a suspended APL, you can refer to the task global variables SUB\*SUSPEND provided with AOCQRY. All automation flags (Automation, Initstart, Start, Recovery, Terminate, and Restart) are set to 'S' and that setting is also propagated down to minor resources, if applicable. While being suspended, DISPFLGS and DISPSTAT will not allow you to change the status of any of these automation flags. You must first resume the resource before you can change the status of a flag.

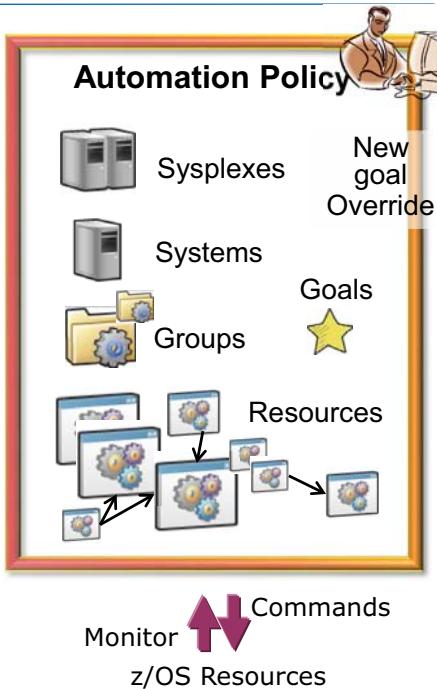
Implications:

- INGSET/INGAUTO can not be used for suspended resources
  - Except to set automation status to IDLE
- Suspended group member behaves like a passive member
  - It can not be selected by the group
- Suspended group behaves like a passive group
  - No votes are generated
  - No members can be selected
- A suspended MTR does not monitor anymore (Timer is canceled)
- Job Log Monitoring is stopped for suspended resources. To enable Job Log Monitoring again, you must do it manually using INGJLM, or make sure that the resource is first resumed and a start request or vote exists for automation to start it.
- The following capabilities stay in place, even while a resource is suspended:
  - OUTREP processing takes place to save the reply ID, if a WTOR was received that is normally handled for the resource
  - Exit routines registered at the Status Observer by a user script or registered internally by automation are called when the resource changes
  - The status of a Workload Scheduler special resource follows any status change of the resource
  - Message capturing, threshold handling, and WLM resources management are done by default
  - The resource is not exempted from IPL-complete monitoring

# Lesson 3 Policy-based automation

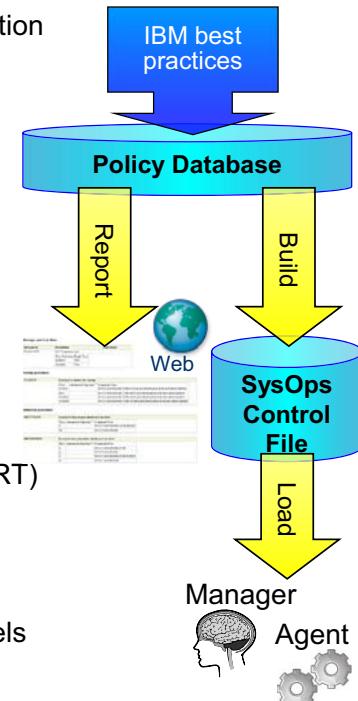
## Lesson 3: Policy-based automation

- Automation policy features
- Resources
- Automation policy for applications
- Starting applications
- Stopping applications



# Automation policy features

- Resource models for: application, system, group, monitor, automation operators, user, and so on
- IBM best practices policy
  - Applications and application classes covering all aspects from start/stop to messages, groups, and dependencies
  - Easy to use and serviced by IBM
- Enterprise-wide shareable and cloneable automation policy
  - Define once, reuse identically or cloned
  - Supports z/OS system symbols and special automation symbols
  - Automation variables like resource or parent name
- Hierarchical application classes make policy definition easier
- Easy one stop message automation
- Link message to resource status
- Tailored NetView automation (AT) and message revision table (MRT)
  - Creates MPF member
  - Full flexibility to control AT, MRT, and MPF artifacts out of the policy
    - Modifications of what is generated out-of-the-box
    - User-specific additions with full syntax checking
- Defaults at sysplex, system, MVS component, and application levels
- Documentation in web and flat file reports
- Multiple users can browse or edit policy
- Activity log for PDB changes



© Copyright IBM Corporation 2019

2-29

## Automation policy features

On this slide are the key automation policy features in the automation product.

- Policies are maintained with ISPF-based customization dialogs that access **policy databases** (PDB).
- The policy supports resource models for application, system, group, monitor, automation operators, and user specified models.
- Your first step in defining automation policy is to **create a policy database** using the sample policies as a model.
- A flat file or HTML report function makes documentation easy.
- The PDB cannot be used by the run time environment directly, a so-called build step is required. The build generates the Automation Configuration File, which is used by the automation agent and the automation manager. It includes a tailored NetView automation table and a tailored MPF file. So the policy is a single point of definition for message automation.

The **Plug and Play automation using best practices** contains easy to use policy that is serviced by IBM containing applications and application classes covering all aspects from start/stop to

messages, groups, and dependencies. It is highly recommended to start with the best practices policy and to compare it from time to time whenever IBM ships updates.

The automation product has an **enterprise-wide shareable and clone-able automation policy**. Define the policy once and reuse the policy identically or cloned. The recommendation is to have one policy database for your entire z/OS enterprise. This allows you to gain maximum advantage by defining the policy once and sharing and reusing it on other systems.

Cloning supports z/OS system symbols and special automation symbols.

You can assign automation symbols to each system named &AOCCLONE, Valid characters are A–Z, a–z, 0–9, @ (X'7C'), # (X'7B'), \$ (X'5B'), ? (X'6F'), \_(X'6D'), ! (X'5A'), and ¢ (X'4A').

Additionally, you may specify the tilde (~) to get the 0th (unnumbered) AOCCLONE value substituted. The entered data is not automatically translated to upper case. When you use an automation symbol in a job name, each system can run with a unique job name while the application or subsystem shares a single automation policy across systems. The substitution can occur anywhere within the job name. Any of the available automation symbols can be used. You can specify more than one automation symbol.

Hierarchical application classes make policy definition easier. Applications can inherit policy from a hierarchy of application classes. This allows you to change policy for all linked applications with a single change at the class level.

Automation policy supports also using variables like resource, jobname, parent name, and many more. This allows reusing policy or automation procedures without changes even when parent or jobnames are different.

The policy allows easy one stop message automation using just the policy:

- Automated response using command or reply
- Link message to resource status or health status
- Message filtering with generated message revision table (MRT) or MPF
- Tailored NetView automation table (AT)

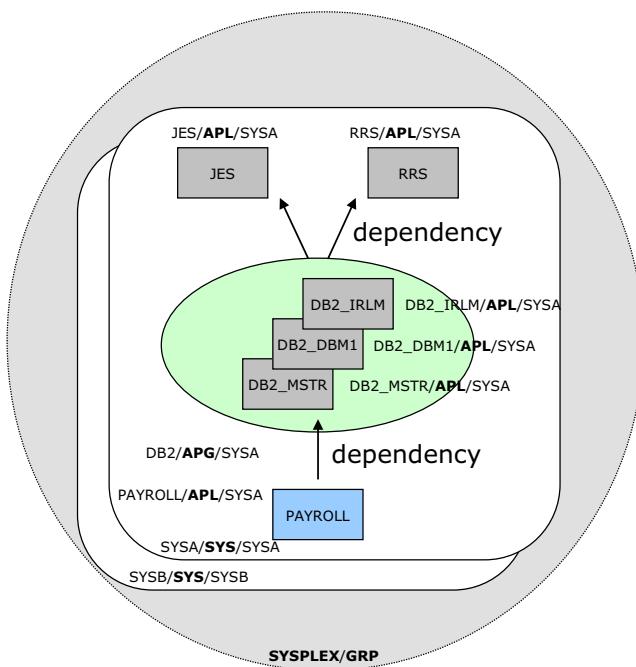
You have full flexibility to control AT, MRT, and MPF artifacts out of the policy to modify what is generated out-of-the-box including full AT syntax checking.



**Note:** It is highly recommended to use the generated message revision table and NetView automation table by adding to your CNMSTGEN: COMMON.AOFSMARTMAT=3

Definition of sysplex, system, MVS, and application defaults can reduce policy definitions.

# Automation resources



An automation resource represents any instance in a z/OS system that can be monitored and automated

Typical instances are:

- Applications to automate started tasks, USS processes, CICS regions, and many more
- Groups of applications to manage their members as one entity or to realize cross-system failover capabilities

Additionally, special resources exist such as

- z/OS systems to manage system add/leave
- Groups of systems to manage resources in a SAplex
- Monitor resources to monitor the health of applications, the system, or any other object in a z/OS system

## Automation resources

System Automation resources are, as explained earlier, automated items that are defined in the policy database. The automation agents start, stop, and monitor automated resources.

Typical instances are provided here:

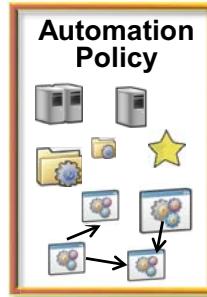
- Applications to automate started tasks, USS processes, CICS regions, and many more
- Groups of applications to manage their members as one entity or to realize cross-system failover capabilities

Additionally, special resources exist such as:

- z/OS systems to manage system add/leave
- Groups of systems to manage resources in a SAplex
- Monitor resources to monitor the health of applications, the system, or any other object in a z/OS system

# Automation policy for applications (APL)

- APL is a z/OS subsystem, **started task**, batch job, USS, non-MVS, or reference resource
- Special support for JES, CICS, DB2, IMS, USS, and IBM Workload Scheduler
- **Transient** job type for applications that terminate themselves
- **Desired available status**, **runtokens**, startup, and restart options
- **Monitor** command and interval
- **Relationships** for start and stop.
  - Also for start and stop preparation, force down, and **external**
- User definable **start types** and parameters
  - Automation can be start type sensitive
- Three **stop types**: normal, immediate, and force
- Start or stop can be paced and with multiple commands and passes
- Phases before and after start or stop (PRESTART, POSTSTART...; SHUTINIT, SHUTFINAL)
- Startup and shutdown **triggers**
- Exceeded critical error **threshold** can stop recovery
- **Links to application groups**, service period, pacing gate, **class**, and trigger
- **Messages** to be automated or captured also from JES job log
- Many more definitions for JES, WLM, ARM, automation symbols and flags, minor resources, timeouts, owner, info link, notification, description, name, jobname, category, subcategory...



## Automation policy for applications

An application is a **z/OS subsystem, started task, batch job, USS, non-MVS or reference** resource.

Special support exists for JES, CICS, DB2, IMS, USS, and IBM Workload Scheduler for z/OS.

A **transient job** type is for applications that terminate themselves and should not be restarted.

**Desired available status** and various startup and restart options can be specified.

The specified **monitor command** is executed each interval to determine the status of the application. Clients can use their own monitoring routine or use one of the predefined monitoring routines including the built-in ASID scan monitor that reduces CPU overhead for monitoring.

An application can be a target or source of **relationships**. Relationships define the dependencies for start or stop, for start or stop preparation, for forcing down and whether the start or stop is external.

User definable **start types** and **startup parameters** can be defined. The automation can be start type sensitive.

- Three **stop types** are supported: normal, immediate, and force.
- Start or stop can be with multiple commands and stages including init stages.
- External start or stop is honored by automation.

Startup and shutdown **triggers** can be defined.

Exceeded critical error **threshold** can stop recovery.

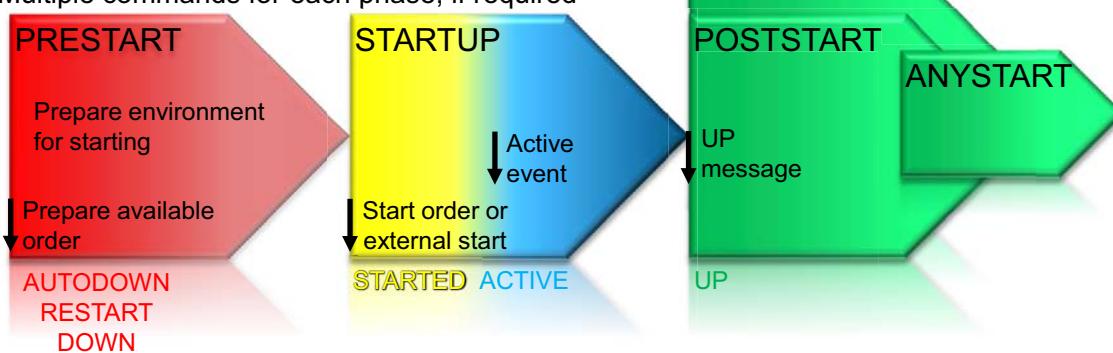
An application can be linked to **application groups**, **service period**, **pacing gate**, **class**, and **trigger**.

**Messages** to be automated or captured can be specified. Messages are typically a WTO or WTOR but can be also from JES job logs. Actions include setting the status, commands, responding to replies, actions dependant on certain codes or text within a message, Message Automation Definitions and User Data Processing.

Many more definitions are available for JES, WLM, ARM, automation symbols and flags, minor resources, timeouts, owner, info link, notification, description, name, jobname, category, subcategory....

# Starting applications

- A start has three major phases:
  - PRESTART: Prepare environment for starting, if required
  - STARTUP: The actual start commands
  - Optionally, after startup is completed:
    - POSTSTART: The actions after the application is UP
    - REFRESHSTART: After a NetView recycle or INGAMS refresh only
    - ANYSTART: Issued after POSTSTART and after REFRESHSTART
- Different start types for each phase are possible, such as COLD, RECOVERY, and so on
- Multiple commands for each phase, if required



© Copyright IBM Corporation 2019

2-32

## Starting applications

Within the automation product, application startup processing occurs in three phases: **PRESTART**, **STARTUP**, and **POSTSTART**. Start types can be defined for each of the three phases. The start type can be an installation-specified value such as COLD for an IPL or RECOVERY to automate the restart of a failed application. The default start type is NORM. Commands in each phase can be specific for a start type. If the desired goal of the resource is MakeAvailable, then the prestart commands for the dependent resources are issued as soon as the specified condition is satisfied for the supporting resource.

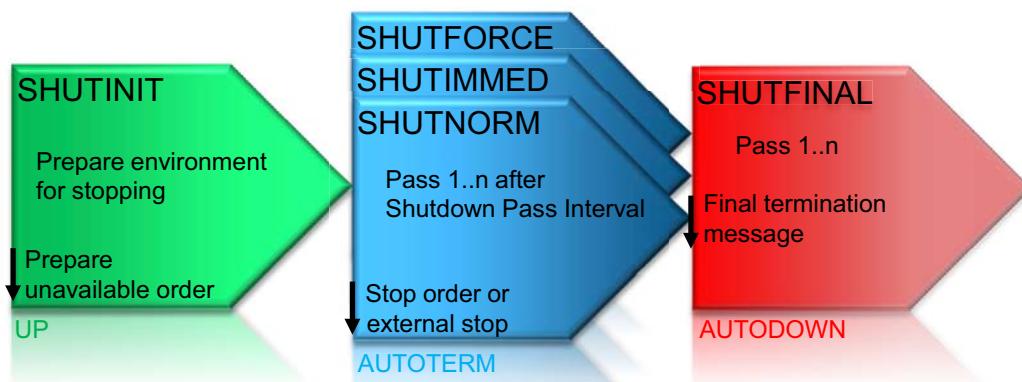
The phases, types, and commands are defined in the customization dialogs. Return code zero can be specified as a condition to indicate the success of a prestart command. A failed prestart command terminates the desired state request actions. Prestart actions are started when an agent receives a PrepareAvailable order. Prestart or poststart commands do not change agent status.

**Refreshstart**, formerly known as ACORESTART commands and defined under pseudo message ACORESTART in MESSAGES/USER DATA policy. Similar for **ANYSTART** which was formerly defined as INGTIMER pseudo message ID.

REFRESHSTART and ANYSTART commands are issued regardless of any automation flag specification in case of NetView recycle or INGAMS REFRESH processing. During the start-up process of an application the ANYSTART commands are only issued if the START flag is turned on.

# Stopping applications

- A stop has three major phases:
  - SHUTINIT: Prepare environment for stopping , if required
  - SHUTxxxx: The actual stop commands
    - NORM: Normal shutdown
    - IMMED: Immediate shutdown
    - FORCE: Force a subsystem off the system without any delay
  - SHUTFINAL: Activities after stop is completed, if required
- Multiple passes for SHUTxxxx and SHUTFINAL, if required



© Copyright IBM Corporation 2019

2-33

## Stopping applications

The shutdown of an application can be done in three phases:

- SHUTINIT**: To prepare the environment so that the application can be stopped. If the desired goal of the resource is unavailable, then the SHUTINIT commands for the dependent resource are issued as soon as the specified condition for the supporting resource is satisfied. If there is no PrepareUnavailable, then the SHUTINIT commands are issued when the INGREQ is issued.
- SHUTxxxx is one or all of the following types:
  - SHUTNORM**: Commands for a normal, orderly shutdown; for example, stop VTAM normally with a Z NET,QUICK.
  - SHUTIMMED**: Commands for an immediate shutdown; for example, stop VTAM immediately with a Z NET QUICK or Z NET,CANCEL.
  - SHUTFORCE**: Commands to force the application to close; for example, a z/OS PURGE or CANCEL command.

The SHUTxxxx types are fixed (NORM, IMMED, or FORCE). Stop types are not supported.

A common implementation of SHUTxxxx processing uses **PASSes**. For example, when stopping TSO, you can define **PASS1** to send a shutdown notification message to all logged-on users. Then, you use **PASS2** to issue the command to end TSO.

3. **SHUTFINAL:** Commands to issue when the application ends. SHUTFINAL commands are issued only if the automation product stops the resource.

# Lesson 4 Monitoring and status

## Lesson 4: Monitoring and status

- Passive monitoring
- Active monitoring
- Automation agent status
- Automation manager resource statuses
- Mapping agent status to manager status values
- Application lifecycle and related statuses

### Monitoring and status

This lesson is about passive and active monitoring, the automation agent and automation manager resource statuses, mapping of automation agent status to automation manager status values.

## Passive monitoring

---

- Messages can be linked to agent status in policy
- When message is trapped, agent status is set as defined
- Preferred method as no active monitoring is required
- Sysgone events also update status implicitly
- Also used for MTR resources

### *Passive monitoring*

Messages can be linked to the agent status in the automation policy. When a message is trapped, the agent status is set as defined.

Passive monitoring is the preferred method as no active monitoring is required. Sysgone events also update the status implicitly.

Passive monitoring can be also used for MTR resources.

## Active monitoring

- Programmed routines run at regular intervals
  - Should not run too often due to overhead
  - Many built-in monitors available
  - Default monitor uses fast control block scan
- The status of each resource is checked for consistency with the address space status (active)
  - Active monitoring status of ACTIVE does not guarantee that resource is operable
- The status is set to ACTIVE or INACTIVE, if inconsistent
- Can detect status mismatches if messages for passive monitoring are missing
- For non-z/OS resources, other monitor routines apply
- Status can also be set from scripts
- Also used for MTR resources

### Active monitoring

All of the automation agent actions that are described so far are based on orders from the automation manager or triggered by messages that are defined in automation policy. Occasionally, some important messages are forgotten when the automation policy is defined. To compensate for the messages that are omitted, System Automation for z/OS actively monitors the status of defined resources. A monitor is run at a specified interval.

At regular intervals, the automation agent initiates what is termed a **monitor cycle**. During a monitor cycle, each resource is checked to ensure that its current automation status is consistent with its actual state. For example, if an address space is not running (inactive) and its agent status is UP, there is an inconsistency. In this case, the monitor routine sets the agent status to INACTIVE. Similarly, if the started task is active, and its agent status is different, its agent status is changed to ACTIVE.

Monitor cycles and intervals can also be defined for each resource. In these cases, agent status values of ACTIVE and INACTIVE indicate some failure in the automation agent processing. The failure can be caused by missing automation table statements. Operator action is typically needed.

## Automation agent status

---

- The status includes a fixed set of status values
- Status values are assigned by agents
- Status depends on monitor status, desired status, recent history, and intended action
- Status can be categorized:
  - **STARTING**: All starting phases
  - **AVAILABLE**: Initialized and ready for work
  - **STOPPING**: All termination types
  - **SOFTDOWN**: Down, but can be automated
  - **HARDDOWN**: Down, but manual intervention required
- Status is written to an automation status file
- Can be set with the SETSTATE command

### Automation agent status

The automation agent processes z/OS messages and issues commands. The automation agent maintains an agent status for each automated resource. The value of the agent resource status conveys several pieces of information:

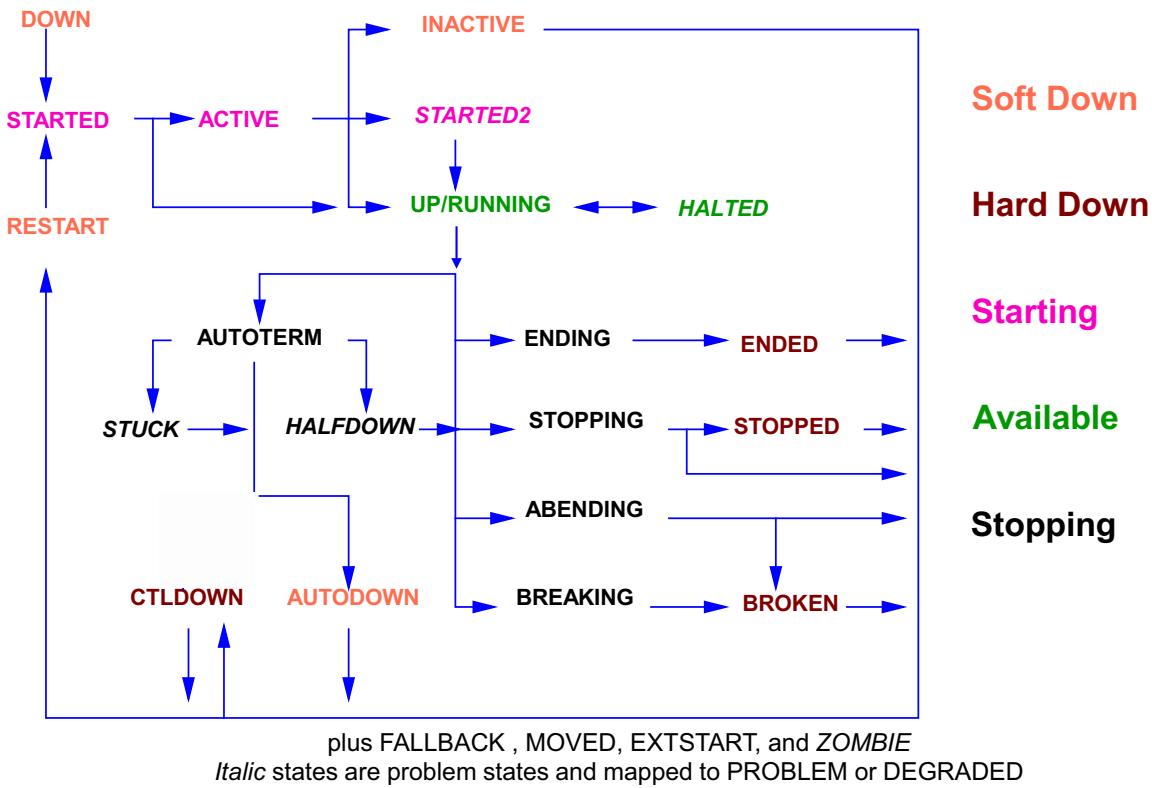
- Whether the resource is active
- Whether it is operating as desired
- How it was started or stopped
- How many times it failed

Status can be categorized:

- **STARTING**: All starting phases
- **AVAILABLE**: Initialized and ready for work
- **STOPPING**: All termination types
- **SOFTDOWN**: Down, but can be automated
- **HARDDOWN**: Down, but manual intervention required

The automation agent status is written to an automation status file and can be set with the SETSTATE command.

# Agent statuses and transitions



© Copyright IBM Corporation 2019

2-38

## Agent statuses and transitions

With a fixed number of states, the automation agent can be defined as a finite state model. The agent status of a resource also determines, in part, how the agent responds to future events, and what future status is assigned to the resource. The agent automation status values use common English (or pseudo-English) words; however, each value has a specific meaning within the automation agent model. The agent status mapping to the *observed status* is on the right.

States in *italic* font are problem states and mapped to PROBLEM or DEGRADED.

Soft down statuses mean they can be automated whereas hard down statuses need manual intervention.

The state transitions are triggered by:

- messages, like IEF403 for ACTIVE, the UP message for UP or the ABENDING or even final termination message
- Actions, like start or stop.

Then there are error statuses like:

- STUCK: resource still up, but no shutdown commands are available any more
- STARTED2: UP message has not arrived in specified time interval
- BREAKING: termination messages or codes have occurred that do not allow restart.
- ZOMBIE: resource should be down but the active monitor still is able to see it.

There are many agent statuses, but the primary flows that are listed are:

- A normal flow for a resource that is being started:
  - AUTODOWN -> RESTART -> STARTED -> ACTIVE -> UP
- A normal flow for a resource that is being stopped:
  - UP -> AUTOTERM -> AUTODOWN
- When ABENDS occur, the flow is:
  - UP -> ABENDING -> RESTART or BREAKING -> BROKEN

## Key status meanings

Agent status	Meaning
ACTIVE	The resource is running, but the <i>Up</i> message was not received yet. It is not ready for work
UP ENDED	The resource is <i>available</i> . It finished initializing and is ready for work
ABENDING	The resource encountered the <i>termination</i> message of an abend type that can be restarted
BREAKING BROKEN	The resource encountered an abend type that prevents a restart. Manual intervention required
AUTODOWN	The <i>final termination</i> message of the resource has arrived. Ready for restart if start conditions are satisfied
DOWN	Automation has never seen the resource active

### Key status meanings

The scenarios that are described in this section use terminology such as up message or active message. This slide explains the terms. Use this slide as a reference during the scenarios.

A default Desired Available setting in the policy of ALWAYS is also assumed for these scenarios, and System Automation for z/OS attempts to start the resource. In some cases, you might not want a resource active after an IPL. For those resources, you can code a Desired Available setting in the policy of OnDemand.

You can specify IEF404I as the final message, IEF403I as the active message, or IEF450I as the abend message. However, those messages are not the only ones that System Automation for z/OS processes. Final message, active message, and abend message are three phrases of System Automation for z/OS terminology.

A resource status of ABENDING indicates a termination message for a recoverable abend (it had ABEND=YES coded on the message call). Unrecoverable ABENDs have the termination message, BREAK=YES. System Automation for z/OS provides a mapping of most messages to these categories. More messages can be defined.

Not all messages apply to all resources. For example, non-MVS applications do not have the IEF403I active message or have an active message that is used as an up message.

## Automation manager resource status

Status category	Description
Observed status	Mapping of the resource status that the automation agent sent to the manager
Desired status	Available or Unavailable. The manager uses information from automation policy or requests to set the value
Startability status	Can be started
Automation status	Status of any orders sent to the automation agent
Health status	Status is determined by a monitor resource
Compound status	Aggregation of the previous five status values

### Automation manager resource status

The automation manager maintains six status values and three flags for each resource. Use the INGLIST or INGINFO commands to display these status values. For more information about each of these statuses and their possible values, including a status transition flow, see the *IBM System Automation for z/OS: User's Guide*.

The statuses that are maintained by the automation manager are:

- **Observed status** as monitored
- **Desired status** reflects the goal of a resource
- **Automation status**: Whether automation is working on the resource
- **Startability status**: Whether it is possible to start the resource if this is requested
- **Health status** as a result of monitoring
- **Compound status** is the composition of the other statuses

## Observed status values

Status	Description
AVAILABLE	Resource is up and running
STARTING	Resource is being started
STOPPING	Resource is stopping
SOFTDOWN	Resource is down. Automation can restart the resource
HARDDOWN	Resource is down. Automation cannot restart it
WASAVAILABLE	Has lost contact with the agent, but resource was available
UNKNOWN	No information is available
SYSGONE	<i>Host system left the SAplex</i>
PROBLEM	Resource has a serious problem, unusable
DEGRADED	For a server group, only some members are available. For an application, HALFDOWN or HALTED
STANDBY	Primary or secondary defined

Note: A suspended API's observed status is either AVAILABLE or SOFTDOWN

### Observed status values

The **observed status** is the up or down status of the resource. For applications, observed status is the interpretation of the status that the automation agent sends to the manager. These are the meanings of the status values:

- AVAILABLE: The application is UP or ENDED (for transient resources). It is operating.
- STARTING: The application is being started, either by automation or some other means. STOPPING is similar.
- SOFTDOWN: System Automation for z/OS can start the application.
- HARDDOWN: System Automation for z/OS cannot start the application.
- WASAVAILABLE: The manager lost contact with the agent. (Perhaps the agent is being recycled.) The application was available until contact was lost.
- UNKNOWN: The automation manager has no observed status for the resource because the agent is not responding. The manager assumes that the resource is unavailable.
- SYSGONE: The system on which the application runs left the SAplex.
- PROBLEM: The application is in a status that indicates it might be unusable.

- DEGRADED: The application is in either HALFDOWN or HALTED status.
- STANDBY: The application is an ARM application that is not active on its primary system. A copy of the application is active on another system.

For *application groups*, the status is derived from the statuses of its members, and all of these statuses are possible. DEGRADED means that some of the selected members of the group are available.

## Desired status values

Status	Description
AVAILABLE	Resource observed status must be AVAILABLE or WASAVAILABLE
UNAVAILABLE	Resource observed status can be HARDDOWN, SOFTDOWN, or SYSGONE

- Desired status determined by vote with highest priority.
- If no explicit vote exists, then default desired status is used.
- Default desired status is based on the Desired Available setting in the policy:
  - ALWAYS (default) - AVAILABLE
  - ONDEMAND - UNAVAILABLE
  - ASIS - UNAVAILABLE/AVAILABLE

### Desired status values

The automation manager tries to maintain the desired status (goal driven automation) for each resource that it controls. Several sources can generate requests to set or change the desired status for a resource:

Operators:

- Automation operators (REXX EXECs)
- End-to-end (E2E) automation manager
- Schedules
- IBM Workload Scheduler requests
- Application group requests
- Propagated votes

The status values have these meanings:

- **AVAILABLE:** The observed status must be AVAILABLE or WASAVAILABLE.
- **UNAVAILABLE:** The observed status can be HARDDOWN, SOFTDOWN, or SYSGONE.

**Desired Available** setting in the policy, a value that is specified in the customization dialogs for each automated resource, sets default desired status values.

- If Desired Available is set to ALWAYS, then the default desired status is AVAILABLE. Desired Available = ALWAYS is the default setting.
- If Desired Available is set to ONDEMAND, then the default desired status is UNAVAILABLE.
- If the Desired Available is set to ASIS, then the desired status is set to the observed status.

The default desired status applies when none of the current requests affect the resource.

## Startability status values

Status	Description
YES	Resource can be started
NO	Resource cannot be started because of a problem with the resource itself
INHIBITED	Resource cannot be started because of a problem with a supporting resource, or because automation is prohibited
DENIED	Resource cannot be started because automation is denied

### Startability status values

The **startability status** indicates whether a resource can be started. Startability status is derived from a combination of the following status values:

- Observed status
- Automation status
- Automation manager flags
- Startability of its supporting resources

## Automation status values

Status	Description
UNKNOWN	No connection to agent
IDLE	No orders were sent, no actions in place, but new orders can be sent. The resource is in a steady state
ORDERED	Order was sent, but not acknowledged
BUSY	Agent is processing the order, resource is starting or stopping, but not complete
DENIED	Agent cannot process last order received
PROBLEM	Agent detected a problem while processing order
INTERNAL	Automation of the resource is being handled internally

### Automation status values

**Automation status** indicates the status of an automation order that the manager sent to an agent. Sometimes, the automation manager sets the status; other times, the automation agent sets the status because the resource status changed. The values have these meanings:

- UNKNOWN: The automation manager does not have a connection to the automation agent that is responsible for the resource. No automation is done.
- IDLE: No order is outstanding. This status is typical for most resources. New orders can be sent to the agent.
- ORDERED: The manager is awaiting a response to an order that it sent.
- BUSY: The agent is processing an order; or a start or stop action not initiated by the automation manager is currently running.
- DENIED: The automation agent cannot process the order. A likely cause for this status is that one of the six automation flags is disabled. Another reason can be that the manager automation flag is off.
- PROBLEM: The agent encountered a problem while carrying out processing for the resource.
- INTERNAL: The resource automation is being processed internally.

## Health status values

Status	Description
NORMAL	Resource health is good
WARNING	Resource health is degraded
MINOR	Similar to WARNING, but more severe
CRITICAL	Similar to MINOR, but more severe
FATAL	Similar to CRITICAL, but more severe. ForceDown and failover for the application associated with the monitor
UNKNOWN	The monitor is not running, and it must be either started or fixed

The monitor resource (MTR) status also includes BROKEN and FAILED

© Copyright IBM Corporation 2019

2-45

### Health status values

The **health status** is calculated from the health status that is returned by health monitor resources. N/A means that there is no health monitor resource that is attached to the resource you are looking at.

The Health status values are:

- NORMAL: Resource health is good.
- WARNING: Resource health is degraded.
- MINOR: Similar to WARNING, but more severe.
- CRITICAL: Similar to MINOR, but more severe.
- FATAL: Similar to CRITICAL, but more severe. ForceDown and failover for the application associated with the monitor.
- UNKNOWN: The monitor is not running, and it must be either started or fixed.

## MTR Monitor Resource

The Monitor Resource entry type enables you to monitor the performance and health of an application. That is, a separate status informs you about the application's health. This health status

can be used by the automation manager to make decisions and, if necessary, trigger automation for the subject application.

The Monitor Resource (MTR) entry type allows you to obtain the health state of an object in two different ways:

- Actively, by polling, that is, executing a monitoring command periodically
- Passively, by processing events

Monitor resources are connected to application resources (APLs), systems, or application group resources (APGs) by means of the HasMonitor relationship. The health status of the monitored object is propagated along the HasMonitor relationship to the APLs or APGs and results in a health status there.

The monitor resource (MTR) status also includes:

- BROKEN: Both the monitor and recovery failed. This is a permanent condition.  
The monitor will not be re-invoked
- FAILED: The monitor has failed. Recovery may be in progress. No acceptable health status was provided.

Exception-based monitoring with OMEGAMON can also set the health status:

Internally there is also a DEFER status used in INGMTRAP to monitor OMEGAMON exceptions:  
Health status is set independent of the monitor routine by ING080I messages generated by INGMTRAP for each OMEGAMON exception found. An existing health state remains in effect until a new health state is set during message automation.

## Suspended flag values

Status	Description
DIR	The resource is <b>directly suspended</b> (got a suspend request) via INGSUSPD
IND	The resource is <b>indirectly suspended</b> (because a supporting resource was suspended) via INGSUSPD
PEN	The suspend request is <b>PENDING</b> as the resource is currently in a start or stop process. Therefore, the request is held back until the desired status has been reached
blank	The resource is not suspended

### Suspended flag values

**The suspended flag** is provided by the automation manager for all supported resource types. It is displayed on the INGLIST panel and can be queried by the INGDATA command. If the flag has the values DIR (Direct) or IND (Indirect), the manager is not going to start or stop the resource.

Possible values for the automation manager suspend flag are:

- DIR The resource is **directly suspended** (got a suspend request) via INGSUSPD or the suspend file.
- IND The resource is **indirectly suspended** (because a supporting resource was suspended) via INGSUSPD or the suspend file.
- PEN The suspend request has been recognized but the resource is currently in a start or stop process. Therefore, the request is held back until the desired status has been reached.  
Note: As long as the suspend request on a resource is **pending**, the agent doesn't get a suspend order and the agent automation flags do not display an 'S' for the automation agent flag value.
- **UNSUSPENDED/blank** The resource is not suspended. In INGINFO, you see the value of UNSUSPENDED behind the manager suspend flag, but in INGLIST this is only shown as a blank field.

## Compound status values

Status	color	Description
SATISFACTORY	Green	Observed status = desired status
DEGRADED	Yellow	APG: A subset of its desired members is available APL: Can be HALTED, HALFDOWN, or has a DEGRADED health status
INAUTO	Green	Resource is being started or stopped
INHIBITED	Pink	Resource not in desired status, and automation cannot proceed because of a problem with a supporting resource
AWAITING	Green	Waiting for supporting resources to reach the desired status
DENIED	Pink	Resource is not in desired status and automation unable to proceed. This status might be caused by automation manager flag settings or a resource that is suspended
PROBLEM	Red	Resource has a serious problem, and operator intervention is required

Note: A suspended APL's compound status is always SATISFACTORY

### Compound status values

**Compound status** is an aggregate of all the statuses for a resource. The compound status is used to determine the color of the resource on the INGLIST panel. INGLIST is described in the Commands and Operations unit.

DEGRADED can also mean that the resource is Starting or Stopping.

The compound status is illustrated in the following example:

The automation manager can display AWAITING for the compound status of TSO because it is waiting for VTAM to start. If the VTAM start failed, then the TSO status changes to INHIBITED.

## Mapping agent status to manager status values

Agent status	Observed status	Automation status
UP, ENDED	AVAILABLE	IDLE
AUTODOWN, DOWN, RESTART	SOFTDOWN	IDLE
BROKEN, CTLDOWN, STOPPED	HARDDOWN	IDLE
STARTED, EXTSTART, ACTIVE, RUNNING, ENDING	STARTING	BUSY
STARTED2	PROBLEM	PROBLEM
STUCK	STOPPING	PROBLEM
AUTOTERM, ABENDING, BREAKING, STOPPING	STOPPING	BUSY

plus HALFDOWN, STUCK, Fallback, MOVED, EXTSTART, and ZOMBIE

© Copyright IBM Corporation 2019

2-48

### Mapping agent status to manager status values

Whenever the agent detects a status change for an application, it updates the observed status and the automation status of the resource in the automation manager. For example, if the automation status is BUSY and the observed status is STARTING, then the agent status for the resource would be STARTED. To disable automation for a resource, set the automation manager flag to OFF or suspend the resource.

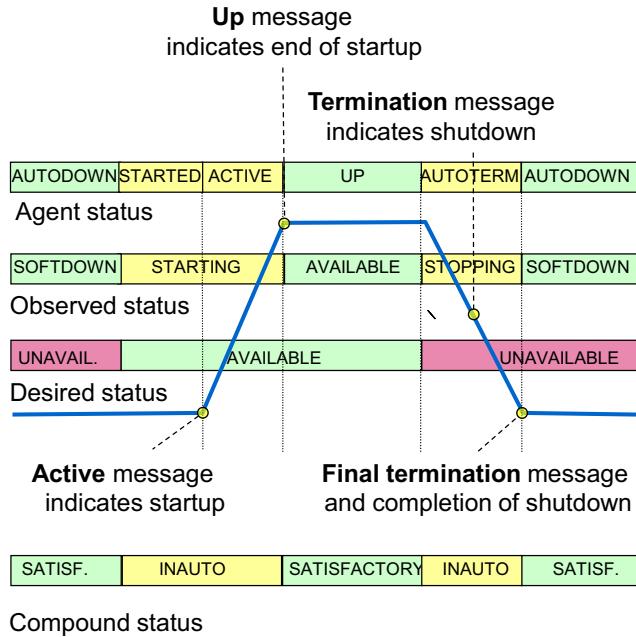
This table shows a few examples of the mapping of these changes. For the complete list of all 24 agent status values, see the *IBM System Automation for z/OS User's Guide*.

All agent status values are described in detail when you use the EXPLAIN command, as shown in the following example:

EXPLAIN STARTED2

The EXPLAIN command provides detailed information about the automation agent and the automation manager statuses.

# Application lifecycle and related statuses



- Multiple statuses are maintained for each resource:
  - Observed status as told by messages or monitors
  - Desired status is set by the policy goal and by (operator) requests
  - More statuses not covered here
  - Compound status, an aggregation of all statuses
- Messages cause change of resource status
- Status changes cause SA to react such that a satisfactory compound status is reached

(simplified)

© Copyright IBM Corporation 2019

2-49

## Application lifecycle and related statuses

This slide displays a simplified application lifecycle.

The desired status changes to AVAILABLE which triggers a startup. The compound status is set to INAUTO. The agent status is set to STARTED and the observed status to STARTING.

The result of the startup is the **active message** and a change of the agent status to ACTIVE.

Eventually the resource issues the **up message** which causes a change of the agent status to UP and an observed status of AVAILABLE. The compound status is set to SATISFACTORY as the desired status matches the observed status.

The desired status changes to UNAVAILABLE which triggers a shutdown. The agent status is set to AUTOTERM and the observed status is set to STOPPING. The compound status is set to INAUTO.

**Termination messages** might occur but they do not change the status for planned shutdowns. They are used to capture unplanned shutdowns and set the agent status to ABENDING to trigger recovery or to BREAKING.

The **final termination message** marks the completion of shutdown which causes a change of the agent status to AUTODOWN as well as an observed status of SOFTDOWN.

The compound status is set to SATISFACTORY as the desired status matches the observed status.

# Lesson 5 Relationships

## Lesson 5: Relationships

- Resources frequently depend on other resources
- Automation uses relationships for the specification of dependencies
- Relationships define how one resource depends on another resource for:
  - Start or stop of a resource
  - Preparation of start or stop of a resource
- Relationships exist between a **dependent resource** and one or more **supporting resources**
- Specialized, non-dependency external relationships, monitors, and for forcing down



© Copyright IBM Corporation 2019

2-50

### *Relationships*

Resources sometimes depend on other resources such as start and stop sequences. Each relationship expresses something of the nature of the dependency that it describes such as, HasParent, MakeAvailable, PrepareUnavailable, and so on.

These dependencies are modeled in your policy as relationships. Relationships can be active or passive. Active relationships propagate goals for resources across the dependency. Passive relationships do not.

For example, almost everything requires JES, the VTAM product, or TCPIP to be active. When the automation product wants to start TSO then it propagates the request across along the relationships to the VTAM product and then to JES.

## How relationships work (1 of 2)



- Relationships have a direction
  - Dependent resource has the dependency
  - Supporting resource can fulfill the dependency
- Applicable to applications and application groups
- Some relationships can have an extra **condition** to be evaluated
  - The relationship is evaluated only if the additional condition is true
  - The relationship conditions that are available depend on the relationship
- Relationships do not generate requests
- Relationships that define the start/stop order are evaluated at start/stop time of the resource only
  - Relationships not applicable to the intended action are not used

© Copyright IBM Corporation 2019

2-51

### How relationships work (1 of 2)

Relationships have a direction from the source resource that has the dependency to the target or supporting resource. Valid sources are a managed resource or a resource group. Valid targets are resources or resource groups. Relationships identify an action and a **condition** that they 'add' to that action being taken. The relationship is evaluated only if the additional condition is true. The relationship conditions that are available depend on the relationship.

Relationships do not generate requests. They merely can propagate votes.

### Relationship Evaluation

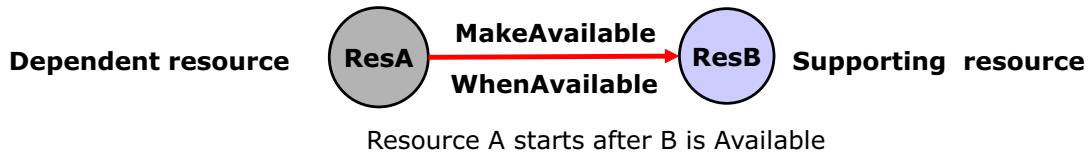
Relationships that define the start order are evaluated at start time of the resource only.

Relationships that define the stop order are evaluated at stop time of the resource only.

When a supporting resource changes the state after the order has been sent, it has no effect.

Relationships not applicable to the intended action are not used. For example, when there is a MakeAvailable request, MakeUnAvailable relationships are not used.

## How relationships work (2 of 2)



Defining relationships:

- Specify relationship type
  - Preparing for startup and shutdown: PrepAvailable, PrepUnavailable
  - Startup and shutdown: MakeAvailable, MakeUnavailable
- Specify optional, allowed condition (Default condition applies)
- Automation option: Active versus Passive
  - For an active relationship automation attempts to bring the supporting resource into a status such that the condition is fulfilled
- Chaining: Strong versus Weak
  - STRONG: The status of the attached subtree is considered
  - WEAK: Only the status of the supporting resource is considered
- ForceDown: Forces a shutdown of the dependent resource

© Copyright IBM Corporation 2019

2-52

How relationships work (2 of 2)

Definitions for resource dependencies can involve:

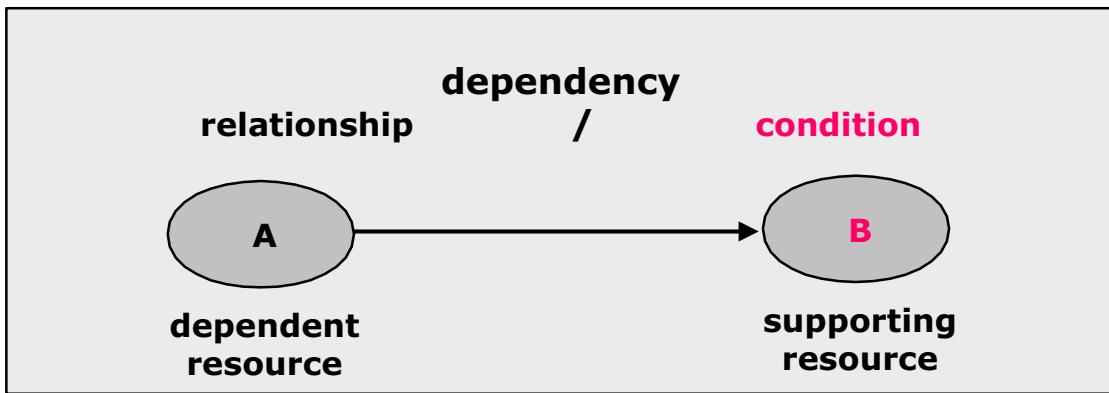
- Preparing for startup and shutdown: PrepAvailable, PrepUnavailable
- Startup and shutdown: MakeAvailable, MakeUnavailable
- Resource availability or unavailability **conditions** associated with relationship
- Enabling the automation to change the status of a resource to achieve a goal: **Active versus Passive**. For an active relationship automation attempts to bring the supporting resource into a state such that the condition is fulfilled.
- **Chaining: Strong versus weak**: Defines whether the status of the supporting resource or attached subtree is to be considered:
  - STRONG: The automation product considers the status of the attached subtree.
  - WEAK: The automation product considers only the status of the supporting resource.

In no way does a relationship cause an action to be taken (except for ForceDown), it simply adds another condition that must be fulfilled before that action can be taken. They can propagate demand and votes, which can change the desired state of the supporting resource, but they are merely passing things along, not creating things.

A **ForceDown** relationship identifies a condition under which a high priority stop request is injected (which in turn generates a vote), it does not directly inject anything.

