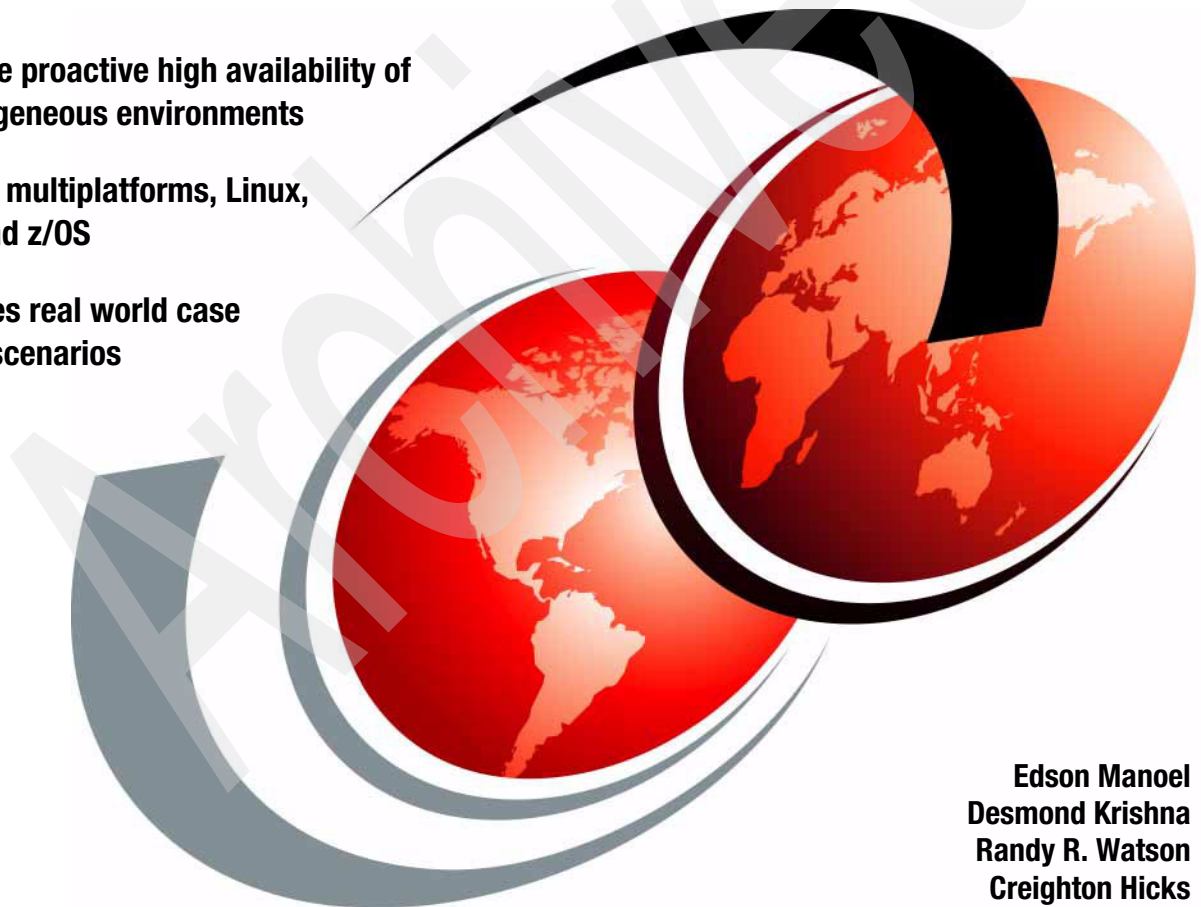


End-to-end Automation with IBM Tivoli System Automation for Multiplatforms

Achieve proactive high availability of
heterogeneous environments

Covers multiplatforms, Linux,
AIX, and z/OS

Includes real world case
study scenarios



Edson Manoel
Desmond Krishna
Randy R. Watson
Creighton Hicks



International Technical Support Organization

**End-to-end Automation with IBM Tivoli System
Automation for Multiplatforms**

November 2005

Archived

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

First Edition (November 2005)

This edition applies to IBM Tivoli System Automation for Multiplatforms V2.1 and IBM Tivoli System Automation for z/OS V3.1.

Note: This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. We recommend that you consult the product documentation or follow-on versions of this redbook for more current information.

© Copyright International Business Machines Corporation 2005. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	ix
Tables	xi
Examples	xiii
Notices	xvii
Trademarks	xviii
Preface	xix
The team that wrote this redbook	xx
Become a published author	xxii
Comments welcome	xxii
Part 1. Fundamentals	1
Chapter 1. IBM Tivoli System Automation for Multiplatforms V2.1	3
1.1 IBM Tivoli System Automation for Multiplatforms V2.1 overview	4
1.1.1 Main features	4
1.2 Base Component overview	6
1.2.1 Reliable Scalable Cluster Technology	8
1.2.2 Resource Managers	8
1.2.3 End-to-end Automation Adapter	9
1.3 End-to-end Automation Management Component overview	9
1.3.1 Automation Engine	12
1.3.2 End-to-end Automation Manager	12
1.3.3 Operations Console	12
1.3.4 Automation database	14
1.3.5 Automation policy	14
1.3.6 End-to-end Automation Adapter	14
1.4 Communication between end-to-end components	15
1.5 Concepts and terminology	24
1.5.1 High Availability and IBM Tivoli System Automation for Multiplatforms	24
1.5.2 Terms used in IBM Tivoli System Automation for Multiplatforms	25
Chapter 2. IBM Tivoli System Automation for z/OS V3.1	33
2.1 IBM Tivoli System Automation for z/OS V3.1 overview	34
2.2 What is new in IBM Tivoli System Automation for z/OS V3.1	37

2.2.1 Enhancements to the Customization Dialog	37
2.2.2 IBM Tivoli OMEGAMON integration	39
2.2.3 GDPS Integration	40
2.2.4 IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter	40
2.3 Overview Planning for installation	44
Part 2. Case study scenario implementation	45
Chapter 3. Case study scenario overview	47
3.1 Scenario overview	48
Chapter 4. Case study scenario:	
HTTP Servers on Linux first-level automation domain	53
4.1 Apache automation domain overview	55
4.1.1 Installation	56
4.1.2 Automation requirements	56
4.2 Automation domain configuration	56
4.2.1 Create the first-level automation domain	57
4.2.2 Define resources in the automation domain	58
4.2.3 Create the automation policy using relationship definitions	67
4.2.4 Change the operational state of the resource group	69
4.2.5 Configuration error and recovery example	71
4.2.6 Exercising the automation policy example	72
4.3 End-to-end Automation Adapter configuration	75
4.3.1 Configure the End-to-end Automation Adapter	77
4.3.2 Replicate configuration files to nodes in the automation domain	80
4.3.3 Define the End-to-end Automation Adapter automation policy	81
4.4 Miscellaneous information	83
Chapter 5. Case study scenario:	
Application Servers on AIX first-level automation domain	89
5.1 Application server automation domain overview	91
5.1.1 Automation requirements	92
5.2 Automation domain configuration	92
5.2.1 Create the first-level automation domain	93
5.2.2 Define automation domain resources	95
5.2.3 Create the automation policy using relationship definitions	109
5.2.4 Change the Operational State of the resource group	112
5.2.5 Verify the operational quorum and tie breaker definition	115
5.3 End-to-end Automation Adapter configuration	118
5.3.1 Generate End-to-end Automation Adapter configuration files	120
5.3.2 Replicate the End-to-end Automation Adapter configuration files	124
5.3.3 Define the End-to-end Automation Adapter automation policy	125

5.4 Maintaining defined policies	127
--	-----

Chapter 6. Case study scenario:

IBM DB2 on z/OS first-level automation domain	129
6.1 IBM DB2 on z/OS automation domain overview	131
6.2 IBM DB2 on z/OS automation domain configuration	131
6.2.1 Configure NetView for IBM Tivoli System Automation for z/OS ...	132
6.2.2 Automate NetView startup procedure	134
6.2.3 Allocate System-Unique data sets	134
6.2.4 Configure the Automation Manager	135
6.2.5 Allocate data sets for the ISPF customization dialog	136
6.2.6 Update PARMLIB data sets	136
6.2.7 Update PROCLIB data sets	137
6.2.8 Define the base automation policy	144
6.3 Configuring automation policies for IBM DB2	145
6.3.1 Identify required IBM DB2 messages	146
6.3.2 Create scenario automation policy database	147
6.3.3 Populate the scenario policy database	149
6.3.4 Define policies for monitoring IBM DB2 application tasks	152
6.3.5 Import customized scenario policy database into production	165
6.3.6 Create application group and define group membership	170
6.3.7 Verify Relationships in the automation policy	176
6.4 End-to-end Automation Adapter configuration	181
6.4.1 Check prerequisites and dependencies	184
6.4.2 Configure NetView and IBM Tivoli System Automation for z/OS ..	184
6.4.3 Enabling the Event Automation Service	186
6.4.4 Configure the Global Initialization File	186
6.4.5 Configure the NetView Message Adapter Service	187
6.4.6 Customize the End-to-end Automation Adapter	190
6.4.7 Perform configuration for security	196
6.4.8 Verify startup of the Automation Adapter	197
6.4.9 Solve timeout problems	198

Chapter 7. Case study scenario:

End-to-end automation domain	201
7.1 End-to-end Automation Management Component installation	203
7.2 Installation verification tasks	211
7.2.1 EAUTODB and OPCONDB databases	211
7.2.2 End-to-end Automation Management Component automation engine startup	213
7.2.3 End-to-end Automation Management Component applications status. 214	
7.2.4 JDBC providers connection	215

7.2.5	ISC portal application startup	216
7.2.6	System Automation operations console	217
7.3	Users and group management	218
7.3.1	Creating users.	219
7.3.2	Creating user groups.	219
7.3.3	Assigning access permissions to user groups	221
7.3.4	Assigning users to user groups.	224
7.3.5	Assigning access roles to user groups	226
7.4	End-to-end Automation Management Component configuration	228
7.5	Defining the end-to-end automation policy	229
7.5.1	Automation requirements and policy overview	230
7.5.2	Creating the end-to-end automation policy file	231
7.5.3	Verifying the end-to-end automation policy file	240
7.5.4	Activating the end-to-end automation policy file	240
Part 3.	Appendixes	245
Appendix A.	Troubleshooting overview	247
	Communication between end-to-end components.	248
	Location of the root directories	248
	Tivoli common directory	249
	Log and trace files	250
	End-to-end Automation Management Component automation engine	250
	End-to-end Automation Management Component automation manager	251
	End-to-end Automation Adapter	251
	IBM Tivoli System Automation for Multiplatforms Operations Console	252
	The log viewer tool	252
	The TraceWizard utility.	252
Appendix B.	Additional material	255
	Locating the Web material	255
	Using the Web material	255
	System requirements for downloading the Web material	256
	How to use the Web material	256
Abbreviations and acronyms		257
Related publications		259
	IBM Redbooks	259
	Other publications	259
	Online resources	260
	How to get IBM Redbooks	260
	Help from IBM	260

Index 261

Archived

Figures

1-1	IBM Tivoli System Automation for Multiplatforms V2.1 components	6
1-2	Sample scenario: Base Component	7
1-3	Sample Scenario: End-to-end Automation Management Component . .	10
1-4	Basic role of the End-to-end Automation Manager	11
1-5	IBM Tivoli System Automation for Multiplatforms Operations Console .	13
1-6	Basic role of the End-to-end Automation Adapter	15
1-7	Communication overview	16
1-8	End-to-end automation management environment startup workflow. .	17
1-9	First-level automation adapter startup	19
1-10	Resource monitoring workflow	20
1-11	Request against a resource reference workflow	22
1-12	Event from a referenced resource workflow	23
1-13	Subclusters from a cluster	26
1-14	Tie breaker.	28
2-1	Monitor, control, and automation functions	35
2-2	E2E Automation Adapter communication	41
3-1	Application environment.	48
3-2	End-to-end automation case study scenario	49
3-3	IBM Tivoli System Automation for Multiplatforms scenario components	50
4-1	Apache first-level automation domain	54
4-2	Web application tier configuration	55
4-3	Overview of the Apache automation domain	68
4-4	End-to-end domain and apache_SA_Domain interaction	76
4-5	Configure the automation adapter for the Apache automation domain .	76
4-6	Adapter Tab data fields	77
4-7	Host using adapter tab.	78
4-8	Automation Tab	79
4-9	Modified End-to-end Automation Adapter configuration files.	80
4-10	Replicate configuration files to other nodes in automation domain . . .	80
4-11	Adapter configuration replication completion dialog box	81
4-12	Defining automation policy for End-to-end Automation Adapter	82
5-1	Application server first-level automation domain	90
5-2	Web application tier configuration	91
5-3	Defined relationships for the scenario	112
5-4	Network tie breaker	116
5-5	End-to-end domain and was_SA_Domain interaction	119
5-6	End-to-end Automation Adapter configuration tool	120
5-7	Configuration tool: Adapter tab	121

5-8	Configuration tool: Host using adapter tab.	122
5-9	End-to-end Automation Adapter configuration tool: Automation tab . .	123
5-10	End-to-end Automation Adapter configuration files	124
5-11	End-to-end Automation Adapter configuration tool: Replication	124
5-12	End-to-end Automation Adapter configuration tool: Replication results	125
5-13	End-to-end Automation Adapter configuration tool: Defining policies .	125
6-1	IBM DB2 on z/OS first-level automation domain	130
6-2	End-to-end automation domain and SC64N interaction	181
6-3	End-to-end Automation Adapter communication	183
6-4	com.ibm.eez.aab.invocation-timeout-seconds variable definition . . .	200
7-1	Our scenario's end-to-end automation domain	202
7-2	End-to-end Automation Management Component scenario server. . .	205
7-3	LTPA Properties: SSO domain name	206
7-4	Operations Console Database.	207
7-5	Security using IBM DB2 database user registry option	207
7-6	ISC settings	208
7-7	ISC port number settings	209
7-8	Console Help Server port number	209
7-9	ISC application server names	210
7-10	End-to-end automation domain name	210
7-11	Installation successful message	211
7-12	End-to-end Automation Management Component applications	215
7-13	JDBC Providers connection.	216
7-14	End-to-end Automation Management Component operations console	218
7-15	Required groups definition.	221
7-16	Granting access to the Integrated Solution Console pages.	223
7-17	Granting access to ISC operations console.	224
7-18	User association to a user group	225
7-19	Mapping roles to user groups	227
7-20	First-level automation domain credentials	229
7-21	Case study scenario end-to-end automation policy	231
7-22	Resource reference selection	234
7-23	Relationship definitions	238
7-24	Operations Console: Policy information.	242
7-25	Operations Console: Policy selection	243
7-26	Operations Console: Populated policy information	244

Tables

5-1	End-to-end Automation Adapter automation resources	126
7-1	End-to-end Automation Management Component required groups . .	220

Examples

4-1	Display automation domain	58
4-2	Automation domain status	58
4-3	ServiceIP definition input file	59
4-4	Display the apache_SIP ServiceIP	59
4-5	Display equivalency apache_nieq	61
4-6	Apache HTTP Server automation script	62
4-7	Application definition input file	63
4-8	Verify application resource definition	64
4-9	Create resource group and populate with resources	66
4-10	Display resource group information	66
4-11	Display relationship definitions	68
4-12	Display resource group operational state	69
4-13	Application resources OpState online	70
4-14	Initial ServiceIP definition input file data contents	71
4-15	Display of ServiceIP resource after initial definition	71
4-16	Clear error with start/stop of IBM.RecoveryRM	72
4-17	Display current state of the Apache automation domain	72
4-18	Display state of the Apache automation domain after failure of a node	73
4-19	Display status of the ServiceIP after failure of tsa001	73
4-20	lsrsrc output	74
4-21	Post adapter configuration resource group display	82
4-22	Post adapter configuration resource group display	83
4-23	Display of all resource groups in the Apache automation domain	83
4-24	Display of all applications in the automation domain	83
4-25	ServiceIP resources	84
4-26	Apache automation domain equivalencies	85
4-27	Apache automation domain relationships	86
4-28	ifconfig on tsa001 with End-to-end automation	87
5-1	was_SA_Domain automation domain OpState	94
5-2	IBM.RecoveryRM status	94
5-3	Monitoring script for application resource: WebSphere.sh	97
5-4	IBM.Application WebSphere resource definition file	97
5-5	IBM WebSphere Application Server application resource	98
5-6	IBM.Application db2 resource definition file	100
5-7	Monitoring script for the J2EE application	101
5-8	trade3_start.jacl script	102
5-9	trade3_stop.jacl script	102
5-10	IBM.Application trade3 resource definition file	102

5-11	Application resource OpState	103
5-12	IBM.ServiceIP resource definition file	104
5-13	ServiceIP resource.....	104
5-14	ServiceIP OpState	105
5-15	Determining the CommGroup	106
5-16	Network equivalency status	107
5-17	Resource group membership.....	108
5-18	Resource group operational state	109
5-19	Managed relationships.....	111
5-20	Operational State.....	113
5-21	Application resources OpState online	114
5-22	Operational quorum	115
5-23	IBM.TieBreaker resource definition file	117
5-24	Tie breaker properties	117
5-25	netmon.cf configuration file	118
5-26	End-to-end Automation Adapter resources status	126
5-27	Backing up defined policies	127
5-28	Resource group XML definition	127
6-1	VTAM major node definition.....	133
6-2	COMMAND64 member	134
6-3	Case scenario PARMLIB HSAPRM00.....	135
6-4	Case scenario Automation Manager startup procedure	138
6-5	Case scenario Automation Manager start	140
6-6	Our Netview subsystem interface (AOFASSI)	141
6-7	AOFASSI Initialization Message	142
6-8	NetView startup procedure (Agent)	142
6-9	AOFAPPL initialization message	143
6-10	Customization Dialog Primary Menu	147
6-11	Adding IBM DB2 sample Policy Database.....	148
6-12	Define new IBM DB2 Policy Database.....	148
6-13	Select DB2 Policy Database	148
6-14	Create a New Policy Database	148
6-15	Data set information.....	149
6-16	Policy Database Selection panel	149
6-17	IBM DB2 Automation Policy Database	150
6-18	Define Entry panel	150
6-19	Link Instance to Class panel	151
6-20	DB2_MSTR Policy Selection panel	151
6-21	Command Prefix	152
6-22	Selection of DB2_MSTR	153
6-23	DB2 CONTROL option.....	154
6-24	Our case study scenario DB2 Control Entries panel	154
6-25	Renaming Application to DB8QMSTR.....	155

6-26	Entry Rename panel	156
6-27	DB8QDBM1 Application.	156
6-28	Entry Type Selection panel	157
6-29	Entry Name Selection panel	158
6-30	RELATIONSHIPS: Policy Selection panel	158
6-31	Define Relationship to JES2	159
6-32	Relationship Selection List panel	159
6-33	Adding additional resource	160
6-34	Adding additional supporting resource.	160
6-35	Message Processing	161
6-36	Entry Name Selection panel	162
6-37	Policy Selection panel	162
6-38	Define Relationship of C-DB2_DEPENDENTS	163
6-39	Relationship Selection List.	163
6-40	Link Instance to Class called C_DB2_DEPENDENTS	164
6-41	Link Class to Instances	165
6-42	SA z/OS 3.1 Customization Dialog Primary Menu.	166
6-43	Data Management Menu	166
6-44	Import entries from other Policy Database.	166
6-45	Entry Type Selection Panel	167
6-46	Policy Data Base Selection	168
6-47	Import entries from other Policy Database.	168
6-48	Class Entry Name	169
6-49	Selected Entry Names for Import.	169
6-50	Confirm Entry Name List For Import	169
6-51	Start of import process.	170
6-52	Entry Type Selection	170
6-53	Define New Entry Type	171
6-54	Define new entry of type ApplicationGroup	171
6-55	Updating the Automation Name.	172
6-56	Selecting the group to a system.	173
6-57	Where Used.	173
6-58	Entry Type Selection	174
6-59	ApplicationGroup	174
6-60	Policy Selection	175
6-61	Applications For ApplicationGroup.	175
6-62	Selections for Applications for ApplicationGroup panel	176
6-63	NetView Logon screen.	177
6-64	NetView main menu.	177
6-65	IBM Tivoli System Automation for z/OS Main Menu	178
6-66	SDF panel: DISPSTAT	178
6-67	IBM DB2 application dependencies	179
6-68	Checking APPLGR_DB8Q Group	180

6-69	APPLGR_DB8Q members	180
6-70	DISPAOPS command response	185
6-71	Message adapter task	187
6-72	IHSAMCFG member settings	188
6-73	AOFAEVNT startup procedure	188
6-74	End-to-end Automation Adapter startup script	191
6-75	End-to-end Automation Adapter master configuration file	193
6-76	End-to-end Automation Adapter plug-in configuration file	195
6-77	The JAAS configuration file and adapter.jaas.properties	196
6-78	Status of resources E2E_EAS and E2E_ADPT	197
6-79	AOFAADPT application console output	198
7-1	Database verification	212
7-2	EAUTODB database tables	213
7-3	Automation engine status	213
7-4	ISC startup	216
7-5	PolicyInformation element definition	232
7-6	ResourceReference element definitions	235
7-7	ResourceGroup element definitions	237
7-8	Relationship element definitions	238
7-9	Policy checker tool	240

Notices

This information was developed for products and services offered in the U.S.A.

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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


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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

eServer®
Redbooks (logo) ™
pSeries®
xSeries®
z/OS®
zSeries®
AIX®
CICS®
Database 2™
DB2®

ESCON®
FICON®
Geographically Dispersed
Parallel Sysplex™
GDPS®
IBM®
IMS™
Language Environment®
MVS™
NetView®

OMEGAMON®
OS/390®
Parallel Sysplex®
Redbooks™
RACF®
RMF™
Tivoli®
VTAM®
WebSphere®

The following terms are trademarks of other companies:

Java, JDBC, JVM, J2EE, PDB, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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

Other company, product, or service names may be trademarks or service marks of others.

Preface

IBM Tivoli System Automation for Multiplatforms monitors and automates applications distributed across Linux®, AIX®, and z/OS® operating systems by introducing a new product structure with two major components:

- ▶ IBM Tivoli System Automation for Multiplatforms Base Component
Provides high availability and disaster recovery capabilities for Linux, Linux on zSeries® and AIX clusters
- ▶ IBM Tivoli System Automation for Multiplatforms End-to-end Automation Management Component
Provides automated operations and monitoring capabilities for increasing availability and eases operations of heterogeneous business applications

IBM Tivoli System Automation for Multiplatforms utilizes an adapter infrastructure to integrate with IBM Tivoli System Automation for z/OS, allowing for more effective high availability, automation, and management of multi-tier applications.

This IBM Redbook introduces the new versions of the IBM® Tivoli® Systems Automation product family and gives you a broad understanding of the new architecture and components of both IBM Tivoli System Automation for Multiplatforms V2.1 and IBM Tivoli System Automation for z/OS V3.1 using a scenario-based approach.

This redbook is a valuable addition to the existing product documentation and should be read in conjunction with the official product documentation, which complements some of the concepts explained in this redbook.

The instructions given in this redbook are very detailed and explicit. These instructions are not the only way to install the products and related prerequisites. They are meant to be followed by anyone to successfully install, configure, and set up end-to-end automation management using IBM Tivoli System Automation for Multiplatforms V2.1 and IBM Tivoli System Automation for z/OS V3.1 in environments of any size.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Edson Manoel is a Certified IT Specialist at IBM working in the ITSO, Austin Center, in the systems management area. Prior to joining the ITSO, Edson worked in the IBM Software Group, Tivoli Systems, and in IBM Brazil Global Services Organization. He was involved in numerous projects in designing and implementing systems management solutions for IBM clients and Business Partners. Edson holds a Bachelor of Science degree in applied mathematics from Universidade de Sao Paulo, Brazil.

Desmond Krishna is a System Engineer at Standard Bank of South Africa for Group IT specializing in IBM Tivoli System Automation for z/OS. His areas of expertise are NetView®, SA z/OS, and Mainframe Operations. He has 24 years experienced in the IT industry, of which 6 years have been in Mainframe Systems Automation.

Randy R. Watson is an IBM Certified IT Specialist assigned to the Global Response Team (GRT) within IBM Software Services Tivoli (ISST) and supports customers around the world using Tivoli Workload Scheduler, Tivoli Systems Automation for Multiplatforms, Tivoli Storage Manager and the Tivoli Framework core products. He joined IBM in 1995 and has been a software developer, systems software support consultant, systems programmer, project manager and software services consultant working with most major distributed computing platforms and the IBM Mainframe. He is a certified ITIL process consultant with extensive experience in multiple industries and countries. Randy holds a degree in Computer and Information Science from Ohio State University. He currently resides in the Miami Florida area.

Creighton Hicks is a member of the IBM Tivoli Software Advanced Technologies (SWAT) team responsible for technical pre-sales with IBM Tivoli System Automation for Multiplatforms, IBM Tivoli Provisioning Manager, and IBM Tivoli Intelligent Orchestrator. He joined IBM in 2001 and has worked in Level 3 Support for IBM Tivoli Enterprise Console and in software development under the Extreme Blue program. Creighton holds a Bachelor of Science degree in Computer Sciences from The University of Texas at Austin.

Thanks to the following people for their contributions and technical guidance and review during the development of this redbook:

Bob Haimowitz

Budi Darmawan

International Technical Support Organization, Austin Center

Dennis Sample

IBM Software Group - IBM Tivoli Systems Automation Product Management

Barbara Fierro

Matthew Boulton

IBM Software Group - Product Introduction

Chad Smith

Moji Trasti

IBM Software Group - Solutions Test

Bernd Dowedeit

Bernd Jostmeyer

Claus Rauh

Elmar Meyer zu Bexten

Enrico Joedecke

Frank Blaschka

Joachim Schmalzried

Matthias Haeussler

Sven Lange-Last

Sylvia Koch

Thomas Drews

Wolfgang Schawer

IBM Software Group - IBM Tivoli Systems Automation Development, Boeblingen
Germany

Rainer Rentschler

Ruth Nolting

IBM Software Group - Information Development, Boeblingen Germany

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbook@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. JN9B Building 905
11501 Burnet Road
Austin, Texas 78758-3493



Part 1

Fundamentals

Archived

IBM Tivoli System Automation for Multiplatforms V2.1

This chapter gives an overview of the major features and functionality of IBM Tivoli System Automation for Multiplatforms V2.1.

We discuss the following topics:

- ▶ Overview of IBM Tivoli System Automation for Multiplatforms V2.1 components, features, and functionality
- ▶ View of the IBM Tivoli System Automation for Multiplatforms V2.1 Base Component
- ▶ Introduction to IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component
- ▶ Communication between the various components of IBM Tivoli System Automation for Multiplatforms V2.1
- ▶ Concepts and terminology used in this IBM Redbook

1.1 IBM Tivoli System Automation for Multiplatforms V2.1 overview

IBM Tivoli System Automation for Multiplatforms V2.1 consists of two major components:

1. IBM Tivoli System Automation for Multiplatforms Base Component
2. IBM Tivoli System Automation for Multiplatforms End-to-end Automation Management Component

IBM Tivoli System Automation for Multiplatforms V2.1 provides the capability to manage the availability of applications running on AIX or Linux systems using the Base Component. The End-to-end Automation Management Component manages the availability of applications running on a heterogeneous mixture of Linux, AIX, and z/OS clusters.

IBM Tivoli System Automation for Multiplatforms uses policies to determine what actions are required to maintain a system's health in response to an event and issues commands to perform those actions, including: activities such as shutting down the components of an application and moving them to another system.

1.1.1 Main features

The two major IBM Tivoli System Automation for Multiplatforms components above provide the following key features:

High availability and resource monitoring

IBM Tivoli System Automation provides a high availability environment for applications and business systems. High availability describes a system which is continuously available and which has a self-healing infrastructure to prevent downtime caused by system problems. Thus it relieves operators from manual monitoring, remembering application components and relationships, and can eliminate operator errors.

Policy-based automation

IBM Tivoli System Automation for Multiplatforms allows you to configure high availability systems through the use of policies that define the relationships among the various components. Once you establish the relationships, IBM Tivoli System Automation for Multiplatforms will assume responsibility for managing the applications on the specified nodes as configured per policy. With IBM Tivoli System Automation for Multiplatforms V2.1, policy and resource definition can be produced using XML-based definition files. For more details, see 7.5, "Defining the end-to-end automation policy" on page 229.

Automatic recovery

IBM Tivoli System Automation for Multiplatforms quickly and consistently performs an automatic restart of failed resources or whole applications either in place or on another system of a Linux or AIX cluster.

Automatic movement of applications

IBM Tivoli System Automation for Multiplatforms manages the cluster-wide relationships among resources for which it is responsible. If applications need to be moved among nodes, IBM Tivoli System Automation for Multiplatforms automatically handles the start and stop relationships, node requirements, and any preliminary or follow-up actions.

Resource grouping

You can group resources together in IBM Tivoli System Automation for Multiplatforms. Once grouped, all relationships among the members of the group can be established, such as location relationships, or start and stop relationships. After you complete configuration, operations can be performed against the entire group as a single entity.

End-to-end automation management

IBM Tivoli System Automation for Multiplatforms now provides all the above features for a heterogeneous server environment (z/OS, Linux, and AIX) enabling true business application automation, as you see in Figure 1-1.

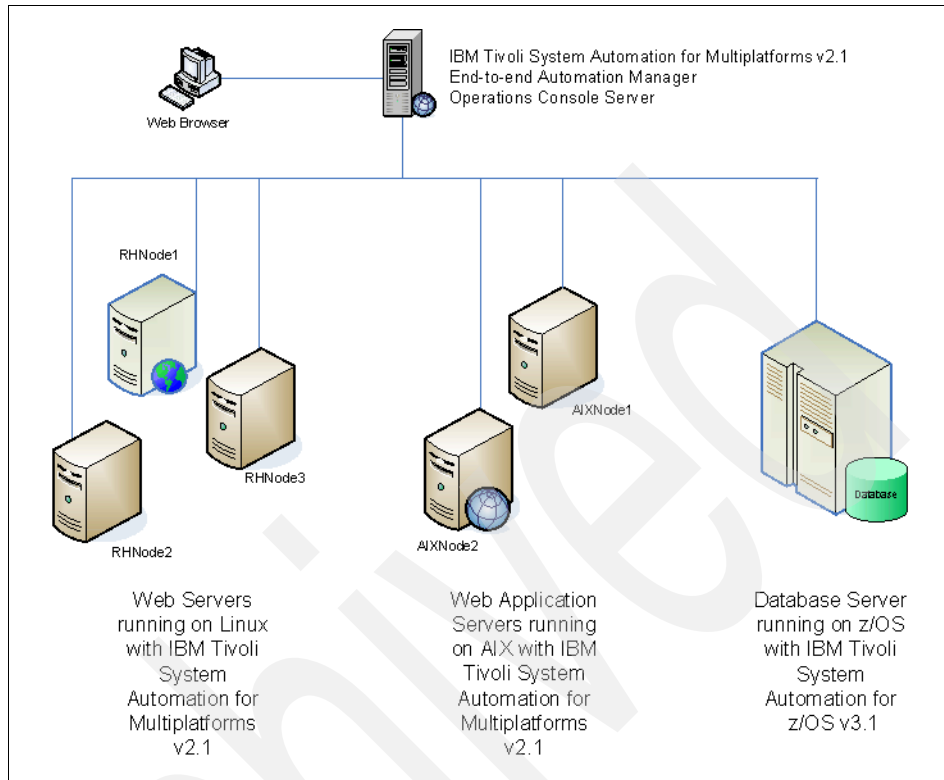


Figure 1-1 IBM Tivoli System Automation for Multiplatforms V2.1 components

The IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component performs the End-to-end automation management capability. We present this component in more detail later in this chapter.

1.2 Base Component overview

As we discussed earlier in this chapter, IBM Tivoli System Automation for Multiplatforms V2.1 Base Component provides features and capabilities to manage the availability of applications running on AIX or Linux systems.

As an example of how you can use these features and capabilities, consider an organization with three application servers (node1, node2, and node3) and a single static IP Address (IPAddr1). The application (appl1) must be active on one and only one of the three available nodes at any time with IPAddr1.

Now, node1 is scheduled for maintenance on Saturday at 9:00 pm. At the scheduled change window time (9pm), node1 is taken offline via IBM Tivoli System Automation for Multiplatforms by an operator. As it turns out, node1 is the active server, IBM Tivoli System Automation for Multiplatforms sees the pseudo outage and moves the IPAddr1 to node2 and starts the application on node2, all without additional operator intervention. The node1 is placed back into online mode at the end of the change window. The following Saturday, node2 is scheduled for maintenance and the above is repeated in reverse (node1 or node3 is made active). This assumes that node2 remained the active application server during the intervening week. There are other variations possible with this simple scenario, such as, the maintenance node is not the active application server, among others. See Figure 1-2.

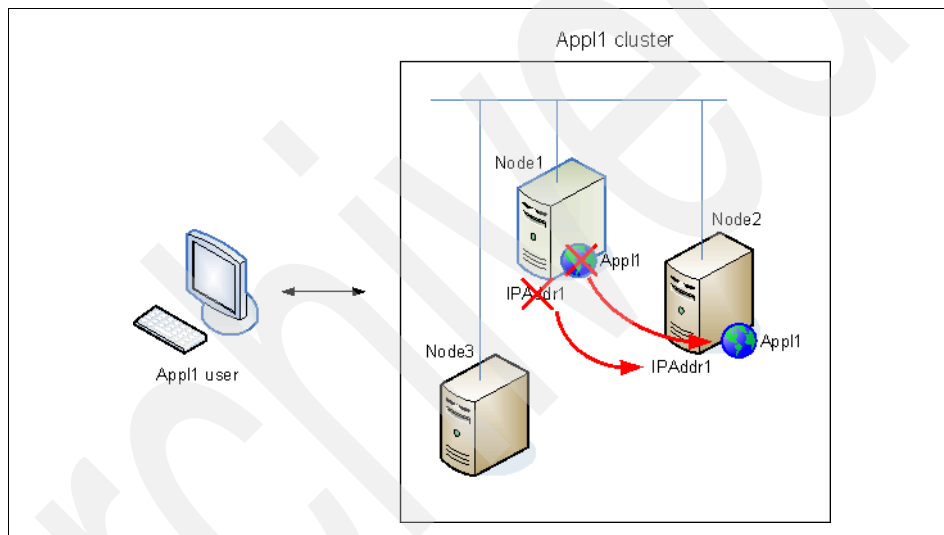


Figure 1-2 Sample scenario: Base Component

While this is a simple scenario, it shows the general idea of IBM Tivoli System Automation for Multiplatforms.

The key products of IBM Tivoli System Automation for Multiplatforms V2.1 Base Component are:

- ▶ Reliable Scalable Cluster Technology
- ▶ Resource Managers
- ▶ End-to-end Automation Adapter

We discuss these products in the following sections.

1.2.1 Reliable Scalable Cluster Technology

Reliable Scalable Cluster Technology (RSCT) is a software product that provides a comprehensive clustering environment for AIX and Linux. RSCT is the infrastructure used by IBM Tivoli System Automation for Multiplatforms to provide Linux and AIX clusters with improved system availability and scalability. We list the major components of RSCT below, and you can learn more about these RSCT components in the *IBM Reliable Scalable Cluster Technology Administration Guide*, SA22-7889.

- ▶ Resource Monitoring and Control (RMC)
- ▶ High Availability Group Services (HAGS)
- ▶ High Availability Topology Services (HATS)

1.2.2 Resource Managers

The core components of IBM Tivoli System Automation for Multiplatforms are user-defined automation policies to monitor and control cluster resources. IBM Tivoli System Automation for Multiplatforms categorizes cluster resources in predefined classes. These resource classes are managed by the various IBM Tivoli System Automation for Multiplatforms resource managers (RM), depending on what type of resource is being managed. Resource managers are the software layer acting as interface between resources and RSCT, specifically, RMC.

The main resource managers provided by IBM Tivoli System Automation for Multiplatforms include:

- ▶ Recovery RM (IBM.RecoveryRM)

This resource manager serves as the decision engine for IBM Tivoli System Automation for Multiplatforms. Once a situation develops that requires intervention, the Recovery RM drives the decisions defined in the automation policy.

- ▶ Global Resource RM (IBM.GblResRM)

The Global Resource RM provides the classes that define the behavior of application and IP address resources. These are the IBM.Application and IBM.ServiceIP resource classes.

- ▶ Configuration RM (IBM.ConfigRM)

This resource manager is used during cluster definitions. It also provides means of ensuring data integrity via cluster quorum support (See 1.5, “Concepts and terminology” on page 24).

- Event Response RM (IBM.ERRM)

Allows IBM Tivoli System Automation for Multiplatforms to monitor conditions and situations in the cluster.

These resource managers provide IBM Tivoli System Automation for Multiplatforms with the capabilities to define, monitor, group, and manage resources within a Linux or AIX cluster.

1.2.3 End-to-end Automation Adapter

The IBM Tivoli System Automation for Multiplatforms V2.1 Base Component uses the End-to-end Automation Adapter to provide services needed by the new End-to-end Automation Management Component.

There can be only one End-to-end Automation Adapter per IBM Tivoli System Automation for Multiplatforms cluster (see cluster definition in 1.5, “Concepts and terminology” on page 240). Therefore, we highly recommend you have the End-to-end Automation Adapter highly available. You accomplish this by configuring End-to-end Automation Adapter as a resource managed by the IBM Tivoli System Automation for Multiplatforms Base Component.

We provide more information about the End-to-end Automation Adapter in the following section.

1.3 End-to-end Automation Management Component overview

In this section, we describe the general features, capabilities, and components of IBM Tivoli System Automation for Multiplatforms End-to-end Automation Management Component.

The End-to-end Automation Management Component is new to IBM Tivoli System Automation for Multiplatforms V2.1 and provides coordinated, cross-cluster automation. This capability allows organizations to operate their homogeneous (Linux, AIX, and z/OS) clusters, managed by IBM Tivoli System Automation for Multiplatforms Base Component, in an integrated manner.

While High Availability Clusters have greatly improved system availability, organizations also require high availability of business applications. Many of these applications require the services of heterogeneous systems environments. IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component provides the ability to safely ensure higher business application availability.

End-to-end Automation provides the capability to automate the operation of resources within heterogeneous environments (called *first-level automation domains*) that each have a local automation technology of their own. For example, consider a multi-tiered business application which components run in heterogeneous platforms. Each tier of this application runs on a dedicated infrastructure, such as HTTP Servers on Linux, the application servers on AIX, and database servers on a z/OS Sysplex, as seen in Figure 1-3.

As each tier of this business application is made high available by IBM Tivoli System Automation for Multiplatforms and IBM Tivoli System Automation for z/OS, the End-to-end Automation Management Component is able to ensure high availability of the entire infrastructure used by the business application. This is accomplished by defining resources, and logical relationships between them. In Figure 1-3, we show some of the possible relationships. For example, the Application resource will only be started after the resource Database starts completely, and the HTTP resource will be forced offline in case of a failure of all resources that belong to Group01.

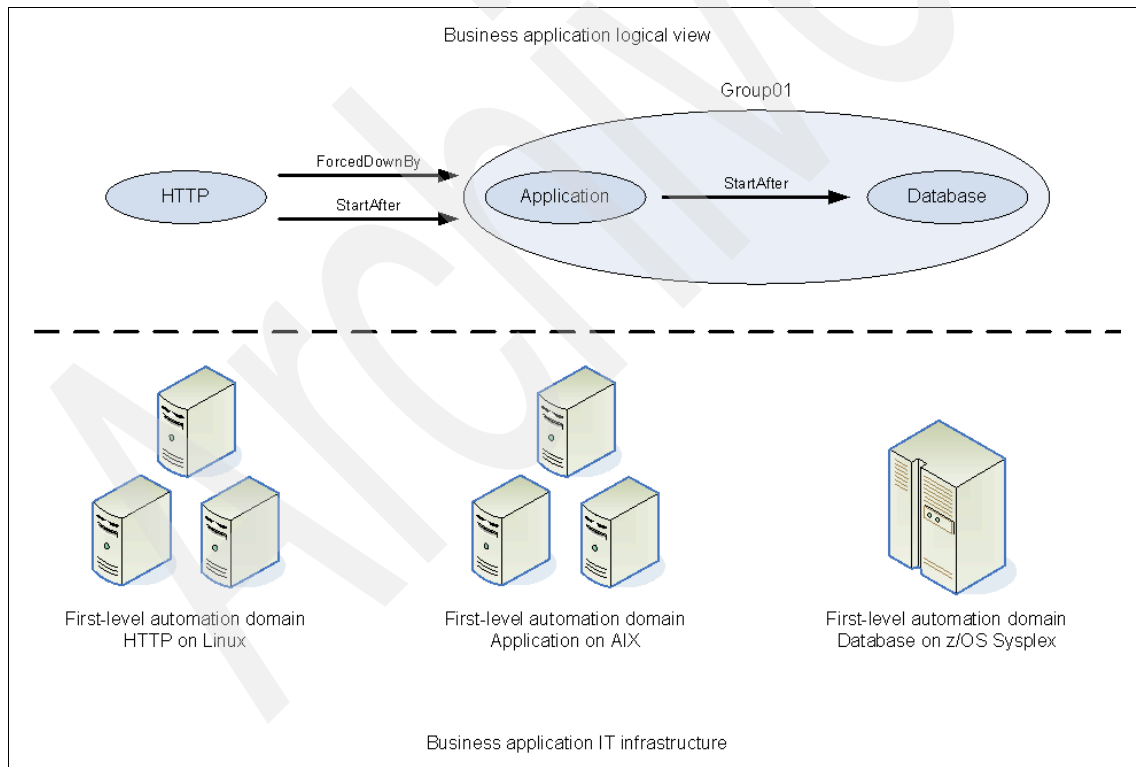


Figure 1-3 Sample Scenario: End-to-end Automation Management Component

The core parts that make up the End-to-end Automation Management Component of IBM Tivoli System Automation for Multiplatforms are:

- ▶ Automation Engine
- ▶ End-to-end Automation Manager
- ▶ Operations Console
- ▶ Automation database
- ▶ Automation policy
- ▶ End-to-end Automation Adapter

Figure 1-4 shows the End-to-end Automation Management Component and the relationships among the parts that make it up.

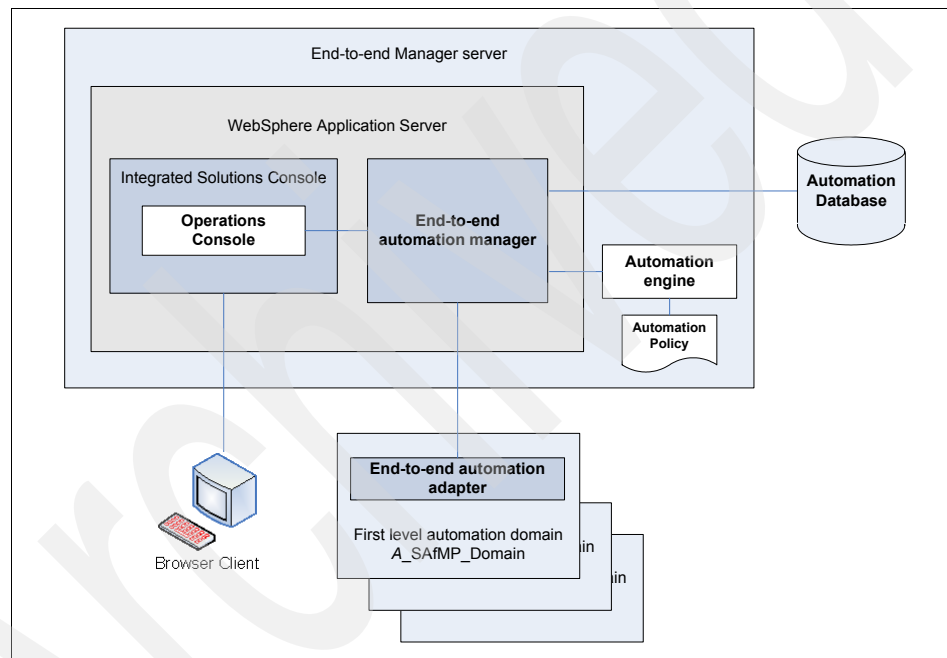


Figure 1-4 Basic role of the End-to-end Automation Manager

Note: The above figure shows the IBM Tivoli System Automation for Multiplatforms Operations Console and the Integrated Solutions Console on the same physical server. It is also possible to have it installed on a separate machine from the End-to-end Manager server using its own IBM WebSphere Application Server instance.

1.3.1 Automation Engine

The automation policy is notified of state changes of resources managed by the End-to-end Automation Management Component, compares that state to the desired state defined in the automation policies, and determines the appropriate set of actions to take. The Automation Engine must run on the same system on which IBM WebSphere Application Server hosting the End-to-end Automation Manager J2EE™ application environment (IBM WebSphere Application Server) runs.

1.3.2 End-to-end Automation Manager

The End-to-end Automation Manager logically sits between the Operations Console and the first-level automation domains with interaction with the automation engine. The End-to-end Automation Manager stores resource state information in the automation database. Once an automation policy is activated, the End-to-end Automation Manager will monitor and manage first-level automation domains according to the established automation policy and operator requests. The automation domain controlled by the End-to-end Automation Manager is often referred to as the *end-to-end automation domain*.

The basic configuration of the End-to-end Automation Manager is done at End-to-end Automation Management Component installation time. However, additional configuration information is still required, or changes to the existing configuration information can be performed. End-to-end Automation Management Component provides a End-to-end Automation Manager configuration tool. You can find additional information in Chapter 7, “Case study scenario: End-to-end automation domain” on page 201.

The End-to-end Automation Manager makes use of an adapter to communicate with the first-level automation domains. This adapter is named the *first-level automation manager resource adapter*.

1.3.3 Operations Console

The Operations Console is the web-based graphical user front-end that provides Operators access and control to the end-to-end automation domain and to the first-level automation domains. For an example, see Figure 1-5 on page 13.