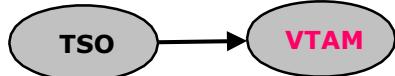
A ForceDown relationship is evaluated whenever the supporting resource reaches the specified condition.

## Relationships example



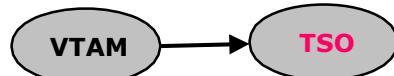
### Examples

#### MakeAvailable / WhenAvailable



Before TSO can be started, VTAM must be available

#### MakeUnavailable / WhenUnavailable



Before VTAM can be stopped, TSO must be unavailable

#### Relationships example

A dependency states that before an action is taken against one resource (the dependent resource), there must be a specific condition on another resource (the supporting resource). A dependency is formally defined as a combination of a relationship and a condition. The relationship between two resources indicates a desired action for the dependent resource. A relationship can be forward or backward.

Condition indicates the required status of the supporting resource. Before the manager starts the action for the dependent resource, the condition of the supporting resource must be satisfied. In this sense, a dependency acts as an inhibitor. It prevents the manager from sending an order until some defined condition is satisfied. Dependencies also have a role in determining the actions that are taken. Dependencies can be overridden by options on operator commands.

There is a MakeAvailable relationship between TSO and its supporting resource, VTAM. The condition is WhenAvailable. Therefore, the dependency is MakeAvailable/WhenAvailable. A forward relationship is a relationship of the dependent resource (TSO, for example) to the supporting resource (VTAM, for example). A backward relationship is a relationship where the resource acts as a supporting resource. There are some situations that are exceptions to this model. Only Forward relationships can be defined in the customization dialogs.

Backward relationships are generated automatically.

## List of relationships

- HasParent: Startup and stopping
    - Can be expressed by a combination of MakeAvailable and MakeUnavailable relationships
  - HasPassiveParent: A passive HasParent
  - MakeAvailable: Starting
  - MakeUnavailable: Stopping
  - PrepAvailable: Preparation of starting – issue prestart command or commands
  - PrepUnavailable: Preparation of stopping – issue shutinit command or commands
- Specialized, non-dependency relationships:
- Externally: External resource is responsible for starting or stopping
  - HasMonitor: Points to monitor resources
  - PeerOf: Votes are propagated to dependent resource
  - ForceDown: Forces a shutdown of the dependent resource
  - HasMember (generated automatically for groups)

### *List of relationships*

HasParent can be used for simple relationships between two resources. It is really a combination of two dependencies (MakeAvailable/WhenAvailable and MakeUnavailable/WhenUnavailable). ForceDown is the only relationship that generates a vote (MakeUnavailable) by the automation manager and is then withdrawn when the vote is satisfied.

Startup and shutdown MakeAvailable, MakeUnavailable, ForceDown.

Preparing for startup and shutdown: PrepAvailable, PrepUnavailable

Externally: External resource is responsible for starting or stopping.

HasMonitor: Points to monitor resources.

PeerOf propagates MakeAvailable votes from a supporting resource to dependent resources but halt the propagation of MakeUnavailable votes unless 'Scope=ALL' is specified on the stop request.

HasMember (for groups).

## List of conditions

WhenObservedAssumedDown	WhenRunning
WhenObservedAvailable	WhenRunningOrStarting
WhenObservedDown	WhenStoppableOrAssumedDown
WhenObservedHardDown	WhenStoppableOrDown
WhenObservedRunning	WhenStoppableOrSoftdown
WhenObservedSoftDown	WhenGroupHasNotFailed
WhenObservedAssumedDownOrStopping	WhenHardDown
WhenObservedDownOrStopping	WhenHealthAssumedNormal
WhenObservedWasAvailable	WhenHealthFatal
WhenObservedWasAvailableUnknownOrSysgone	WhenHealthNeitherNormalNorFatal
WhenAssumedDown	WhenHealthNormal
WhenAssumedDownOrStopping	WhenHealthNotFatal
WhenAvailable	WhenHealthNotNormal
WhenAvailableOrStarting	Only for HASPARENT, HASPASSIVEPARENT, EXTERNALLY: • StartsMe • StartsMeAndStopsMe • StoopsMe
WhenDown	
WhenDownOrStopping	
WhenSoftDown	

Note: There is a default condition for each relationship.  
Not all conditions are available for a specific relationship.

### *List of conditions*

A condition indicates the required status of the supporting resource. Many different conditions can apply to a dependency. The observed status of a supporting resource tells the automation manager if a dependency is satisfied. For more information, see System Automation for z/OS Defining Automation Policy.

Conditions are related to different resource types and purposes:

- Application observed or “assumed” status. Combinations with stoppable
- Group observed and recovery status
- Health status
- Specification of role of parent for parent and external dependencies

Dependency examples:

- MakeAvailable/WhenAvailable: Make resource available only when supporting resource is available
- MakeAvailable/WhenDown: Make resource available only when supporting resource is down
- MakeAvailable/WhenStoppableOrDown: Make resource available only when supporting resource is stoppable or down



**Note:** There is a default condition for each relationship. Not all conditions are available for a specific relationship

# Lesson 6 Service periods

## Lesson 6: Service periods

- Service periods define UP and DOWN periods
  - Inject MakeAvailable or MakeUnavailable votes at the start of a defined service window
  - Withdraw the vote when the period expires
- The automation manager controls service periods
- The service period definition (schedule)
  - Have up to six service windows each day
  - Specify service windows by day of week, weekend, or daily
  - Priority high and low
  - Linked to multiple resources
- Use the command INGSCHED to override active service periods

© Copyright IBM Corporation 2019

2-56

### Service periods

**Service Periods** (schedules) are used to define windows of time during which linked resources are automatically started or stopped. These windows can be modified (overridden) at runtime. Schedules are defined independent of resources, and each schedule can link to multiple resources. Use the INGSCHED command to override schedule definitions. You can use schedules to manage an entire week (weekly). Each week is assumed to have requirements similar to every other week. There are no provisions for special days, such as holidays.

The automation manager handles all schedules. Requests to be up or down are effectively requests that the desired status of a linked resource might be Available or Unavailable. At the beginning of an UP service period, a MakeAvailable request is created. At the end of the service period, the manager withdraws the request. At the end of the MakeAvailable window, the resource might remain available if there is no MakeUnavailable vote against it. As an alternative to service periods, IBM Workload Scheduler for z/OS can be used as a scheduling mechanism.

One common use of service periods is to define a 24x7 down window for resources that must be down after an IPL. As an alternative, those resources can have a Desired Available setting in the policy of ONDEMAND. Resources that are defined with a Desired Available policy of ONDEMAND are not started when operators IPL the system. Operators must manually start those resources.

With SA V4.1 it is also possible to suspend resources after an IPL. Operators must manually resume those resources.

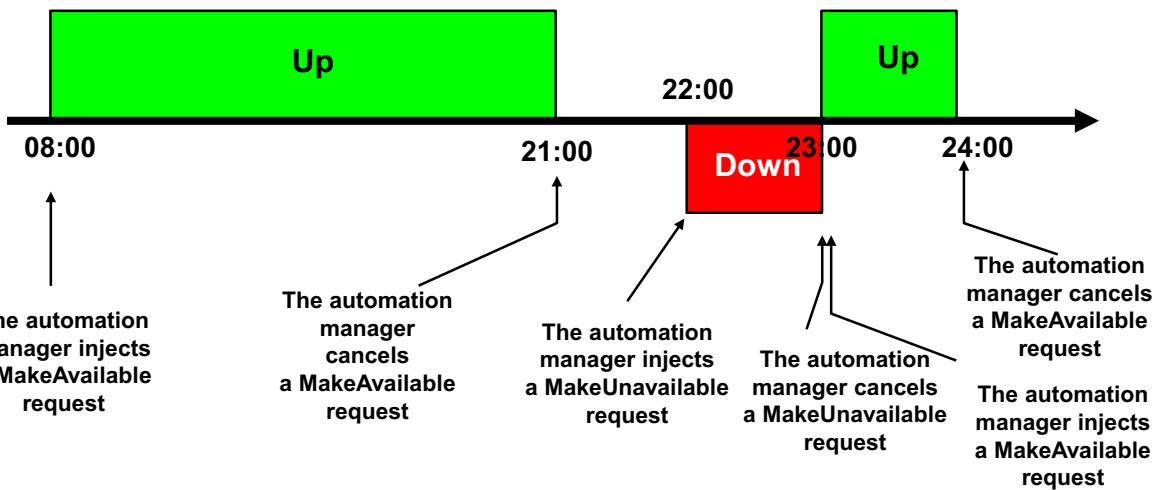
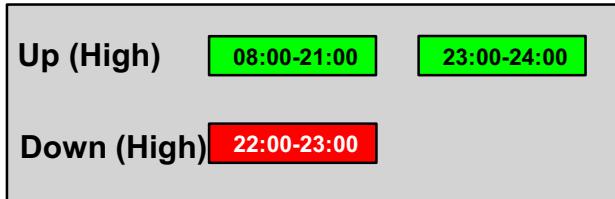
When defining a service period, keep in mind that the startup time is the time at which automation is to begin startup processing. It is not the time at which the resource is available. The time until availability depends on how long the startup of an application or an application group takes.

For each UP and DOWN schedule, you can specify a priority:

- Low priority. Orders resulting from a service period with low priority can be overruled by an operator's INGREQ command. This is the default.
- High priority. H schedules override L schedules and low priority operator requests.

According to the goal driven concept, a vote to start a resource is generated from a service period that you enter in a UP service window. A vote to stop a resource is generated from each service period that you enter in a DOWN service window. These votes must compete with other votes that have been probably issued (for example, due to operator request). At the end of the service period, these votes are removed from automation. An end of an UP service window does not mean implicitly the start of a DOWN service window.

## Service period example



© Copyright IBM Corporation 2019

2-57

### Service period example

This slide shows an example of how MakeAvailable and MakeUnavailable requests are generated and canceled for resources that are linked to service periods. Service window requests are canceled when a service window ends. Canceling the request does not necessarily mean that there is a status change. The resource can continue in its existing state, depending on other requests for the application.

For example, use the service periods that are shown on the slide:

- At 08:00, the automation Manager issues a MakeAvailable request.
- At 21:00, the automation manager removes that request. That does not mean that the application is brought down. If there are no other requests, it remains up.
- At 22:00, the automation manager issues a MakeUnavailable request to stop the application.
- At 23:00, the automation manager removes the MakeUnavailable request, then the automation manager issues a MakeAvailable request to start the application.

In the example, at 21:00 the application can continue to be Available even though its Up window ended. If there are no stop requests received by the automation manager to close the application, then it is not stopped. What you are specifying is to keep the application available between 21:00 and 22:00, unless there is another request that wants it unavailable.

## Overriding service periods

- The service period can be overridden with the INGSCHED command
- There are two types of overrides
  - Service period override
    - Overrides to the service period definition
    - Applies to all resources linked to the service period
  - Resource override
    - Overrides for a single resource only
    - Takes precedence over service period overrides

© Copyright IBM Corporation 2019

2-58

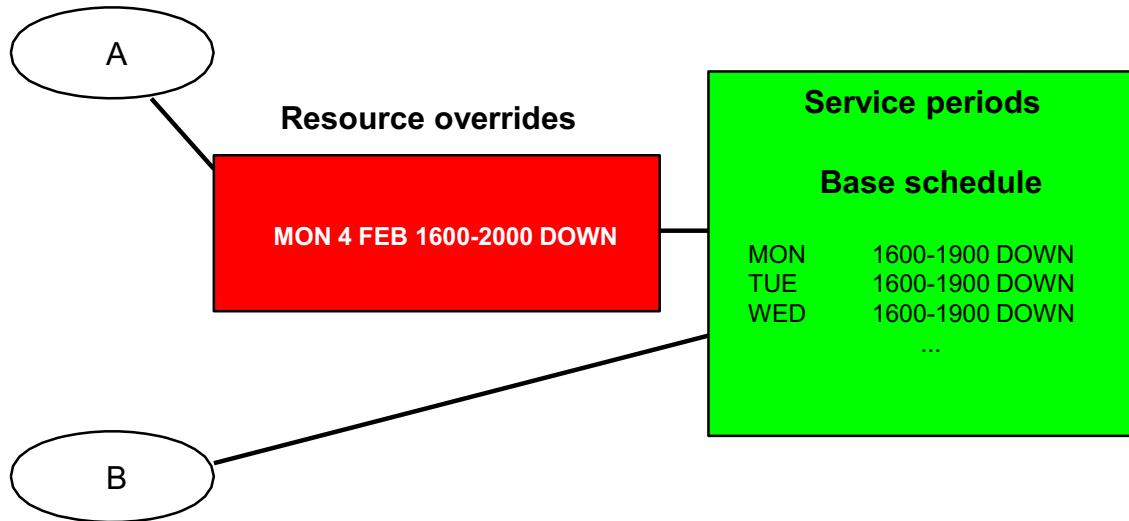
### Overriding service periods

Operationally, it is possible to override these items:

- A service period definition. A service period override affects all resources that are linked to the service period.
- The service period for a specific resource.

**Overriding a service period** does not alter the base service period that is defined in the policy. Override definitions consist of changes to service windows for a single day that are used in place of the base schedule. The next slide demonstrates the concept.

## Service period resource override example



Resource overrides are applied to a specific resource only

### Service period resource override example

You can make overrides that apply to only one resource. Service windows in a resource override are changed in the same way as they are for base schedule overrides; that is, a day at a time. The overrides in a resource override are only applied to the one resource linked to the override. Other resources use the unchanged base schedule.

The following example is shown on the slide:

- On Monday, February 4, application A is stopped between 16:00 and 20:00, according to the override. A MakeUnavailable vote is generated.
- On all other days in the future, including all other Mondays, the base schedule is used.
- Application B is stopped between 16:00 and 19:00 according to the base schedule.

# Lesson 7 Application groups

## Lesson 7: Application groups

- Definition of an application group
- Types and natures of application groups

© Copyright IBM Corporation 2019

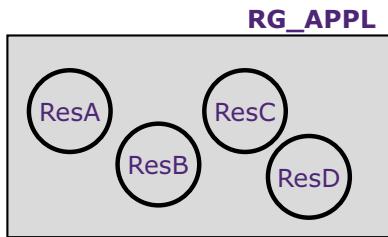
2-60

### *Application groups*

This lesson introduces **application groups**, **group types**, **group natures**, and their attributes. The unit highlights and explains the benefits of groups when managing automated business applications in an enterprise.

# Application groups (1 of 2)

- Container for a collection of application resources (APL) or resource references (REF)
- Allows the group to be handled as one entity
- Frees operator from knowing the various pieces that the application consists of
  - Operator can monitor or control
  - Allows activation or deactivation of resource groups and their members
- Status derived from the aggregated status of its members



© Copyright IBM Corporation 2019

2-61

## Application groups (1 of 2)

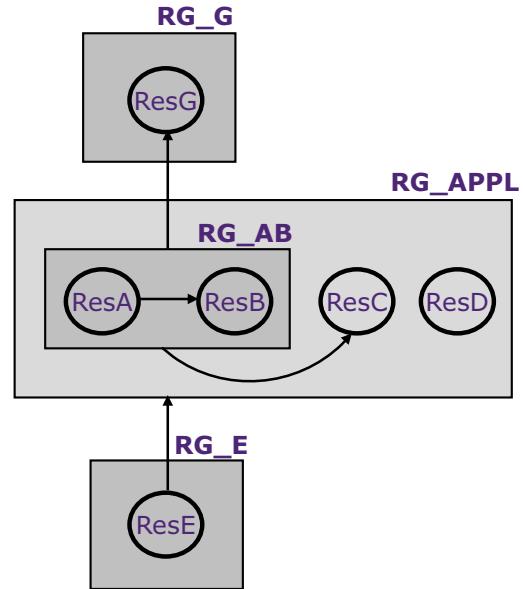
### An **application group**

- Is a logical container for a collection of resources
- Is used for multiple resources as a single logical entity which can reduce the complexity of applications
- Is the primary mechanism for operations within IBM System Automation and frees operators from knowing the various pieces that the application consists of

The Resource group status derived from the aggregated status of its members.

## Application groups (2 of 2)

- Members can be on different systems in the SAplex, other SAplexes or non-z/OS systems
- Applications can be in multiple application groups
- Can be part of any relationship
- Can be nested



© Copyright IBM Corporation 2019

2-62

### Application groups (2 of 2)

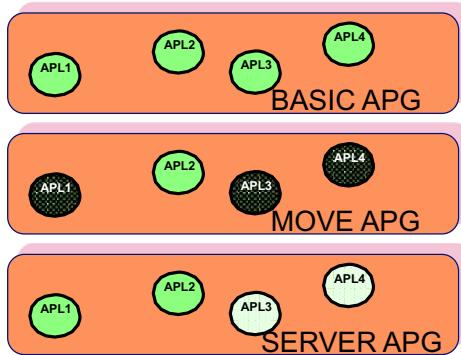
Application groups can have any number of applications as members.

Also, application groups can be defined in such a way that their members can be on different systems in the SAplex, other SAplexes or non-z/OS systems.

An application can be a member of multiple application groups and a group can be part of any relationship.

Application groups can also be nested, meaning that applications can be split into several resource groups which themselves are part of another higher level application group.

## Group type and nature



- Type is either system or sysplex
- Three ‘natures’ of groups
  - BASIC: The group is available when all of its resource members are available
  - MOVE: The group is available when one resource member is available
  - SERVER: The group is available when one or more resource members are available
- Groups can be active or passive

### Group type and nature

There are two types of application groups:

- **System application groups:** All members must run on a single system.
- **Sysplex application groups:** Members can run on any system in the SAplex that an automation manager controls. Typically, the members are applications which can be moved across systems in the SAplex.

Within a System Automation for z/OS managed SAplex, a named system application group can be defined on multiple systems in the SAplex. The names of sysplex application groups must be unique within the SAplex.

The **nature of an application group** describes its availability behavior:

- How it ensures that the correct number of applications are started to satisfy the requirements of the group
- How applications are selected to start or stop
- How the status of the group is determined from the status of its members

**BASIC:** The group is available when all of its resource members are available.

**MOVE:** The group is available when one resource member is available.

**SERVER:** The group is available when one or more resource members are available.

**Groups can be active or passive.** A passive group does not propagate votes to its members.

# Lesson 8 More automation policy

## Lesson 8: More automation policy

- Defaults
- Major and minor resources
- Agent automation flags
- Threshold processing
- Notify operators
- Message policy
- Message capturing
- Runmode
- Pacing gates

© Copyright IBM Corporation 2019

2-64

### *More automation policy*

This lesson explains:

- Defaults and major and minor resources
- Agent automation flags: If an event occurs that triggers automation, automation checks the appropriate flag to determine whether automation is currently on
- Threshold processing can be used to stop recovery for an automated resource
- Notify operators are users who are defined to receive automation messages
- Message automation policy
- Message capturing
- Runmodes to start or stop your system in stages
- Pacing gates to avoid CPU peaks when starting or stopping bulk applications

## Defaults

- Policy values are searched from most to least specific :
  1. **Resource**: application, application groups, MVS component, monitor
  2. **Class** (application only)
  3. ADF **Application** defaults (applications only)
  4. MDF MVSCOMP defaults (**MVS components** only)
  5. SDF **System** defaults (not for sysplex application groups )
  6. XDF **Sysplex** defaults (sysplex application groups only)
  7. Product defaults
- Defaults only cover a policy subset like:
  - Desired Available
  - Inform List
  - Automation flags and thresholds
  - APPLICATION INFO (selection)

### *Defaults*

This slide shows the search by the automation product when it queries policy definitions for a resource. For each resource, the installation can specify many attributes. These attributes include policy items such as thresholds, restart specifications, start timeout, and shut delay. It is not necessary to manually define every attribute for each resource. Default values can be specified for many attributes.

For example, if an automation flag is not specified for a resource, the application default is used. If no value is specified in application defaults, the system default is used. If none is specified there, the default (YES) is used.

One of the benefits is to easily change the automation policy. For example, all automation can be turned off by setting the AUTOMATE flag in System Defaults to NO. Specific definitions always override default values.

Policy values are searched from most to least specific:

1. Resource: application, application groups, MVS component, monitor
2. Class (application only)
3. ADF Application defaults (applications only)

4. MDF MVSCOMP defaults (MVS components only)
5. SDF System defaults (not for sysplex application groups )
6. XDF Sysplex defaults (sysplex application groups only)
7. Product defaults

Defaults only cover a policy subset like:

- Desired Available
- Inform List
- Automation flags and thresholds
- APPLICATION INFO (selection)

## Major and minor resources

- Used for agent automation flags and threshold processing
- Major resources are:
  - Applications
  - Generic settings
    - SUBSYSTEM: all applications
    - MVSESA: all MVS components
    - System DEFAULTS
- Minor resources belong to a major resource and can be pre-defined or dynamically created, representing for example:
  - Messages
  - Transactions
  - Single MVS component under MVSESA:
    - LOG, SYSLOG, LOGREC, MVSDUMP and SMFDUMP
- Minor resources notation:
  - Major\_resource\_name.minor\_resource\_name
  - Example: MVSESA.SMFDUMP, VTAM.IST020I

© Copyright IBM Corporation 2019

2-66

### Major and minor resources

Major and minor resources are used for agent automation flags and threshold processing.

**Major resources** are:

- Applications
- Generic settings
- SUBSYSTEM: all applications
- MVSESA: all MVS components
- System DEFAULTS

You can also define your own resources, called **minor resources**. Minor resources belong to a major resource and can be pre-defined in the policy or dynamically created by the operator, representing for example:

- Messages
- Transactions
- Single MVS component under MVSESA:  
LOG, SYSLOG, LOGREC, MVSDUMP, and SMFDUMP

Minor resources notation:

- Major\_resource\_name.minor\_resource\_name
- Example: MVSESA.SMFDUMP, VTAM.IST020I, CICS.TRANS

Minor resources take, by default, the automation settings of their major resources. The main purposes in defining minor resources are to:

- Enable automation for minor resources
- Override the automation settings of major resources for a single minor resource, or a group of minor resources

Minor resources can be pre-defined in the policy using the Minor Resource Definitions panel which is displayed if you select the MINOR RESOURCES policy item in the Policy Selection panel for the MVS Component or applications entry types.

## Agent automation flags values

- Control what automation is active at any time
- Automation flags can be predefined in the customization dialogs or can be set during run time
- Effective flag value is searched from most to least specific:
  - – There is no explicit setting at this level
  - Y The flag is turned on
  - N The flag is turned off
  - E The value of the automation flag depends upon the values returned by one or more user exit and will be determined when the value is required
  - L Resulting commands or replies are logged only
  - S The flag is turned off, because the resource is suspended
- Turn automation flags off for a scheduled time period
- Commands: DISPFLGS, INGAUTO, DISPSCHD

### Agent automation flags

After defining the policy for all your resources, you might want to disable some policy for the following reasons:

- Changes to operating policy
- A phased implementation
- Failure of some segment of the defined policy

Switches within the automation agent allow you to control how much of your defined automation is active at any time. These switches are called **agent automation flags**. With the agent automation flags, you can control these items:

- Actions that are driven by specific messages
- Actions such as shutdown for individual resources
- All actions to individual resources
- Specified actions to all resources
- All actions to all resources

The agent automation flags are defined in the policy and can be overridden by the operator.

If an event occurs that triggers automation, automation checks the appropriate flag to determine whether automation is currently on.

The **effective flag value** is searched from most to least specific (see defaults).

These automation **flag values** are valid:

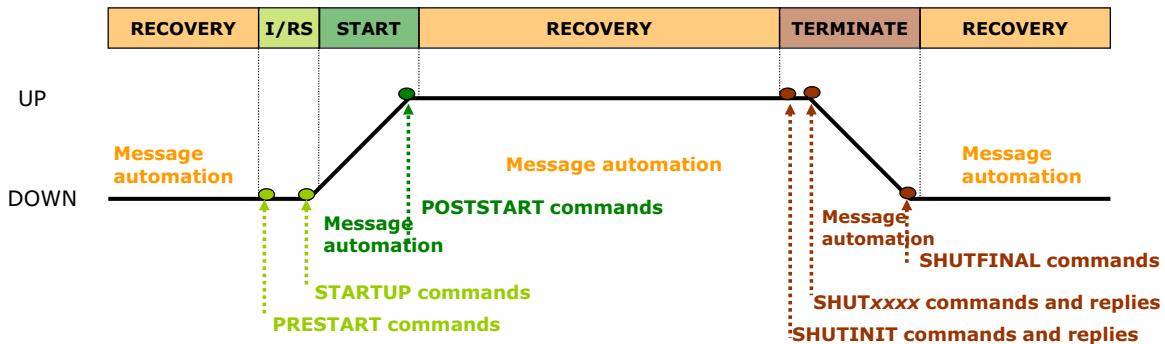
- YES: Allow automation of the resource
- NO: Do not allow automation of the resource
- EXIT: Identifies user exit that is called. Within the exit, it is determined whether to automate the resource
- L: Resulting commands or replies are logged only
- S: The flag is turned off, because the resource is suspended

The customization dialog allows you to turn automation flags off for a scheduled time period. You can view these time period settings by issuing the *DISPSCHD* command.

To determine the actual flag values that have been set in the customization dialog or during runtime, together with the effective flag values for a particular subsystem, use the *DISPFLGS* command for this subsystem. It displays the actual and effective flag values for the selected subsystem and for related minor resources.

To change automation flags use the *INGAUTO* command for this subsystem. It also allows to set a specific time period to turn automation on or off for all or selected flags.

## Agent automation flags



- InitStart flag (I): Checked after IPL only, when the application has a true DOWN status
- Restart flag (RS): Tested in all other DOWN statuses
- Start flag (S): Checked for automation after the STARTUP command is issued and for POSTSTART commands
- Terminate flag (T): Controls all shutdown commands and automation during shutdown
- Recovery flag (R): Indicates whether recovery of the resource is allowed
- Automation flag (A): Global automation flag for the resource. If NO, all flags are NO

© Copyright IBM Corporation 2019

2-68

### Agent automation flags

From initialization through normal operation to shutdown, a resource can have several automation modes, such as being started, in an up state, abending, and shutting down.

Automation uses resource states to track these automation modes for monitored resources. A resource must have a particular status for certain automated actions to occur. The effect of these actions may, in turn, change the resource status from one value to another.

If an event occurs that triggers automation, automation uses the associated resource's status to determine the setting of the flag that controls that specific phase in the lifetime of a resource.

The different agent automation flags are:

The **AUTOMATION** flag of the agent is an overriding flag for all the other agent flags. If the AUTOMATION flag for a resource is set to N (No) or S (Suspended), then the agent denies all automation requests for the resource. By default, message capturing, threshold checking, and WLM resource management are still done, even when this flag is set to N or S.

In all cases, the AUTOMATION flag is checked first, and only if it is set to YES are the other flags checked. Therefore, you can use the AUTOMATION flag to check all automation for a resource.



**Note:** The value of S (Suspended) can occur only if a resource is suspended using the INGSUSPD command or if it's part of the suspend file. It is neither possible to set the agent automation flag manually to S nor defining the value S in the customization dialog.

The **INITSTART** flag controls whether a resource startup is allowed following IPL or NetView restart. It covers only the issuing of the initial startup command. It does not cover any other restart actions. INITSTART flag applies only when agent status is DOWN.

The **START** flag controls any additional automation actions that are done during the startup process. Automatic actions include automated replies to WTORs, and reactions to specified messages. Despite its name, the start flag does not determine whether start commands are issued.

The **RESTART** flag activates and deactivates restart actions. The value of this flag determines whether start requests are honored.

Any start after the initial start is considered a restart. The restart can come from an operator, service period, IBM Workload Scheduler, for example.

The **TERMINATE** flag (often also called the SHUTDOWN flag) controls whether a shutdown request (for example, INGREQ STOP) can be initiated or continued. It also controls whether any automation is done when replying to WTORs and issuing subsequent commands as part of the shutdown process.

The **RECOVERY** flag is not used for resources except to control JES spool monitoring. It is used in sysplex automation to manage the couple data sets, coupling facility, system logger, paging data sets, and write to operators (WTO) enqueues.

When the automation agent receives a start or stop order from the automation manager, the agent will check the relevant automation agent flag before issuing the actual start or stop command for the resource. If automation is not allowed according to the flag setting, the automation agent will reschedule itself on a timer and re-check the flag setting. The automation status is set to Denied.



**Important:** Agent automation flags can block goal driven automation. Therefore they should not be used to control the start or the stop of a resource. Usage of goals including the suspend goal are recommended instead. Note, there is also an automation manager automation flag. If the automation flag is off, the automation manager does not send any orders for that resource to the automation agent. The automation and compound status are then DENIED. If the agent automation flag is off, INGREQ denies entering a request.

## Threshold processing

- Before a major resource is restarted or a minor resource is recovered or automated, specific or generic failure thresholds are checked
- There are three thresholds: Infrequent, frequent, and critical
- Actions are taken when a threshold is met

Threshold	Actions taken
<b>INFRequent</b>	Issue message Attempt restart. Recover or automate
<b>FREQuent</b>	Issue more serious message Attempt restart. Recover or automate
<b>CRITical</b>	Issue critical message Do NOT attempt restart; status is BROKEN Do NOT recover or automate

© Copyright IBM Corporation 2019

2-69

### Threshold processing

A *threshold* is a counter that tracks the number of post failure restarts or automation actions that a resource experienced during a specified period.

The automation product uses three thresholds when automating the restart of resources or when automating or recovering minor resources. These thresholds can be used to identify the frequency of failures or automation actions for an automated resource.

The primary use of error thresholds is to track subsystem abends and ensure that the abend and restart cycle does not become an infinite loop, but they may also be customized for other uses.

Automation provides three threshold levels for which you can specify criteria:

- Infrequent (INFR)
- Frequent (FREQ)
- Critical (CRIT)

In all cases, when a threshold is met or exceeded, a message is issued. This message is written to the NetView log and can alert operators.

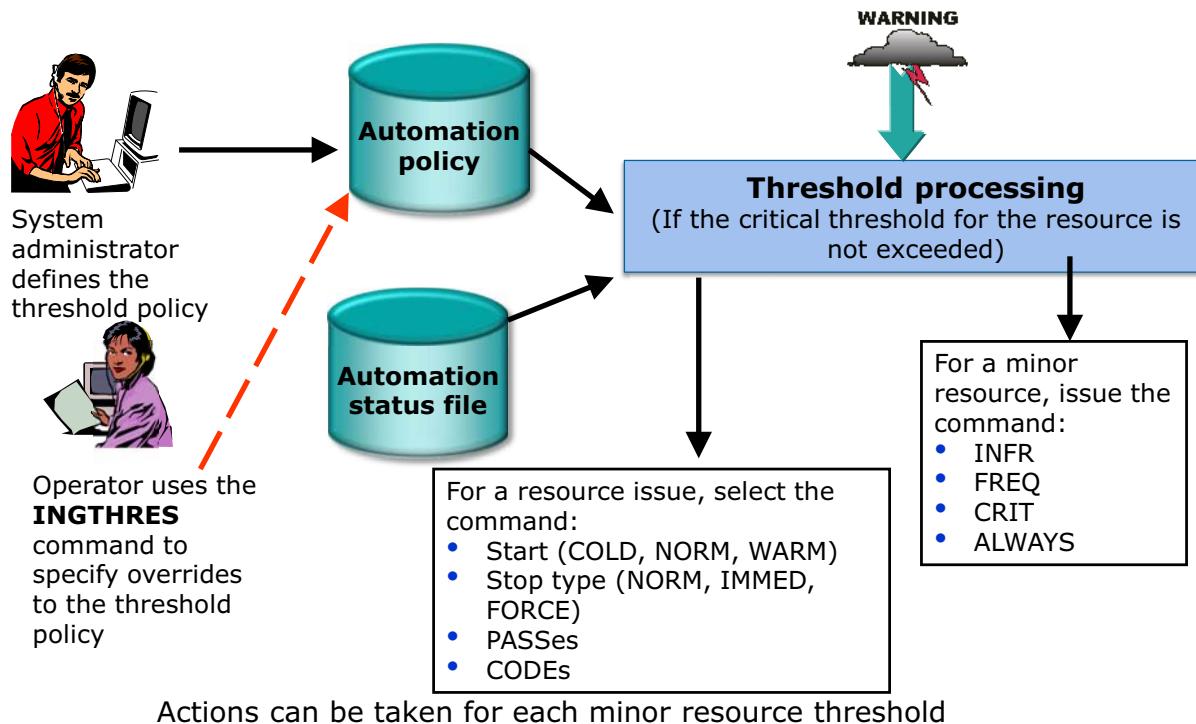
If the application has not exceeded its critical threshold, the automation product posts the application to RESTART and attempts to restart it. Restart automation is stopped when a defined

CRIT (critical) threshold is met.

When the critical threshold is met:

- Automatic restart of the resource is halted
- Its agent status is set to BROKEN
- Its manager compound status is set to PROBLEM
- Notify operators can be alerted.
- Operator intervention is required for the automation product to restart the resource.

## Threshold processing and minor resources



© Copyright IBM Corporation 2019

3-70

### Threshold processing and minor resources

Minor resource thresholds allow for the automation product administrators to define automated actions as reactions to each threshold, or a default action for all thresholds.

Automation tests minor resource thresholds for commands that are issued for a resource; for example, from PASSes or CODEs.

LOG, SYSLOG, LOGREC, MVSDUMP and SMFDUMP are now defined as minor resources of MVSESA with thresholds and actions. You can define actions for INFRequent, FREquent, and CRITical minor resource thresholds. You can also define actions for automation to take if no minor resource threshold is exceeded (ALWAYS). No actions are taken if the resource exceeds its critical threshold.

If you want to display all defined thresholds or add, change, or delete threshold settings for a particular resource, use the INGTHRES command. This displays the related Command Dialogs panel.

For z/OS components, such as dump data sets or log data sets, you can define thresholds to limit the frequency of how often they may be deleted after they have filled up without an action being taken or a notification being sent to the operator.

If you want to check the error counts for resources on a specific system, issue a DISPERRS command to this target system. SA z/OS displays the DISPERRS Command Dialogs panel with a list of all the errors that have been recorded for resources on the target system. Although DISPERRS displays the errors that have occurred for all of a system's resources, DISPASF lets you see detailed information about errors for a resource on a specific system.

## Message assignment

- Messages can be assigned by message prefix
- The NetView **ASSIGN** command is used to assign unsolicited messages that are generated by automated resources
- Unsolicited messages are assigned to automation operators (autotasks) so that all messages from same job are routed to same autotask
- Automation routes messages to assigned operators
  - Ensures that messages are processed in order of arrival
  - Enables concurrent processing of messages through the message automation table

### *Message assignment*

To ensure efficient message processing through the NetView automation table, the automation product uses the NetView **ASSIGN** command. Unsolicited messages that are generated by automated resources are assigned to various automation operators. When unsolicited messages are received, they are routed to the automation operators to which they are assigned. Each automation operator processes its assigned messages through the automation table. Message assignment ensures that unsolicited messages are processed promptly in the order in which they are received.

For example, all messages that are generated by an automated resource are assigned to the same automation operator for automation table processing. The automation operator processes all start and stop messages from the resources. Therefore, all processing for the resource is done in the sequence in which the messages are received.

Message assignment also allows concurrent message processing for multiple resources. Each AOFWRKnn (automation work) operator routes to itself all messages that are generated by the resources for which it is responsible. When the automation configuration file is loaded, automation resolves the defined associations of a resource jobname with its messages. AOFWRKnn operator can also route any other messages to itself based on the message prefix.

All other messages are ASSIGNED to named automated functions in which automation operators are defined. For example, all IEA\* messages are routed to RECOPER which is an automated function. Automation operator AUTREC is the task that is defined in the automated function.

You can use the DISPAOPS command to see all automation operator definitions. The DISPMSG command displays message assignments.

## Notify operators

System Automation z/OS generates messages that can be routed to groups of operators

- Logged-on users can be identified in the policy and assigned to receive one or more classes of messages
- Messages can be any of the following types:
  - Routed from the automation agent or automation manager
  - Routed from the automation table, by using the FWDMMSG command
  - Assigned by class
  - Held on the NetView screen of the user
- Notify operators provide a simple alerting mechanism
  - Can be added, deleted, or modified dynamically
- When Sysops is defined as notify operator, then many SA internal messages are sent to the MCS console

### Notify operators

**Notify operators** are users who are defined to receive automation messages. Each message can be associated with 1 - 10 message classes.

A notify operator receives messages of one or more specific classes. These messages are displayed on the operator screen. Optionally, these messages can also be held on the operator screen. For example, one notify operator can be defined to receive all IMS related messages, whereas another might receive all Intervention required messages. In that case, each operator is defined to receive different classes of messages.

A notify operator can receive messages from the automation manager, automation agent, or directly from the automation table (if sent by the FWDMMSG command). Notify operators can be defined in the automation configuration file or set up and changed dynamically after NetView and the automation product have initialized.



**Note:** When Sysops is defined as notify operator, then many automation internal messages are sent to the MCS console

## Message policy

Messages can be defined in MESSAGES/USER DATA policy of:

- Applications (APL), also at the class level
  - Jobname is checked at run time
- MVS Components (MVC) for system messages
- Application groups only support pseudo message INGALERT
- Monitor resources (MTR)
  - Setting health state and actions

### Message policy

Messages can be defined in the MESSAGES/USER DATA policy of applications (APL), both for instances and classes, of MVS Components (MVC) for system messages and of Monitor resources (MTR) for setting health states and actions.

Application groups only support definition of pseudo message INGALERT to define alerting,

For applications the jobname is checked at run time.

# Message policy supports:

- Codes (1 to 3) for filtering and to assign selection
  - Code supports patterns with wildcards
  - Optional AT override to parse message into CODEx variables
- Command or reply response
  - Passes or selection for command or reply response
  - Selection can be a start type or can be set in code policy
  - Passes are reset after shutdown or NetView restart
  - Automated function that the command is to run under
- Mapping to agent status
  - Status commands
  - Extended Status Commands using consumer linked to provider (INGLINK command)
- Severities (only for WTORs and captured messages)
- Controlling where entries are placed within the AT and additional conditions
- Thresholds and automation flags, also for minor resources
  - Can be at minor resource level or at higher level, like application or MVS components
- Pseudo messages: CAPMSGS, INGALERT, prefix UP\_, DN\_, ISUP\_, or ISDN\_ followed by the provider name, CICS ABCODESYSTM, ABCODETRAN...
- AT, MRT and MPF statements are generated but can be overridden

## Message policy supports

**Command or reply responses** can use the command ISSUEACT.

Each command or reply can be assigned to a **pass** or to selection. The selection can be a start type or can be set in code policy. Passes count how often a message is automated and are reset after application shutdown or NetView restart.

Each command or reply can be assigned to a automated function that the command is to run under

## Code Processing

Message policy supports three **codes** for filtering and to assign selection. The code supports patterns with wildcards. A **Value Returned** can be defined when the codes matches. Code match checking is used by automation in several situations including **ISSUEACT**, OMEGAMON exceptions, termination messages, WTORs, INGALERT, captured messages, CICS...

The special value \*IGNORE\* prevents any operation by the command **ISSUEACT**.

An AT override to parse message into CODEx variables is usually required.

Code Processing is also used for OPC Workstation Domains.

A message can be **mapped to an agent status** which in turn can execute **status commands**.

**Extended Status Commands** use consumers linked to a provider, see pseudo messages below and also the INGLINK command. Commands can be issued if two linked or dependent applications reach an “up” or “down” state.

**Severities** can be assigned, but only for WTORs and captured messages.

AT, MRT and MPF statements are generated but can be overridden. Controlling where entries are placed within the AT and additional matching conditions are also supported.

**Thresholds and automation flags** can be defined for minor resources such as messages or statuses. You can do this using the MINOR RESOURCES policy item of an APL or MVC policy object. Definition can be at the minor resource level or at a higher level, like application or MVS components

Automation uses a number of **pseudo messages** to request certain functions:

- CAPMSGS: Defines the captured messages in the code match table
- INGALERT: Defines alert ID, jobname, notification target, and the Value Returned, like event severity, if the alert ID and jobname matches.  
Also supported for application groups
- JOBLOGALL: Job Log Monitoring
- ABCODESYSTM: Processing of CICS abend messages
- ABCODETRAN: Processing of CICS or IMS transaction abend recovery
- ABCODEPROG: Processing of IMS BMP region abends
- ...

#### **Looping Address Space Suppression.**

- INGCATEGORY: category rule for WLM Service Class, address space job type, and jobname
- INGRECOVERY: Recovery Category and pass: warning, diagnostic, stop, quiesce, reset

#### **Extended Status Command Support:**

In order to execute a command by a consumer APL whenever a provider APL becomes UP or DOWN, use:

- UP\_provider-subs - to hold the commands that are executed when *provider-subs* becomes UP
- DN\_provider-subs - to hold the commands that are executed when *provider-subs* becomes DOWN
- ISUP\_provider-subs - to hold the commands that are executed when *provider-subs* is UP
- ISDN\_provider-subs - to hold the commands that are executed when *provider-subs* is DOWN

## Message capturing

- Policy defines **captured messages** and their severity:
  - IGNORE and NORMAL
  - UNUSUAL, IMPORTANT, and CRITICAL are exceptional messages
- Captured messages can be viewed in the context of the automation resource that issued the message
  - For applications (APL), use the DISPINFO command
  - For monitor resources (MTR), use the DISPMTR command
  - For an MVS component (MVC), use the DISPSYS command
  - Exceptional messages can be displayed or deleted using the INGMSG command
- Retained until they are deleted automatically or by the operator
- Captured Messages Limit and Exceptional Messages Limit
- Inform List field allows to alert

### Message capturing

Automation is largely message driven. Messages that pass all the filters (MPFLSTxx, NetView Message Revision Table) end up in the NetView automation table containing action rules. However, in certain situations this is not sufficient.

Instead, it is necessary that the messages that represent such situations are retained and presented to the operations team. The messages must be retained until the situation is resolved, in which case they can be deleted automatically or manually.

### Captured message definition

The AT Status Specification panel allows a captured message specification that should only be used when messages are not captured by default. Pseudo message ID CAPMSGS defines the severity.

## Messages Captured by Default

Automation also captures certain messages by default. This happens when:

- A threshold (infrequent, frequent, or critical) has been reached for a command, a subsystem, or a minor resource
- Commands fail with a return code > 0 and return code checking is turned on
- Subsystem automation status changes
- WTORs are retained for reply processing
- CICS transactions are recovered and for short-on-storage conditions
- IMS transaction or program abnormally ends
- DB2 recovery takes place

## View the messages

Automation allows you to ***capture messages*** and view the messages in the context of the automation resource that issued the message. For applications (APL), the captured messages are shown by the DISPINFO command. Captured messages associated with monitor resources (MTR) are shown by the DISPMTR command, and messages captured for an MVS component (MVC) are shown by the DISPSYS command.

### ***Exceptional messages***

Messages with a severity of UNUSUAL, IMPORTANT, or CRITICAL are referred to as ***exceptional messages***. Such messages are also (in addition to the commands mentioned above) displayed by the INGMSG command. The INGMSG command displays all important (that is, exceptional) messages that currently exist for a given system. The command is also used to delete exceptional messages.

Exceptional messages are retained until they are deleted automatically or explicitly by the operator.

## Alarm Captured Messages

If you want to present captured messages not only on the NetView panels displayed by the commands described above, you can use the Inform List field in the policy to specify one or more targets that the message can be relayed to for alarming purposes.

## Message limits

Automation allows you to limit the number of messages that should be captured on an individual automation resource level (APL, MTR), or by using MVS component defaults (MDF), system defaults (SDF), or application defaults (ADF). Exceptional messages have their own limit that can be specified by setting the Exceptional Messages Limit field in the system defaults (SDF).

## Finding captured messages

Captured messages  
messages can be seen in  
DISPINFO or DISPSYS

Policy uses pseudo message  
CAPMSGS

Code 1 is message ID  
Code 2 is Jobname  
Code 3 is user defined

CAPMSGS	Severity definition for captured msgs			
	Code 1	Code 2	Code 3	Value Returned
DFHIR3785	*			CRITICAL
DFHLG0507	*			CRITICAL
DFHLG0730	*			CRITICAL
DFHLG0731	*			CRITICAL
DFHLG0734	*			CRITICAL
DFHLG0735	*			CRITICAL
DFHME0116	*			IMPORTANT
DFHTS1310	*			CRITICAL
DFHTS1311	*			CRITICAL
DFHZC3482	*			CRITICAL
DFHZC5966	*			CRITICAL
DFHSM0102	*			IMPORTANT
DFHSI1502I	*			NORMAL

Value returned is severity

Display in DISPINFO,  
INGMSG  
Inform List determines target

### *Finding captured messages*

For applications (APL), the captured messages are shown by the DISPINFO command. Captured messages associated with monitor resources (MTR) are shown by the DISPMTR command, and messages captured for an MVS component (MVC) are shown by the DISPSYS command.

Pseudo message ID CAPMSGS defines the severity.

- Code 1 is message ID
- Code 2 is Jobname
- Code 3 is user defined

Value returned is severity

## Runmodes

- Named automation elements Runmode and Runtoken
- Loosely termed desired-availability tags
- Runmode can be set:
  - At initialization time as a reply to message AOF603D
  - With INGRUN command
- An active runmode causes the following actions:
  - Generation of stop request with SOURCE=INGRUN (DEFAULT) and priority “should be down” against sysname/SYG/sysname resource
  - STOP votes are propagated to all resources of that system and along active relationships to other systems
  - When a resource is (forced) qualified for a runmode, the stop vote is still visible but shown as not winning
- The runmode, \*ALL indicates no active runmodes; all resources in the SYG are unaffected

© Copyright IBM Corporation 2019

2-77

### Runmodes

Runmodes are a flexible way to control the availability of resources without the need to place explicit START or STOP requests against them or manipulate their automation flag. Runmode requests can be used to switch between different setup scenarios such as day shift versus night shift or weekday versus weekend. You can also use runmodes to IPL a system in a progressive fashion. You define and activate runmodes at the system level.

A runmode consists of named automation elements runmode and runtoken that are defined in the automation policy database. A runmode can be loosely termed a *desired-availability tag*. A runmode has one or more runtokens (character strings that are associated with runmodes) that are associated with it. These character string identifiers are called runtokens and are used to link runmodes to resources on a system.

By assigning the runtoken to a resource you request that a resource qualifies for the runmodes where that runtoken is listed. The same runtoken can be listed in one or more runmodes and one or more applications.

The goal of runmodes is to make qualified resources available and unqualified resources unavailable. A resource is considered qualified for one of the three following reasons:

- The qualification is forced for the resource using the INGRUN command.

- The resource is a member of a group that is forced to qualify.
- At least one of its runtokens matches a runtoken defined for the current runmode.

When a runmode is activated on a system either by a reply to message AOF603D or by means of the INGRUN command, an implicit stop request is generated against the system group (SYG) for that system. STOP votes are then propagated to all resources of that system.

When a resource is (forced) qualified for a runmode, the stop votes are still visible but shown as not winning or as being propagated if it is the highest priority stop vote. The stop vote is propagated along active relationships to other systems if the resource has a relationship to resources in other systems. So, even though a runmode is set for a system, it can affect other systems that have no or different runmodes. The stop vote carries the runtokens.

The runmode, \*ALL indicates no active runmodes; all resources in the SYG are unaffected.

Resources are then made available (started) if they are linked to a runtoken that is associated with the active runmode. If a resource has no runtoken and one of the runtokens that is specified in the runmode definition is **\*NULL**, the resource is started.

## Runmode scenarios

- Starting your system in stages

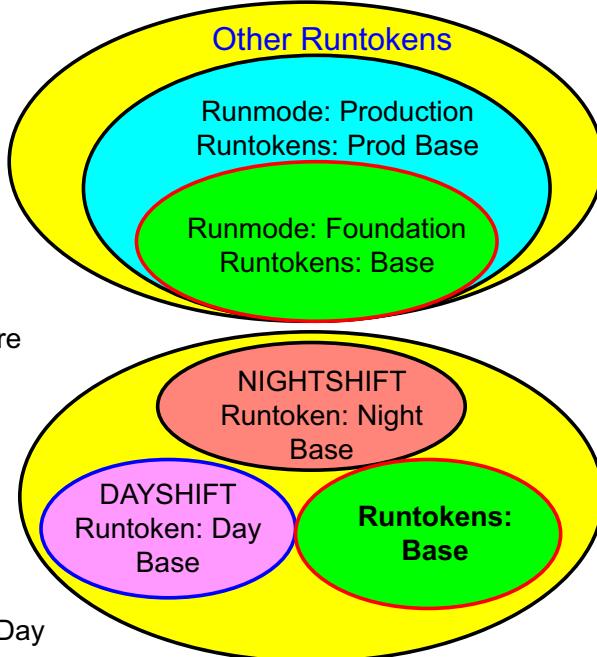
- Start base infrastructure  
RUNMODE Foundation: Base
- Start Production  
RUNMODE Production: Prod Base
- Start complete system  
RUNMODE \*ALL

- Stopping your system in stages

- Stop all but Production and base infrastructure  
RUNMODE Production: Prod Base
- Stop Production but not base infrastructure  
RUNMODE Foundation: Base
- Stop complete system

- Switch between two setups

- Day shift RUNMODE DAYSHIFT: Base and Day
- Night shift RUNMODE NIGHTSHIFT: Base and Night



© Copyright IBM Corporation 2019

2-78

### Runmode scenarios

You can use runmodes to start your system in stages. This slide shows an example in which the foundation or base infrastructure uses one runtoken: *Base*.

In addition to the runtoken *Base* the *Prod* runtoken is used by RUNMODE Production.

- **Base** qualifies the base infrastructure resources for the system.
- **Prod** qualifies the set of production resources.

Switch between two or more scenarios with the same configuration. For instance, you can switch between Nightshift and Dayshift, or Normal and disaster recovery.

For example, a **DR** runmode could qualify resources that are needed during disaster recovery testing. Starting the system in stages creates checkpoints at IPL. The same can be done during shutdown.

## Characteristics of runmodes

- Linked to systems in a SAplex
- Displayed at run time with the DISPSYS command
- Only one can be active on a system
- They are persistent; HOT start required
- Configuration file changes, or refresh does not affect them
  - Active runmodes are shown on INGAMS panel
- Easy to override with a start or stop request
- Each has one or more runtokens (character strings that are associated with runmodes)

Runmode	Runtokens
Foundation	Base *NULL
Production	Prod Base *NULL

- Runtokens are used to link runmodes to resources

### Characteristics of runmodes

Runmodes are defined in the policy database and are linked to systems in the SAplex. At run time, use the DISPSYS command to display the valid runmodes that are linked to a system.

Only one runmode can be active on a system. Runmodes are persistent. They survive agent recycles and IPLs with automation manager start-type of HOT. Runmodes are not changed when a new configuration file is loaded, or the automation control file is refreshed. If a configuration file with defined runmodes is loaded and the active runmode is missing from the file, System Automation for z/OS generates a notification WTOR. On the INGAMS panel, you can press PF11 to see the active runmode.

A runmode can have a single runtoken, or a list of runtokens. Runtokens are the ID strings that tell the automation manager which runmodes are linked to which resources. A \*NULL runtoken allows resources without runtokens to qualify.

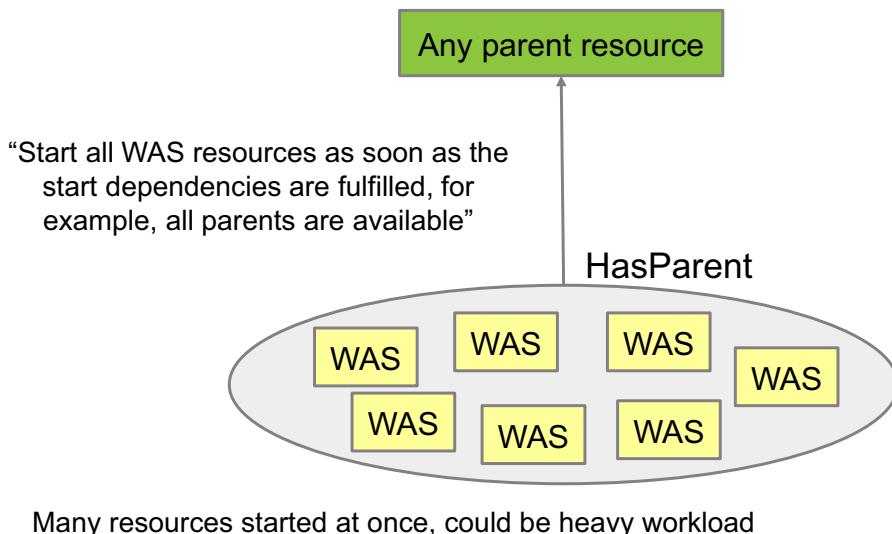
The QUAL column on the INGLIST panel shows the qualification for resources with these values:

- **T:** Resource qualifies because of its runtokens
- **G:** Resource qualifies because it is in one or more groups that qualify
- **R:** Resource qualifies because it was explicitly added with the INGRUN command

## Application Pacing motivation

- Avoid CPU peaks when starting or stopping bulk applications
- Allow resource start to be delayed (e.g. during IPL) to avoid CPU consumption peaks, start only a certain number of applications at a time

**System Automation manages dependencies between resources**



© Copyright IBM Corporation 2019

2-80

### Pacing gates motivation

The resource consumption of applications (that is, of type APL) during the start and stop phases can vary a lot. While many applications are started or stopped very quickly, other applications may consume a lot of CPU resources and may even tend to dominate how the system assigns CPU resources among the started tasks in need.

To facilitate the management of those applications aiming for efficient resource utilization while avoiding customization efforts, the application pacing capability can be used to control how many applications of a kind can be started or stopped at the same time. For this, any application that is eligible to receive a start or stop order from the automation manager has to 'transit' through a defined Pacing Gate.

If the number of applications that are currently in transition reaches the maximum concurrency level, additional applications are held back (waiting) until another application finished this transition and reached the final desired status or terminated during that transition.

In order to use application pacing, you need to define one or more Pacing Gate entries using the Customization Dialog, option 13 (PAC) and specify the concurrency level of starting and stopping applications. Next, you assign those applications that you want to more tightly control to a Pacing Gate. This is done by selecting a pacing gate using the PACING GATE policy.

## Application Pacing – how does it work?

- Application Pacing allows you to control rate and pace for start and stop of CPU intensive work
- SA z/OS Pacing Gates allow you to determine how many resources are started or stopped at the same time without introducing artificial groups and dependencies between groups or without requiring defining triggers that are set upon custom timers



2-81

### Application Pacing – how does it work?

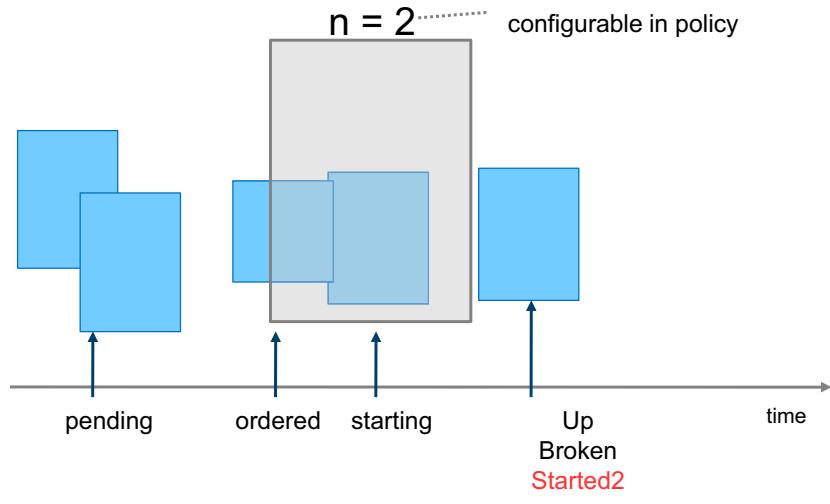
Application Pacing allows you to control rate and pace for start and stop of CPU intensive work

SA z/OS Pacing Gates allow you to determine how many resources are started or stopped at the same time without introducing artificial groups and dependencies between groups or without requiring defining triggers that are set upon custom timers

## Pacing Gate approach...

"Gate mechanism ensures that only up to  $n$  automation resources can pass the gate at any given time"

### Pacing Gate side-view



© Copyright IBM Corporation 2019

2-82

### Pacing Gate policy

Gate mechanism ensures that only up to  $n$  automation resources can pass the gate at any given time. The automation manager delays the order of the applications waiting in front of the gate. When the gate is open a start order is sent the agent and the status becomes starting.

The gate will be opened again when the startup is done successfully, indicated by an UP status or unsuccessfully indicated by a STARTED2 or BROKEN agent status.

## Pacing Gate policy

- Pacing Gate entry type defining
  - Start Concurrency Limit
  - Stop Concurrency Limit
- Link application to Pacing Gate (only one)
- Command INGPAC
  - The INGPAC command displays information about pacing gates, their characteristics and runtime statistics.
  - It also shows details about pacing gate contention with the resources currently granted access to the gate and those waiting for access to the gate.

*Pacing Gate approach...*

In order to use application pacing, you need to define one or more Pacing Gate entries using the Customization Dialog, option 13 (PAC) and specify the concurrency level of starting and stopping applications.

Next, you assign those applications that you want to more tightly control to a Pacing Gate. This is done by selecting a pacing gate using the PACING GATE policy. Note, each application can only be associated with one single Pacing Gate at a time.

Typically, application start-up and shutdown times can be minimized if you allow for maximum parallelism that the system can handle. Therefore, it is not recommended to use Pacing Gates as a general vehicle to control all applications.

Using them for certain CPU intensive workloads, however, can help to decrease system IPL, shutdown times or both, as Pacing Gates can prevent too many workloads dominating the CPU resources during these times.

To see how Pacing Gates affect the applications associated with them, invoke the INGPAC command from the NetView console. INGPAC shows a list of Pacing Gates based on the selection criteria specified by the operator. It also allows you to display all applications that are currently either in transition or waiting for transition through the Pacing Gate.

# Lesson 9 End-to-end automation

## Lesson 9: End-to-end automation (SA z/OS only)

- End-to-end automation architecture
- Reference resources and remote domains
- Cross sysplex automation
- End-to-end adapter
- End-to-end agent
- Universal Automation Adapter

© Copyright IBM Corporation 2019

2-84

### *End-to-end automation*

With SA z/OS V4.1, the automation manager is able to automate resources cross multiple sysplexes (SAplexes) and resources across platforms (non-z/OS systems). This capability of cross sysplex/platform automation is also called end-to-end automation.

This lesson covers:

- End-to-end automation architecture
- Reference resources and remote domains
- Cross sysplex automation
- End-to-end adapter
- End-to-end agent
- Universal Automation Adapter

## End-to-end automation functions

- Start and stop remote resources
- Monitor remote resources
- Manage cross domain dependencies between local and remote resources or between two remote resources
- Compose business applications that have high availability across multiple automation domains
- Manage a group of remote resources

### *End-to-end automation functions*

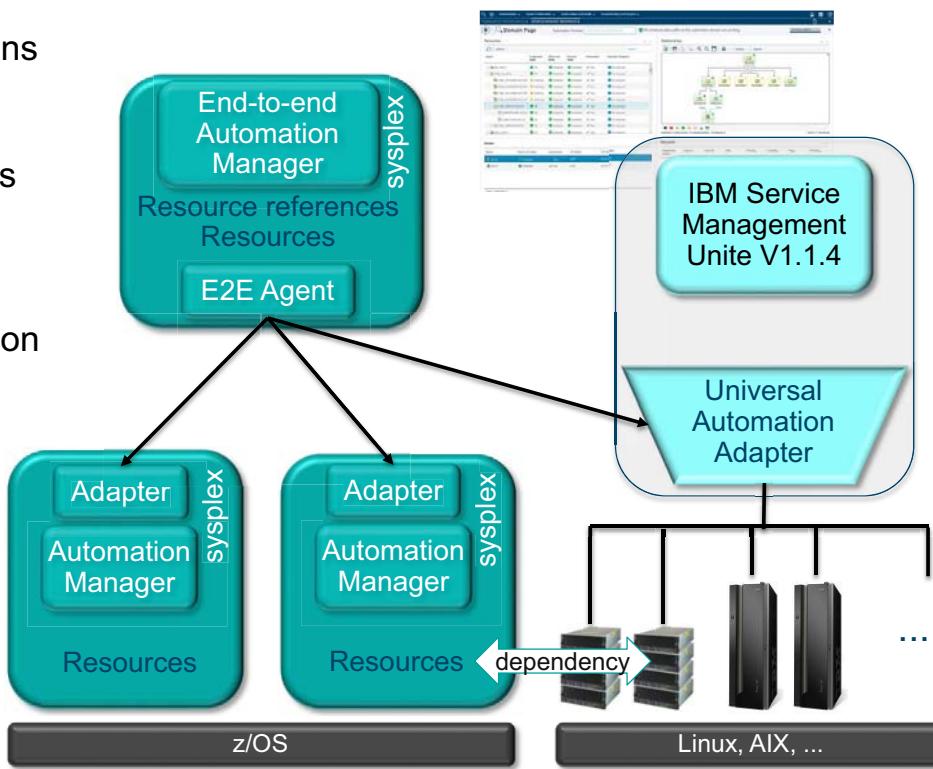
With end-to-end automation, you can

- Start and stop remote resources
- Monitor remote resources
- Manage cross domain dependencies between local and remote resources or between two remote resources
- Compose business applications that have high availability across multiple automation domains
- Manage a group of remote resources

# End-to-end automation (E2E)

Control applications running on z/OS sysplexes and on distributed servers

Manage cross platform automation dependencies



© Copyright IBM Corporation 2019

1-86

## End-to-end automation architecture

Before using end-to-end (E2E) automation, you must have activated it. If activated, the primary automation manager (PAM) also becomes the **E2E manager** and automates the resources in other SAplexes or distributed servers.

Originally, you define a relationship between resources that all run in the same SAplex, for example, hasParent relationship between a child and its parent. With SA z/OS 4.1, you can define a cross-sysplex relationship to resources outside of the local SAplex. Along with APAR OA55386, you can also define a cross platform relationship to resources on remote non-z/OS systems.

The **automation adapters** and **Universal Automation Adapter** send events and receive synchronous requests for execution from the E2E manager.

This allows you to:

- Control applications running on z/OS sysplexes and on distributed servers
- Manage cross platform automation dependencies

# End-to-end automation concepts

- **E2E manager**
  - Manages dependencies between resources in different automation domains
  - The PAM becomes the end-to-end automation manager
  - Does not replace the local automation products on a remote domain
  - Requests to start or stop resource references that point to remote resources
- **E2E agent**
  - Executes start/stop orders
  - Collects status of resources running on remote automation domains
  - Interfaces with one or multiple remote automation adapters
- **Service Management Unite (SMU) Server**
  - Connects to E2E automation adapters to provide SA dashboards
  - Connects to local Universal Automation Adapter to manage remote non z/OS resources
  - Provides policy with start, stop, and monitor commands for non z/OS resources
- **E2E automation adapter**
  - Connects an SA z/OS automation domain with the SMU server or with the E2E agent
- **Universal Automation Adapter**
  - Connects an automation domain with the SMU server or with the E2E agent
  - Connects to the remote non-z/OS systems using Secure Shell (SSH)
  - Invokes start or stop commands for remote resources on behalf of E2E automation manager

## End-to-end automation concepts

- **E2E manager**
  - Manages dependencies between resources in different automation domains
  - The PAM becomes the end-to-end automation manager
  - Does not replace the local automation products on a remote domain
  - Requests to start or stop resource references that point to remote resources

- **E2E agent**

In conjunction with the E2E manager, there is a new address space, which represents the E2E agent. The E2E agent assists the E2E manager by

- Executing start/stop orders
- Collecting status of resources running on remote domains
- Interfacing with one or multiple remote automation adapters

- **Service Management Unite (SMU) Server**

- Connects to E2E automation adapters to provide SA dashboards
- Connects to local Universal Automation Adapter to manage remote non z/OS resources

- Provides policy with start, stop, and monitor commands for non z/OS resources
- **E2E automation adapter**
  - Connects a remote domain with the SMU server or with the E2E agent
  - Monitors status of resources within the automation domain
  - Sends resource status change events
  - Starts and stops resources within the automation domain
  - Provides information about resources in response to queries
- **Universal Automation Adapter**
  - Connects a remote domain with the SMU server or with the E2E agent
  - Connects to the remote non-z/OS systems using Secure Shell (SSH)
  - Invokes start or stop commands for remote resources on behalf of the E2E automation manager

## End-to-end automation concepts (cont.)

- Remote resource
  - Is a real resource on a remote automation domain
  - Managed either by SA z/OS E2E automation adapter or by the Universal Automation Adapter
- Resource reference
  - Resource of type REF that references a real resource on a remote automation domain
  - The real resource is either on a remote SAplex and managed by SA z/OS E2E automation adapter, or on a non-z/OS domain and managed by the Universal Automation Adapter
  - Can be added to groups (APG) and used for dependencies
  - Displayed in INGLIST, INGINFO...
- Automation domain (DMN)
  - SAplex connected to the E2E automation agent by the E2E automation adapter
  - Non-z/OS domain connected to the E2E automation agent by the Universal Automation Adapter
  - Displayed in INGLIST, INGINFO...

SA z/OS - Command Dialogs										Line 1 of 4	
Domain Id . . .		operator id . . .		domain name		Sysplex = AOC4PLEX		INGLIST		Date . . . : 07/25/18	
A Update	B	H DISPTRG	I	J INGSCHED	K INGGROUP	L INGCICS	M INGIMIS	N INGINFO	O INGPAC	Time . . . : 09:44:45	
R Resume	S Suspend	T INGTWS	S	P User	Q Compound	R Sus	S Desired	T scroll	U	V	W
CMD Name	Type	System	Sus	Desired	Compound	Sus	Desired	scroll	Observed	Nature	
LINUXENV	DMN	REF	LINUXENV	SATISFACTORY	AVAILABLE		SATISFACTORY	AVAILABLE	AVAILABLE		
SMU	REF	LINUXENV	REF	SATISFACTORY	AVAILABLE		SATISFACTORY	AVAILABLE	AVAILABLE		
VNC	REF	LINUXENV	REF	SATISFACTORY	AVAILABLE		SATISFACTORY	AVAILABLE	AVAILABLE		
WEB_SRV				PROBLEM	AVAILABLE				HARDDOWN		

reference resource name
domain name where REF resource belongs to

© Copyright IBM Corporation 2019

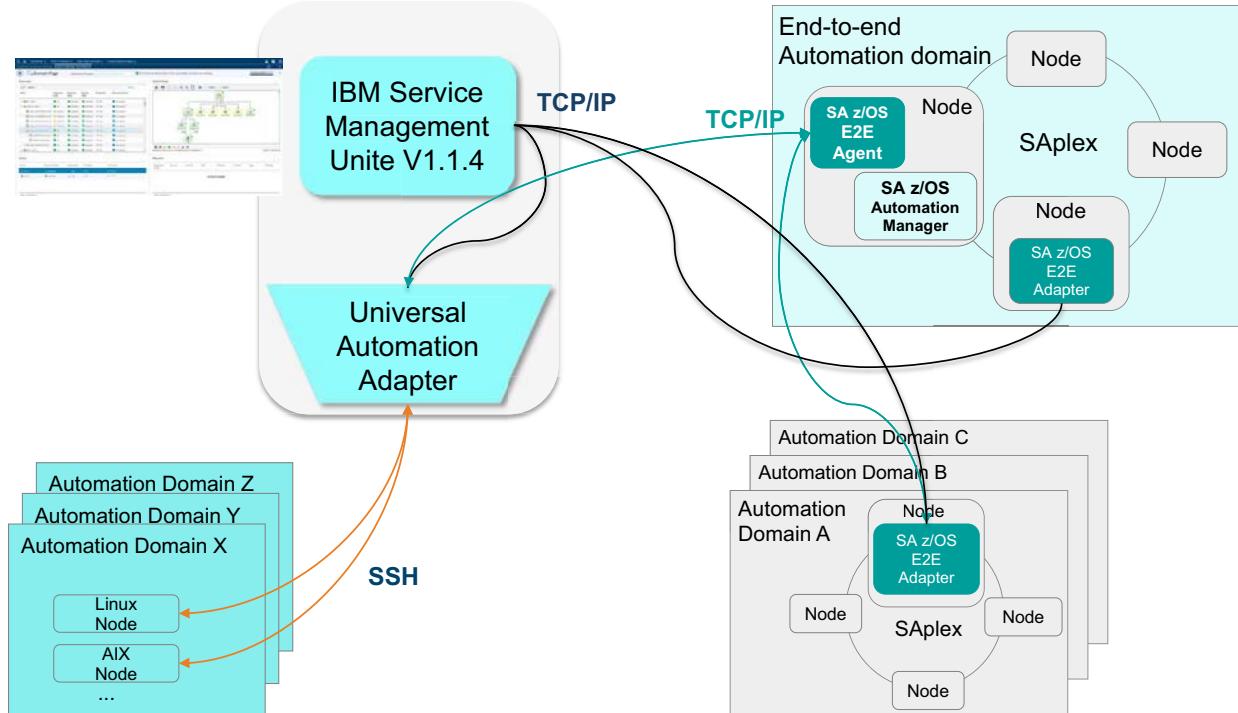
2-88

### End-to-end automation concepts

- **Remote Resource**
  - Is a real resource on a remote domain
  - Managed either by the remote SA z/OS or by the end-to-end automation manager using the Universal Automation Adapter
- **Resource Reference**
  - Resource of type REF that references a real resource on a remote domain
  - Can be added to groups (APG) and used for dependencies
  - The status of the resource reference represents the status of the remote resource
  - Displayed in INGLIST, INGINFO
  - Manageable in INGREQ, INGSET, INGSCHED...
- **Remote Domain (DMN)**
  - SAplex connected to the E2E automation agent by the E2E automation adapter
  - Non-z/OS domain connected to the E2E automation agent by the Universal Automation Adapter

- Displayed in INGLIST, INGINFO...

# End-to-end automation architecture details



© Copyright IBM Corporation 2019

2-89

## End-to-end automation architecture details

Each SA z/OS automation domain represents an SAplex. It contains one or multiple z/OS systems with System Automation for z/OS and the end-to-end automation adapter. The automation adapter can run on only one system in the SAplex. The system where it runs is automatically selected as the primary agent. This is shown in command INGAMS by YES in column E2E.

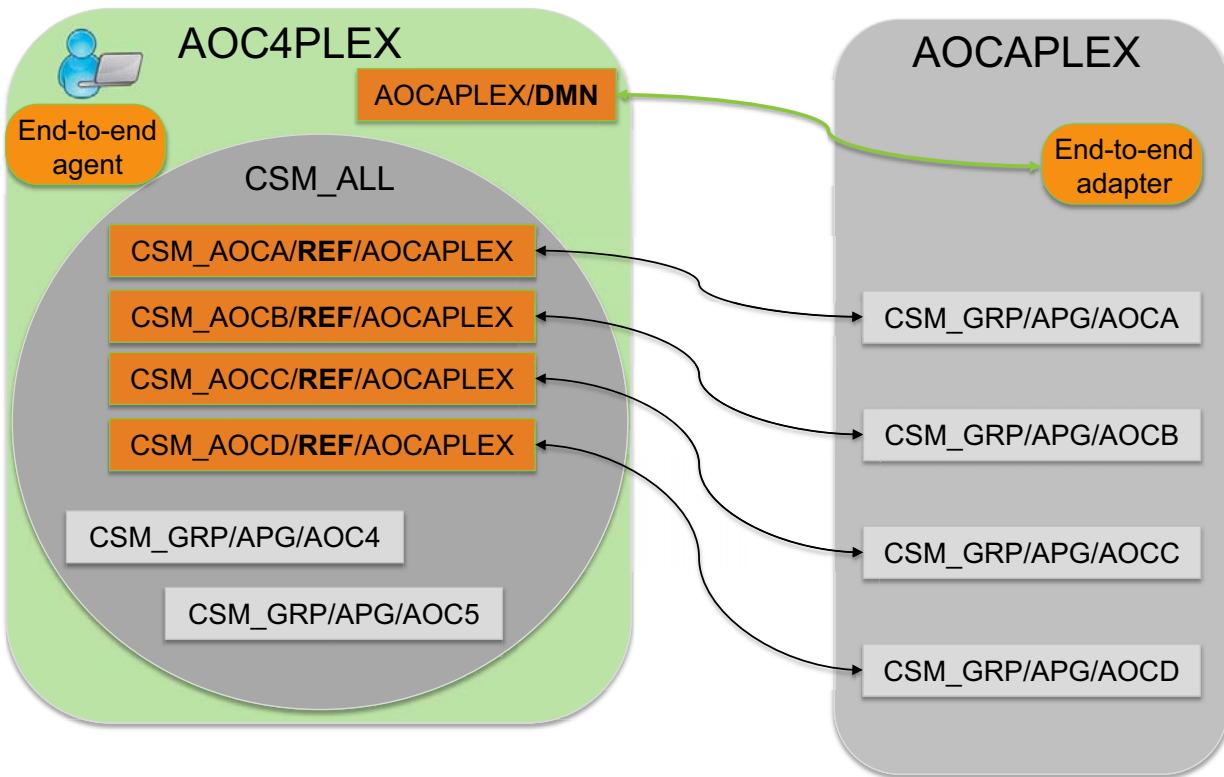
## Configuration and activation

The end-to-end agent can be configured as part of the SA Configuration Assistant. To activate the end-to-end automation, you must start the E2E agent address space. During initialization, the E2E agent loads the REF and DMN objects from the automation policy and registers these resources to the primary automation manager and thereby activates end-to-end automation for the corresponding REF and DMN resources.

## Security

When the end-to-end automation manager issues requests against remote resources, it must authenticate itself to the first-level automation domains that host the remote resources. For authentication, the end-to-end automation manager uses the user credentials (user ID and password) that are specified in the credential file of the E2E agent.

## Cross sysplex automation example



© Copyright IBM Corporation 2019

2-90

### Cross sysplex automation example

In automation domain AOC4PLEX on the left we have reference resources pointing to automation domain AOCAPEX. The reference resources are a member of an application group that also contains members of type APG, application group, of the local SAplex.

In automation domain AOCAPEX you have the real resources of type APG, application group running on different systems AOCA to AOCD.

Automation domain AOCAPEX is defined as DMN resource in automation domain AOC4PLEX which runs the E2E agent.

Automation domain AOCAPEX has to run the E2E adapter.

# End-to-end adapter

Executes requests

Informs about

- Resource state changes
- Domain (SAplex) join/leave
- Policy change

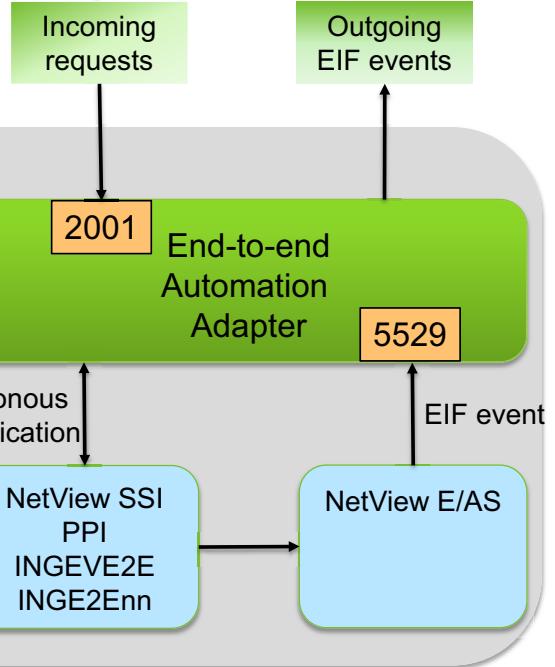
**z/OS system**

SA z/OS Primary Automation Manager (PAM)

**z/OS system**

NetView SA z/OS Agent

Request issuer and event receiver can be WebUI (SMU) and E2E agent



© Copyright IBM Corporation 2019

2-91

*End-to-end adapter*

The successful initialization of the automation adapter on a system makes the System Automation NetView/Agent to be the automation agent that communicates with the PAM, wherever it is. INGAMS command shows this indication.

## Synchronous Communication

The automation adapter receives a request from the SMU server or from the E2E agent and schedules it via PPI communication to an SA z/OS task execution request processor that runs on the automated operator function E2EOPER or E2EOPRnn.

## Asynchronous Communication

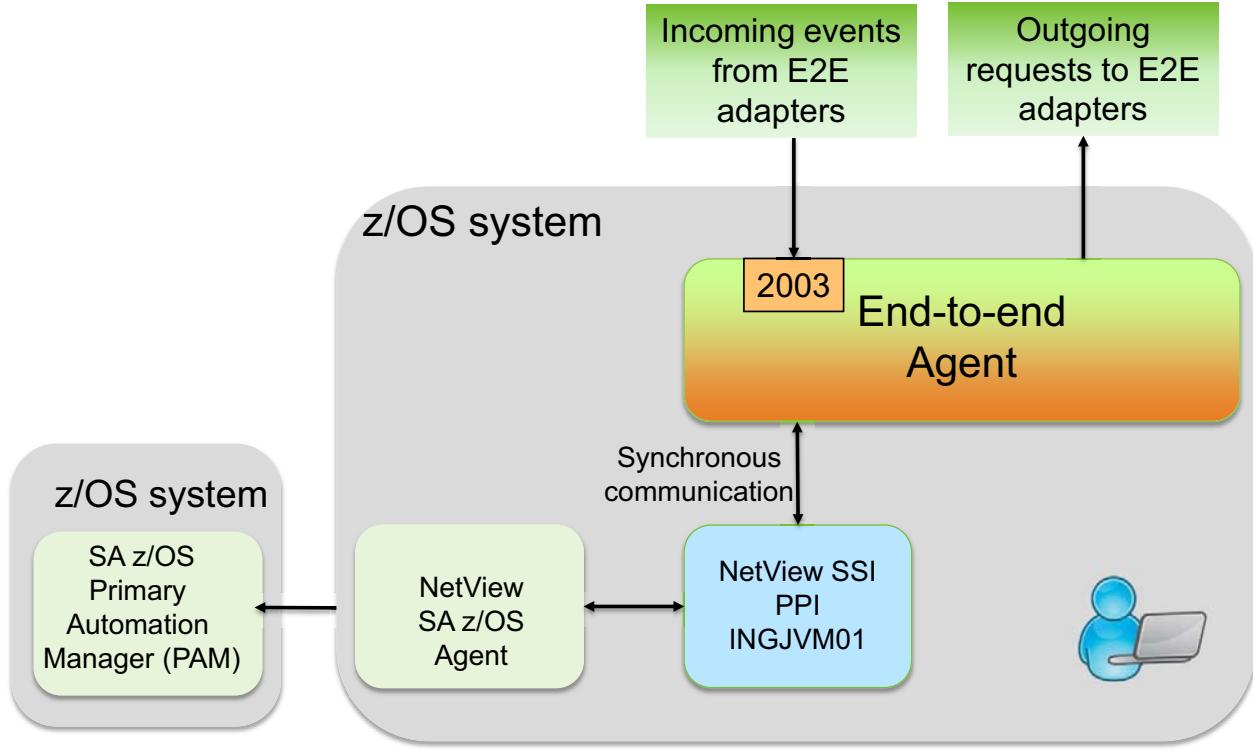
The automation adapter provides an EIF event receiver and an EIF event emitter.

SA z/OS acts as an asynchronous data provider and sends SA z/OS specific events via E/AS to the EIF event receiver of the automation adapter. The automation adapter converts the events to a format understood by SMU server or by the E2E agent. The formatted events are sent to one or both targets through the EIF emitter component.

## The NetView Event/Automation Service (E/AS)

The message adapter service of the NetView event/automation service (E/AS) is used to convert and forward messages from NetView to a designated event server, such as the end-to-end automation adapter. The end-to-end automation adapter requires a separate E/AS address space for its own. INGEVE2E is specified as the PPI receiver name in E/AS initialization member IHSAINIT.

## End-to-end agent



© Copyright IBM Corporation 2019

2-92

### End-to-end agent

End-to-end automation can be connected or disconnected to an automation domain. You connect an automation domain to end-to-end automation when you start the corresponding E2E automation adapter on the remote domain and the E2E agent is able to establish a TCP/IP connection to this E2E automation adapter.

There is a synchronous and asynchronous communication path from the E2E agent to the remote automation adapter. INGJVM01 is the PPI receiver queue name used by the E2E agent.

A check-health algorithm combined with heartbeat events determines periodically the observed status of the DMN. The observed status of the DMN object is AVAILABLE only if synchronous and asynchronous communication states are OK. Otherwise, the observed status is UNKNOWN or HARDDOWN if a problem was detected. It is only at the first time after a cold or warm start of the PAM that the observed status of a DMN is SYSGONE.

The E2E agent needs to be recycled when NetView recycles, when the NetView SSI address space recycles, when the PAM recycles or moves.

## REF resources on SMU

The top screenshot shows the Domain Page for Automation Domain AOC4PLEX INGXSGA4. It lists four reference resources (CSM\_AOCA/REF/AOCAPLEX, CSM\_AOCB/REF/AOCAPLEX, CSM\_AOCC/REF/AOCAPLEX, CSM\_AOCD/REF/AOCAPLEX) pointing to Automation Domain AOCAPLEX. The bottom screenshot shows the Domain Page for Automation Domain AOCAPLEX INGXSGA0 after viewing a remote resource, listing three resources (CSM\_GRP/APG/AOCB, CSM\_GRP/APG/AOCC, CSM\_GRP/APG/AOCD) pointing to Automation Domain AOCAPLEX.

Name	Compound State	Observed State	Desired State	Automated	Operator Request
CSM_AOCA/REF/AOCAPLEX	OK	Available	Available	Yes	No request
CSM_AOCB/REF/AOCAPLEX	Warning	Unknown	Available	Suspended	No request
CSM_AOCC/REF/AOCAPLEX	OK	Available	Available	Yes	No request
CSM_AOCD/REF/AOCAPLEX	Warning	Unknown	Unavailable	Yes	Stop

Name	Compound State	Observed State	Desired State	Automated	Operator Request
CSM_GRP/APG/AOCB	Warning	Unknown	Available	Yes	No request
CSM_GRP/APG/AOCC	OK	Available	Available	Yes	No request
CSM_GRP/APG/AOCD	Warning	Unknown	Available	Yes	No request

© Copyright IBM Corporation 2019

2-93

### REF resources on SMU

You can display REF resources on SMU using the domain widget.

The top screenshot shows automation domain AOC4PLEX. It has reference resources pointing to automation domain AOCAPLEX. You can transfer to automation domain AOCAPLEX by right clicking the REF resource and selecting View Remote Resource (left screenshot).

Then a new tab with the domain widget is opened with the real resource in automation domain AOCAPLEX.

## Summary

---

Now that you completed this unit, you can perform the following tasks:

- Describe the architecture
- Describe the automation agent role and operation
- Describe the automation manager role and operation
- Explain the key automation concepts
- Describe goal driven automation
- Explain the automation statuses and their effect on automation
- Describe automation policy for applications
- Describe resource dependencies and relationships
- Provide an overview of application groups
- Explain automation flags, threshold processing, message policy, and notify operators
- Describe end-to-end automation architecture

### Summary

Now that you completed this unit, you have the prerequisite knowledge for the details unit coming next as well as for the operations and administration classes. Go back to the lessons of this unit whenever you need the automation architecture and concepts.



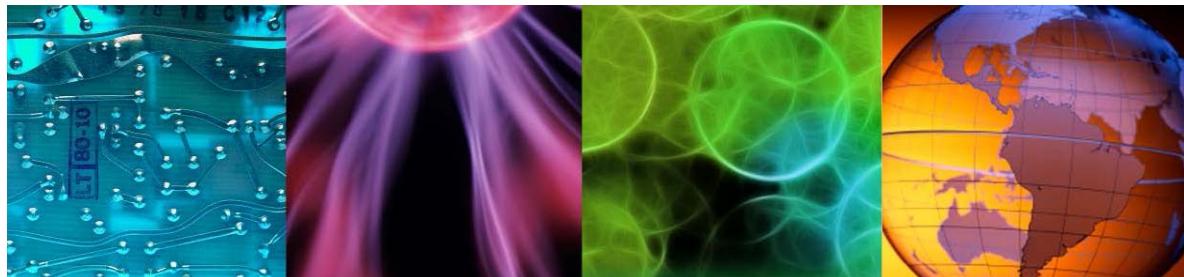


## 3 Details

IBM System Automation for z/OS 4.1



## Unit 3: Details



© Copyright IBM Corporation 2019

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This chapter covers the automation product details and extends the *Architecture and Concepts* unit.

# Objectives

---

When you complete this unit, you can perform the following tasks:

- Describe a monitor resource and its effect on the health status of linked resources
- Describe events and triggers
- List MVS automation
- Describe resource relationships details
- Explain the order process
- List and describe factors that can influence goal-driven automation
- List types and natures of application groups
- Explain behavior or attributes of application groups
- List transient resource automation

## *Objectives*

When you complete this unit, you can perform the following tasks:

- Describe a monitor resource and its effect on the health status of linked resources
- Describe events and triggers
- List MVS automation
- Describe resource relationships details
- Explain the order process
- List and describe factors that can influence goal-driven automation
- List types and natures of application groups
- Explain behavior or attributes of application groups
- List transient resource automation

# Lesson 1 Monitor resources

## Lesson 1: Monitor resources

- Monitor resources overview
- Active and passive monitoring
- Health status

### Monitor resources

This lesson introduces monitor resources, active and passive monitoring using monitor resources, and a health status criteria example.

Monitor resources update the health status of other resources. This health status is propagated and aggregated into the compound status for the monitored resource. The results are shown on INGLIST panels. You can display and manage monitor resources with the DISPMTR command., .

## Monitor resources overview

- Monitor resources are policy objects
  - Monitor performance and health
    - Applications
    - Application groups
  - Provide status gradients between up and down
    - Health status maintained by the automation manager
- They are defined in the automation policy database
  - monitor\_name/MTR/system\_name
    - Example: MTRNETV/MTR/MVSA

### Monitor resources overview

Monitor resources are policy objects that are used to monitor the performance and health of applications and application groups. They are defined in the customization dialogs option 11 - MTR in the policy database.

A monitor resource provides the health status for applications and application groups that are linked to it. The automation manager maintains health status, and supplies status gradients between the up and down states of monitored resources. Health status is displayed on INGLIST and is propagated to the compound status of monitored resources.

A policy object is created for each new monitor resource definition.

Example:

MTRNETV/MTR/MVSA is a monitor resource name

where:

- MTRNETV is the name of the monitor definition
- MTR is the resource type
- MVSA is the system on which the monitor is running

## Monitor resources overview (continued)

- Monitor resources work in these ways:
  - Are started and stopped with the INGREQ command
  - Can be linked to defined service periods
  - Are managed with the DISPMTR command
    - INGLIST row-command M
- Monitoring can be passive or active

### *Monitor resources overview (continued)*

Monitor resources are started and stopped with the INGREQ command, and can be linked to service periods. Functions of the DISPMTR command allow users to view the monitor data and manage the monitor resources. For example, in the case of a bad health status, an operator can use the DISPMTR command to display detailed information about the monitor resource.

A monitor can have several states that correspond to a health status:

- ACTIVE: The monitor is running. Events or monitor routines determine health status.
- INACTIVE: The monitor is not running. The health status is UNKNOWN.
- BROKEN: The monitoring routine failed, and is not capable of running. The health status is UNKNOWN.
- FAILED: The monitoring routine failed, but is rescheduled. The health status is UNKNOWN.

To reset (restart) a BROKEN monitor, you use a row-command on the DISPMTR panel. Monitoring can be active or passive.

# Characteristics of active and passive monitoring

- Active monitoring
  - Status determined by a monitor routine
  - Routine is triggered at a defined interval (Timer)
- Passive monitoring
  - Status based on predefined events driving the INGMON command
  - No monitor routine or interval is defined
- Active monitoring can use events to trigger the INGMON command to update status

## *Characteristics of active and passive monitoring*

An *active* monitor has these characteristics:

- Uses a defined routine to check the condition of resources that are linked to it. The monitor routine is proactive.
- Status is assigned based on return codes from the monitor routine.
- A timer triggers the monitor routine, which runs at a defined interval.

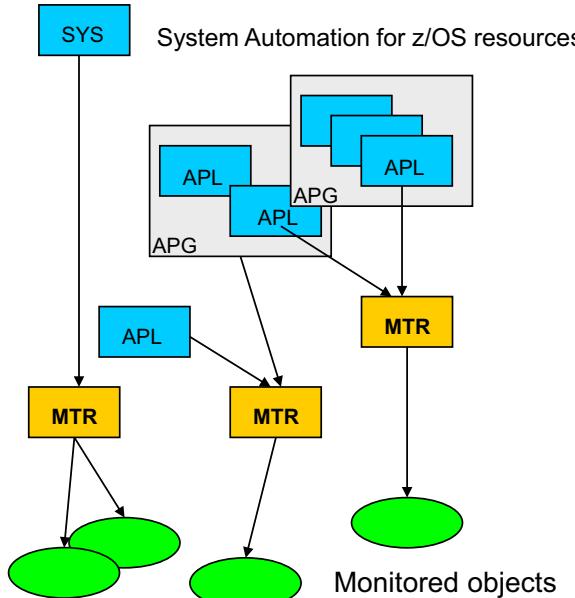
A *passive* monitor has these characteristics:

- Does not use a monitor routine. It is not proactive but reactive.
- The health status that is assigned is based on events. The events and reactions to the events are defined in the MTR policy. The INGMON command is used to set health status.

It is possible to combine both types of monitors. You can define an active monitor to poll for status plus define critical messages to update the status..

# Monitor resources at a glance

- Creates resource in the automation policy
  - monitor\_name / MTR / system\_name
- Determines the health status of the object it monitors
- Is typically associated with an application or application group and defined with HasMonitor relationship
- Shows health status
  - Obtained either periodically or based on an event
  - Propagated to an associated application and application group



© Copyright IBM Corporation 2019

3-7

## *Monitor resources at a glance*

Monitors have a HasMonitor relationship. *Monitored resources* can be applications, application groups, or systems. Monitored resources are linked to the monitors.

## Health status criteria example

- Proactively monitor TCP/IP stack response time on MVSA
- Set health status based on PING round trip time (RTT)

Health status	PING RTT in milliseconds
NORMAL	Less than or equal to 2
WARNING	Less than 4
MINOR	Less than 6
CRITICAL	Less than 8
FATAL	Less than 10
FAILED	10 or greater
BROKEN	Security error or no response

### *Health status criteria example*

This slide shows an example of the design criteria that an active monitor uses to show the health status of a TCP/IP stack. The criteria uses round-trip response time to the PING command

# Lesson 2 Triggers and events

## Lesson 2: Triggers and events

- Event
  - An indication that something happened
  - Essentially a switch (set or unset)
  - Set by user programming, automation table, or the INGEVENT command
- Trigger
  - A set of conditions (events) when, if satisfied, an application can be started or stopped
  - If unsatisfied, the action is delayed until it is satisfied
  - The application must have the appropriate desired status before the start or stop is attempted
  - Managed with the DISPTRG command
- They are a complementary pair

### Triggers and events

An *event* is essentially a binary switch. It has a status of set or unset. Usually, it indicates that something happened, like the successful completion of a batch job. An event can indicate that a resource is in a determined state or that a performance measurement is within a certain range. Programmed actions, including actions specified in automation table statements, set events.

A *trigger* is a combination of several events, or just one event. A trigger is satisfied when all its constituent events are set. A trigger can be defined for starting or stopping an application. If it is defined for starting, and the trigger is not satisfied, then the application is not started. Therefore, a trigger defines a set of conditions that must be true for a linked application to start or stop.

Despite its name, trigger does not mean that an application is started as soon as its start trigger is satisfied. The name relates to its use in earlier releases, when triggers might cause the start or stop of applications. The application must have the appropriate desired status. Only then does the automation manager check the trigger.

## Events

- Defined by the system administrator
- Linked to a trigger
- An event state is kept separately for each resource
- Use the DISPEVTS command to view events
- Use the INGEVENT command to set and unset events
- An automatic reset can be specified  
Unset can be done if the UNSET condition is fulfilled
- The following example shows how to set an event

```
IF MSGID = 'IEF404I'  
&   JOBNAME = 'CICSHK99'  
THEN EXEC (CMD ('INGEVENT EVTCICA9') );
```

Events can be set (or unset) for all resources in the sysplex or a specific resource

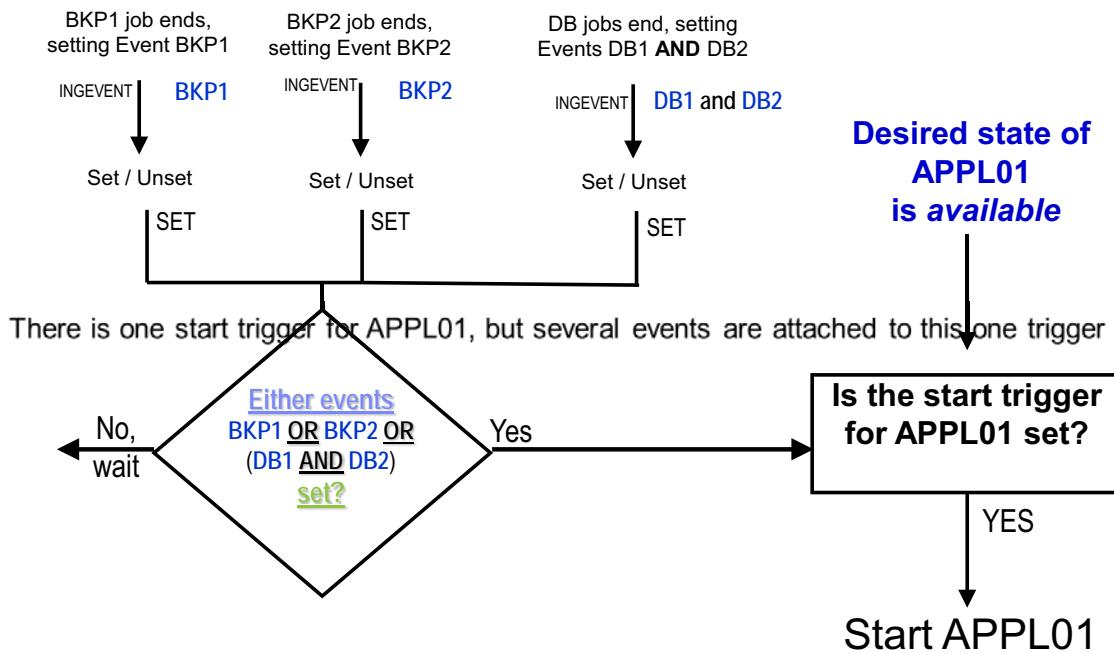
### Events

An event setting is maintained separately for each application that is linked to the event. If the same event is used to start several applications, there is internally a separate switch for each application, with the same name for each. Operators can set and reset events.