The Operations Console is Integrated Solutions Console (ISC)-based and provides Operators Web-based access to:

- ▶ Activate and deactivate automation policies
- ▶ Monitor and perform problem analysis
- ▶ View logs and trace information
- ▶ Control resources and submit requests against them

- Control cluster configurations
- Work with console user preferences

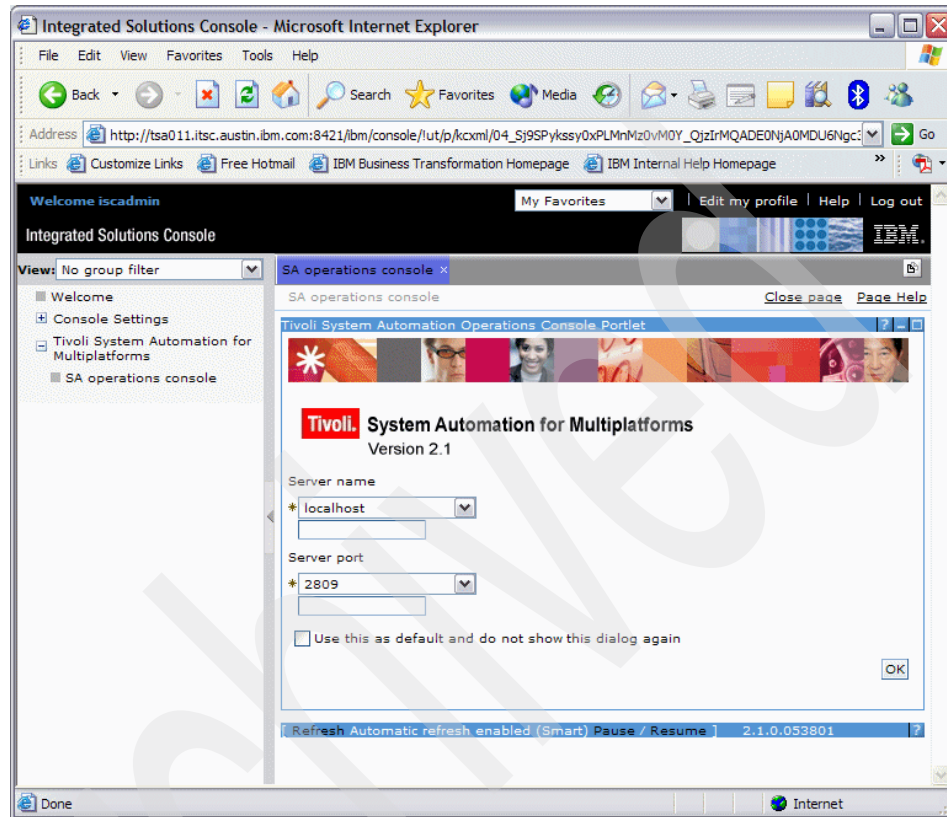


Figure 1-5 IBM Tivoli System Automation for Multiplatforms Operations Console

The Operations Console can be used in three different modes:

- End-to-end automation mode
Operators can manage and monitor resources controlled by the End-to-end Automation Manager as well as resources of all first level domains that are connected to the End-to-end Automation Manager.
- First-level automation mode
Operators can manage and monitor resources controlled by both IBM Tivoli System Automation for Multiplatforms Base Component and IBM Tivoli System Automation for z/OS.
- Direct access mode

Operators can manage and monitor resources controlled by IBM Tivoli System Automation for Multiplatforms Base Component only.

1.3.4 Automation database

The automation database stores persistent information about the end-to-end automation domain and all the first-level automation domains controlled by the the End-to-end Automation Manager. The automation database also contains partial information contained in the automation policy.

IBM Tivoli System Automation for Multiplatforms V2.1 only supports IBM DB2 UDB Enterprise Edition Version 8.2.3.

1.3.5 Automation policy

The automation policy contain definitions of all resource references, resource groups, and their relationships and desired state (See definitions in 1.5, “Concepts and terminology” on page 24).

Automation policies are defined using XML format. IBM Tivoli System Automation for Multiplatforms V2.1 provides a schema definition to ease the definition of automation policies. The schema for the automation policy is named EEZPolicy.xsd. IBM Tivoli System Automation for Multiplatforms V2.1 also provides a policy checking tool so that automation policy files can be verified before activation.

Several automation policy files may be available for the automation engine, but only one will be active at any time.

Additional details can be found in 7.5, “Defining the end-to-end automation policy” on page 229, when we define the automation policy we use in our case study scenario environment.

1.3.6 End-to-end Automation Adapter

An automation adapter process must run on each first-level automation domain that the End-to-end Automation Management Component will manage. The automation adapter process provides an interface between the first-level automation manager resource adapter and the End-to-end Automation Manager.

The functions of the End-to-end Automation Adapter include:

- Monitor resources defined in the IBM Tivoli System Automation for Multiplatforms cluster.

- Communicate resource status and attribute changes to the End-to-end Automation Management Component.
- Perform tasks mandated by the End-to-end Automation Management Component to resources defined in the IBM Tivoli System Automation for Multiplatforms cluster.

The End-to-end Automation Adapter uses the Tivoli Event Integration Facility (EIF) to communicate with the End-to-end Automation Management Component.

IBM Tivoli System Automation for Multiplatforms provides a tool for configuring the End-to-end Automation Adapter. Later in this IBM Redbook, we present a case study scenario in which we configure the End-to-end Automation Adapter on different IBM Tivoli System Automation for Multiplatforms environments, including Linux, AIX, and z/OS environments.

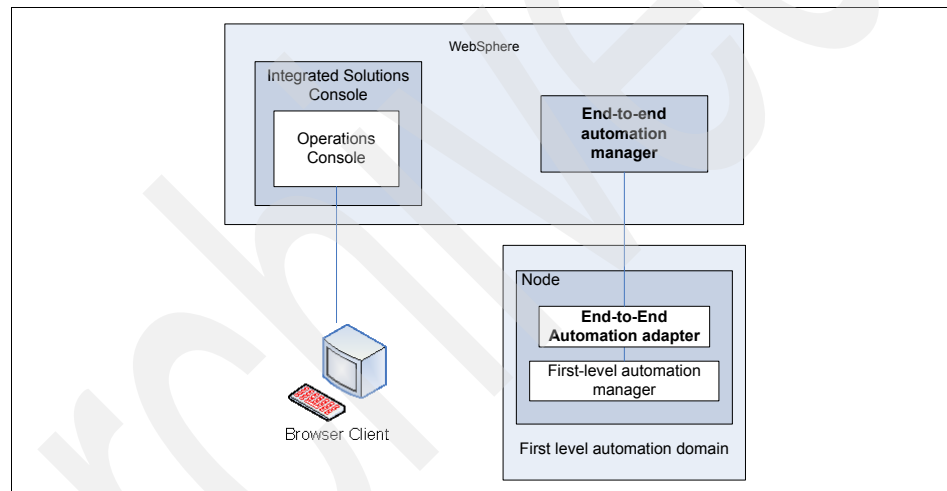


Figure 1-6 Basic role of the End-to-end Automation Adapter

1.4 Communication between end-to-end components

Below is a more detailed overview of the End-to-end Automation Management Component environment. The depiction is one possible configuration, showing the Operations Console and End-to-end Automation Manager on the same node. It is possible to host the Operations Console on a separate system. Figure 1-7 on page 16 shows an overview of the configuration and communications for the End-to-end Automation Adapter and the End-to-end Automation Manager with a

single first-level automation domain manager. The port numbers represent the default communication ports' values.

IBM Tivoli System Automation for Multiplatforms End-to-end Automation Management Component uses the Tivoli Event Integration Facility (EIF) to communicate the status changes of resources.

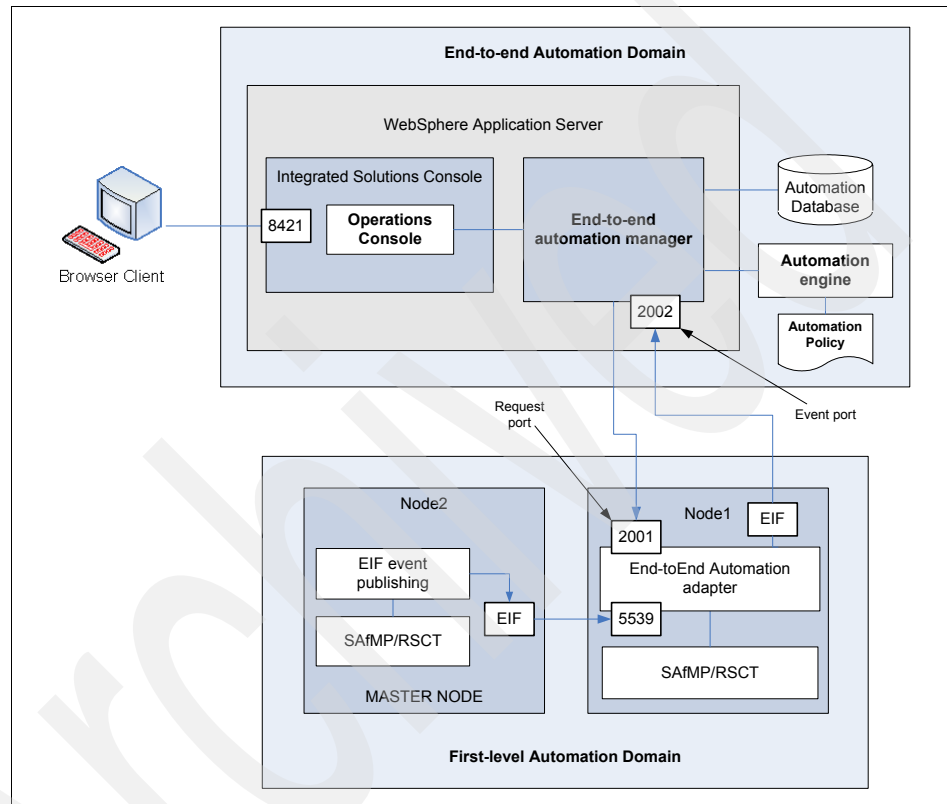


Figure 1-7 Communication overview

All communications between the End-to-end Automation Manager and the End-to-end Automation Adapters running on first-level automation domains are secure and use the SSH protocol.

As mentioned earlier, the End-to-end Automation Manager uses the first-level automation manager resource adapter for communication with the End-to-end Automation Adapters running on first-level automation domains. This adapter is responsible for an asynchronous communication to the first-level automation domains.

For a more detailed overview of the communication flows between the various components of IBM Tivoli System Automation for Multiplatforms End-to-end Automation Management Component, we describe the following scenarios.

- ▶ End-to-end automation management environment startup
- ▶ First-level automation adapter startup
- ▶ Resource monitoring
- ▶ Requests against a resource reference
- ▶ Events from referenced resource

You can find additional scenarios in the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-End Automation Management User's Guide and Reference, SC33-8211-00*, manual.

End-to-end automation management environment startup

The following communication flow figure (Figure 1-8) shows the actions that are taken between end-to-end components when the end-to-end automation management environment is started.

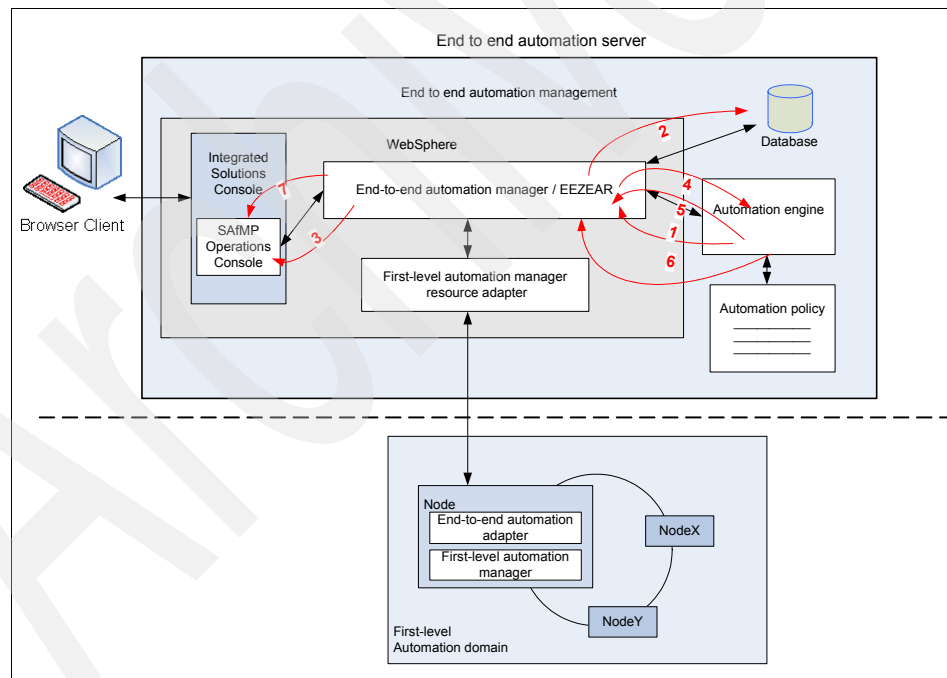


Figure 1-8 End-to-end automation management environment startup workflow

1. The automation engine component starts and then sends an event to the End-to-end Automation Manager (EEZEAR) to join the domain.

2. The End-to-end Automation Manager checks the automation database to determine the last activated automation policy in the end-to-end automation domain.
3. Once the End-to-end Automation Manager has successfully processed the event generated in step 1, it sends an event to inform the Operations Console of the successful join. This will cause the end-to-end automation domain to be listed in the topology tree of the Operations Console.
4. The End-to-end Automation Manager makes a request to the automation engine for activation of the last active automation policy determined in step 2. The automation engine creates all resources, groups, and relationships as per policy definition.
5. The automation engine issues a subscription request to all first-level automation domains in which there is a resource that is referenced in the active automation policy per the automation database.
6. The automation engine is notified of the state of resources controlled by the first-level automation domain and sends an event to notify of a change in domain policy.
7. The Operations Console receives the event in Step 6 and causes the state of the resource to change accordingly.

First-level automation adapter startup

The following communication flow figure (Figure 1-9) shows what actions are taken between end-to-end components when a first-level automation adapter is started.

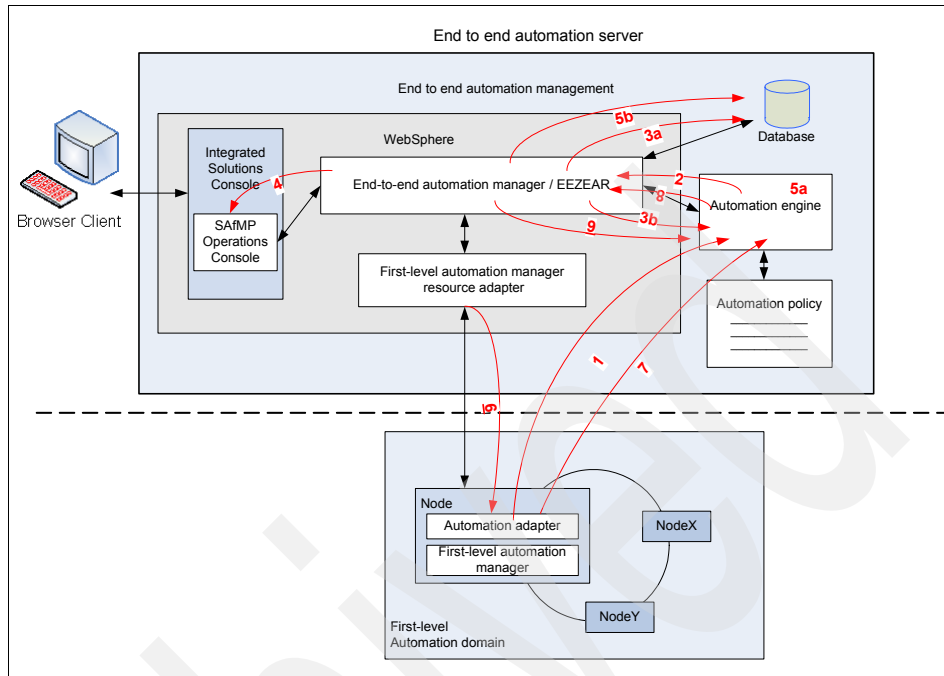


Figure 1-9 First-level automation adapter startup

1. The End-to-end Automation Adapter is started on the node on the first-level automation domain, and it sends an event to the automation engine to request to join the end-to-end automation domain.
2. The automation engine converts the EIF event sent in step 1 to a JMS event that is sent to the End-to-end Automation Manager.
3. The End-to-end Automation Manager checks on the automation database (3a) if the automation engine is subscribed to the first-level automation domain that sent the original event in step 1. The End-to-end Automation Manager also sends an event to the automation engine (3b).
4. Once the End-to-end Automation Manager has processed the event sent to it in step 2 successfully, it will send an event notifying the successful join of the first-level automation domain to the Operations Console. The first-level automation domain is now shown in the Operations Console's topology tree as active.
5. The automation engine processes the event sent to it in Step 3 and subscribes to the resources on the first-level automation domain which are referenced by the active automation policy (5a). The subscriptions are added to the automation database (5b).

6. Each subscription generated in Step 5 is forwarded through the first-level automation manager resource adapter to the End-to-end Automation Adapter.
7. The End-to-end Automation Adapter will confirm each resource subscription with an event to the automation engine.
8. The automation engine converts the EIF event sent in step 7 to a JMS event that is sent to the End-to-end Automation Manager.
9. The End-to-end Automation Manager checks if the automation engine is subscribed to the first-level automation domain that sent the event in step 7. The End-to-end Automation Manager sends an event to the automation engine.

Resource monitoring

The following communication flow figure (Figure 1-10) shows what actions are taken between end-to-end components when an operator requests a resource to be monitored.

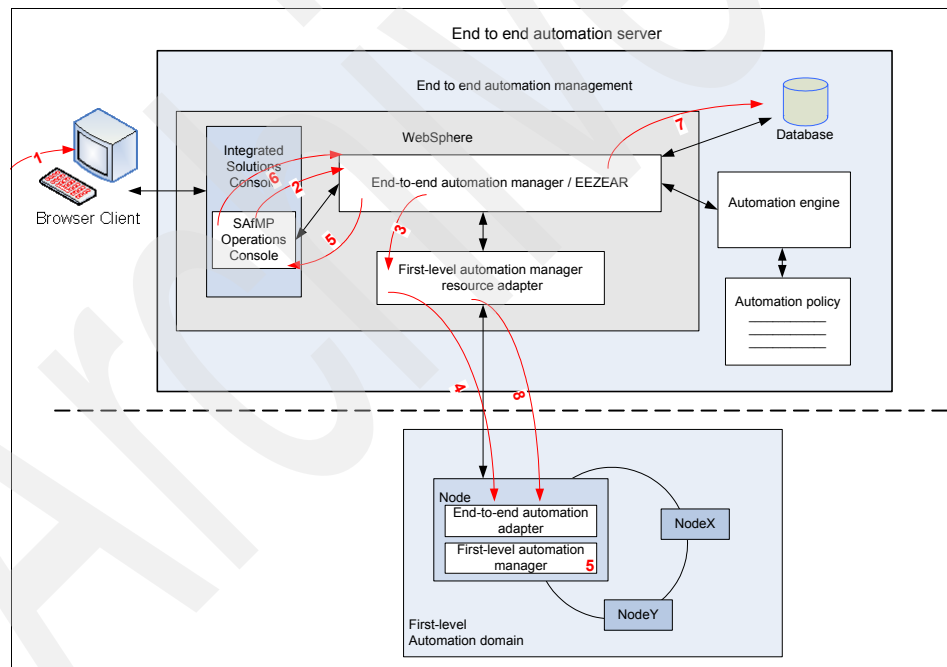


Figure 1-10 Resource monitoring workflow

1. Using the browser client, the operator selects an automation domain in the topology tree on the Operations Console to view the resources of that domain.

2. The Operations Console queries the End-to-end Automation Manager for all top level resources of the specified domain.
3. The End-to-end Automation Manager forwards the query to the first-level automation resource adapter.
4. The query is forwarded from the first-level automation resource adapter to the End-to-end Automation Adapter on the first-level automation domain. The resource information is gathered and the list of top level resources is returned through the first-level automation resource adapter.
5. The End-to-end Automation Manager sends an event to inform the Operations Console of the successful query. This will cause the resources to be listed in the resource tree of the Operations Console.
6. The Operations Console sends a subscription request for all returned resources that are not subscribed yet to the End-to-end Automation Manager.
7. The End-to-end Automation Manager adds the current operator to the list of subscribers to the first-level automation manager resource adapter. The subscription request is forwarded to the End-to-end Automation Adapter on the first-level automation domain.

Requests against a resource reference

The following communication flow figure (Figure 1-11) shows what actions are taken between end-to-end components when an operator issues a request against a resource reference.

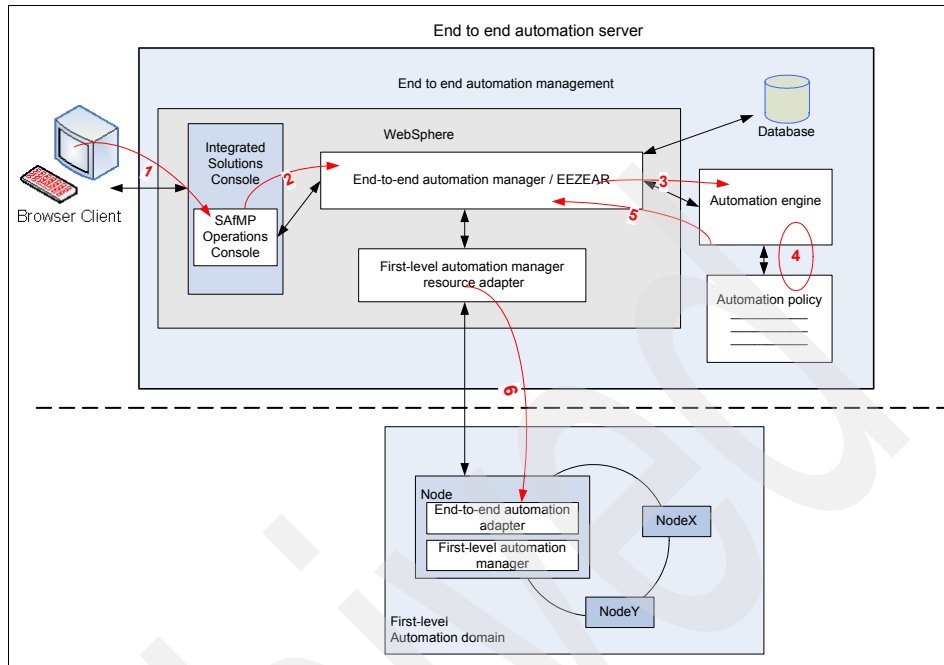


Figure 1-11 Request against a resource reference workflow

1. Using a browser client, the operator submits a request against a resource reference in the Operations Console.
2. The request is forwarded to the End-to-end Automation Manager.
3. The event is forwarded to the automation engine.
4. The automation engine calculates resulting requests which must be issued against referenced resources by taking into account relationships defined between resources in the automation policy.
5. Resulting requests generated in Step 4 are passed from the automation engine to the End-to-end Automation Manager.
6. The events sent to the End-to-end Automation Manager in Step 5 are forwarded through the first-level automation manager resource adapter to the End-to-end Automation Adapter on the first-level automation domain. The End-to-end Automation Adapter forwards the event to the first-level automation manager, which processes the request.

Events from referenced resource

The following communication flow figure (Figure 1-12) shows what actions are taken between end-to-end components when a subscribed resource in a first-level automation domain has a state change.

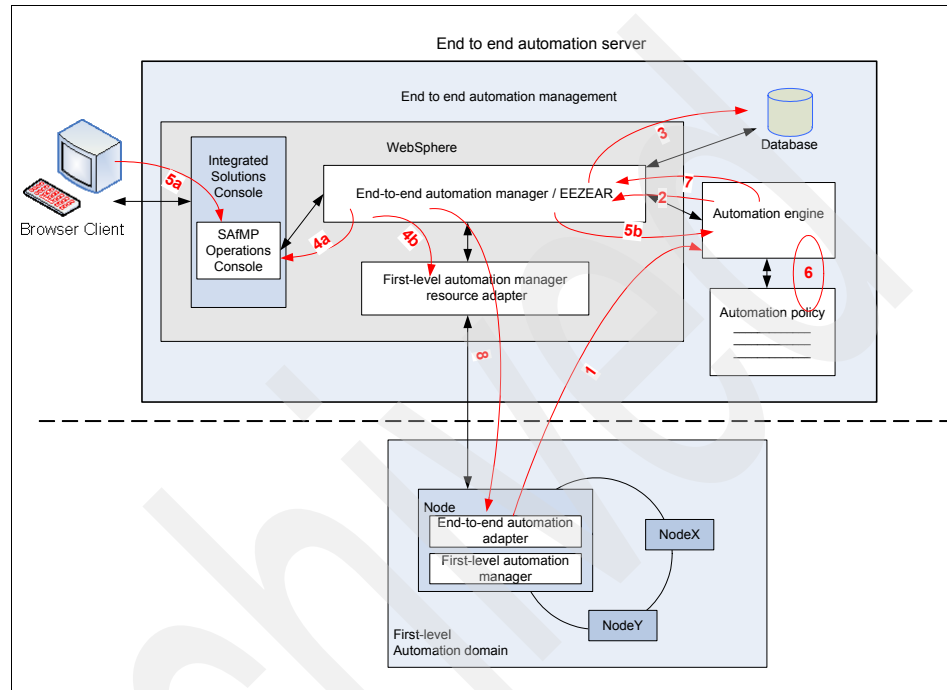


Figure 1-12 Event from a referenced resource workflow

1. The first-level automation manager observes a state change of a resource in the first-level automation domain. Because this is a referenced resource, the End-to-end Automation Adapter sends an event to the automation engine to notify the state change.
2. The automation engine converts the EIF event sent in step 1 to a JMS event that is sent to the End-to-end Automation Manager.
3. The End-to-end Automation Manager queries the list of subscribers for this particular resource stored in the automation database.
4. The JMS event is published where it is received by the Operations Console (4a). In addition, the event is forwarded to the first-level automation manager resource adapter (4b).
5. The JMS event triggers a refresh of the browser window to update the view on the Operations Console (5a). In addition, the End-to-end Automation Manager forwards the event to the automation engine (5b).

6. The automation engine calculates new states of the resource reference pointing to this resource. It also calculates resulting actions against this or other referenced resources.
7. Requests generated in Step 6 are forwarded to the End-to-end Automation Manager.
8. The events sent to the End-to-end Automation Manager in Step 6 are forwarded through the first-level automation manager resource adapter to the End-to-end Automation Adapter, which forwards the event to the first-level automation manager. The first-level automation manager handles the requests accordingly.

1.5 Concepts and terminology

This section discusses IBM Tivoli System Automation for Multiplatforms concepts and some terminology. For more complete information, see the *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04, and the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-End Automation Management User's Guide and Reference*, SC33-8211-00.

1.5.1 High Availability and IBM Tivoli System Automation for Multiplatforms

One definition of high availability is: "The continuous operation of systems over time". There are two properties that are important to keep in mind when examining a high availability solution:

- ▶ Liveness
- ▶ Safety

The liveness property relates to number of points in time that the system is operational, generally the more the better. In addition, there are certain states in the system that lead to failure -- Such as a single IP address being active on two different nodes at the same point in time. This is said to violate the safety property of the system. A system preserves its safety property if it is guaranteed that the system will never enter an unsafe state.

IBM Tivoli System Automation for Multiplatforms strives for the greatest amount of liveness possible with maximum safety. The advantage is that, if implemented correctly, making a system highly available will not compromise the integrity of the system, and, at the same time, will increase the amount of uptime that system achieves with virtually no risk.

IBM Tivoli System Automation for Multiplatforms ensures maximum safety by implementing two concepts (defined later in this chapter):

- ▶ Quorum
- ▶ Tie breaker

IBM Tivoli System Automation for Multiplatforms refers to the set of nodes in the system, commonly called a cluster, as a *peer domain*. All nodes in a peer domain continually send and receive heartbeats over communication groups. A *communication group* is a set of nodes that can talk to each other over a common communication medium. A common example of a communication group would be network interface cards residing on several nodes connected to the same network. IBM Tivoli System Automation for Multiplatforms automatically detects and configures communication groups.

1.5.2 Terms used in IBM Tivoli System Automation for Multiplatforms

This section provides definitions to commonly used IBM Tivoli System Automation for Multiplatforms terminology used in this IBM Redbook.

Clusters

The group of host systems upon which IBM Tivoli System Automation for Multiplatforms manages resources is known as a cluster. A cluster can consist of one or more systems or nodes. Some IBM Tivoli System Automation for Multiplatforms manuals may use the term “peer domain” when referring to a cluster. The two terms are interchangeable. IBM Tivoli System Automation supports up to 32 Linux or AIX nodes within a cluster. The z/OS environment has for quite some time managed clusters using Parallel Sysplex®.

Subclusters

A cluster may split into two or more subclusters in the case where no communication is possible between some set of nodes in the cluster. This could be caused by some form of communications failure. Subclusters are not aware of each other.

A subset of the peer domain that is fully connected by heartbeat is called a subcluster. When all the nodes in a peer domain are fully connected (meaning no network or node failures), then the subcluster is the same as the cluster or peer domain. When a split among the peer domain that creates multiple subclusters occurs, then at the most one subcluster will become the active subcluster. This active subcluster will be the only subcluster that will be allowed to start resources. This subcluster is said to have *operational quorum*. There will be at all times at most one subcluster that will have operational quorum. See Figure 1-13.

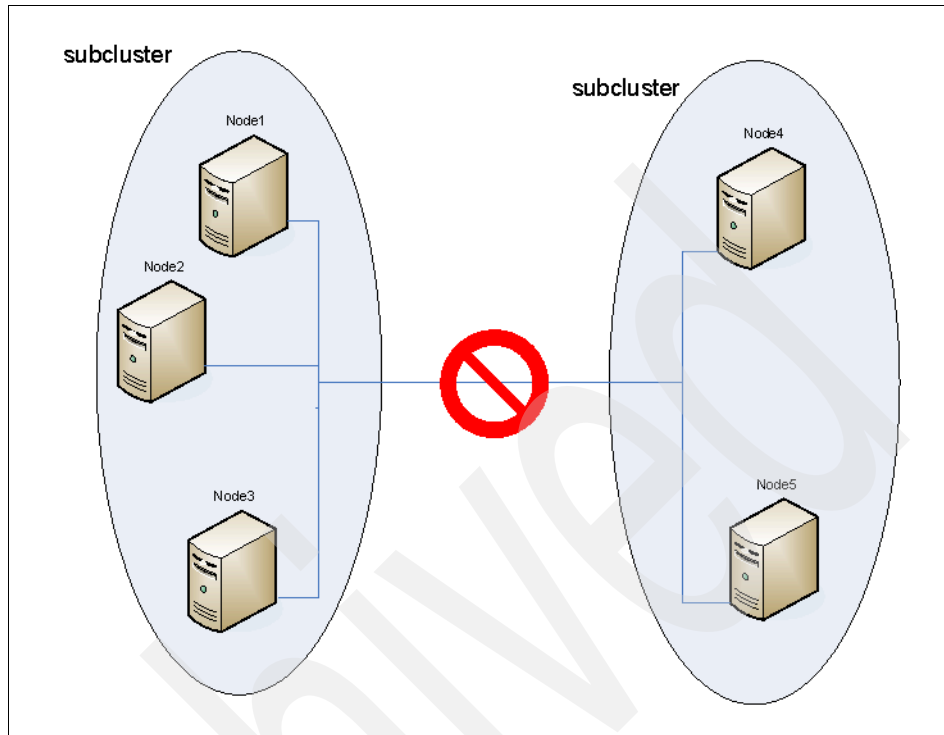


Figure 1-13 Subclusters from a cluster

Quorum

The goal of quorum operations is to keep data consistent and to protect critical resources. Look at quorum as the number of nodes in a cluster that are required to modify the cluster definition or perform certain cluster operations. In general, a subcluster must have a majority of nodes in order to have quorum.

Configuration quorum

Configuration quorum determines when configuration changes will be accepted for the cluster. Changes are permitted only when a strict majority of the nodes are online. All changes to Cluster membership and resource definitions require configuration quorum.

Operational quorum

Operational quorum determines whether a node may be safely started. Operational quorum is determined in the following way: A subcluster must have a clear majority of nodes in the Cluster to gain operational quorum. If we assume that a Cluster has N nodes, then for a subcluster to gain operational quorum it must have *floor* defined by the following expression:

$$\lfloor N/2 \rfloor + 1$$

Operational quorum may be determined with the aid of a Tie breaker.

Tie breaker

Clusters with an even number of nodes require a method to ensure operational quorum in the event that no subcluster has a majority of nodes.

If we have a cluster with two nodes ($N=2$), then a subcluster must have two nodes to maintain operational quorum. This is clearly not the optimal situation so IBM Tivoli System Automation for Multiplatforms contains a Tie breaker function.

A Tie breaker ensures that exactly one subcluster will win the Tie breaker and all others will lose. The one subcluster that wins the Tie breaker will act as if that subcluster has one additional node for the purpose of operational quorum. A Tie breaker is only called when a subcluster has exactly half of the nodes in a cluster. Therefore, for a Tie breaker to be called, the total number of nodes in a Cluster must be even. Now if we have two nodes in our cluster, if the heartbeat between them fails, they will become two subclusters of one node each. At this point both subclusters will enter a race condition for the Tie breaker and only one subcluster will win that Tie breaker. The subcluster that wins the Tie breaker will then gain operational quorum while the other subcluster will lose operational quorum.

There are three main groups of Tie breakers in IBM Tivoli System Automation for Multiplatforms V2.1: Disk Tie breakers (SCSI, ECKD, and so on), the Operator Tie breaker, and the Network Tie breaker.

The Operator Tie breaker involves manual intervention by an operator to resolve operational quorum, and, therefore, is usually discarded for one of the other Tie breakers. It is important to remember that Tie breakers are only useful when the total number of nodes in a cluster is odd.

The Network Tie breaker is implemented as an EXEC Tie breaker meaning that the Tie breaker is issued by executing a script. This script is included in IBM Tivoli System Automation for Multiplatforms V2.1 and no modifications to the script are needed. To define the Network Tie breaker, you must supply the target node (usually the network gateway). When the script is executing, both subclusters attempt to ping the target node a predefined number of times (default is 2). If a node is able to ping the target node, then it wins the Tie breaker, and, therefore, will gain operational quorum. A node that is unable to ping the target node will lose the Tie breaker and, therefore, lose operational quorum.

It is very important to understand how the Network Tie breaker differs from Disk Tiebreakers. In a Disk Tie breaker one subcluster is taking away a resource (for

example, a SCSI disk) from another subcluster. The Network Tie breaker does not do this, but instead relies on the assumption that:

If subcluster A can communicate (ping) with the target and subcluster B can communicate (ping) with the target THEN subcluster A must be able to communicate (heartbeat) subcluster B.

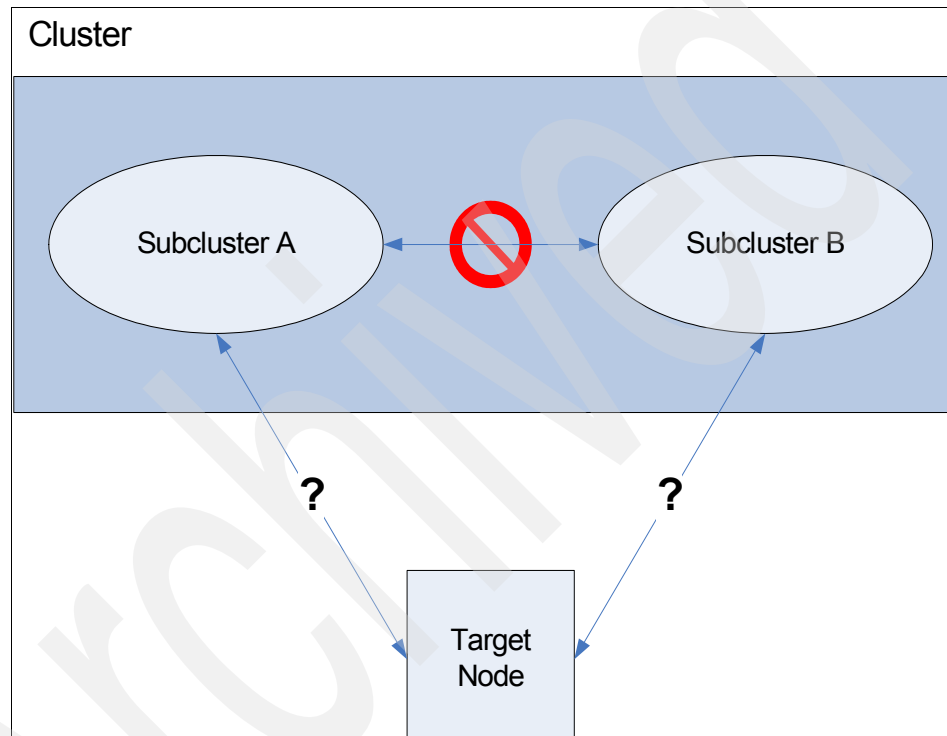


Figure 1-14 Tie breaker

If this assumption was violated in some way, then it could be the case where subcluster A and subcluster B lose heart beat and therefore enter a race condition for the Tie breaker. Furthermore, if both subclusters A and B were able ping the gateway then both subclusters A and B would win the Tie breaker and gain operational quorum. However, since subclusters A and B cannot communicate with each other, this is why the tie breaker was called in the first place, safety would be violated as a split brain would occur.

To prevent this, we recommend that the following be true to use the Network Tie breaker:

- ▶ All nodes in the Cluster are located on the same network segment (in other words, on the same switch).
- ▶ The target used for the Network Tie breaker be the network gateway for all nodes in the Cluster. This ensures that there is only one hop between each node in the Cluster and the target node.

Resource

A resource is any piece of hardware or software that can be defined to IBM Tivoli System Automation. These resources can be either defined manually by the administrator or through the *harvesting* functionality of the cluster infrastructure, where resources are automatically detected and prepared for use, for example, a network interface card.

Critical Resource

A critical resource is a resource that must not be active on more than one node at any point in time. If such a resource is active on two or more separated nodes, then data or functional integrity of the cluster is endangered. The unsafe situation where such a critical resource is running on two separate nodes is referred to as a *split brain situation*. An example is an IP address, which must be active on one and only one node.

When a node loses operational quorum while that node is running a critical resource, then, by default to ensure safety, IBM Tivoli System Automation for Multiplatforms will perform a hard reboot of that node. This is to ensure a maximum blend of safety and liveness. As soon as that node loses operational quorum, if another node is able to, that other node will have operational quorum and try to start the critical resource right away. To maintain that no critical resource exists on two different nodes, we must stop the critical resource on the node that lost operational quorum as soon as possible. The best way to do this is to perform a hard reboot on that node.

Important: The default behavior when a subcluster loses operational quorum is to perform a hard reboot on all nodes running critical resources. We strongly suggest that this behavior is left as default functionality. However, after completely understanding the consequences, you can change this. Instructions can be found in the *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04.

Resource Reference

A virtual resource hosted by the End-to-end Automation Manager that points to an actual resource on other automation domains controlled by IBM Tivoli System Automation for Multiplatforms Base Component or IBM Tivoli System Automation for z/OS. These domains are also known as First Level Automation Domains (FLA).

Resource groups

A collection of resources, resource references (End-to-end Automation Management Component only), or other resource groups that share the same automation goal. The current state of a resource group is an aggregation of the states of its members.

Choice groups

An End-to-end Automation Manager resource group that represents an alternative configuration, for example, a fallback or reduced capacity configuration. A configuration that must be selected or activated by an operator.

Resource States

The actual state of a resource as follows:

- ▶ **Observed State**
Current runtime status of a resource, online, offline, and so on.
- ▶ **Desired State**
Intended target state of a resource as expressed within an automation policy or as a result of operator requests, Online or Offline.
- ▶ **Operational State**
Detailed automation processing status, such as Online, Offline, Pending Online, Failed Offline, and so on.
- ▶ **Compound State**
Used to select the icon that appears on the Operations Console for domains, groups, and resources. Possible values: OK, Warning, Error, and Fatal.

Relationships

These define the relationships between the resources defined in a cluster. Relationships provide the possibility to define relationships between resource groups, resources, and equivalencies. It is also possible to define relationships between resources running on different systems in the cluster.

IBM Tivoli System Automation for Multiplatforms allows the definition of the following relationships:

- Start-stop relationships

Relationships are used to define start and stop dependencies between resources. The following are allowed start-stop relationships:

StartAfter, StopAfter, DependsOn, DependsOnAny, and ForcedDownBy

- Location relationships

Location relationships are applied when resources must, or should if possible, be started on the same or a different node in the cluster. The following are allowed location relationships:

Collocation, AntiCollocation, Affinity, AntiAffinity, and IsStartable.

A simple example is that a Web server and its corresponding service IP address, which could be started on any node in the cluster, should always be kept together.

Equivalency

An equivalency is a collection of resources that provide the same functionality. This allows IBM Tivoli System Automation for Multiplatforms to select any resource in the equivalency to perform an operation. For example, a common practice is to define a set of network adapters in an equivalency, so that in case of a failure of a network adapter, IBM Tivoli System Automation for Multiplatforms is able to define an IP address to another network adapter in the equivalency.

Note: There is an implicit Collocation location relationship between multiple floating resources that have a DependsOn relationship defined between them and the same equivalency. In case this functionality is not desired, it is possible to define multiple identical equivalencies each differing only by the name. Now each resource would instead have a DependsOn relation from that resource to one of the newly created equivalencies.

Later in this IBM Redbook, we present a case study scenario in which several relationships are defined. Also, refer to *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04, for additional information on relationships.

IBM Tivoli System Automation for z/OS V3.1

This chapter provides an overview of IBM Tivoli System Automation for z/OS V3.1 functionality and new features. We discuss the following topics:

- ▶ “IBM Tivoli System Automation for z/OS V3.1 overview” on page 34
- ▶ “What is new in IBM Tivoli System Automation for z/OS V3.1” on page 37
- ▶ “Overview Planning for installation” on page 44

2.1 IBM Tivoli System Automation for z/OS V3.1 overview

IBM Tivoli System Automation for z/OS is an IBM Tivoli NetView for z/OS base software product that provides a single point of control for a various range of systems management functionality. IBM Tivoli System Automation for z/OS plays a key role in supplying high end-to-end automation solutions.

IBM delivers integrated cross-platform management functions. IBM Tivoli System Automation for z/OS functions include monitoring, control, and automation of a large range of system elements spanning both the hardware and software resources of your enterprise.

IBM Tivoli System Automation for z/OS is a system management program with a single point of control. You see a single system image for a full range of essential systems management functions. IBM Tivoli System Automation for z/OS monitors, controls, and automates the following:

- ▶ Monitors the following resources to respond before they affect users:
 - Hardware components
 - Software products and applications
 - Automated processes
 - Messages and alerts
- ▶ Controls and takes action on resources based on conditions:
 - Start and stop your entire enterprise system; initiate hardware and software startup and shutdown sequences
 - Manage both remote and local operations and support any zSeries and 390-CMOS processors within a Parallel Sysplex
 - Manage several operating systems: z/OS, OS/390®, MVS™, VM, and VSE
 - Control a coupling facility as a target system with coupling links in a Parallel Sysplex environment
 - React to errors and unscheduled events
- ▶ Automates many repetitive and complex tasks:
 - Start/shutdown software resources
 - Control channels and channel paths
 - Control availability of I/O devices
 - Control switching of I/O device ports

- Detect and respond to system messages
- Perform initial program load (IPL)
- Perform system power-on reset (POR)
- Build an automation policy for your enterprise
- Extend the built-in automation routines by writing your own automation policies

IBM Tivoli System Automation for z/OS is a IBM Tivoli NetView for z/OS-based application. Its functions include monitoring, controlling, and automating a large range of system elements, spanning both hardware and software resources of an enterprise (see Figure 2-1).

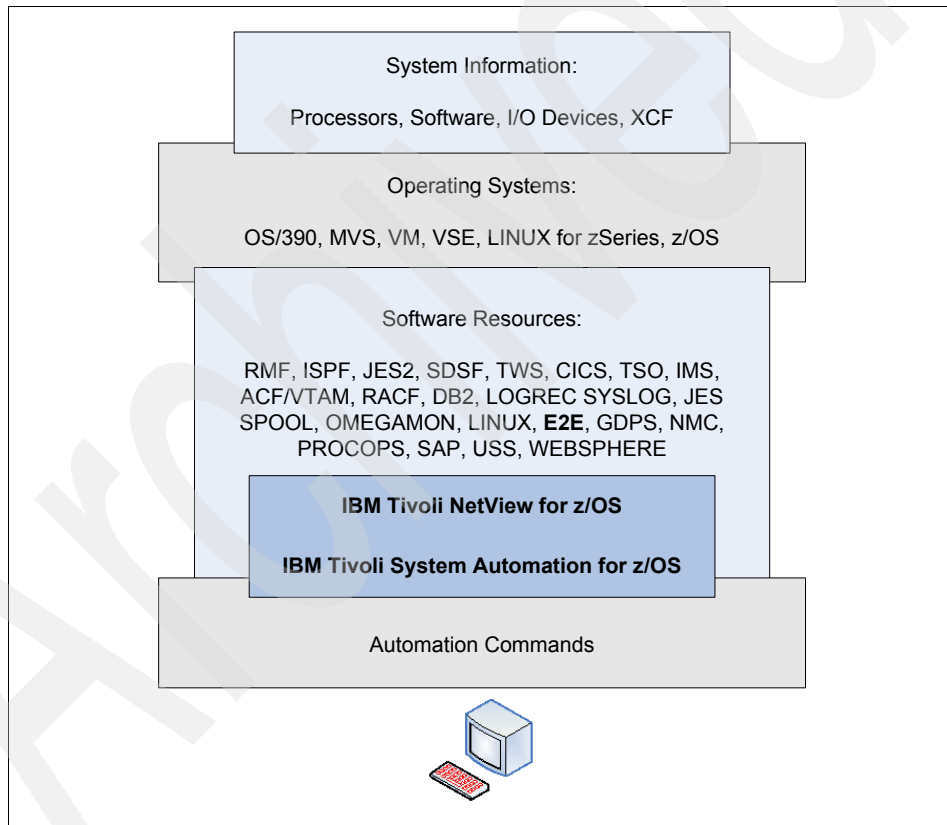


Figure 2-1 Monitor, control, and automation functions

IBM Tivoli System Automation for z/OS perform the following major operations:

- ▶ System Operations
- ▶ Processor Operations
- ▶ I/O Operations

System Operations

System Operations (SysOps) monitors and controls system operations applications and subsystems such as IBM Tivoli NetView for z/OS, SDSF, JES, RMF™, TSO, RODM, ACF/VTAM®, DB2®, CICS®, IMS™, OMEGAMON®, and Tivoli Workload Scheduler. With system operations, you can automate Parallel Sysplex applications. IBM Tivoli System Automation for z/OS can automate applications distributed over a sysplex by virtually removing system boundaries for automation through its automation manager/automation agent design. IBM Tivoli System Automation for z/OS reduces the complexity of managing a Parallel Sysplex through its goal driven automation and its concepts such as grouping and powerful dependency support which enable you to model your configuration. Single systems are also fully supported; the automation scope is then just one system. IBM Tivoli System Automation for z/OS uses Enterprise monitoring to update the NetView Management Console (NMC) resource status information which is stored in the Resource Object Data Manager (RODM). IBM Tivoli System Automation for z/OS provides controlled startup, controlled shutdown, and automated recovery of various software resources on your systems. System Operations (SysOps) provides the ability to monitor and control all subsystems in a sysplex from any system in the sysplex. In other words, if your Focal Point (FP) is down for whatever reason, you can view the status from any other system within a SysPlex.