The INGEVENT command is used to set the switch to indicate that external events occurred. An internal event setting (for an application) can be used to reset (UNSET) if the linked application goes to, or is found in, certain agent state.

The automation table example shows how the event can be set with a simple message for a job. If the event, EVTCICA9, is used for several applications, then setting it here sets it for all of the applications.

## One trigger and several events example



### One trigger and several events example

In this example, the APPL01 application cannot start before either of its overnight backup jobs (BKP1 and BKP2) or both of its database jobs (DB1 and DB2) complete successfully.

Three events are defined:

- One for the BKP1 backup job
- One for the BKP2 backup job
- One for both database jobs

Only one Start Up trigger is defined for the application, APPL01. If one of the three events is set, then the trigger is satisfied and APPL01 is started. If the desired status of the APPL01 application is AVAILABLE, the start-up for APPL01 is effectively delayed until one of the three events is set. If the batch jobs complete and the desired status is **not** AVAILABLE, the application is not started.

## Unsetting events

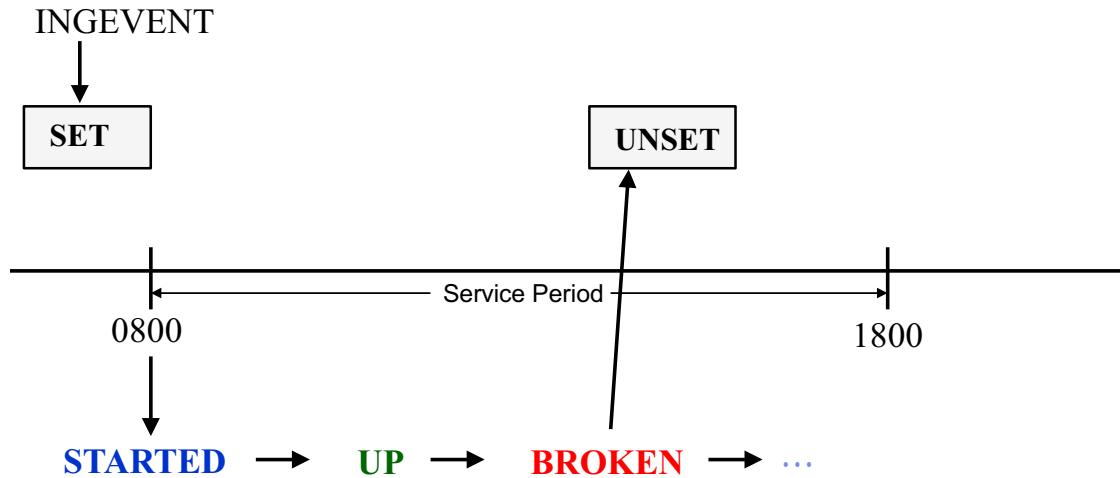
Events that are associated with a specific application can be *unset* automatically when the application is set to, or discovered in, a specific agent status

Unset condition	Agent status
DOWN	AUTODOWN, CTLDOWN, BROKEN, ENDED, RESTART, STOPPED, DOWN
START	STARTED, ACTIVE, EXTSTART
UP	UP, RUNNING
NONE	Done by programming or the INGEVENT command

### Unsetting events

System Automation for z/OS can automatically unset an event when the application that is linked to the event reaches a specific agent status. The slide shows the four possible conditions that can be specified for the automatic unset of an event. The agent status values are in the **Agent status** column. NONE indicates that no automatic unset is done. The unset must be done with a program or the INGEVENT command.

## UNSET=DOWN processing



The resource is not restarted until the following conditions are true:

- The event is set again
- The status is changed to SOFTDOWN

### UNSET=DOWN processing

In this example, INGEVENT sets an event for an application. The application starts at 08:00 when its UP service period begins. The automation manager generates a MakeAvailable request.

During the day, a problem is encountered. The application fails, and its status changes to BROKEN. The failure automatically unsets the event. The application cannot be restarted because the event is not set. When the event is set, an operator must also change the status of the resource before it can be started.

# Lesson 3 MVS and JES automation

## Lesson 3: MVS and JES automation

The MVS Component Entry Type allows to automate:

- z/OS logs:
  - SMF data sets
  - LOGREC data sets
  - SYSLOG data sets
- Dump data sets
- Console buffers
- JES spool (defined in JES application)

### MVS and JES automation

The **MVS Component** Entry Type allows you to apply automation to various z/OS data sets, resources, and facilities, which are referred to as MVS components.

You can set error thresholds for the following MVS components:

- **MVSDUMP** (SYS1.DUMP00–xx)
- **LOGREC** (SYS1.LOGREC)
- **SYSLOG** (SYS1.SYSLOG.INDX)

In addition, you can customize the recovery procedure that is used for a WTOBUF overflow and automate responses to messages that are issued for MVS components.

**JES spool** automation is defined in the JES application.

# Managing z/OS logs, data sets, and console buffers

- Automation is provided when the following situations occur:
  - An SMF data set fills and switches
  - A LOGREC data set is nearly full
  - A SYSLOG data set is spun off to spool
  - DUMP data sets are used
  - Console buffer queues are above the threshold
- Use of policies enable the following actions:
  - Minor resource thresholds to be set
  - Actions to be specified:
    - Might depend on thresholds
    - Might relate to specific data set or console
  - Automation flags to control the level of automation

Examples:

- Aggregate SMF data for a day or a week
- Archive a dump data set

## Managing z/OS logs, data sets, and console buffers

The automation product reacts to messages that z/OS generates, and its own monitoring, to automate some z/OS housekeeping tasks. When defined events are detected, automation initiates actions that are defined in the automation policy.



**Note:** The automation agent, *not the automation manager*, does all z/OS automation. Just as the resource restarts, it is possible to specify thresholds. If the critical threshold is met, the desired action is no longer performed. An error message is created, which can be sent to a *notify operator*. SMF and console buffer queue automation cannot be customized.

# Spool management

- Monitors spool utilization thresholds
- Provides commands to manage spool utilization
- Operates within the Multi-Access Spool (MAS)
- Implemented as MTR resource\$
- Is part of the JES recovery actions  
Controlled by the automation agent RECOVERY flag

## Spool management

Automation reacts to messages from JES2 and its own monitoring to manage spool utilization. Spool management uses actions that are defined in the automation policy to resolve situations of exceeded thresholds. These definitions can also ensure that duplicate actions are not run in other images in a MASplex.

Spool management is internally treated as a RECOVERY action of JES2. The agent RECOVERY automation flag for JES2 controls the actions. The RECOVERY flag is also used for command flooding, write to operator reply (WTOR) buffer shortage recovery, and long-running enqueue automation.

# Lesson 4 Dependency relationships details

## Lesson 4: Dependency relationships details

- Relationships
- When dependencies are checked; before issuing orders
  - MakeAvailable
  - MakeUnavailable
  - PrepareAvailable
  - PrepareUnavailable
- Propagation of votes
  - Active dependencies only
- How dependencies are used by the automation manager
- The scope of dependent relationships
  - Resource dependencies that span the SAplex (SA z/OS only)

### *Dependency relationships details*

Consider revisiting the Relationships lesson of the previous unit.

This slide lists the topics that are covered in the lesson. When requests are received, the automation manager uses dependencies to:

1. Create votes for supporting resources to which there is an active dependency.
2. When the manager is ready to send an order, it checks all dependency relationships for the resource. If any dependencies are not satisfied, the order is not sent until the dependencies are satisfied or overridden.

In System Automation for z/OS, dependency relationships can span the SAplex. That is, a resource can have a dependency on any other resource that is defined in the same SAplex.

# Relationships

- The dependency link between two resources: application, application group, resource reference, or monitor resource
- A relationship establishes the condition of the dependency
- Relationships are conditional actions
- There are several conditional actions
  - HasParent
  - HasPassiveParent
  - MakeAvailable
  - MakeUnavailable
  - PrepAvailable
  - PrepUnavailable
  - Externally
  - HasMonitor
  - PeerOf
  - ForceDown

© Copyright IBM Corporation 2019

3-18

## *Relationships*

A *relationship* is a link between two resources. There are several types of relationships. A relationship normally includes an action (order) for the dependent resource. For example, MakeAvailable creates a request to make a resource available. Its desired status is available. This request implies that the desired status remains available when the resource reaches the desired state (the request is *persistent*).

PeerOf tells automation to propagate MakeAvailable votes from a supporting resource to dependent resources but halt the propagation of MakeUnavailable votes unless ‘Scope=ALL’ is specified on the stop request.

HasParent can be used for simple relationships between two resources. It is really a combination of two dependencies (MakeAvailable/WhenAvailable and MakeUnavailable/WhenUnavailable). ForceDown is the only relationship that generates a vote (MakeUnavailable) by the automation manager and is then withdrawn when the vote is satisfied.

Many different conditions apply to a dependency. A condition indicates a combination of the desired status and the observed status of a supporting resource. The Observed status of a supporting resource tells the automation manager if a dependency is satisfied. For more information, see *System Automation for z/OS Defining Automation Policy*.

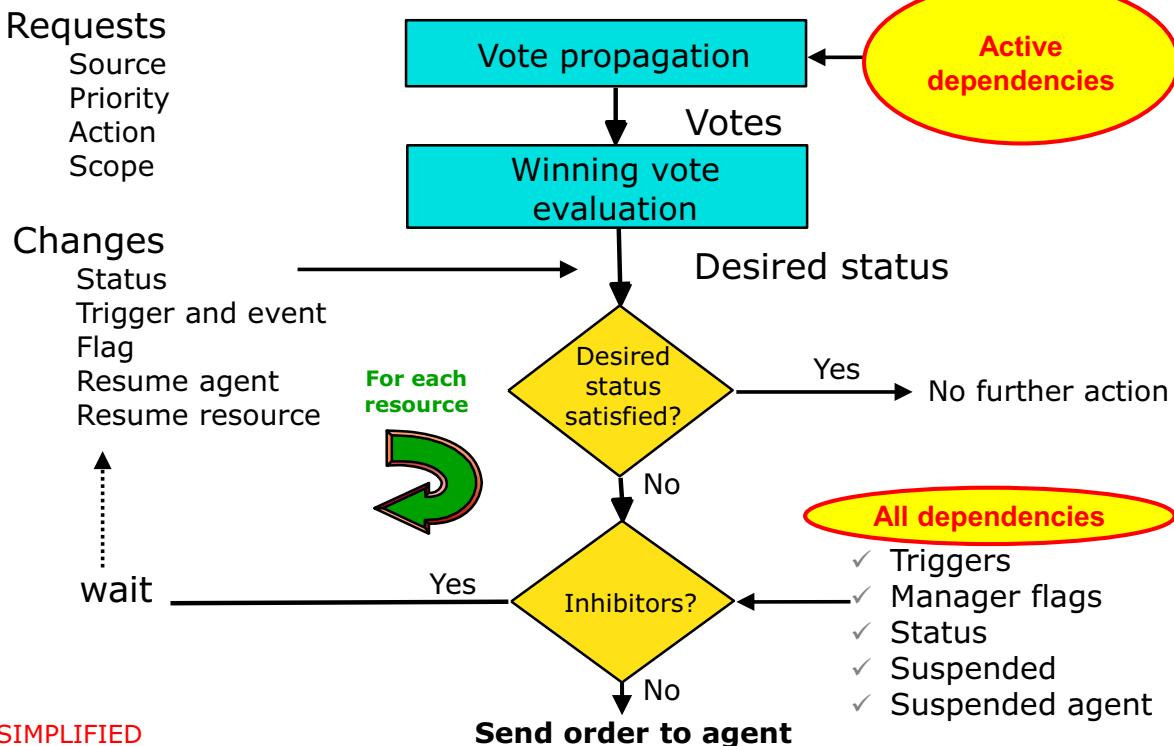
**Conditions:**

WhenAvailable, WhenObservedAvailable, WhenRunning, WhenObservedRunning,  
WhenAvailableOrStarting, WhenRunningOrStarting, WhenSoftDown, WhenObservedSoftDown,  
WhenHardDown, WhenObservedHardDown, WhenDown, WhenObservedDown,  
WhenAssumedDown, WhenDownOrStopping, WhenStoppableOrSoftDown,  
WhenStoppableOrDown, WhenStoppableOrAssumedDown, WhenObservedAssumedDown,  
WhenObservedGroupHasNotFailed

Dependency Examples:

- MakeAvailable/WhenAvailable
- MakeAvailable/WhenDown
- MakeAvailable/WhenStoppableOrDown

# How dependencies affect the request process



© Copyright IBM Corporation 2019

3-19

## How dependencies affect the request process

Votes get propagated over relationships – backwards and forwards. There's no evaluation of the dependencies, just a simple propagation (and sometimes transformation) based on the name of the relationships. A MakeAvailable/WhenAvailable relationship propagates a MakeAvailable vote as a MakeAvailable vote. It doesn't propagate a MakeUnavailable vote at all.

The relationships act more like a filter that transforms or blocks the vote from the dependent resource as it is copied to the supporting resource at the other end of the relationship.

Vote propagation works on 'flood fill' when the vote is worth propagating further propagation continues. The worth of propagating a vote is based on whether it would significantly change anything. If a resource has two MA votes, for example, it only propagates the highest priority one.

For sending orders, the AM uses three basic conditions:

- Should: This condition is driven by the desired status. It addresses whether it should send a start order.
- Can: This condition is driven by dependencies. It addresses whether the dependencies for the action are satisfied.
- May: This condition is driven by the automation and suspend flags (Inhibitors) whether the AM has permission to send the order.

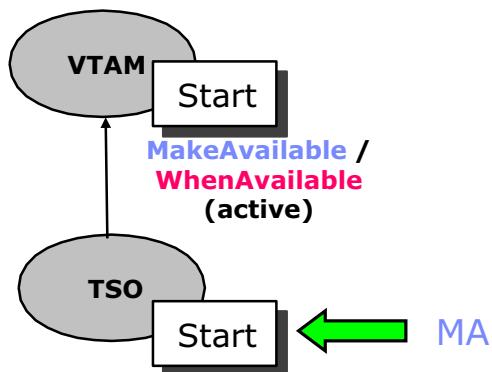


**Note:** At this point in the course, dependencies are explained from the application point of view. Later in the course, the explanations are expanded to include application groups and monitor resources.

# Characteristics of active and passive dependencies

## Active

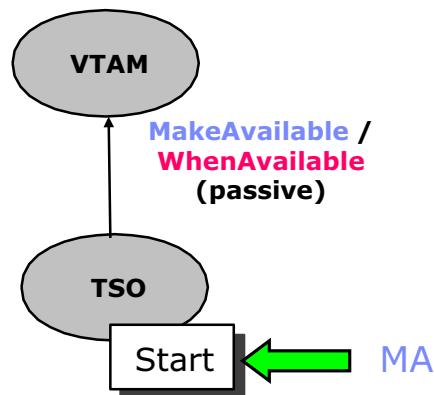
Attempts to fulfill the condition to satisfy the dependency



Start Vote is propagated to VTAM

## Passive

Waits until the condition is fulfilled to satisfy the dependency



Start Vote for TSO only

### Characteristics of active and passive dependencies

When evaluating start or stop requests, the automation manager examines the state of each supporting resource that has an ACTIVE dependency link to the dependent resource. The manager propagates votes to supporting resources to satisfy required conditions.

For PASSIVE dependency links, no votes are propagated to supporting resources. Therefore, a requested action can remain unsatisfied if the supporting resource is not in the required state.

For example, there is an ACTIVE MakeAvailable/WhenAvailable dependency between TSO and VTAM. A MakeAvailable request for TSO causes an internal MakeAvailable request (vote) to be made for VTAM, even if VTAM is available. If VTAM has ACTIVE MakeAvailable dependencies to other resources, votes are propagated to these dependencies also.

If the dependency is PASSIVE, then the request to MakeAvailable TSO does NOT propagate any votes to VTAM, or any of its supporting resources. TSO waits to start until VTAM is available.

Active and passive dependencies apply to MakeAvailable and MakeUnavailable dependencies. If not specified, the default is ACTIVE. (It is not relevant for external relationships, and is forced to PASSIVE for PrepAvailable and PrepUnavailable relationships.) Vote propagation is described in more detail in the next lesson.

Another attribute that influences the process is *chaining*. Chaining identifies which supporting resources are considered (checked) before an order is sent to an agent. There are two types of chaining:

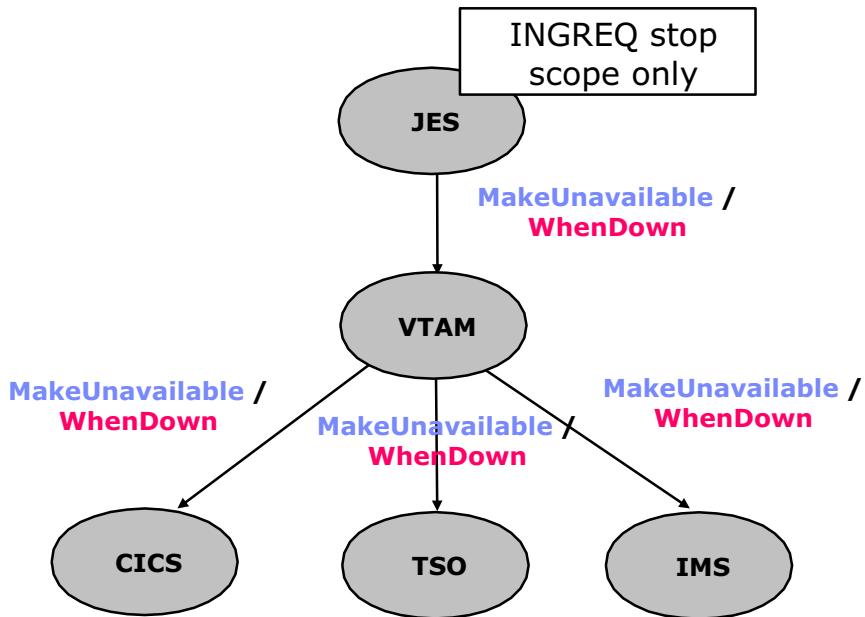
- **Weak:** Specifies that for the relationship, evaluation is done on status of the supporting resource ONLY.
- **Strong:** Specifies that the manager checks all supporting resources on the dependency tree until either a weak relationship or a supporting resource in incorrect status is found.

WEAK is the default value for MakeAvailable and MakeUnavailable relationships. WEAK is assumed for *externally* relationships. The Externally relationship is described later in this unit. Where a weak relationship is found, only its supporting resource is included in the check.



**Note:** Chaining applies only to the checking that is done before a resource is started or stopped. It has no role in the propagation of requests for Active dependencies.

## MakeUnavailable example



© Copyright IBM Corporation 2019

3-21

### MakeUnavailable example

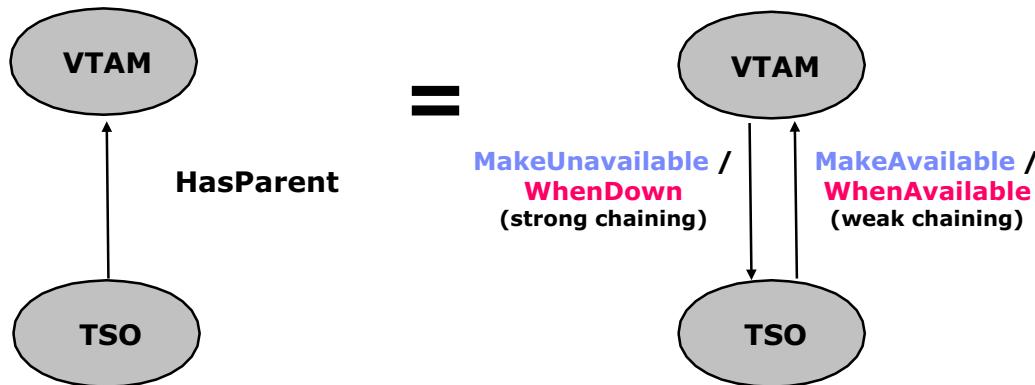
This slide shows a simple set of MakeUnavailable dependencies. JES can be stopped only when VTAM is in a down condition and JES and VTAM have a desired status of UNAVAILABLE. All resources under VTAM with a desired state of Unavailable must be AssumedDown.

Down means a combination of automation manager statuses. The observed status is one of HARDDOWN, SOFTDOWN, OR SYSGONE. The desired status is UNAVAILABLE. Again, VTAM being down is not sufficient to allow JES to be stopped. JES must also have a desired status of UNAVAILABLE.

MakeUnavailable conditions can also be active or passive. An operator can also use the SCOPE parameter (for example, SCOPE=ALL) of the INGREQ command to generate requests.

There is no difference between INGREQ STOP and INGREQ STOP,SCOPE=ALL. There is for SCOPE=CHILDREN and SCOPE=ONLY as the scopes modify the vote propagation process.

## HasParent relationship



HasParent is equivalent to a combination of MakeAvailable/WhenAvailable and MakeUnavailable/WhenDown

### HasParent relationship

The HasParent relationship is equivalent to the two relationships shown (MakeUnavailable/WhenDown plus MakeAvailable/WhenAvailable). A HasParent dependency is really two dependencies:

- For a start request, TSO is the dependent resource and VTAM is the supporting resource.
- For a stop request, VTAM is the dependent resource and TSO is the supporting resource.

MakeAvailable/WhenAvailable means that a dependent resource can be started only when its parents are available. Weak chaining means that the resource requires only its immediate parents to be available.

MakeUnavailable/WhenDown means that when the observed status is SOFTDOWN, HARDDOWN, SYSGONE, or UNKNOWN the resource can be stopped. Strong chaining means that the parent can be stopped only when all of its dependent resources are unavailable. In this scenario, it is only TSO, but if TSO was a parent too, its dependent resource must be unavailable as well.

Strong chaining on a down condition adds an AndStoppable condition to the relationship condition and requires that the supporting resources stop dependency be satisfied.

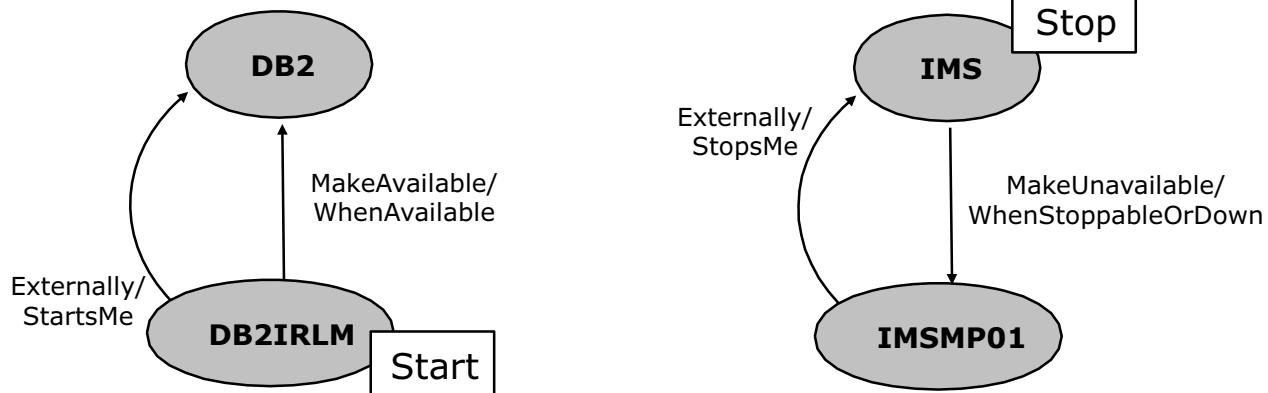
Strong Chaining for an up condition adds AndStartable and requires that the resource start dependency be satisfied.

The chaining conditions can also flip if the relationship is an inverting relationship:

MakeAvailable/WhenAssumedDownAndStoppable

Strong chaining applied to a MakeAvailableWhenDown relationship.

## Examples of external relationships



Externally relationships tell the manager not to send start or stop orders

### Examples of external relationships

These relationships tell the automation manager that an external source starts or stops the resource. The automation manager does not send start or stop orders to the automation agent when the desired status changes. Typically, the externally relationship is used where there are also MakeAvailable or MakeUnavailable dependencies for the resource.

External relationships are used along with the EXTERNAL START and EXTERNAL STOP policy option. You must specify both an option and a matching externally relationship for the external start or stop process to work. The external relationship identifies the resource responsible for performing the external start or stop action.

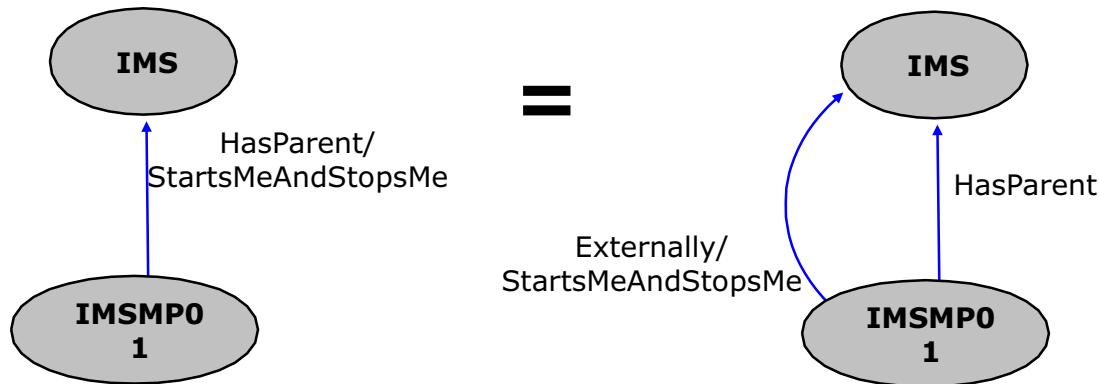
When the conditions under which an external start or stop is expected, the agent is sent an order telling it to expect the action to be performed externally rather than to perform it itself.

The example that is shown on the slide illustrates that DB2 starts application DB2IRLM. If a request to start DB2IRLM is issued, a start vote is propagated to DB2. When DB2 becomes UP (Available), the manager sees that DB2IRLM is down and that subsystem responsible for externally starting it is UP, so an order is sent to the DB2IRLM agent telling it to expect DB2IRLM to be externally started.

Similarly, a case of an IMS message processing region is shown on the right. It expects the IMS control region to stop it. For the external relationship, you can specify one of these conditions:

- StartsMe
- StopsMe
- StartsMeAndStopsMe

# Combining dependency relationships



© Copyright IBM Corporation 2019

3-24

## Combining dependency relationships

You can also combine HasParent and externally relationships. For example, the HasParent/StartsMeAndStopsMe dependency can be thought of as a combination of four dependencies:

- MakeAvailable/WhenAvailable (weak)
- MakeUnavailable/WhenUnavailable (strong)
- Externally/StartsMe
- Externally/StopsMe

# Lesson 5 The order process

## Lesson 5: The order process

- After the desired state change, the order process to the agents is done
  - Automation status set to Ordered
  - The agent sets it to either idle (action complete) or busy (working on it)
- Starting applications
  - Prepare available
  - Make available
- Stopping applications
  - Prepare unavailable
  - Make unavailable
- Actions after an IPL
- External shutdown

### The order process

After the desired state change the order process to the agents is done.

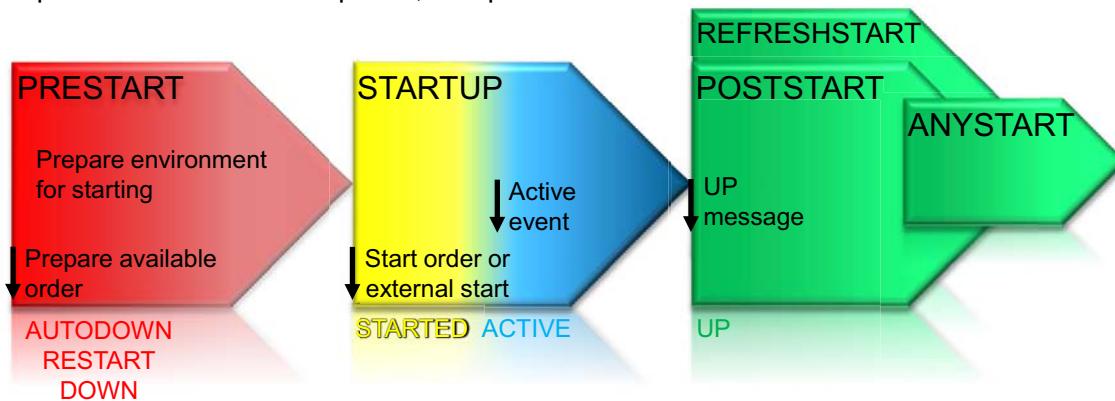
When the manager sends the agent an order, it sets the automation status for the resource to Ordered. The agent is expected to respond by setting it to either idle (action complete) or busy (working on it). If the ordered state isn't acknowledged by the agent within 15 minutes, the manager issues an error message.

This lesson covers orders to start and stop applications as well as Prepare available and Prepare unavailable orders.

Actions after an IPL and External shutdown are also described.

# Starting applications revisited

- A start has three major phases:
  - PRESTART: Prepare environment for starting, if required
  - STARTUP: The actual start commands
  - Optionally after startup is completed:
    - POSTSTART: The actions after the application is UP
    - REFRESHSTART: After a NetView recycle or INGAMS refresh only
    - ANYSTART: Issued after POSTSTART and after REFRESHSTART
- Different start types for each phase are possible, such as COLD, RECOVERY, and so on
- Multiple commands for each phase, if required



© Copyright IBM Corporation 2019

3-26

## Starting applications revisited

Within the automation product, application startup processing occurs in three phases:

1. PRESTART: Before the application is started
2. STARTUP: To start the application
3. POSTSTART: After the application has started

Start types can be defined for each of the three phases. The start type can be an installation-specified value such as COLD for an IPL or RECOVERY to automate the restart of a failed application. The default start type is NORM. Commands in each phase can be specific for a start type. If the desired goal of the resource is MakeAvailable, then the prestart commands for the dependent resources are issued as soon as the specified condition is satisfied for the supporting resource.

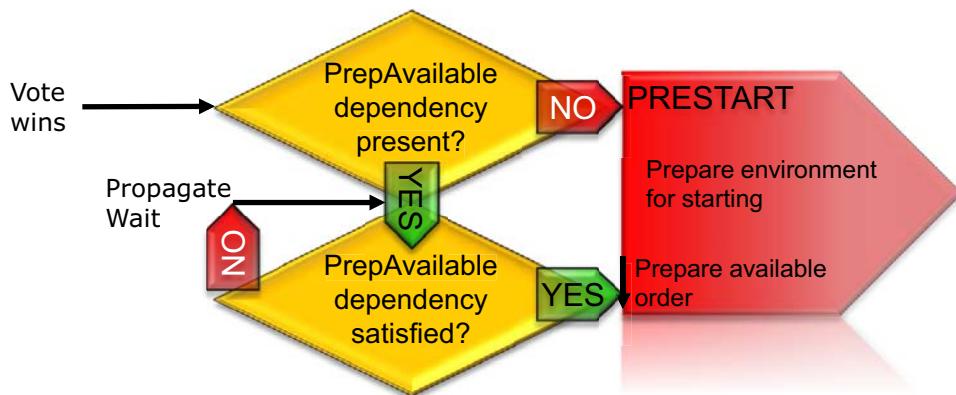
The phases, types, and commands are defined in the customization dialogs. Return code zero can be specified as a condition to indicate the success of a prestart command. A failed prestart command terminates the desired state request actions. Prestart actions are executed when an agent receives a PrepareAvailable order. Prestart or poststart commands do not change agent status.

# Starting applications: Prepare available order

## Prepare available order

- Should be sent if the application is down and has a desired state of available
- Might be sent if the automation flags permit
- Can be sent when the PrepAvailable dependencies, if any, are satisfied

Independent from MakeAvailable dependencies



© Copyright IBM Corporation 2019

3-27

## Starting applications: Prepare available order

The prepare available order should be sent if the application is down and has a desired state of available.

The prepare available order can be sent if no PrepAvailable dependency exists or the PrepAvailable dependency is satisfied as soon as the specified Condition has been satisfied for the supporting resource. This means that PRESTART commands are issued before a MakeAvailable dependency is fulfilled. Therefore, if the PRESTART commands rely on another resource running, you should define a PrepAvailable relationship.

The PrepAvailable Default Condition is WhenAvailable. Prestart actions are executed when an agent receives a PrepareAvailable order. The prepare available order might be sent if the automation flags permit it.

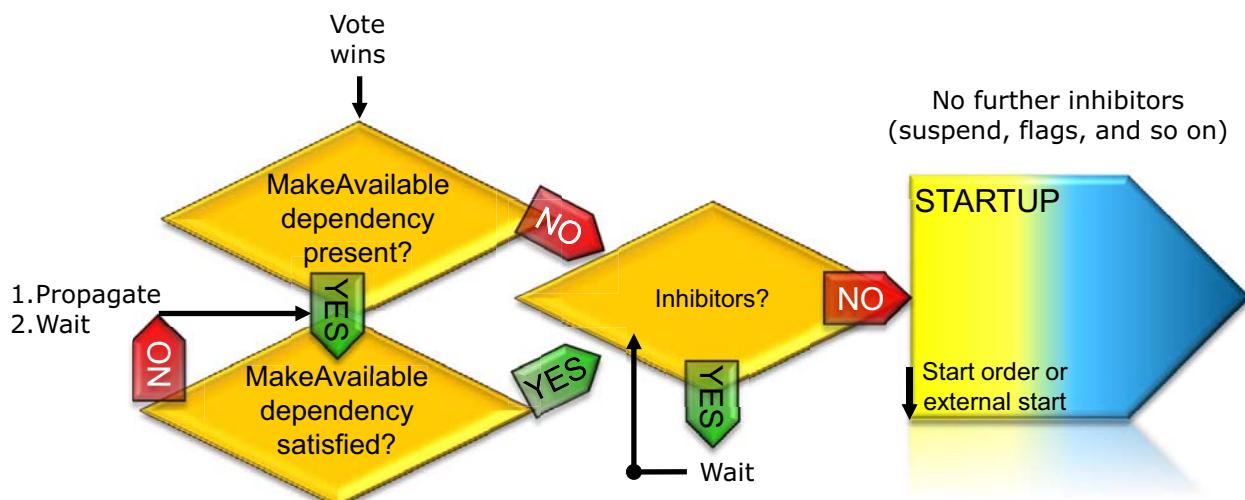
You can also use this facility for performance improvements. You can have the automation product issue a sequence of prestart commands for a resource while the supporting resource is starting.

# Starting applications: Make available order

## Make available order

- Should be sent if the application is down and has a desired state of available
- Might be sent if the automation flags permit
- Can be sent when the PrepAvailable dependencies, if any, are satisfied
- Can be sent when the Start dependencies, if any, are satisfied

Independent from PrepAvailable dependencies



© Copyright IBM Corporation 2019

3-28

## Starting applications: Make available order

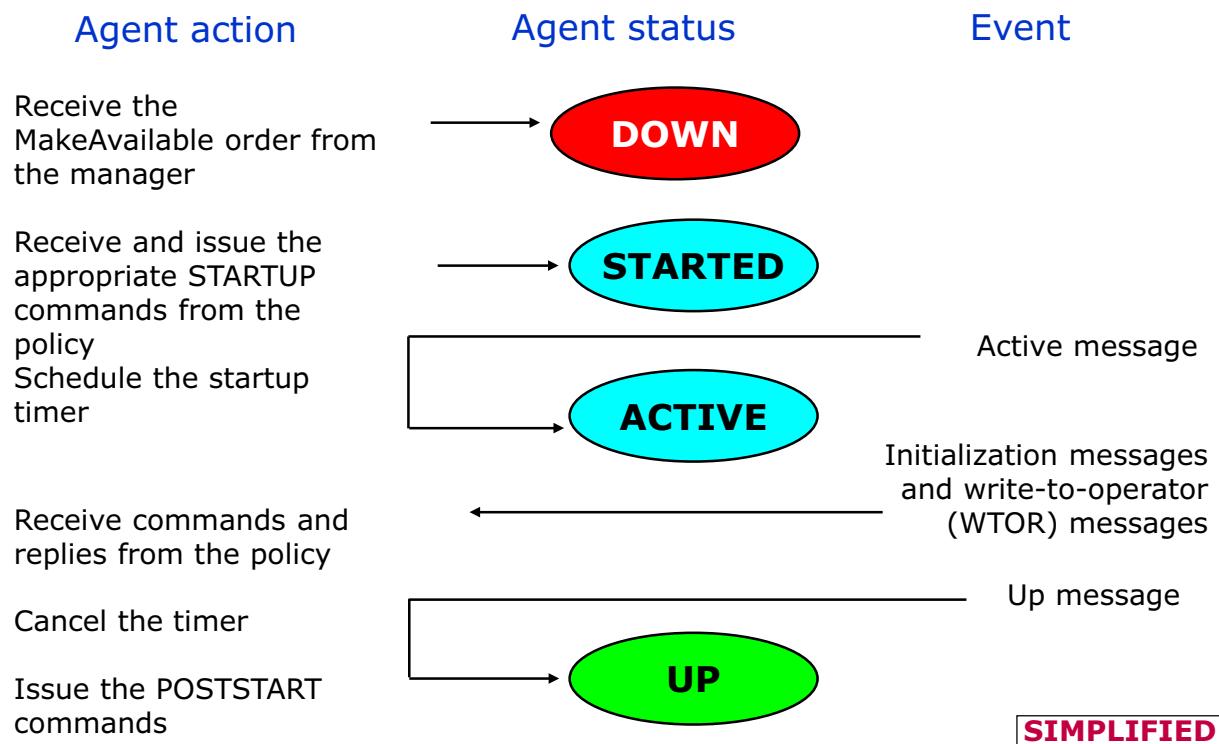
The make available or start order should be sent if the application is down and has a desired state of available.

The prepare available order can be sent when the following criteria is complete:

- The PreStart dependencies, if any, are satisfied. If, for some reason, the PreStart is unsatisfied, it does not send the start order.
- The Start dependencies, if any, are satisfied.

The prepare available order might be sent if the automation flags permit it. Inhibitors, like startup triggers must not be present to send the start order.

## MakeAvailable order scenario



© Copyright IBM Corporation 2019

3-29

### MakeAvailable order scenario

When a MakeAvailable order is received, the agent retrieves and issues the start command for the resource. After issuing the start command, the automation agent initiates these tasks:

- Sets the agent status for the resource to STARTED.
- Starts the Start Up timer, which expires after a defined interval.
- Sends the new status to the automation manager.

The automation product uses *active message* processing in the automation table to verify that the job starts. For those resources that are z/OS-started tasks, the next event that the automation agent processes is the active message. The agent uses predefined automation table statements. These statements are used to process the active message and run a command to set the resource status to ACTIVE.

A status of ACTIVE means that there is a started address space. It does not mean that the application is running as desired, or is fully initialized. ACTIVE is a transient status.

Typically, the next event that is processed is a message, which indicates that the resource is fully initialized. This message is often called the UP message. An automation table statement schedules

a program that sets the agent status to UP. The UP message triggers the action and cancels the outstanding startup timer.

After the resource is ACTIVE, in some cases, it might issue messages or WTORs signaling that some operator action is needed. These actions typically prevent the initialization from completing. An example is RMF can issue a WTOR requesting whether initialization should proceed.

The operator reaction to such messages is typically some command or reply to the WTOR. These commands and replies can be defined in the policy and automated. When the appropriate message is received, automation looks up and performs any defined action. As these commands and replies are issued, there is no change in the agent status of the resource. Any defined poststart commands run now. Similar to prestart, successful command completion can be defined for poststart commands.



**Note:** **POSTSTART** commands are issued only for resources that automation starts.

The resource is fully initialized and the startup is complete according to the automation agent.

# Stopping applications revisited

- A stop has three major phases:
  - SHUTINIT: Prepare environment for stopping
  - SHUTxxxx: The actual stop commands
    - NORM: Normal shutdown
    - IMMED: Immediate shutdown
    - FORCE: Force a subsystem off the system without any delay
  - SHUTFINAL: After stop is completed
- Multiple passes for SHUTxxxx and SHUTFINAL, if required



3-30

## Stopping applications revisited

The shutdown of an application can be done in three phases:

1. SHUTINIT: To prepare the environment so that the application can be stopped. If the desired goal of the resource is unavailable, then the SHUTINIT commands for the dependent resource are issued as soon as the specified condition for the supporting resource is satisfied. If there is no PrepareUnavailable, then the SHUTINIT commands are issued when the INGREQ is issued.
2. SHUTxxxx is one or all of the following types:
  - SHUNORM: Commands for a normal, orderly shutdown; for example, stop VTAM normally with a Z NET,QUICK.
  - SHUTIMMED: Commands for an immediate shutdown; for example, stop VTAM immediately with a Z NET,CANCEL.
  - SHUTFORCE: Commands to force the application to; for example, a z/OS PURGE or CANCEL command.

The SHUTxxxx types are fixed (NORM, IMMED, or FORCE). Stop types are not supported.

A common implementation of SHUTxxxx processing uses **PASSes**. For example, when stopping TSO, you can define **PASS1** to send a shutdown notification message to all logged-on users. Then, you use **PASS2** to issue the command to end TSO.

3. SHUTFINAL: Commands to issue when the application ends. SHUTFINAL commands are issued only if the automation product stops the resource.

# Stopping applications: Prepare unavailable order

## Prepare unavailable order

- Should be sent if the application is up and has a desired state of unavailable
- Might be sent if the automation flags permit
- Can be sent when the PrepUnAvailable dependencies, if any, are satisfied

Independent from MakeUnAvailable dependencies



© Copyright IBM Corporation 2019

3-31

## Stopping applications: Prepare unavailable order

The prepare unavailable order is sent when a vote becomes the winning vote and no PrepUnAvailable dependency exists or the PrepAvailable dependency is satisfied as soon as the specified Condition has been satisfied for the supporting resource. This means that SHUTINIT commands are issued before a MakeUnAvailable dependency is fulfilled. Therefore, if the SHUTINIT commands rely on another resource running, you should define a PrepUnavailable relationship.

The PrepUnavailable Default Condition is WhenDown. Active dependencies create votes to put supporting resources in the required condition.

SHUTINIT actions are executed when an agent receives a PrepareAvailable order.

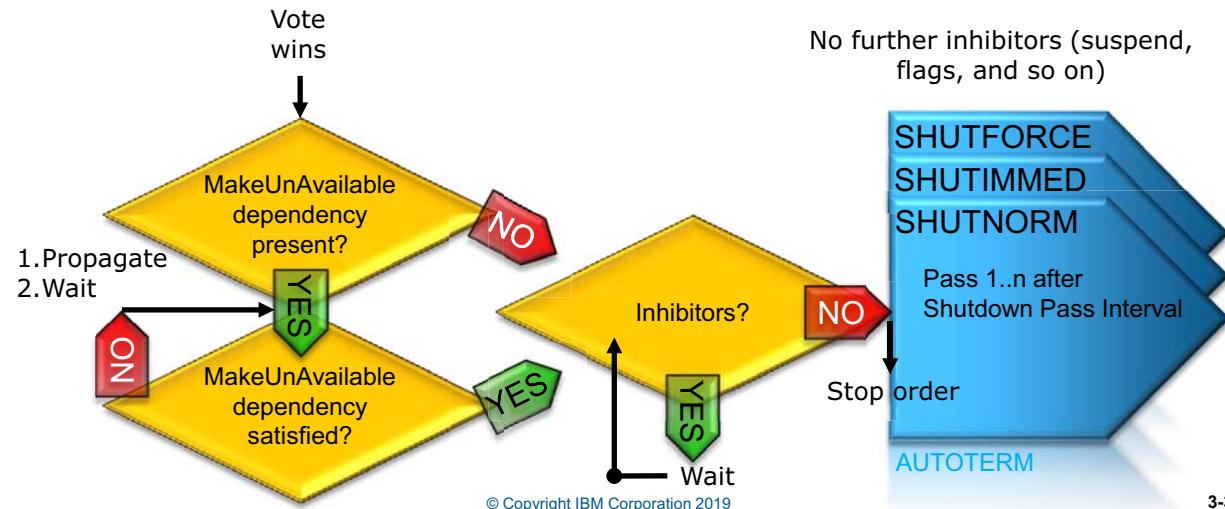
You can also use this facility for performance improvements. You can have the automation product issue a sequence of prestart commands for a resource while the supporting resource is starting.

# Stopping applications: Make unavailable order

## Prepare unavailable order

- Should be sent if the application is up and has a desired state of unavailable
- Might be sent if the automation flags permit
- Can be sent when the PrepUnavailable dependencies, if any, are satisfied
- Can be sent when the Stop dependencies, if any, are satisfied

Independent from PrepUnavailable dependencies



3-32

*Stopping applications: Make unavailable order*

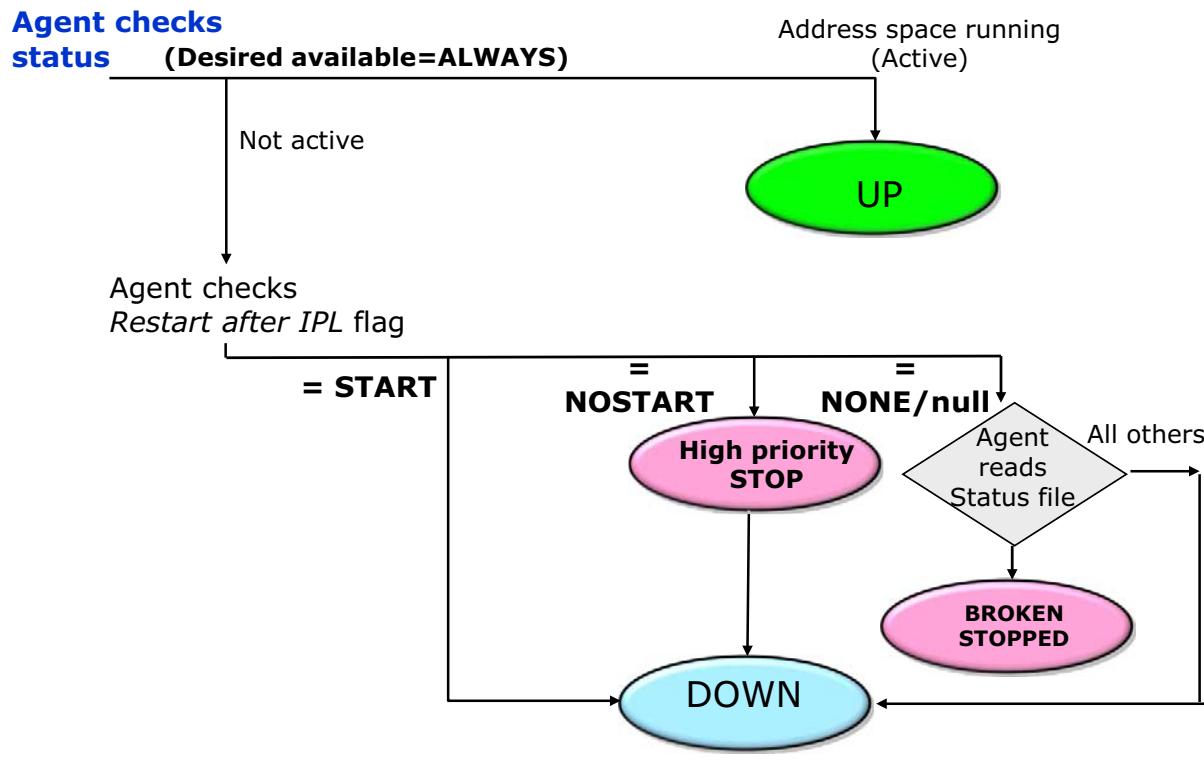
The make unavailable or stop order is sent when a vote becomes the winning vote and no MakeUnavailable dependency exists or the MakeUnavailable dependency is satisfied as soon as the specified Condition has been satisfied for the supporting resource.

Active dependencies create votes to put supporting resources in the required condition.

Inhibitors, like shutdown triggers or an observed status of PROBLEM or HARDDOWN, must not be present to send the stop order.

When the agent executes the order, further inhibitors are checked.

## Actions after an IPL



© Copyright IBM Corporation 2019

3-33

### Actions after an IPL

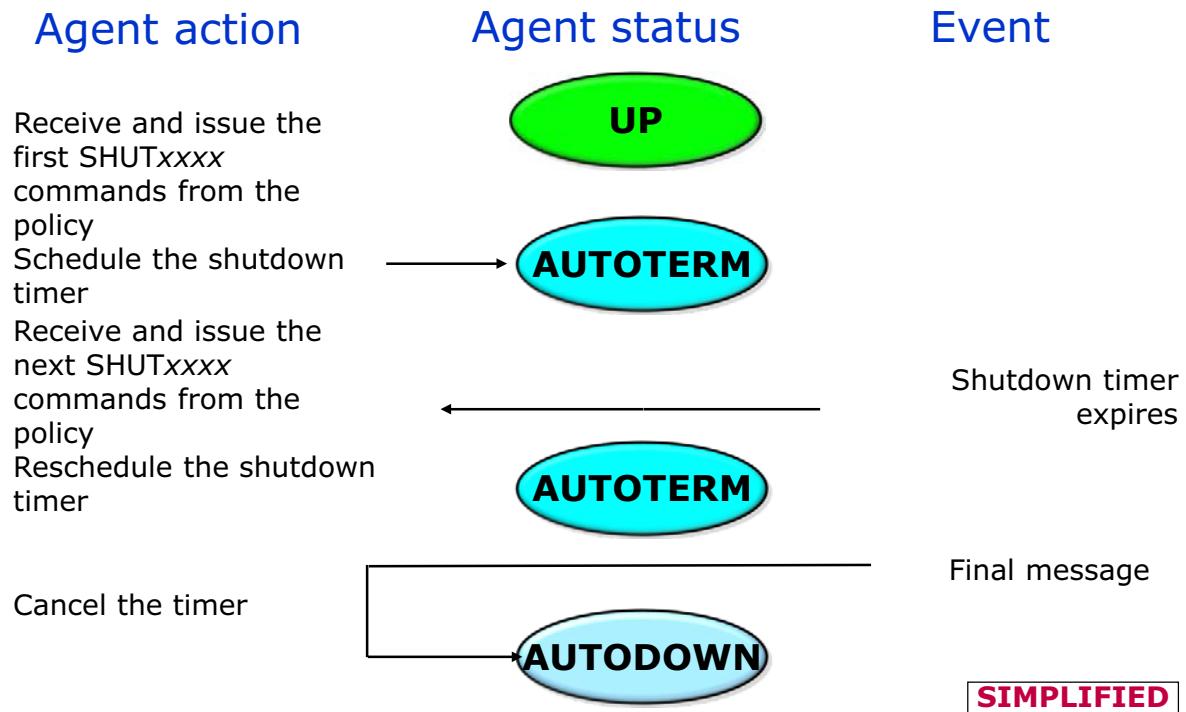
This slide shows the automation agent processing for an MVS resource immediately after a system IPL. For resources with a Desired Available policy of ALWAYS (the default), the following actions occur:

- Automation checks whether the resource is active. If it is, automation assumes that the resource is UP. There is no simple way that it can check whether the resource is fully initialized; so, the assumption is reasonable.
- This slide illustrates the flow during a system IPL where only the automation address spaces are active. The flow is identical when the automation agent is recycled.
- Any application that has a value of START for RESTART AFTER IPL indicates that the preshutdown status of the application is ignored. Automation sets the resource status to DOWN.
- Any application that has the RESTART AFTER IPL flag of NOSTART is stopped with a high priority INGREQ STOP request. These resources require operator attention before they can be started.
- Any application that has a blank in the RESTART AFTER IPL field indicates that the application is startable, depending on its status. If the field is left blank, the value can be inherited from a class or the application defaults definitions. If the status is STOPPED, BROKEN, or CTLDOWN

it is not changed and the application is not started. Applications in these states require operator intervention. All other applications are set to a status of DOWN. A value of NONE in this field is the same as blank or null, except that the value is not inherited from a *class* or application defaults definition.

If the saved status is BREAKING or STOPPING, the resource is not restarted and its status is set to BROKEN or STOPPED. Setting the agent status to DOWN means that there is no status inhibitor to starting the resource. If the desired state is AVAILABLE, then the INITSTART flag must be set to YES before the agent can start the resource.

## MakeUnavailable scenario



© Copyright IBM Corporation 2019

3-34

### MakeUnavailable scenario

A shutdown of an automated resource is done with the INGREQ STOP command. Based on policy definitions and current states, the automation manager sends a MakeUnavailable order to the automation agent.

When a MakeUnavailable order is received, the agent retrieves and issues the commands for the defined passes as the following actions occur:

- Begin the shutdown process for the stop-type.
- Start the shutdown (Pass interval) timer.
- Set the agent status of the resource to AUTOTERM. This status indicates that the shutdown is in progress.

When a final message (*job ended*) is received, these actions occur:

- The final message is processed.
- The resource status is set to AUTODOWN.
- The timer is canceled.
- Any defined SHUTFINAL commands are run.

However, if the shutdown timer expires and there is a command for another pass in the policy, the shutdown timer is reset and the new command is issued.

AUTODOWN indicates that the automation agent successfully shut down the resource after receiving an order from the automation manager. A status of AUTODOWN indicates that a resource can be started again, if necessary.

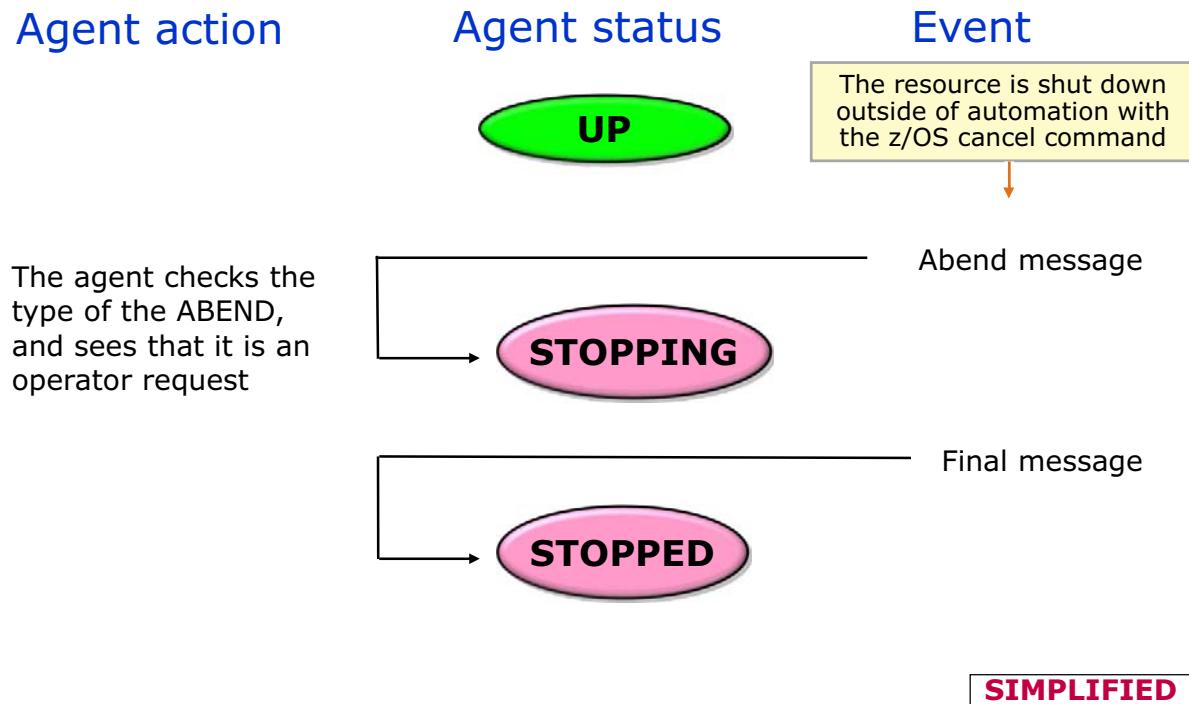
A resource can generate messages as it shuts down. These messages can require some action (command or reply). The actions can be stored in the policy. Again, the automation agent looks up and takes the appropriate action.

Address spaces do not always shut down cleanly when you issue a shutdown command. For example, issuing the MVS Z NET,QUICK command does not always shut down VTAM. In this case, the shutdown timer scheduled by the MakeUnavailable processing expires and the automation agent regains control. It first checks to determine whether the resource is active.

If it is not, the agent status is set to AUTODOWN and the shutdown is complete. If the resource is still active, the automation agent checks the automation policy for another command (*another pass*) for the stop type. If another pass is defined, the agent issues the command and reschedules the shutdown timer. For the VTAM example, the second pass command might be MVS Z NET,CANCEL. The agent status remains AUTOTERM.

If the final message is received before the shutdown timer expires, the agent sets the status of the resource to AUTODOWN. If the final message is not received before the shutdown timer expires, the automation agent repeats the check for more passes process until no more passes are available.

## Unexpected shutdown or ABEND scenario



© Copyright IBM Corporation 2019

3-35

### Unexpected shutdown or ABEND scenario

From outside of the automation product, operators can terminate an address space with a CANCEL or PURGE command. This flow describes the automation processing that occurs when a resource is stopped outside of automation.

When a resource is canceled externally (outside of the automation product), z/OS generates message IEF450I with a special system code. The system code indicates the type of ABEND. When the automation agent examines the message, the resource status is set to STOPPING, if the ABEND type indicates an operator cancel.

When the resource issues the final message, the automation agent sets the status to STOPPED. A status of STOPPED indicates these two items:

- The automation product did not initiate the shutdown.
- The RESTART OPTION for the resource is not set to ALWAYS.

No SHUTFINAL commands are issued.

When ABENDS occur, the flow is UP to ABENDING > RESTART or BREAKING > BROKEN.

STUCK when run out of shutdown commands and ZOMBIE when term message was issued, but address space is still there.

# Lesson 6 Factors that can influence the request process

## Lesson 6: Factors that can influence the request process

- Votes
- Vote propagation
- Service Periods
- Triggers and Events
- Inhibitors

© Copyright IBM Corporation 2019

3-36

### Factors that can influence the request process

This lesson explains the factors that can influence the request process:

- Vote sources and priorities
- Vote propagation to group members and along dependencies
- Service Periods that generate votes
- Triggers and Events
- Inhibitors for requests

## Where do votes come from?

- The automation manager receives requests from different sources and originators
- The defined Desired Available policy determines the default vote
- Requests are translated to votes based on the following criteria
  - Active dependencies
    - Only the highest priority start and stop votes are forwarded
    - INGVOTE displays winning vote and propagated vote
  - Scope specified: ALL, ONLY, CHILDREN
- Votes are generated by the automation manager



© Copyright IBM Corporation 2019

3-37

Where do votes come from?

A request is an action to change the desired status for a resource. Requests can originate from several sources. Votes can also come from the automation manager and the automation agent. The defined Desired Available policy determines the default vote.

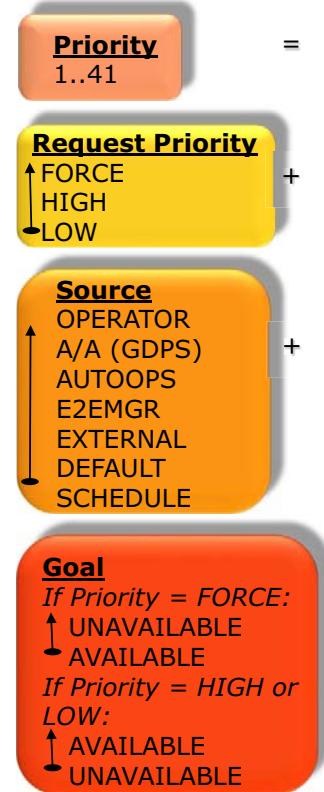
A request can generate one or more votes against multiple resources. The votes are propagated to the resources based on the scope of the request and any *active dependencies* that are defined.

Only the highest priority start and stop votes are propagated.

INGVOTE displays the winning vote, indicated by a Y in the WIN column, and the propagated vote, indicated by \*. If the winning vote is a start vote then the propagated vote is a stop vote and vice versa. If a vote is removed, the highest priority remaining vote is chosen.

## Vote priority and life-span

- Numerical priorities are assigned to requests
  - Requests are used to change the desired state of a resource
  - Requests can originate from several sources with different priorities:
    - Operators
    - Autotasks
- Requests and their resulting votes are persistent: they remain active until removed or until they expire



© Copyright IBM Corporation 2019

3-38

### Votes priority and life-span

The request priority indicates three specifications in the request to the automation manager:

- Where the request originates
- What the external priority in the request is: FORCE, HIGH, or LOW
- What action is requested

Requests are *persistent*. In other words, they remain active until one of the following occurs:

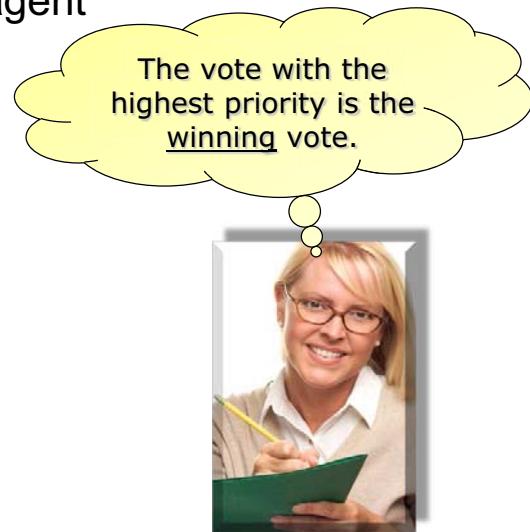
- Removed by an operator manually
- A *service period* expires, removing its request
- An expiration time that is provided on an INGREQ command is reached
- A WARM or COLD start of the automation manager removes all votes

It is possible, for example, that an *unsatisfied winning* vote can be *satisfied* later.

For each targeted resource, the automation manager keeps only one request from each source. Only the latest request is kept.

## The winning vote

- There can be multiple requests and votes against one resource
- The winning vote has the highest priority
- The winning vote determines the desired state
- Orders are sent to the automation agent
  - No orders are sent to the agent if the following occurs:
    - Observed status is PROBLEM or HARDDOWN
    - Automation flag is NO
    - System is suspended
    - Resource is suspended



© Copyright IBM Corporation 2019

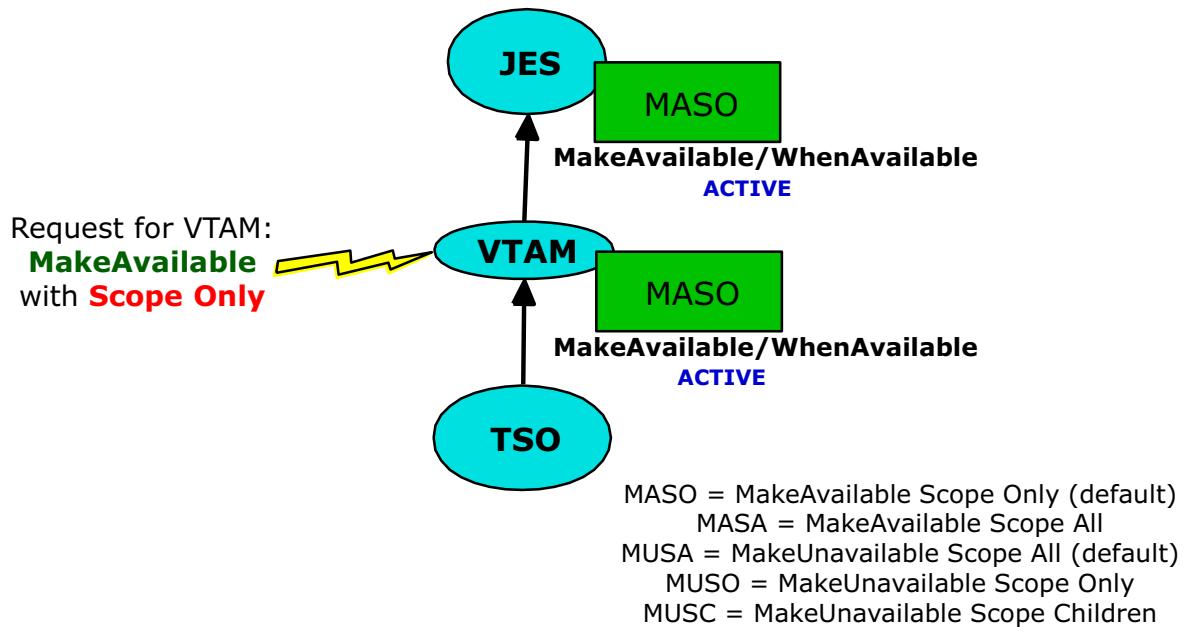
3-39

### The winning vote

When there are several votes for a single resource, the manager chooses a winner. The desired state of the winning vote becomes the desired status of the resource.

If the actual status of the resource and its desired status do not match, the automation manager attempts to send an order to the agent to start or stop the resource. Before the manager sends an order, a number of possible *inhibitors* are checked. These inhibitors can delay the sending of the order until certain conditions are satisfied, or overridden. When all inhibitors are satisfied, the manager sends the order to the agent.

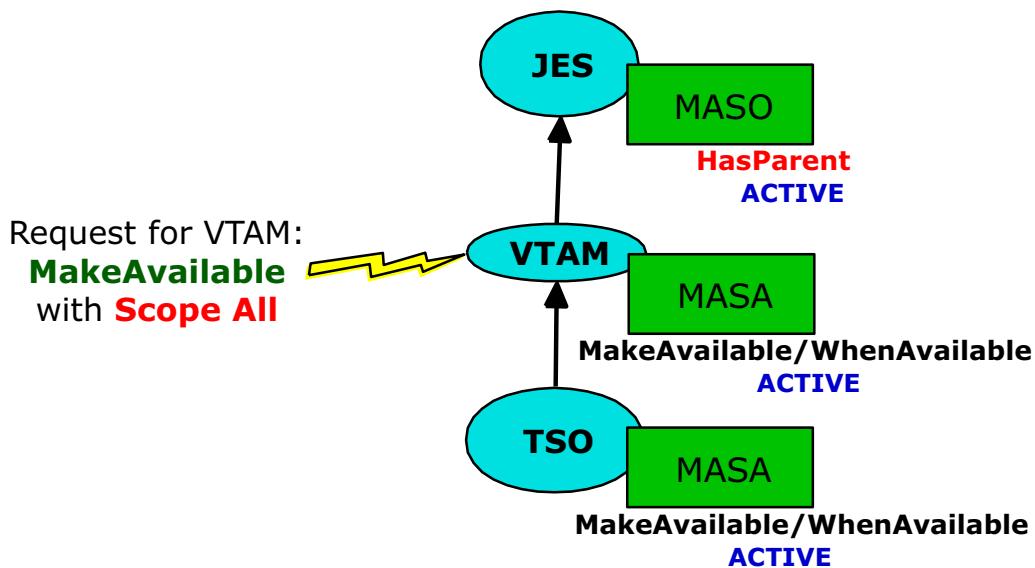
## MakeAvailable/WhenAvailable with Scope Only



### MakeAvailable/WhenAvailable with Scope Only

This example shows a MakeAvailable/WhenAvailable (MAWA) relationship between VTAM and JES. A request is made to start VTAM with Scope Only, MakeAvailable Scope Only (MASO). JES also receives a MASO vote because it has an active MAWA relationship with VTAM. The request to start VTAM results in two votes, one to make VTAM available and one to make JES available. Vote propagation can be different for other request types, such as MASO, MASA, or MUSO.

## MakeAvailable/WhenAvailable and HasParent with Scope All



© Copyright IBM Corporation 2019

3-41

### MakeAvailable/WhenAvailable and HasParent with Scope All

This example shows a MakeAvailable/WhenAvailable (MAWA) relationship between TSO and VTAM. The relationship between VTAM and JES is a HasParent.

A start request with Scope All against VTAM generates a MakeAvailable Scope All (MASA) vote. TSO also receives a MASA vote because it is a dependent of VTAM. Scope All causes the MASA vote against VTAM to be propagated to all the children of VTAM, TSO, and any children of TSO. JES receives a MASO vote because it has an active HasParent relationship with VTAM.

The request to start VTAM results in three votes: one to make TSO available, one to make VTAM available, and one to make JES available. The JES vote is different because its relationship is different.

## How the winning vote is chosen

- Requests come from several sources
  - Each results in a vote against a resource
  - Each is persistent
  - Votes are propagated through the dependency tree
- How the winning vote is determined
  - Request priority scheme
  - Vote priority
  - Vote action value

### How the winning vote is chosen

Requests can originate from several sources. Each request results in one or more votes against the target of the request. Votes remain active until satisfied or removed.

Votes are assigned a numeric value, called a *vote priority*. The vote priority is based on four factors:

- Source of the request
- Priority level of the request
- Type of the request
- Automation manager resource status

## Request sources

Source	Description
Operator	A logged-on operator enters a request with INGREQ
A/A	By GDPS Continuous Availability Controller
AUTOOPS	A REXX procedure or an automation table entry issues an INGREQ or similar command in an autotask
E2EMGR	Handles requests from the end-to-end automation manager
External	An IBM Workload Scheduler for z/OS request through the automation to IBM Workload Scheduler for z/OS interface
DEFAULT	By any other source/method, for instance runmode requests originating from INGRUN
Schedule	A standard service period definition, or operator overrides
Automation Manager (implied request)	MakeAvailable when desired available is ALWAYS (default) MakeUnavailable when desired available is ONDEMAND

### Request sources

The table shows the sources from which a request or vote can originate. From the automation manager, there is always a default or implied vote of MakeAvailable against each resource. Votes for implied MakeAvailable requests are not visible; however, votes for implied MakeUnavailable requests can be seen. They each have a request priority of 00000000.

When an operator generates a request, the request is routed to the automation manager. The origin of the command is a NetView task who is a logged-on operator. In NetView terms, the task has the *attended* attribute. Most automation operators are *unattended* automation operators, *autotasks*.

Requests can be sent from IBM Workload Scheduler for z/OS through the provided automation interface. IBM Workload Scheduler for z/OS requests are always external.

*Service periods*, used to automate planned UP and DOWN windows of time for chosen resources, can generate internal requests.

Several requests can target a single resource, each from a different source. However, only one request is accepted from each source, the latest request. Use the INGVOTE command to view current requests or votes.

When a group propagates a vote to one of its members, it increments the Priority by 1000. For nested groups, this can result in multiples of 1000s.

## Persistence rules for requests

- Requests remain active until removed (canceled or expired), replaced, or automatically removed at SYSGONE
- One request from one source for one resource at a time
  - New requests replace previous requests
  - Low priority requests can replace high priority requests
- A request can supersede an earlier request from the same source
  - For example, a request from operator OPER1 on SYSA can replace a request from OPER2 on SYSB
- The default (implied) is a MakeAvailable request from the automation manager
  - Can be changed by setting the Desired Available policy to OnDemand or ASIS

© Copyright IBM Corporation 2019

2

### Persistence rules for requests

Requests are persistent (can survive IPLs). However, a WARM or COLD start of the automation manager removes all votes. For each targeted resource, the automation manager keeps only one request from each source. Only the latest request is kept.



**Note:** Using the Advanced Automation Option (AAO) INGREQ\_Originator, you can have multiple requests from operators. For more information, see *System Automation for z/OS Customization and Programming Guide*.

When the term *source* is used, it has a specific meaning for operators and auto operators. Operator means any NetView operator. A request from one operator replaces an existing request from a different operator. The operators might even be logged on to different systems within the SAplex. Similarly, for auto operators (*autotask*), a request from one autotask replaces an existing request from another autotask. A request can also be explicitly canceled, with one of the following commands: INGVOTE, or INGREQ REQ=CANCEL.

# Request priority scheme

Requests have priorities that are associated with them based on these items:

- Source
  - OPERATOR
  - A/A
  - AUTOOPS
  - E2EMGR
  - EXTERNAL
  - DEFAULT
  - SCHEDULE
- Request Priority level
  - LOW
  - HIGH
  - FORCE
- Action
  - START
  - STOP
- Scope
  - ALL
  - ONLY
  - CHILDREN

## Request priority scheme

The request priority indicates three specifications in the request to the automation manager:

- Where the request originates
- What the external priority in the request is
- What action is requested

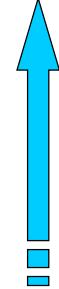
When there is more than one vote for a resource, the highest priority vote wins. The winning vote determines the state that the automation manager applies as the desired status for that resource. If the winning vote is a MakeAvailable vote, the desired status is set to AVAILABLE. If the winning vote is a MakeUnavailable vote, the desired status is set to UNAVAILABLE.

If the current observed status of an application is inconsistent with its desired status, the automation manager can send orders to the agent to start or stop the application. If there are two votes with identical priority, one of them is arbitrarily chosen as the winning vote. The decision does not affect any action because identical priority votes are requesting the same action. It does affect what the operator sees in INGVOTE displays.

# Request priorities by source

Request priority levels

Request source	Low (default)	High	Force
Operator	Yes	Yes	Yes (highest)
A/A	Yes	Yes	Yes
Autoops	Yes	Yes	Yes
E2EMGR	Yes	Yes	Yes
External	Yes	Yes	Yes
DEFAULT	Yes	Yes	Yes
Schedule	Yes	Yes	No



© Copyright IBM Corporation 2019

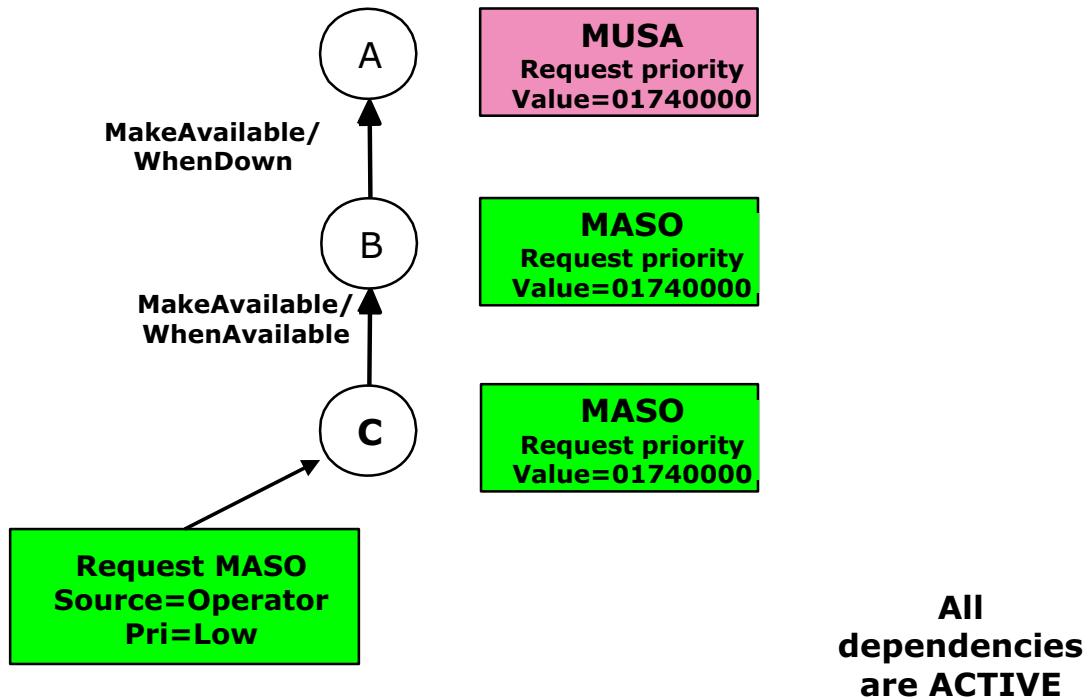
3-46

## Request priorities by source

Three priority-level values are used: Low, High, and Force. However, not all priority values can be specified in all requests. As the table shows, there is no Force priority for any of schedule requests (*service period requests*). For an operator request, the priority is displayed in a field on the INGREQ panel.

By default, an operator request has the highest priority from the source perspective, overriding all other requests for a resource. This approach enables the operator to override requests that arise based on the normal policy definitions.

## MakeAvailable vote propagation example



© Copyright IBM Corporation 2019

3-47

### MakeAvailable vote propagation example

Assume that there is a MakeAvailable/WhenDown dependency for a dependent resource. If a MakeAvailable request is issued for the dependent resource, it generates a MakeUnavailable vote for the supporting resource.

This slide shows an example of a low priority MASO (Make Available Scope Only) request from operator, against application C. The vote priority value is 01740000. The example shows that the MUSA (Make Unavailable Scope-All) vote uses the original 01740000 priority from the MASO (MakeAvailable Scope Only) request.

Suppose the supporting resource already has a MakeAvailable vote from the same source. The differential vote of the vote action values ensures that the supporting resource retains its desired status of AVAILABLE.

Each MASO adds 0008 to the initial priority (01740000). The MUSA (Make Unavailable Scope All) adds 0004 to the initial priority. The added priority values are not displayed on the screen.

Alternatively, suppose that the supporting resource already has a MakeUnavailable Scope Only vote from the same source. In this case, the differential vote of the vote action values ensures that the supporting resource obtains a winning vote with Scope=All specified.

# Inhibitors

- Prevent the automation manager from sending orders to the automation agent
- Affect the start and the stop of resources
- Have the following impact:
  - Desired status remains unchanged
  - Requests and votes remain unchanged
- Operators can override most inhibitors
  - Can lead to unpredictable results

3-48

## *Inhibitors*

At the end of the vote generation process, each application ends up with a winning vote. The winning vote determines the desired status of an application. If there are no votes for an application, the default action is to set its desired status to Available (based on the default value of the desired available policy, ALWAYS).

The manager attempts to start or stop an application when its desired status and observed status do not match. As a result, the manager sends a start (MakeAvailable) or stop (MakeUnavailable) order to the appropriate automation agent.

Some conditions, called ***inhibitors***, can prevent the manager from sending an order to the agent. They inhibit, but do not alter the action. After all inhibitors for an application are removed, the manager sends the order to the agent. Inhibitors play no part in the decision-making process. They delay, not terminate, orders to the agent.

Operators can **override** inhibitors with the INGREQ command, but this can lead to unpredictable results.

## Possible inhibitors

- Unsatisfied dependencies
- Automation manager flags
- Suspended resource
- Suspended automation
- Automation status is PROBLEM
- Observed Status is HARDDOWN
- Unset trigger

3-49

### *Possible inhibitors*

This slide shows some possible inhibitors:

- **Unsatisfied Dependencies:** For example, if VTAM has a MakeAvailable/WhenAvailable dependency on JES2, then VTAM cannot be started until JES2 is in an Available condition. If you attempt to start VTAM before JES is available, it fails. (Remember, the default action is to start a resource.) More than one inhibitor can exist at the same time.
- Before issuing such a start order, the manager checks whether all MakeAvailable dependencies are met. (Remember that all dependencies are checked, whether active or passive, and that the checking can involve strong or weak chaining for each dependency.) This action ensures that VTAM is not started until the start is likely to be successful.
- When a dependency condition is not satisfied, the compound status is set to AWAITING if the supporting application is startable. If the supporting application is not startable (for example,

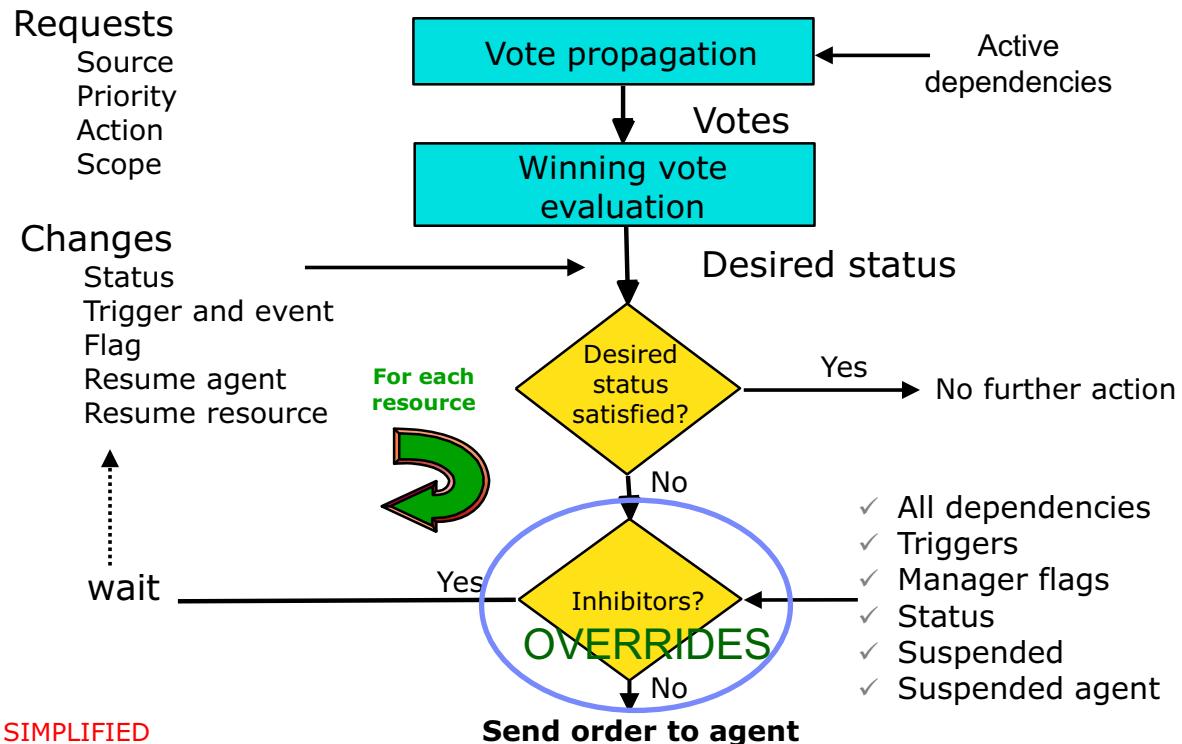
agent status is BROKEN), the compound status is set to INHIBITED. Similar conditions apply when stopping an application.

- **Automation manager flags:** There are two automation manager flags. The flags affect only resources under the control of the automation manager. Use the INGLIST panel command or the programming interface command, INGSET to set these flags.
- **Suspended resource:** automation allows the operator or administrator to suspend a resource. While it is suspended, automation does not attempt to start or stop this resource.
- **Suspended automation:** You can suspend automation for a system with the INGAMS SUSPEND command. All requests result in a compound status of DENIED until the processing is resumed for the system.
- **Automation status of PROBLEM:** When an application goes to a status of STUCK, ZOMBIE, or HALFDOWN, its automation status is set to a value of PROBLEM. Its compound status is set to PROBLEM indicating that an operator action is required.
- In such cases, stop orders to the application are inhibited. The operator must either set the status to UP or complete the shutdown of the application manually. In that way, the application goes to one of the inactive statuses and sets the automation status to IDLE. If the status is set to UP, the stop order can then be sent to the application. If the application becomes inactive, it satisfies the MakeUnavailable request.
- The automation status can also be set to PROBLEM if the agent issued a command that is defined as must succeed and a nonzero return code is received. This situation can occur as part of a prepare or a make order. In such cases, you can use API command INGSET, or A (Update row-command on INGLIST panel) to reset the automation status.
- Agent status of CTLDOWN, BROKEN, or STOPPED require similar operator actions. These agent status values set the automation manager observed status to HARDDOWN and the compound status to PROBLEM. To correct the situation, the operator must set the status to AUTODOWN before orders can be sent.
- **Trigger Condition Not Set:** Triggers are combinations of events. Events are switches that indicate whether something happened within the SAplex. A trigger is satisfied when all of its events are set. Triggers are used to indicate whether a specified set of conditions exists.
- Automation policy for an application can specify that it can start only when a trigger is satisfied. The application is using a start-up trigger. The unsatisfied trigger is an inhibitor to the MakeAvailable order. If an application has several start-up triggers, it can be started when any one of the triggers is satisfied.
- Applications can also have shutdown triggers that inhibit MakeUnavailable orders. When a shutdown trigger is unsatisfied, the compound status of the application is set to AWAITING.
- **Automation agent flags:** These flags are used for resources that the agent starts, stops and monitors. If an agent flag is set to NO, and it prevents the order from being carried out, the

agent tries the request at regular intervals until it is successful. Both the compound status and automation status are set to DENIED.

- Automation agent flags are not true inhibitors because the automation manager does not control them. They are included in this explanation because they can prevent the start or stop of an application.
- These flags can optionally be set in the customization dialogs and set or reset with the INGAUTO command. The agent automation flags are not real inhibitors, although they act as such.
- Operationally, some of the inhibitors can be overridden by using the OVERRIDE options of the INGREQ command.

## Location of inhibitors in the request flow



© Copyright IBM Corporation 2019

3-50

### Location of inhibitors in the request flow

Before the manager sends an order, a number of possible *inhibitors* are checked. These inhibitors can delay the sending of the order until certain conditions are satisfied, or overridden. When all inhibitors are satisfied, the manager sends the order to the agent.

Overriding inhibitors can be done by modifying the original request, but this can lead to unpredictable results as the inhibitors are there for a reason.

Instead an operator can determine the inhibitors and try to resolve them by entering requests to put supporting resources in the required condition, setting flags, resolving problem states, setting triggers and events and so on.

# Lesson 7 Application group details

## Lesson 7: Application group details

- Characteristics of application group
- Types and natures of application groups
- System application groups
- Sysplex application groups
- Active and passive application groups

### *Application group details*

This lesson highlights and explains the benefits of groups when managing automated business applications in an enterprise.

# Characteristics of an application group

- The automation manager controls application groups
- They can have these items:
  - Dependency links
  - Runtokens
  - Service periods
  - Triggers
  - Monitors
- They can be linked to multiple systems or SAplexes
- They are System Automation for z/OS resources
  - Resource type is application group; designator APG
  - Status is assigned, based on group-type and the status of members
  - Can be monitored, started, or stopped as a group

Applications can be linked to more than one application group

## *Characteristics of an application group*

Application groups have many of the characteristics that are associated with applications:

- Resource dependency relationships
- Links to schedules (Group members do not inherit schedule values.)
- Runtokens

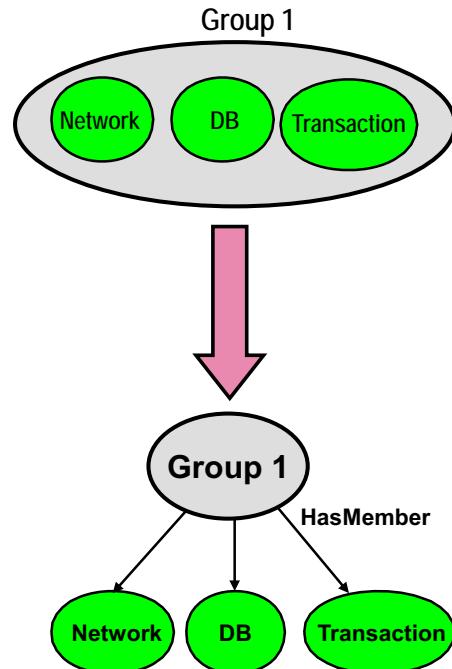
Like applications, you can enter start and stop requests for application groups. The start or stop of a group is a propagation of requests to its members. Members of a qualified application group inherit the runmode qualification of the group. A trigger does not prevent the start or stop of a group. A trigger is checked when the manager sends an order to a group member. Members can inherit triggers from the application group.

Application groups can be linked to multiple systems or SAplexes in an enterprise. Application groups that are linked to multiple systems or SAplexes are logically different; and different automation managers control them. Similar to an application, each application group has a value for each of the automation manager statuses. These values are derived from the status of the members of the group. Additionally, server group status is correlated against policy definitions for the *availability target* and *satisfactory target*. Applications can be linked to multiple groups;

however, use caution when taking advantage of this capability. For better performance, define service periods and triggers against an application group instead of its members.

## Application group members

- Defined in the customization dialogs
- Identified by the *HasMember* relationship
- Managed with the following commands
  - INGLIST
  - INGGROUP
  - INGMOVE
  - INGINFO
  - INGRELS
  - SDF



### Application group members

Application groups allow operators to focus on the complete business application instead of numerous individual resources with complex dependency relationships. Application groups use a *HasMember* relationship. This relationship does not alter any start or stop dependencies of individual resources.

The INGGROUP command displays the members in a group, and is the basis for group management actions. The automation manager, not the agent, maintains, and manages application groups. However, the manager sends the status of groups to automation agents for display on Status Display Facility (SDF) panels. The status of all groups is available at the SDF focal point. SDF does not display the members of a group.

## Characteristics of system application groups

- All members must be on a single system
- Members are
  - Applications
  - System application groups (nesting)
  - Resource references
- Naming convention **automation\_name/APG/system\_name**  
Example: AIRLINE\_NET/APG/SYSA
- Cannot contain *sysplex application groups* or monitor resources
- Same group can be defined on multiple systems

### Characteristics of system application groups

All members of system application groups must be on a single system with the exception of the resource a resource reference is referencing to.

Members are:

- Applications
- System application groups (nesting)
- Resource references

System Application Groups can also contain nested System Application Groups. Because a System Application Group is confined to a single system, it cannot contain a nested Sysplex Application Group. The resource name is displayed in the operator panels (INGGROUP) and in the Customization Dialog panels. The policy entry name, AIRLINE\_RES in this example, must be unique within the PDB. In many cases, the automation name (defined in the APPLGROUP INFO policy item for the APG) is the same as the entry name. The default setting for the automation name is the entry name. The automation\_name, AIRLINE\_RES, does not have to be unique. Because this is a System Application Group, you can define the same automation name on multiple systems. When the group name is qualified with a system name, AIRLINE\_RES/APG/SYSA, for example, it forms a unique representation of the group.

## Characteristics of sysplex application groups

- Contains members from any system within the SAplex
  - Applications on multiple systems
  - Other system application groups
  - Other sysplex application groups
- Can be linked to more than one SAplex
- Uses naming convention Automation\_name/APG
  - An example is AIRLINE\_RES/APG
  - Does not require the system name because it is not linked to a particular system
  - Name must be unique within the sysplex

### *Characteristics of sysplex application groups*

A Sysplex Application Group contains members from multiple systems within the sysplex. The members can be Applications, System Application Groups, or nested Sysplex Application Groups. Again, notice the naming convention. It is displayed in both the operator panels, for instance, INGGROUP, and the Customization Dialogs. AIRLINE\_RES/APG refers to the AIRLINE\_RES Sysplex Application Group. Because this is a Sysplex Application Group, you can define the same automation name on multiple sysplexes, linked to different sysplexes.

The following resources are generated when the members are linked to the APG:

- *automation\_name/APG/[system\_name]* The third part shown in brackets is not created for SYSPLEX APGs.
- *member\_name/APL/system\_name*

## Group behavior: Active or passive

- Groups can be active or passive
- Active groups (default)
  - Start and stop votes are propagated to the members
  - Active groups support monitoring and automation
- Passive groups
  - No votes are propagated to the members
  - Passive groups can be monitored only
  - Passive groups can have dependency relationships

### Group behavior: Active or passive

The default for group behavior is active. Most groups are active groups; votes are propagated to their members. No votes are propagated to any member of a passive group. The group must be controlled manually outside of the normal automation processes.

## Nature of groups

- Every application group has a nature
  - Basic
  - Server
  - Move
- The group nature affects how the group is managed and monitored
- Example
  - Determination of group status
  - Which members receive start or stop votes
- *Group members can be SAplex-wide (SA z/OS only)*

### Nature of groups

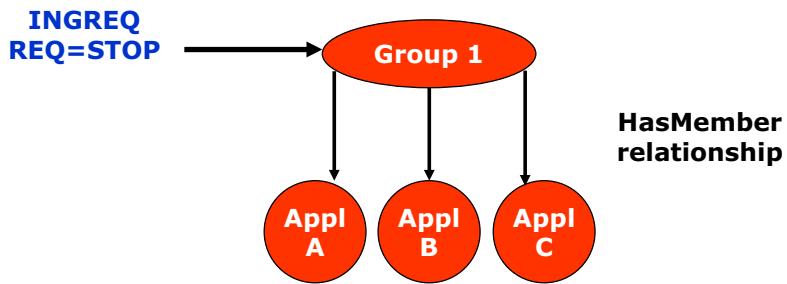
The nature of an application group describes its availability behavior:

- How it ensures that the correct number of applications are started to satisfy the requirements of the group
- How applications are selected to start or stop
- How the status of the group is determined from the status of its members

The status of a group is the aggregate of the status of the members. The status of the members influence the OBSERVED status of the group. The DESIRED status of a group influences the DESIRED status of its members.

## Behavior of groups with basic nature

- Starting or stopping a basic group results in an attempt to start or stop all members in the group
- The Desired Available policy sets the default desired status
  - A desired available of ALWAYS sets the desired status to AVAILABLE
  - A desired available of ONDEMAND sets the desired status to UNAVAILABLE
  - A desired available of ASIS sets the desired status to value in observed status
- A service period can change the desired status



© Copyright IBM Corporation 2019

3-58

### Behavior of groups with basic nature

A basic group does not propagate any votes to its members unless there is a start or stop request against the group. When a basic group is started, each of its members receives a MakeAvailable vote from the group resource. If successful, the desired status of each member is set to AVAILABLE and orders can be sent to the automation agent to start the members.