Processor Operations

Processor operations monitor and control processor hardware operations. It provides a connection from a focal point processor to a target processor. With IBM Tivoli NetView for z/OS on the focal point system, processor operations automate operator and system consoles for monitoring and recovering target processors.

The Processor Operations facility (ProcOps), formerly known as Target System Control Facility (TSCF), provides connections from a Focal Point processor to target processors. In addition, ProcOps allow you to power multiple target processors on, off, and reset systems; perform IPLs; set time of day clocks; respond to messages; monitor status; and detect and recover wait states.

I/O Operations

I/O operations provide a single point of control for managing connectivity in your active I/O configurations. It takes an active role in detecting unusual I/O

conditions and lets you view and change paths between a processor and an input/output device, which can involve using dynamic switching, such as the enterprise systems connection (ESCON®) or fiber channel connection (FICON®) switch. I/O operations changes paths by letting you control channels, ports, switches, control units, and input/output devices. You can do this through an operator console or API.

2.2 What is new in IBM Tivoli System Automation for z/OS V3.1

This section contains an overview of important changes and enhancements to IBM Tivoli System Automation for z/OS V3.1. There are a number of enhancements with this release of IBM Tivoli System Automation for z/OS. For a complete list, refer to the *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual.

Here we present a summary, as follows:

- ▶ Enhancements to the customization dialog
- ▶ IBM Tivoli OMEGAMON integration
- ▶ GDPS® integration
- ▶ IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter

2.2.1 Enhancements to the Customization Dialog

The section introduces the most important changes in the behavior of the customization dialogs, such as:

- ▶ Adding, updating, and deleting large amounts of data
- ▶ Concurrent multi-user access to system definitions
- ▶ Policy database import
- ▶ Enhancements to the entry name selection panel
- ▶ Additional sample policy databases

Adding, updating, and deleting large amounts of data

There is an alternative to entering or modifying policy data with the customization dialog. You can create sequential data sets with specifications of the policy data in a newly designed text format. These sequential data sets allow one to see multiple policy items of multiple database entries at a glance, but in contrast with data base reports. The syntax of these imports cannot be violated but only be imported into the policy data base.

There are two keywords: new and update, in the sequential data set control whether the important will add entries to the policy database. Thereafter, you can modify the entries. The data sets for import into policy databases can be created either “from scratch” or by exporting selected parts of an existing policy data base.

In addition to deleting single policy objects one by one, you can now delete any number of policy objects all at once. With this new feature, you can specify how often you want to be asked for confirmation.

Your options are:

- ▶ Confirmation with each policy object to be deleted
- ▶ Confirmation only applies to those policy objects that are still connected to other policy objects (via links, membership, class instantiation, and so on)
- ▶ No confirmation at all

If you confirm deletion of an object that is still connected, any links to it will be removed, too.

Concurrent multi-user access to system definitions

Concurrent write-access by multiple users is now possible for policy objects.

Policy database import

IBM System Automation z/OS 2.3 allows you to import one or multiple policy objects of a single entry from another policy data base into the current one, but the link was not included with the import. This restriction has now been removed in IBM Tivoli System Automation for z/OS V3.1.

The Policy Data Base importation supports the inclusion of the LINKED objects for the following entry types:

- ▶ APG (application group imported together with APLs, SVPs, and TRGs)
- ▶ APL (application instance imported together with the class APL, and class APL imported together with its APL instances, this together with SVPs, TRGs, CSAs, and ISA that is linked to them)
- ▶ TRG (trigger imported along with EVT)

This enables one to import the entire object in a single import. For instance, the entire application group and everything that is associated with this group will be copied to the target policy database.

Enhancements to entry name selection panel

On the Policy Data Base, there is a new QUERYSTAT command that shows Statistics Information about the object. The following information is displayed when a Q (QUERYSTAT) is entered next to the object.

- ▶ The user id that made the last update
- ▶ The date and time of the last update
- ▶ The date and time of the last build
- ▶ The ACF fragment name (if built into the ACF)

Additional sample policy databases

The Policy Data Base samples that are shipped with IBM Tivoli System Automation for z/OS V3.1 have been completely restructured and rewritten.

There is now a distinction between:

- ▶ Basic samples
- ▶ Add-on samples

There are only two basic samples:

- ▶ BASE
- ▶ EMPTY

The new add-on sample policies shipped with IBM Tivoli System Automation for z/OS V3.1 are:

- ▶ IBM CICS
- ▶ IBM DB2
- ▶ IBM End-to-end Automation Adapter
- ▶ IBM Geographically Dispersed Parallel Sysplex™ (GDPS)
- ▶ IBM Information Management Systems (IMS)
- ▶ IBM NetView Management Console (NMC)
- ▶ IBM Tivoli OMEGAMON
- ▶ IBM Processor Operations (PROCOPS)
- ▶ SAP
- ▶ IBM Tivoli Work Scheduler (TWS)
- ▶ IBM UNIX® System Services (USS)
- ▶ IBM WebSphere Application Server

2.2.2 IBM Tivoli OMEGAMON integration

IBM Tivoli System Automation for z/OS V3.1 now integrates with the IBM Tivoli OMEGAMON suite by means of automated sessions. This complements message-based automation with new proactive monitoring capabilities. For details refer to the *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual.

2.2.3 GDPS Integration

To set up IBM Tivoli System Automation for z/OS V3.1 and IBM Tivoli NetView for z/OS for GDPS has been made significantly easier because IBM now provides a predefined GDPS environment as part of IBM Tivoli System Automation for z/OS V3.1. For details, refer to the *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual.

2.2.4 IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter

The IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component provides end-to-end automation. The IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component can be used to automate the operation of resources within heterogeneous environments (called *first-level automation domains*) that each has a local automation technology of its own.

Each first-level automation domain is connected to the End-to-end Automation Manager by an End-to-end Automation Adapter. The End-to-end Automation Manager automation domain is named *end-to-end automation domain*.

The IBM Tivoli System Automation for Multiplatforms V2.1 operations console is a Web-based graphical user front-end to the end-to-end automation domain and to the first-level automation domains. This means resources managed by IBM Tivoli System Automation for z/OS V3.1 or IBM Tivoli System Automation for Multiplatforms V2.1, or other automation tools can be managed by the IBM Tivoli System Automation for Multiplatforms End-to-end Automation Management Component using the IBM Tivoli System Automation for Multiplatforms V2.1 operations console.

The End-to-end Automation Manager has a role equivalent to the Automation Manager in IBM Tivoli System Automation for z/OS V3.1. For further details, refer to the *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271, manual.

Each first-level automation domain managed by IBM Tivoli System Automation for z/OS V3.1 has an End-to-end Automation Adapter and a Primary Automation Agent that together communicate with the End-to-end Automation Manager. The End-to-end Automation Adapter and Primary Automation Agent must run on the same host system and must be linked to the sysplex group. The End-to-end Automation Adapter (E2E Automation Adapter) can run on only one system in the sysplex (See Figure 2-2).

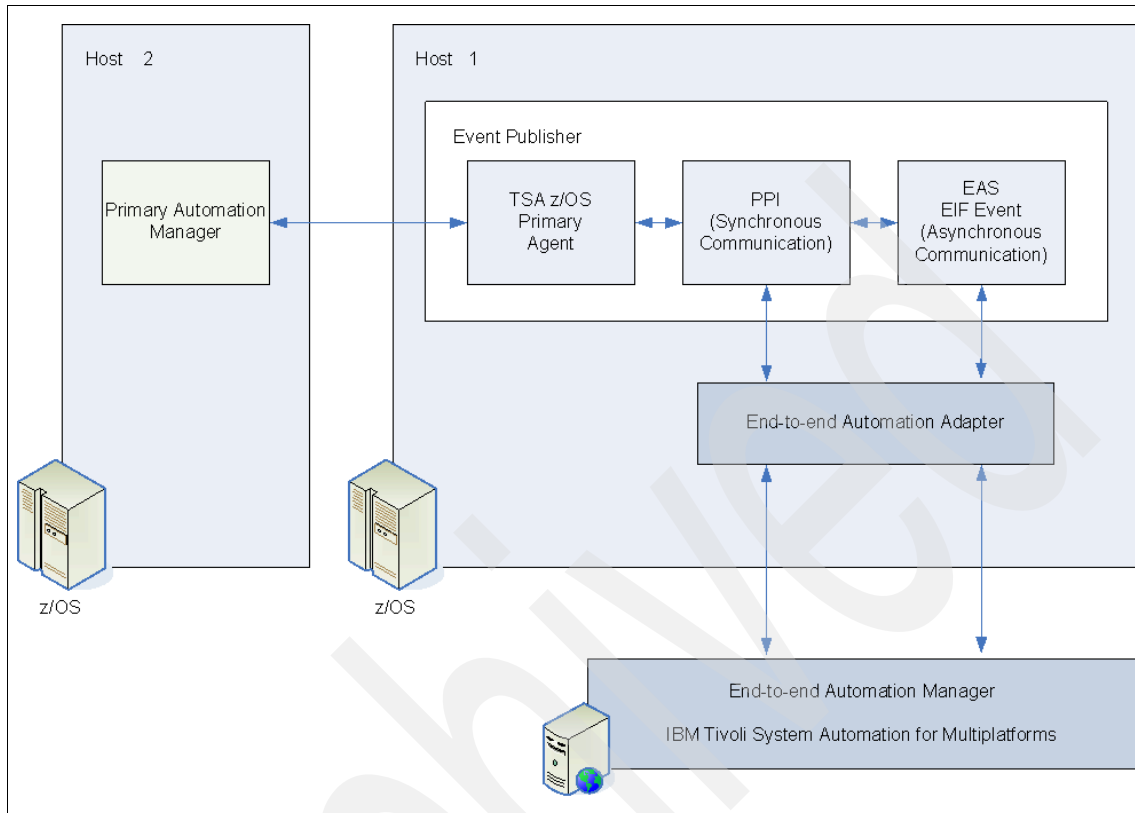


Figure 2-2 E2E Automation Adapter communication

Shown by Figure 2-2, the IBM Tivoli System Automation for z/OS V3.1 Primary Automation Manager (on Host 2) can run on any system within the sysplex, other than the system on which the End-to-end Automation Adapter is running (on Host 1).

The End-to-end Automation Adapter acts as the link between the End-to-end Automation Manager and its first-level automation domain within managed by IBM Tivoli System Automation for z/OS in the sysplex group.

Note: The End-to-end Automation Adapter must be located on the same system as the primary IBM Tivoli System Automation for z/OS V3.1 agent (Host 1 on Figure 2-2).

The functions of the End-to-end Automation Adapter are:

- ▶ Monitor resources within its first-level automation domain
- ▶ Propagate resource attribute changes to the End-to-end Automation Manager
- ▶ Start and stop resources within the first-level automation domain by request of the End-to-end Automation Manager
- ▶ Provide information about resources that are available within the first-level automation domain in response to queries from operators

The End-to-end Automation Adapter can communicate either synchronously or asynchronously. Figure 2-2 represents this function.

▶ Synchronous Communication

The End-to-end Automation Adapter schedules an IBM Tivoli System Automation for z/OS V3.1 task execution request via Program-to-Program Interface synchronous communication (see Figure 2-2) to a task execution command handler that runs the automated operator function E2EOPER or E2EOPRnn. The task execution request contains one or more end-to-end automation requests.

The Program-to-Program Interface (PPI) enables application programs to send or receive data buffers from other application programs that are running on the same host. It is an optional facility of the IBM Tivoli NetView for z/OS subsystem address space and can be initialized with its PPIOPT start option. The PPI option can only be requested for one subsystem address space of a system. PPI is used for synchronous communication between the primary IBM Tivoli System Automation for z/OS V3.1 automation agent and the automation adapter.

▶ Asynchronous Communication

The End-to-end Automation Adapter provides an Event Integration Facility (EIF) event receiver and an EIF event emitter. IBM Tivoli System Automation for z/OS V3.1 acts as an asynchronous data provider and sends specific events to the End-to-end Automation Adapter's EIF event receiver. This then delegates the events to the End-to-end Automation Adapter's event mapping function. Once an event has been mapped and not rejected by the End-to-end Automation Adapter, it is sent to the End-to-end Automation Manager via the EIF emitter component.

The message adapter service of the Event Automation Service (EAS) converts and forwards messages from the IBM Tivoli NetView for z/OS message automation to a designated event server, such as the End-to-end Automation Adapter. So that the End-to-end Automation Adapter can receive events from IBM Tivoli System Automation for z/OS V3.1, EAS registers with the IBM Tivoli NetView for z/OS PPI. IBM Tivoli System Automation for z/OS V3.1 sends automation adapter-specific events over PPI to the message

adapter service. The EAS message adapter service converts the messages into EIF events using the message adapter format file. The resulting events are forwarded to the End-to-end Automation Adapter.

Prerequisites and dependencies

To implement the End-to-end Automation Adapter, the following prerequisites and dependencies apply:

- ▶ IBM Tivoli System Automation for z/OS V3.1
- ▶ IBM Tivoli NetView for z/OS V5.1
- ▶ z/OS 1.3 or higher
- ▶ Java™ Runtime Environment (JRE) 1.4.2 installed on z/OS
- ▶ The JRE Software Development Kit (SDK) if you choose to use the facility to create sample keys. Refer to the *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-00, manual for details.

You need to configure the following components of these prerequisites:

- ▶ Event/Automation Service (EAS) component of IBM Tivoli NetView for z/OS.
- ▶ Full z/OS UNIX System Services (USS) with hierarchical file system
- ▶ Subsystem Interface (SSI) address space with PPI function of IBM Tivoli NetView for z/OS
- ▶ IBM Tivoli System Automation for Multiplatforms V2.1 Automation Manager
- ▶ IBM Tivoli System Automation for z/OS V3.1 Automation Agent

Primary Automation Agent and Primary Automation Manager

The successful initialization of the End-to-end Automation Adapter on a system makes the automation agent on that system the *primary automation agent*.

The End-to-end Automation Adapter and the primary agent must run on the same system (see Figure 2-2).

The primary automation agent communicates the fact that it has become the primary automation agent on that system to a *primary automation manager* (PAM). The primary automation manager ensures that there must exist only one primary automation agent in the sysplex.

The primary automation manager can run on any system in the sysplex other than the system on which the End-to-end Automation Adapter is running (see Figure 2-2).

The End-to-end Automation Adapter acts as the link between the IBM Tivoli System Automation for Multiplatforms End-to-end Automation Manager and its first-level automation domain within the z/OS sysplex.

The primary automation agent forwards events to the End-to-end Automation Adapter after it has registered with the primary automation agent and subscribed for them. After a system failure, event subscriptions may have been lost and so the End-to-end Automation Adapter has to re-subscribe. The primary automation manager sends events to the primary automation agent over Crosslink Coupling Facility (XCF).

Note that the End-to-end Automation Adapter can only run on one system in a domain, although we may have more than one domain in a sysplex. There can, however, be a maximum of only one End-to-end Automation Adapter per domain.

2.3 Overview Planning for installation

These are the major tasks required to install IBM Tivoli System Automation for z/OS V3.1:

- ▶ SMPE/E installation
- ▶ Allocate System Unique Data Sets
- ▶ Allocate Data Sets for the Customization Dialog
- ▶ Customization of SYS1.PROCLIB
- ▶ Customization of SYS1.PARMLIB
- ▶ Customize IBM Tivoli NetView for z/OS
- ▶ Customization of the IBM Tivoli System Automation for z/OS Automation Manager
- ▶ Installation of ISPF Dialog Panels
- ▶ Defining Automation Policy
- ▶ Automate System Operations Startup

Refer to the *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual for details about each of these steps.

We present details for this implementation in our case study scenario chapter, Chapter 6, “Case study scenario: IBM DB2 on z/OS first-level automation domain” on page 129.



Part 2

Case study scenario implementation

Case study scenario overview

This chapter outlines a case study environment, which we intend to use to represent a simplified customer environment.

We describe an environment, along with the rules to automate it, and we use this environment throughout this IBM Redbook to demonstrate the activities required for a typical high availability and automation solution using both IBM Tivoli System Automation for Multiplatforms V2.1 and IBM Tivoli System Automation for z/OS V3.1.

3.1 Scenario overview

In many cases, enterprise software applications are multi-tiered with their components running on multiple nodes, and, often, on multiple operating system platforms. In this chapter, we show a scenario using a three-tiered application environment running a Java 2 Platform Enterprise Edition (J2EE) application.

The application environment in this scenario uses Web servers (Linux) providing front end and Web access services, Web application servers (AIX) providing the J2EE environment for running the application and transaction services, and back-end database servers (z/OS). The J2EE application accesses the database through Java Database Connectivity (JDBC™).

Each of the application tiers we describe consists of multiple servers. Figure 3-1 shows the application environment.

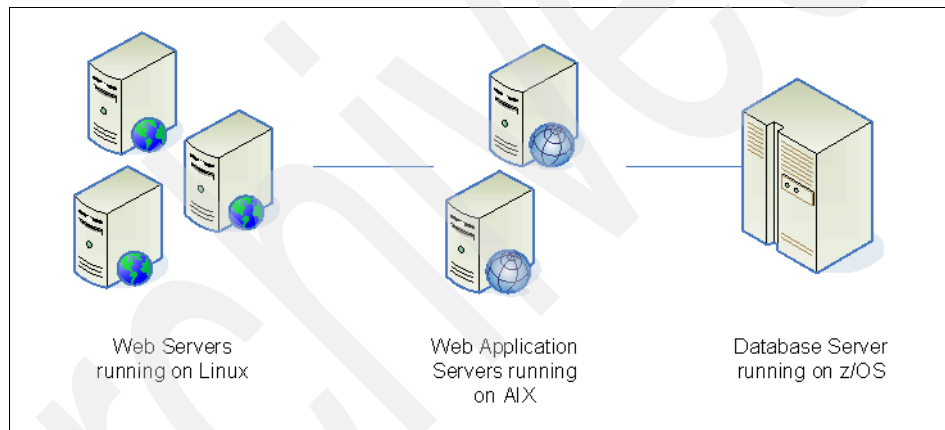


Figure 3-1 Application environment

The first two tiers of the application environment in this scenario consist of a homogeneous cluster. A single z/OS system represents the third tier. The following are the characteristics of each tier presented in Figure 3-1:

- ▶ Tier one has three servers running Apache HTTP Server on Red Hat RHEL 3.0 ES.
- ▶ Tier two has two servers running IBM WebSphere Application Server on IBM AIX 5.3.
- ▶ Tier three runs IBM DB2 Server V8.1.5 on z/OS 1.06.

This chapter focuses on the automation aspects of this heterogeneous environment using IBM Tivoli System Automation for Multiplatforms V2.1. We create a first-level automation management for each homogeneous cluster using

the IBM Tivoli System Automation for Multiplatforms V2.1 Base Component and IBM Tivoli System Automation for z/OS V3.1. In addition, we create an End-to-end Automation Management environment for our application by building automation controls upon each first-level automation domain and integrating them. Figure 3-2 presents this environment.

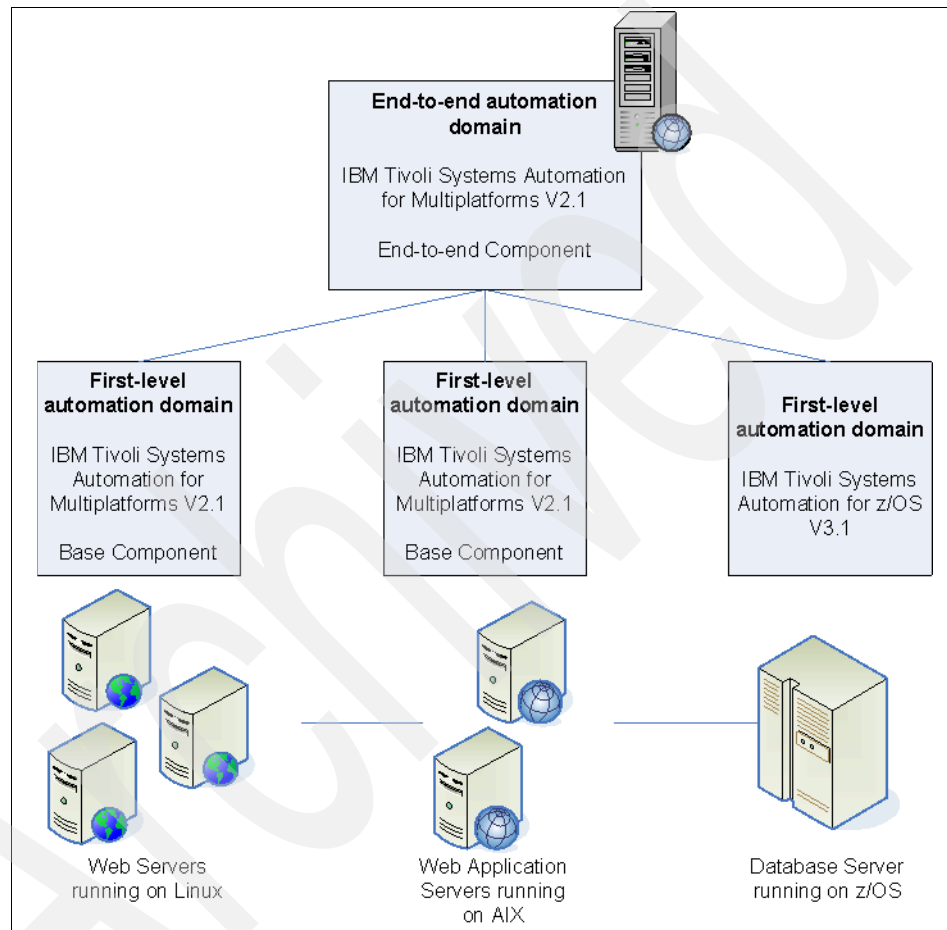


Figure 3-2 End-to-end automation case study scenario

We demonstrate how to create relationships, dependencies, and automation policies between the resources belonging to each homogeneous cluster. This configuration ensures high availability of resources local to each cluster and makes up each first-level automation domain. We show the configuration steps required to prepare the first-level automation domains for end-to-end automation. This implies configuring an automation adapter in each of the first-level automation domains, as we see in Figure 3-3.

The following figure shows the communication between components and details of the automation domains in our case study scenario.

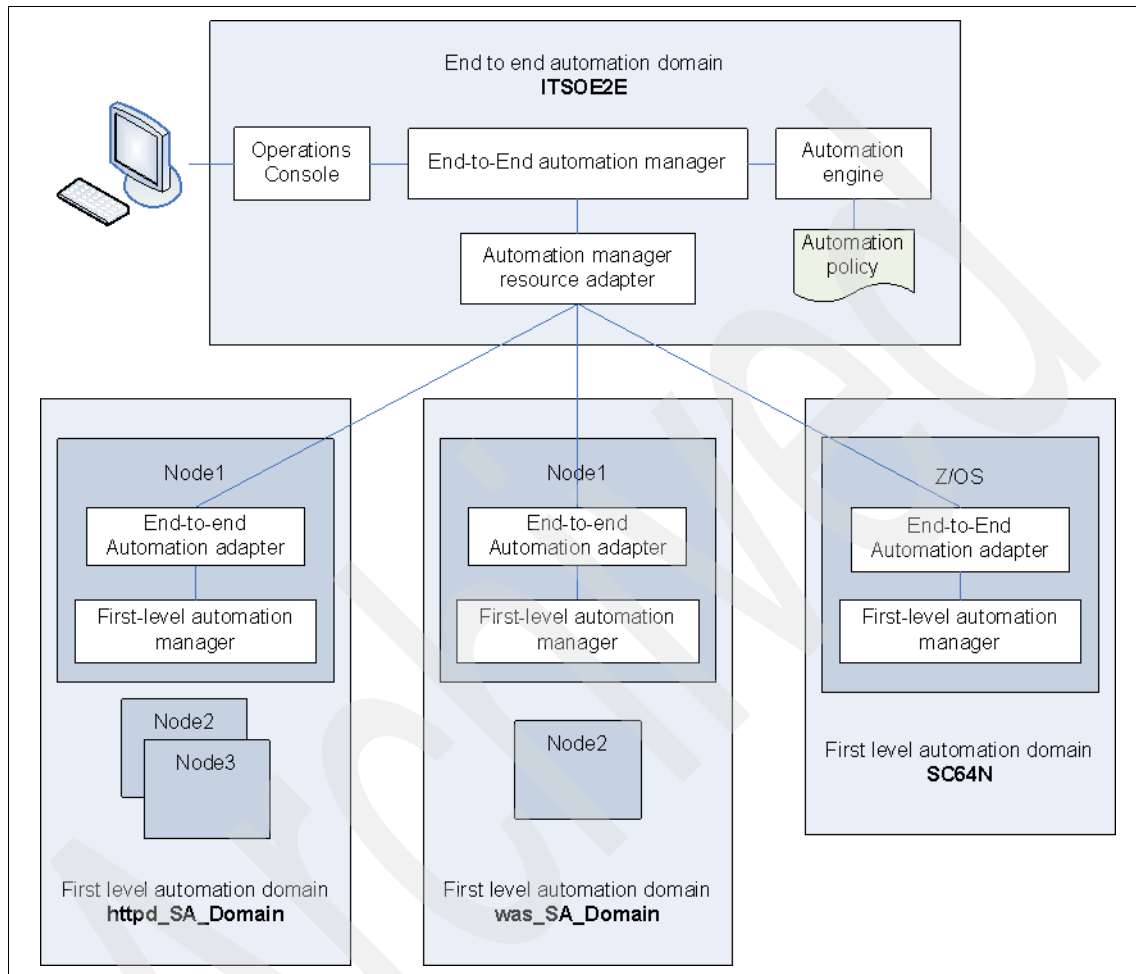


Figure 3-3 IBM Tivoli System Automation for Multiplatforms scenario components

We discuss this in the following chapters:

- Chapter 4, “Case study scenario: HTTP Servers on Linux first-level automation domain” on page 53.

This chapter describes the activities for achieving high availability and automation of the scenario HTTP servers cluster by setting up the **httpd_SA_Domain** first-level automation domain using IBM Tivoli System Automation for Multiplatforms V2.1 Base Component running on Red Hat Linux.

- ▶ Chapter 5, “Case study scenario: Application Servers on AIX first-level automation domain” on page 89.

This chapter describes the activities for achieving high availability and automation of the scenario IBM WebSphere Application Server cluster by setting up the **was_SA_Domain** first-level automation domain using IBM Tivoli System Automation for Multiplatforms V2.1 Base Component running on IBM AIX.

- ▶ Chapter 6, “Case study scenario: IBM DB2 on z/OS first-level automation domain” on page 129.

This chapter describes the activities for achieving high availability and automation of the scenario database server by setting up the **SC64N** first-level automation domain using IBM Tivoli System Automation for z/OS V3.1.

- ▶ Chapter 7, “Case study scenario: End-to-end automation domain” on page 201.

We complete the scenario by creating a second-level automation domain, specifically an end-to-end automation management domain named **ITSOE2E**.

This domain controls the relationships between the resources of every first-level automation domain to ensure high availability of our application environment. In order to accomplish this, we install and configure all elements of the IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component on a single server. The ITSOE2E end-to-end automation management domain uses its end-to-end automation manager and automation manager resource adapter to communicate to the end-to-end automation adapters running on the first-level automation domains.

Case study scenario: HTTP Servers on Linux first-level automation domain

In this chapter, we discuss the high availability and automation of the servers performing HTTP Server services for our sample application environment. We do this by creating and configuring an IBM Tivoli System Automation for Multiplatforms first-level automation domain named `apache_SA_Domain`.

In addition, we configure an End-to-end Automation Adapter so that this first-level automation domain can be managed by the IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component.

We discuss the following topics:

- ▶ “Apache automation domain overview” on page 55
- ▶ “Automation domain configuration” on page 56
- ▶ “End-to-end Automation Adapter configuration” on page 75
- ▶ “Miscellaneous information” on page 83

Figure 4-1 shows the portion of the entire case study scenario we cover in this chapter.

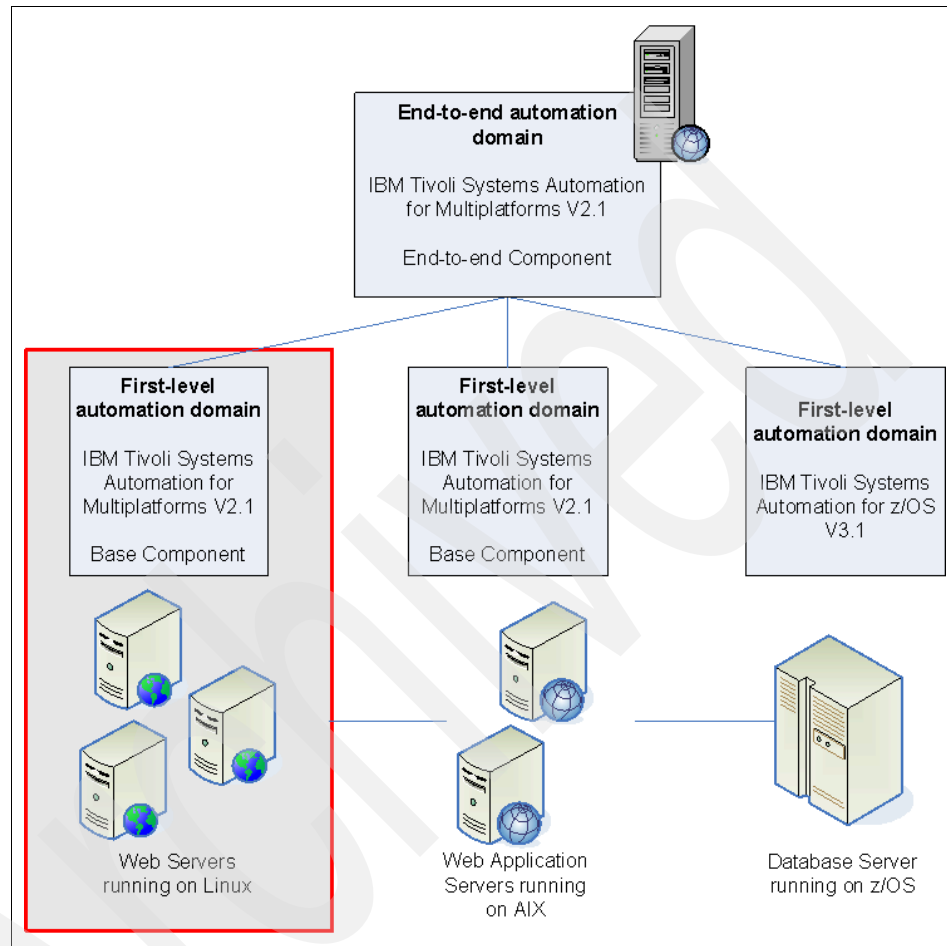


Figure 4-1 Apache first-level automation domain

4.1 Apache automation domain overview

This section describes the configuration steps for setting up high availability for the servers providing HTTP services for our sample application. Our HTTP server environment consists of three Linux systems running on xSeries® servers. All three servers in the cluster have identical hardware and software configurations.

This three node cluster provides the HTTP environment and is configured to service any Web request to the address `tsahttpd.itsc.austin.ibm.com` address and redirect the request to the IBM WebSphere Application Server cluster at the `tsawas.itsc.austin.ibm.com` address. We modified the `/etc/httpd/conf/httpd.conf` file to achieve this redirection.

This environment is meant to have one active instance of `httpd` running on any of the three available servers using a single IP Address (ServiceIP - 9.3.5.29) mapped to the `tsahttpd.itsc.austin.ibm.com` address.

Figure 4-2 shows the configuration of one of the servers in the cluster and all elements that will be part of the IBM Tivoli System Automation for Multiplatforms first-level automation domain configuration.

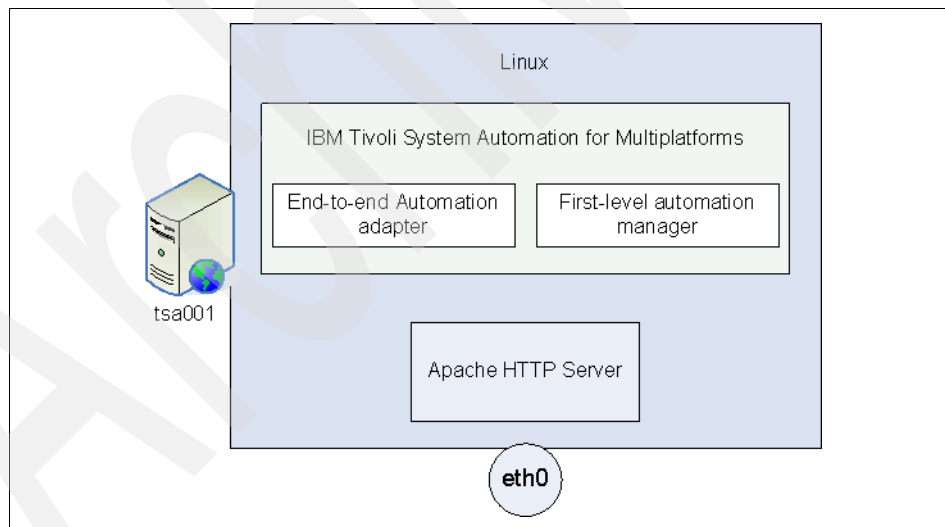


Figure 4-2 Web application tier configuration

IBM Tivoli System Automation for Multiplatforms is configured to ensure that only one instance of Apache is active on any of the three servers and that server is configured to use the ServiceIP.

4.1.1 Installation

We install IBM Tivoli System Automation for Multiplatforms V2.1 on the three servers of our Apache HTTP cluster: tsa001, tsa002, and tsa003. Installation of IBM Tivoli System Automation for Multiplatforms is straightforward and uneventful. See the *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04, for further information.

4.1.2 Automation requirements

We define the following high availability and automation requirements for our environment:

- ▶ The Apache HTTP Server must be startable on any node in the domain.
- ▶ A single instance of the Apache HTTP Server is active on only *one* node in the domain *at any given point in time*.
- ▶ In case of a failure, the Apache HTTP Server must be restarted on any node in the domain.
- ▶ We have installed the Apache HTTP Server on all nodes in our domain.
- ▶ The running instance of the HTTP server must provide the same URL to the user and this URL must be associated to the same IP address, regardless of the node on which the HTTP server is running. This IP address serves as the Service IP for the domain.

The following sections go into detail about how we meet these requirements in our scenario.

4.2 Automation domain configuration

This section contains the steps we use to create the IBM Tivoli System Automation for Multiplatforms environment to manage our Apache HTTP server cluster.

We perform the following steps:

- ▶ Create the automation domain
- ▶ Define resources within the automation domain
- ▶ Create the automation policy using relationship definitions
- ▶ Verify the automation domain is operational

4.2.1 Create the first-level automation domain

The first step in the creation of our automation domain is to prepare the target servers for domain membership. The node preparation task enables security on the node, so we can define this node in an automation domain. It allows for automation operations to be performed on the node. This task must be done before the node can join an automation domain. Then, we can define the automation domain itself, specifying the nodes that will be members of the domain.

Prepare the nodes

We do this using the IBM Reliable Scalable Cluster Technology (RSCT) **preprnode** command.

We must run the **preprnode** command on *every* node that will be part of the automation domain. In our environment, we issue the following command on *each* of the three servers, tsa001, tsa002 and tsa003:

```
# preprnode tsa001 tsa002 tsa003
```

Note: To remove a single point of failure, create entries for each node of the cluster in their local `/etc/hosts` files on all nodes and ensure the nameserver entries are identical.

Create the automation domain

The next step is to create the automation domain using the RSCT command **mkrpdomain**.

For the purposes of this case study scenario, the IBM Tivoli System Automation for Multiplatforms configuration contains a single automation domain named **ttpd_SA_Domain**, of which the three Apache HTTP servers, tsa001, tsa002, and tsa003, are members. We create our domain with the command:

```
# mkrpdomain httpd_SA_Domain tsa001 tsa002 tsa003
```

This command *must* be executed only once on any of the target nodes of the automation domain. From this point on, any IBM Tivoli System Automation for Multiplatforms or RSCT command may be issued from any node in the automation domain.

Use the command **lsrpdomain** to display the IBM Tivoli System Automation for Multiplatforms automation domain status. See Example 4-1.

Example 4-1 Display automation domain

```
# lsrpdomain
Name           OpState RSCTActiveVersion MixedVersions TSPort GSPort
httpd_SA_Domain Offline 2.4.3.0           No           12347 12348
```

The newly created automation domain has an operational state (OpState) of Offline. We change the OpState by issuing a **startlrpdomain** command to bring the new defined automation domain online.

```
# startlrpdomain httpd_SA_Domain
```

At this point, you can verify the automation domain status with the **lsrpdomain** command. See Example 4-2.

Example 4-2 Automation domain status

```
# lsrpdomain
Name           OpState RSCTActiveVersion MixedVersions TSPort GSPort
httpd_SA_Domain Online  2.4.3.0           No           12347 12348
```

4.2.2 Define resources in the automation domain

Now that we have defined the extent of our Apache automation domain (nodes: tsa001, tsa002, and tsa003), and it is in the online operational state, it is time to define the resources within this automation domain we wish to control. We need to define the application (Apache), the Service IP Address (9.3.5.29), and to indicate to IBM Tivoli System Automation for Multiplatforms that the network adapters on nodes tsa001, tsa002, and tsa003 are appropriate and equivalent for the Service IP Address.

Resources have two type of attributes, Persistent and Dynamic. The resource definitions specify the persistent attributes. Dynamic attributes are set by RSCT and IBM Tivoli System Automation for Multiplatforms.

Define ServiceIP resource

In our scenario, we have an IP Address that we want to be active on the same server as our active HTTP Server. This is an example of a resource that we must define to IBM Tivoli System Automation for Multiplatforms. We create RSCT resources with the command **mkrsrc**. We can provide all resource characteristics in command line parameters, but the **mkrsrc** command also accepts a definition file in plain text. We use the second approach with a definition file named **apache.resourcedef.IBM.ServiceIP**, which contains the following (see Example 4-3:

Example 4-3 ServiceIP definition input file

```
PersistentResourceAttributes::  
  NodeNameList={"tsa001","tsa002","tsa003"}  
  Name="apache_SIP"  
  NetMask=255.255.255.0  
  IPAddress=9.3.5.29  
  ResourceType=1
```

Note: The parameter `ResourceType=1` indicates a *floating* resource, in other words, not fixed. The default is 1, floating resource, no matter how many entries are supplied in the `NodeNameList` parameter.

The resource representing the ServiceIP for our HTTP servers is of class `IBM.ServiceIP`. The command we use to create our ServiceIP resource using the above definition file as input is:

```
# mkrsrc -f apache.resourcedef.IBM.ServiceIP IBM.ServiceIP
```

We verify the resource creation by issuing a `lsrsrc` command. See Example 4-4.

Example 4-4 Display the apache_SIP ServiceIP

```
# lsrsrc -l IBM.ServiceIP  
Resource Persistent Attributes for IBM.ServiceIP  
resource 1:  
  Name           = "apache_SIP"  
  ResourceType   = 0  
  AggregateResource = "0x2029 0xffff 0x421b1763 0xf3c328b7 0x8f9a0344  
0xd236b078"  
  IPAddress      = "9.3.5.29"  
  NetMask        = "255.255.255.0"  
  ProtectionMode = 1  
  ActivePeerDomain = "httpd_SA_Domain"  
  NodeNameList   = {"tsa003.itsc.austin.ibm.com"}  
resource 2:  
  Name           = "apache_SIP"  
  ResourceType   = 0  
  AggregateResource = "0x2029 0xffff 0x421b1763 0xf3c328b7 0x8f9a0344  
0xd236b078"  
  IPAddress      = "9.3.5.29"  
  NetMask        = "255.255.255.0"  
  ProtectionMode = 1  
  ActivePeerDomain = "httpd_SA_Domain"  
  NodeNameList   = {"tsa002.itsc.austin.ibm.com"}  
resource 3:  
  Name           = "apache_SIP"  
  ResourceType   = 0
```

```

AggregateResource = "0x2029 0xffff 0x421b1763 0xf3c328b7 0x8f9a0344
0xd236b078"
IPAddress          = "9.3.5.29"
NetMask            = "255.255.255.0"
ProtectionMode     = 1
ActivePeerDomain   = "httpd_SA_Domain"
NodeNameList       = {"tsa001.itsc.austin.ibm.com"}

resource 4:
Name               = "apache_SIP"
ResourceType       = 1
AggregateResource  = "0x3fff 0xffff 0x00000000 0x00000000 0x00000000
0x00000000"
IPAddress          = "9.3.5.29"
NetMask            = "255.255.255.0"
ProtectionMode     = 1
ActivePeerDomain   = "httpd_SA_Domain"
NodeNameList       =
{"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.austin.
ibm.com"}

```

Define equivalency for the network adapters

Our three xSeries servers in the automation domain have multiple network adapters (eth0 and eth1). When a node in the cluster has multiple network attachments, they all may not be equally suited to host the ServiceIP resource (apache_SIP). An equivalency definition specifies the network adapters that you can use to carry the apache_SIP address. *Equivalency* means that each of the adapters in the equivalency can provide the same required function regardless of its own unique characteristics. Since the HTTP Server should be startable on each node in the cluster, at least one of the adapters on each node has to appear in the equivalency. An equivalency groups together a set of resources from another class. Network adapters belong to a class named IBM.NetworkInterface. There is no need to provide resource definitions for all the network adapters on the cluster nodes, since RSCT has a harvesting function which automatically creates appropriate resource definitions for many system-defined resources.

The following command creates a static equivalency named apache_nieq, which contains a network adapter from each node of the cluster:

```
# mkequ apache_nieq IBM.NetworkInterface:eth0:tsa001,eth0:tsa002,eth0:tsa003
```

Important: We recommend to define equivalencies in a dynamic fashion, especially when interfacing the first-level domain with the IBM Tivoli System Automation for Multiplatforms Operations Console or the End-to-end Automation Management Component. Chapter 5, “Case study scenario: Application Servers on AIX first-level automation domain” on page 89 provides step by step instructions about how to define a dynamic equivalency.

You can verify the equivalency with the command **lsequ**. See Example 4-5.

Example 4-5 Display equivalency apache_nieq

```
# lsequ -e apache_nieq
Displaying Equivalency information:
For Equivalency "apache_nieq".

Equivalency 1:
    Name = apache_nieq
    MemberClass = IBM.NetworkInterface
    Resource:Node[Membership] =
{eth0:tsa001.itsc.austin.ibm.com,eth0:tsa002.itsc.austin.ibm.com,eth0:tsa003.it
sc.austin.ibm.com}
    SelectString = ""
    SelectFromPolicy = ANY
    MinimumNecessary = 1
    Subscription = {}
    ActivePeerDomain = httpd_SA_Domain
    ConfigValidity =
```

Define application resource

Now, we define the Apache HTTP Server application. Before defining the application resource, we need to provide scripts for IBM Tivoli System Automation for Multiplatforms to be able to:

- ▶ Start the application
- ▶ Stop the application
- ▶ Monitor the application, in other words, query the status of the application

In addition, we have to define time-out values for the Start, Stop, and Monitor operations.

For our implementation, we decide to use a single directory repository for all scripts (/usr/local/IBM/TSA/scripts). We have to copy these scripts to all nodes in the domain, and the scripts must be located in the same directory path.

We use the automation scripts part of an predefined automation policy provided by IBM at:

<http://www.ibm.com/software/tivoli/products/sys-auto-linux/downloads.html>

The supplied automation policy uses a single script with parameters to start, stop, and query the status of the Apache HTTP Server process. We adapt the supplied script for our scenario-specific location of our httpd.pid file, as presented in the following example (Example 4-6).

Example 4-6 Apache HTTP Server automation script

```
#!/bin/ksh -p
#
# init section
#
UNKNOWN=0
ONLINE=1
OFFLINE=2

APACHECMD=/usr/sbin/apachectl
APACHEPID=/var/run/httpd.pid

Action=${1:-status}

#
# main section
#
case ${Action} in
    start)
        if [ ! -f ${APACHECMD} ]
        then
            logger -i -t "SAM-apache" "the application is not
found"
            RC=1
        else
            ${APACHECMD} start >/dev/null 2>&1

            logger -i -t "SAM-apache" "start order issued for
Apache"
            RC=0
        fi
        ;;
    stop)
        ${APACHECMD} stop >/dev/null 2>&1

        logger -i -t "SAM-apache" "stop order issued for Apache"
        RC=0
        ;;
    status)
        # PIDFILE=`cat ${APACHECMD} | grep PIDFILE= | cut -d'=' -f2`
        if [ -f ${APACHEPID} ] ; then
            PID=`cat ${APACHEPID}`
            if [ "x${PID}" != "x" ] && kill -0 ${PID} 2>/dev/null ;
then
                RC=${ONLINE}
            else
                RC=${OFFLINE}
            fi
        fi
    esac
```

```

                fi
            else
                RC=${OFFLINE}
            fi
        ;;
    *)
        #incorrect input --> log and exit
        logger -i -t "SAM-apache" "Error: Incorrect parameter"
    >${Action}<"
        RC=${UNKNOWN}
        ;;
esac
exit ${RC}

```

Note: For us it was necessary to make a small modification to the supplied Apache HTTP automation script for our environment. We define the path to the Apache PIDFILE within the script. See the “bold” lines in the above listing of the script.