When a basic group is stopped, each of its members receives a MakeUnavailable vote from the group. If successful, the desired status of each member is set to UNAVAILABLE and orders can be sent to the automation agent to stop the members. All requests are subject to dependencies defined for the individual members of the group.

By default, basic groups are always active unless there is an action to stop the group, for example:

- An INGREQ stop command is issued
- A desired available policy of **ondemand**
- The group is linked to a schedule with service windows of downtime

For basic groups, the implied MakeAvailable vote is not propagated to its members. The members have their own implied request. The build process in the customization dialogs builds a system group resource for each system. For example, MVSA/SYG/MVSA is a basic system application group.

## Status values for basic groups

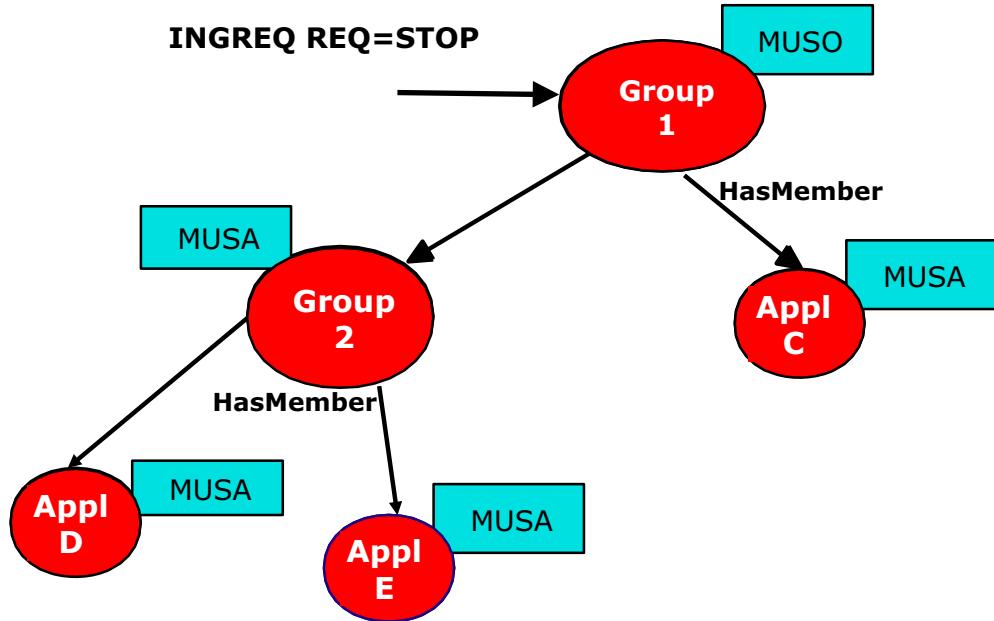
Observed group status	Meaning
AVAILABLE	All members are AVAILABLE
PROBLEM	At least one member is in PROBLEM state
SOFTDOWN HARDDOWN SYSGONE	All members of the group are UNAVAILABLE
STARTING	Some members are AVAILABLE

The status of a group is an aggregation of the status of its members

### *Status values for basic groups*

The status for a basic group depends upon the status of its members. In general, if all members of the group are AVAILABLE, then the group is AVAILABLE. If all members of the group are UNAVAILABLE, then the group is UNAVAILABLE. It is possible to have status values other than the ones listed in the table.

## Vote propagation in basic groups



© Copyright IBM Corporation 2019

3-60

### Vote propagation in basic groups

This slide shows how stop votes are propagated for a basic group. An INGREQ STOP request to the group creates a MakeUnavailable Scope Only (MUSO) vote for the group. Because of the HasMember relationships, a MakeUnavailable Scope All (MUSA) vote is propagated to each member. If one of the members is a group, the member group continues the propagation of the MUSA vote to its members.

# Factors that influence the behavior of server groups

- Availability target
- Satisfactory target
- Preference values
  - Base preference value
  - Effective preference value
  - Bonus value (Automation manager built-in algorithm)
- Group operating modes
  - Normal or
  - Recovery

## Factors that influence the behavior of server groups

These factors influence the behavior of server groups:

- **Availability target (AVT):** The number of members that must be active in the group to be AVAILABLE. The availability target is defined in the customization dialogs. During operations, the availability target can be overridden on an INGGROUP command panel. An availability target can be a number from zero to the number of members in the group.

The availability target counts the number of active members, not the number of applications. If a group member is a nested group, the nested group is one member of the main group.

- **Satisfactory target:** The minimum number of members that must be active for the automation manager to consider the group compound status as SATISFACTORY. The satisfactory target is specified in the customization dialogs. During operations, the satisfactory target can be overridden on an INGGROUP command panel.

The satisfactory target can be null, meaning that no satisfactory target is defined. Both the availability target and satisfactory target can be specified in a delta notation format. For example, the availability target can be equal to -1 (minus one) which means all members except one must be available. A satisfactory target of -1 (minus one) means that the server

group is in a SATISFACTORY state if the count of active members is greater than or equal to one less than the availability target.

- **Preference value:** A number which is a priority value that is used to identify which members are started to reach the availability target. In the customization dialogs, each member is assigned a *base preference* value. Preference values can also be used to define regular and backup members.

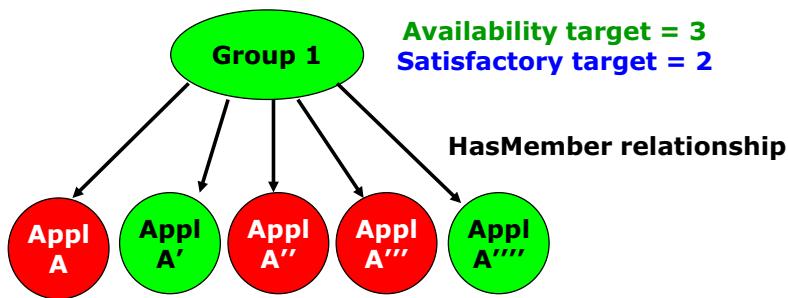
The *effective preference* value is determined by adding a bonus value from the automation manager to any runtime adjustments that are made to the base preference value.

The automation manager bonus of +250 is a result of two bonus values from the manager.

When the automation manager attempts to start applications in a group, it chooses those members with the highest resulting preference values. For example, if you adjust the preference value for an application with a base preference value of 700 by +200. The adjustment of 200 results in a preference value of 900. If that preference value is the highest, and the application is selected to meet the satisfactory target, the automation manager adds a bonus value of 250. The sum of the base value, adjustment, and bonus produces an effective preference value of 1150. The automation manager issues a MakeAvailable vote for the member.

## Behavior of groups with server nature

- Members are typically instances of the same application, such as a web server
- The group status is based on the defined *availability target* and the *satisfactory target*
- Possibly only a subset of the members is started to meet the defined availability target



© Copyright IBM Corporation 2019

3-62

### Behavior of groups with server nature

For a group with a server nature to be available; it can require, all or some of its members to be available. This requirement is based on an availability target which is specified when the group is defined. As an SA z/OS only example, a server group can be defined for a CICSplex application which is not required to run concurrently on every system in the sysplex.

*Availability target* (AVT) defines the optimum number of group members that must be active for the started status of the group to be SATISFACTORY. In this example, the AVT is three.

*Satisfactory target* defines the minimum number of group members that must be active for the started status of the group to be SATISFACTORY. In this example, the satisfactory target is two.

The example server group contains five copies of application A. With SA z/OS, members can be on five different systems. The availability target is 3 and the satisfactory target is 2. When an INGREQ START is issued, automation attempts to activate only three of the five. If more than three are active, then the additional copies are stopped. Preference values determine which three copies of application A start.

An explicit INGREQ command is not required to start and stop each member; the group propagates the votes. Availability target, and preference values determine which members get votes. Stopping a server group is exactly like stopping a basic group. All members receive a MakeUnavailable vote.

## Status values for server groups

Observed group status	Meaning
AVAILABLE	The availability target goal is satisfied or the availability target goal is not satisfied, but the satisfactory target is satisfied
DEGRADED	Some members are available, but less than the number required by the satisfactory target
SOFTDOWN	All members of the group are unavailable
HARDDOWN	

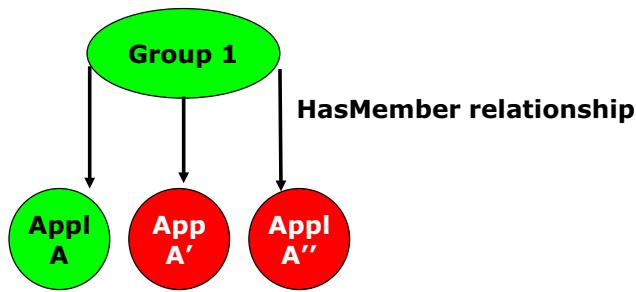
### Status values for server groups

The group status is based on the status of the members as they relate to the availability target and satisfactory target as described on the previous slides.

The **desired available** policy default value (always) sets a **desired status** of available for server groups. Because of this policy, selected members receive MakeAvailable Scope Only (MASO) votes to start; while others receive MakeUnavailable Scope Only (MUSO) votes to stop. Every member of the group receives a vote. The server group is not started when it has a Desired Available policy of **ondemand**.

## Behavior of groups with move nature

- Behaves like a server group with an availability target of 1
- Only one member is started
- Members are typically instances of the same application
- Active member can be changed
- The move mode can be one of these types:
  - Serial: One member is stopped before another is started
  - Parallel: Members are started and stopped simultaneously



© Copyright IBM Corporation 2019

3-64

### Behavior of groups with move nature

A group with a move nature is similar to a server group because it does not need all members to be active for the group to be available. Availability target limitation is the main difference. The availability target of a move group must be one. A move mode definition of serial can be used to control the start of applications when only one instance of the application can be active at any one point in time.

A common implementation of a move group in System Automation for z/OS is to define server groups as the members of the move group. That implementation enables a move of the entire server group between systems.

# Initiating application moves

- External events
  - System leaves the SAplex (SA z/OS only)
  - Application fails or has recovered
- Policy goals
  - INGGROUP command options
    - Exclude
    - Avoid
    - Include
- Manual actions (Targeted moves)
  - Manually move applications from one system to another using
    - INGMOVE: Select the target member to start the application
    - INGGROUP: Adjust preference values for application group members

## Initiating application moves

You can initiate application moves in one of three ways: externally, goal driven, or targeted.

Externally initiated moves can be a reaction to the loss of a system or when a member becomes unusable. Externally initiated moves can also be triggered when a preferred member becomes usable again, *or when a system rejoins the sysplex automated by System Automation for z/OS*.

In goal driven moves, the choice of members might vary. A member can be excluded, avoided, or included when deciding which members to start. Such actions are usually part of a planned move to accommodate a scheduled outage.

Targeted moves occur when group members are chosen manually at run time. The INGGROUP command can be used to modify member preference values.

Instead of the INGGROUP command, the INGMOVE command can be used to initiate immediate or planned moves. Using the INGMOVE command is simpler than calculating the results of adjusting preference values. Application moves apply to both move groups and server groups.

## Summary of group availability attributes

Property	Basic group	Server group	Move group
Is available when	All members are available	Some (availability target) of its members are available	One of its members is available
Is unavailable when	All members are unavailable	All members are unavailable	All members are unavailable
Availability target	Not applicable	Zero or more (availability target can be modified)	One (availability target cannot be modified)
Satisfactory target	Not applicable	From zero to availability target	Not applicable

### Summary of group availability attributes

This slide summarizes the three group natures. Basic groups do not have an availability target. Move groups can have an availability target of only **1**.

Server groups have an availability target and a satisfactory target. The availability target can be a number from zero up to the number of members in the group. The satisfactory target is a subset of the availability target and can be from zero up to the availability target value.

Basic groups do not have an explicit availability target. However, conceptually the availability target equals the number of members in the application group.

—

## Preference values and their effect on group policy

Value	Description
3100	The resource is always selected regardless of the state of the resource.
2900	The resource is always selected unless the observed status of the resource is SYSGONE, or its system is stopping and there is a viable alternative resource. It will still be selected if its system has been excluded or it has an observed status of HardDown.
2400	The resource is always selected unless the observed status of the resource is SYSGONE or HARDDOWN, or its system is stopping and there is a viable alternative resource. It will still be selected if its system has been excluded.
1000	The resource is always started, other resources <= 700 are stopped.
900	The resource is started when the group is started. Members with preference 500 will be stopped in favor of this resource.
700	The resource is started when the group is started and is not stopped unless a higher preference (1000) alternative becomes available.
<600	(Used as a threshold.) The resource with the preference value of 599 and lower is not selected when the group starts. The resource is selected only if the group is in recovery mode.
500	Like 100, however it will be stopped when an alternative with a calculated preference >750 is available.
300	Like 100, however it will be stopped when an alternative with a calculated preference >550 is available.
100	The resource is selected only when the group enters recovery mode. Stopped as soon as better alternative (for example, preference 500 and greater) becomes available.
1	The member is always deselected and always has a MakeUnavailable vote that is propagated to it.
0	The member is passive and never receives any vote that is propagated by the group. It cannot be started or stopped.

© Copyright IBM Corporation 2019

3-67

### Preference values and their effect on group policy

This table summarizes the meaning of several preference values. The preference value affects the role of a resource. The maximum preference value is 3200. Any value greater than 3200 is changed to the maximum.

In addition to the above table, if a resource has an effective preference value that is greater than 2600, it keeps its effective preference value. If the system is SysGone or HardDown and the preference is between 2000 and 2599 then the effective preference value is set to 1 if there is a viable alternative.

Primary group members are members with a base preference value of 600 or more. Backup group members are members with a base preference value of 2 - 599 and are selected only if the group is in *recovery mode*.

Base preference values of zero and one are special cases. A base preference value of one means that the member is always deselected, and always receives a MakeUnavailable vote. This value can be useful if complete manual control of a move process is required.

A base preference value of zero means that the member is a passive member and never receives a vote. If the preference value of a member is changed and the member becomes passive, all votes to the member are withdrawn.

For greater detail, see *IBM System Automation for z/OS: Defining Automation Policy*. See also “Controlling Application Move with Preference Values” in *IBM System Automation for z/OS User’s Guide*.

## Automation manager bonus values

Value	Description
+ 225	The member is AVAILABLE or WASAVAILABLE
+ 220	The member is STARTING, STOPPING or DEGRADED
+ 175	The sticky bonus is on and the adjusted preference is greater than 1000
+ 25	The sticky bonus is on and the adjusted preference is less than or equal to 1000
- 400	The member is SYSGONE and the adjusted preference is greater than 1500

- The *sticky bonus* favors members in the group that were active last
- More than one bonus value can be assigned

### Automation manager bonus values

This table summarizes the bonus values that the automation manager assigns. The purpose of the sticky bonus is to give a higher priority to keeping or restarting a member on a system where it ran before. The sticky bonus is added when a group member is selected. It stays on until another member is selected.

In most cases, the automation manager adds more than one bonus. For example, if an application is available or was available, the manager adds 225 points. Then based on the original base preference value, the manager might also add another 25 or 175 points as the sticky bonus.

Basic group members have a default base preference value of 700 assigned to them. The operator cannot adjust the value. Therefore, the adjusted preference value is always zero. Preference values apply only to move groups and server groups.

If a basic group member is available, the automation manager adds 225 to make it 925:

Effective preference value (925) = base preference value (700) + adjusted preference value (0) + automation manager bonus (225)

If a server group member has an adjusted preference value of 2800, the automation manager assigns it a bonus value of +175. This sticky bonus is added to the base preference value and any adjusted preference value, for a total of 2975. For adjusted preference values of less than 1000, the sticky bonus is +25.

## Effective preference value tie breaker

- When multiple resources have the same effective preference value, one is selected at random
- This selection becomes sticky due to the previously selected bonus
  - A value of +25 or +175 is added

© Copyright IBM Corporation 2019

3-69

### *Effective preference value tie breaker*

If multiple resources have the same effective preference value, the automation manager randomly selects which resource to use.

After an application is selected, it is assigned a sticky bonus of + 175 or + 25, depending on the base preference value:

- + 25 if the base preference value was  $\leq 1000$
- + 175 if the base preference value was  $> 1000$

You can use the INGGROUP command to modify the preference values.

# Group modes

- Normal

- All selected members have preference values of 600 or higher
  - When started, activates only members with preference values 600 or higher

- Recovery

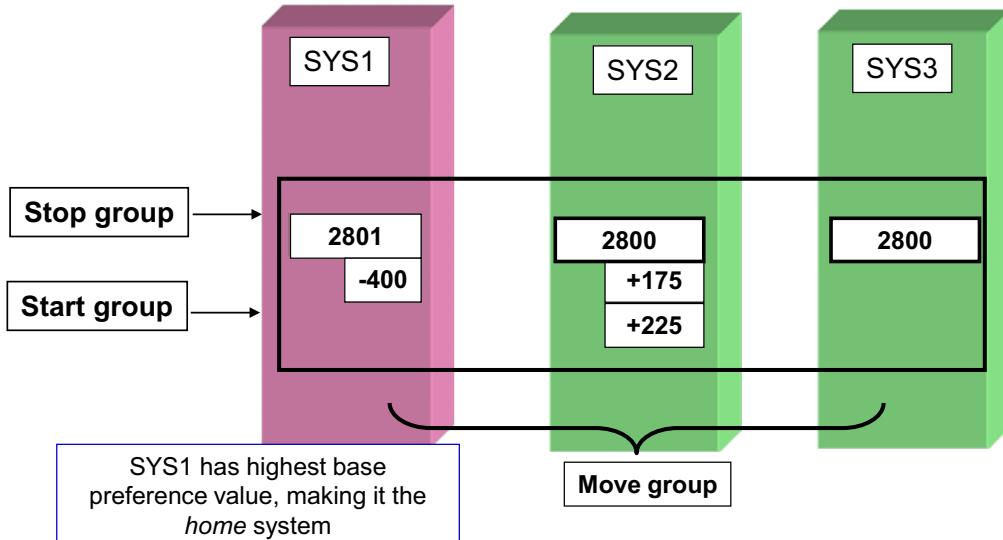
- An active member or its system fails
  - At least one selected member has a preference value less than 600 but greater than one
  - When all started applications have a preference value 600 or higher, the group leaves recovery mode

## Group modes

A group is in Normal mode when all of its active members have a preference value greater than 600. When a group is started, it selects only members with preference values greater than 600.

If an active member or its system fails, the group enters recovery mode. If at least one selected member has a preference value of less than 600, the group is in recovery mode. When all started members have preference values > 600, the group leaves recovery mode.

## Example of a sticky move



The application remains on SYS2 because of the sticky bonus

### Example of a sticky move

This slide shows how an example move group uses the sticky bonus:

1. SYS1 which is defined as the primary system for a resource that is defined in a move group fails. Assume that the member never started. The observed status of the resource is SYSGONE and the preference value becomes 2401 (2801 - 400). Assuming that the member was running on SYS1, the effective preference value becomes  $2801 - 400$  (SYSGONE bonus) + 175 (sticky bonus) = 2576. This effective preference value is less than 3100 (Always selected). The automation manager looks for other members. The effective preference is less than the 2800 defined on the other systems. Another member is picked from one of those systems.
2. The automation manager selects another system for the resource by looking at SYS2 and SYS3. Because both systems have a preference value of 2800, the automation manager randomly picks one. Assume that SYS2 is selected.
3. The resource is started on SYS2, and the preference value becomes 3200 (2800 base + 225 bonus for being active + 175 sticky bonus).
4. Assume that SYS1 is restarted. The preference value is 2801 on SYS1. However, the 2801 is less than the 3200 for SYS2. The resource stays active on SYS2.

5. An operator issues an INREQ REQ=STOP against the move group for the resource. The operator action changes the preference value on SYS2 to 2975 (2800 base + 175 sticky bonus).
6. The move group is restarted later. The preference value on SYS2 is 2975, which is still greater than the 2801 on SYS1. The resource is restarted on SYS2 again.

# Lesson 8 Transient resources

## Lesson 8: Transient resources

- A transient resource must run and complete before dependents can start
  - Examples: CAS9, EZAZSSI, ABENDAID, IXFPbbb
- The following status values are included:
  - RUNNING
  - ENDED
- ENDED is treated as UP for startup
- AUTODOWN is treated as DOWN for shutdown

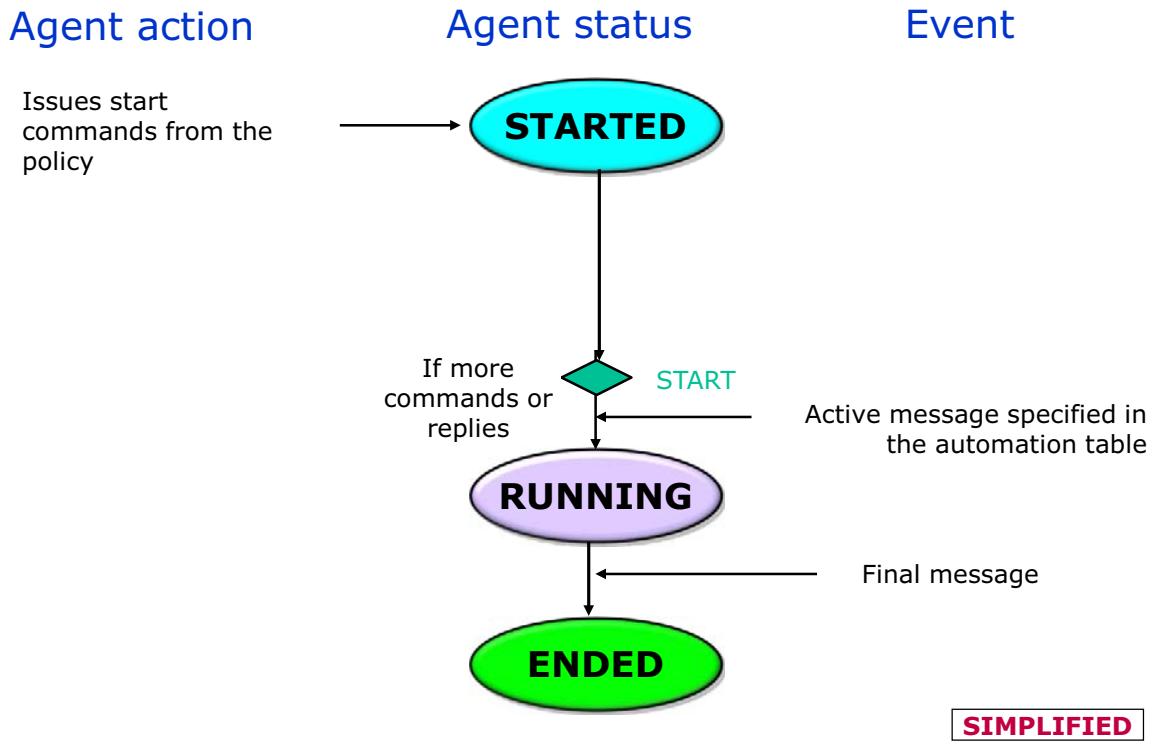
### Transient resources

A special form of dependency is one in which an initialization job must run before a resource can be started. Examples are the initialization CA90s job or the TCPIP initialization job EZAZSSI. These jobs do not stay active, but they do complete normally. Only then is the dependent resource started.

These resources are defined as **transient jobs**. A transient job is handled a little differently by the automation agent. When its UP message is received, the agent status of the transient job is RUNNING instead of UP. When it ends, its agent status is ENDED instead of AUTODOWN. In many cases, the UP message is not defined.

A status of RUNNING for a transient resource does not allow its dependents to start if there is a WhenAvailable condition. It can start if there is a WhenRunning or similar condition. Some transient jobs must run within an IPL lifetime. Others can be run as often as required.

# Startup for transient jobs



© Copyright IBM Corporation 2019

3-73

## Startup for transient jobs

This slide shows a simplified view of agent status values for a transient job.

1. The agent issues start commands from the policy.
2. The agent status becomes STARTED.
3. If there are more commands or replies, they are issued as well.
4. The active message as specified in the automation table updates the agent status to RUNNING.

The final message as specified in the automation table updates the agent status to ENDED which is like the UP status of non transient jobs, so any dependent resources can start.

## Summary

Now that you completed this unit, you can perform the following tasks:

- Describe a monitor resource and its effect on the health status of linked resources
- Describe events and triggers
- List MVS automation
- Describe resource relationships details
- Explain the order process
- List and describe factors that can influence goal-driven automation
- List types and natures of application groups
- Explain behavior or attributes of application groups
- List transient resource automation

### Summary

No that you completes this unit, you can perform the following tasks:

- Describe a monitor resource and its effect on the health status of linked resources
- Describe events and triggers
- List MVS automation
- Describe resource relationships details
- Explain the order process
- List and describe factors that can influence goal-driven automation
- List types and natures of application groups
- Explain behavior or attributes of application groups
- List transient resource automation



# 4 Processor Operations

IBM System Automation for z/OS 4.1



## Unit 4: Processor Operations



© Copyright IBM Corporation 2018

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

4-1

This unit covers the automation product component processor operations.

Processor operations monitors and controls processor hardware resources. It provides a connection from a focal point processor to multiple target processors. With NetView on the focal point system, processor operations uses the Hardware Messages console, the Operating System Messages consoles, or both for monitoring and controlling target processors and target systems.

Processor operations allows you, among others, to power on and off multiple target processors and reset them, perform IPLs, respond to messages and WTORs, monitor status, and detect and resolve wait states.

## Objectives

---

When you complete this unit, you can perform the following tasks related to Processor Operations:

- Describe architecture including implementation options
- Explain usage and operator interface
- Describe automation policy

### *Objectives*

When you complete this unit, you can perform the following tasks related to processor operations:

- Describe architecture including implementation options
- Explain usage and operator interface
- Describe automation policy

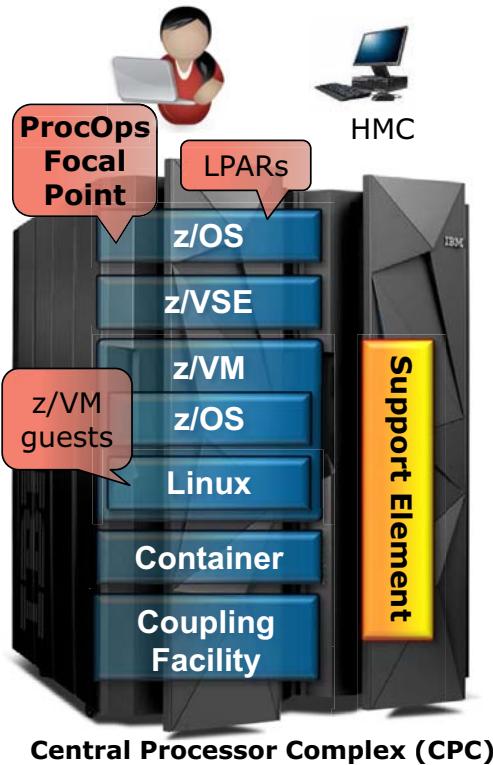
# Lesson 1 Processor Operations architecture

## Lesson 1: Processor Operations architecture

- Overview
- Architecture
- Interfaces

This lesson provides a processor operations overview and explains the architecture and interfaces.

## Component Processor Operations



Central Processor Complex (CPC)

- Single point of control for z Systems
- LPAR startup, recovery, shutdown
- LPAR capacity and weight management
- Hardware status changes and alerts
- Power management
- z/VM guest system support
- NetView and SNMP based
- Secured through SNMP V3 encryption and SAF protected access

© Copyright IBM Corporation 2018

4-4

### Component processor operations

Component processor operations runs on the NetView platform and can control z Systems logical partitions (LPARs), operating systems like z/OS, and virtual machines over the network.

Processor operations can help operators manage more systems with greater efficiency by providing z Systems **external automation** and a **single point of control**, which is easy to configure and easy to use. One operator in one location can initialize, configure, monitor, shut down, and recover multiple systems, both local and remote, and respond to various detected conditions. Automation Control and System Automation for z/OS are the only products that provide one platform for internal and external automation at z/OS run time and at IML and NIP time when internal automation is not yet available.

Highlights include LPAR Capacity Management, power mode control, and support for Linux as a z/VM guest.

Management is done by SNMP connection either to the support element (SE) or the Hardware Management Console (HMC) using the System z Application Programming Interface.

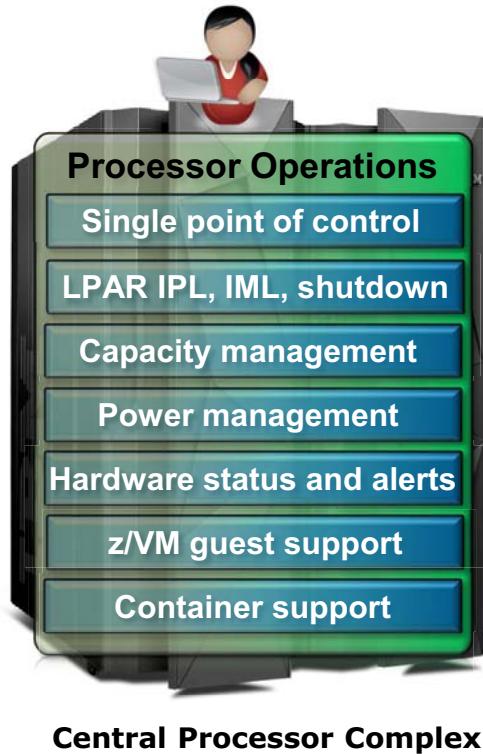
SNMP v2 (System Automation for z/OS 3.4 and older): Security has been the biggest weakness of SNMP since the beginning. Authentication in SNMP Versions 1 and 2 amounts to nothing more than a password (community string) sent in clear text between a manager and agent.

Solution:

SNMP v3 makes no changes to the protocol aside from the addition of cryptographic security:

- Confidentiality: Encryption of packets to prevent snooping by an unauthorized source.
- Integrity: Message integrity to ensure that a packet is not tampered with while in transit including an optional packet replay protection mechanism.
- Authentication: To verify that the message is from a valid source.

## **z Systems external automation and single point of control**



### **External Automation**

- Power-on reset, activate, IPL (NIP time)...
- Respond to system or operator console messages including priority messages (synchronous WTORs)
- Detect and resolve wait states
- CPC capacity changes, powermode control
- Server Time Protocol and LPAR management

### **Easy to use common commands for:**

- CPCs, LPARs, systems, and z/VM guests
- z/OS, Linux, z/VM, z/VSE, container

### **Easy to configure**

- Policy based

© Copyright IBM Corporation 2018

4-5

#### *z Systems external automation and single point of control*

You can perform Power-On reset, activate, initial program load (IPL), respond to hardware messages, monitor hardware status, and detect and resolve wait states.

Hardware alerts and priority messages (synchronous WTORs), some of which must be responded to in a matter of seconds can be automated to increase availability.

Manage LPAR capacity, for instance by adjusting capacity across LPARs and WLM capacity groups automatically.

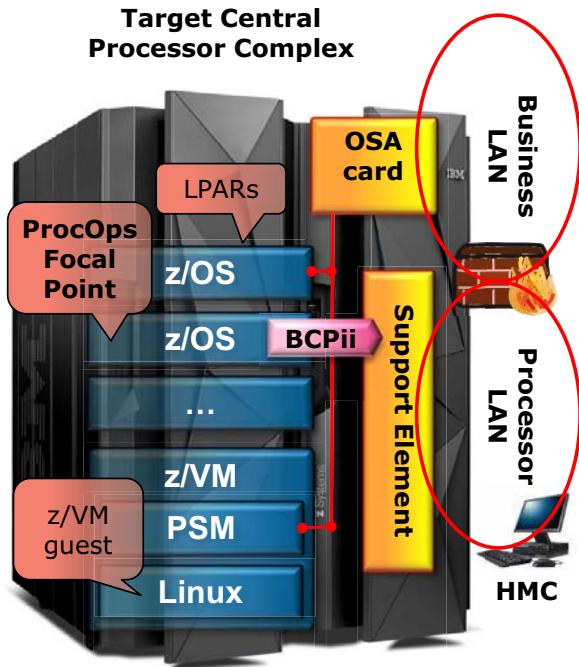
Save energy with power management based on time of day or system load.

Use the Server Time Protocol to improve time synchronization in a sysplex or non-sysplex configuration.

Using one standard interface, the operator can do all that across multiple types of systems. Automated routines for frequently used functions can speed the work of a skilled operator and help less-experienced operators become more productive.

Processor operations are easy to configure using automation policy, which is integrated with system operations.

# Processor Operations building blocks



1. Central Processor Complex
2. Focal point processor
3. Target processor
4. Logically Partitioned (LPAR) mode
5. Support Element
6. Hardware Management Consoles
7. Processor LAN
8. Business LAN and OSA card
9. Communications links
  - a) SNMP
  - b) TCP/IP
  - c) BCPii
10. ProcOps Service Machine (PSM)
11. z/VM guest

© Copyright IBM Corporation 2018

4-6

*Processor operations building blocks*

## Central Processor Complex (CPC)

A physical collection of hardware that consists of central storage, (one or more) central processors, (one or more) timers, and (one or more) channels.

## Processor operations focal point

In a multisystem environment, processor operations needs to be installed only on one system that you can choose, the processor operations focal point. This is where you enter commands you want to process. NetView and SA z/OS are installed on this processor.

## Target processor

The processor that is controlled by a processor operations focal point. It can be a local or remote processor, or a logical partition of a processor. When you send a command from a focal point system, the target is the processor that you want the command to affect. The operating system can be z/OS, z/VM, z/VSE, Linux, zAware, Secure Service Container infrastructure for IBM z Systems, or a coupling facility. If the operating system is z/OS or Linux, SysOps should be installed on the target processor for full automation functionality. If the operating system is z/VM or z/VSE, or if the target processor is a coupling facility, only processor operations functions are available for the target.

## Logically Partitioned (LPAR) mode

A logical subdivision of a CPC. The System z Processor Resource/Systems Manager (PR/SM) facility creates logical partitions and assigns resources, processing capacity, and weights to them. A processor with the PR/SM feature can be divided into partitions with separate logical system consoles that allocate hardware resources among several logical partitions (LPAR). It is called logical because the processor is not physically divided, but divided only by definition. The partitions are defined, monitored, and activated separately by processor operations. The processor operations interface also uses the integrated console of each LPAR for message monitoring and command forwarding. Finally, commands such as ACTIVATE or LOAD are provided with processor operations to manage LPARs.

## CPC Support Element (SE)

The processor hardware interfaces use the Systems Management functions of the SE to operate the CPC and its LPARs. SE customization is required before the processor hardware interfaces can access and use it.

The Support Elements provide the System z Application Programming Interface, which is used to perform hardware commands like LOAD or SYSTEM RESET to control the hardware and hardware images.

## Hardware Management Consoles (HMC)

At least one HMC is attached and operational in a valid CPC environment. The HMC must be customized before the processor hardware interfaces can use the console.

## Processor LAN and Business LAN Customer Network

Together with the CPC and its own Intra Network, a processor LAN environment is available, enabling the SE – HMC communication for one or multiple CPCs. Support Elements are always attached to the processor LAN. An extra HMC, configured for the processor hardware interfaces, might be attached to the Customer Network, the Business LAN. An Open Systems Adapter (OSA) card is the physical network interface used by the LPARs.

## Communications links

Paths that connect the processor operations focal point to target processors and target systems (images) so that commands, messages, and alerts can flow. The types of communication links are described below.

## SNMP Path

The processor operations hardware interface requires a TCP/IP infrastructure for its SNMP-based communication. This means, that in the z/OS LPAR where the automation product is running, an IP stack must be active, with a connection (OSA-card) to the customer network. This enables the communication with an attached HMC. Usually, Firewall/Bridge/Router network components are used to secure and isolate the two networks. This might require addition customization steps to grant TCP/IP based processor hardware interface communication.



**Note:** This is a hybrid interface, allowing you to redirect communications over BCPii instead of TCP/IP. Use hostname ISQET32 as the SNMP IP address for the SE or HMC in the SA PDB processor policy to define BCPii redirection.

### TCP/IP Path

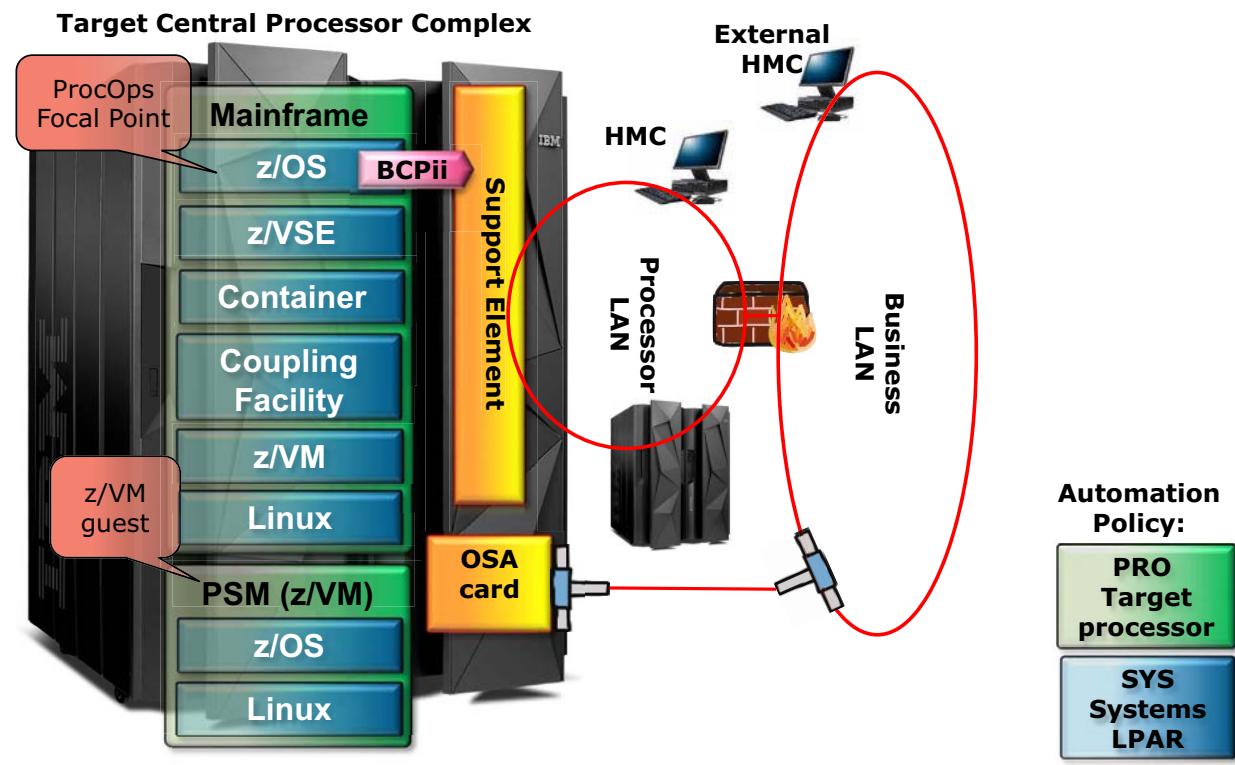
The communication between processor operations and a **z/VM guest** virtual machine is based on TCP/IP socket service. On the z/VM side, the **ProcOps Service Machine** (PSM) component of processor operations provides this support. Similar to the SNMP path, this requires a TCP/IP infrastructure. On the z/VM PSM host side, TCP/IP infrastructure is required.

### INTERNAL Path

The Basic Control Program Internal Interface (BCPii) allows communication between the LPAR Management interface of the automation product and the SE of the local processor. More network elements, such as IP stack, network cards, or routers are not required. For Automation Control, the complete physical connection is located inside the CPC cage.

The BCPii implementation that is used in the automation product is not compatible with the BCPii base component of z/OS. The available z/OS BCPii documentation does not apply for the automation product. Application programs exploiting the z/OS BCPii and the automation product might run concurrently on the same z/OS system.

# Processor Operations architecture



© Copyright IBM Corporation 2018

4-7

## Processor operations architecture

Component processor operations runs on the NetView platform and is integrated with the system operations component through common policy for systems and groups of systems as well as for message automation and distribution to operators.

It is only required on the Focal point processor, for backup purposes, a backup Focal point processor is recommended.

Processor operations does not need any code on the target systems.

Target systems are controlled over one of the Communications links options.

The same console tasks are supported as on the Hardware Management Console, so processor operations can be used as a backup console, however, its more important use is for hardware automation and for external automation of an operating system.

Hardware events and messages, as well as operating system messages, can be automated using automation policy.

Operating system messages are picked up through the system console in the SE, but only if the "Integrated 3270 Console" located on the Hardware Management Console is disconnected, see [page 372](#)

Issuing operating system commands is also possible.

You can choose to receive all messages for a specific target system operator console using the

processor operations Interested Operator List. These messages are displayed at your NetView console session. The ISQXMON command adds you to or deletes you from the interested operator list for a target system console

Messages from the target console that are displayed include new message lines displayed on the Support Element Operating System Messages window, sent by z/OS, z/VM, Linux on System z, z/VSE, Secure Service Container infrastructure for IBM z Systems, or the Coupling Facility Control Code (CFCC). Multiline messages issued by z/OS look like several single-line messages from the console. Similarly, if an z/OS message extends over two lines, it looks like multiple separate messages from the processor operations console.

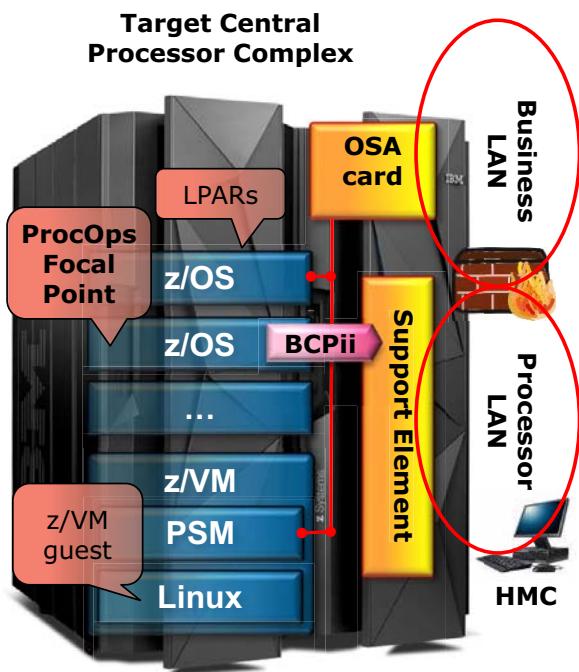
When you are monitoring a system console, make sure it is enabled to receive regular messages. For z/OS, refer to the VARY CONSOLE command parameters ACTIVE, DEACTIVATE.

All processor operations messages from a target system begin with the processor operations identifier ISQ900I or ISQ901I plus the target system name and the console type. Messages prefixed by ISQ900I are not displayed but are used by the automation. User automation should key on the ISQ900I messages. Messages prefixed by ISQ901I are sent to interested operators and should not be used for user automation. Multiline messages appear as multiple messages.

## **z/VM**

On the z/VM side, the ProcOps Service Machine (PSM) component of processor operations allows to manage any z/VM guests. This includes guest operating system message and virtual hardware events and actions like starting guests. z/VM itself cannot be automated through PSM, however, z/VM can be managed as target system as described above.

## Processor Operations architected interfaces



Processor Operations provides the following interfaces:

1. Processor Operations interface through business LAN using:
  - a. SNMP to HMC
  - b. **SNMP to Support Element**
2. LPAR Management interface: BCPii using System z internal communication service (SCLP) to local Support Element
3. TCP/IP to ProcOps Service Machine (PSM) for z/VM guests

© Copyright IBM Corporation 2018

4-8

*Processor operations architected interfaces*

Processor operations provides the following interfaces:

### SNMP Path

The processor operations hardware interface requires a TCP/IP infrastructure for its SNMP-based communication. This means, that in the z/OS LPAR where the automation product is running, an IP Stack must be active, with a connection (OSA-card) to the customer network. This enables the communication with an attached HMC. Usually, Firewall/Bridge/Router network components are used to secure and isolate the two networks. This might require addition customization steps to grant TCP/IP based processor hardware interface communication.

SNMP to Support Element is the preferred option over SNMP to HMC as more functions like the hardware messages are available from the SE.

For HMC, you can define primary/alternate Hardware Management Consoles and SEs.

SNMP v2 (System Automation for z/OS 3.4 and older): Security has been the biggest weakness of SNMP since the beginning. Authentication in SNMP Versions 1 and 2 amounts to nothing more than a password (community string) sent in clear text between a manager and agent.

The solution is SNMP v3, which makes no changes to the protocol aside from the addition of cryptographic security:

- Confidentiality: Encryption of packets to prevent snooping by an unauthorized source.
- Integrity: Message integrity to ensure that a packet is not tampered with while in transit including an optional packet replay protection mechanism.
- Authentication: To verify that the message is from a valid source.

In the customization dialog, specify SNMPv3 YES and the user name and Password to allow the new protocol. Community Name might still be required for compatibility, even if it is not used with SNMP v3. The password should be stored in the SA z/OS Password Store using the INGPW command.

## INTERNAL Path

Geographically Dispersed Parallel Sysplex (GDPS) customers can use a subset of the processor operations command set for target processors and target systems defined with connection type INTERNAL to perform LPAR management. In the SA manual, "Operator's Commands" or processor operations online help (NetView command 'ISQHELP'), section 'Restrictions and Limitations' shows if the command can be used for BCPii connections / LPAR management or not. Asynchronous processor operations commands, for example, LOAD or ACTIVATE are not supported for INTERNAL connections, but SNMP only.

For accessing remote CPCs, the local SE routes the request over the business LAN using a Master HMC. For more information, see the Operator's commands.

## TCP/IP Path

The communication between processor operations and a z/VM guest virtual machine is based on the TCP/IP socket service. On the z/VM side, the ProcOps Service Machine (PSM) component of processor operations provides this support. Similar to the SNMP path, this requires a TCP/IP infrastructure. On the z/VM PSM host side, TCP/IP infrastructure is required. The PSM has a logging feature, which produces log files daily. A process is available to delete the log files automatically to clean up VM minidisk A1 once a day. This is controlled by the ISQPARM DATA option CLEANUP, which also allows to specify the number of days to retain logger files.

## Performance improvements in SA z/OS V3.5:

When ProcOps starts a connection to a CPC, one step is to check ALL images of that CPC (command CPCDATA). With LPAR SCOPE set to DEFONLY in the Processor Information policy, only those images defined as processor target systems are processed which reduces execution time for command CPCDATA significantly. Processor connection start (command ISQXIII) finishes faster because CPCDATA command takes less time. ProcOps common command CPCDATA only processes and display the defined target systems.

When ProcOps starts a connection to a CPC, it also triggers monitoring of that connection at regular intervals. One monitoring step is to perform a CPCDATA command to retrieve data of all

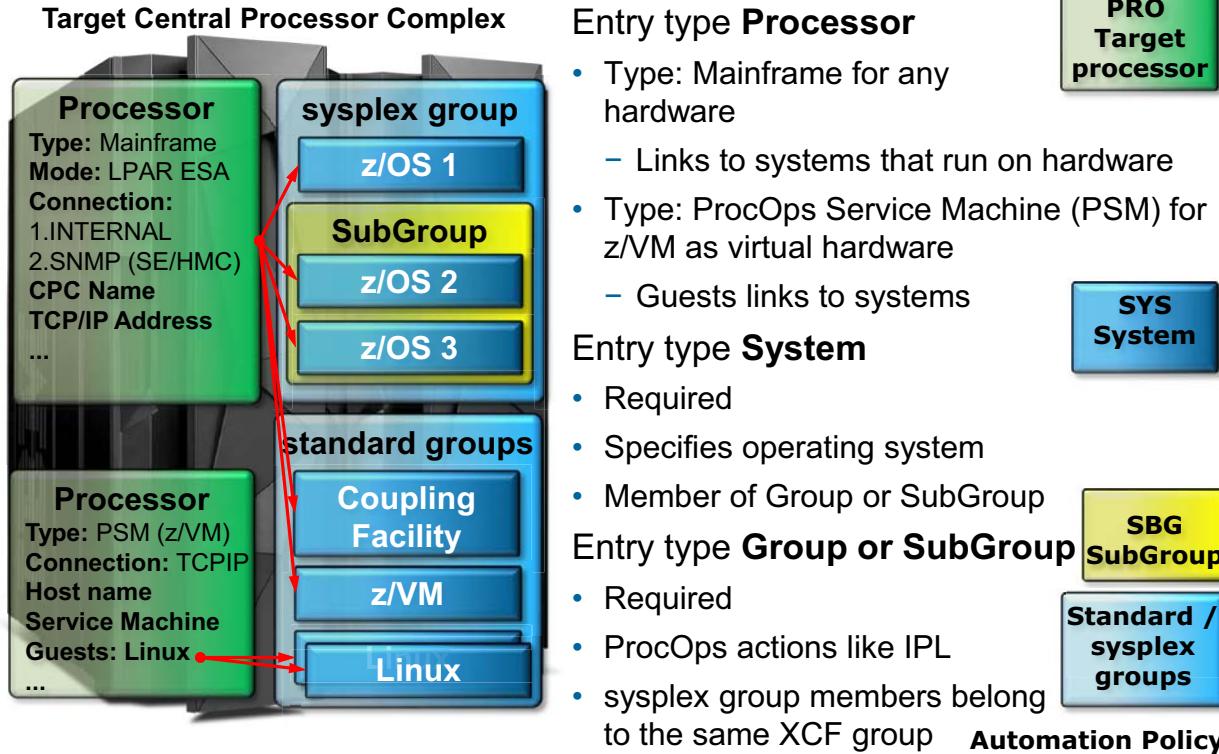
images for checking purposes. With the new path poll option 'Connection Only', only the CPC connection is checked and no CPCDATA is performed which saves a considerable amount of time.



**Important:** If you have system environments that use and depend on automated IPLs, do not use the 'Integrated 3270 Console', defined as a HMCS console, function on the Hardware Management Console at all.

If automation is used to remotely IPL z/OS, the HMCS console (the "Integrated 3270 Console" located on the Hardware Management Console) must be disconnected from the z/OS partition that is to be IPLED, before the IPL is initiated. Failure to do so will prevent automation from receiving messages issued during the IPL and prevent automation from taking any appropriate actions.

## Processor Operations automation policy



© Copyright IBM Corporation 2018

4-9

### Processor operations automation policy

The processor operations automation policy can be defined using the customization dialogs and the same policy database as the system operations component.

Entry types System, Group, or SubGroup are shared with the system operations component.

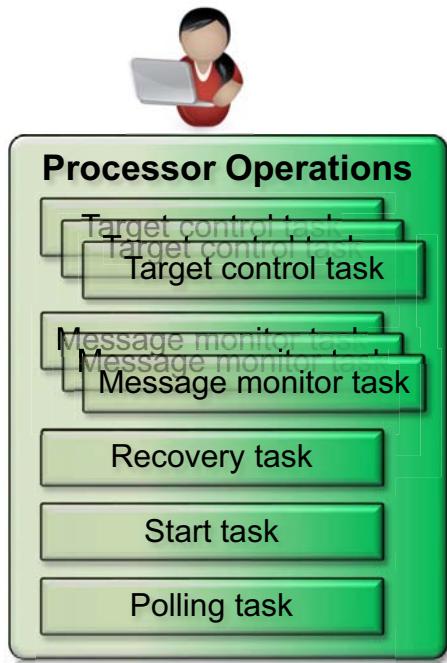
Entry type Processor is the main processor operations automation policy object. Use type Mainframe for LPARs. Use type PSM for z/VM guests.

Entry type Processor specifies the target mode for each defined LPAR (for example ESA, CF, LINUX) and links to Entry type system, which specifies the operating system type (MVS, VM, VSE, LINUX, CF, ZAware). Entry type system must be linked to Entry type Group or SubGroup.

Groups or SubGroups allow ProcOps actions like IPL of multiple systems in parallel with a single command.

A group can be a standard or sysplex group. Sysplex group members belong to the same XCF group.

## Task structure



- Target control tasks
  - Command processing
  - Up to 999: Automatically calculated and set
- Message monitor tasks
  - Used for SNMP, TCP/IP connections only
  - Receive SNMP traps, PSM, and z/VM guest messages
  - Broadcast to the appropriate tasks and operators
  - Up to 999: Automatically calculated and set
- One recovery task
  - Automation for resource control messages
- One start task
  - Establish the processor operations environment
- One polling task to poll the processors

© Copyright IBM Corporation 2018

4-10

### Task structure

Target control tasks do command processing. Up to 999 Target control tasks are possible. Target control tasks are automatically calculated and set.

**Message monitor** tasks perform the following functions:

- Tasks are used for SNMP, TCP/IP connections only.
- Tasks receive SNMP traps, PSM, and z/VM guest messages.
- Tasks broadcast to the appropriate tasks and operators.
- Up to 999 Target monitor tasks are possible. Target control tasks are automatically calculated and set.

One **Recovery** task is available for automation for resource control messages.

One **Start** task is available to establish the processor operations environment.

One **Polling** task is available to poll the processors.

# Lesson 2 Details

## Lesson 2: Details

- Target System status
- Built-in automation: IPL
- Target System Grouping
- Target System status and alerting
- Priority messages (synchronous WTORs)

© Copyright IBM Corporation 2018

4-11

### Details

This lesson covers:

- Target System status
- Built-in automation: IPL
- Target System Grouping
- Target System status and alerting
- Priority messages (synchronous WTORs)

## Target system status (1 of 3)

- **CLOSED**: The target system is closed.
- **DISABLED WAIT**: A target system attention status indicating that a disabled wait condition has been detected.
- **INITIALIZED**: The connection to the SE/HMC was established successfully. The addressed target system LPAR is ready to receive commands over the SYSTEM CONSOLE interface and sends asynchronous events back to ProcOps (such as status changes, HW messages).
- IPL COMPLETE for z/OS: IEE389I message indicates the z/OS is now accepting commands.
- **IPL FAILED**: Error messages were generated during the initial program load process.
- **LOAD FAILED**: The Load or Load Clear operation did not complete successfully.
- **NOT ACTIVE**: The target system's partition (LPAR) is not activated.

© Copyright IBM Corporation 2018

4-12

*Target system status (1 of 3)*

Above values can be displayed on the ISQXDST panel to describe the status of target systems.

## Target system status (2 of 3)

- **NOT OPERATING:** Target hardware indicates a not operating condition for the image. No CP status is available for problem determination.
- **POWER-ON RESET REQUIRED:** A POWER-ON RESET of the target system hardware is required before a BCP load.
- **POWERED OFF:** The target hardware is physically powered off.
- **POWERSAVE:** SNMP Path Only: Power utility for the target hardware failed. Target hardware put in power save state.
- **SERIOUS ALERT:** SNMP Path Only.
- **SERVICE:** SNMP Path Only: Service status enabled for the target hardware.
- **SNMP SESSION PROBLEM:** SNMP Connection returned an error.
- **STAGE-1 ACTIVATE COMPLETE:** The support element power-on reset has occurred but the load is not yet complete.

## Target system status (3 of 3)

- **SUSPENDED:** The connection path to the target hardware of the target system has been suspended.
- **UNDECIDABLE:** SNMP Path Only: The target hardware is not communicating with the support element.
- **UNKNOWN:** No attempt has been made to initialize or activate this target system. This status can be set only by a cold start of processor operations.
- **WAITING FOR IEA101A START MESSAGE:** (Yellow or red) MVS only: Waiting for the IEA101A message.
- **WAITING FOR IEA347A MESSAGE:** (Yellow or red) MVS only: Processor operation has replied to the IEA101A message and is waiting for the IEA347A message.
- **WAITING FOR VM START MESSAGE:** (Yellow or red) z/VM only: Waiting for the first VM IPL message on the operator console.

## **Built-in automation: IPL**

```
COMMANDS HELP
-----
MVS Target System IPL Information
Command ==> _____
Entry Type : System          PolicyDB Name   : JHSAPLEX_V320
Entry Name  : KEYA           Enterprise Name : ACME
Enter responses to the following messages for the target system.
IEA101A specify system parameters
R 00,101A _____
IEA347A specify master catalog parameter
R 00,347A _____
IEA213A or IEA214A DUPLICATE VOLUME
1111 2222 _____
Provide default response when neither device is specified . . . YES
```

- IPL operator prompts can be predefined in the IPL INFO policy for a defined target system
  - At Procps runtime, these definitions can be overwritten using the ISQXOPT command
  - Also supported for z/VM

© Copyright IBM Corporation 2018

4-15

## *Built-in automation: IPL*

For z/OS the following settings can be used:

- IEA101A specifies system parameters

Enter the desired response to the IEA101A message. All IEA101A message responses begin with the characters R 0. You do not have to begin your response with these characters, they are provided by SA z/OS automation.

Quotation marks are not necessary. The customization dialog does not verify your entry.

- IEA347A specifies master catalog parameter

Enter the desired response to the IEA347A message. The customization dialog does not verify your entry.

- IFA213A/IFA214A DUPLICATE VOLUME

Specify the device addresses to be kept online if duplicate DASD volumes exist on a z/OS system at IPL time. You have 24 entries to specify real or generic DASD device addresses. Example for a generic device address: 012\*. This indicates that all addresses from 0120 to 012F must remain online.

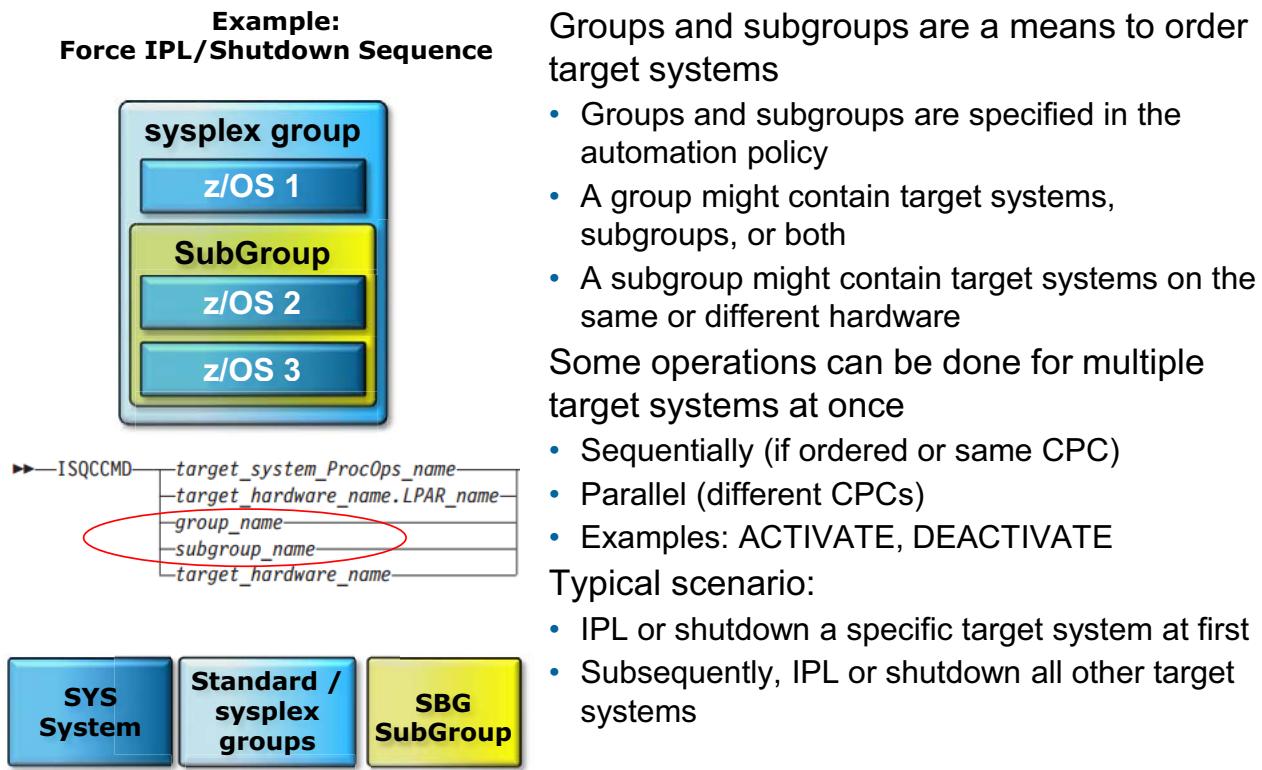
This entry works with the automatic response selection.

Provide default response when neither device is specified. Specify YES or NO to indicate whether a default response is required if neither or both of the devices in the IEA213A/IEA214A

message match those identified as required devices. For IEA214A messages, the current IPL (SYSRES) device is considered as being implicitly defined. If YES is selected, an automated response is always provided. For a SYSRES device, the device shown in the message is dismounted. Otherwise, the following applies:

- If neither device is in the reply list, the first device is unmounted.
- If only one device is in the reply list, the remaining devices are unmounted.
- If both devices are in the reply list, the first device in the list as scanned from left to right remains mounted and the other device is unmounted.
- If both devices are in the reply list due to the same generic device mask in the list as scanned from left to right, the second device is unmounted. A selection of NO results in a response allowing the IPL to continue only if one device is identified as being required. Otherwise, message ISQ1203 is issued and the IPL process is suspended. In such a case, the operator needs to react to system message ISQ1203. A selection of NO and a generic device address of \*\*\*\* always stops the IPL process if a duplicate volume is detected.

## Target system grouping



Groups and subgroups are a means to order target systems

- Groups and subgroups are specified in the automation policy
- A group might contain target systems, subgroups, or both
- A subgroup might contain target systems on the same or different hardware

Some operations can be done for multiple target systems at once

- Sequentially (if ordered or same CPC)
- Parallel (different CPCs)
- Examples: ACTIVATE, DEACTIVATE

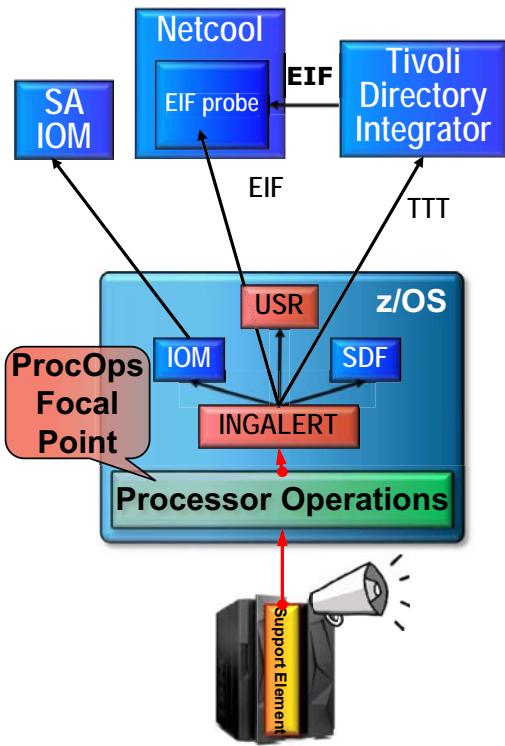
Typical scenario:

- IPL or shutdown a specific target system at first
- Subsequently, IPL or shutdown all other target systems

*Target system grouping*

The ISQCCMD supports the specification of group and subgroup names.

## Target system status and alerting



Alerts on processor operations panels:

- Target hardware problem
- Wait state detected
- Console lost
- Console access lost
- Communications task failed
- SE/HMC Hardware messages: Power problems, thermal problems, running on battery power

Examples of some of the initialization statuses

- Stage-1 Activate complete
- IPL complete, IPL failed
- Load failed
- Waiting for IEA101A Start Message (z/OS)

© Copyright IBM Corporation 2018

4-17

### *Target system status and alerting*

Escalating hardware alerts can avoid outages.

On the processor operations Focal Point system, you can use the automation policy to trap hardware alerts and to either automate them, if possible, or to escalate using the SysOps alerting capabilities provided by INGALERT.

INGALERT can display messages on SDF, escalate through System Automation for Integrated Operations Management, or send alerts to Netcool OMNIbus or other Event Integration Facility (EIF) receiver.

Events can be sent to the Tivoli Enterprise Portal also.

System Automation for Integrated Operations Management (SA IOM) escalates using SMS text messages with confirmation.

Using TTT in the automation policy Inform List sends alerts to Tivoli Directory Integrator, which in turn can open trouble tickets in supported problem management tools. Tivoli Directory Integrator can also send events to the Tivoli Enterprise Portal using EIF.

## Priority messages (Synchronous WTORs)

A synchronous WTOR indicates a critical situation which requires immediate action. Synchronous WTORs have a reply id of '00'; reply requires priority flag set. See the 'Priority' check box on the SE's Operating System Messages window:



ProcOps solution:

- Command ISQSNDH can send requests to the operating system with priority flag set. Example: Reply to message IEA394A (Server has lost connection to time source):
  - ISQSNDH target\_system\_name OC R 00,RETRY
- Check incoming Operating System messages if it's a priority message and flag ISQ90x messages with new qualifier OCP – non priority message has qualifier OC – to ease message automation

© Copyright IBM Corporation 2018

4-18

### Priority messages (Synchronous WTORs)

Some priority messages must be answered in a matter of seconds. Synchronous WTOR means that the system is stopped until the reply has been answered. Delay in replying affects other systems in the sysplex.

Synchronous messages are WTOR messages that can be issued during initialization or recovery situations. When a synchronous WTOR is issued, message CNZ4215W is issued to all active MCS consoles on the system issuing the synchronous WTOR. The highlighted WTO will indicate:

- the message id of the synchronous WTOR for which a reply is needed
- on which console the synchronous WTOR is currently being displayed
- if the message is being managed by the z/OS® auto-reply function.



**Important:** You must respond to synchronous WTOR messages promptly. Synchronous WTOR messages (such as IOS115I and IEA367A) prevent the system from updating its status on the sysplex couple data set. This, in turn, can lead to Sysplex Failure Management (SFM) deciding that the system is not responding normally, and removing it from the sysplex.

If you have a standard response to synchronous WTOR messages such as these, you may want to automate the reply to these messages to avoid delay and avoid SFM partitioning the system out of the sysplex.

Note that MPF and typical automation products are unable to automate a response to these messages. However, Auto-reply is able to automate these messages and, of course, SA ProcOps.

- A Synchronous WTOR is a high priority message with a reply id “00”.
- Reply to a Sync WTOR must be flagged as high priority command.
- A Sync WTOR must be replied to prior to non-priority WTORs.

A synchronous WTOR is displayed:

- Only on consoles attached to the system that issues the WTOR
- Only on one operator console at a time (more on this later)
- WTOR is moved to another console (on the same system) if a reply is not given within ~2 minutes
- Reply is only accepted from the console on which the WTOR is currently displayed
- Operator consoles not involved with the WTOR process appear “hung”
- BLW004A RESTART INTERRUPT DURING jobname stepname ASID= asid ...

Important Synchronous WTORs include:

- IEA015A THIS SYSTEM HAS LOST ALL CONNECTION TO THE SYSPLEX TIMER
- IEA367A MULTIPLE CONSOLE SUPPORT INOPERATIVE ERROR CODE = xxxx REPLY WITH ANY CHARACTER TO CONTINUE WITHOUT MULTIPLE CONSOLE SUPPORT
- IEA394A THIS SERVER HAS LOST CONNECTION TO ITS SOURCE OF TIME.
- IEA500A RESTART INTERRUPT DURING jobname stepname ASID=aaaa
- IEA502A RESTART REASON COULD NOT BE OBTAINED. REPLY WITH RESTART

# Lesson 3 Processor Operations scenarios

## Lesson 3: Processor Operations scenarios

- Processor Operations scenarios overview and details
- Status Display Facility
  - ProcOps views are generated by ISQSTART
- Processor Operations usage areas
- Hardware management console tasks

© Copyright IBM Corporation 2018

4-19

### *Processor operations scenarios*

This lesson covers:

- Processor operations scenarios overview and details
- Status Display Facility
  - ProcOps views are generated by ISQSTART
- Processor operations usage areas
- Hardware management console tasks

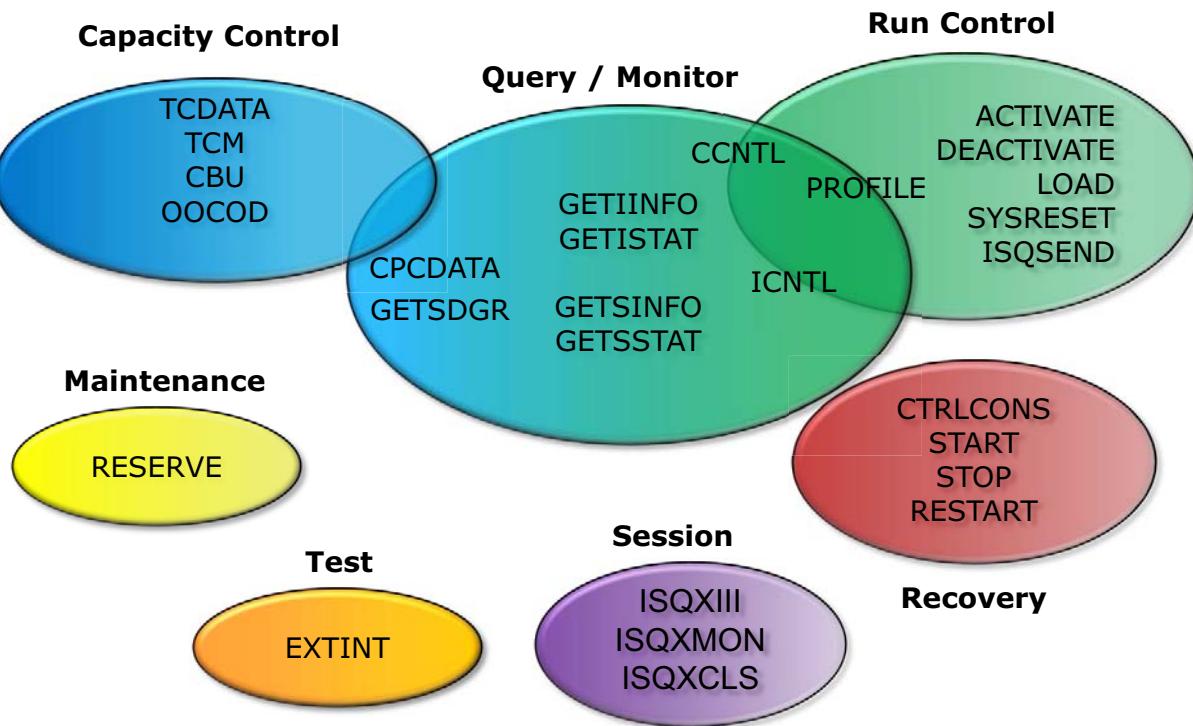
## Frequent Processor Operations scenarios

- Managing mainframe connections with different security demands
- HMC security logs: Automating and controlling the mainframe (SE/HMC)
- LPAR Management
- CPC Management
- Hardware events
- Managing System z appliances, Containers, or Coupling Facilities
- Automating and controlling z/VM guests on System z
- Automating Operating System image messages from the SE/HMC or z/VM guest
- Conducting IPLs (z/OS, Linux and others)
- Moving Operating System images (Systems) between mainframes/LPARs

### *Frequent processor operations scenarios*

These are frequent processor operations scenarios. They are described in more details in the Hardware Management Console tasks.

## Processor Operations usage areas



© Copyright IBM Corporation 2018

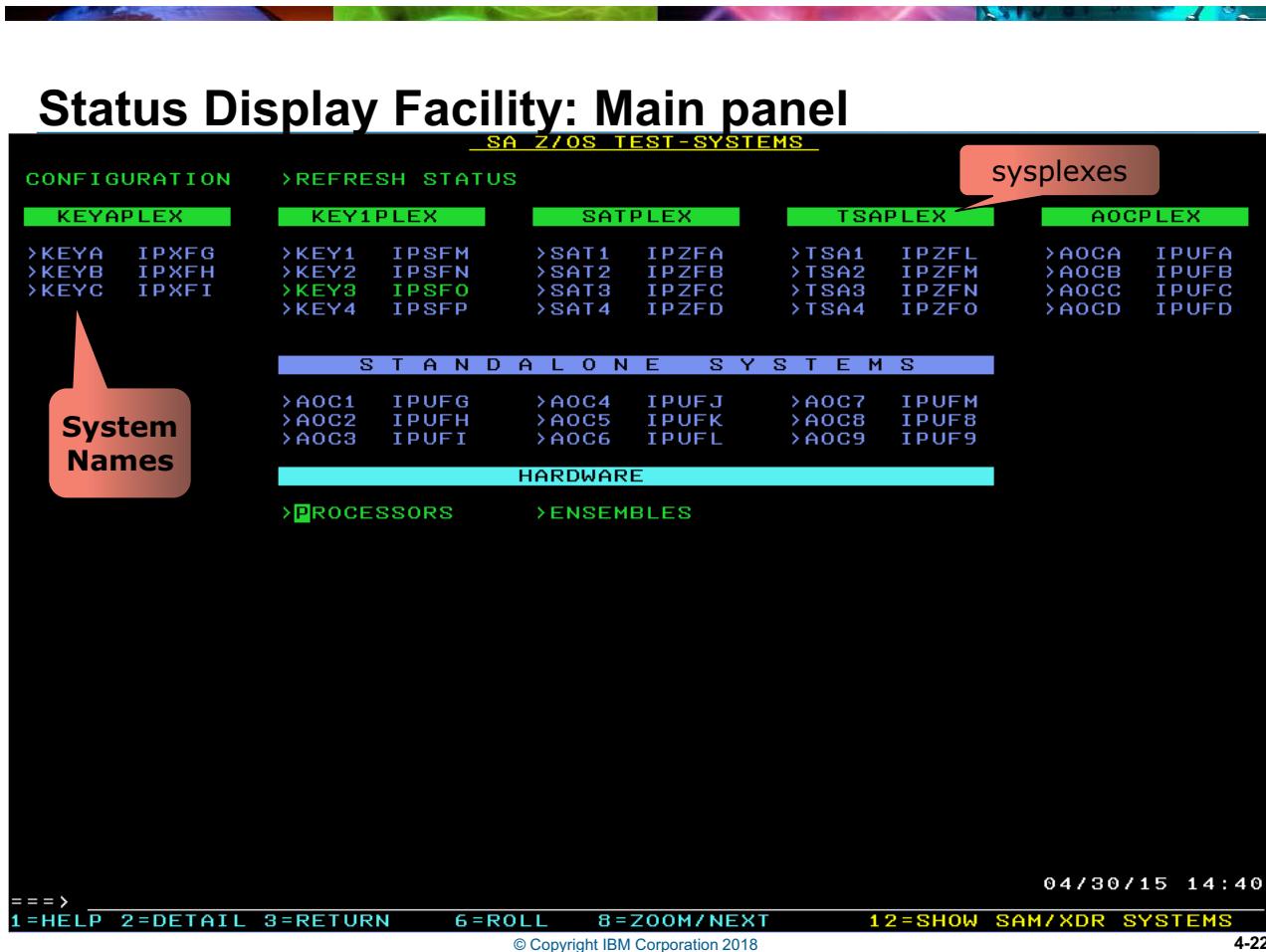
4-21

### Processor operations usage areas

The processor operations usage areas are:

- Capacity Control
- Query / Monitor, which overlaps slightly with Run Control
- Run Control for hardware actions like IML or IPL
- Reserve hardware for maintenance
- Test
- Session management
- Recovery

All the listed commands are documented and available for client use.



Status display facility: Main panel

This is an example of a Status Display Facility Main panel showing hardware resources managed by ProcOps.

SDF is the abbreviation of Status Display Facility. It consists of a set of hierarchical panels, which are delivered as samples.

The system administrator can also define user-specific panel layouts.

If the status of an object changes, it is dynamically reflected on the panels and is propagated up the hierarchy.

With SDF you can monitor the status of applications, application groups, monitor resources, WTORs, and even IBM Workload Scheduler applications.

The setup of SDF is easy and is based on the delivered samples and generation of panels.

With the F8 key, you can zoom to the next lower level and the F2 key shows you details about a status item.

## Status display facility

LPARs for #ECL3		
LPAR	ProcOps Status	Details
XDR2	CLOSED	SYSTEM=XDR2 MODE=ESA OSTYPE=MVS
XDR3	CLOSED	SYSTEM=XDR3 MODE=ESA OSTYPE=MVS
KEY3	INITIALIZED	System=KEY3 Mode=ESA OStype=MVS
VMTSA	INITIALIZED	System=BOEVMTSA Mode=ESA OSType=VM

04/30/15 14:45  
1=HELP 2=Detail 3=RETURN 6=ROLL 7=UP 17=ISQXIII 18=ISQXCLS 23=ISQXDST 24=TGT view  
© Copyright IBM Corporation 2018 4-23

*Status display facility*

ProcOps views are generated by ISQSTART.

This SDF panel displays the LPARs for a selected CEC.

LPAR names, status, and details are displayed.

## Status display facility: Target hardware

Target Hardware

THW	Name	Status	Details
#S35	OK	Type=2964-N96 Site='Bldg_19'	
PSM1	OK	Type=PSM Site='BB-Lab_Bldg_12/13'	
PSM2	OK	Type=PSM Site='BB-Lab_Bldg_12/13'	
PSMVM91	OK	Type=PSM Site='BB-Lab_Bldg_12/13'	
#ECL3	OK	Type=2097-E26 Site='BB-Lab_Bldg_12/13'	
#GRY2	OK	Type=2817-M32 Ensemble=GRY2ZBX Site='Block_1'	
#P57	OK	Type=2827-H43 Ensemble=GRY2ZBX	
#ECL2	OK	Type=2097-E40 Site='BB-Lab_Bldg_12/13'	
#T99	OK	Type=2094-S18 Site='BB-Lab_Bldg_12-13'	
#P35	OK	Type=2827-H66 Site='Bldg_19'	
#G14	UNKNOWN	TYPE=Mainframe	

04/30/15 14:44

====> 1=HELP 2=Detail 3=RETURN 6=ROLL 7=UP 8=ZOOM  
17=ISQXIII 18=ISQXCLS 23=ISQXDST 24=THW view

© Copyright IBM Corporation 2018 4-24

The diagram shows three callout boxes pointing to specific parts of the SDF panel:

- A red callout box labeled "Target Hardware Name" points to the list of hardware names on the left.
- A red callout box labeled "Status" points to the "Status" column in the table.
- A red callout box labeled "Details" points to the "Details" column in the table.

### Status display facility: Target hardware

This SDF panel displays the defined target hardware.

Target hardware names, status, and details are displayed.

# ProcOps messages in netlog

Hardware messages, example for deletion of messages on HMC

```
STATMON.BROWSE ACTS NETWORK LOG FOR 04/29/15 (15119) COLS 017 174 15:02
 DOMAIN: IPSFO SCROLL => CSR
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
IPSFO 22:13:40 U ISQ9001 #S35 SC A0FA0400 04-29-2015 22:13:40:043 A logically partitioned mode event occurred. Select Details for information.
IPSFO 09:45:11 U ISQ9001 #S35 SC A0FA0400 04-30-2015 09:45:11:529 A logically partitioned mode event occurred. [Problem # 1482]
IPSFO 09:52:14S U ISQ9001 #S35 SC A0FA0400 04-30-2015 09:52:14:522 Disabled wait. [Problem # 1483]
IPSFO 09:53:01 U ISQ9001 #S35 SC A0FA0400 04-30-2015 09:53:01:539 A logically partitioned mode event occurred. Select Details for information.
IPSFO 09:54:06 U ISQ9001 #S35 SC A0FA0400 04-30-2015 09:54:06:374 Disabled wait. [Problem # 1483]
IPSFO 09:59:53 U ISQ9001 #S35 SC A0FA0400 04-30-2015 09:59:53:141 A logically partitioned mode event occurred. Select Details for information.
IPSFO 10:09:24 U ISQ9001 #S35 SC A0FA0400 04-30-2015 10:09:24:353 A logically partitioned mode event occurred. Select Details for information.
IPSFO 10:09:40 U ISQ9001 #S35 SC A0FA0400 04-30-2015 10:09:40:538 A logically partitioned mode event occurred. Select Details for information.
IPSFO 10:12:38 U ISQ9001 #S35 SC A0FA0400 04-30-2015 10:12:37:971 A logically partitioned mode event occurred. Select Details for information.
IPSFO 10:53:04 U ISQ9001 #S35 SC A0FA0400 04-30-2015 10:53:04:762 A logically partitioned mode event occurred. Select Details for information.
IPSFO 11:09:13 U ISQ9001 #S35 SC A0FA0400 04-30-2015 11:09:13:654 A logically partitioned mode event occurred. Select Details for information.
IPSFO 11:12:14 U ISQ9001 #S35 SC A0FA0400 04-30-2015 11:12:14:430 A logically partitioned mode event occurred. Select Details for information.
IPSFO 11:13:43 U ISQ9001 #S35 SC A0FA0400 04-30-2015 11:13:43:096 A logically partitioned mode event occurred. Select Details for information.
IPSFO 11:13:57 U ISQ9001 #S35 SC A0FA0400 04-30-2015 11:13:57:411 A logically partitioned mode event occurred. Select Details for information.
IPSFO 11:20:24 U ISQ9001 #S35 SC A0FA0400 04-30-2015 11:20:24:584 A logically partitioned mode event occurred. Select Details for information.
IPSFO 11:20:54 U ISQ9001 #S35 SC A0FA0400 04-30-2015 11:20:53:894 A logically partitioned mode event occurred. Select Details for information.
IPSFO 11:27:32 U ISQ9001 #S35 SC A0FA0400 04-30-2015 11:27:31:786 A logically partitioned mode event occurred. Select Details for information.
IPSFO 12:07:26 U ISQ9001 #ECL3 SC A0FA0400 04-30-2015 12:07:26:26 External Time Source contact via Network Time Protocol (NTP) message
IPSFO 12:17:25 U ISQ9001 #ECL3 SC A0FA0400 04-30-2015 12:17:25:75 External Time Source contact via Network Time Protocol (NTP) message
IPSFO 12:21:28 U ISQ9001 #S35 SC A0FA0400 04-30-2015 12:21:28:15 A logically partitioned mode event occurred. Select Details for information.
IPSFO 12:40:13 U ISQ9001 #S35 SC A0FA0400 04-30-2015 12:40:13:182 A logically partitioned mode event occurred. Select Details for information.
IPSFO 13:23:11 U ISQ9001 #S35 SC A0FA0400 04-30-2015 13:23:10:831 A logically partitioned mode event occurred. Select Details for information.
IPSFO 13:33:53 U ISQ9001 #S35 SC A0FA0400 04-30-2015 13:33:53:125 A logically partitioned mode event occurred. Select Details for information.
IPSFO 14:03:31 U ISQ9001 #S35 SC A0FA0400 04-30-2015 14:03:31:655 A logically partitioned mode event occurred. Select Details for information.
IPSFO 14:29:08 U ISQ9001 #S35 SC A0FA0400 04-30-2015 14:29:07:874 A logically partitioned mode event occurred. Select Details for information.
IPSFO 14:30:03 U ISQ9001 #S35 SC A0FA0400 04-30-2015 14:30:03:172 Disabled wait. [Problem # 1484]
***** END OF LOG ****
```

© Copyright IBM Corporation 2018

4-25

## ProcOps messages in netlog

This screen capture shows the NetView netlog.

From left to right, you see the NetView domain, a time stamp, the message type, in this case U, the origin of the message, #S35 as the target hardware name, SC for system console, message ID, original time stamp followed by the message text.

You can easily scroll the netlog in all directions and also easily filter, search, and archive the netlog. The NetView netlog is also part of the NetView CANZLOG.

Here you see hardware messages that are an example for deletion of messages on HMC.

## Getting started

Processor Operations is started on the focal point system

Syntax: ISQSTART ACF

Stop ProcOps

Syntax: ISQSTOP

Restart ProcOps to activate configuration changes

Syntax: ISQSTOP

ISQSTART ACF

### *Getting started*

You use the ISQSTART command to start processor operations on the focal point system.

Issuing ISQSTOP in NetView on the focal point system stops ProcOps.

## Target system overview: ISQXDST

The screenshot shows the ISQXDST status panel. At the top, it displays:

- ISQSUM SA z/OS
- oc-Ops Target Status Summary
- Updates: Dynamic
- NMC Bridge: INACTIVE
- Debug Mode: OFF

Below this is a table of target system status:

Cmd	Target System	Status	Focal Points	Primary:	Backup:
I	isqxi	C	isqxcls	O	isqxopt
	BOEVMTSA (*)	INITIALIZED			
	KEYA	INITIALIZED			
	KEYB	INITIALIZED			
	KEY1	INITIALIZED			
	KEY1CFF	INITIALIZED			
	KEY1CF1	INITIALIZED			
	KEY2	SNMP SESSION PROBLEM			
	KEY6	INITIALIZED			
	KEY6cff	CLOSED			
	KEY6cf1	INITIALIZED			
	KEY7	IPL COMPLETE			
	LNXT1	UNKNOWN			

At the bottom, there are function keys:

- Enter=Static PF1=Help PF3=Exit PF4=Tgt Sys Summary PF5=Debug On/Off
- PF6=Roll PF7=Up PF8=Down PF9=Tgt HW Summary PF11=PATH Details PF12=Quit

© Copyright IBM Corporation 2018

4-27

### Target system overview: ISQXDST

Processor operations allows you to monitor target processors, target systems, and focal point communication path by using the ISQXDST status panels of your NetView operator console. These panels are available on a NetView operator console where ProOps is started.

The ISQXDST command displays status panels, which provide the following information:

- Status summary (all target systems)
- Individual target hardware status (Updated dynamically)
- Individual target hardware connection status (Updated dynamically)
- Individual target system status
- Interested operator list

The PF11 key displays the Path Detail panel, which provides detailed status information about a specific NetView path. First, place the cursor on the line with the name of the path that you want more information about, and then press PF11.

The PF9 key displays the Target Hardware Summary panel. This panel provides detailed status information about the target hardware that the target system is defined on.

Line commands are available to start and close the session to the target hardware to display target system IPL-options, netlog, and events.

You can change the status from INITIALIZED, LOAD FAILED, or IPL FAILED to IPL COMPLETE by issuing the ISQVARS command to change the internal variable tstat.

You would want to do this when you perform a cold start of processor operations while a target system is already running. After the cold start, you issue the ISQXIII command to the target system that is already running so that its status becomes INITIALIZED, then you change the value of tstat to IPL COMPLETE.

You can also want to change the status manually if the cause of the LOAD FAILED status was corrected by using the pass-through facility of processor operations, or if it was corrected locally at the site of the target system.

You can also issue the ISQVARS command to change the internal variable tstat from IPL COMPLETE to IPL FAILED. Processor operations sets the status to IPL COMPLETE when it initiates a load of a target system and receives an operating system-specific message that indicates that the operating system received a level where it is ready to work. However, neither of these operating systems provide a sufficient indication that the load process failed, preventing processor operations from changing the target system status appropriately.

.

## Target system summary

```
ISQETARG          SA z/OS - Proc-Ops Target System Summary Updates: Dynamic

Target System
  Name       : KEY3
  Description :
  O.S.      : MVS
  Load Profile :
  Status    : INITIALIZED
  Attention  :

Target Hardware
  Name       : #ECL3           Mode : LPAR
  Status     : OK
  Path Status: ACTIVE

Target Hardware LPAR
  Name       : KEY3           Mode : ESA
  Image Profile: KEY3

Last Significant Message:

Enter=Static PF1=Help PF3=Exit PF6=Roll PF7=Oper List PF8=Guests
PF9=Target Hardware PF11=Path Detail PF12=Quit
```

© Copyright IBM Corporation 2018

4-28

*Target system summary*

This panel has the following PF keys:

- The PF7 key displays the processor operations Interested Operator List panel (the one accessed with the ISQXMON command).
- The PF9 key displays the Target Hardware Summary panel. This panel provides detailed status information about the target hardware that the target system is defined on.
- The PF11 key displays the connection Path Detail panel. This panel provides detailed status information about a specific connection path.

**Target Hardware Name:** The name assigned to this definition of target hardware in the customization dialog.

**Target Hardware Type:** Indicates the machine type of a z Systems processor.

**Target Hardware Mode:** LPAR, ESA

**Target Hardware Description:** Short textual description of this definition of target hardware, defined in the customization dialog.

## Hardware communication sessions

The ISQXIII-command initializes a target system and establishes a hardware connection (if required) between the systems provided here:

- Processor Operations focal point system
- Target hardware

Hardware sessions are monitored as specified in the automation policy

- Processor Operations automatically recovers (# of retries times) sessions that do not respond to the path polling request

```
Path Poll Frequency . . . . . 30          (0 to 99 minutes)
Path Poll Retries . . . . . 2             (0 to 99)
```

Hardware sessions remain up until Processor Operations terminates

### *Hardware communication sessions*

The ISQXIII command initializes (or reinitializes) a target system, or a target system connection. When initializing a target system, ISQXIII establishes a connection to the target processor and enables its console connections.

When initializing a target hardware, ISQXIII establishes a connection to the target processor and enables the console connection for all target systems defined that run on the specified target hardware.

Hardware sessions are monitored as specified in the automation policy.

Processor operations automatically recovers (# of retries times) sessions that do not respond to the path polling request.

Hardware sessions remain up until processor operations terminates.

## Sample task: Issue operating system commands

### Supporting operating systems

- All control programs with console integration facility enabled
- z/OS, z/VM, z/VSE, Linux on System z, Container, and Coupling Facility Control Code
- Some stand-alone utilities such as SADMP
- Syntax: ISQSEND target\_system OC command

Example: ISQSEND KEY6 OC D IPLINFO

```
ISQ901I SA02.KEY6 SC AOFA0200 BCPCMD CC(00000000)
ISQ901I SA02.KEY6 OC IEE254I 17.14.58 IPLINFO DISPLAY 111
ISQ901I SA02.KEY6 OC SYSTEM IPLED AT 19.56.56 ON 01/07/2017
ISQ901I SA02.KEY6 OC RELEASE z/OS 01.09.00 LICENSE = z/OS
      ...
ISQ901I KEY6      SC ISQ417I BCPCMD STATUS(SUCCESS)
ISQ017I ISQSEND completed successfully.
```

© Copyright IBM Corporation 2018

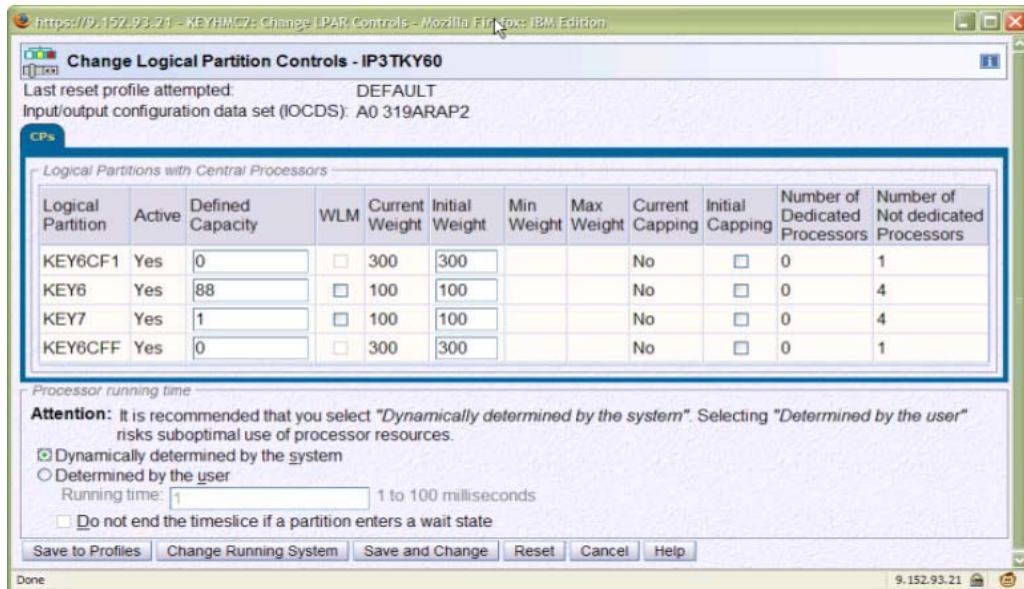
4-30

### Sample task: Issue operating system commands

- The ISQSEND command sends the command to a target operating system for processing, using the processor operations connection to the processor hardware.
- Because the connection to a target hardware and the operating systems running on that hardware is available at target system initialization time or system shutdown time, the ISQSEND command can be used to respond to IPL prompt messages or to answer outstanding replies at system shutdown time.
- The ISQSEND command addresses the operating system running on a processor hardware. The command ISQCCMD addresses the processor hardware or a processor hardware image (logical partition). The processor hardware that can be monitored and controlled by processor operations provide the operations command facility (OCF). This system management interface is called by ISQCCMD to perform hardware commands, for example, ACTIVATE or SYSRESET.

## Sample task: Query / Update LPAR settings (1 of 2)

From HMC: Select CPC, then open “Change LPAR Controls”



© Copyright IBM Corporation 2018

4-31

Sample task: Query / Update LPAR settings (1 of 2)

To change LPAR Controls from the HMC, select CPC, then open “Change LPAR Controls”.

Then, you get a table of all LPARs, their defined capacity, weight, capping, and number of dedicated processors.