The application resource defined in IBM Tivoli System Automation for Multiplatforms is of class IBM.Application. We again use the **mkrsrc** command to define our Application resource, using the **mkrsrc -f** form with a definition file named **apache.resourcedef.IBM.Application**, which contains the following (see Example 4-7):

Example 4-7 Application definition input file

```

PersistentResourceAttributes::
    Name="apache"
    StartCommand="/usr/local/IBM/TSA/scripts/apache start"
    StopCommand="/usr/local/IBM/TSA/scripts/apache stop"
    MonitorCommand="/usr/local/IBM/TSA/scripts/apache status"
    MonitorCommandPeriod=5
    MonitorCommandTimeout=5
    NodeNameList={"tsa001","tsa002","tsa003"}
    StartCommandTimeout=10
    StopCommandTimeout=10
    UserName="root"
    ResourceType=1

```

The command we use to create our Application resource using the above definition file as input is:

```
# mkrsrc -f apache.resourcedef.IBM.Application IBM.Application
```

We then verify the resource definition with the following command (see Example 4-8):

Example 4-8 Verify application resource definition

1srsrc IBM.Application

Resource Persistent Attributes for IBM.Application

resource 1:

```
Name = "apache"
ResourceType = 0
AggregateResource = "0x2028 0xffff 0x3b4eb30e 0x61e50e32 0x8f9bd741
0x149af380"
StartCommand = "/usr/local/IBM/TSA/scripts/apache start"
StopCommand = "/usr/local/IBM/TSA/scripts/apache stop"
MonitorCommand = "/usr/local/IBM/TSA/scripts/apache status"
MonitorCommandPeriod = 5
MonitorCommandTimeout = 5
StartCommandTimeout = 10
StopCommandTimeout = 10
UserName = "root"
RunCommandsSync = 1
ProtectionMode = 0
HealthCommand = ""
HealthCommandPeriod = 10
HealthCommandTimeout = 5
InstanceName = ""
InstanceLocation = ""
ActivePeerDomain = "httpd_SA_Domain"
NodeNameList = {"tsa003.itsc.austin.ibm.com"}
```

resource 2:

```
Name = "apache"
ResourceType = 0
AggregateResource = "0x2028 0xffff 0x3b4eb30e 0x61e50e32 0x8f9bd741
0x149af380"
StartCommand = "/usr/local/IBM/TSA/scripts/apache start"
StopCommand = "/usr/local/IBM/TSA/scripts/apache stop"
MonitorCommand = "/usr/local/IBM/TSA/scripts/apache status"
MonitorCommandPeriod = 5
MonitorCommandTimeout = 5
StartCommandTimeout = 10
StopCommandTimeout = 10
UserName = "root"
RunCommandsSync = 1
ProtectionMode = 0
HealthCommand = ""
HealthCommandPeriod = 10
HealthCommandTimeout = 5
InstanceName = ""
InstanceLocation = ""
ActivePeerDomain = "httpd_SA_Domain"
NodeNameList = {"tsa002.itsc.austin.ibm.com"}
```

resource 3:


```

        Name = "apache"
        ResourceType = 0
        AggregateResource = "0x2028 0xffff 0x3b4eb30e 0x61e50e32 0x8f9bd741
0x149af380"
        StartCommand = "/usr/local/IBM/TSA/scripts/apache start"
        StopCommand = "/usr/local/IBM/TSA/scripts/apache stop"
        MonitorCommand = "/usr/local/IBM/TSA/scripts/apache status"
        MonitorCommandPeriod = 5
        MonitorCommandTimeout = 5
        StartCommandTimeout = 10
        StopCommandTimeout = 10
        UserName = "root"
        RunCommandsSync = 1
        ProtectionMode = 0
        HealthCommand = ""
        HealthCommandPeriod = 10
        HealthCommandTimeout = 5
        InstanceName = ""
        InstanceLocation = ""
        ActivePeerDomain = "httpd_SA_Domain"
        NodeNameList = {"tsa001.itsc.austin.ibm.com"}
resource 4:
        Name = "apache"
        ResourceType = 1
        AggregateResource = "0x3fff 0xffff 0x00000000 0x00000000 0x00000000
0x00000000"
        StartCommand = "/usr/local/IBM/TSA/scripts/apache start"
        StopCommand = "/usr/local/IBM/TSA/scripts/apache stop"
        MonitorCommand = "/usr/local/IBM/TSA/scripts/apache status"
        MonitorCommandPeriod = 5
        MonitorCommandTimeout = 5
        StartCommandTimeout = 10
        StopCommandTimeout = 10
        UserName = "root"
        RunCommandsSync = 1
        ProtectionMode = 0
        HealthCommand = ""
        HealthCommandPeriod = 10
        HealthCommandTimeout = 5
        InstanceName = ""
        InstanceLocation = ""
        ActivePeerDomain = "httpd_SA_Domain"
        NodeNameList =
{"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.austin.
ibm.com"}

```

At this point, the resources are configured and ready for management by IBM Tivoli System Automation for Multiplatforms. To accomplish this, we need to

create a resource group, place our resources (apache and apache_SIP) into this resource group, and then bring the resource group online.

Tip: You can limit the data returned from the `lsrsrc` command by specifying the attribute names for the resource class, as follows:

```
# lsrsrc -l IBM.Application Name NodeNameList
Resource Persistent Attributes for IBM.Application
resource 1:
    Name          = "apache"
    NodeNameList  = {"tsa003.itsc.austin.ibm.com"}
resource 2:
    Name          = "apache"
    NodeNameList  = {"tsa002.itsc.austin.ibm.com"}
resource 3:
    Name          = "apache"
    NodeNameList  = {"tsa001.itsc.austin.ibm.com"}
resource 4:
    Name          = "apache"
    NodeNameList  =
{"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.aust
in.ibm.com"}
```

Define resource group

All resources groups defined in this automation domain are RSCT resources groups. IBM Tivoli System Automation for Multiplatforms makes use of these resources groups to perform automation operations.

Our resource group is named `apache_rg`. We create it and add the ServiceIP and application resources to it with the following commands (see Example 4-9):

Example 4-9 Create resource group and populate with resources

```
# mkrsg apache_rg
# addrgmbr -g apache_rg IBM.Application:apache
# addrgmbr -g apache_rg IBM.ServiceIP:apache_SIP
```

We display our new resource group with the following command (Example 4-10):

Example 4-10 Display resource group information

```
# lsrg -m
Displaying Member Resource information:
Class:Resource:Node[ManagedResource] Mandatory MemberOf OpState
IBM.ServiceIP:apache_SIP                True      apache_rg  Offline
IBM.Application:apache                   True      apache_rg  Offline
#
```

4.2.3 Create the automation policy using relationship definitions

Now, we are ready to define the relationships between our resources. This defines the automation policy to which IBM Tivoli System Automation for Multiplatforms will adhere and execute.

To this point, our Apache automation domain contains the following resources:

Defined resources and resource group:

Application resource	apache
ServiceIP	apache1IP
Equivalency	apache_nieq
Resource group	apache_rg

In order to complete our automation definition, we must create relationships between the defined resources. For details and rules about valid relationship definitions, refer to the *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04.

According to the automation requirements we define in Section 4.1, “Apache automation domain overview” on page 55, we need to create the following relationships:

- ▶ The Apache application depends on ServiceIP apache_SIP to be active.
- ▶ The ServiceIP apache_SIP depends on the network interface equivalency apache_nieq to be active.

Figure 4-3 shows an overview of our Apache automation domain automation policy:

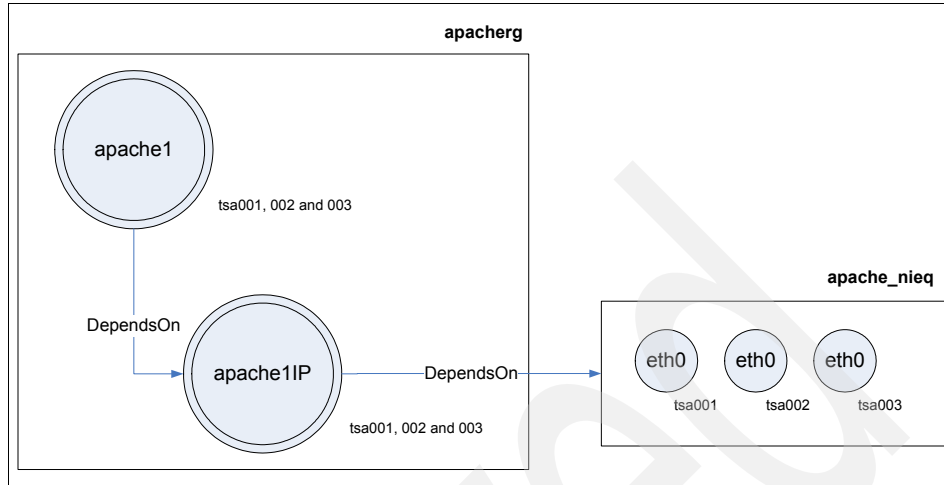


Figure 4-3 Overview of the Apache automation domain

First, we define the relationship between our apache application resource (apache) and our apache ServiceIP resource (apache_SIP) with the following command.

```
# mkrel -p DependsOn -S IBM.Application:apache -G IBM.ServiceIP:apache_SIP
apache_dependson_ip
```

Next, we define the relationship between our ServiceIP resource (apache_SIP) and the network interface equivalency (apache_nieq) with the following command.

```
# mkrel -p DependsOn -S IBM.ServiceIP:apache_SIP -G IBM.Equivalency:apache_nieq
apache_SIP_dependson_apache_nieq
```

You can display the relationship definitions with the following command (Example 4-11):

Example 4-11 Display relationship definitions

```
# lsrel -l
Displaying Managed Relations :
```

Managed Relationship 1:

```
Name = apache_SIP_dependson_apache_nieq
Class:Resource:Node[Source] = IBM.ServiceIP:apache_SIP
ResourceGroup[Source] = apache_rg
```

Managed Relationship 2:

```
Name = apache_dependson_ip
```

```
Class:Resource:Node[Source] = IBM.Application:apache
ResourceGroup[Source]      = apache_rg
```

4.2.4 Change the operational state of the resource group

We now are ready to activate the resource group `apache_rg`. This will modify the operational state of the resource group as well as all the resources defined as its members. We use the **chrg** command as presented as follows:

```
# chrg -o online apache_rg
```

At last, we have a fully defined and configured environment. We verify the status of the resource group with the following command (Example 4-12):

Example 4-12 Display resource group operational state

```
# lsrg -g apache_rg
Displaying Resource Group information:
For Resource Group "apache_rg".
```

Resource Group 1:

```
  Name           = apache_rg
  MemberLocation = Collocated
  Priority        = 0
  AllowedNode     = ALL
  NominalState    = Online
  ExcludedList    = {}
  Subscription    = {}
  Owner           =
  Description     =
  InfoLink        =
  ActivePeerDomain = httpd_SA_Domain
  OpState         = Online
  TopGroup        = apache_rg
  ConfigValidity  =
  TopGroupNominalState = Online
```

Note: The nominalState, this is the state the resource should be in, Online after **chrg -o online apache_rg** or Offline after **chrg -o offline apache_rg**. The OpState is the current operation state, which will match the nominalState in a fully functional environment.

In addition, the managed resources should not be started/stopped by a third party, for example, the Linux run level or manually by the operator. The resources defined should be exclusively under the control of IBM Tivoli System Automation for Multiplatforms once they are managed, in other words, a member of a Resource Group.

Once the apache_rg resource group has an online OpState, we validate the policy definitions we have done so far. We need to check the operational state of all defined application resources. An OpState value of 1 indicates the application resource is online on the node, as shown in the following example (Example 4-13):

Example 4-13 Application resources OpState online

```
# lsrsrc IBM.Application Name NodeNameList OpState
Resource Persistent and Dynamic Attributes for IBM.Application
resource 1:
    Name           = "apache"
    NodeNameList    = {"tsa003.itsc.austin.ibm.com"}
    OpState         = 2
resource 2:
    Name           = "apache"
    NodeNameList    = {"tsa002.itsc.austin.ibm.com"}
    OpState         = 2
resource 3:
    Name           = "apache"
    NodeNameList    = {"tsa001.itsc.austin.ibm.com"}
    OpState         = 1
resource 4:
    Name           = "apache"
    NodeNameList    =
{"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.austin.ibm.com"}
    OpState         = 1
```

In Example 4-13, the Apache HTTP Server, represented by the apache application resource defined in this case study scenario, is running on a single node, tsa001, according to the defined policy (relationships).

4.2.5 Configuration error and recovery example

The pre-GA version of the *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04, manual provides an example to create the ServiceIP with command line input attributes. We convert that example into an input file. Our definition file at first looks like the following example (Example 4-14):

Example 4-14 Initial ServiceIP definition input file data contents

```
PersistentResourceAttributes::
  NodeNameList="{ 'tsa001','tsa002','tsa003' }"
  Name="apache_SIP"
  NetMask=255.255.255.0
  IPAddress=9.3.5.29
  ResourceType=1
```

This seems consistent with the line command example provided in the pre-GA version of the above mentioned manual. We issue the **mkrsrc -f** command and it returned without error. However, our ServiceIP resource did not have any entries in the NodeNameList, see Example 4-15 below.

Example 4-15 Display of ServiceIP resource after initial definition

```
# lsrsrc -l IBM.ServiceIP
Resource Persistent Attributes for IBM.ServiceIP
resource 1:
  Name                = "apache_SIP"
  ResourceType        = 1
  AggregateResource    = "0x3fff 0xffff 0x00000000 0x00000000 0x00000000
0x00000000"
  IPAddress            = "9.3.5.29"
  NetMask              = "255.255.255.0"
  ProtectionMode       = 1
  ActivePeerDomain     = "httpd_SA_Domain"
  NodeNameList        = {}
```

We did not notice this state until later when we were unable to bring the resource group (apache_rg) operational state to Online. Once we identified the empty NodeNameList for the ServiceIP resource as the source of our problem, we correct our definition of the ServiceIP (apache_SIP) with the following **chrsrc** command:

```
# chrsrc -s 'Name == "apache_SIP"' -a IBM.ServiceIP
NodeNameList={"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003
.itsc.austin.ibm.com"}
```

We were still unable to bring the ServiceIP into a known state. At that point, we stop and start the IBM.RecoveryRM daemon which clears the problem in our environment as follows (Example 4-16):

Example 4-16 Clear error with start/stop of IBM.RecoveryRM

```
# stopsrc -s IBM.RecoveryRM
0513-044 The IBM.RecoveryRM Subsystem was requested to stop.

# lssrc -s IBM.RecoveryRM
Subsystem      Group          PID           Status
IBM.RecoveryRM rsct_rm        25293         inoperative

# startsrc -s IBM.RecoveryRM
0513-059 The IBM.RecoveryRM Subsystem has been started. Subsystem PID is 25293.
```

4.2.6 Exercising the automation policy example

With the IBM Tivoli System Automation for Multiplatforms first-level automation domain configured and operational, we can verify the automation policy definitions and operations by simulating a system failure.

We begin with our Apache application running on tsa001 (OpState = 1), as seen in the query below (Example 4-17).

Example 4-17 Display current state of the Apache automation domain

```
# lsrsrc -l IBM.Application NodeNameList OpState
Resource Persistent and Dynamic Attributes for IBM.Application
resource 1:
  NodeNameList = {"tsa003.itsc.austin.ibm.com"}
  OpState      = 2
resource 2:
  NodeNameList = {"tsa002.itsc.austin.ibm.com"}
  OpState      = 2
resource 3:
  NodeNameList = {"tsa001.itsc.austin.ibm.com"}
  OpState      = 1
resource 4:
  NodeNameList =
{"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.austin.
ibm.com"}
  OpState      = 1
```

Note: There is no significance that tsa001 is the current operational Apache HTTP Server. The node on which the application is operational is defined by IBM Tivoli System Automation for Multiplatforms.

The server tsa001 now loses network connectivity (we unplugged the network cable) and IBM Tivoli System Automation for Multiplatforms assigns the ServiceIP to a new node and starts the Apache HTTP Server on the node, in this case, tsa002 node. Note that the status of tsa001 is unknown (OpState = 3) to IBM Tivoli System Automation for Multiplatforms due to the lack of network connectivity (Example 4-18).

Example 4-18 Display state of the Apache automation domain after failure of a node

```
# lsrsrc -l IBM.Application Name NodeNameList OpState
Resource Persistent and Dynamic Attributes for IBM.Application
resource 1:
    Name          = "apache"
    NodeNameList = {"tsa003.itsc.austin.ibm.com"}
    OpState       = 2
resource 2:
    Name          = "apache"
    NodeNameList = {"tsa002.itsc.austin.ibm.com"}
    OpState       = 1
resource 3:
    Name          = "apache"
    NodeNameList = {"tsa001.itsc.austin.ibm.com"}
    OpState       = 3
resource 4:
    Name          = "apache"
    NodeNameList = {"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.austin.ibm.com"}
    OpState       = 1
```

We verify the ServiceIP reassign with the following command (Example 4-19):

Example 4-19 Display status of the ServiceIP after failure of tsa001

```
# lsrsrc -l IBM.ServiceIP Name NodeNameList OpState ResourceType
Resource Persistent and Dynamic Attributes for IBM.ServiceIP
resource 1:
    Name          = "apache_SIP"
    NodeNameList = {"tsa003.itsc.austin.ibm.com"}
    OpState       = 2
    ResourceType = 0
resource 2:
    Name          = "apache_SIP"
    NodeNameList = {"tsa002.itsc.austin.ibm.com"}
    OpState       = 1
    ResourceType = 0
resource 3:
    Name          = "apache_SIP"
```

```

        NodeNameList = {"tsa001.itsc.austin.ibm.com"}
        OpState      = 3
        ResourceType = 0
resource 4:
        Name          = "apache_SIP"
        NodeNameList =
{"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.austin.
ibm.com"}
        OpState      = 1
        ResourceType = 1

```

Later, when network connectivity is reestablished with tsa001, we see the following status for the IBM.Application and IBM.ServiceIP resources.

Example 4-20 Isrsrsrc output

```

# Isrsrsrc -l IBM.Application Name NodeNameList OpState
Resource Persistent and Dynamic Attributes for IBM.Application
resource 1:
        Name          = "apache"
        NodeNameList = {"tsa003.itsc.austin.ibm.com"}
        OpState      = 2
resource 2:
        Name          = "apache"
        NodeNameList = {"tsa002.itsc.austin.ibm.com"}
        OpState      = 1
resource 3:
        Name          = "apache"
        NodeNameList = {"tsa001.itsc.austin.ibm.com"}
        OpState      = 2
resource 4:
        Name          = "apache"
        NodeNameList =
{"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.austin.
ibm.com"}
        OpState      = 1

# Isrsrsrc -l IBM.ServiceIP Name NodeNameList OpState ResourceType
Resource Persistent and Dynamic Attributes for IBM.ServiceIP
resource 1:
        Name          = "apache_SIP"
        NodeNameList = {"tsa003.itsc.austin.ibm.com"}
        OpState      = 2
        ResourceType = 0
resource 2:
        Name          = "apache_SIP"
        NodeNameList = {"tsa002.itsc.austin.ibm.com"}
        OpState      = 1
        ResourceType = 0

```

```

resource 3:
    Name          = "apache_SIP"
    NodeNameList = {"tsa001.itsc.austin.ibm.com"}
    OpState       = 2
    ResourceType = 0
resource 4:
    Name          = "apache_SIP"
    NodeNameList =
{"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.austin.
ibm.com"}
    OpState       = 1
    ResourceType = 1

```

You should notice that IBM Tivoli System Automation for Multiplatforms is aware of the change in the status of tsa001, but it kept the Apache HTTP Server and ServiceIP running on tsa002.

Note: OpState has the following mapping:

0 = Unknown, 1 = Online, 2 = Offline, 3 = Failed Offline

In case Unknown states should arise, it is important to remedy the situation since resources in an Unknown state cannot be effectively controlled by IBM Tivoli System Automation for Multiplatforms. In addition, resources that have a relationship either to or from the resource in an Unknown state cannot be effectively controlled by IBM Tivoli System Automation for Multiplatforms.

As a first step to troubleshooting resources in an Unknown state, check the following:

- ▶ Start, stop, and monitor automation scripts.
- ▶ Timeout values defined for the resources.
- ▶ Start, stop, and monitor automation commands defined for the resource.

4.3 End-to-end Automation Adapter configuration

We now prepare our first-level automation domain for end-to-end automation management through the IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component.

We configure our Linux-based Apache first-level automation domain using the **cfgsamadapter** command found in the IBM Tivoli System Automation for Multiplatforms install directory path: /opt/IBM/tsamp/sam/bin.

Figure 4-4 shows the interaction between our first-level automation domain and the End-to-end Automation Management Component. We discuss the End-to-end Automation Management Component installation and configuration in Chapter 7, “Case study scenario: End-to-end automation domain” on page 201.

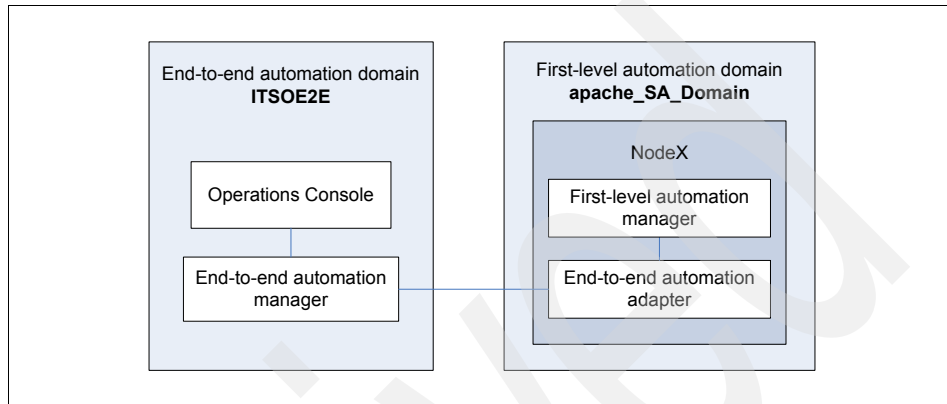


Figure 4-4 End-to-end domain and apache_SA_Domain interaction

We follow the directions provided in *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04, to configure the End-to-end Automation Adapter. The following dialog is launched by cfigsamadater. See Figure 4-5.

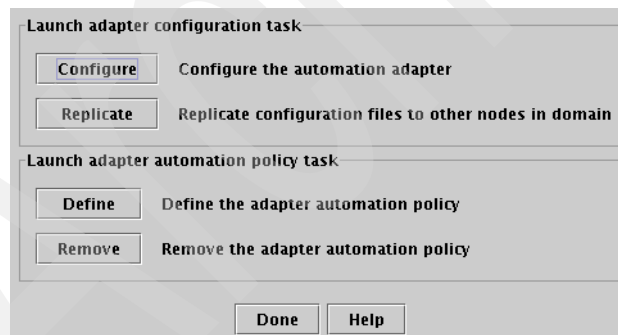


Figure 4-5 Configure the automation adapter for the Apache automation domain

The following sections discuss three of the four operations presented on the dialog box.

- ▶ Configure
- ▶ Replicate
- ▶ Define

4.3.1 Configure the End-to-end Automation Adapter

We select the **Configure** button on Figure 4-5 above to begin configuration of our first-level automation domain End-to-end Automation Adapter. We now have a tabbed dialog box to provide configuration data on the following topics:

- ▶ Adapter
- ▶ Host using adapter
- ▶ Automation
- ▶ Security
- ▶ Logger

We use the defaults for the last two tabs, Security and Logger. We discuss the settings for the remaining tabs below.

Adapter tab

On this tab, we enter the ServiceIP address used by the IBM Tivoli System Automation for Multiplatforms End-to-end Automation Adapter in our Apache automation domain. See Figure 4-6.

Configuration status: Re-configuration

Adapter Host using adapter Automation Security Logger

Automation adapter host

Host name or IP address 9.3.5.23

Request port number 2001

Event port number 5539

Port used to receive requests from the host using the automation adapter
Port used to receive events from the system automation domain

Advanced...

Save Cancel Help

Figure 4-6 Adapter Tab data fields

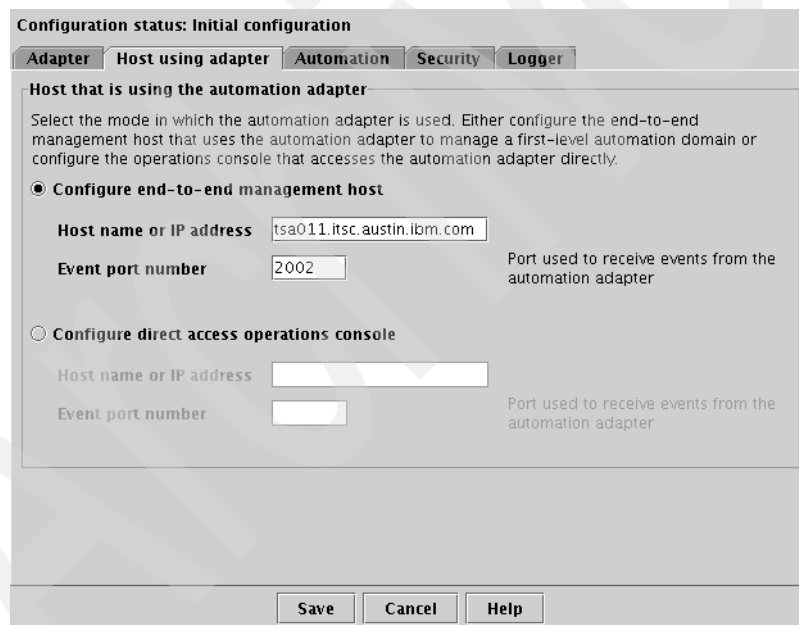
We obtain an IP Address (9.3.5.23) for use by the End-to-end Automation Adapter for our Linux automation domain. This End-to-end Automation Adapter application uses this address on whichever Linux system the adapter is active.

The End-to-end Automation Adapter is much like our Apache application; it is active on one and only one node of our cluster using a fixed (ServiceIP) address. We enter this address into the Host name or IP address entry field and leave the Request port number and Event port number fields with the provided default values.

Note: Use a Host name only if the End-to-end Automation Adapter is not a floating resource, in other words, fixed to a single node of the cluster. This limits functionality should that node fail.

Host using adapter tab

Next, we select the Host using adapter tab. See Figure 4-7. Here we supply the fully qualified host name of the system where our IBM Tivoli System Automation for Multiplatforms End-to-end Automation Management Component is installed, see Chapter 7, “Case study scenario: End-to-end automation domain” on page 201.



The screenshot shows a configuration window titled "Configuration status: Initial configuration". It has five tabs: "Adapter", "Host using adapter" (which is selected), "Automation", "Security", and "Logger". The "Host using adapter" tab contains the following content:

Host that is using the automation adapter

Select the mode in which the automation adapter is used. Either configure the end-to-end management host that uses the automation adapter to manage a first-level automation domain or configure the operations console that accesses the automation adapter directly.

☒ **Configure end-to-end management host**

Host name or IP address:

Event port number: Port used to receive events from the automation adapter

☐ **Configure direct access operations console**

Host name or IP address:

Event port number: Port used to receive events from the automation adapter

At the bottom of the window are three buttons: "Save", "Cancel", and "Help".

Figure 4-7 Host using adapter tab

Automation tab

Next, we select the Automation tab. See Figure 4-8. Here we supply the information necessary to provide high availability to our IBM Tivoli System Automation for Multiplatforms adapter. We check the **Automate adapter in**

system automation domain box. Ensure that the first-level automation domain is online and use the **Query** button to populate the configuration panel with all the nodes in online status. In our case, we want the End-to-end Automation Adapter to be able to run on nodes tsa001, tsa002, and tsa003.

We provide the IBM Tivoli System Automation for Multiplatforms adapter IP Address (9.3.5.23), subnet mask (255.255.255.0), and a prefix for the Apache automation domain (apache_E2E_).

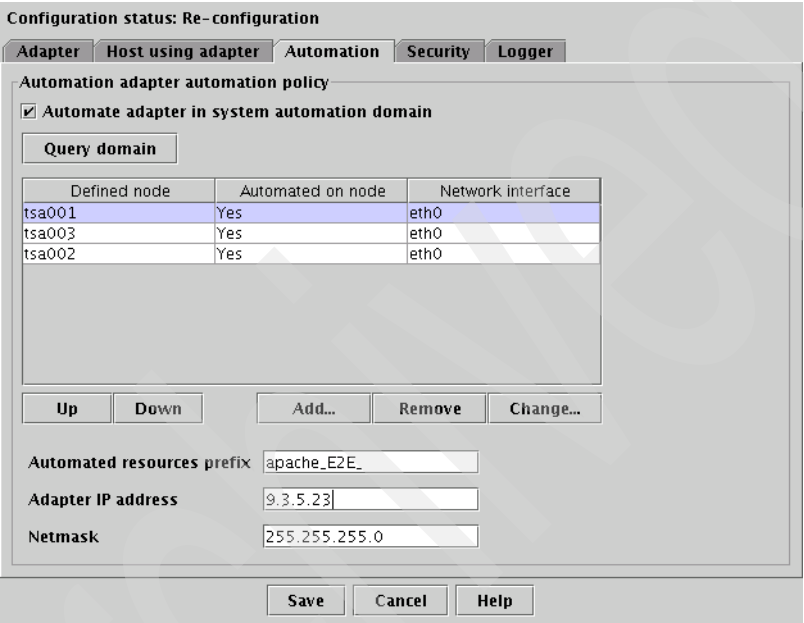


Figure 4-8 Automation Tab

We use the default values and settings for the Security and Logger tabs.

Note: We choose to use a trailing underscore ‘_’ rather than the default dash ‘-’ for the prefix. After our configuration is complete and active, we notice that the IBM.Application resource for the adapter is named:

apache_E2E_

Later, after the configurations are finalized, we are unhappy with the appearance of the trailing underscore character and we rename the resource using the following command:

```
chrsrc -s 'Name == "apache_E2E_"' IBM.Application Name=apache_E2E
```

We then select the **Save** button. As you see in Figure 4-9, we modify several configuration files.

Configuration updated successfully	
Configuration file	Update
/etc/Tivoli/tec/samPublisher.conf	OK
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.conf	OK
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.properties	OK
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.plugin.properties	No change
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.ssl.properties	No change
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.jaas.properties	No change
/etc/opt/IBM/tsamp/sam/cfg/EEZPublisher.conf	OK
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.jlog.properties	OK
<div>OK Help</div>	

Figure 4-9 Modified End-to-end Automation Adapter configuration files

4.3.2 Replicate configuration files to nodes in the automation domain

Now that we have provided the IBM Tivoli System Automation for Multiplatforms End-to-end Automation Adapter configuration data, we need to replicate this data to all the nodes in the automation domain that will be eligible to run the End-to-end Automation Adapter. We execute the `cfgsamadapter` command again and select the Replicate button in the main dialog box, shown in Figure 4-5 on page 76.

A new dialog box appears and we select all configuration files and all nodes using the respective **Select all** buttons. We then provide the login user id and password of those target nodes and select the **>> Replicate >>** button. See Figure 4-10.

Select the configuration files to be replicated	Target node login	Select the replication target nodes
<div>Configuration files</div> <div>/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.conf /etc/opt/IBM/tsamp/sam/cfg/sam.adapter.properties /etc/opt/IBM/tsamp/sam/cfg/sam.adapter.plugin.proper... /etc/opt/IBM/tsamp/sam/cfg/sam.adapter.jaas.properties /etc/opt/IBM/tsamp/sam/cfg/sam.adapter.ssl.properties /etc/opt/IBM/tsamp/sam/cfg/EEZPublisher.conf /etc/opt/IBM/tsamp/sam/cfg/sam.adapter.jlog.properties</div> <div>Select all</div>	<div>User ID root</div> <div>Password *****</div> <div>>> Replicate >></div> <div>Done Help</div>	<div>Other nodes in domain</div> <div>tsa002.itsc.austin.ibm.com tsa003.itsc.austin.ibm.com</div> <div>Select all</div>

Figure 4-10 Replicate configuration files to other nodes in automation domain

This process contacts each node, and processing time varies with the number of nodes.

Important: The replication task uses the user credentials provided in the User ID and Password fields. Ensure that every single node uses the same user ID and password combination. In addition, the Replication process uses secure shell communications to transfer the configuration files over to the target nodes. In our case, sshd is operational on every node.

Figure 4-11 appears at the completion of the configuration data replication.

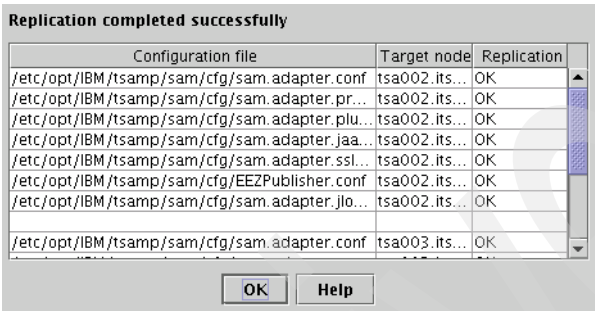


Figure 4-11 Adapter configuration replication completion dialog box

4.3.3 Define the End-to-end Automation Adapter automation policy

Next, we have to define the policy for automating the End-to-end Automation Adapter in the automation domain. This task defines resources, resource groups, equivalencies, and finally relationships between resources.

We launch the End-to-end Automation Adapter configuration tool by executing the `cfgsamadapter` command (Figure 4-5 on page 76) and selecting **Define**.

At completion, Figure 4-12 on page 82 displays.

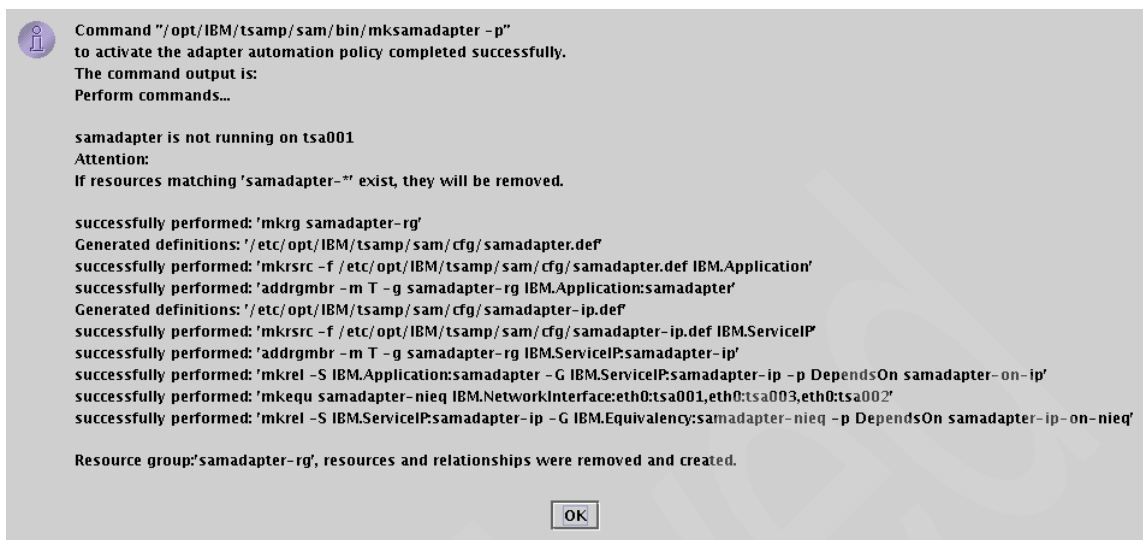


Figure 4-12 Defining automation policy for End-to-end Automation Adapter

We then review the modifications to our IBM Tivoli System Automation for Multiplatforms automation domain. We display the resource groups as follows (Example 4-21):

Example 4-21 Post adapter configuration resource group display

```
# lsrg -m
Displaying Member Resource information:
Class:Resource:Node[ManagedResource] Mandatory MemberOf OpState
IBM.ServiceIP:apache_E2E_ip True apache_E2E_rg Offline
IBM.Application:apache_E2E True apache_E2E_rg Offline
IBM.ServiceIP:apache_SIP True apache_rg Online
IBM.Application:apache True apache_rg Online
```

Note that the End-to-end Automation Adapter configuration created a new resource group (apache_E2E_rg) in our first-level automation domain.

At this point, we start the End-to-end Automation Adapter application with the following command:

```
# chrg -o online apache_E2E_rg
```

This command is valid on any of our three nodes in the automation domain (Example 4-22).

Example 4-22 Post adapter configuration resource group display

```
# lsrg -m
```

Displaying Member Resource information:			
Class:Resource:Node[ManagedResource]	Mandatory	MemberOf	OpState
IBM.ServiceIP:apache_E2E_ip	True	apache_E2E_rg	Online
IBM.Application:apache_E2E	True	apache_E2E_rg	Online
IBM.ServiceIP:apache_SIP	True	apache_rg	Online
IBM.Application:apache	True	apache_rg	Online

4.4 Miscellaneous information

This section contains examples and displays of various information about our first-level automation domain upon completion of our configuration, customization, and activation.

Resource group

We have two resource groups as seen below (Example 4-23):

Example 4-23 Display of all resource groups in the Apache automation domain

```
# lsrg -m
```

Displaying Member Resource information:			
Class:Resource:Node[ManagedResource]	Mandatory	MemberOf	OpState
IBM.ServiceIP:apache_E2E_ip	True	apache_E2E_rg	Online
IBM.Application:apache_E2E	True	apache_E2E_rg	Online
IBM.ServiceIP:apache_SIP	True	apache_rg	Online
IBM.Application:apache	True	apache_rg	Online

Application

For the status of our two applications (Apache and End-to-end adapter), see Example 4-24.

Example 4-24 Display of all applications in the automation domain

```
# lsrsrc -l IBM.Application Name NodeNameList OpState
Resource Persistent and Dynamic Attributes for IBM.Application
resource 1:
    Name           = "apache_E2E"
    NodeNameList   = {"tsa003.itsc.austin.ibm.com"}
    OpState        = 2
resource 2:
    Name           = "apache_E2E"
    NodeNameList   = {"tsa001.itsc.austin.ibm.com"}
    OpState        = 2
resource 3:
```

```

        Name          = "apache_E2E"
        NodeNameList = {"tsa002.itsc.austin.ibm.com"}
        OpState       = 1
resource 4:
        Name          = "apache_E2E"
        NodeNameList =
{"tsa002.itsc.austin.ibm.com","tsa001.itsc.austin.ibm.com","tsa003.itsc.austin.
ibm.com"}
        OpState       = 1
resource 5:
        Name          = "apache"
        NodeNameList = {"tsa003.itsc.austin.ibm.com"}
        OpState       = 2
resource 6:
        Name          = "apache"
        NodeNameList = {"tsa002.itsc.austin.ibm.com"}
        OpState       = 2
resource 7:
        Name          = "apache"
        NodeNameList = {"tsa001.itsc.austin.ibm.com"}
        OpState       = 1
resource 8:
        Name          = "apache"
        NodeNameList =
{"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.austin.
ibm.com"}
        OpState       = 1

```

ServiceIP

Our ServiceIP resources. See Example 4-25.

Example 4-25 ServiceIP resources

```

# lsrsrc -l IBM.ServiceIP Name NodeNameList IPAddress OpState
Resource Persistent and Dynamic Attributes for IBM.ServiceIP
resource 1:
        Name          = "apache_E2E_ip"
        NodeNameList = {"tsa003.itsc.austin.ibm.com"}
        IPAddress     = "9.3.5.23"
        OpState       = 2
resource 2:
        Name          = "apache_E2E_ip"
        NodeNameList = {"tsa001.itsc.austin.ibm.com"}
        IPAddress     = "9.3.5.23"
        OpState       = 2
resource 3:
        Name          = "apache_E2E_ip"
        NodeNameList = {"tsa002.itsc.austin.ibm.com"}

```

```

        IPAddress    = "9.3.5.23"
        OpState      = 1
resource 4:
        Name         = "apache_E2E_ip"
        NodeNameList =
{"tsa002.itsc.austin.ibm.com","tsa001.itsc.austin.ibm.com","tsa003.itsc.austin.
ibm.com"}
        IPAddress    = "9.3.5.23"
        OpState      = 1
resource 5:
        Name         = "apache_SIP"
        NodeNameList = {"tsa003.itsc.austin.ibm.com"}
        IPAddress    = "9.3.5.29"
        OpState      = 2
resource 6:
        Name         = "apache_SIP"
        NodeNameList = {"tsa002.itsc.austin.ibm.com"}
        IPAddress    = "9.3.5.29"
        OpState      = 2
resource 7:
        Name         = "apache_SIP"
        NodeNameList = {"tsa001.itsc.austin.ibm.com"}
        IPAddress    = "9.3.5.29"
        OpState      = 1
resource 8:
        Name         = "apache_SIP"
        NodeNameList =
{"tsa001.itsc.austin.ibm.com","tsa002.itsc.austin.ibm.com","tsa003.itsc.austin.
ibm.com"}
        IPAddress    = "9.3.5.29"
        OpState      = 1

```

Equivalency

We display our equivalency definitions with both persistent and dynamic attributes below. See Example 4-26.

Example 4-26 Apache automation domain equivalencies

```

# lsequ -A b
Displaying Equivalency information:
All Attributes

Equivalency 1:
        Name                     = apache_E2E_nieq
        MemberClass               = IBM.NetworkInterface
        Resource:Node[Membership] =
{"eth0:tsa002.itsc.austin.ibm.com,eth0:tsa001.itsc.austin.ibm.com,eth0:tsa003.it
sc.austin.ibm.com"}

```

```

        SelectString                        = ""
        SelectFromPolicy                   = ANY
        MinimumNecessary                   = 1
        Subscription                       = {}
        ActivePeerDomain                   = httpd_SA_Domain
        Resource:Node[ValidSelectResources] =
{eth0:t5a002.itsc.austin.ibm.com,eth0:t5a001.itsc.austin.ibm.com,eth0:t5a003.it
sc.austin.ibm.com}
        Resource:Node[InvalidResources]    = {}
        ConfigValidity                     =
        AutomationDetails[CompoundState]   = Undefined

Equivalency 2:
        Name                              = apache_nieq
        MemberClass                        = IBM.NetworkInterface
        Resource:Node[Membership]          =
{eth0:t5a001.itsc.austin.ibm.com,eth0:t5a002.itsc.austin.ibm.com,eth0:t5a003.it
sc.austin.ibm.com}
        SelectString                      = ""
        SelectFromPolicy                   = ANY
        MinimumNecessary                   = 1
        Subscription                       = {}
        ActivePeerDomain                   = httpd_SA_Domain
        Resource:Node[ValidSelectResources] =
{eth0:t5a001.itsc.austin.ibm.com,eth0:t5a002.itsc.austin.ibm.com,eth0:t5a003.it
sc.austin.ibm.com}
        Resource:Node[InvalidResources]    = {}
        ConfigValidity                     =
        AutomationDetails[CompoundState]   = Undefined

```

Relationships

We show our relationships below (see Example 4-27).

Example 4-27 Apache automation domain relationships

```

# lsrel -l
Displaying Managed Relations :

Managed Relationship 1:
        Name                              = apache_E2E_ip-on-nieq
        Class:Resource:Node[Source]        = IBM.ServiceIP:apache_E2E_ip
        ResourceGroup[Source]              = apache_E2E_rg

Managed Relationship 2:
        Name                              = apache_SIP_dependson_apache_nieq
        Class:Resource:Node[Source]        = IBM.ServiceIP:apache_SIP
        ResourceGroup[Source]              = apache_rg

```

Managed Relationship 3:

Name = apache_E2E_on-ip
Class:Resource:Node[Source] = IBM.Application:apache_E2E
ResourceGroup[Source] = apache_E2E_rg

Managed Relationship 4:

Name = apache_dependson_ip
Class:Resource:Node[Source] = IBM.Application:apache
ResourceGroup[Source] = apache_rg

Active ServiceIPs on network adapter

The node on which our Apache application resource and the End-to-end Automation Adapter currently run has three IP addresses assigned to it. Two of these IP addresses are the defined ServiceIP and the third one is the fixed IP address for the node (in this example, tsa001). See Example 4-28.