## Sample task: Query / Update LPAR settings (2 of 2)

Or using the Processor Operations common command ICNTL

- Syntax: ISQCCMD *target\_object* ICNTL [*parameter ...*]
- Example:

Image control command  
(default is to merely list  
current settings)

```
ISQCCMD KEY6 ICNTL
ISQ901I KEY6      SC  ISQ417I ICNTL STATUS(SUCCESS)
ISQ901I SA02.KEY6 SC  AOFA0007 ICNTL SA02.KEY6 STATUS(ACCEPTED)
CPCSNAME(DEIBMIP1.IP3TKY60) PT(GPP) TSTIME(090116124025)
ISQ901I SA02.KEY6 SC  AOFA0007 CLUSTR(KEY6PLEX)
ISQ901I SA02.KEY6 SC  AOFA0007 DEFCAP(88)
ISQ901I SA02.KEY6 SC  AOFA0007 GRPPRF()
ISQ901I SA02.KEY6 SC  AOFA0007 PWI(100)
ISQ901I SA02.KEY6 SC  AOFA0007 PWIC(NO)
ISQ901I SA02.KEY6 SC  AOFA0007 PWMN(0)
ISQ901I SA02.KEY6 SC  AOFA0007 PWMX(0)
ISQ901I SA02.KEY6 SC  AOFA0007 PWC(100)
ISQ901I SA02.KEY6 SC  AOFA0007 PWCC(NO)
ISQ901I SA02.KEY6 SC  AOFA0007 WLME(NO)
ISQ901I SA02.KEY6 SC  AOFA0007 ICNTL REPORT COMPLETE
ISQ419I ISQCCMD ICNTL processing on KEY6 is complete.
```

Defined Capacity

Initial Capping = NO

Weights (current, initial,  
minimum, and  
maximum)

Machine not capped,  
WLM CPU mgmt = NO

© Copyright IBM Corporation 2018

4-32

### Sample task: Query / Update LPAR settings (2 of 2)

Common commands are preceded by the ISQCCMD command. Use common commands in APIs whenever possible because they provide a single product image across various hardware and software implementations. Regardless of the processor type or the operating system running at your target system, the common command is the same. This can potentially minimize the need for future modifications to your automation routines should you modify or upgrade your processor hardware or operating system type.

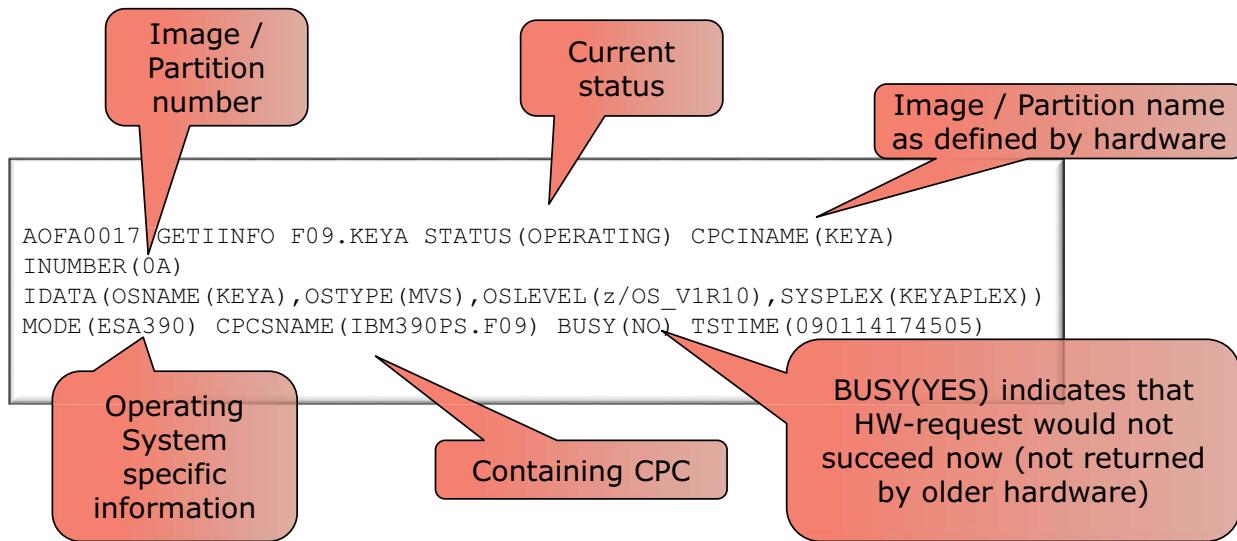
To change LPAR Controls using processor operations, you use common command ICNTL.

The ICNTL common command allows you to query and change LPAR-specific values and settings. The PR/SM hardware component or WLM uses this information to manage partition performance and to distribute shared processor resources among the images of a CPC. Changing the image control values and settings with ICNTL allows you to influence LPAR performance and LPAR resource capacities at run time.

## Sample task: Query target image information

Command syntax for programming / automation:

- Syntax: ISQCCMD target\_system GETIINFO
- Result:



© Copyright IBM Corporation 2018

4-33

*Sample task: Query target image information*

The GETIINFO command returns a subset of the CPCDATA command response. It allows you to retrieve the status and mode of the specified target system LPAR together with the defined LPAR number. If available, OS-related information such as operating system name or version data is returned.

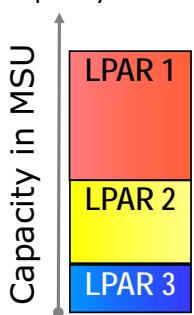
Use the GETIINFO command if you know the LPAR name or system name and you are interested only in HW information for this LPAR or system. For performance reasons, it is not recommended to use CPCDATA in such cases.

## Sample task: Activation profile management (1 of 2)

Image (LPAR) characteristics



Group capacity allocation



Hardware (CPC) characteristics  
Operating system IPL-address

Common command PROFILE can be used to

- List all activation profiles known on a CPC
- Browse the contents of a particular activation profile
- Read or update a single activation profile attribute

Usage

- Centralized activation profile management
- Automatic activation profile compliance checking
- Avoidance of cumbersome manual work at the HMC

## Sample task: Activation profile management (2 of 2)

Access all activation profiles on a given CPC

- Example: ISQCCMD SA02 PROFILE CMD (OPEN)

Get a list of activation profiles for a given type

- Example: ISQCCMD SA02 PROFILE CMD (LIST) TYPE (LOAD)

```
ISQ901I SA02.KEY6 SC AOFA0020 APROF SA02 STATUS(ACCEPTED) CPCNAME...
ISQ901I SA02.KEY6 SC AOFA0020 LOAD(DEFAULTLOAD)
...
ISQ901I SA02.KEY6 SC AOFA0020 LOAD(ISQ0IPUFM)
ISQ901I SA02.KEY6 SC AOFA0020 APROF REPORT COMPLETE
```

Update single activation profile attribute

- Example: ISQCCMD SA02 PROFILE CMD (UPDATE) TYPE (LOAD)  
NAME (DEFAULTLOAD) VAR (IPLADR) VAL (881D)

Finish work with activation profiles

- Example: ISQCCMD SA02 PROFILE CMD (CLOSE)

### Sample task: Activation profile management (2 of 2)

The PROFILE common command allows you to access the activation profiles of a CPC. Activation profiles contain configuration information about the CPC itself and its images (LPARs), and load information for the operating systems to be initialized on the CPC or its LPARs.

With PROFILE, the names of the activation profiles can be listed and the content of a profile can be queried. The profile content can be changed.

The activation profiles are stored in the Support Element of a CPC. They are used for CPC or image activation and for processor load operations that are executed using load profile information.

## ProcOps Hardware Management Console tasks

Daily	Recovery	Configuration	Timer	Console
Hardware Messages	Start All	Change LPAR Controls	System (Sysplex) Time	System Information
Operating System Messages	Stop All	Customize Activation Profiles		View Security Logs
Activate	Reset Normal	Change LPAR Group Controls		
Reset Normal	PSW Restart			Shutdown or Restart
Deactivate	Reset Clear	Special Control	Perform Model Conversion	
Grouping	Load	Customize Console Services	Set Power Saving	
	Test		Energy Control	
	Interrupt			

Same console tasks supported as on the Hardware Management Console

### ProcOps Hardware Management Console tasks

Processor operations supports the same console tasks as the Hardware Management Console including hardware messages, actions against groups, hardware commands, LPAR control, capacity value, activation profiles, exclusive control, Server Time Protocol, temporary capacity upgrade actions, query, and set the CPC power mode, system information, viewing security logs, restart or shutdown a SE or HMC.

Here are some examples, what you can do:

- SYSRESET and ACTIVATE an LPAR
- LOAD image (might also be initiated by the Activation Profile)
- Manage ACTIVATION PROFILES
- Manage temporary capacity using either CBU (Capacity Backup) or OOCoD (On/Off Capacity on Demand) – a corresponding contract needs to be in place

## ProcOps: Hardware messages task

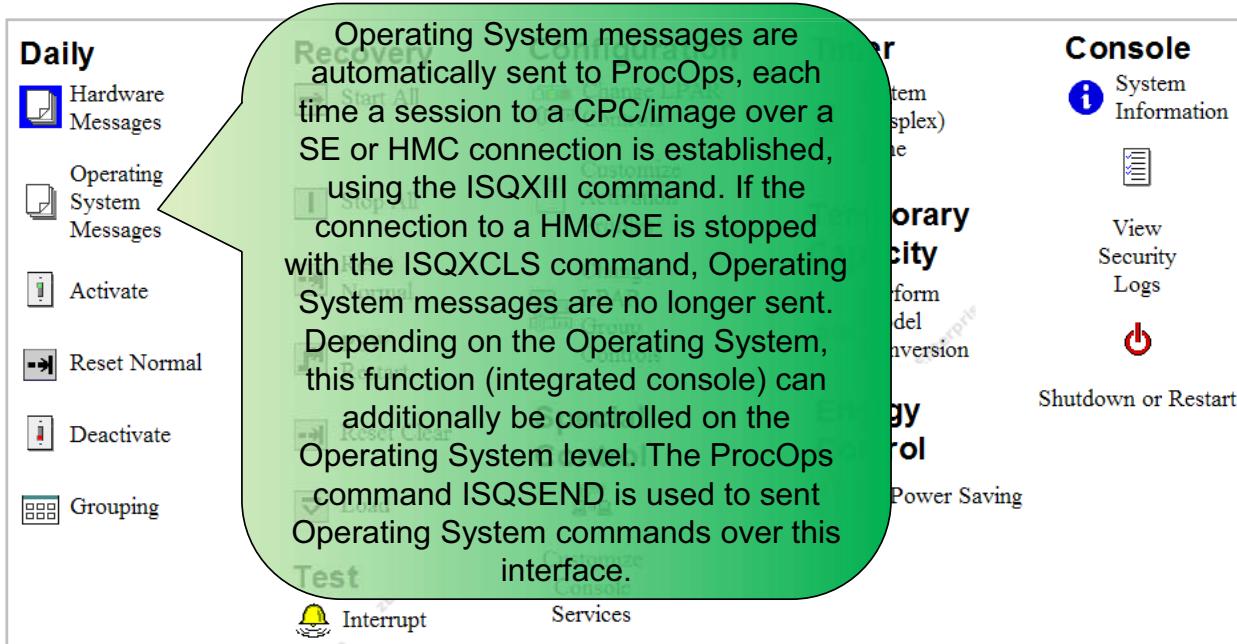


Same console tasks supported as on the Hardware Management Console

### ProcOps: Hardware messages task

Hardware messages originating from a CPC, LPAR, or the SE, HMC itself are automatically sent to ProcOps after a session to a CPC has been established, using the ISQXIII command. If the connection to a HMC/SE is stopped with the ISQXCLS command, hardware messages are no longer sent. Housekeeping - Single hardware messages can be directly deleted from the SE/HMC Hardware message window using the ISQCCMD CLRHWMSG common command. (OA47967)

## ProcOps: Operating system messages task



Same console tasks supported as on the Hardware Management Console

© Copyright IBM Corporation 2018

4-38

### ProcOps: Operating system messages task

Operating System messages are automatically sent to ProcOps, each time a session to a CPC/image over a SE or HMC connection is established, using the ISQXIII command. If the connection to a HMC/SE is stopped with the ISQXCLS command, Operating System messages are no longer sent.

Depending on the Operating System, this function (integrated console) can additionally be controlled on the Operating System level. The ProcOps command ISQSEND is used to send Operating System commands over this interface.

## ProcOps: Hardware commands tasks



Same console tasks supported as on the Hardware Management Console

© Copyright IBM Corporation 2018

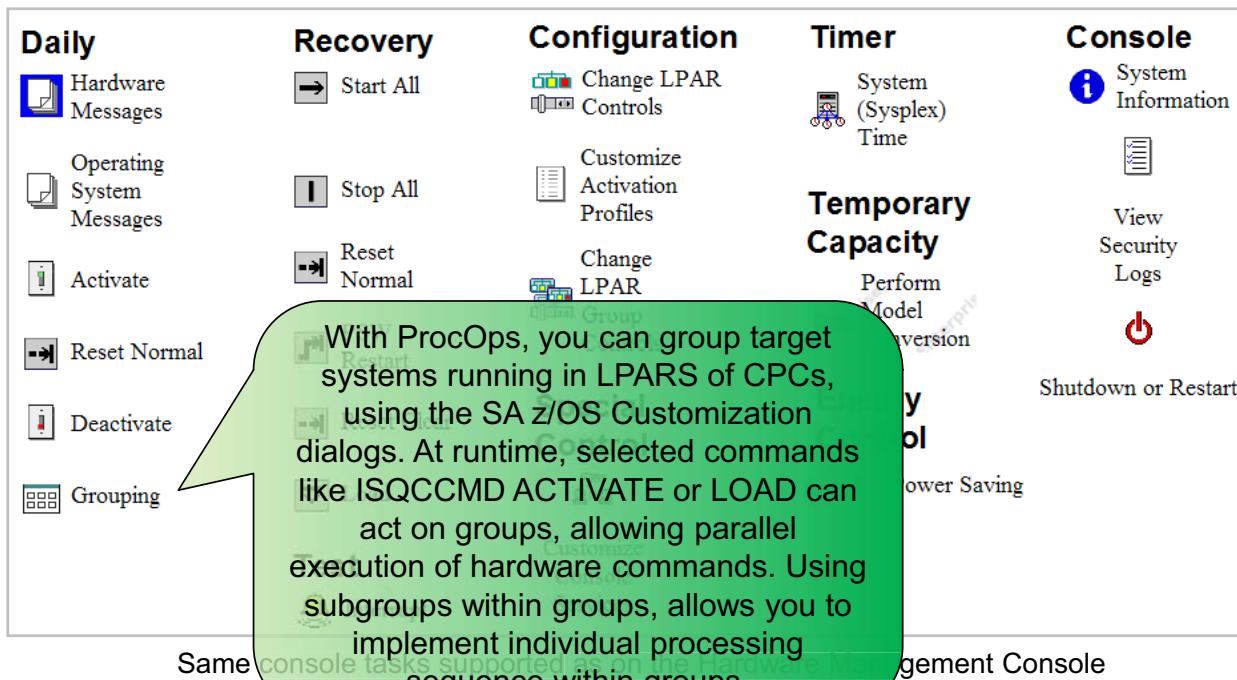
4-39

### ProcOps: Hardware commands tasks

Daily task hardware commands like ACTIVATE, DEACTIVATE, or RESET are executed in ProcOps, using its common hardware command interface ISQCCMD.

ACTIVATE and DEACTIVATE are also the common commands to Power-On and Power-Off a Processor Complex.

## ProcOps: Actions on groups



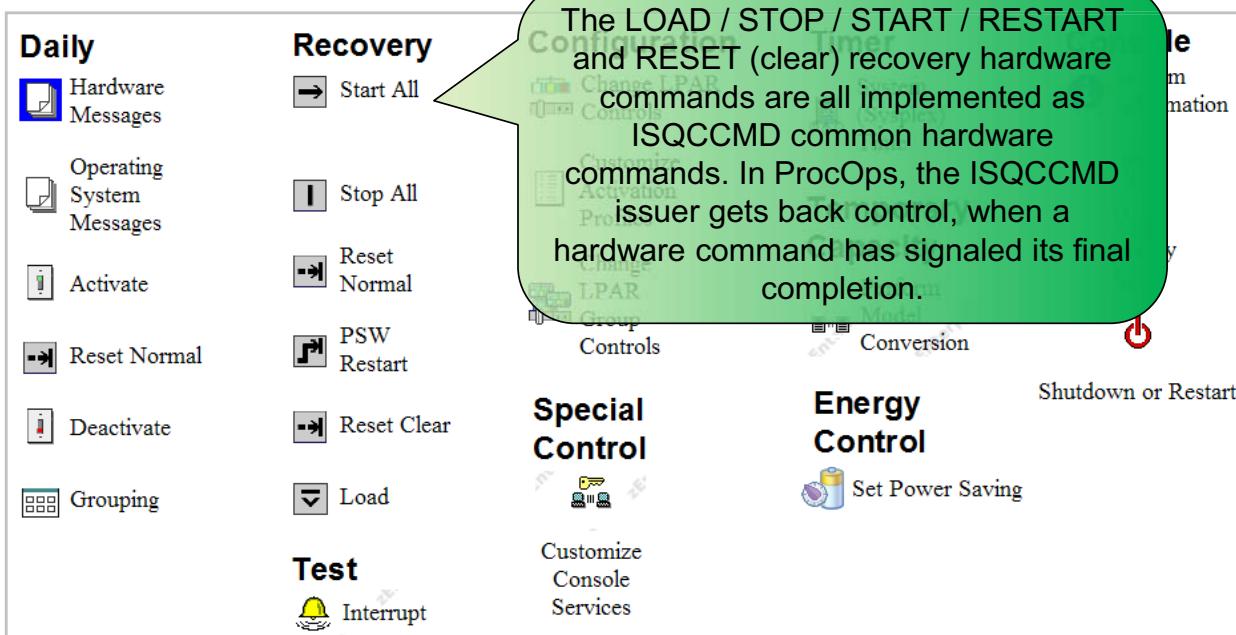
© Copyright IBM Corporation 2018

4-40

### ProcOps: Actions on groups

With ProcOps, you can group target systems running in LPARS of CPCs, using the SA z/OS Customization dialogs. At runtime, selected commands like ISQCCMD ACTIVATE or LOAD can act on groups, allowing parallel execution of hardware commands. Using subgroups within groups, allows you to implement individual processing sequence within groups.

## ProcOps: Hardware recovery tasks



Same console tasks supported as on the Hardware Management Console

© Copyright IBM Corporation 2018

4-41

### ProcOps: Hardware recovery tasks

The LOAD / STOP / START / RESTART and RESET (clear) recovery hardware commands are all implemented as ISQCCMD common hardware commands. In ProcOps, the ISQCCMD issuer gets back control, when a hardware command has signaled its final completion.

## ProcOps: Test task for z/OS

Daily	Recovery	Configuration	Timer	Console
Hardware Messages	Start All	Change LPAR Controls	System (Sysplex) Time	System Information
Operating System Messages	Stop All	Customize Activation Profiles		View Security Logs
Activate	Reset Normal	Change LPAR Group Controls		
Reset Normal	PSW Restart	Perform Model Conversion		
Deactivate	Reset Clear	Customize Console Services		
Grouping	Load	Special Control		
	Test	Interrupt		

Same console tasks supported as on the Hardware Management Console

When targeting a z/OS LPAR, the ISQCCMD EXTINT common command can be used to test if the ProcOps and SA z/OS hardware automation environment is working, without disrupting anything. Since z/OS no longer supports EXTINT, WTO message IEA035I is issued with this information.

© Copyright IBM Corporation 2018

4-42

### ProcOps: Test task for z/OS

When targeting a z/OS LPAR, the ISQCCMD EXTINT common command can be used to test if the ProcOps and SA z/OS hardware automation environment is working, without disrupting anything. Since z/OS no longer supports EXTINT, WTO message IEA035I is issued with this information.



**Important:** When targeting other Operating Systems, first make sure that EXTINT is not disruptive.

## ProcOps: Hardware LPAR control tasks

Daily	Configuration	Timer	Console
<p>The ISQCCMD ICNTL (image control) command allows to query and set control values related to the targeted LPAR. When changed, some of the values go into effect immediately for the addressed LPAR. Controls that can be changed are, for example, the LPAR capacity MSU value, used for software prizing, or the LPAR weight.</p> <p>Absolute Capping values can be set with ICNTL (OA47967).</p>	<p> Change LPAR Controls</p> <p> Customize Activation Profiles</p> <p> Change LPAR Group Controls</p> <p> Special Control</p> <p> Customize Console Services</p>	<p> System (Sysplex) Time</p> <p> Temporary Capacity</p> <p> Energy Control</p>	<p> System Information</p> <p> View Security Logs</p> <p> Shutdown or Restart</p>
Same console tasks supported as on the Hardware Management Console			

© Copyright IBM Corporation 2018

4-43

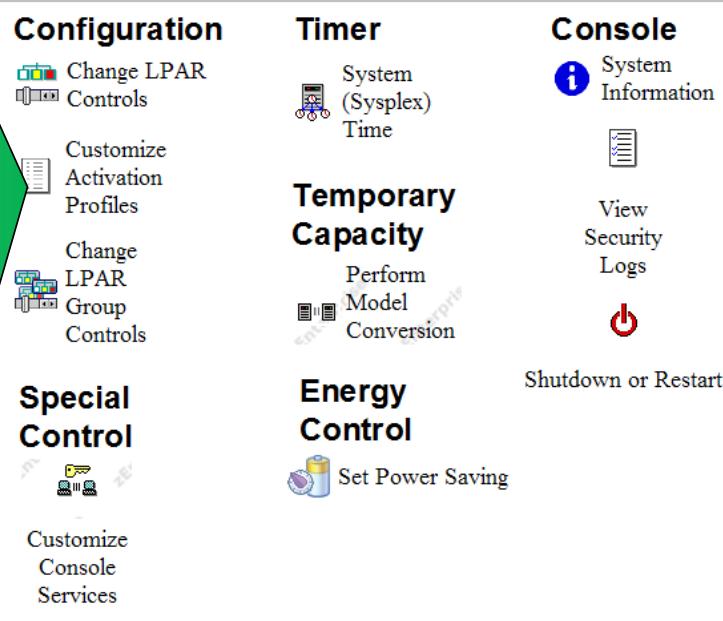
### ProcOps: Hardware LPAR control tasks

The ISQCCMD ICNTL (image control) command allows to query and set control values related to the targeted LPAR. When changed, some of the values go into effect immediately for the addressed LPAR. Controls that can be changed are, for example, the LPAR capacity MSU value, used for software prizing, or the LPAR weight.

Absolute Capping values can be set with ICNTL (OA47967).

## ProcOps: Activation profiles tasks

Activation profiles are used during CPC and LPAR ACTIVATION. They hold information about the configured resources for the CPC, LPAR object. In addition, MSU capacity groups and LOAD profiles, holding base CP IPL information can be queried and updated using the ISQCCMD PROFILE command. System Programmers can use the ISQCCMD PROFILE command to perform scheduled profile checks or mass updates as an alternative to the manual HMC operation.



Reported as on the Hardware Management Console

© Copyright IBM Corporation 2018

4-44

### ProcOps: Activation profiles tasks

Activation profiles are used during CPC and LPAR ACTIVATION. They hold information about the configured resources for the CPC, LPAR object. In addition, MSU capacity groups and LOAD profiles, holding base CP IPL information can be queried and updated using the ISQCCMD PROFILE command. System Programmers can use the ISQCCMD PROFILE command to perform scheduled profile checks or mass updates as an alternative to the manual HMC operation.

## ProcOps: Hardware group capacity tasks

Daily	Recovery	Configuration	Timer	Console
Hardware Messages	Start All	Change LPAR Controls	System (Sysplex) Time	System Information
Operating System Messages	Stop All	Customize Activation Profiles		View Security Logs
Activate	Reset Normal	Change LPAR Group Controls		
Reset Normal	PSW Restart	Special Control		
Deactivate	Reset Clear	Customize Console Services		
Grouping	Load	Test	Perform Model Conversion	
				Temporary Capacity

The capacity value (MSU) of the group, where the addressed LPAR is a member of, can be queried and changed with the IQCCMD ICNTL GRPCAP command. Valid changes go into effect immediately.

Same console tasks supported as on the Hardware Management Console

### ProcOps: Hardware group capacity tasks

The capacity value (MSU) of the group, where the addressed LPAR is a member of, can be queried and changed with the IQCCMD ICNTL GRPCAP command. Valid changes go into effect immediately.

## ProcOps: Hardware exclusive control

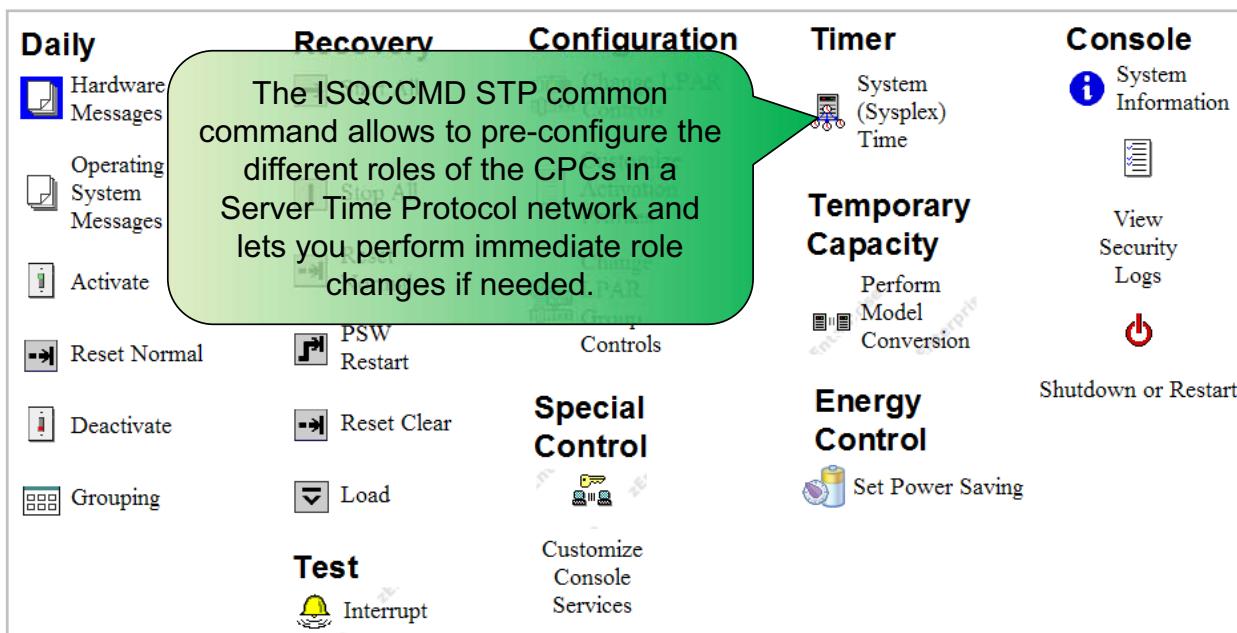
Daily	Recovery	Configuration	
Hardware Messages	Start All	Change LPAR Controls	The ISQCCMD RESERVE command lets you gain exclusive control over a Support Element.
Operating System Messages	Stop All	Customize Activation Profiles	In this mode, no automation program, including the ProcOps that issued the reserve can issue hardware commands. Queries are still possible. hardware commands from an HMC directed to the reserved SE are also blocked. The issuer of the reserve is responsible to release it.
Activate	Reset Normal	Change LPAR Group Controls	Rebooting a reserved SE does not release it.
Reset Normal	PSW Restart		
Deactivate	Reset Clear		
Grouping	Load	Customize Console Services	
Test		Special Control	
	Interrupt		

Same console tasks supported as on the Hardware Management Console

### ProcOps: Hardware exclusive control

The ISQCCMD RESERVE command lets you gain exclusive control over a Support Element. In this mode, no automation program, including the ProcOps that issued the reserve can issue hardware commands. Queries are still possible. hardware commands from an HMC directed to the reserved SE are also blocked. The issuer of the reserve is responsible to release it. Rebooting a reserved SE does not release it.

## ProcOps: Hardware time-related tasks



Same console tasks supported as on the Hardware Management Console

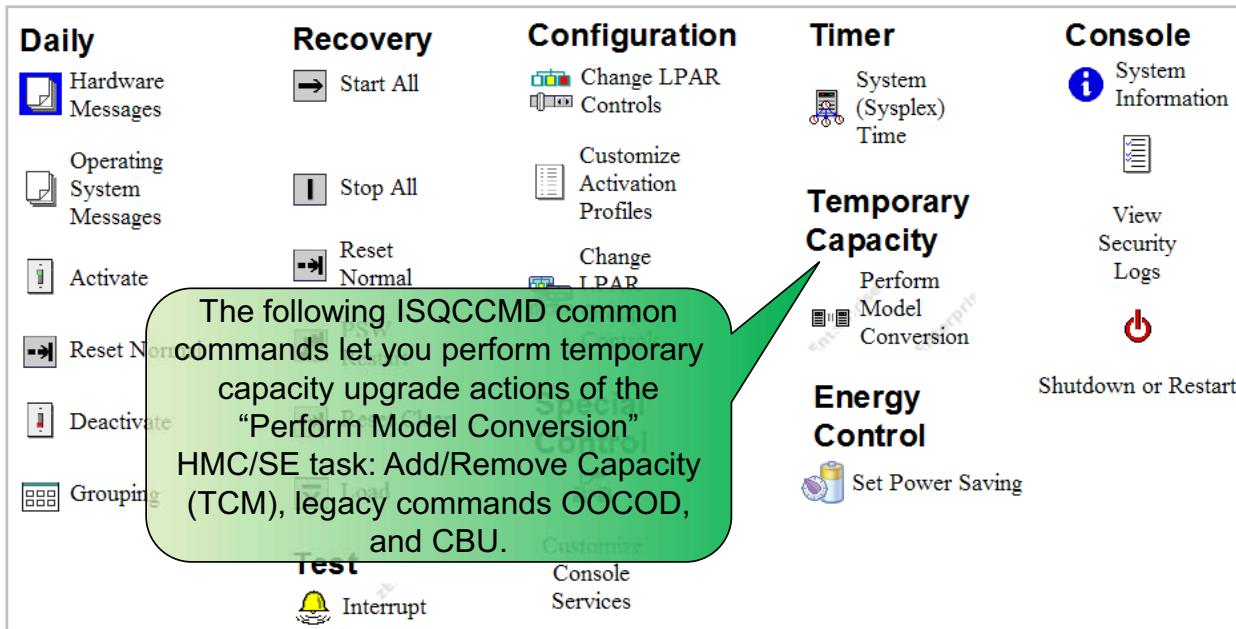
© Copyright IBM Corporation 2018

4-47

### ProcOps: Hardware time-related tasks

The ISQCCMD STP common command allows to pre-configure the different roles of the CPCs in a Server Time Protocol network and lets you perform immediate role changes if needed.

# ProcOps: Hardware temporary capacity upgrade tasks



Same console tasks supported as on the Hardware Management Console

© Copyright IBM Corporation 2018

4-48

## ProcOps: Hardware temporary capacity upgrade tasks

The following ISQCCCMD common commands let you perform temporary capacity upgrade actions of the “Perform Model Conversion” HMC/SE task: Add/Remove Capacity (TCM), legacy commands OOCOD, and CBU.

## ProcOps: Hardware energy control tasks

Daily	Recovery	Configuration	Timer	Console
Hardware Messages	Start All	Change LPAR Controls	System (Sysplex) Time	System Information
Operating System Messages	Stop All	Customize Activation Profiles		View Security Logs
Activate	Reset Normal	Change LPAR Group Controls	Temporary Capacity	
Reset Normal	PSW Restart		Perform Model Conversion	Shutdown or Restart
Deactivate			Energy Control	
Grouping			Set Power Saving	

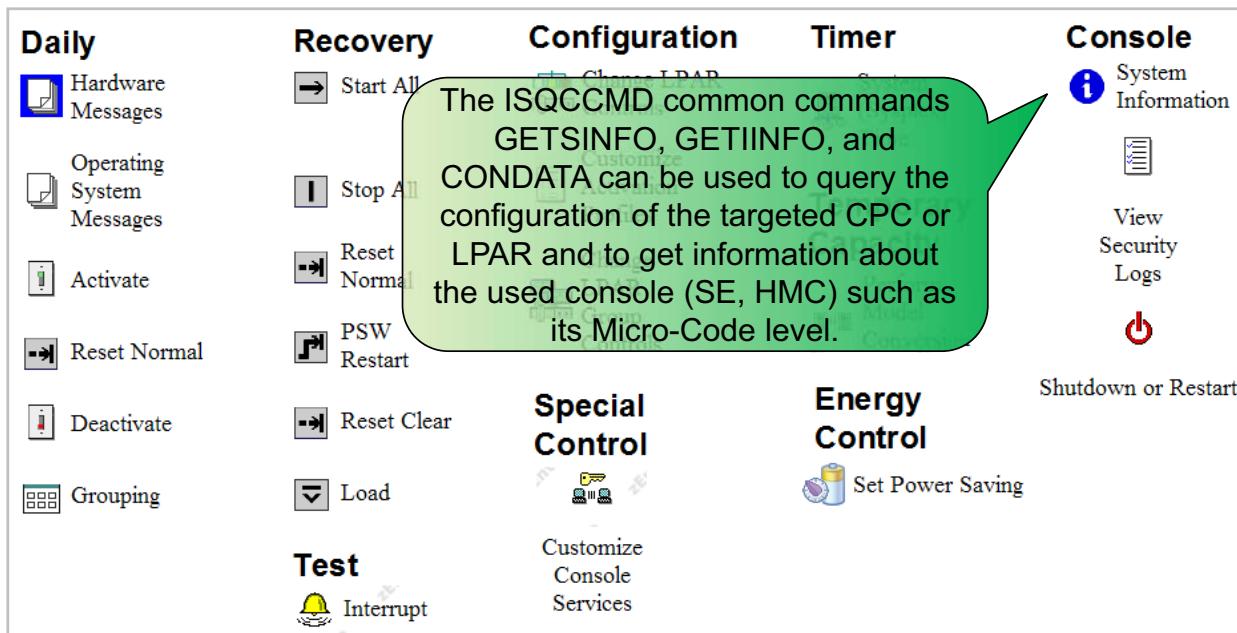
The ISQCCMD POWERMOD common command allows to query the currently set CPC power mode or if a mode change is supported. Finally, the command can be used to change from normal to power save mode and back.

Same console tasks supported as on the Hardware Management Console

### ProcOps: Hardware energy control tasks

The ISQCCMD POWERMOD common command allows to query the currently set CPC power mode or if a mode change is supported. Finally, the command can be used to change from normal to power save mode and back.

## ProcOps: Hardware query tasks



Same console tasks supported as on the Hardware Management Console

### ProcOps: Hardware query tasks

The ISQCCMD common commands GETSINFO, GETINFO, and CONDATA can be used to query the configuration of the targeted CPC or LPAR and to get information about the used console (SE, HMC) such as its Micro-Code level.

## ProcOps: Hardware security log monitoring

Daily	Recovery	Configuration	Timer	Console
Hardware Messages	Start All	Change LPAR Controls	System (Sysplex) Time	System Information
Operating System Messages	Stop All	Customize Activation Profiles		View Security Logs
Activate	Reset Normal	Change LPAR Group Controls		
Reset Normal	PSW Restart	Special Control		
Deactivate	Reset Clear	Customize Console Services		
Grouping	Load			
Test				
	Interrupt			

After the ISQCCMD SECLOG is turned ON, updates of the console security log are forwarded to ProcOps for monitoring. User access, hardware command execution, console mirroring, CPC configuration changes can be tracked.

Same console tasks supported as on the Hardware Management Console

### ProcOps: Hardware security log monitoring

After the ISQCCMD SECLOG is turned ON, updates of the console security log are forwarded to ProcOps for monitoring. User access, hardware command execution, console mirroring, CPC configuration changes can be tracked.

## ProcOps: Shutdown or restart HMC task

Daily	Recovery	Configuration	Timer	Console
Hardware Messages	Start All	Change LPAR Controls	System (Sysplex) Time	System Information
Operating System Messages	Stop All	Customize Activation Profiles		View Security Logs
Activate	Reset Normal	Change LPAR Group Controls		
Reset Normal	PSW Restart			Shutdown or Restart
Deactivate	Reset Clear			
Grouping	Load	Special Control		
	Test	Customize Console Services		
	Interrupt			
			<b>Temporary Capacity</b>	
			Perform Model Conversion	
			<b>Energy Control</b>	

Same console tasks supported as on the Hardware Management Console

© Copyright IBM Corporation 2018

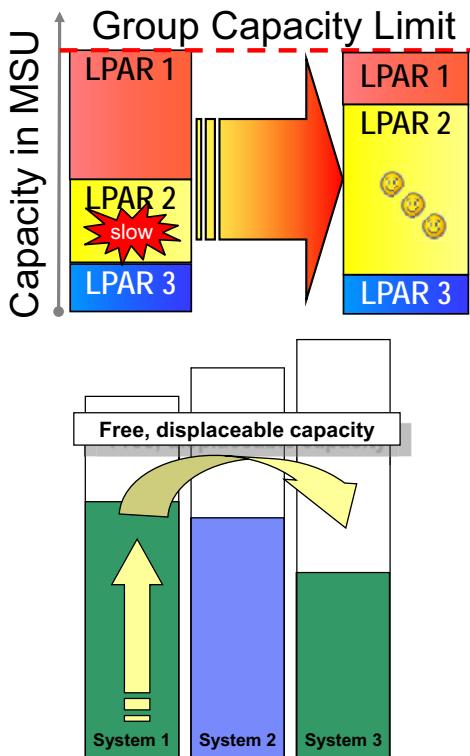
4-52

ProcOps: Shutdown or restart HMC task

CTRLCONS is the ISQCCMD common command to Restart or Shutdown a SE or HMC.

CTRLCONS is the ISQCCMD common command to Restart or Shutdown a SE or HMC.

## Do your own smart capping and load distribution



### Software cost challenges:

- Charges based on peak rolling 4-hour average or the LPAR Defined Capacity whichever is lower
- Capping might cap priority workload

### Solutions using SA z/OS:

- Move workload to under-utilized system using server/move groups based on
  - Predicted free capacity and WLM data
  - Using OMEGAMON metrics
- Use ProcOps API to adjust capacity across LPARs and WLM capacity groups automatically
  - Absolute capping supported
- Policy-based looping jobs resolution

© Copyright IBM Corporation 2018

4-53

#### *Do your own smart capping and load distribution*

The killer function is that processor operations allows you to change LPAR weights based on schedule, workload, and application. This can be used to lower the defined capacity to lower your software costs.

Background:

A System z LPAR can be assigned resources like CPU, memory, or weight for WLM LPAR CPU management.

The problem is that often applications are licensed by capacity, sometimes calculated as average.

Clients often need temporary higher capacity, for example for quarter end processing or for a marketing campaign. If they exceed their allowed capacity, they must either pay extra or reduce to a lower capacity to meet the average capacity they are licensed for.

Automation software can measure the four-hour rolling average, know when it's approaching the capping level, recognize peaks before they become peaks, and take appropriate action before the usage becomes a problem. These actions can smooth out usage spikes and, in fact, allow you to run at a lower capping level than otherwise practical, without sacrificing your service levels. Therefore, you get the full benefits of subcapacity pricing.

## Solutions using SA z/OS

- Moving workload to under-utilized system based on the predicted free capacity and WLM data or based on OMEGAMON metrics.
- Adjusting LPAR Weights of systems that are in the same capacity group.
- Move LPARs into another capacity group to leave more capacity for more important workload.
- Adjusting capacity across LPARs and WLM capacity groups automatically.
- Automatically moving low priority workloads to systems with free capacity.
- Detect and resolve job loops and transaction abends that lead to countless recoveries.

## z/OS Workload Management (WLM)

With z/OS Workload Management, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as CPU or storage, should be given to it to meet the goal. Workload Management constantly monitors the system and adapt processing to meet the goals. The 4-hour rolling average is tracked by the z/OS Workload Manager approximately every 10 seconds, and if that average exceeds the defined capacity, the LPAR is soft-capped by PR/SM to limit the LPAR capacity to the defined capacity value.

A well defined and working WLM policy is important to ensure that workloads meet their objectives.

### 4-hour rolling average (4HRA)

The 4-hour rolling average is calculated by WLM. The 4-hour rolling average (SMF70LAC) is the average of the 48 last measures of instantaneous MSU (IMSU or ACTMSU, the real consumption).

### Defined Capacity

Defined Capacity is the size of an LPAR in MSUs that you set indicating the capacity of an LPAR. This is also referred to as a soft cap. The defined capacity specifies a maximum rolling 4-hour average consumption for an LPAR. This means that it is possible for the actual capacity being used by the LPAR at a given time to be higher than the defined capacity.

### Soft capping

'Soft capping' is a technique that is primarily intended to help you control your software costs. The process of limiting the amount of capacity that can be consumed by an LPAR (and as a result, controlling your software bills) is often called 'soft capping' or 'defined capacity' or Capacity Group Limit.

Capping starts when the rolling 4-hour average reaches the Defined Capacity. At that point, the LPAR is capped to the Defined Capacity. The WLM Capping% (SMF70NSW) represents the percentage of time during which the LPAR was soft-capped during the interval.

## Capacity groups

Capacity groups are an extension of the concept of defined capacities. However, a defined capacity only applies to a single LPAR, a group capacity limit (as its name implies) specifies a limit on the rolling 4-hour average utilization for a group of LPARs. The share of the group capacity limit that each member of the group is entitled to is determined by that LPAR's relative weight compared to the other members of the capacity group, combined with a share of any capacity that other group members might not be using, also called Unused Capacity (UC). Capacity groups and defined capacities can be combined.

Most clients use capacity groups as they allow a member to consume Unused Capacity of other members without more charges. An LPAR can only be a member of one capacity group at a time, but it can be moved to another capacity group. All the members of the capacity group must be on the same CPC, but they do not need to be in the same sysplex.

## Automatic CPC and LPAR capacity changes

- ProcOps Management API to:
  - Query current CPC configuration details
  - Query and set LPAR-specific controls
  - Query and set the default CPC RESET activation profile
  - Manage RESET, IMAGE, and LOAD activation profiles
  - Automate IPLs from SCSI devices (Linux, z/VM)
- Allows you to automate based on schedule, workload, application
  - **LPAR weight (defined, initial, minimal, maximal, current (R/O))**
  - **Defined capacity to lower your software costs**
  - Provisioning of new CPs to adapt to increased workload
  - Switch between WLM and PR/SM management
  - IPL profiles for easier operations
- ISQCCMD target\_system\_ProcOps\_name ICNTL  
target.hardware\_name.lparname CMD(UPDATE) VAR(vnm) VAL(vvl)

### Automatic CPC and LPAR capacity changes

ProcOps Management API to:

- Query current CPC configuration details
- Query and set LPAR-specific controls
- Query and set the default CPC RESET activation profile
- Manage RESET, IMAGE, and LOAD activation profiles
- Automate IPLs from SCSI devices (Linux, z/VM)

Allows you to automate based on schedule, workload, application

- LPAR weight (defined, initial, minimal, maximal, current (R/O))
- Defined capacity to lower your software costs
- Provisioning of new CPs to adapt to increased workload
- Switch between WLM and PR/SM management
- IPL profiles for easier operations

ISQCCMD target\_system\_ProcOps\_name ICNTL target\_hardware\_name.lparname  
CMD(UPDATE) VAR(vnm) VAL(vvl)

# Programming considerations: Processor Operations

- Depending on the CPC type and model of the HMC Console Workplace version, not all of the shown HMC tasks might be available. For more information about restriction and limitation information, see the ISQCCMD Common Commands documentation and online help.
- Internally ProcOps exploits the System z SNMP APIs for the ISQCCMD command. Not all task subfunctions available with the SE/HMC GUIs, might be implemented in the APIs. For more information, see “*z Systems Application Programming Interfaces*”, available for viewing or download on IBM Resource Link.
- If you use ISQCCMD directly in a NetView PIPE, make sure the EXPOSE TOTRAP stage is specified.
- Instead of waiting for **command related response messages** within or outside of the PIPE that issues the ISQCCMD, you can use PIPE KEEP name ISQ.SNMP to inspect the responses, including the final hardware command completion code.
- If an ISQCCMD hardware command triggers **other event messages**, in addition to the hardware command completion message, use 'TRAP AND WAIT' or have an AT entry available to catch them.

## Programming considerations: processor operations

Depending on the CPC type and model of the HMC Console Workplace version, not all of the shown HMC tasks might be available. For more information about restriction and limitation information, see the ISQCCMD Common Commands documentation and online help.

Internally ProcOps exploits the System z SNMP APIs for the ISQCCMD command. Not all task subfunctions available with the SE/HMC GUIs, might be implemented in the APIs. For more information, see “*z Systems Application Programming Interfaces*” SB10-7030-16, available for viewing or download on IBM Resource Link.

If you use ISQCCMD directly in a NetView PIPE, make sure the EXPOSE TOTRAP stage is specified.

Instead of waiting for command related response messages within or outside of the PIPE that issues the ISQCCMD, you can use PIPE KEEP name ISQ.SNMP to inspect the responses, including the final hardware command completion code.

If an ISQCCMD hardware command triggers other event messages, in addition to the hardware command completion message, use 'TRAP AND WAIT' or have an AT entry available to catch them.

## Summary

Now that you completed this unit, you can perform the following tasks related to Processor Operations:

- Describe architecture including implementation options
- Explain usage and operator interface
- Describe automation policy

### *Summary*

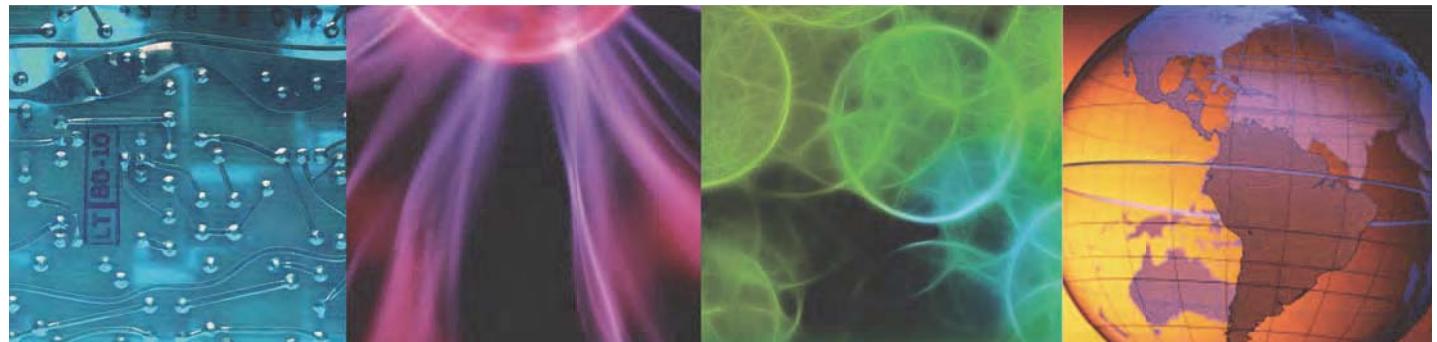
Now that you completed this unit, you can perform the following tasks related to processor operations:

- Describe architecture including implementation options
- Explain usage and operator interface
- Describe automation policy





SM917 1.2



[ibm.com/training](http://ibm.com/training)

Authorized  
**IBM | Training**