Example 4-28 ifconfig on tsa001 with End-to-end automation

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:02:55:C6:E4:E4
          inet addr:9.3.5.97  Bcast:9.3.5.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:254931 errors:0 dropped:0 overruns:0 frame:0
          TX packets:153243 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:30260658 (28.8 Mb)  TX bytes:25404806 (24.2 Mb)

eth0:0    Link encap:Ethernet  HWaddr 00:02:55:C6:E4:E4
          inet addr:9.3.5.29  Bcast:9.3.5.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

eth0:1    Link encap:Ethernet  HWaddr 00:02:55:C6:E4:E4
          inet addr:9.3.5.23  Bcast:9.3.5.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:379 errors:0 dropped:0 overruns:0 frame:0
          TX packets:379 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:50795 (49.6 Kb)  TX bytes:50795 (49.6 Kb)
```

Case study scenario: Application Servers on AIX first-level automation domain

In this chapter, we discuss the high availability and automation of the sample application running on the IBM WebSphere Application Server cluster. We show this by creating and configuring a IBM Tivoli System Automation for Multiplatforms first-level automation domain. In addition, we configure an End-to-end Automation Adapter so that the IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component can manage this first-level automation domain.

We discuss the following topics:

- ▶ “Application server automation domain overview” on page 91
- ▶ “Automation domain configuration” on page 92
- ▶ “End-to-end Automation Adapter configuration” on page 118
- ▶ “Maintaining defined policies” on page 127

Figure 5-1 shows the portion of the entire case study scenario we cover in this chapter.

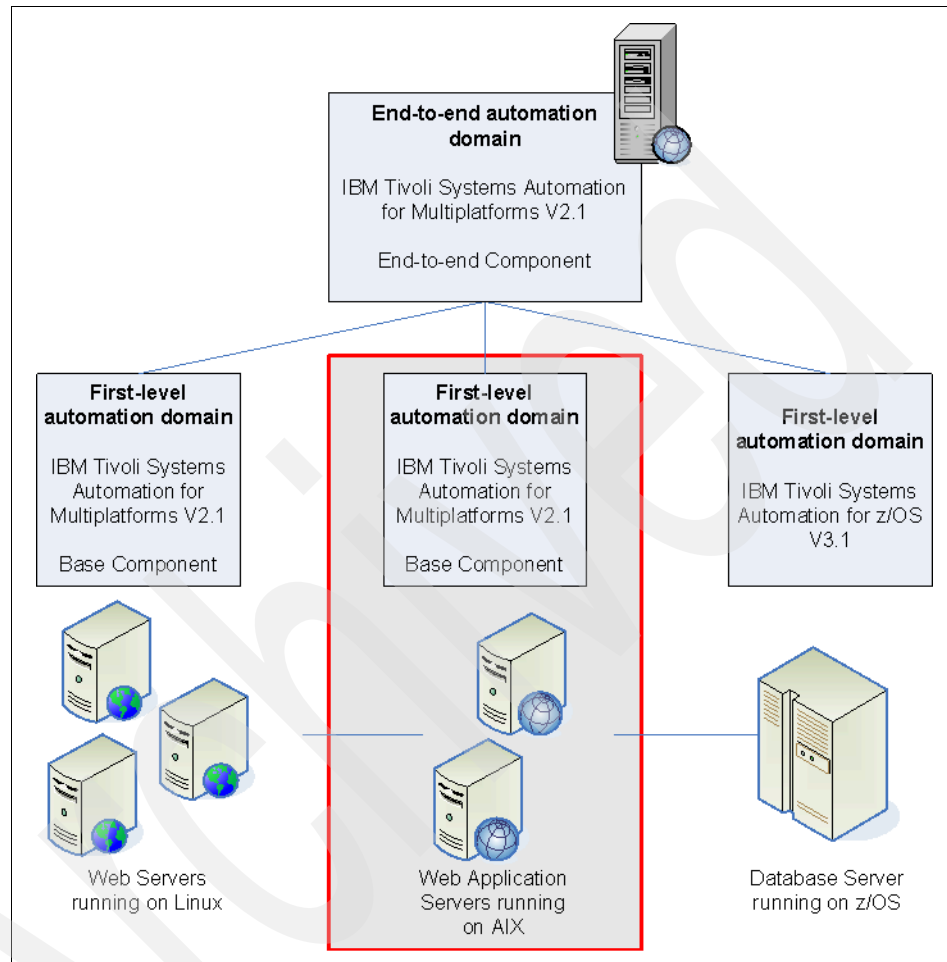


Figure 5-1 Application server first-level automation domain

5.1 Application server automation domain overview

This section describes the configuration steps for setting up high availability for the servers that provide middleware services for our sample application. As you see in Figure 5-2, these servers are running AIX 5.3, IBM WebSphere Application Server, IBM DB2 Client, and the J2EE application, TRADE3. Both servers in the cluster have identical hardware and software configurations.

Figure 5-2 shows the configuration of one of the servers in the cluster and all elements that will be part of the IBM Tivoli System Automation for Multiplatforms first-level automation domain configuration.

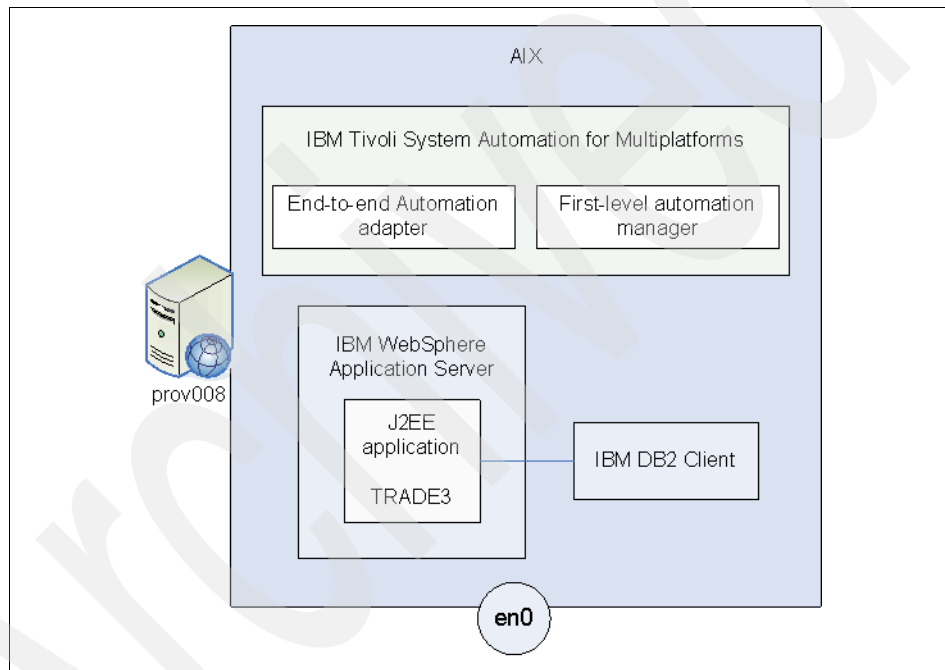


Figure 5-2 Web application tier configuration

We install IBM Tivoli System Automation for Multiplatforms V2.1 Base Component on all nodes that will be part of the automation domain. In our scenario, they are prov008 and prov009. For installation instructions, refer to *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04.

5.1.1 Automation requirements

We must define the following high availability and automation requirements for our environment:

- ▶ IBM WebSphere Application Server, IBM DB2 client, and the Trade3 application must be startable on any node in the domain.
- ▶ IBM WebSphere Application Server, IBM DB2 client, and the Trade3 application will be active on only one node in the domain at any given point in time.
- ▶ In case of a failure, IBM WebSphere Application Server, IBM DB2, and the Trade3 application must be capable of being restarted on any node in the domain.
- ▶ IBM WebSphere Application Server, IBM DB2, and the Trade3 application must run on the same node in the domain.
- ▶ IBM WebSphere Application Server, IBM DB2, and the Trade3 application are installed on all nodes in our domain.
- ▶ The HTTP servers (defined in Chapter 4, “Case study scenario: HTTP Servers on Linux first-level automation domain” on page 53) must always access the Trade3 application using the same IP address, regardless of on which node IBM WebSphere Application Server is running. This IP address serves as the Service IP for the domain.

The following sections go into detail about how we meet these requirements in our environment scenario.

5.2 Automation domain configuration

Setting up the high availability scenario requires the following configuration steps:

- ▶ Create the automation domain
- ▶ Define the resources for automation
- ▶ Create the automation policy using relationship definitions
- ▶ Change the Operational State of the resource group
- ▶ Verify the operational quorum and create a tie breaker

The following sections provide details about how we accomplish these tasks for our case scenario.

5.2.1 Create the first-level automation domain

In order to create the automation domain for our case study environment, we need to perform the following tasks:

- ▶ Perform node preparation
- ▶ Define and start the automation domain
- ▶ Verify the recovery resource manager

Perform node preparation

The **preprnode** command prepares security on the node on which the command is run so it can be defined in a domain. The **preprnode** command allows for peer domain operations to be performed on this node and must be run before the node can join a domain.

The **preprnode** command must be run on every node in order for that node to be defined to the new domain. This gives the necessary authority to create the domain configuration on each new node.

In our environment, we issue the following command on all servers (servers prov008 and prov009) that become part of the automation domain:

```
# preprnode prov008 prov009
```

Note: Once the **preprnode** command has been successfully run on all servers that will be part of the automation domain, all other commands shown in this chapter can be issued from *any* node in the automation domain as long as the node is operational.

Define and start the automation domain

We use the automation domain to provide high availability services for all the nodes that belong to it. Before creating the automation domain, ensure that you have properly prepared the nodes using the **preprnode** command.

In order to define the automation domain in our scenario environment, we use the **mkrpdomain** command. You can issue this command from any node that will be part of the automation domain.

```
# mkrpdomain was_SA_Domain prov008 prov009
```

The **mkrpdomain** command above creates an automation domain named `was_SA_Domain` and the nodes `prov008` and `prov009` are defined in the domain. We also refer to the automation domain as a *cluster*.

To check the status of the newly created was_SA_Domain automation domain, issue the **lsrpdomain** command as follows:

```
# lsrpdomain
Name           OpState RSCTActiveVersion MixedVersions TSPort GSPort
was_SA_Domain Offline 2.4.3.0           No           12347 12348
```

Note that the operational state (OpState) of the automation domain is set to offline. We need to bring the automation domain to an online operational state using the **starttrpdomain** command as follows:

```
# starttrpdomain was_SA_Domain
```

Issue the **lsrpdomain** command to verify the OpState of the automation domain as shown in Example 5-1.

Example 5-1 was_SA_Domain automation domain OpState

```
# lsrpdomain
Name           OpState RSCTActiveVersion MixedVersions TSPort GSPort
was_SA_Domain Pending online 2.4.3.0           No           12347 12348
#
# lsrpdomain
Name           OpState RSCTActiveVersion MixedVersions TSPort GSPort
was_SA_Domain Online 2.4.3.0           No           12347 12348
#
```

In order to verify the status of the nodes that are part of the automation domain, issue the **lsrpnod** command as follows:

```
# lsrpnod
Name   OpState RSCTVersion
prov008 Online 2.4.3.0
prov009 Online 2.4.3.0
```

Verify the recovery resource manager

All nodes that are part of the automation domain have an IBM Tivoli System Automation for Multiplatforms daemon running in case their OpState is online. IBM.RecoveryRM is the name of the daemon.

In order to check the status of the IBM.RecoveryRM daemon, issue the following **lssrc** command, as shown in Example 5-2.

Example 5-2 IBM.RecoveryRM status

```
# lssrc -ls IBM.RecoveryRM
Subsystem      : IBM.RecoveryRM
PID            : 28904
Cluster Name   : was_SA_Domain
```

```
Node Number      : 1
Daemon start time : Mon Aug 22 14:32:51 CDT 2005
```

Daemon State:

```
My Node Name      : prov008
Master Node Name   : prov009 (node number = 2)
Our IVN           : 2.1.0.0
Our AVN           : 2.1.0.0
Our CVN           : 00 (0x0)
Total Node Count   : 2
Joined Member Count : 2
Config Quorum Count : 2
Startup Quorum Count : 1
Operational Quorum State: HAS_QUORUM
In Config Quorum   : TRUE
In Config State     : TRUE
Replace Config State : FALSE
```

Information from malloc about memory use:

```
Total Space      : 0x007502b0 (7668400)
Allocated Space: 0x00712d60 (7417184)
Unused Space     : 0x000259c0 (154048)
Freeable Space   : 0x00000000 (0)
```

Note that the output in Example 5-2 shows one of the nodes in the domain as the *Master Node*. The IBM.RecoveryRM daemon running on the Master Node is responsible for driving all the required high availability actions and decisions in the domain. In our case, at this point in time, the node prov009 is the Master Node in our domain.

In case the IBM.RecoveryRM daemon is inactive, use the **startsrc -s IBM.RecoveryRM** command. To stop the daemon, use **stopsrc -s IBM.RecoveryRM**.

5.2.2 Define automation domain resources

In order to define the automation domain resources and group them into logical manageable entities for our case study environment, we need to perform the following tasks:

- ▶ Define application resources
- ▶ Define network resources
- ▶ Create network equivalencies
- ▶ Define resource groups

Define the application resources

In our scenario, we want to achieve a high availability configuration for the servers running our sample J2EE application on IBM WebSphere Application Server.

Based on the high availability requirements defined in 5.1, “Application server automation domain overview” on page 91, we need to define the following application resources. These resources represent the applications running on our servers:

- ▶ IBM WebSphere Application Server
- ▶ IBM DB2 client
- ▶ Our sample scenario J2EE application: Trade3

Note: All application resources defined in this section are Reliable Scalable Cluster Technology (RSCT) resources. IBM Tivoli System Automation for Multiplatforms makes use of these resources to perform automation operations. The application resource we define here is of class **IBM.Application**.

We define these application resources to IBM Tivoli System Automation for Multiplatforms as *floating* resources, since they can run on any node in the domain.

Before we can define the application resources, we must define how IBM Tivoli System Automation for Multiplatforms will act in the following situations: Start the application, stop the application, and monitor the status of the application. We define this by informing IBM Tivoli System Automation for Multiplatforms of the commands and scripts to use in these situations.

As a best practice, for our implementation, we decide to use a single directory repository for all scripts (/usr/local/IBM/TSA/scripts). We must copy these scripts to all nodes in the domain and they must be located in the same directory path.

For more information about how to define application resources, refer to *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04.

IBM WebSphere Application Server application resource definition

We create a single monitoring script for starting, stopping, and monitoring IBM WebSphere Application Server on the nodes.

Example 5-3 shows the script we use in our scenario. Note that all nodes in our domain are running AIX 5.3.

Example 5-3 Monitoring script for application resource: WebSphere.sh

```
#!/bin/ksh

# Set WAS_HOME
WAS_HOME=/usr/IBM/WebSphere/AppServer

OPSTATE_ONLINE=1
OPSTATE_OFFLINE=2

Action=${1}

case ${Action} in
    start)
        $WAS_HOME/bin/startServer.sh server1 > /dev/null 2>&1
        logger -i -t "SAM-WebSphere" "WebSphere started"
        RC=0
        ;;
    stop)
        $WAS_HOME/bin/stopServer.sh server1 > /dev/null 2>&1
        logger -i -t "SAM-WebSphere" "WebSphere stoped"
        RC=0
        ;;
    status)
        RC=${OPSTATE_OFFLINE}
        STATUS=`/usr/IBM/WebSphere/AppServer/bin/serverStatus.sh
server1 | grep STARTED` && RC=${OPSTATE_ONLINE}
        ;;
    esac

exit $RC
```

Example 5-3 must be created and located in the same directory path in every node in the domain.

In order to define the application resource, we create a file named **WebSphere.resourcedef.IBM.Application** with the definition attributes. See Example 5-4. We use this file later as an input parameter for the **mkrsrc** command.

Example 5-4 IBM.Application WebSphere resource definition file

```
PersistentResourceAttributes::
    Name="websphereResource"
    StartCommand="/usr/local/IBM/TSA/scripts/websphere.sh start"
    StopCommand="/usr/local/IBM/TSA/scripts/websphere.sh stop"
    MonitorCommand="/usr/local/IBM/TSA/scripts/websphere.sh status"
```

```
MonitorCommandPeriod=90
MonitorCommandTimeout=85
NodeNameList={"prov008","prov009"}
StartCommandTimeout=80
StopCommandTimeout=80
UserName="root"
ResourceType=1
```

In Example 5-4, `ResourceType=1` specifies that the application resource will be a floating resource. A floating resource can run on several nodes in the domain. The floating resource is also referred to as an *aggregate* resource.

In our scenario, the application resource `WebSphereResource` is a serial floating resource. We define this by the attribute `NodeNameList` containing multiple entries (`NodeNameList={"prov008","prov009"}`). Being a serial floating resource, the IBM WebSphere Application Server is able to run on multiple nodes in the domain, but only one instance will be active at any given point in time.

Note: The above `Period` and `Timeout` values defined in Example 5-4 are not proper for production environments. You must tune these values to each environment depending on how often IBM Tivoli System Automation for Multiplatforms should monitor the resource, and how long it takes to start and stop the application resource. We use the above values in our scenario environment for testing purposes only.

In order to define the IBM WebSphere Application Server application resource of class `IBM.Application`, we use the following command.

```
# mkrsrc -f WebSphere.resourcedef.IBM.Application IBM.Application
```

You can verify the `IBM.Application` resource definition by using the `lsrsrc` command, as shown in Example 5-5.

Example 5-5 IBM WebSphere Application Server application resource

```
# lsrsrc -l IBM.Application
Resource Persistent Attributes for IBM.Application
resource 1:
  Name = "websphereResource"
  ResourceType = 0
  AggregateResource = "0x2028 0xffff 0xd37b4a18 0x78890a88 0x8f9be739 0x409148bd"
  StartCommand = "/usr/local/IBM/TSA/scripts/websphere.sh start"
  StopCommand = "/usr/local/IBM/TSA/scripts/websphere.sh stop"
  MonitorCommand = "/usr/local/IBM/TSA/scripts/websphere.sh status"
  MonitorCommandPeriod = 90
  MonitorCommandTimeout = 85
  StartCommandTimeout = 80
```

```

StopCommandTimeout      = 80
UserName                 = "root"
RunCommandsSync          = 1
ProtectionMode            = 0
HealthCommand            = ""
HealthCommandPeriod      = 10
HealthCommandTimeout     = 5
InstanceName             = ""
InstanceLocation         = ""
ActivePeerDomain         = "was_SA_Domain"
NodeNameList             = {"prov009.itsc.austin.ibm.com"}
resource 2:
  Name                   = "websphereResource"
  ResourceType          = 0
  AggregateResource      = "0x2028 0xffff 0xd37b4a18 0x78890a88 0x8f9be739 0x409148bd"
  StartCommand           = "/usr/local/IBM/TSA/scripts/websphere.sh start"
  StopCommand            = "/usr/local/IBM/TSA/scripts/websphere.sh stop"
  MonitorCommand         = "/usr/local/IBM/TSA/scripts/websphere.sh status"
  MonitorCommandPeriod   = 90
  MonitorCommandTimeout  = 85
  StartCommandTimeout    = 80
  StopCommandTimeout     = 80
  UserName               = "root"
  RunCommandsSync        = 1
  ProtectionMode          = 0
  HealthCommand          = ""
  HealthCommandPeriod    = 10
  HealthCommandTimeout   = 5
  InstanceName           = ""
  InstanceLocation       = ""
  ActivePeerDomain       = "was_SA_Domain"
  NodeNameList           = {"prov008.itsc.austin.ibm.com"}
resource 3:
  Name                   = "websphereResource"
  ResourceType          = 1
  AggregateResource      = "0x3fff 0xffff 0x00000000 0x00000000 0x00000000 0x00000000"
  StartCommand           = "/usr/local/IBM/TSA/scripts/websphere.sh start"
  StopCommand            = "/usr/local/IBM/TSA/scripts/websphere.sh stop"
  MonitorCommand         = "/usr/local/IBM/TSA/scripts/websphere.sh status"
  MonitorCommandPeriod   = 90
  MonitorCommandTimeout  = 85
  StartCommandTimeout    = 80
  StopCommandTimeout     = 80
  UserName               = "root"
  RunCommandsSync        = 1
  ProtectionMode          = 0
  HealthCommand          = ""
  HealthCommandPeriod    = 10
  HealthCommandTimeout   = 5

```

```

InstanceName      = ""
InstanceLocation  = ""
ActivePeerDomain  = "was_SA_Domain"
NodeNameList      = {"prov008.itsc.austin.ibm.com","prov009.itsc.austin.ibm.com"}

```

#

Note that Example 5-5 shows three resources. The aggregate resource has a `ResourceType` attribute value of 1. The other resources identified with `ResourceType=0` are created by the resource manager `IBM.Application` on each node on which the resource is supposed to run. The resources with `ResourceType=0` are called *constituent resources* (a fixed resource), and the `NodeNameList` contains one node only. At the time of creation, the other attributes have identical values as the aggregate resource.

IBM DB2 client application resource definition

Since our sample application requires IBM DB2 Client, we also define an application resource for IBM DB2. We use the IBM DB2 monitoring scripts provided by IBM at:

<http://www.ibm.com/software/tivoli/products/sys-auto-linux/downloads.html>

In order to define the application resource, we create a file named **db2.resourcedef.IBM.Application** with the definition attributes, shown in Example 5-6. We use this file later as an input parameter for the `mkrsrc` command.

Example 5-6 IBM.Application db2 resource definition file

```

PersistentResourceAttributes::
  Name="db2Resource"
  StartCommand="/usr/local/IBM/TSA/scripts/db2_start.ksh db2inst1"
  StopCommand="/usr/local/IBM/TSA/scripts/db2_stop.ksh db2inst1"
  MonitorCommand="/usr/local/IBM/TSA/scripts/db2_monitor.ksh db2inst1"
  MonitorCommandPeriod=30
  MonitorCommandTimeout=25
  NodeNameList={"prov008","prov009"}
  StartCommandTimeout=20
  StopCommandTimeout=20
  UserName="root"
  ResourceType=1

```

In order to define the IBM DB2 application resource of class `IBM.Application`, we use the following command.

```
# mkrsrc -f db2.resourcedef.IBM.Application IBM.Application
```

Trade3 application resource definition

We created a single monitoring script for starting, stopping, and monitoring the sample J2EE application used in this scenario. Example 5-7 shows the monitoring script used in our scenario.

Example 5-7 Monitoring script for the J2EE application

```
#!/bin/ksh

# Set WAS_HOME
WAS_HOME=/usr/IBM/WebSphere/AppServer

hostname=`hostname`
url="http://$hostname.itsc.austin.ibm.com:9080/trade"

OPSTATE_ONLINE=1
OPSTATE_OFFLINE=2

Action=${1}

case ${Action} in
    start)
        $WAS_HOME/bin/wsadmin.sh -f trade3_start.jacl > /dev/null 2>&1
        logger -i -t "SAM-WebSphere" "Trade3 started"
        RC=0
        ;;
    stop)
        $WAS_HOME/bin/wsadmin.sh -f trade3_stop.jacl > /dev/null 2>&1
        logger -i -t "SAM-WebSphere" "Trade3 stopped"
        RC=0
        ;;
    status)
        /usr/local/IBM/TSA/scripts/wget $url > /dev/null 2>&1
        if [ $? == 0 ];
        then
            RC=${OPSTATE_ONLINE}
            rm /usr/local/IBM/TSA/scripts/index.html
        else
            RC=${OPSTATE_OFFLINE}
        fi
        ;;
esac

exit $RC
```

The scripts in Example 5-7 start and stop the sample J2EE application by executing the **wsadmin** command using JACL scripts as input. Example 5-8 shows the `trade3_start` JACL script.

Example 5-8 trade3_start.jacl script

```
set c11 [$AdminControl queryNames
type=ApplicationManager,node=prov008Node01,process=server1,*]
$AdminControl invoke $c11 startApplication trade3
```

Example 5-9 shows the `trade3_stop` JACL script.

Example 5-9 trade3_stop.jacl script

```
set c11 [$AdminControl queryNames
type=ApplicationManager,node=prov008Node01,process=server1,*]
$AdminControl invoke $c11 stopApplication trade3
```

In order to define the application resource for the sample application, we create a file named `trade3.resourcedef.IBM.Application` with the definition attributes, shown in Example 5-10. We use this file later as an input parameter for the **mkrsrc** command.

Example 5-10 IBM.Application trade3 resource definition file

```
PersistentResourceAttributes::
  Name="trade3Resource"
  StartCommand="/usr/local/IBM/TSA/scripts/trade3.sh start"
  StopCommand="/usr/local/IBM/TSA/scripts/trade3.sh stop"
  MonitorCommand="/usr/local/IBM/TSA/scripts/trade3.sh status"
  MonitorCommandPeriod=90
  MonitorCommandTimeout=85
  NodeNameList={"prov008","prov009"}
  StartCommandTimeout=80
  StopCommandTimeout=80
  UserName="root"
  ResourceType=1
```

In order to define the Trade3 application resource of class `IBM.Application`, we use the following command.

```
# mkrsrc -f trade3.resourcedef.IBM.Application IBM.Application
```

In order to check the operational state of all defined application resources, we issue the **lsrsrc IBM.Application Name NodeNameList OpState** command as shown in Example 5-11. OpState value of 2 indicates the resource is offline.

```
# 1srsrc IBM.Application Name NodeNameList OpState
Resource Persistent and Dynamic Attributes for IBM.Application
resource 1:
    Name          = "trade3Resource"
    NodeNameList = {"prov009.itsc.austin.ibm.com"}
    OpState       = 2
resource 2:
    Name          = "trade3Resource"
    NodeNameList = {"prov008.itsc.austin.ibm.com"}
    OpState       = 2
resource 3:
    Name          = "trade3Resource"
    NodeNameList =
{"prov008.itsc.austin.ibm.com","prov009.itsc.austin.ibm.com"}
    OpState       = 2
resource 4:
    Name          = "db2Resource"
    NodeNameList = {"prov009.itsc.austin.ibm.com"}
    OpState       = 2
resource 5:
    Name          = "db2Resource"
    NodeNameList = {"prov008.itsc.austin.ibm.com"}
    OpState       = 2
resource 6:
    Name          = "db2Resource"
    NodeNameList =
{"prov008.itsc.austin.ibm.com","prov009.itsc.austin.ibm.com"}
    OpState       = 2
resource 7:
    Name          = "websphereResource"
    NodeNameList = {"prov009.itsc.austin.ibm.com"}
    OpState       = 2
resource 8:
    Name          = "websphereResource"
    NodeNameList = {"prov008.itsc.austin.ibm.com"}
    OpState       = 2
resource 9:
    Name          = "websphereResource"
    NodeNameList =
{"prov008.itsc.austin.ibm.com","prov009.itsc.austin.ibm.com"}
    OpState       = 2
```

Define the network resources

You must create an IP address to represent any server in the first-level automation domain created in this section. In our scenario, the HTTP servers use this IP address to access the IBM WebSphere Application Server in the domain.

The name of this IP address is ServiceIP. The Service IP is active in whichever application resource defined above that is running. The ServiceIP is defined as a floating resource since it can be active at any node in the domain.

A Service IP address is an aggregate resource with one constituent resource per node. It is a floating aggregate resource since it can only have one constituent active at a time.

Note: The ServiceIP is a RSCT resource of type IBM.ServiceIP. We use the IBM.ServiceIP resource class to manage IP addresses that can be started, stopped, and moved between adapters and nodes within a domain.

In order to define the ServiceIP resource, we create a file named WebSphere.resourcedef.IBM.ServiceIP with the definition attributes, shown in Example 5-12. We use this file later as an input parameter for the **mkrsrc** command.

Example 5-12 IBM.ServiceIP resource definition file

```
PersistentResourceAttributes::
  NodeNameList={"prov008","prov009"}
  Name="websphereResourceIP"
  NetMask=255.255.255.0
  IPAddress=9.3.5.28
```

In order to define the ServiceIP, we use the following command.

```
# mkrsrc -f websphere.resourcedef.IBM.ServiceIP IBM.ServiceIP
```

You can verify the IBM.ServiceIP resource definition by using the **lsrsrc** command, as shown in Example 5-13.

Example 5-13 ServiceIP resource

```
# lsrsrc -l IBM.ServiceIP
Resource Persistent Attributes for IBM.ServiceIP
resource 1:
  Name           = "websphereResourceIP"
  ResourceType   = 0
  AggregateResource = "0x2029 0xffff 0xd37b4a18 0x78890a88 0x8f9becb3 0x510e7e81"
  IPAddress      = "9.3.5.28"
  NetMask        = "255.255.255.0"
  ProtectionMode  = 1
  ActivePeerDomain = "was_SA_Domain"
  NodeNameList   = {"prov009.itsc.austin.ibm.com"}
resource 2:
  Name           = "websphereResourceIP"
  ResourceType   = 0
```



```

AggregateResource = "0x2029 0xffff 0xd37b4a18 0x78890a88 0x8f9bebc3 0x510e7e81"
IPAddress         = "9.3.5.28"
NetMask           = "255.255.255.0"
ProtectionMode    = 1
ActivePeerDomain  = "was_SA_Domain"
NodeNameList      = {"prov008.itsc.austin.ibm.com"}
resource 3:
  Name             = "websphereResourceIP"
  ResourceType     = 1
  AggregateResource = "0x3fff 0xffff 0x00000000 0x00000000 0x00000000 0x00000000"
  IPAddress        = "9.3.5.28"
  NetMask          = "255.255.255.0"
  ProtectionMode   = 1
  ActivePeerDomain = "was_SA_Domain"
  NodeNameList     = {"prov008.itsc.austin.ibm.com","prov009.itsc.austin.ibm.com"}

```

In order to check the operational state of the ServiceIP resource, we issue the **lsrsrc** command as shown in Example 5-14. OpState value of 2 indicates the resource is offline.

Example 5-14 ServiceIP OpState

```

# lsrsrc IBM.ServiceIP Name NodeNameList OpState
Resource Persistent and Dynamic Attributes for IBM.ServiceIP
resource 1:
  Name             = "websphereResourceIP"
  NodeNameList     = {"prov009.itsc.austin.ibm.com"}
  OpState          = 2
resource 2:
  Name             = "websphereResourceIP"
  NodeNameList     = {"prov008.itsc.austin.ibm.com"}
  OpState          = 2
resource 3:
  Name             = "websphereResourceIP"
  NodeNameList     = {"prov008.itsc.austin.ibm.com","prov009.itsc.austin.ibm.com"}
  OpState          = 2

```

Create network equivalency

We now have to define which network interfaces can host the ServiceIP defined in the previous section. We accomplish by several steps. One of them is to define a equivalency definition of resources in which we can utilized the ServiceIP. Equivalency, in this case, means that each of the adapters in the equivalency definition can be set up with the ServiceIP.

Note: The RSCT has already collected all of the information on the network adapters of nodes in our domain automatically. These network adapters of nodes in our domain are defined by the class IBM.NetworkInterface.

The equivalency definition for the network adapters uses resources defined by the IBM.NetworkInterface class.

In our scenario, since the IBM WebSphere Application Server must be startable on each node in the domain, the network adapters for each server must be part of the equivalency definition.

In this section, we show a different method for creating a network equivalency than the method described in Chapter 4, “Case study scenario: HTTP Servers on Linux first-level automation domain” on page 53. Here we create a network equivalency using the dynamic select string method.

In order to define a network equivalency using the dynamic select string method, we first issue the `lsrsrc IBM.NetworkInterface` command to determine the name of the CommGroup to which the network interfaces we are interested in belong. See Example 5-15.

Example 5-15 Determining the CommGroup

```
# lsrsrc IBM.NetworkInterface
Resource Persistent Attributes for IBM.NetworkInterface
resource 1:
    Name           = "en0"
    DeviceName      = "ent0"
    IPAddress       = "9.3.5.33"
    SubnetMask      = "255.255.255.0"
    Subnet          = "9.3.5.0"
    CommGroup       = "CG1"
    HeartbeatActive = 1
    Aliases         = {}
    ActivePeerDomain = "was_SA_Domain"
    NodeNameList    = {"prov009.itsc.austin.ibm.com"}
resource 2:
    Name           = "en0"
    DeviceName      = "ent0"
    IPAddress       = "9.3.5.32"
    SubnetMask      = "255.255.255.0"
    Subnet          = "9.3.5.0"
    CommGroup       = "CG1"
    HeartbeatActive = 1
    Aliases         = {}
    ActivePeerDomain = "was_SA_Domain"
```

```
NodeNameList      = {"prov008.itsc.austin.ibm.com"}
```

In order to define the network equivalency named **netequ**, we use the following command, using the CommGroup string collected in Example 5-15.

```
# mkequ -D 'CommGroup=="CG1"' netequ IBM.NetworkInterface
```

You can verify the equivalency definition by using the **lsequ -Ab** command, as shown in Example 5-16.

Example 5-16 Network equivalency status

```
# lsequ -Ab -e netequ
Displaying Equivalency information:
All Attributes
For Equivalency "netequ".

Equivalency 1:
  Name                      = netequ
  MemberClass                = IBM.NetworkInterface
  Resource:Node[Membership]  = {}
  SelectString               = "CommGroup=="CG1""
  SelectFromPolicy           = ANY
  MinimumNecessary           = 1
  Subscription               = {}
  ActivePeerDomain           = was_SA_Domain
  Resource:Node[ValidSelectResources] =
{en0:prov009.itsc.austin.ibm.com,en0:prov008.itsc.austin.ibm.com}
  Resource:Node[InvalidResources] = {}
  ConfigValidity             =
  AutomationDetails[CompoundState] = Automation
```

Note the network interfaces participating in the equivalency are listed in the **Resource:Node[ValidSelectResources]** attribute.

Define resource groups

Now, we need to group the logical resources we defined in previous sections into logical units called *resource groups*. We do this to facilitate the management of resources and the definitions of relationships among these logical groups.

Note: All resources groups we define in this section are RSCT resources groups. IBM Tivoli System Automation for Multiplatforms makes use of these resources groups to perform automation operations.

In our environment, we define a resource group named WebSphererg to group both application and ServiceIP resources defined in “Define the application resources” on page 96, using the **mkrp** command as follows.

```
# mkrp websphererg
```

Once we define the resource group, we add the application resource (websphereResource) and ServiceIP resource (websphereResourceIP) to it using the **addrgmbr** command as follows.

```
# addrgmbr -g websphererg IBM.Application:websphereResource
#
# addrgmbr -g websphererg IBM.Application:db2Resource
#
# addrgmbr -g websphererg IBM.Application:trade3Resource
#
# addrgmbr -g websphererg IBM.ServiceIP:websphereResourceIP
#
```

To verify the membership definitions, use the **lsrg -lm** command as follows in Example 5-17.

Example 5-17 Resource group membership

```
# lsrg -lm
```

Displaying Member Resource information:

Member Resource 1:

```
Class:Resource:Node[ManagedResource] = IBM.Application:trade3Resource
Mandatory                               = True
MemberOf                                = websphererg
OpState                                  = Offline
```

Member Resource 2:

```
Class:Resource:Node[ManagedResource] = IBM.Application:db2Resource
Mandatory                               = True
MemberOf                                = websphererg
OpState                                  = Offline
```

Member Resource 3:

```
Class:Resource:Node[ManagedResource] = IBM.ServiceIP:websphereResourceIP
Mandatory                               = True
MemberOf                                = websphererg
OpState                                  = Offline
```

Member Resource 4:

```
Class:Resource:Node[ManagedResource] = IBM.Application:websphereResource
Mandatory                               = True
MemberOf                                = websphererg
```

You can verify the operational and nominal state of the resource group by issuing the **lsrg** command as follows in Example 5-18.

Example 5-18 Resource group operational state

```
# lsrg -g websphererg
Displaying Resource Group information:
For Resource Group "websphererg".
```

```
Resource Group 1:
  Name                = websphererg
  MemberLocation      = Collocated
  Priority             = 0
  AllowedNode         = ALL
  NominalState       = Offline
  ExcludedList        = {}
  Subscription        = {}
  Owner               =
  Description         =
  InfoLink            =
  ActivePeerDomain    = was_SA_Domain
  OpState            = Offline
  TopGroup            = websphererg
  ConfigValidity      =
  TopGroupNominalState = Offline
```

The objective is to have the resource group operational and nominal states defined as online. Before we can bring the resource group online, we have to define relationships and dependencies to the defined resources.

5.2.3 Create the automation policy using relationship definitions

We define relationships between a resource (named *source resource*) and one or more resources (named *target resources*). For example, resource A must be online so that resource B can start. In this example, the relationship to be defined is a StartAfter relationship. For details and rules about valid relationship definitions, refer to *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04.

Based on the requirements we define in 5.1, “Application server automation domain overview” on page 91 for our case study scenario, we know that both the application resource defined for IBM WebSphere Application Server and the

ServiceIP resource must be started and active on the same node in the domain. The available relationship for this requirement is a *Collocated relationship*.

Also, the ServiceIP resource must be active so that the application resource defined for IBM WebSphere Application Server can be started. We define this as a *StartAfter relationship*.

IBM Tivoli System Automation for Multiplatforms provides a relationship that combines both StartAfter and Collocated requirements described above into one relationship definition: *DependsOn relationship*.

We must define another relationship between the ServiceIP and the interfaces in which the ServiceIP can be defined. We can only set the ServiceIP to an active network interface. Since we have defined an equivalency for the available network interfaces of the nodes in the domain, we must define the relationship between the ServiceIP and the network equivalency.

In order to define the relationships described above, we used the `mkrel` command as follows:

- ▶ The following defines a DependsOn relationship between the websphereResource application resource and the ServiceIP resource name:

```
# mkrel -p DependsOn -S IBM.Application:websphereResource -G  
IBM.ServiceIP:websphereResourceIP websphere_dependson_SIP
```

- ▶ The following defines a DependsOn relationship between the ServiceIP and the network equivalency:

```
# mkrel -p DependsOn -S IBM.ServiceIP:websphereResourceIP -G  
IBM.Equivalency:netequ SIP_dependson_netequ
```

The Trade3 application requires access to a database running on IBM DB2. We provide this access by using a JDBC connection from the IBM WebSphere Application Server. In order for the JDBC access to succeed, IBM DB2 client must be running on the same node as IBM WebSphere Application Server. The available relationship for this requirement is a Collocated relationship. In order to ensure that IBM DB2 is up and running before IBM WebSphere Application Server, we define a DependsOn relationship instead, as follows:

```
# mkrel -p DependsOn -S IBM.Application:websphereResource -G  
IBM.Application:db2Resource websphere_dependson_db2
```

The Trade3 application also requires IBM WebSphere Application Server up and running. In this case, we define a DependsOn relationship as follows:

```
# mkrel -p DependsOn -S IBM.Application:trade3Resource -G  
IBM.Application:websphereResource trade3_dependson_websphere
```

You can verify the relationship definitions using the **lsrel** command. See Example 5-19.

Example 5-19 Managed relationships

```
# lsrel -l
Displaying Managed Relations :

Managed Relationship 1:
  Name                               = websphere_dependson_SIP
  Class:Resource:Node[Source]       = IBM.Application:websphereResource
  ResourceGroup[Source]             = webspherg

Managed Relationship 2:
  Name                               = websphere_dependson_db2
  Class:Resource:Node[Source]       = IBM.Application:websphereResource
  ResourceGroup[Source]             = webspherg

Managed Relationship 3:
  Name                               = trade3_dependson_websphere
  Class:Resource:Node[Source]       = IBM.Application:trade3Resource
  ResourceGroup[Source]             = webspherg

Managed Relationship 4:
  Name                               = SIP_dependson_netequ
  Class:Resource:Node[Source]       = IBM.ServiceIP:websphereResourceIP
  ResourceGroup[Source]             = webspherg
```

Figure 5-3 on page 112 shows the relationship requirements for our environment.

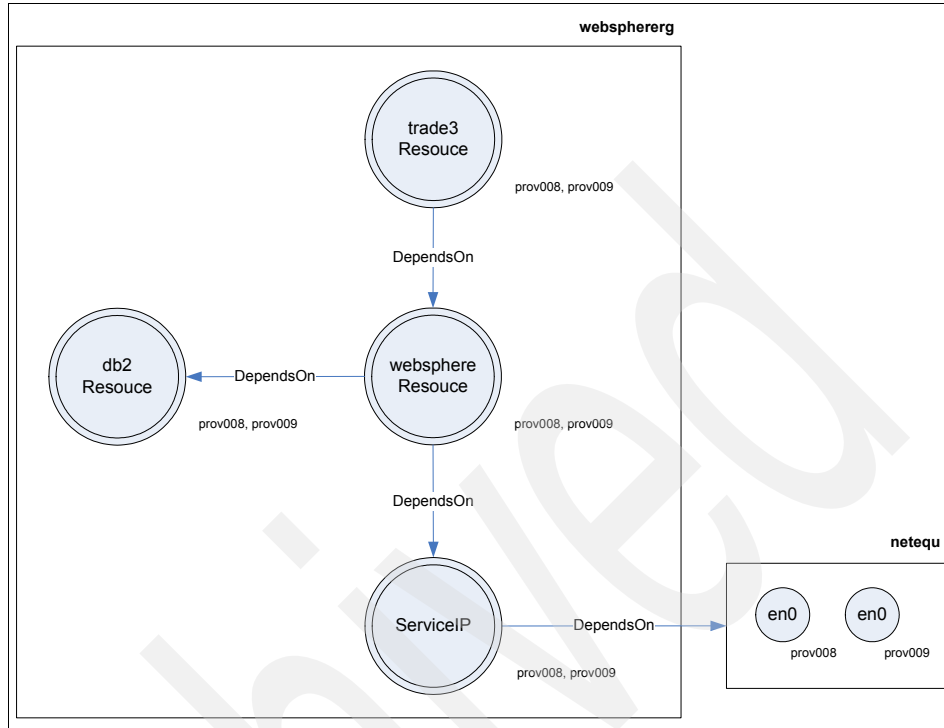


Figure 5-3 Defined relationships for the scenario

5.2.4 Change the Operational State of the resource group

Once we have defined all of our resources into a resource group, all of our resources are made manageable using the *resource group definition*. Initially, When we define resource groups and add members to them, their automation goal is set to offline by default.

Now that all the relationships and dependencies between the resources are properly defined for our environment, we set the resource group to an automation goal of online. This action also changes the operational state of all members of the resource group. This is achieved using the **chrg** command as follows.

```
# chrg -o online webspherg
```

You can check the operational and nominal state of the resource group by issuing the **lsrg** command. Example 5-20 shows a sequence of **lsrg** commands. Note that the first execution shows the Operational State as Pending online. This means all the resources of that resource group are not yet available.


```
# lsrg -g websphererg
```

```
Displaying Resource Group information:  
For Resource Group "websphererg".
```

```
Resource Group 1:
```

```
    Name           = websphererg  
    MemberLocation = Collocated  
    Priority        = 0  
    AllowedNode     = ALL  
    NominalState    = Online  
    ExcludedList    = {}  
    Subscription    = {}  
    Owner           =  
    Description     =  
    InfoLink        =  
    ActivePeerDomain = was_SA_Domain  
    OpState         = Pending online  
    TopGroup        = websphererg  
    ConfigValidity  =  
    TopGroupNominalState = Online
```

```
#
```

```
# lsrg -g websphererg
```

```
Displaying Resource Group information:  
For Resource Group "websphererg".
```

```
Resource Group 1:
```

```
    Name           = websphererg  
    MemberLocation = Collocated  
    Priority        = 0  
    AllowedNode     = ALL  
    NominalState    = Online  
    ExcludedList    = {}  
    Subscription    = {}  
    Owner           =  
    Description     =  
    InfoLink        =  
    ActivePeerDomain = was_SA_Domain  
    OpState         = Online  
    TopGroup        = websphererg  
    ConfigValidity  =  
    TopGroupNominalState = Online
```

Once the resource group has an online OpState, we validate the policy definitions we have done so far. We need to check the operational state of all

defined application resources. An OpState value of 1 indicates the application resource is online on the node. See Example 5-21.

Example 5-21 Application resources OpState online

```
# lsrsrc IBM.Application Name NodeNameList OpState
Resource Persistent and Dynamic Attributes for IBM.Application
resource 1:
    Name          = "trade3Resource"
    NodeNameList = {"prov009.itsc.austin.ibm.com"}
    OpState       = 2
resource 2:
    Name          = "trade3Resource"
    NodeNameList = {"prov008.itsc.austin.ibm.com"}
    OpState       = 1
resource 3:
    Name          = "trade3Resource"
    NodeNameList = {"prov008.itsc.austin.ibm.com","prov009.itsc.austin.ibm.com"}
    OpState       = 1
resource 4:
    Name          = "db2Resource"
    NodeNameList = {"prov009.itsc.austin.ibm.com"}
    OpState       = 2
resource 5:
    Name          = "db2Resource"
    NodeNameList = {"prov008.itsc.austin.ibm.com"}
    OpState       = 1
resource 6:
    Name          = "db2Resource"
    NodeNameList = {"prov008.itsc.austin.ibm.com","prov009.itsc.austin.ibm.com"}
    OpState       = 1
resource 7:
    Name          = "websphereResource"
    NodeNameList = {"prov009.itsc.austin.ibm.com"}
    OpState       = 2
resource 8:
    Name          = "websphereResource"
    NodeNameList = {"prov008.itsc.austin.ibm.com"}
    OpState       = 1
resource 9:
    Name          = "websphereResource"
    NodeNameList = {"prov008.itsc.austin.ibm.com","prov009.itsc.austin.ibm.com"}
    OpState       = 1
```

In Example 5-20, all the Application resources we define in this case study scenario are running on a single node, prov008, according to the defined policy (relationships).

5.2.5 Verify the operational quorum and tie breaker definition

We use the operational quorum to decide if resources can be safely activated without causing conflicts with other resources. The operational quorum is determined based on the number of online nodes and the use of a tie breaker to resolve certain situations. IBM Tivoli System Automation for Multiplatforms is only able to manipulate resources when an operational quorum exists.

In our scenario environment, we have only two nodes in the domain. This represents a problem in case one of the nodes is in a failed state. IBM Tivoli System Automation for Multiplatforms periodically tries to reach each network interface of the nodes in the domain. In our environment, we have a two node domain. In case one interface fails on one of the nodes, the other node will not be able to get a response from the peer node and is flagged offline by IBM Tivoli System Automation for Multiplatforms. In any domain with an even number of nodes, a tie breaker must be defined to decide which remaining nodes are able to run the resources.

In order to obtain the operational quorum for a domain, we use the `lssrc` command as shown in Example 5-22.

Example 5-22 Operational quorum

```
# lssrc -ls IBM.RecoveryRM
Subsystem      : IBM.RecoveryRM
PID            : 28602
Cluster Name   : was_SA_Domain
Node Number    : 1
Daemon start time : Mon Aug 22 18:39:18 CDT 2005

Daemon State:
  My Node Name      : prov008
  Master Node Name  : prov009 (node number = 2)
  Our IVN           : 2.1.0.0
  Our AVN           : 2.1.0.0
  Our CVN           : 81124823583 (0x8430b721f)
  Total Node Count  : 2
  Joined Member Count : 2
  Config Quorum Count : 2
  Startup Quorum Count : 1
Operational Quorum State: HAS_QUORUM
  In Config Quorum   : TRUE
  In Config State    : TRUE
  Replace Config State : FALSE
```

```
Information from malloc about memory use:
Total Space      : 0x007702c0 (7799488)
```

Allocated Space: 0x00719108 (7442696)
Unused Space : 0x0003f4e0 (259296)
Freeable Space : 0x00000000 (0)

Use the IBM.TieBreaker class to configure a tie breaker. There are several types of tie breaker that you can use. Refer to *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04, for details. Even though you can define multiple tie breakers, still only one is activated in a domain at time.

In our scenario, we choose to define a network tie breaker. It will use an external IP address that is available to the nodes, and it is used to resolve a tie situation. We decide to use the default gateway of the subnetwork in which all our nodes belong. Figure 5-4 represents our scenario.

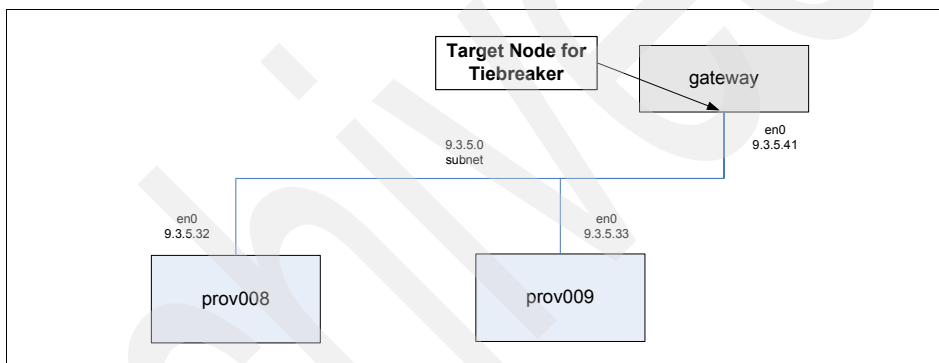


Figure 5-4 Network tie breaker

The definition of the network tie breaker is an RSCT tie breaker type EXEC. For additional information on the valid types of tie breakers, refer to the *IBM Reliable Scalable Cluster Technology Administration Guide*, SA22-7889-07.

Important: To ensure maximum safety, we advise, when using the network tie breaker, that you have all nodes in the cluster on the same subunit, and for the target for the network tie breaker to be the gateway of that subunit. Refer to “Tie breaker” on page 27.

IBM Tivoli System Automation for Multiplatforms provides a script to use as a network tie breaker. The name of the script is **samtb_net** and **samtb_net** is located in the `/usr/sbin/rsct/bin` directory.

In order to define the tie breaker for our environment, we create a file named `websphere.resourcedef.IBM.TieBreaker` with the definition attributes that you see

in Example 5-23. We use this file later as an input parameter for the `mkrsrc` command.

Example 5-23 IBM.TieBreaker resource definition file

```
PersistentResourceAttributes::
  Type=EXEC
  Name="websphereTieBreaker"
  DeviceInfo="PATHNAME=/usr/sbin/rsct/bin/samtb_net Address=9.3.5.41"
  PostReserveWaitTime=30
  HeartbeatPeriod=10
```

In Example 5-23, a tie breaker named `websphereTieBreaker` is created of the type: `EXEC`. The executable script is the network tie breaker `samtb_net` shipped with IBM Tivoli System Automation for Multiplatforms and `Address=9.3.5.41` specifies the IP address of our default gateway.

In order to define the tie breaker, we use the following command.

```
# mkrsrc -f websphere.resourcedef.IBM.TieBreaker IBM.TieBreaker
```

You can verify the tie breaker definition by issuing the `lsrsrc` command as shown in Example 5-24.

Example 5-24 Tie breaker properties

```
# lsrsrc IBM.TieBreaker
Resource Persistent Attributes for IBM.TieBreaker
resource 1:
  Name           = "Fail"
  Type           = "Fail"
  DeviceInfo     = ""
  ReprobeData    = ""
  ReleaseRetryPeriod = 0
  HeartbeatPeriod = 0
  PreReserveWaitTime = 0
  PostReserveWaitTime = 0
  NodeInfo       = {}
  ActivePeerDomain = "was_SA_Domain"
resource 2:
  Name           = "websphereTieBreaker"
  Type           = "EXEC"
  DeviceInfo     = "PATHNAME=/usr/sbin/rsct/bin/samtb_net
Address=9.3.5.41"
  ReprobeData    = ""
  ReleaseRetryPeriod = 0
  HeartbeatPeriod = 10
  PreReserveWaitTime = 0
  PostReserveWaitTime = 30
```

```

        NodeInfo          = {}
        ActivePeerDomain  = "was_SA_Domain"
resource 3:
    Name                  = "Operator"
    Type                  = "Operator"
    DeviceInfo            = ""
    ReprobeData           = ""
    ReleaseRetryPeriod    = 0
    HeartbeatPeriod       = 0
    PreReserveWaitTime    = 0
    PostReserveWaitTime   = 0
    NodeInfo              = {}
    ActivePeerDomain      = "was_SA_Domain"

```

Once we define the tie breaker, we must activate it. We use the **chrsrc** command as follows.

```
# chrsrc -c IBM.PeerNode OpQuorumTieBreaker="websphereTieBreaker"
```

In addition to the tie breaker definition, we need some additional configuration changes so that IBM Tivoli System Automation for Multiplatforms can detect network interface failures. On *every* node in the domain, we must create the following file: `/usr/sbin/cluster/netmon.cf`.

Each line of the `netmon.cf` file should contain a hostname or IP address. It is not necessary to specify which line of the `netmon.cf` file pertains to which network interface; IBM Tivoli System Automation for Multiplatforms automatically uses the IP address or host name that exists in the same subunit as the appropriate network interface. We highly recommend that you use the IP address of the gateway for that network interface.

Example 5-25 shows the `netmon.cf` file we use in our scenario.

Example 5-25 netmon.cf configuration file

```
# Definition required for operational quorum for websphere resource group
9.3.5.41
```

5.3 End-to-end Automation Adapter configuration

This section describes the configuration steps we use in our environment to configure the End-to-end Automation Adapter we use in our case study scenario. We configure the End-to-end Automation Adapter so that our first-level automation domain can be operated by the IBM Tivoli System Automation for

Multiplatforms V2.1 End-to-end Automation Management Component. The End-to-end Automation Adapter in our scenario will:

- ▶ Monitor resources defined at the first-level automation domain
- ▶ Propagate resource attribute changes to the End-to-end Automation Management Component automation manager
- ▶ Manage resources defined at the first-level automation domain from the End-to-end Automation Management Component

Figure 5-5 shows the interaction between our first-level automation domain and the End-to-end Automation Management Component. We discuss the End-to-end Automation Management Component installation and configuration in Chapter 7, “Case study scenario: End-to-end automation domain” on page 201.

There can be only one End-to-end Automation Adapter active in a first-level automation domain. In our scenario, we configure the End-to-end Automation Adapter to be high available. This configuration allows the End-to-end Automation Adapter to run on any node of our automation domain, in case of failure of the node in which the End-to-end Automation Adapter is running.

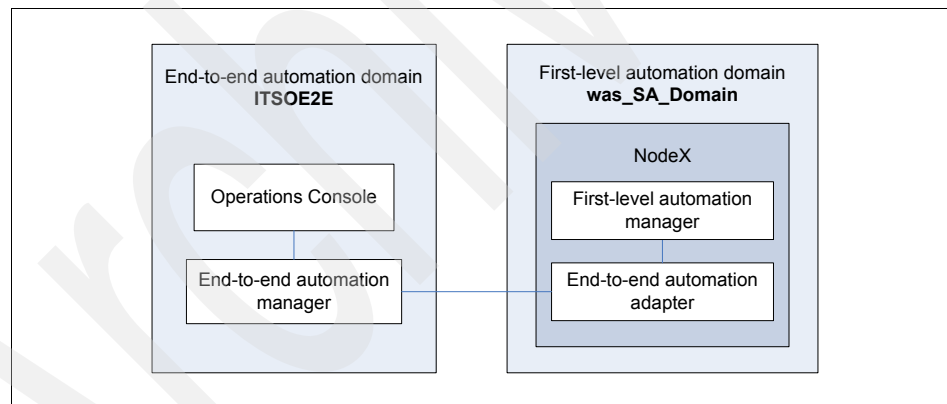


Figure 5-5 End-to-end domain and was_SA_Domain interaction

Important: The first-level domain that we present in this chapter is made up of pSeries® servers running AIX 5.3. The End-to-end Automation Adapter uses the Secure shell (SSH) for communication with the End-to-end Automation Management Component.

Therefore, the installation of both the **openssl** package and the **openssh** package is mandatory on AIX servers. You obtain these packages at the IBM AIX Toolbox for Linux Applications Web site:

https://www14.software.ibm.com/webapp/iwm/web/reg/download.do?source=aixtoolbox&S_PKG=dlaixww&cp=UTF-8

In order to configure the End-to-end Automation Adapter, we use the IBM Tivoli System Automation for Multiplatforms adapter configuration tool to perform the following tasks:

- ▶ Generate End-to-end Automation Adapter configuration files
- ▶ Replicate the End-to-end Automation Adapter configuration files to other servers in the automation domain
- ▶ Define the End-to-end Automation Adapter automation policy

We discuss these tasks in detail in the following sections.

5.3.1 Generate End-to-end Automation Adapter configuration files

In order to use the IBM Tivoli System Automation for Multiplatforms adapter configuration tool, issue the following command: **cfigsamadapter**. This brings up the window you see in Figure 5-6.

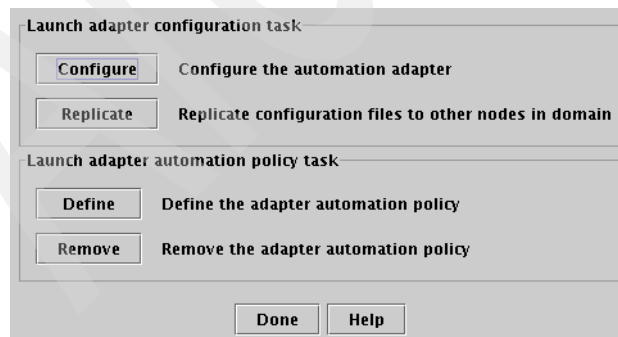


Figure 5-6 End-to-end Automation Adapter configuration tool

You can start the configuration by selecting the **Configure** button. Figure 5-7 displays.

Configuration status: Initial configuration

Adapter Host using adapter Automation Security Logger

Automation adapter host

Host name or IP address 9.3.5.40

Request port number 2001

Event port number 5539

Port used to receive requests from the host using the automation adapter
Port used to receive events from the system automation domain

Advanced...

Save Cancel Help

Figure 5-7 Configuration tool: Adapter tab

The **Adapter** tab allows you to define the IP address. Use this as the ServiceIP for the End-to-end Automation Adapter.

In cases where you configure the adapter to be high available, this tab allows you to define the port numbers on which the End-to-end Automation Adapter listens for requests from the End-to-end Automation Management Component automation manager (Request port number), and the End-to-end Automation Adapter listens for events from the first-level automation manager (Event port number). We use the default values for both Request and Event port numbers.

We also keep the default values defined under the **Advanced** button.

Click **Save**.

Figure 5-8 displays.

Configuration status: Initial configuration

Adapter Host using adapter Automation Security Logger

Host that is using the automation adapter

Select the mode in which the automation adapter is used. Either configure the end-to-end management host that uses the automation adapter to manage a first-level automation domain or configure the operations console that accesses the automation adapter directly.

☒ Configure end-to-end management host

Host name or IP address tsa011.itsc.austin.ibm.com

Event port number 2002 Port used to receive events from the automation adapter

☐ Configure direct access operations console

Host name or IP address

Event port number 2002 Port used to receive events from the automation adapter

Save Cancel Help

Figure 5-8 Configuration tool: Host using adapter tab

Under the **Host using adapter** tab, we define the adapter mode to manage our first-level automation domain. We specified the host name of our End-to-end Automation Management Component automation manager and the port number on which the End-to-end Automation Management Component automation manager listens for events from the End-to-end Automation Adapter. The value for this port number has to match with the port number used during the installation of the End-to-end Automation Management Component. We use the default value 2002 as shown in Figure 5-8.

Click **Save**. Figure 5-9 displays.

Configuration status: Initial configuration

Adapter Host using adapter Automation Security Logger

Automation adapter automation policy

☒ Automate adapter in system automation domain

Query domain Domain name: was_SA_Domain

Defined node	Automated on node	Network interface
prov008	Yes	en0
prov009	Yes	en0

Up Down Add... Remove Change...

Automated resources prefix websphere_E2E_

Adapter IP address 9.3.5.40

Netmask 255.255.255.0

Save Cancel Help

Figure 5-9 End-to-end Automation Adapter configuration tool: Automation tab

The **Automation** tab, shown in Figure 5-9, allows us to define the resources and policy for a high available End-to-end Automation Adapter. Check the **Automate adapter in system automation domain** box. Ensure that the first-level automation domain is online and use the Query button to populate the configuration panel with all the nodes in online status. You can specify which nodes will be able to run the End-to-end Automation Adapter. We define all resources for the automation of this End-to-end Automation Adapter using a prefix (Automated resources prefix). We also specify the IP address and network mask of the ServiceIP.

We keep the default values for the **Security** and **Logger** tabs. By using the **Save** button, we create and change the End-to-end Automation Adapter configuration files. Figure 5-10 shows the configuration files that the End-to-end Automation Adapter of our scenario uses.

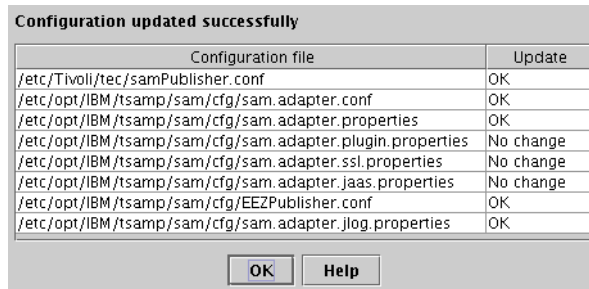


Figure 5-10 End-to-end Automation Adapter configuration files

5.3.2 Replicate the End-to-end Automation Adapter configuration files

Once we properly configure the End-to-end Automation Adapter, we must propagate the configuration files to all the nodes on which the End-to-end Automation Adapter is set to run. The End-to-end Automation Adapter configuration tool provides a way to automatically propagate these files.

Note: The replication tasks utilize the Secure shell (SSH) to propagate the configuration files. Ensure that SSH communication is active between the server on which you are configuring the End-to-end Automation Adapter and all the other nodes in the domain.

In order to replicate the End-to-end Automation Adapter, start the End-to-end Automation Adapter configuration tool and select **Replicate**. See Figure 5-11.

We select to replicate all configuration files to all nodes in the automation domain as shown in Figure 5-11.

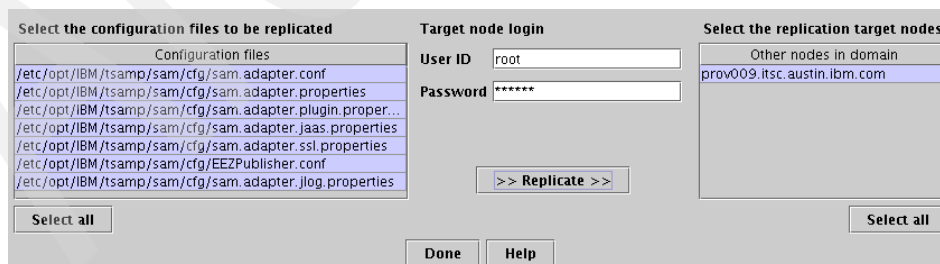


Figure 5-11 End-to-end Automation Adapter configuration tool: Replication

Figure 5-12 shows the replication results in our environment.

Replication completed successfully

Configuration file	Target node	Replication
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.conf	prov009.it...	OK
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.prop...	prov009.it...	OK
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.plugi...	prov009.it...	OK
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.jaas...	prov009.it...	OK
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.ssl.p...	prov009.it...	OK
/etc/opt/IBM/tsamp/sam/cfg/EEZPublisher.conf	prov009.it...	OK
/etc/opt/IBM/tsamp/sam/cfg/sam.adapter.jlog...	prov009.it...	OK
/etc/Tivoli/tec/samPublisher.conf	prov009.it...	OK

OK Help


Figure 5-12 End-to-end Automation Adapter configuration tool: Replication results

5.3.3 Define the End-to-end Automation Adapter automation policy

Now we are ready to create the resources used by IBM Tivoli System Automation for Multiplatforms to automate the End-to-end Automation Adapter. We do this by using the End-to-end Automation Adapter configuration tool.

Issue the `cfigsamadapter` command and select **Define**. Resources are defined in the automation domain using the prefix defined during the End-to-end Automation Adapter configuration, shown in Figure 5-9 on page 123.

Figure 5-13 shows the results of the resources definition tasks for our environment.



Command `"/opt/IBM/tsamp/sam/bin/mksamadapter -p"` to activate the adapter automation policy completed successfully.
The command output is:
Perform commands...

samadapter is not running on prov008
Attention:
If resources matching `'websphere_E2E.*'` exist, they will be removed.

-e successfully performed: `'mkrg websphere_E2E_rg'`
Generated definitions: `"/etc/opt/IBM/tsamp/sam/cfg/websphere_E2E_def"`
-e successfully performed: `'mksrc -f /etc/opt/IBM/tsamp/sam/cfg/websphere_E2E_def IBM.Application'`
-e successfully performed: `'addrgmbr -m T -g websphere_E2E_rg IBM.Application:websphere_E2E.'`
Generated definitions: `"/etc/opt/IBM/tsamp/sam/cfg/websphere_E2E_ip.def"`
-e successfully performed: `'mksrc -f /etc/opt/IBM/tsamp/sam/cfg/websphere_E2E_ip.def IBM.ServiceIP'`
-e successfully performed: `'addrgmbr -m T -g websphere_E2E_rg IBM.ServiceIP:websphere_E2E_ip'`
-e successfully performed: `'mkrel -S IBM.Application:websphere_E2E_ -G IBM.ServiceIP:websphere_E2E_ip -p DependsOn websphere_E2E_on-ip'`
-e successfully performed: `'mkequ websphere_E2E_nieq IBM.NetworkInterface:en0:prov008,en0:prov009'`
-e successfully performed: `'mkrel -S IBM.ServiceIP:websphere_E2E_ip -G IBM.Equivalency:websphere_E2E_nieq -p DependsOn websphere_E2E_ip-on-nieq'`

Resource group `'websphere_E2E_rg'`, resources and relationships were removed and created.

OK

Figure 5-13 End-to-end Automation Adapter configuration tool: Defining policies

Table 5-1 presents the resources defined in our environment.

Table 5-1 End-to-end Automation Adapter automation resources

Resource Name	Resource Class	Comment
websphere_E2E_rg	IBM.ResourceGroup	All defined resources are managed using this resource group.
websphere_E2E_	IBM.Application	The application representing the End-to-end Automation Adapter.
websphere_E2E_ip	IBM.ServiceIP	The ServiceIP on which the End-to-end Automation Adapter runs.
websphere_E2E_nieq	IBM.Equivalency	Equivalency definition for all the network interfaces on which the ServiceIP will be defined.
websphere_E2E_on-ip	IBM.ManagedRelationship	The WebSphere_E2E_ application resource depends on the ServiceIP.
websphere_E2E_ip-on-nieq	IBM.ManagedRelationship	The ServiceIP depends on the network equivalency.

Once all resources are defined, the resource group websphere_E2E_rg status is offline. Issue the **chrg** command to bring the resources defined in the resource group online as follows:

```
# chrg -o online websphere_E2E_rg
```

Verify the status of the resources defined in the resource group using the **lsrg** command as shown in Example 5-26.

Example 5-26 End-to-end Automation Adapter resources status

```
# lsrg -m
Displaying Member Resource information:
Class:Resource:Node[ManagedResource] Mandatory MemberOf OpState
IBM.ServiceIP:websphere_E2E_ip True websphere_E2E_rg Online
IBM.Application:websphere_E2E_ True websphere_E2E_rg Online
IBM.Application:trade3Resource True websphererg Online
IBM.Application:db2Resource True websphererg Online
```

IBM.ServiceIP:websphereResourceIP	True	websphererg	Online
IBM.Application:websphereResource	True	websphererg	Online

5.4 Maintaining defined policies

One new feature of IBM Tivoli System Automation for Multiplatforms V2.1 is the ability to maintain all defined policies to ensure that you have defined policies properly and the flexibility of knowing you can recreate them at any time.

As described in Chapter 1, “IBM Tivoli System Automation for Multiplatforms V2.1” on page 3, you can create or restore policies using expressions in XML format. IBM Tivoli System Automation for Multiplatforms provides the **sampolicy** command, a policy management tool that you can use to activate, backup, restore, and replace policies.

In this section, we use the **sampolicy** command to back up all policies defined in our environment for the was_SA_Domain automation domain as presented in Example 5-27.

Example 5-27 Backing up defined policies

```
# sampolicy -s /usr/local/IBM/TSA/scripts/was_SA_Domain_policies.xml

.samadapter version: driver: 0.2.2.053102
.....The current policy has been saved to file
/usr/local/IBM/TSA/scripts/was_SA_Domain_policies.xml.
```

The file was_SA_Domain_policies.xml created above will contain all policies for our automation domain defined using expressions in XML format. You can use this file to expand or recreate our IBM Tivoli System Automation for Multiplatforms automation domain.

The following example, Example 5-28, shows a resource group definition extracted from the was_SA_Domain_policies.xml file created above.

Example 5-28 Resource group XML definition

```
<ResourceGroup name="websphererg" class="IBM.ResourceGroup">
  <DesiredState>Online</DesiredState>
  <Members>
    <MoveGroup name="websphereResource" class="IBM.Application"
      selectFromPolicy="Ordered" mandatory="True"/>
    <MoveGroup name="db2Resource" class="IBM.Application"
      selectFromPolicy="Ordered" mandatory="True"/>
    <MoveGroup name="trade3Resource" class="IBM.Application"
      selectFromPolicy="Ordered" mandatory="True"/>
  </Members>
</ResourceGroup>
```

```
<MoveGroup name="websphereResourceIP" class="IBM.ServiceIP"
  selectFromPolicy="Ordered" mandatory="True"/>
</Members>
<MemberLocation>Collocated</MemberLocation>
<Priority>0</Priority>
<AllowedNode>ALL</AllowedNode>
</ResourceGroup>
```

Case study scenario: IBM DB2 on z/OS first-level automation domain

In this chapter, we provide the steps to provide high availability and automation to the database environment of the sample application we use in our case study scenario. We do this by creating and configuring an IBM Tivoli System Automation for z/OS V3.1 first-level automation domain. In addition, we set up the End-to-end Automation Adapter so that IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component can manage this first-level automation domain.

We discuss the following topics:

- ▶ “IBM DB2 on z/OS automation domain overview” on page 131
- ▶ “IBM DB2 on z/OS automation domain configuration” on page 131
- ▶ “Configuring automation policies for IBM DB2” on page 145
- ▶ “End-to-end Automation Adapter configuration” on page 181

Figure 6-1 shows the portion of the entire case study scenario we cover in this chapter.

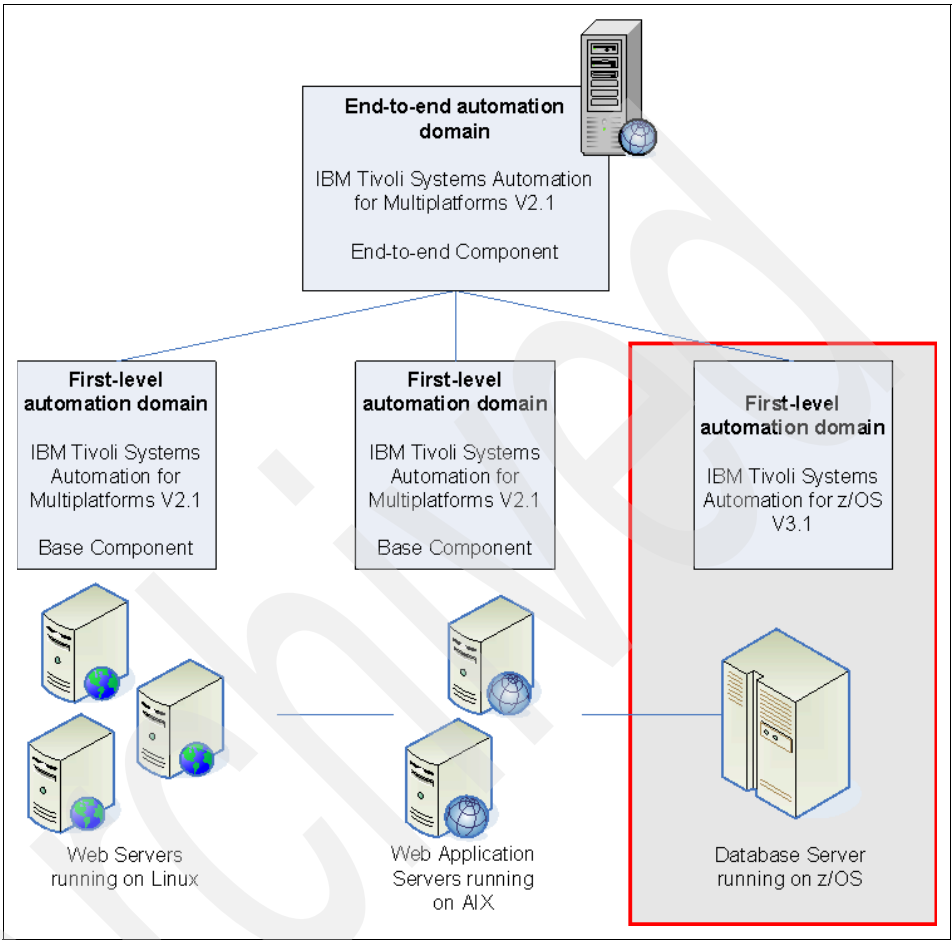


Figure 6-1 IBM DB2 on z/OS first-level automation domain

6.1 IBM DB2 on z/OS automation domain overview

We base this chapter on a case study scenario detailed in Chapter 3, “Case study scenario overview” on page 47 and we aim to achieve a fully automated database environment provided by IBM DB2 on z/OS for our sample application.

We assume the following scenario specific tasks have already been performed:

- ▶ z/OS Version 1.6 is fully functional, including VTAM, JES, TSO, and TCP/IP configured, up, and running.
- ▶ IBM Tivoli NetView for z/OS Version 5.1 is installed, up, and running.
- ▶ IBM DB2 environment for our sample application is fully operational.

In order to achieve our goal, we must perform the following major tasks:

- ▶ IBM Tivoli System Automation for z/OS V3.1 installation and configuration.

We performed the SMP/E installation procedure without any problems following the steps described in the *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual. We also followed instructions provided by the IBM Tivoli System Automation for z/OS V3.1 Program Directory, which shipped with the installation media of IBM Tivoli System Automation for z/OS V3.1.

- ▶ IBM Tivoli NetView for z/OS configurations for IBM Tivoli System Automation for z/OS V3.1.

We describe these configuration tasks in the “IBM DB2 on z/OS automation domain configuration” on page 131.

- ▶ Define and configure automation policies for IBM DB2.

We describe these configuration tasks in the “Configuring automation policies for IBM DB2” on page 145.

6.2 IBM DB2 on z/OS automation domain configuration

In order to implement IBM DB2 high availability and automation with IBM Tivoli System Automation for z/OS V3.1, we must install and customize IBM Tivoli NetView for z/OS. In this chapter, we explain the steps needed to customize IBM Tivoli System Automation for z/OS V3.1 and IBM Tivoli NetView for z/OS data sets to get IBM Tivoli NetView for z/OS active and IBM Tivoli System Automation for z/OS V3.1 prepared for automating our database environment.

For the purposes of this IBM Redbook, we assume that most z/OS installations run IBM Tivoli NetView for z/OS (NetView). In our case study scenario, we only have one NetView installation which is used for IBM Tivoli System Automation for

z/OS V3.1. The NetView domain ID used in our example is defined as **SC64N** and the system name Host of the z/OS system that NetView runs on is SC64TS.

We use system cloning techniques. System cloning enables the use of variables or symbolics in procedures, data sets, VTAM majnode definitions, and so forth. This allows for one procedure defined in a common proclib for execution on multiple systems. For further details about the cloning techniques, refer to *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual.

Preparing our IBM Tivoli System Automation for z/OS V3.1 environment for IBM DB2 automation requires the following configuration steps:

- ▶ “Configure NetView for IBM Tivoli System Automation for z/OS” on page 132
- ▶ “Automate NetView startup procedure” on page 134
- ▶ “Allocate System-Unique data sets” on page 134
- ▶ “Configure the Automation Manager” on page 135
- ▶ “Allocate data sets for the ISPF customization dialog” on page 136
- ▶ “Update PARMLIB data sets” on page 136
- ▶ “Update PROCLIB data sets” on page 137
- ▶ “Define the base automation policy” on page 144

6.2.1 Configure NetView for IBM Tivoli System Automation for z/OS

In order to implement IBM Tivoli System Automation for z/OS V3.1, we must install and customize NetView. In this section, we give an overview of the steps required to customize NetView and IBM Tivoli System Automation for z/OS V3.1 basic implementation in our environment.

Note: If you are running two NetView programs on the same system, refer to the *Tivoli NetView for z/OS Installation: Configuring Additional Components*, SC31-8874, manual.

VTAM Major Node Definition

For details about VTAM major node names and definitions, refer to *Tivoli NetView for z/OS V5.1 Installation: Getting Started*, SC31-8872, manual, Appendix B.

Example 6-1 shows the definition of VTAM major node in our environment.

Example 6-1 VTAM major node definition

```
*****
* NETVIEW MAIN TASK *
*****
&SYSNAME.A APPL AUTH=(VPACE,ACQ,PASS),PRTCT=&SYSNAME.A, X
              MODETAB=AMODETAB,DLOGMOD=DSIL6MOD, X
              APPC=YES,PARSESS=YES, X
              DMINWNL=4,DMINWNR=4,DSESLIM=8,VPACING=10, X
              AUTOSSES=2
*
              STATOPT='NETVIEW'
*****
* NETVIEW PRIMARY POI - (PROGRAM OPERATOR INTERFACE) *
*****
&SYSNAME.APPT APPL AUTH=(NVPACE,SPO),PRTCT=&SYSNAME.A,EAS=1, X
              MODETAB=AMODETAB,DLOGMOD=DSILGMOD
*
              STATOPT='NETVIEW PPT'
*****
* NETWORK MANAGEMENT PRODUCTS (NETVIEW) *
*****
&SYSNAME.ALUC APPL AUTH=ACQ,PARSESS=YES,MODETAB=AMODETAB, X
              DLOGMOD=DSINLDM, X
              PRTCT=&SYSNAME.A
*
              NGFINC=OMIT
*
              STATOPT='LUC TASK'
&SYSNAME.ASPT APPL AUTH=(SPO),EAS=5
*
              NGFINC=OMIT
*
              STATOPT='STATMON-VTAM'
*****
```

Customize NetView DSIPARM data set

We must change these members in the DSIPARM data set.

- ▶ CNMSTYLE/CxxSTGEN
- ▶ DSIDNMK
- ▶ AOFMSGSY
- ▶ INGMMSG01
- ▶ INGXINIT
- ▶ DSICMSYS/AOFCMDSO

The *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261-00, manual provides instructions about what you need to change in the above members. For our case study environment, we followed the referenced manual with no problems.

6.2.2 Automate NetView startup procedure

Add commands to the COMMNDxx member of ing.PARMLIB to start NetView automatically when z/OS starts. You may also need to modify an IEASYSxx member of ing.PARMLIB to specify which COMMNDxx or other PARMLIB members to use during z/OS IPL.

IBM Tivoli System Automation for z/OS V3.1 initialization becomes automated and begins with starting System Operations. You accomplish this by making changes to the ing.PARMLIB data set member COMMNDxx. Make sure that the procedure names you choose match those specified in the ing.PROCLIB data set. Compare the contents of the COMMNDxx member with the INGECON member, which resides in the SINGSAMP sample library.

Example 6-2 shows the COMMNDxx member of our case study scenario.

Example 6-2 COMMAND64 member

```
COM='SET MPF=00'  
COM='MN JOBNAME,T'  
COM='K M,AMRF=Y'  
COM='K S,DEL=RD,SEG=20,CON=N,RNUM=20,RTME=001,MFORM=M,L=01'  
COM='S AOFASSI,SUB=MSTR'  
COM='S AOFAPPL'  
COM='S HSAMSC64,SUB=MSTR'  
COM='S JES2'
```

6.2.3 Allocate System-Unique data sets

The following data sets are required several times across the focal point and the target systems. To allocate these data sets, sample jobs are provided in the following members of the SINGSAMP data set. These jobs must be run on every system where the data sets are required. These data sets are:

- ▶ INGALLC1 - HCD trace file for I/O OPS
- ▶ INGALLC2 - Automation status file
- ▶ INGALLC3 - All Automation Managers data sets
- ▶ INGALLC4 - IPL data collection

Before you run these jobs, you need to edit them as per your environment specifications. The value that you fill in may vary from system to system.

Note: There may be a requirement to allocate additional data sets for your enterprise. For more information about how to allocate these data sets, refer to *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261.

6.2.4 Configure the Automation Manager

In order to configure the Automation Manager, we need to customize the PARMLIB member HSAPRMxx.

The HSAPRMxx PARMLIB member contains information required for the initialization of the automation manager and default values for other operational parameters. The member is designed for common use by all automation manager instances within the automation subplex.

Alternatively, you can put the automation manager PARMLIB member in any partitioned data set. See Example 6-3. Then, you need to insert a statement HSAPLIB DD into the automation manager startup procedure member which refers to this partitioned data set. A sample member called HSAPRM00 is provided in the SINGSAMP sample library. You must copy this sample into your PARMLIB of the automation manager (DD name HSAPLIB) when you allocate this data set as described.

Example 6-3 Case scenario PARMLIB HSAPRM00

```
*****
* Name....: HSAPRM00                                     *
* Function: SA z/OS default PARMLIB member for Automation manager *
* Notes...: For a detailed description of each keyword see       *
*           SA z/OS Planning and Installation                   *
*****
*
DELAY=0
*
*GRPID=id
*   Only if you plan to use multiple "logical" sysplex configurations
*   in your real sysplex environment
*
TAKEOVERFILE=SA310USR.VSAM.TAKEOVER
*
*MQM=subn
COMM=XCF
*
PROMPT=NO
*
*LOGSTREAM=NO
```

2@09C

```
*          NO - Indicates that the automation manager does not connect
*          to the System Logger
BUILDTIMEOUT=180
*
CFGDSN=SA310USR.SAV3R1.ACF
*
STOPDELAY=30
```

6.2.5 Allocate data sets for the ISPF customization dialog

When performing this step, we use the sample job INGEDLGA in SINGSAMP to allocate data sets required for the I/O operations and the customization dialog.

These data sets are normally allocated only on the focal point system, where you use the customization dialog.

You must include these data sets for z/OS system, processor, and I/O operations as follows:

Systems operations:

ING.CUSTOM.AOFTABL	ISPF customization table for customization dialog
ING.CUSTOM.SOCNTL	System operations control file

Processor operations:

ING.CUSTOM.AOFTABL	ISPF customization table for customization dialog
ING.CUSTOM.POCNTL	Processor operations control file
ING.CUSTOM.POLOG	Processor operations control file log

I/O operations:

ING.CUSTOM.IHVCONF	I/O operations configuration file
--------------------	-----------------------------------

For further information about how to Install and customize the ISPF Dialog Panels, refer to the *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual.

6.2.6 Update PARMLIB data sets

We modify the following members of the PARMLIB in our environment:

- ▶ PROGxx
- ▶ MPFLSTxx

Refer to the *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual for additional PARMLIB members that you need to update in your environment.

Update PROGxx

You can define the following authorized libraries in a PROGxx member. This dynamically authorizes the program facility called APF. Once you have updated PROGxx, you can issue the command SET PROG=xx to dynamically add without having to IPL the system. We change our PROGxx member to include:

- ▶ ING.SINGMOD1
- ▶ ING.SINGMOD2
- ▶ ING.SINGMOD3

Update MPFLSTxx

We recommend that you update the MPFLSTxx member after you install the ISPF Customization Dialog.

Using the customization dialog, you can define the automation policy and create a list of messages involved in automation. The customization dialog also allows you to define header and trailer lines for the message list, therefore building a complete MPFLSTxx member called MPFLSTSA. The MPFLSTSA is built after you have built the Policy Data Base (PDB™). Alternatively, you can update the contents of the MPFLSTxx member manually using the INGEMPF member. This sample member resides in the SINGSAMP sample library that ships with IBM Tivoli System Automation for z/OS V3.1. Edit the MPFLSTxx member so that it includes all the statements in the INGEMPF member.

This adds the IBM Tivoli System Automation for z/OS V3.1 message automation and console display suppression specifications to the MPFLSTxx member. The AUTO(YES) in the NO_ENTRY statement is required to gather all unknown WTORs. If you ensure that the unknown WTORs are routed to automation via the general MPF exit IEAVMXIT and you have all messages that are specified in the NetView message automation table also specified in the MPF with AUTO(YES), you can specify AUTO(NO) for the NO_ENTRY statement.

In our case study scenario, we use only the following prefixes of messages:

- ▶ AOF*,SUP(NO),AUTO(YES)
- ▶ DSN*,SUP(NO),AUTO(YES)
- ▶ IEF*,SUP(NO),AUTO(YES)

6.2.7 Update PROCLIB data sets

We modified the following startup procedures in our case study environment.

- ▶ Systems operations (SYSOPS)
- ▶ Automation Manager
- ▶ NetView subsystem Interface
- ▶ NetView subsystem Agent

System Operations (SYSOPS)

You may require changes to the startup procedure members in the PROCLIB data set. We recommend that you either back up the startup procedure members that you are going to change or that you create new members. If you are implementing IBM Tivoli System Automation for z/OS V3.1 for the first time, then this is not relevant.

We copied the following members from the SINGSAMP data set to members of our PROCLIB and followed the customization instructions that are contained within these members:

- ▶ INGPICU
- ▶ INGPHOM
- ▶ INGPICL
- ▶ HSAIPLC

Additional details can be found in the detail in the *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual.

Start up procedure for Automation Manager

We copied sample startup procedures called INGEAMSA from the data set ing.SINGSAMP into our PROCLIB. All the required data sets were allocated there.

We include the procedure that runs the Automation Manager into the PROCLIB. A separate NON-APF authorized task library is required in addition to the authorized steplib. See Example 6-4 of the procedure for the Automation Manager we use in this case study scenario.

This is described in more detail in the *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual.

Example 6-4 shows the startup procedure of the Automation Manager we use in our environment.

Example 6-4 Case scenario Automation Manager startup procedure

```
//*****
//* Name....: INGEAMSA
//INGEAMSA PROC TYPE=COLD,           Start type (HOT | WARM | COLD)
//                                D=0,           Default start delay is none
```

```

//          M=00,                Default parmlib suffix is 00
//          P=NO,                Default is NO prompting
//          HLQINST='SA310',     SMP/E installed target lib
//          HLQV='SA310USR.VSAM', HLQ of shared and unique AM
//          HLQ3='SA310USR.V3R1', HLQ of shared and unique AM
//          SLQ='SC64N',         2nd-level qualifier for unique
//          HLQCLB='CBC',        Default C/C++ library HLQ
//          HLQCEE='CEE'         Default LE/390 library HLQ
//*****
//AMSA      EXEC PGM=HSAPINIT,REGION=200M,TIME=1440,                      X
//          PARM='MEMBER=&M,START=&TYPE,DELAY=&D,PROMPT=&P'
//*------
/** STEPLIB - APF Authorized libraries
/**      Remove this statement if SINGMOD1 is part of
/**      LNKLSTxx concatenation
/**-----
//STEPLIB DD DSN=&HLQINST..SINGMOD1,DISP=SHR
//*------
/** HSAMODLE - Non-APF Authorized library concatenation (TASKLIB)
/**-----
//HSAMODLE DD DSN=&HLQINST..SINGMOD1,DISP=SHR      SA z/OS
//          DD DSN=&HLQCLB..SCLBDLL,DISP=SHR      C/C++
//          DD DSN=&HLQCEE..SCEERUN,DISP=SHR      LE/390
//*------
/** HSAOVR - Required schedule override file (shared)
/**-----
//HSAOVR DD DSN=&HLQV..HSAAMOVR,DISP=SHR
//*------
/** HSACFGIN - Required file that saves AM warm start info (shared)
/**      Note: This is NOT the Automation Control File
/**-----
//HSACFGIN DD DSN=&HLQ3..SHSACFG0,DISP=SHR
//*------
/** HSAPLIB - Parameter library containing HSAPRMxx member
/**-----
//HSAPLIB DD DSN=&HLQ3..PARMLIB,DISP=SHR
//*------
/** SYSOUT DATASET used by LE
/**      DISP=OLD prevents duplicate start of this proc
/**-----
//SYSOUT DD DSN=&HLQ3..&SLQ..SYSOUT,DISP=OLD
//*SYSOUT DD SYSOUT=*
//SYSPRINT DD DUMMY,SPACE=(TRK,(2,200))
//*------
/** CEEDUMP - is used by the Language Environment Dump Services.
/**      CEEDUMP must be a sequential data set.
/**      See below for the recommended size of the data set.
/**-----
//CEEDUMP DD DUMMY,SPACE=(TRK,(30,100)),

```

```
//          DCB=(RECFM=FB,LRECL=133,BLKSIZE=13330)
//*-----
//* TRACE DATASETS. Uncomment in case of needed trace information
//*-----
//TRACETO DD DSN=&HLQ3..&SLQ..TRACETO,DISP=SHR
//TRACET1 DD DSN=&HLQ3..&SLQ..TRACET1,DISP=SHR
```

The Automation Manager must be started *cold* on the first startup. The cold start is performed by default, unless you specify the warm start option. Do not select the warm start option, because you might have policy data from earlier releases in your warm start cache. In Example 6-5, you also see that the ACF load had completed successfully. The following message should appear on the system console. See Example 6-5.

Example 6-5 Case scenario Automation Manager start

```
1 J E S 2   J O B   L O G   --   S Y S T E M   S C 6 4   --   N O D E   W T S C P L X 2
0
20.35.44 STC03420 ---- THURSDAY, 25 AUG 2005 ----
20.35.44 STC03420 IEF695I START HSAMSC64 WITH JOBNAME HSAMSC64 IS ASSIGNED TO
USER STC      , GROUP SYS1
20.35.44 STC03420 $HASP373 HSAMSC64 STARTED
20.35.44 STC03420 IEF403I HSAMSC64 - STARTED - TIME=20.35.44 - ASID=007C -
SC64
20.35.46 STC03420 IEF761I HSAMSC64 HSAMSC64 HSAPLIB HSAPSPLM/A DD IS ALREADY
ALLOCATED AND WILL BE USED BY THIS TASK.
20.35.46 STC03420 IEE252I MEMBER HSAPRM00 FOUND IN SA310USR.V3R1.PARMLIB
20.35.47 STC03420 HSAM1003I ARM REGISTER WAS SUCCESSFUL FOR
ELEMENT=HSAAM_SC64$$$$1 TYPE=HSAMGR RESTART=NO.
20.35.47 STC03420 HSAM1005I COMMUNICATION IS XCF ONLY. THE TAKEOVER FILE WILL
BE USED FOR HOT START AND RECOVERY PURPOSES.
20.35.50 STC03420 HSAM1000I AUTOMATION MANAGER SUBTASKS ARE BEING STARTED.
20.35.51 STC03420 HSAM1315I SUCCESSFULL ALLOCATION OF TAKEOVER FILE
SA310USR.VSAM.TAKEOVER.
20.35.51 STC03420 +HSAL1083I AUTOMATION MANAGER HAS CREATED THE INITIAL POSIX
THREAD WITH PID=84017230.
20.35.51 STC03420 +HSAL1107I REFRESH OF TAKEOVER FILE
DSN=SA310USR.VSAM.TAKEOVER WITH CURRENT AUTOMATION MANAGER DATA STARTED.
20.35.52 STC03420 +HSAL1108I REFRESH OF TAKEOVER FILE
DSN=SA310USR.VSAM.TAKEOVER COMPLETED.
20.36.02 STC03420 HSAM1308I SA z/OS PRIMARY AUTOMATION MANAGER INITIALIZATION
COMPLETE, TYPE=COLD.
20.42.13 STC03420 HSAM1330I LOAD_ACF REQUEST COMPLETED SUCCESSFULLY ON SC64.
```

NetView subsystem Interface startup

NetView provides a sample subsystem Interface startup procedure in member CNMSJ010. We copy and adapt this member from our NetView library to suit our environment.

Also, the INGENVSA member of the SINGSAMP sample library contains a sample startup procedure. This is new to IBM Tivoli System Automation for z/OS V3.1.

The following are recommendations for NetView subsystem interface startup:

- ▶ Ensure that the edited copy of the NetView application startup procedure contains the appropriate data set concatenation order.
- ▶ Verify that your new NetView application startup procedure member uses the correct data set names for your system.
- ▶ Rename the NetView application startup procedure member to agree with the four-character prefix defined in the IEFSSNxx PARMLIB member as per *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual guidelines.
- ▶ You must define the Program to Program Interface (PPI). Refer to IBM instructions in the supplied sample subsystem Interface startup procedure (CNMSJ010). In our case study scenario environment, we call the subsystem Interface application AOFASSI.

Note: Add your `ing.SINGMOD1` library and the NetView CNMLINK library containing CNMPSSI to the steplib. Alternately, you may add these libraries to LINKLST. You should have already APF-authorized these libraries. See Example 6-6 of the AOFASSI application.

Example 6-6 Our Netview subsystem interface (AOFASSI)

```
//CNMPSSI PROC SQ1='NETVIEW.V5R1M0', ** SYSTEM DSN HIGH LEVEL QUALIFIER
//      PROG=CNMINIT,          ** PGM USED TO START NETVIEW SUBSYSTEM
//      REG=1250,              ** REGION SIZE(IN K)
//      MBUF=4000,             ** NUMBER OF MESSAGE BUFFERS TO USE
//      CBUF=200,              ** NUMBER OF COMMAND BUFFERS TO USE
//      DSIG='',               ** Subsystem command designator
//      MSGIFAC='SYSTEM',      ** SSI/EXTENDED CONSOLE OVERRIDE SWITCH
//      PPIOPT='PPI',          ** PPI OPTIONS SWITCH
//      ARM='*NOARM',          ** AUTOMATIC RESTART (ARM) USAGE
//      PFXREG='ONE',          ** Prefix Registration option.
//      P256BUF=300,           ** Number of 256 byte PPI buffers to use
//      P4000BUF=0             ** Number of 4000 byte PPI buffers to use
//NETVIEW EXEC PGM=&PROG,TIME=1440,REGION=&REG.K,
//      PARM=(&MBUF,&CBUF,'&DSIG','&MSGIFAC','&PPIOPT','&ARM',
```

```
//          'PFXREG',&P256BUF,&P4000BUF),DPRTY=(13,13)
//STEPLIB DD DSN=&SQ1..CNMLINK,DISP=SHR
```

We issue the **MVS S AOFASSI** command to start the AOFASSI PPI. Example 6-7 shows the initialization message.

Example 6-7 AOFASSI Initialization Message

```
J E S 2   J O B   L O G   --   S Y S T E M   S C 6 4   --   N O D E

18.58.24 STC03409 ---- THURSDAY, 25 AUG 2005 ----
18.58.24 STC03409 IEF695I START AOFASSI WITH JOBNAME AOFASSI IS ASSIGNED TO
U
18.58.24 STC03409 $HASP373 AOFASSI STARTED
18.58.24 STC03409 IEF403I AOFASSI - STARTED - TIME=18.58.24 - ASID=009F - SC64
18.58.24 STC03409 CNM226I NETVIEW PROGRAM TO PROGRAM INTERFACE INITIALIZATION
18.58.24 STC03409 CNM541I NETVIEW SUBSYSTEM INITIALIZED SUCCESSFULLY
```

NetView subsystem Agent startup

We configure the NetView subsystem Agent startup procedure to suit our environment as presented in Example 6-8. In our case study scenario environment, the subsystem Agent is AOFAPPL.

Example 6-8 NetView startup procedure (Agent)

```
//INGENVSA PROC DOMAIN=&SYSNAME.N, ** NETVIEW DOMAIN NAME
//      PROG=DSIMNT,          ** PGM FOR AUTOMATION NETVIEW
//      Q1='NETVIEW.V5R1USER',** USER DSN HIGH LEVEL QUALIFIER
//      SQ1='NETV',           ** NETVIEW DSN HIGH LVL QUALIFIER
//      SQ2=ING,              ** SA z/OS DSN HIGH LVL QUALIFIER
//      VQ2=ING,              ** SA z/OS DSN HIGH LVL QUALIFIER-VSAM
//      EQQ=EQQ,              ** OPC/TWS DSN HIGH LVL QUALIFIER
//      CPSM='ing.CPSM',      ** CPSM DSN HIGH LVL QUALIFIER
//      VQ1='NETVIEW.V5R1M0', ** VSAM DSN HIGH LVL QUALIFIER
//      REG=0,                ** REGION SIZE(IN M) FOR NetView
//      SUBSYM='',            ** SYMBOLIC SUBSTITUTION SWITCH
//      NV2I=''
//NETVIEW EXEC PGM=&PROG,TIME=1440,
//      REGION=&REG.M,
//      PARM=(24K,200,
//      '&DOMAIN',' ',' ','&SUBSYM','&NV2I'),
//      DPRTY=(13,13)
//DSICLD DD DSN=&SQ2..SINGNREX,DISP=SHR          SA TARGET
//      DD DSN=&SQ1..CNMCLST,DISP=SHR            NETVIEW TARGET
//      DD DSN=&SQ1..CNMSAMP,DISP=SHR            NETVIEW SAMPLIB
//DSIOPEN DD DSN=&SQ1..SDSIOPEN,DISP=SHR
//DSIPARM DD DSN=&Q1..&DOMAIN..DSIPARM,DISP=SHR  USER TARGET
//      DD DSN=&Q1..DSIPARM,DISP=SHR            USER TARGET
```

```

//      DD   DSN=&SQ2..SINGNPRM,DISP=SHR           SA TARGET
//      DD   DSN=&SQ1..DSIPARM,DISP=SHR             NETVIEW TARGET
//DSILIST DD   DSN=&Q1..&DOMAIN..DSILIST,DISP=SHR    USER TARGET
//DSIASRC DD   DSN=&Q1..&DOMAIN..DSIASRC,DISP=SHR    USER TARGET
//DSIARPT DD   DSN=&Q1..&DOMAIN..DSIARPT,DISP=SHR    USER TARGET
//DSIVTAM DD   DSN=&Q1..&DOMAIN..VTAMLST,DISP=SHR    USER TARGET
//DSIPRF  DD   DSN=&SQ2..SINGNPRF,DISP=SHR           SA TARGET
//      DD   DSN=&SQ1..DSIPRF,DISP=SHR             NETVIEW TARGET
//DSIMSG  DD   DSN=&SQ2..SINGMSG,DISP=SHR            SA US
//      DD   DSN=&SQ1..SDSIMSG1,DISP=SHR            NETVIEW TARGET
//BNJPNL1 DD   DSN=&SQ1..BNJPNL1,DISP=SHR
//BNJPNL2 DD   DSN=&SQ1..BNJPNL2,DISP=SHR
//CNMPNL1 DD   DSN=&SQ2..SINGNPNL,DISP=SHR           SA US
//      DD   DSN=&SQ1..CNMPNL1,DISP=SHR            NETVIEW US
//AOFSTAT DD   DSN=&Q1..&DOMAIN..STATS,DISP=SHR
//HSAIPL  DD   DSN=&VQ2..IPLDATA,DISP=SHR
//DSILOGP DD   DSN=&VQ1..&DOMAIN..DSILOGP,
//      DISP=SHR,AMP='AMORG,BUFNI=20,BUFND=20'
//DSILOGS DD   DSN=&VQ1..&DOMAIN..DSILOGS,
//      DISP=SHR,AMP='AMORG,BUFNI=20,BUFND=20'
//DSITRCP DD   DSN=&VQ1..&DOMAIN..DSITRCP,DISP=SHR,AMP=AMORG
//DSITRCS DD   DSN=&VQ1..&DOMAIN..DSITRCS,DISP=SHR,AMP=AMORG
//DSISVRT DD   DSN=&VQ1..&DOMAIN..DSISVRT,
//      DISP=SHR,AMP=AMORG
//INGDUMP DD   DSN=&VQ2..&DOMAIN..INGDUMP,DISP=SHR
//SYSPRINT DD  SYSOUT=&SOUTA

```

We issue the **MVS S AOFAPPL** command to start the NetView subsystem Agent. Example 6-9 shows the initialization message.

Example 6-9 AOFAPPL initialization message

```

SDSF OUTPUT DISPLAY AOFAPPL STC03422 DSID      2 LINE 0          COLUMNS 01- 80
COMMAND INPUT ==>          SCROLL ==>  CSR
*****
***** TOP OF DATA *****
      J E S 2  J O B  L O G  --  S Y S T E M  S C 6 4  --  N O D E

20.41.53 STC03422 ---- THURSDAY,  25 AUG 2005 ----
20.41.53 STC03422 IEF695I START AOFAPPL WITH JOBNAME AOFAPPL IS ASSIGNED TO
20.41.53 STC03422 $HASP373 AOFAPPL  STARTED
20.41.53 STC03422 IEF403I AOFAPPL -  STARTED - TIME=20.41.53 - ASID=0080 - SC64

```

*****Page few pages down to see message id AOF540I *****

20.47.51 STC03422 AOF540I 20:47:51 : INITIALIZATION RELATED PROCESSING HAS BEEN COMPLETED.

Note: After you install the host components (automation manager, interface, and agent) of IBM Tivoli System Automation for z/OS V3.1, we recommend that you perform a system IPL, per the guidelines provided in *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261, manual.

Important: In our environment , IBM Tivoli System Automation for z/OS V3.1, the procedure Agent did not produce a message A0F617I SA z/OS INITIALIZATION STARTED after message DW0854I VTAM is active.

The following corrections were made to initialize IBM Tivoli System Automation for z/OS V3.1:

Update the xxxSTGEN member of DSIPARM (in our environment the member name is C64STGEN) with the following statement as the last statement in the file: *TOWER.SA = SYSOPS*

Make sure COMSTNXT is copied into your DSIPARM. Follow the instructions provided by the COMSTNXT member as required.

6.2.8 Define the base automation policy

Before you can start using automation, you need to define your base automation policy using the IBM supplied policies. This involves the following actions:

- ▶ If applicable, migrate/merge existing policy information; you can use the sample job INGEBMIG in the SINGSAMP sample library.

Since we have a brand new IBM Tivoli System Automation for z/OS V3.1 installation, we use the sample automation policy.

- ▶ Add further policy definitions.

In our case, we create new policy definitions for IBM DB2. We show this in “Configuring automation policies for IBM DB2” on page 145.

- ▶ Distribute the policy definitions (the policy database) where required.

Since we have a single z/OS system, we do not need to perform this task.

Since we have started our IBM Tivoli System Automation for z/OS V3.1 from scratch, we use the IBM samples delivered with the product and create our new policy database.

We use the following process to define the base automation policy in our environment:

1. We have our customization dialog installed on our single z/OS system, SC64. We create the policy database using the ISPF input panels of the customization dialog, as described in the *IBM Tivoli System Automation for z/OS V3.1 Defining Automation Policy*, SC33-8262-01 manual.
2. After our policy database has been created we used the BUILD function of the customization dialog to produce the following:
 - The System operations control files: Automation Control File (ACF) and Automation Manager Configuration File (AMC)
 - NetView Automation Table (AT)
 - The message processing facility for IBM Tivoli System Automation for z/OS: MPFLSTSA.

As we are automation a standalone system, there is no need to distribute above the generated output to other sysplexes or other standalone systems for automation. Refer to the *IBM Tivoli System Automation for z/OS V3.1 Defining Automation Policy*, SC33-8262-01 manual for details on how to distribute the control files

IBM Tivoli System Automation for z/OS V3.1 provides a sample job named INGEBCLD in the SINGSAMP sample library to perform the build in batch mode. This job is configured accordingly to your the installation requirements. In our case study scenario we used the ISPF Customization dialog.

The BUILD command is available from various panels of the customization dialog. For more information on how to perform this step, refer to the *IBM Tivoli System Automation for z/OS V3.1 Defining Automation Policy*, SC33-8262-01 manual.

Important: Do not change the system operations control files manually. You must use the customization dialog to manipulate the policy objects.

6.3 Configuring automation policies for IBM DB2

IBM DB2 Automation on IBM Tivoli System Automation for z/OS V3.1 has been produced to provide automated functions for the database environment used in our case study scenario. This will be shown by automating stop, start, and recovery procedures of the IBM DB2 software running on our z/OS system.

In order to accomplish automation of our IBM DB2 environment, the following tasks must be performed:

- ▶ “Identify required IBM DB2 messages” on page 146
- ▶ “Create scenario automation policy database” on page 147
- ▶ “Populate the scenario policy database” on page 149
- ▶ “Define policies for monitoring IBM DB2 application tasks” on page 152
- ▶ “Import customized scenario policy database into production” on page 165
- ▶ “Create application group and define group membership” on page 170
- ▶ “Verify Relationships in the automation policy” on page 176

The above tasks are in accordance to the *IBM Tivoli System Automation for z/OS V3.1 Customizing and Programming, SC33-8260-01* manual, which provides instructions on how to perform IBM DB2 automation using IBM Tivoli System Automation for z/OS V3.1.

6.3.1 Identify required IBM DB2 messages

Certain messages must be available in MPFLSTxx for the automated operations to function for IBM DB2. These messages are produced by IBM DB2 and are made available in the MPFLSTxx per our configuration shown in “Update PARMLIB data sets” on page 136.

IBM DB2 required recovery commands are defined for the message in the automation policy item MESSAGES/USER DATA for any IBM DB2 subsystem that requires IBM DB2 critical event message recovery. If the IBM DB2 subsystem is known to IBM Tivoli System Automation for z/OS as an application of type IBM DB2, event message recovery can be controlled by parameters entered via the IBM DB2 control policy item for the subsystem.

Automation Table statements that call the generic routine ISSUECMD or a IBM DB2-specific routine are only created during the build process for messages that have commands defined in the Automation Control File. If a recovery action is the only action processed when the triggering message is issued by a subsystem of type DB2, the created automation table statement is labeled with the group name DB2. Otherwise the Automation Table statement is created without a label.

Created automation table statements that call the generic routine ISSUECMD are conditional and can be overwritten via automation policy item MESSAGES/USER DATA.

Restrictions and Limitations Critical event monitoring is only done if the IBM DB2 subsystem is defined to IBM Tivoli System Automation for z/OS. These IBM DB2 messages are provided by IBM DB2 Automation policy as default.

6.3.2 Create scenario automation policy database

The policy database samples contain samples of IBM DB2 Automation (all except the *DEFAULT sample). The IBM DB2 entries are contained in the CLASS/INSTANCE application (APL) relationships. Refer to the *IBM Tivoli System Automation for z/OS V3.1 Defining Automation Policy*, SC33-8262-01, manual for detailed information about how to implement these samples and other entries into your automation policy.

Once we create the policy database (PDB) for our basic z/OS system with the base automation policy, and install all prerequisites, the following tasks must be initialized and running before proceeding with the IBM DB2 automation definition.

- ▶ Program to Program Interface
- ▶ The Automaton Manager
- ▶ The Agent subsystem

Now we are ready to add IBM DB2 Automation requirements. We want to create a policy database using the sample DB2 policy database as model.

In order to achieve this, we perform the following steps:

1. On the IBM Tivoli System Automation for z/OS Customization Dialog Primary Menu, select Option 4 (Policies - Maintain Policy Database list). See Example 6-10.

Example 6-10 Customization Dialog Primary Menu

```
SA z/OS 3.1 Customization Dialog Primary Menu
Option ==> 4

0 Settings          User parameters
1 Open              Work with the Policy Database
2 Build             Build functions for Policy Database
3 Report            Generate reports from Policy Database
4 Policies           Maintain Policy Database list
5 Data Management  Import policies/Migrate files into a Policy Database
U User              User-defined selections

X Exit              Terminate Customization Dialog
```

To switch to another Policy Database, specify the Policy Database name in the following field, or specify a ? to get a selection list.

Current Policy Database . . . ITS0_V3R1_PDB

2. See Example 6-11. On the Menu, select **COMMANDS**. The system prompts you with the following dialog or you can enter N (new). Create a new database by selecting Option 2.

Example 6-11 Adding IBM DB2 sample Policy Database

MENU	COMMANDS	ACTIONS	VIEW	HELP
2	1. OPEN.....(O)	- Open selected database		
	2. NEW.....(N)	- Create new database		
	3. EDIT.....(E)	- Edit selected database		
	4. DELETE....(D)	- Delete selected database		
	5. ADD.....(A)	- Add database to list		
	6. REMOVE....(M)	- Remove database from list		
	7. BUILD.....(B)	- Build functions		
	8. MANAGE....(G)	- Data Management		
	9. REPORT....(R)	- Generate database report		
	10. VIEW.....(V)	- Cycle thru alternate views		
	11. LOCATE....(L)	- Locate entry		
	12. END.....	- Exit panel		

3. Press Enter. You will be prompted with the next dialog. We enter the Policy Database Name, Enterprise Name, and Data Set Name as in Example 6-12.

Example 6-12 Define new IBM DB2 Policy Database

To define a new Policy Database, specify the following information:			
Policy Database Name	.	.	CASE_STUDY_SCENARIO
Enterprise Name	.	.	SANDBOX_DB2
Data Set Name	.	.	'SA310USR.DB2.PDB1'
Model Policy Database	.	.	?
			Policy Database name or "?" for list of names

4. When prompted to select model policy database, we select DB2 from the list as in Example 6-13.

Example 6-13 Select DB2 Policy Database

Select the database to serve as a model:			
Action	Status	PolicyDB Name	Enterprise Name
		*BASE	BASE
		*EMPTY	EMPTY
S		DB2	TES3

5. Now the Create a New Policy Database panel contains the DB2 set as the policy database model as in Example 6-14.

Example 6-14 Create a New Policy Database

To define a new Policy Database, specify the following information:			
Policy Database Name	.	.	CASE_STUDY_SCENARIO
Enterprise Name	.	.	SANDBOX_DB2
Data Set Name	.	.	'SA310USR.DB2.PDB1'

6. When prompted for the data set information, we keep the default attributes as in Example 6-15.

Example 6-15 Data set information

New Policy Database : CASE_STUDY_SCENARIO			
Attributes to be used for allocation of the new data set:			
		More:	+
Managed storage.	NO	YES	NO
Management class	TIV001	Blank for default management class	
Storage class.		Blank for default storage class	
Volume serial.		Blank for authorized default volume	
Data class		Blank for default data class *	
Space units.	CYLINDERS	CYLS TRKS BLKS KB MB	
Primary quantity . . .	1	1 to 999 - In above units	
Secondary quantity . .	1	0 to 999 - In above units	
Directory blocks . . .	50	1 to 999	
Record format. . . . :	FB		
Record length. . . . :	80		
Block size	32720		
Data Set Name type . .	PDS	LIBRARY	PDS

7. This completes the process for creating a new policy database named CASE_STUDY_SCENARIO using the DB2 policy database as the model.

6.3.3 Populate the scenario policy database

After we build the scenario policy database, we complete the following steps to populate it with the sample IBM DB2 automation policy.

- 1. From the Policy Database Selection panel, we select our newly created policy database: CASE_STUDY_SCENARIO, as shown in Example 6-16.

Example 6-16 Policy Database Selection panel

MENU COMMANDS ACTIONS VIEW HELP		

Policy Database Selection		Row 1 to 3 of 3
Command ==>		SCROLL==> PAGE
Action	Policy Database	Enterprise Name/Data Set Name
S	CASE_STUDY_SCENARIO	SANDBOX_DB2
		'SA310USR.DB2.PDB1'
	E2E	END_TO_END_MPV2R1
		'SA310.E2E.MPV2R1.PDB'
	ITS0_V3R1_PDB	ITS0TESTSC64

2. The Entry Type Selection panel displays as in Example 6-17.

Example 6-17 IBM DB2 Automation Policy Database

Entry Type Selection			
Option ==>			
1 ENT	Enterprise	30 TMR	Timers
2 GRP	Group	31 TMO	Timeout Settings
3 SBG	SubGroup	32 TPA	Tape Attendance
4 SYS	System (*)	33 MVC	MVS Component
5 APG	ApplicationGroup (*)	34 MDF	MVSCOMP Defaults
6 APL	Application (*)	35 SDF	System Defaults
7 EVT	Events	36 ADF	Application Defaults
8 SVP	Service Periods	37 AOP	Auto Operators
9 TRG	Triggers	38 NFY	Notify Operators
10 PRO	Processor	39 NTW	Network
11 MTR	Monitor Resource (*)	40 NNT	NNT Sessions
		41 RES	Resident CLISTs
20 PRD	Product Automation	42 SCR	Status Details
		99 UET	User E-T Pairs
	(*) Multi-User-Capable		

3. We select entry type **Application** from the menu (Option 6).
4. We then enter DB2_MSTR as the entry name to represent a policy object for the scenario's IBM DB2 master subsystem. See Example 6-18.
5. On the Define New Entry panel, we enter our IBM DB2 Master Subsystem Name, the Application Type, the Subtype (one of: MSTR, SPAS, IRLM, DBM1, DIST, or WLMS), and the MVS Job Name. Example 6-18 shows this activity.

Example 6-18 Define Entry panel

Subsystem Name	DB8Q
Application Type	: DB2
Subtype	MSTR
Job Name	DB8QMSTR

6. This brings us to the Policy Selection panel for Applications. From here, we select policy item Link Instance to Class. We select the class named C_DB2_MSTR, as shown in Example 6-19.

```

COMMANDS  ACTIONS  VIEW  HELP
-----
Link Instance to Class                                Row 1 to 2 of 2
Command ==>                                           SCROLL==> PAGE

Entry Type : Application                                PolicyDB Name   : CASE_STUDY_SCENARIO
Entry Name  : DB2_MSTR                                Enterprise Name  : SANDBOX_DB2

Action      Status      Entry Name
          S      C_DB2_DEPENDENTS
          S      C_DB2_MSTR
***** Bottom of data *****

```

For example, the NORM START command looks like this: `MVS &SUBSCMDPFX STA DB2 &EHKVAR1`. Also, this class provides proper procedures to stop IBM DB2. For example, it includes the `INGRDTH &SUBSAPPL S` command to cancel with notification any outstanding threads prior to IBM DB2 shutdown. If you would prefer that threads are not cancelled, this command should be changed to read: `INGRDTH &SUBSAPPL S N` as per IBM DB2 defaults.

- Example 6-20 DB2_MSTR Policy Selection panel*

Chapter 6. Case study scenario: IBM DB2 on z/OS first-level automation domain **151**

8. When presented with the panel Application Information, we enter the command prefix characters of our IBM DB2 master subsystem in the **Command Prefix** entry as seen in Example 6-21.

Example 6-21 Command Prefix

COMMANDS HELP

		Application Information	Top of data
Command ==>			
Entry Type : Application		PolicyDB Name : CASE_STUDY_SCENARIO	
Entry Name : DB2_MSTR		Enterprise Name : SANDBOX_DB2	
			More: +
Application Type	: DB2	(STANDARD IMAGE JES2 JES3 CICS IMS DB2 OPC USS)	
Subtype	MSTR	(For STANDARD, CICS, IMS, DB2, OPC)	
Subsystem Name	DB8Q		
Job Type		(MVS NONMVS TRANSIENT)	
Job Name	DB8QMSTR		
Transient Rerun.		(YES NO)	

-----PAGE DOWN to see the page 2 of the panel-----

Command Prefix	-DB8Q
Message Prefix	
Sysname	(System name)

6.3.4 Define policies for monitoring IBM DB2 application tasks

We populated the scenario policy database named CASE_STUDY_SCENARIO with the IBM-supplied IBM DB2 policy database, and we initially configured the scenario policy database with information about our IBM DB2 master subsystem. We described these activities in previous sections.

In this section, we define an automation policy for our environment's IBM DB2 application tasks.

When our IBM DB2 master subsystem task (DB8QMSTR) starts, its children tasks: DB8QDBM1, DB8QDIST, and DB8QIRLM are also started. Similarly, when DB2QMSTR stops, the children tasks are stopped first.

For this reason, the children tasks must also be configured on IBM Tivoli System Automation for z/OS V3.1 for monitoring purposes. IBM Tivoli System

Automation for z/OS V3.1 cannot start and stop the children tasks of DB8QMSTR because these tasks are controlled within DB8QMSTR.

We configure dependency relationships on our IBM Tivoli System Automation for z/OS as follows:

- ▶ The DB8QMSTR task is dependent on z/OS resources (JES2 and TCP/IP).
- ▶ The children tasks DB8QDBM1, DB8QDIST, and DB8QIRLM are dependent on the parent task DB8QMSTR.

Before defining these relationships, we perform additional configuration steps as follows:

- ▶ Set the Active log data set name to our scenario policy database.
- ▶ As we create our scenario policy database using the DB2 policy database as our model, the application names are set to the default values. We need to change these names to the names used in our case study environment.

Setting the Active log data set name

We perform the following steps:

1. From the Policy Database Selection panel, we select our newly created policy database: **CASE_STUDY_SCENARIO**.
2. On the Entry Type Selection panel, we enter Option **6** for Applications.
3. On the Entry Name Selection panel, under the Action, we select DB2_MSTR. See Example 6-22.

Example 6-22 Selection of DB2_MSTR

Entry Name Selection		Row 1 to 7 of 7	
Command ==>		SCROLL==> PAGE	
Entry Type : Application		PolicyDB Name : CASE_STUDY_SCENARIO	
		Enterprise Name : SANDBOX_DB2	
Action	Entry Name	C	Short Description
	C_DB2_DEPENDENTS	*	DB2 Class - DIST,DBM1,IRLM,SPAS
	C_DB2_MSTR	*	DB2 Class - System Services
	DB2_DBM1		DB2 Database Services
	DB2_DIST		DB2 Distributed Data Facility
	DB2_IRLM		DB2 Resource Lock Manager
S	DB2_MSTR		DB2 Subsystem
	DB2_SPAS		DB2 Stored Procedures
***** Bottom of data *****			

4. We then selected DB2 CONTROL, as seen in Example 6-23.

Example 6-23 DB2 CONTROL option

Policy Selection		Row 1 to 13 of 21	
Command ==>		SCROLL==> PAGE	
Entry Type : Application		PolicyDB Name : CASE_STUDY_SCENARIO	
Entry Name : DB2_MSTR		Enterprise Name : SANDBOX_DB2	
Action	Policy Name	Policy Description	
	DESCRIPTION	Enter description	
	LINK TO CLASS	Link instance to class	
	APPLICATION INFO	Define application information	
	AUTOMATION FLAGS	Define application automation flags	
	TRIGGER	Select trigger	
	SERVICE PERIOD	Select service period	
	RELATIONSHIPS	Define relationships	
	MESSAGES/USER DATA	Define application messages and user data	
	STARTUP	Define startup procedures	
	SHUTDOWN	Define shutdown procedures	
	THRESHOLDS	Define error thresholds	
	MINOR RESOURCE FLAGS	Define application sub-component flags	
	SYSTEM ASSOCIATION	Define primary and secondary associations	
	-----	-----RESOURCES-----	
	GENERATED RESOURCES	List resources generated for this entry	
	MEMBER OF	List resources where this entry is a member	
	-----	-----DB2 SPECIFIC POLICY-----	
S	DB2 CONTROL	Define DB2 subsystem specific data	
	-----	-----	
	WHERE USED	List application groups linked to this entry	
	COPY	Copy data from an existing entry	
***** Bottom of data *****			

5. The DB2 Control Entries panel displays. See Example 6-24.

Example 6-24 Our case study scenario DB2 Control Entries panel

Entry Type : Application		PolicyDB Name : CASE_STUDY_SCENARIO	
Entry Name : DB2_MSTR		Enterprise Name : SANDBOX_DB2	
Subsystem : DB8Q			
Subtype : MSTR		Subtype from APPLICATION INFO policy	
		More: +	
Enter or update the following fields:			
DB2 subsystem id DB8Q			
Active log data set name . .			
DB8QU.LOGCOPY1.DS01			
Log full threshold 09		Percentage full	
Shutdown if indoubt. YES		YES NO blank	
Process iteration delays (HH:MM:SS):			

TSO logoff delay 00:00:59
STOP tablespace delay. . . . 00:03:00
Connection Monitor delay . . 00:30:00
Terminate threads delay. . . 00:01:00 Cycles . . . 5
Active log alert 00:00:30 Threshold. . 4
Log offload interval 00:30:00

We enter the data set name of the IBM DB2 Active Log Data set in this panel. This will allow for checks before the command to message DSNJ002I is issued. These information will be advised by the IBM DB2 System Programmer.

Note: The above task cannot be done in CLASS C_DB2_MSTR if there are more than the IBM DB2 master defined and linked to this CLASS. The application names will be different. The policy database build will fail.

6. We save the changes and return to the Policy Selection panel.

Renaming IBM DB2 application names

We chose to rename the applications DB2_DBM, DB2_DIST, and DB2_IRLM to the names of our environment (DB8QDBM1, DB8QDIST, and DB8QIRLM), because we do not have multiple IBM DB2 master subsystems.

In order to rename the application names, we perform the following steps:

- 1. From the Policy Database Selection panel, we select our newly created policy database: CASE_STUDY_SCENARIO.
- 2. On the Entry Type Selection panel, we enter Option 6 for Applications.
- 3. On the Entry Name panel, we select R (Rename) for DB2_DBM1. See Example 6-25.

Example 6-25 Renaming Application to DB8QMSTR

COMMANDS ACTIONS VIEW HELP			

Command ==>		Entry Name Selection	Row 1 to 7 of 7 SCROLL==> PAGE
Entry Type : Application		PolicyDB Name : CASE_STUDY_SCENARIO	
		Enterprise Name : SANDBOX_DB2	
Action	Entry Name	C Short Description	
	C_DB2_DEPENDENTS	* DB2 Class - DIST,DBM1,IRLM,SPAS	
	C_DB2_MSTR	* DB2 Class - System Services	
R	DB2_DBM1	DB2 Database Services	

DB2_DIST	DB2 Distributed Data Facility
DB2_IRLM	DB2 Resource Lock Manager
DB2_MSTR	DB2 Subsystem
DB2_SPAS	DB2 Stored Procedures

***** Bottom of data *****

4. On the Entry Rename panel (Example 6-26), we want to rename DB2_DBM1 to DB8QDBM1.

Example 6-26 Entry Rename panel

COMMANDS ACTIONS VIEW HELP

Entry Name Selection

Row 3 to 7 of 7

SCROLL==> PAGE

Command ==>

Entry Type : Application

PolicyDB Name : CASE_STUDY_SCENARIO

Enterprise Name : SANDBOX_DB2

Action

Entry Name

C Short Description

R

DB2_DBM1

DB2 Database Services

E

Es

Entry Rename

e

e

e Description : DB2 Database Services

e

***** e Old Name : DB2_DBM1

e *****

e

e New Name . . db8qdbm1

e

e Press ENTER to rename member.

e Press CANCEL to cancel rename.

e F1=Help F2=Split F3=End F9=Swap F12=Cancel e

D

Ds

y

N

5. We also update the Application Information panel with the correct job name as seen in Example 6-27.

Example 6-27 DB8QDBM1 Application

Application Information

Command ==>

Entry Type : Application

PolicyDB Name : CASE_STUDY_SCENARIO

Entry Name : DB8QDBM1

Enterprise Name : SANDBOX_DB2

More: +

Application Type : DB2

(STANDARD IMAGE JES2 JES3

CICS IMS DB2 OPC USS)

Subtype DBM1

(For STANDARD, CICS, IMS, DB2, OPC)

Subsystem Name DB8QDBM1

Job Type	(MVS NONMVS TRANSIENT)
Job Name DB8QDBM1	
Transient Rerun.	(YES NO)
Scheduling Subsystem . . .	(MSTR, JES Subsystem or blank)
JCL Procedure Name	
Captured Messages Limit. .	(0 to 999, or blank)
Restart after IPL.	(START NOSTART NONE blank)
Restart after Recycle. . .	(START NOSTART NONE blank)
Start Timeout	(time for "UP" status checks, hh:mm:ss)

6. We repeat the above steps to rename DB2_DIST to DB8QDIST and DB2_IRLM to DB8QIRLM.

Relationships between DB8QMSTR and z/OS resources

To define dependency relationships among the IBM DB2 master subsystem DB8QMSTR and JES2 and TCP/IP resources on z/OS, we perform the following steps:

1. From the Policy Database Selection panel, we select our newly created policy database: CASE_STUDY_SCENARIO.
2. On the Entry Type Selection panel, enter Option **6** for Applications, as seen in Example 6-28.

Example 6-28 Entry Type Selection panel

Entry Type Selection			
Option ==> 6			
1 ENT	Enterprise	30 TMR	Timers
2 GRP	Group	31 TMO	Timeout Settings
3 SBG	SubGroup	32 TPA	Tape Attendance
4 SYS	System (*)	33 MVC	MVS Component
5 APG	ApplicationGroup (*)	34 MDF	MVSCOMP Defaults
6 APL	Application (*)	35 SDF	System Defaults
7 EVT	Events	36 ADF	Application Defaults
8 SVP	Service Periods	37 AOP	Auto Operators
9 TRG	Triggers	38 Nfy	Notify Operators
10 PRO	Processor	39 NTW	Network
11 MTR	Monitor Resource (*)	40 NNT	NNT Sessions
		41 RES	Resident CLISTs
20 PRD	Product Automation	42 SCR	Status Details
		99 UET	User E-T Pairs
	(*) Multi-User-Capable		

- On the Entry Name Selection panel, we select the C_DB2_MSTR class, as seen in Example 6-29.

Example 6-29 Entry Name Selection panel

```

COMMANDS  ACTIONS  VIEW  HELP
-----
                                Entry Name Selection                Row 1 to 7 of 7
Command ==>                                SCROLL==> PAGE

Entry Type : Application                PolicyDB Name   : CASE_STUDY_SCENARIO
                                           Enterprise Name : SANDBOX_DB2

Action      Entry Name                  C Short Description
S           C_DB2_DEPENDENTS            * DB2 Class - DIST,DBM1,IRLM,SPAS
           C_DB2_MSTR                   * DB2 Class - System Services
           DB2_DBM1                     DB2 Database Services
           DB2_DIST                     DB2 Distributed Data Facility
           DB2_IRLM                     DB2 Resource Lock Manager
           DB2_MSTR                     DB2 Subsystem
           DB2_SPAS                     DB2 Stored Procedures
***** Bottom of data *****

```

- See Example 6-30. We then select RELATIONSHIPS on the Policy Selection panel.

Example 6-30 RELATIONSHIPS: Policy Selection panel

```

ACTIONS  HELP
-----
                                Policy Selection                Row 1 to 13 of 16
Command ==>                                SCROLL==> PAGE

Entry Type : Application                PolicyDB Name   : CASE_STUDY_SCENARIO
Entry Name  : C_DB2_MSTR                Enterprise Name : SANDBOX_DB2

Action      Policy Name                  Policy Description
s           DESCRIPTION                  Enter description
           LINK TO INSTANCES            Link class to instances
           APPLICATION INFO              Define application information
           AUTOMATION FLAGS              Define application automation flags
           TRIGGER                       Select trigger
           SERVICE PERIOD                Select service period
           RELATIONSHIPS                 Define relationships
           MESSAGES/USER DATA           Define application messages and user data
           STARTUP                       Define startup procedures
           SHUTDOWN                      Define shutdown procedures
           THRESHOLDS                   Define error thresholds
           MINOR RESOURCE FLAGS          Define application sub-component flags

```

```

-----DB2 SPECIFIC POLICY-----
      DB2 CONTROL          Define DB2 subsystem specific data
-----
      COPY                  Copy data from an existing entry
***** Bottom of data *****

```

- On the Defining Relationship panel, we add the dependency of JES2 using the **Supporting Resource. JES/APL/=** and **Relationship Type = HASPARENT**. See Example 6-31.

Example 6-31 Define Relationship to JES2

```

COMMANDS  HELP
-----
                                Define Relationship
Command ==>

Entry Type : Application          PolicyDB Name  : CASE_STUDY_SCENARIO
Entry Name : C_DB2_MSTR          Enterprise Name : SANDBOX_DB2

                                     More:      +

Subsystem :      C_DB2_MSTR
Description. . . . .

Relationship Type. . HASPARENT      MAKEAVAILABLE MAKEUNAVAILABLE
                                     PREPAVAILABLE PREPUNAVAILABLE
                                     FORCEDOWN EXTERNALLY HASMONITOR
                                     HASPARENT HASPASSIVEPARENT

Supporting Resource. JES/APL/=

Sequence Number. . . 1              Resource Name
                                     Sequence Number (1-99,blank)

Automation . . . . .                ACTIVE PASSIVE
Chaining . . . . .                  STRONG WEAK

```

- We save the changes and the Relationship Selection List panel displays (Example 6-32) showing the Supporting Resource and the Relationship Type. As defined, DB8QMSTR will not start if JES2 is not available.

Example 6-32 Relationship Selection List panel

```

                                Relationship Selection List          Row 1 to 1 of 1
Command ==> N

Entry Type : Application          PolicyDB Name  : CASE_STUDY_SCENARIO
Entry Name : C_DB2_MSTR          Enterprise Name : SANDBOX_DB2

External Startup. . .            (INITIAL ALWAYS NEVER blank)
External Shutdown . .           (FINAL ALWAYS NEVER blank)

```

Action #	Type	Supporting Resource	Auto	Chain
	HASPARENT	JES2/APL/=		
***** Bottom of data *****				

7. To define the relationship to TCP/IP, we perform similar steps. Example 6-33 shows the Define Relationship panel we use for the TCP/IP relationship definition.

Example 6-33 Adding additional resource

COMMANDS HELP

Define Relationship

Command ==>

Entry Type : Application

Entry Name : C_DB2_MSTR

PolicyDB Name : CASE_STUDY_SCENARIO

Enterprise Name : SANDBOX_DB2

Subsystem : C_DB2_MSTR

Description.

Relationship Type. . HASPARENT

MAKEAVAILABLE MAKEUNAVAILABLE
PREPAVAILABLE PREPUNAVAILABLE
FORCEDOWN EXTERNALLY HASMONITOR
HASPARENT HASPASSIVEPARENT

Supporting Resource. TCIP/APL/=

Resource Name

Sequence Number. . .

Sequence Number (1-99,blank)

Automation

ACTIVE PASSIVE

Chaining

STRONG WEAK

8. Example 6-34 shows both relationships defined for DB8QMSTR with Relationship Type HASPARENT.

Example 6-34 Adding additional supporting resource

Relationship Selection List Row 1 to 2 of 2

Command ==>

Entry Type : Application

Entry Name : C_DB2_MSTR

PolicyDB Name : CASE_STUDY_SCENARIO

Enterprise Name : SANDBOX_DB2

External Startup. . .

(INITIAL ALWAYS NEVER blank)

External Shutdown . .

(FINAL ALWAYS NEVER blank)

Action #	Type	Supporting Resource	Auto	Chain
----------	------	---------------------	------	-------


```
HASPARENT      JES/APL/=
HASPARNT      TCIP/APL/=
***** Bottom of data *****
```

9. IBM Tivoli System Automation for z/OS V3.1 allows the user to type particular messages to trigger specific commands in the Message Processing dialog. Example 6-35 shows the messages we use in our environment.

For information about how to add these entries to the MESSAGES/USER DATA policy item, refer to the *IBM Tivoli System Automation for z/OS V3.1 Customizing and Programming*, SC33-8260-01, manual.

Example 6-35 Message Processing

Entry Type	: Application	PolicyDB Name	: CASE_STUDY_SCENARIO				
Entry Name	: C_DB2_MSTR	Enterprise Name	: SANDBOX_DB2				
Define message IDs and their automation actions.							
CMD	= Command	REP	= Reply				
CODE	= CODE	USER	= User Data				
AUTO	= AT Actions	OVR	= AT Override				
Action	Message ID	Cmd	Rep	Code	User	Auto	Ovr
	Description						
	ACORESTART	1					
	Specifications for ACORESTART						
	DATABASE			3			
	Specifications for DATABASE						
	DSNJ002I	1					
	Specifications for DSNJ002I						
	DSNJ115I	1					
	Specifications for DSNJ115I						
	DSNL008I	1					
	Specifications for DSNL008I						
	DSNP007I	1					
	Specifications for DSNP007I						
	DSNR002I	1					
	Specifications for DSNR002I						
	SHUTFORCEDDF	1					
	Specifications for SHUTFORCEDDF						
	UP	1					
	Specifications for UP						

Relationships between children tasks and DB8QMSTR

The children tasks DB8QDBM1, DB8QDIST, and DB8QIRLM are dependent on the parent task DB8QMSTR.

In order to define relationships between the children tasks and the DB8QMSTR, we perform the following tasks.

1. From the Policy Database Selection panel, we select our newly created policy database: CASE_STUDY_SCENARIO.
2. On the Entry Type Selection panel, enter Option **6** for Applications.
3. On the Entry Name Selection panel, we select the C_DB2_DEPENDENTS class, as seen in Example 6-36.

Example 6-36 Entry Name Selection panel

COMMANDS	ACTIONS	VIEW	HELP

Entry Name Selection		Row 1 to 7 of 7	
Command ==>		SCROLL==> PAGE	
Entry Type : Application		PolicyDB Name : CASE_STUDY_SCENARIO	
		Enterprise Name : SANDBOX_DB2	
Action	Entry Name	C Short Description	
S	C_DB2_DEPENDENTS	* DB2 Class - DIST,DBM1,IRLM,SPAS	
	C_DB2_MSTR	* DB2 Class - System Services	
	DB2_DBM1	DB2 Database Services	
	DB2_DIST	DB2 Distributed Data Facility	
	DB2_IRLM	DB2 Resource Lock Manager	
	DB2_MSTR	DB2 Subsystem	
	DB2_SPAS	DB2 Stored Procedures	
***** Bottom of data *****			

4. We then select RELATIONSHIPS on the Policy Selection panel. See Example 6-37.

Example 6-37 Policy Selection panel

Policy Selection		Row 1 to 13 of 14
Command ==>		SCROLL==> PAGE
Entry Type : Application		PolicyDB Name : CASE_STUDY_SCENARIO
Entry Name : C_DB2_DEPENDENTS		Enterprise Name : SANDBOX_DB2
Action	Policy Name	Policy Description
	DESCRIPTION	Enter description
	LINK TO INSTANCES	Link class to instances
	APPLICATION INFO	Define application information
	AUTOMATION FLAGS	Define application automation flags
	TRIGGER	Select trigger
	SERVICE PERIOD	Select service period
S	RELATIONSHIPS	Define relationships
	MESSAGES/USER DATA	Define application messages and user data

STARTUP	Define startup procedures
SHUTDOWN	Define shutdown procedures
THRESHOLDS	Define error thresholds
MINOR RESOURCE FLAGS	Define application sub-component flags

COPY	Copy data from an existing entry
***** Bottom of data *****	

- On the Defining Relationship panel, we add the dependency of JES2 using the **Supporting Resource. DB8Q/APL/=** and **Relationship Type = HASPARENT**. You must also specify the Condition as **StartsMeAndStopsMe**. The purpose of this condition is for DB8QMSTR start procedure to start its children. See Example 6-38.

Example 6-38 Define Relationship of C-DB2_DEPENDENTS

Define Relationship

Command ==>

Entry Type : Application

PolicyDB Name : CASE_STUDY_SCENARIO

Entry Name : C_DB2_DEPENDENTS

Enterprise Name : SANDBOX_DB2

More: +

Subsystem : C_DB2_DEP

Description.

Relationship Type. . HASPARENT

MAKEAVAILABLE MAKEUNAVAILABLE
PREPAVAILABLE PREPUNAVAILABLE
FORCEDOWN EXTERNALLY HASMONITOR
HASPARENT HASPASSIVEPARENT

Supporting Resource. DB8Q/APL/=

Resource Name

Sequence Number. . .

Sequence Number (1-99,blank)

Automation

ACTIVE PASSIVE

Chaining

STRONG WEAK

Chaining

STRONG WEAK

Condition StartsMeAndStopsMe

Satisfy condition
(? for list of possible values)

- We save the changes and the Relationship Selection List panel displays, showing the Supporting Resource and the Relationship Type. See Example 6-39.

Example 6-39 Relationship Selection List

Relationship Selection List

Row 1 to 1 of 1

Command ==>

Entry Type : ApplicationPolicyDB Name : CASE_STUDY_SCENARIOEntry Name : C_DB2_DEPENDENTSEnterprise Name : SANDBOX_DB2

External Startup. . . ALWAYS (INITIAL ALWAYS NEVER blank)External Shutdown . . ALWAYS (FINAL ALWAYS NEVER blank)

Action #TypeSupporting ResourceAutoChain

HASPARENTDB8Q/APL/=

***** Bottom of data *****

- We now must link DB8QDBM, DB8QDIST, and DB8QIRLM to an Instance. In order to do this, we perform the following steps:
- 1. From the Policy Database Selection panel, we select our newly created policy database: CASE_STUDY_SCENARIO.
 - 2. On the Entry Type Selection panel, enter Option **6** for Applications.
 - 3. On the Entry Name Selection panel, we select the C_DB2_DEPENDENTS class.
 - 4. On the Policy Selection panel, we select **LINK TO INSTANCES Link class to instances**. See Example 6-40.

Example 6-40 Link Instance to Class called C_DB2_DEPENDENTS

COMMANDS ACTIONS VIEW HELP		

Policy Selection		Row 2 to 14 of 14
Command ==>		SCROLL==> PAGE
Entry Type : Application	PolicyDB Name : CASE_STUDY_SCENARIO	
Entry Name : C_DB2_DEPENDENTS	Enterprise Name : SANDBOX_DB2	
Action	Policy Name	Policy Description
S	LINK TO INSTANCES	Link class to instances
	APPLICATION INFO	Define application information
	AUTOMATION FLAGS	Define application automation flags
	TRIGGER	Select trigger
	SERVICE PERIOD	Select service period
	RELATIONSHIPS	Define relationships
	MESSAGES/USER DATA	Define application messages and user data
	STARTUP	Define startup procedures
	SHUTDOWN	Define shutdown procedures
	THRESHOLDS	Define error thresholds
	MINOR RESOURCE FLAGS	Define application sub-component flags

	COPY	Copy data from an existing entry

5. Now we require all DB8QMSTR children tasks to be selected to this Link Class to Instances. See Example 6-41.

Example 6-41 Link Class to Instances

COMMANDS	ACTIONS	VIEW	HELP

Link Class to Instances			Row 1 to 5 of 5
Command ==>			SCROLL==> PAGE
Entry Type : Application		PolicyDB Name : CASE_STUDY_SCENARIO	
Entry Name : C_DB2_DEPENDENTS		Enterprise Name : SANDBOX_DB2	
Action	Status	Entry Name	currently linked
	SELECTED	DB2_DIST	C_DB2_DEPENDENTS
	SELECTED	DB2_IRLM	C_DB2_DEPENDENTS
		DB2_MSTR	C_DB2_MSTR
	SELECTED	DB8QDBM1	C_DB2_DEPENDENTS
***** Bottom of data *****			

6. We save all the changes made in the above steps.

Now we are ready to import the CASE_STUDY_SCENARIO policy database into our enterprise policy database named ITSO_V3R1_PDB. There are various ways to do this. In our Case Study Scenario, we use the SA z/OS 3.1 Customization Dialog Primary Menu.

6.3.5 Import customized scenario policy database into production

Our customization of our CASE_STUDY_SCENARIO policy database contains the required automation policy configurations for IBM DB2 in our environment.

These configurations need to be active in order for IBM Tivoli System Automation for z/OS to automate the database environment for our sample application. In order to achieve this, we need to import the automation policy defined in the CASE_STUDY_SCENARIO policy database into the current production policy named ITSO_V3R1_PDB.

We use the following steps to import our customized policy database for IBM DB2 automation.

1. On the SA z/OS 3.1 Customization Dialog Primary Menu, we select Option 5 - Data management. See Example 6-42.

Example 6-42 SA z/OS 3.1 Customization Dialog Primary Menu

SA z/OS 3.1 Customization Dialog Primary Menu

Option ==> 5

0 Settings	User parameters
1 Open	Work with the Policy Database
2 Build	Build functions for Policy Database
3 Report	Generate reports from Policy Database
4 Policies	Maintain Policy Database list
5 Data Management	Import policies/Migrate files into a Policy Database
U User	User-defined selections
X Exit	Terminate Customization Dialog

To switch to another Policy Database, specify the Policy Database name in the following field, or specify a ? to get a selection list.

Current Policy Database . . . ITS0_V3R1_PDB

2. On the Data Management Menu, we select option 1 - Import from another Policy Database. See Example 6-43.

Example 6-43 Data Management Menu

MENU HELP

Data Management Menu

Option ==> 1

1 Import from PDB	Import from another Policy Database
2 Import from Add-on	Import from predefined add-on policies
3 Update via File	Write selected data to file or read data from file
9 Migrate from ACF	Migrate ACF files (agent data) into Policy Database

3. On the Import entries from other Policy Database panel, we have to specify the source policy database and the entry type. See Example 6-44.

Example 6-44 Import entries from other Policy Database

Import entries from other Policy Database

Option ==>

Current Policy Database : ITS0_V3R1_PDB

Enterprise Name : ITS0TESTSC64

1 Import Policy Data
Source Policy Database. . . . ? (? or name)
Entry type ? (? or type)
Import linked entries NO (YES or NO)
(applies to types APG,APL,TRG only)
2 View import report

4. For the Entry type field, we are interested in APG (ApplicationGroups) and APL (Application). Example 6-45 shows the Entry Type Selection panel in which we select the **APG** and **APL** types.

Example 6-45 Entry Type Selection Panel

```
-----
Entry Type Selection                               Row 1 to 14 of 34
Command ===>

Enterprise Name : ITS0TESTSC64      PolicyDB Name : ITS0_V3R1_PDB

Action      Type      Description
ADF         Application Defaults
AOP         Auto Operators
S           APG         ApplicationGroup
S           APL         Application
CCN         CICS link
CSA         CICS State/Action
CVP         Monitoring period
EVT         Event
GRP         Group
IRN         IMS resource name
ISA         IMS State/Action
ISF         IMS Status file
MDF         MVSCOMP Defaults
MTR         Monitor Resource
MVC         MVS Component
NFY         Notify Operators
NNT         NNT Sessions
NTW         Network
OCS         Controller details
ODM         Workstation domainID
OEN         OPC System details
OSR         Special resources
PRO         Processor
RES         Resident CLISTs
SBG         SubGroup
SCR         Status Details
SDF         System Defaults
```

SVP	Service Period
SYS	System
TMO	Timeout Settings
TMR	Timers
TPA	Tape Attendance
TRG	Trigger
UET	User E-T Pairs

5. Example 6-46 is the next panel that displays and it is the Policy Data Base Selection. There we select the policy database containing the automation policy for our database environment: CASE_STUDY_SCENARIO.

Example 6-46 Policy Data Base Selection

Policy Data Base Selection			Row 1 to 3 of 3
Command ==>			SCROLL==> PAGE
Action	PolicyDB Name	Enterprise Name/Data Set Name	
S	CASE_STUDY_SCENARIO	SANDBOX_DB2	
	E2E	END_TO_END_MPV2R1	
	ITS0_V3R1_PDB	ITS0TESTSC64	
***** Bottom of data *****			

6. Now, back to the Import entries from other Policy Database panel (Example 6-47), we select Option 1 to import the CASE_STUDY_SCENARIO policy database.

Example 6-47 Import entries from other Policy Database

```

Import entries from other Policy Database

Option ==> 1

Current Policy Database      : ITS0_V3R1_PDB
Enterprise Name              : ITS0TESTSC64

1 Import Policy Data
    Source Policy Database. . . . CASE_STUDY_SCENARIO      (? or name)
    Entry type . . . . . APL                                (? or type)
    Import linked entries . . . . NO                        (YES or NO)
                                     (applies to types APG,APL,TRG only)

2 View import report

```

7. Example 6-48 displays with further selections. In this panel, we require all applications, including the classes in the Entry Name field, to be imported to our current policy database. See Example 6-48.

Example 6-48 Class Entry Name

Entry Name Selection			Row 1 to 7 of 7
Command ==>			SCROLL==> PAGE
Action Status	Entry Name	Short Description	
S	C_DB2_DEPENDENTS	DB2 Class - DIST,DBM1,IRLM,SPAS	
S	C_DB2_MSTR	DB2 Class - System Services	
S	DB8QDBM1	DB2 Database Services	
S	DB8QDIST	DB2 Distributed Data Facility	
S	DB8QIRLM	DB2 Resource Lock Manager	
S	DB2_MSTR	DB2 Subsystem	
***** Bottom of data *****			

8. Example 6-49 shows the Selected Entry Names for Import panel, confirming our selections.

Example 6-49 Selected Entry Names for Import

Selected Entry Names for Import				Row 1 to 7 of 7
Command ==>				SCROLL==> PAGE
Action	Entry Name	Type C D	Short Description	
	C_DB2_DEPENDENTS	APL	DB2 Class - DIST,DBM1,IRLM,SPAS	
	C_DB2_MSTR	APL	DB2 Class - System Services	
	DB8QDBM1	APL	DB2 Database Services	
	DB8QDIST	APL	DB2 Distributed Data Facility	
	DB8QIRLM	APL	DB2 Resource Lock Manager	
	DB2_MSTR	APL	DB2 Subsystem	
***** Bottom of data *****				

9. Example 6-50 displays. It is the Confirm Entry Name List For Import panel, as seen in the following example.

Example 6-50 Confirm Entry Name List For Import

Confirm Entry Name List For Import				Row 1 to 7 of 7
Command ==>				SCROLL==> PAGE
Press ENTER to start Import Process.				
Press CANCEL or PF3 to return.				
Entry Names	Type	Link Only	Short Description	
C_DB2_DEPENDENTS	APL		DB2 Class - DIST,DBM1,IRLM,SPAS	
C_DB2_MSTR	APL		DB2 Class - System Services	
DB8QDBM1	APL		DB2 Database Services	
DB8QDIST	APL		DB2 Distributed Data Facility	

```
DB8QIRLM          APL          DB2 Resource Lock Manager
DB2_MSTR          APL          DB2 Subsystem
***** Bottom of data *****
```

10. When the import process starts, we follow the messages as you see in Example 6-51.

Example 6-51 Start of import process

```
Starting to read data from Policy Database 'CASE_STUDY_SCENARIO'.
Read data of Application 'C_DB2_DEPENDENTS'.
Read data of Application 'C_DB2_MSTR'.
Read data of Application 'DB8QDBM1'.
Read data of Application 'DB8QDIST'.
Read data of Application 'DB8QIRLM'.
Read data of Application 'DB2_MSTR'.
Store data into Policy Database 'ITSO_V3R1_PDB'.
```

6.3.6 Create application group and define group membership

As we can only automate applications by linking them to systems via an application group, we have to define an application group for the IBM DB2 applications in our environment.

In this section, we create the application group APPL_GROUP_DB2 and link it to the required system on which our IBM DB2 subsystem is to be automated, specifically, our SC64.

Later, we make the DB2_MSTR, DB8QDBM1, DB8QDIST, and DB8QIRLM tasks members of the APPL_GROUP_DB2.

Creating APPL_GROUP_DB2

We perform the following tasks to create the APPL_GROUP_DB2 application group:

- 1. From the Policy Database Selection panel, we select our production policy database: ITSO_V3R1_PDB.
- 2. On the Entry Type Selection panel (Example 6-52), enter Option 5 for ApplicationGroup.

Example 6-52 Entry Type Selection

```
Entry Type Selection

Option ==> 5

1 ENT      Enterprise      30 TMR      Timers
2 GRP      Group          31 TMO      Timeout Settings
```

3	SBG	SubGroup		32	TPA	Tape Attendance
4	SYS	System	(*)	33	MVC	MVS Component
5	APG	ApplicationGroup	(*)	34	MDF	MVSCOMP Defaults
6	APL	Application	(*)	35	SDF	System Defaults
7	EVT	Events		36	ADF	Application Defaults
8	SVP	Service Periods		37	AOP	Auto Operators
9	TRG	Triggers		38	NFY	Notify Operators
10	PRO	Processor		39	NTW	Network
11	MTR	Monitor Resource	(*)	40	NNT	NNT Sessions
				41	RES	Resident CLISTs
20	PRD	Product Automation		42	SCR	Status Details
				99	UET	User E-T Pairs
			(*) Multi-User-Capable			

3. On the Entry Name Selection panel (Example 6-53), we issue the **N APPL_GROUP_DB2** command to create a new application group **APPL_GROUP_DB2**.

Example 6-53 Define New Entry Type

		Entry Name Selection	Row 3 to 13 of 13
Command ==> N APPL_GROUP_DB2			SCROLL==> PAGE
Entry Type : ApplicationGroup		PolicyDB Name : ITS0_V3R1_PDB	
		Enterprise Name : ITS0TESTSC64	
Action	Entry Name	C Short Description	
	APPL_GROUP_MOVE	MOVE Application Group (SYSPLEX)	
	APPL_GROUP_MSTR		
	APPL_GROUP_PROXY	Proxy_Resource_Appl_Grp	
	APPLGR_DB531	DB2 D531 definition	
	APPLGR_SGPLEX	Sysplex group - sg11 + sg12	
	APPLL_GROUP_01	Application group 01	
	CICSPAPBGRP	CICSPAPB group for OPCA0	
	DB2_GROUP	SG11/12 Selector	
	SYS1_DB2_GROUP	SYS1 DB2 Group	
	SYS2_DB2_GROUP	All DB2 systems on SYS2	
	SYS2_DB2_MAINT	SG12 DB2 MAINT mode	
***** Bottom of data *****			

4. On the next panel, note the Automation Name field is empty. All other fields will be filled in by default.

Example 6-54 Define new entry of type ApplicationGroup

Define New Entry	
Command ==>	More: +

Define new entry of type ApplicationGroup

Entry name APPL_GROUP_DB2

Type SYSPLEX (SYSTEM SYSPLEX)
Nature BASIC (BASIC MOVE SERVER)
Default Preference . . . *DEF (0 to 3200, *DEF)
Automation Name
Automatically link . . . YES (for Application-Resources)
Behaviour ACTIVE (ACTIVE PASSIVE)

Short description . . .
Long description 1 . . .
Long description 2 . . .
Long description 3 . . .
Long description 4 . . .
Long description 5 . . .

For our environment, we use the Automation Name called APPLGR_DB8Q.
This Automation Name will be appear in the INGLIST panel and will be
displayed later in “Verify Relationships in the automation policy” on page 176.
See Example 6-55 with the updated Automation Name field.

Example 6-55 Updating the Automation Name

Define New Entry
Command ==>

More: +

Define new entry of type ApplicationGroup

Entry name APPL_GROUP_DB2

Type SYSPLEX (SYSTEM SYSPLEX)
Nature BASIC (BASIC MOVE SERVER)
Default Preference . . . *DEF (0 to 3200, *DEF)
Automation Name APPLGR_DB8Q
Automatically link . . . YES (for Application-Resources)
Behaviour ACTIVE (ACTIVE PASSIVE)

Short description . . .
Long description 1 . . .
Long description 2 . . .
Long description 3 . . .
Long description 4 . . .
Long description 5 . . .

5. Now that we have created the APPL_GROUP_DB2, we must link it to our SC64 system. On the Policy Selection panel below (Example 6-56), we select the WHERE USED option.

Example 6-56 Selecting the group to a system

Command ==>		Policy Selection	Row 2 to 13 of 13 SCROLL==> PAGE
Entry Type : ApplicationGroup		PolicyDB Name : ITS0_V3R1_PDB	
Entry Name : APPL_GROUP_DB2		Enterprise Name : ITS0TESTSC64	
Action	Policy Name	Policy Description	
	APPLGROUP INFO	Define applicationgroup information	
	APPLICATIONS	Select applications for system APG	
	TRIGGER	Select trigger	
	SERVICE PERIOD	Select service period	
	RELATIONSHIPS	Define relationships	
	-----RESOURCES-----		
	RESOURCES	Select resources and set preferences	
	GENERATED RESOURCES	List resources generated for this entry	
	MEMBER OF	List resources where this entry is a member	

s	WHERE USED	List systems linked to this entry	
	COPY	Copy data from existing entry	
*****		Bottom of data *****	

6. On the Where Used panel (Example 6-57), we select the system on which we will automate our IBM DB2 environment: SC64.

Example 6-57 Where Used

Command ==>		Where Used	Row 1 to 1 of 1 SCROLL==> PAGE
Entry Type : ApplicationGroup		PolicyDB Name : ITS0_V3R1_PDB	
Entry Name : APPL_GROUP_DB2		Enterprise Name : ITS0TESTSC64	
Action	Status	Name	Type
S	SELECTED	SC64	SYS
***** Bottom of data *****			

Defining membership for APPL_GROUP_DB2

Now it is time to establish membership to the newly created APPL_GROUP_DB2 application group. In our case, we made the IBM DB2 master subsystem (DB2_MSTR) and its children tasks (DB8QDBM1, DB8QDIST, and DB8QIRLM) part of the application group.

In order to define the membership, we perform the following steps:

1. From the Policy Database Selection panel, we select our production policy database: ITSO_V3R1_PDB.
2. On the Entry Type Selection panel, enter Option **5** for ApplicationGroup, as seen in Example 6-58.

Example 6-58 Entry Type Selection

Entry Type Selection			
Option ==> 5			
1 ENT	Enterprise	30 TMR	Timers
2 GRP	Group	31 TMO	Timeout Settings
3 SBG	SubGroup	32 TPA	Tape Attendance
4 SYS	System (*)	33 MVC	MVS Component
5 APG	ApplicationGroup (*)	34 MDF	MVSCOMP Defaults
6 APL	Application (*)	35 SDF	System Defaults
7 EVT	Events	36 ADF	Application Defaults
8 SVP	Service Periods	37 AOP	Auto Operators
9 TRG	Triggers	38 NFY	Notify Operators
10 PRO	Processor	39 NTW	Network
11 MTR	Monitor Resource (*)	40 NNT	NNT Sessions
		41 RES	Resident CLISTs
20 PRD	Product Automation	42 SCR	Status Details
		99 UET	User E-T Pairs
(*) Multi-User-Capable			

3. On the Entry Name Selection panel (Example 6-59), we select the application group APPL_GROUP_DB2.

Example 6-59 ApplicationGroup

COMMANDS ACTIONS VIEW HELP			

Command ==>		Entry Name Selection	
		Row 2 to 13 of 13	
		SCROLL==> PAGE	
Entry Type : ApplicationGroup		PolicyDB Name : ITSO_V3R1_PDB	
		Enterprise Name : ITSOTESTSC64	
Action	Entry Name	C Short Description	
S	APPL_GROUP_DB2	SC64 DB2 Group	
	APPL_GROUP_MOVE	MOVE Application Group (SYSPLEX)	
	APPL_GROUP_MSTR		
	APPL_GROUP_PROXY	Proxy_Resource_Appl_Grp	
	APPLGR_DB531	DB2 D531 definition	
	APPLGR_SGPLEX	Sysplex group - sg11 + sg12	

APPL_GROUP_01	Application group 01
CICSPAPBGRP	CICSPAPB group for OPCA0
DB2_GROUP	SG11/12 Selector
SYS1_DB2_GROUP	SYS1 DB2 Group
SYS2_DB2_GROUP	All DB2 systems on SYS2
SYS2_DB2_MAINT	SG12 DB2 MAINT mode

***** Bottom of data *****

4. On the Policy Selection panel (Example 6-60), we select the type of member, as in the following example.

Example 6-60 Policy Selection

ACTIONS HELP

Policy Selection Row 1 to 13 of 13
SCROLL==> PAGE

Command ==>

Entry Type : ApplicationGroup PolicyDB Name : ITS0_V3R1_PDB
Entry Name : APPL_GROUP_DB2 Enterprise Name : ITS0TESTSC64

Action	Policy Name	Policy Description
	DESCRIPTION	Enter description
	APPLGROUP INFO	Define applicationgroup information
S	APPLICATIONS	Select applications for system APG
	TRIGGER	Select trigger
	SERVICE PERIOD	Select service period
	RELATIONSHIPS	Define relationships
	-----RESOURCES-----	
	RESOURCES	Select resources and set preferences
	GENERATED RESOURCES	List resources generated for this entry
	MEMBER OF	List resources where this entry is a member

	WHERE USED	List systems linked to this entry
	COPY	Copy data from existing entry

5. On the Applications for ApplicationGroup panel (Example 6-61), we select the applications to be members of the application group, as in the following example.

Example 6-61 Applications For ApplicationGroup

COMMANDS ACTIONS VIEW HELP

Applications for ApplicationGroup Row 14 to 26 of 117
SCROLL==> PAGE

Command ==>

Entry Type : ApplicationGroup PolicyDB Name : ITS0_V3R1_PDB
Entry Name : APPL_GROUP_DB2 Enterprise Name : ITS0TESTSC64

Action	Status	Application
S		DB2Q
S		DB8QDBM1
S		B8QDIST
S		DB8QIRLM

6. The following panel (Example 6-62) displays to show our selections to this group.

Example 6-62 Selections for Applications for ApplicationGroup panel

COMMANDS	ACTIONS	VIEW	HELP

Applications for ApplicationGroup			Row 14 to 26 of 117
Command ==>			SCROLL==> PAGE
Entry Type : ApplicationGroup		PolicyDB Name : ITSQ_V3R1_PDB	
Entry Name : APPL_GROUP_DB2		Enterprise Name : ITS0TESTSC64	
Action	Status	Application	
	SELECTED	DB2Q	
	SELECTED	DB8QDBM1	
	SELECTED	DB8QDIST	
	SELECTED	DB8QIRLM	

All the applications are now selected and part of the APPL_GROUP_DB2 application group.

Per the *IBM Tivoli System Automation for z/OS V3.1 Customizing and Programming*, SC33-8260-01, manual guidelines, you can now create the automation control file (ACF) using the **BUILDF** command. Once you build the ACF, the program to program interface (PPI) AOFASSI, Automation Manager INGEAMSA, and Automation Agent AOFAPPL can be started.

The ACF will be read into cache at Automation Manager INGEAMSA cold startup.

The Agent must be started up last to receive the AOF603D message after the Automation Manager INGEAMSA is initialized.

6.3.7 Verify Relationships in the automation policy

In this section, we verify the relationship definitions of our IBM DB2 automation policy. These relationships were defined as dependencies and represent a hierarchy of resources defined in our policy.

Now we are ready to log on to our IBM Tivoli System Automation for z/OS domain named SC64N as we present in the following steps.

1. Example 6-63 represents the NetView logon screen of our environment.

Example 6-63 NetView Logon screen

```

NN  NN                      VV      VV
NNN NN  EEEEE TTTTTT  VV      VV  II  EEEEE WW      WW  TM
NNNN NN  EE      TT      VV      VV  II  EE      WW      W  WW
NN NN NN  EEEE     TT      VV      VV  II  EEEE     WW  WWW  WW
NN  NNNN  EE      TT      VV  VV      II  EE      WWW  WWW
NN  NNN  EEEEE   TT      VVV      II  EEEEE   WW  WW
NN  NN
                      V

5697-ENV (C) Copyright IBM Corp.      1986, 2002 - All Rights Reserved
U.S. Government users restricted rights - Use, duplication, or disclosure
restricted by GSA ADP schedule contract with IBM corporation.
Licensed materials - Property of IBM Corporation
Domain = SC64N                      NetView V5 - 64

OPERATOR ID ==>  Tiv001 or LOGOFF
PASSWORD ==>
PROFILE ==>      Profile name, blank=default
HARDCOPY LOG ==> device name, or NO, default=NO
RUN INITIAL COMMAND ==> YES or NO, default=YES
Takeover session ==> YES, NO or FORCE, default=NO

```

Enter logon information or PF3/PF15 to logoff

2. Since we are logged onto NetView, we get the main menu panel shown in Example 6-64.

Example 6-64 NetView main menu

```

CNMINETV                      Tivoli NetView for z/OS Version 5                      Main Menu

Operator ID = TIV001 Application = SC64N02C

Enter a command (shown highlighted or in white) and press Enter.

Browse Facility                BROWSE command
Command Facility              NCCF command
News                          NEWS command
PF Key Settings               DISPFK command
Help Facility                 HELP command
Index of help topics          INDEX command
Help Desk                     HELPDESK command

```

To log off or disconnect LOGOFF command or DISC command

TO SEE YOUR KEY SETTINGS, ENTER 'DISPFK'

Action====> aoc

3. On the Action prompt, we use the **AOC** command to access the IBM Tivoli System Automation for z/OS main menu as shown in Example 6-65.

Example 6-65 IBM Tivoli System Automation for z/OS Main Menu

AOFPOPER		SA z/OS - System Operations		
Domain ID = SC64N		----- MAIN MENU -----		Date = 08/31/05
Operator ID = TIV001				Time = 11:04:31
Select	Description	Component		
1	Operator Interface	OPER		
2	Command Dialogs	CD		
3	Status Display Facility	SDF		
I	IMS Automation Feature	IMS		
C	CICS Automation Feature	CICS		
O	OPC Automation Feature	OPC		
L	User defined Local Functions			
SA z/OS Version 3 Release 1				
Licensed Materials - Property of IBM				
5698-SA3 (C) Copyright IBM Corp. 1990, 2005 All Rights Reserved.				
Command ==> 3				
PF1=Help	PF2=End	PF3=Return	PF6=Roll	PF12=Retrieve

4. We use Option **3** to access the Status Display Facility (SDF). This shows all our applications that IBM Tivoli System Automation for z/OS V3.1 controls and monitors. See Example 6-66.

Example 6-66 SDF panel: DISPSTAT

A0FKSTA5	SA z/OS - Command Dialogs	Line 1	of 17
----------	---------------------------	--------	-------

```

Domain ID   = SC64N      ----- DISPSTAT -----      Date = 08/31/05
Operator ID = TIV001      Time = 11:09:35
A ingauto  B setstate  C ingreq-stop  D thresholds  E explain  F info  G tree
H trigger  I service  J all children  K children  L all parents  M parents
CMD  RESOURCE      STATUS      SYSTEM      JOB NAME  A I S R D RS TYPE      Activity
---  -
AOFASSI      UP          SC64        AOFASSI     Y Y Y Y Y Y MVS      --none--
APPC          UP          SC64        APPC        Y Y Y Y Y Y MVS      --none--
ASCH          UP          SC64        ASCH        Y Y Y Y Y Y MVS      --none--
DB8Q          UP          SC64        DB8QMSTR    Y Y Y Y Y Y MVS      --none--
DB8QDBM1     UP          SC64        DB8QDBM1    Y Y Y Y Y Y MVS      --none--
DB8QDIST     UP          SC64        DB8QDIST    Y Y Y Y Y Y MVS      --none--
DB8QIRLM     UP          SC64        DB8QIRLM    Y Y Y Y Y Y MVS      --none--
HSAMPROC     UP          SC64        HSAMSC64    Y Y Y Y Y Y MVS      --none--
G JES2          UP          SC64        JES2        Y Y Y Y Y Y MVS      --none--
LLA          UP          SC64        LLA         Y Y Y Y Y Y MVS      --none--
RMF          UP          SC64        RMF         Y Y Y Y Y Y MVS      --none--
RMFIII       UP          SC64        RMFGAT      Y Y Y Y Y Y MVS      --none--
TSO          UP          SC64        TSO         Y Y Y Y Y Y MVS      --none--
VLF          UP          SC64        VLF         Y Y Y Y Y Y MVS      --none--
VTAM44       UP          SC64        VTAM44      Y Y Y Y Y Y MVS      --none--

```

```

Command ==>
PF1=Help    PF2=End      PF3=Return   PF4=INGLIST  PF5=Filters  PF6=Roll
PF7=Back    PF8=Forward   PF9=Refresh  PF10=Previous PF11=Next    PF12=Retrieve

```

- To check the dependencies of our IBM DB2 applications, we enter the command **G** to JES2. This displays the hierarchy of the IBM DB2 applications starting from JES2. See Example 6-67.

Example 6-67 IBM DB2 application dependencies

```

AOFKTRREE      SA z/OS  - Command Dialogs      Line 1    of 19
Domain ID   = SC64N      ----- DISPTREE -----      Date = 08/31/05
Operator ID = TIV001      Time = 11:24:52

Subsystem ==>  JES2      System ==>  SC64      Dependency ==>  START

JES2
+-- DB8Q
|   +-- DB8QDBM1
|   +-- DB8QDIST
|   +-- DB8QIRLM
+-- VTAM44
|   +-- TCP/IP
|   +-- DB8Q
|       +-- DB8QDBM1

```

```

|      |      +--- DB8QDIST
|      |      +--- DB8QIRLM
|      +--- E2E_EAS
|      +--- E2E_ADPT

```

With these definitions, in case DB8Q (parent task) is set to be stopped, all the other children tasks: DB8QDM1, DB8QDST, DB8QIRLM will be terminated first, and, then, DB8Q will terminate. In case of a DB8Q start, all the children tasks will be started after a successful DB8Q initialization.

- To check the groups and their membership information on IBM Tivoli System Automation for z/OS V3.1, on the command line, enter **INGLIST**. See Example 6-68.

Example 6-68 Checking APPLGR_DB8Q Group

INGKYSTO			SA z/OS - Command Dialogs			Line 1 of 26	
Domain ID = SC64N			----- INGLIST -----			Date = 09/08/05	
Operator ID = TIV001			Sysplex = SANDBOX			Time = 18:28:42	
CMD: A Update		B Start	C Stop	D INGRELS	E INGVOTE	F INGINFO	
G Members		H DISPTRG	I INGSCHED	J INGGROUP	M DISPMTR	/ scroll	
CMD Name		Type System	Compound	Desired	Observed	Nature	
-----		-----	-----	-----	-----	-----	
G	AOFASSI	APL SC64	SATISFACTORY	AVAILABLE	AVAILABLE		
	APPC	APL SC64	SATISFACTORY	AVAILABLE	AVAILABLE		
	APPLGR_DB8Q	APG SC64	SATISFACTORY	AVAILABLE	AVAILABLE	BASIC	
	APPLGRBASIC	APG SC64	SATISFACTORY	AVAILABLE	AVAILABLE	BASIC	
	APPLGROUP01	APG SC64	SATISFACTORY	AVAILABLE	AVAILABLE	BASIC	
	APPLGRPMSTR	APG SC64	SATISFACTORY	AVAILABLE	AVAILABLE	BASIC	

- Now we want to view the members within the APPLGR_DB8Q. Enter **G** (for Members). The result displays in Example 6-69.

Example 6-69 APPLGR_DB8Q members

INGKYSTO			SA z/OS - Command Dialogs			Line 1 of 4	
Domain ID = SC64N			----- INGLIST -----			Date = 09/08/05	
Operator ID = TIV001			Sysplex = SANDBOX			Time = 18:36:16	
CMD: A Update		B Start	C Stop	D INGRELS	E INGVOTE	F INGINFO	
G Members		H DISPTRG	I INGSCHED	J INGGROUP	M DISPMTR	/ scroll	
CMD Name		Type System	Compound	Desired	Observed	Nature	
-----		-----	-----	-----	-----	-----	
DB8Q		APL SC64	SATISFACTORY	AVAILABLE	AVAILABLE		
DB8QDBM1		APL SC64	SATISFACTORY	AVAILABLE	AVAILABLE		
DB8QDIST		APL SC64	SATISFACTORY	AVAILABLE	AVAILABLE		
DB8QIRLM		APL SC64	SATISFACTORY	AVAILABLE	AVAILABLE		

6.4 End-to-end Automation Adapter configuration

As presented in Chapter 1, “IBM Tivoli System Automation for Multiplatforms V2.1” on page 3, you can use IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component to automate the operation of resources across heterogeneous environments (called first-level automation domains), which is managed by an automation manager such as IBM Tivoli System Automation for Multiplatforms V2.1 Base Component or IBM Tivoli System Automation for z/OS V3.1.

An End-to-end Automation Adapter connects each first-level automation domain to the End-to-end Automation Manager.

We must note that there can be only *one* End-to-end Automation Adapter per first-level automation domain. In the context of IBM Tivoli System Automation for z/OS V3.1, that means only one End-to-end Automation Adapter per IBM Tivoli System Automation for z/OS V3.1 z/OS sysplex group.

Although there can be more than one IBM Tivoli System Automation for z/OS V3.1 domain in a sysplex, there can be only one End-to-end Automation Adapter per z/OS system. In addition, the End-to-end Automation Adapter *must run on the same system* as the primary IBM Tivoli System Automation for z/OS V3.1 Agent.

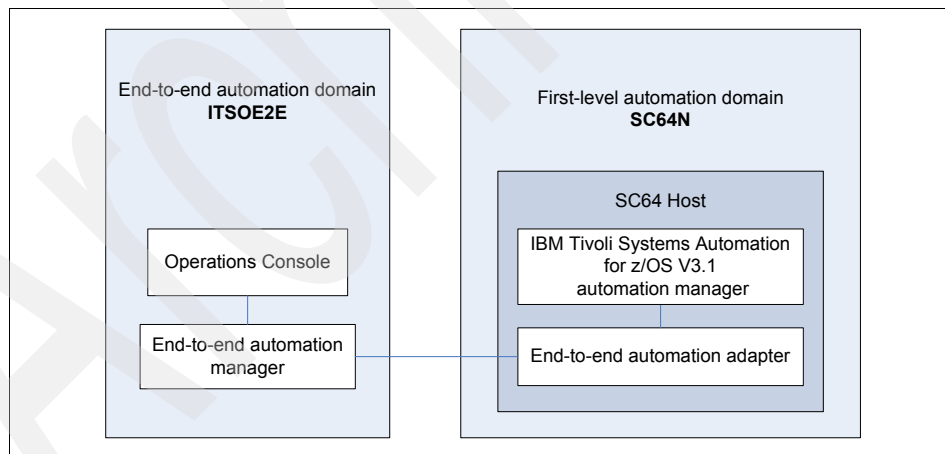


Figure 6-2 End-to-end automation domain and SC64N interaction

When the End-to-end Automation Adapter runs on a z/OS system, the IBM Tivoli System Automation for z/OS V3.1 Agent on that z/OS system is the *primary automation agent*.

The End-to-end Automation Adapter registers with the primary automation agent at startup time. The End-to-end Automation Adapter also subscribes with the primary automation agent to enable communication channels. The primary automation agent then enables the End-to-end Automation Adapter to communicate with the primary IBM Tivoli System Automation for z/OS V3.1 Automation Manager.

After a system failure, event subscriptions are lost and the End-to-end Automation Adapter has to re-subscribe to the primary automation agent.

In our case study scenario environment, we have a single z/OS system. In this case, the End-to-end Automation Adapter primary automation agent and the primary IBM Tivoli System Automation for z/OS V3.1 Automation Manager are all running on the same z/OS system. See Figure 6-3.

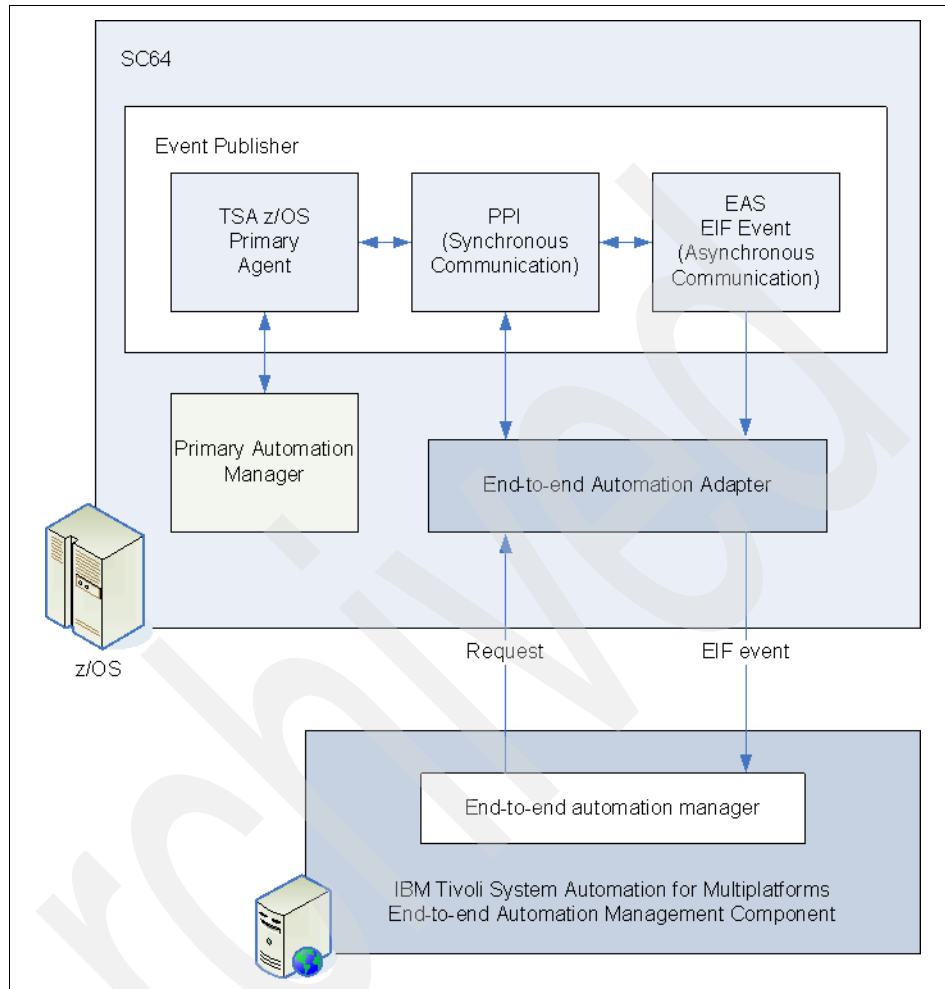


Figure 6-3 End-to-end Automation Adapter communication

Per the *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01, manual, setting up the IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter in our environment requires the following configuration steps:

- ▶ “Check prerequisites and dependencies” on page 184
- ▶ “Configure NetView and IBM Tivoli System Automation for z/OS” on page 184
- ▶ “Enabling the Event Automation Service” on page 186
- ▶ “Configure the Global Initialization File” on page 186
- ▶ “Configure the NetView Message Adapter Service” on page 187

- ▶ “Customize the End-to-end Automation Adapter” on page 190
- ▶ “Perform configuration for security” on page 196
- ▶ “Verify startup of the Automation Adapter” on page 197
- ▶ “Solve timeout problems” on page 198

6.4.1 Check prerequisites and dependencies

In order to implement the End-to-end Automation Adapter on IBM Tivoli System Automation for z/OS V3.1, you must satisfy the following prerequisites and dependencies:

- ▶ IBM Tivoli System Automation for z/OS V3.1
- ▶ IBM Tivoli NetView V5.1
- ▶ z/OS 1.3 or higher
- ▶ Java Runtime Environment (JRE) 1.4.2 installed on z/OS
- ▶ The JRE Software Development Kit (SDK)
- ▶ The Event Automation Service (EAS) and Message Adapter Service components of NetView must be configured
- ▶ Full z/OS UNIX System Services (USS) with hierarchical file system
- ▶ SSI address space with PPI function of NetView
- ▶ TCP/IP

6.4.2 Configure NetView and IBM Tivoli System Automation for z/OS

The IBM Tivoli System Automation for z/OS V3.1 communication task INGPXDST has an initialization member DSIPARM (INGXINIT) that we use to specify End-to-end Automation Adapter-specific parameters. In our case study scenario, we add the following parameters to this member:

- ▶ GRPID=XY

This parameter must be set for the XCF group ID for IBM Tivoli System Automation for z/OS V3.1. This ID must be the same as in the automation adapter plug-in configuration.

- ▶ PPI=YES

This parameter is used for the End-to-end Automation Adapter.

- ▶ PPIBQL=1500

This parameter controls the input request flow into this buffer queue.

For further details about the above parameters, refer to the *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01, manual.

Automated operator functions

End-to-end Automation Adapter uses dedicated automated operator functions in the primary agent to execute requests and to forward events.

The automated operator functions E2EOPER, and E2EOPR01 through E2EOPRNN, are optional for the End-to-end Automation Adapter configuration. If defined, they are used to execute requests from the End-to-end Automation Adapter.

You *must* define the automated operator function EVTOPER. It is used to forward events to the End-to-end Automation Adapter. If not defined, the initialization of the End-to-end Automation Adapter fails.

We use the customization dialogs to define the automated operator functions for the End-to-end Automation Adapter in entry type Auto Operators of the IBM Tivoli System Automation for z/OS V3.1 policy database.

Note that the names of the Automation Operators must match those defined in the DSIPARM data set. The following operator group had to be selected to our z/OS system SC64.

- ▶ EVT_PUBLISHER - Event Publisher
- ▶ E2E_AUTOOPS - Automated Functions for End-to-end Automation Adapter

For further details about automated operators, refer to *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01.

Example 6-70 shows the output of the DISPAOPS command in our environment showing the defined automated operator functions.

Example 6-70 DISPAOPS command response

```
AOFK2S0                      SA z/OS - Command Dialogs      Line 14 of 38
Domain ID = SC64N            ----- DISPAOPS -----      Date = 09/06/05
Operator ID = Tiv001 Time = 14:05:39
```

System	Automated Function	Primary	Status	Secondary	Status
-----	-----	-----	-----	-----	-----
SC64	EVTOPER	AUTEVT1	ACTIV	AUTEVT2	ACTIV
SC64	E2EOPER	AUTE2E	ACTIV		
SC64	E2EOPR01	AUTE2E01	ACTIV		
SC64	E2EOPR02	AUTE2E02	ACTIV		

Command ==>

PF1=Help

PF2=End

PF3=Return

PF6=Roll

PF7=Back

PF8=Forward

PF9=Refresh

PF12=Retrieve

6.4.3 Enabling the Event Automation Service

The Event Automation Service (EAS) can be started either with a job from an MVS system console, or from a UNIX System Service command shell. In our case study scenario, we automate the EAS start on our IBM Tivoli System Automation for z/OS V3.1. Our EAS resource name is called E2E_EAS and the MVS job name is called AOFAEVNT in our case study.

You can find a sample for starting EAS as a job located in NETVIEW.V5R1M0.SCNMUXMS as the member IHSAEVNT. The initialization files are assumed to be located in a data set allocated to DD name IHSSMP3. We follow the description within the member for configuration.

The startup parameters have to be provided in form of the following initialization files, and we discuss configuration in the following sections.

- ▶ Global initialization file
- ▶ Message adapter configuration file

Perform the following updates to the sample to meet our requirements for our installation in the start procedure AOFAEVNT (IBM default member IHSAEVNT):

- ▶ If you do not use the default name IHSAINIT for the global initialization file, pass the name of your file via the parameter INITFILE.
- ▶ If you do not use the default name IHSAMCFG for the message adapter configuration file, pass the name of your file via the parameter MSGCFG.
- ▶ In the DD statement, specify the data set names of your installation.

6.4.4 Configure the Global Initialization File

The default member name for the global initialization file is IHSAINIT. This is found in the IBM-supplied member NETVIEW.V5R1M0.SCNMUXMS.

This member was copied into our data set name NETUSER.SCNMUXMS for IBM Tivoli System Automation for z/OS V3.1 job AOFAEVNT. We follow the description within the member for configuration.

Make sure that the NetView message adapter service is also started when starting EAS. This is done by commenting out the following statement:

```
NOSTART TASK=MESSAGEA
```

Specify the Program to Program Interface (PPI) called INGEV2E2 for the receiver.

```
PPI=INGEVE2E
```

We change the IHSAINIT initialization file so that only the necessary tasks for our environment are started at initialization time. Example 6-71 shows the configuration in the NETVUSER.SCNMUXCL(IHSAINIT) member. We show here only the lines that are relevant to our scenario. All the missing lines are commented out.

Example 6-71 Message adapter task

```
# PPI Receiver ID
PPI=INGEVE2E
# Tasks not started at initialization
NOSTART TASK=ALERTA
#NOSTART TASK=MESSAGEA
NOSTART TASK=EVENTRCV
NOSTART TASK=ALRTTRAP
NOSTART TASK=TRAPALRT
```

For further information, refer to the *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01, manual.

6.4.5 Configure the NetView Message Adapter Service

The default member name for the global initialization file is IHSAMCFG. We copy this member into our data set name NETUSER.SCNMUXMS.

In our environment, the NetView message event adapter configuration file is a member named IHSAMCFG. We must change the following in this member:

1. We specify the address 127.0.0.1 as the server host name.
2. We keep the port number on which the End-to-end Automation Adapter responds as default 5529.
3. As for connection mode, we set it to `connection_oriented`. This allows for the connection to be established at initialization time and closed at End-to-end Automation Adapter termination time.
4. We keep the default value of 4096 for the maximum event size.

5. We specify the name of the NetView message adapter format file as INGMFMTE. The version of this file that is to be used by end-to-end automation is delivered in ING.SINGSAMP(INGMFMTE). No configuration is required.

We have a user-defined data set named NETUSER.SCNMUXCL of which the IHSAMCFG is a member. This data set is included in the NetView message adapter service startup procedure.

Example 6-72 shows the configuration parameters used in our NETUSER.SCNMUXCL(IHSAMCFG) member.

Example 6-72 IHSAMCFG member settings

```

ServerLocation=127.0.0.1
ServerPort=5529
ConnectionMode=connection_oriented
BufferEvents=yes
BufferEventsLimit=0
BufferFlushRate=0
BufEvtPath=/etc/Tivoli/tec/cache_nv390msg
BufEvtMaxSize=64
BufEvtShrinkSize=8
BufEvtRdBlkLen=64
EventMaxSize=4096
AdapterFmtFile=INGMFMTE
FilterMode=out

```

For further details, refer to *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01.

Example 6-73 shows the NetView message adapter service startup procedure: AOFAEVNT.

Example 6-73 AOFAEVNT startup procedure

```

//EASGO  PROC  PROG=IHSAC000,  ** EVENT/AUTOMATION SERVICE
//          REG=32M,          ** REGION SIZE IN K FOR MAIN TASK
//          TCPDATA='',       ** SET FOR TCPIP.DATA EXPLICIT ALLOC
//          MSGCFG=,          ** MESSAGE ADAPTER CONFIG FILE
//          ALRTCFG=,         ** ALERT ADAPTER CONFIG FILE
//          ERCVCFG=,         ** EVENT RECEIVER CONFIG FILE
//          TALRTCFG=,        ** TRAP TO ALERT CONV. CONFIG FILE
//          ALRTTCFG=,        ** ALERT TO TRAP CONV. CONFIG FILE
//          PPI=INGEVE2E,     ** PPI RECEIVER ID
//          INITFILE=,        ** EVENT AUTOAMTION SVC INIT FILE
//          OUTSIZE=,         ** TRACE/ERROR WRAP SIZE IN KBYTES
//          OELINE=           ** SPECIFY OE-STYLE PARAMETERS
//          *
//*****

```

```

/*
//STEP1 EXEC PGM=&PROG,TIME=1440,REGION=&REG,
//          PARM=(' /MSGCFG=&MSGCFG ALRTCFG=&ALRTCFG ERVCFG=&ERVCFG*
//          TALRTCFG=&TALRTCFG ALRTTCFG=&ALRTTCFG PPI=&PPI      *
//          INITFILE=&INITFILE OUTSIZE=&OUTSIZE &OELINE')
/*
//STEPLIB DD DSN=NETVIEW.SCNMUXLK,DISP=SHR
/* UNCOMMENT THE FOLLOWING LINE IF YOU WILL BE USING LE/370 LIBRARIES
/* DD DSN=CEE.V1R7M0.SCEERUN,DISP=SHR
/*
/*  INITIALIZATION PARAMETERS AND CONFIGURATION FILES DATASET
//IHSSMP3 DD DSN=NETVUSER.SCNMUXCL,DISP=SHR
//          DD DSN=NETVIEW.SCNMUXCL,DISP=SHR
/*
/*  EAS MESSAGES DATASET
//IHMSG1 DD DSN=NETVIEW.SCNMUXMS,DISP=SHR
/*
/*  EAS OUTPUT DATASETS
//IHSC DD SYSOUT=A
//IHSM DD SYSOUT=A
//IHSA DD SYSOUT=A
//IHSE DD SYSOUT=A
//IHST DD SYSOUT=A
//IHSL DD SYSOUT=A
//IHSCS DD SYSOUT=A
//IHSMS DD SYSOUT=A
//IHSAS DD SYSOUT=A
//IHSES DD SYSOUT=A
//IHSTS DD SYSOUT=A
//IHSLs DD SYSOUT=A
//IHSCSTD DD SYSOUT=A
//IHSASTD DD SYSOUT=A
//IHSMSTD DD SYSOUT=A
//IHSESTD DD SYSOUT=A
//IHSTSTD DD SYSOUT=A
//IHSLSTD DD SYSOUT=A
/*
/*  TCP/IP EXPLICIT ALLOCATION OF TCPIP.DATA
/*  NOTE: REMOVE THE COMMENT FROM THE FOLLOWING STATEMENT AND
/*  SET THE TCPIPDATA SUBSTITUTION VARIABLE TO EXPLICITLY
/*  ALLOCATE TCPIP.DATA
/*SYSTCPD DD DISP=SHR,DSN=&TCPDATA
/*
/*  RUNTIME LIBRARY MESSAGES
//SYSTEM DD SYSOUT=A
/*
/*  NOTE: REMOVE THE COMMENTS FROM THE NEXT TWO STATEMENTS ALONG WITH
/*  THE JOBCARD WHEN MAKING THIS A BATCH JOB.
/*  PEND

```

6.4.6 Customize the End-to-end Automation Adapter

After we install the necessary prerequisites, we are now ready to configure the End-to-end Automation Adapter.

The End-to-end Automation Adapter SMP/E installation creates a default structure for the various files that are associated with the End-to-end Automation Adapter, as follows:

/usr/lpp/ing/adapter	This contains the executable files, for example, the End-to-end Automation Adapter start and stop scripts.
/usr/lpp/ing/adapter/config	This contains configuration files, for example, the master configuration file.
/usr/lpp/ing/adapter/data	Working files, for example, the cache and log files. This directory is initially empty.
/usr/lpp/ing/adapter/lib	It contains JAR files and DLLs for the End-to-end Automation Adapter.
/usr/lpp/ing/adapter/ssl	Security certificates. This directory is initially empty.

To customize the End-to-end Automation Adapter, we perform configurations on the following elements:

- ▶ “RACF authorization for z/OS UNIX privileges” on page 190
- ▶ “The End-to-end Automation Adapter shell script” on page 191
- ▶ “The link list” on page 193
- ▶ “The End-to-end Automation Adapter master configuration file” on page 193
- ▶ “The End-to-end Automation Adapter plug-in configuration file” on page 194
- ▶ “The End-to-end Automation Adapter JAAS configuration file” on page 195

The following sections go into the details of each of the above configuration steps for our case study environment.

RACF authorization for z/OS UNIX privileges

You can define profiles in the UNIXPRIV class to grant RACF® authorization for certain z/OS UNIX privileges. These privileges are automatically granted to all users with z/OS UNIX superuser authority. By defining profiles in the UNIXPRIV class, you may specifically grant certain superuser privileges with a high degree

of granularity to users who do not have superuser authority. This allows you to minimize the number of assignments of superuser authority at your installation and reduces your security risk.

Since our case study scenario is in a controlled environment, we decide to grant our RACF user ID with superuser authority.

Refer to the *Security Configuration in a TCP/IP Sysplex Environment*, SG24-6527-00, Redbook for instructions on how to manipulate security privileges for your environment.

The End-to-end Automation Adapter shell script

If you want to use your own hierarchical file system (HFS) or a shared HFS, you have to modify the End-to-end Automation Adapter start script and its configuration files.

In our case scenario according to *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01, we update the `ingadapter.sh` *USS shell script* for our environment.

We changed the following parameters in the `ingadapter.sh` script for our case study scenario. Entries that are not mentioned in the following list are not modified and are left with their default values.

- ▶ We comment out the `SSL_PASSWD` parameter because we are not using the `GenerateSampleKeys` function.
- ▶ We set the filenames for the standard output and error streams using the `REDIRSTDOUT` and `REDIRSTDERR` entries in the `ingadapter.sh` script to match the End-to-end Automation Adapter start up procedure script on z/OS (AOFAADPT). The entries in the AOFAADPT are as follows.

```
//STDOUT DD PATH='/usr/lpp/ing/adapter/data/logs/stdout.log',  
//STDERR DD PATH='/usr/lpp/ing/adapter/data/logs/stderr.log',
```
- ▶ We specify the path to JAVA runtime in our installation.

Example 6-74 shows the `ingadapter.sh` for our case study scenario. Changed attributes are marked in bold.

Example 6-74 End-to-end Automation Adapter startup script

```
#-----  
# Customization section begins here  
#-----  
# Root of the automation adapter installation  
INSTALL_DIR=/usr/lpp/ing/adapter  
  
# Password for the truststore generated by function GenerateSampleKeys.
```

```

# Needed only if it is planned to generate sample keys.
# SSL_PASSW=passphrase

# Redirection of stdout and stderr as specified in sample proc SAM(INGXADPT).
# Specify only filename without path.
REDIRSTDOUT=/usr/lpp/ing/adapter/data/logs/stdout.log
REDIRSTDERR=/usr/lpp/ing/adapter/data/logs/stderr.log

# Customize debugging DLLs (only if IBM service requires you to do so)
export INGXDEBUG=1,2,1

#-----
# Additional customization needed if your installation differs from
# the default installation, e.g. different config or data directory
#-----

# Java runtime. Customize path in order to invoke java runtime.
JAVA=/usr/lpp/java/J1.4/bin/java
# Java SDK. Customize path in order to invoke java keytool (optional).
JAVA_KEYTOOL=keytool

# Data directory must match eez-data-directory within ing.adapter.properties
# and must match the log path in ing.adapter.jlogdir.properties
# Directory access must be read-write
DATA_DIR=$INSTALL_DIR/data

# Used by function CreateSampleKeys in order to create and store
# the keystore and trustore files. See also ing.adapter.ssl.properties
# Directory access must be read-write otherwise keys remains in data directory
SSL_DIR=$INSTALL_DIR/ssl

# Configuration directory, where all config files are stored
CONFIG_DIR=$INSTALL_DIR/config

# Master config file for the automation adapter.
# The suffix will be added automatically if required.
ADAPTER_CONFIG=$CONFIG_DIR/ing.adapter${SUFFIX}.properties

# Java Logger configuration file name and
# file name that containst the path to the log-directory
JLOG_FILENAME_LOGPATH=ing.adapter.jlogdir.properties
JLOG_FILENAME=ing.adapter.jlog.properties
JLOG_CONFIG_DIR=$CONFIG_DIR

# Other configuration files
JAAS_CONFIG=$CONFIG_DIR/ing.adapter.jaas.properties
SSL_CONFIG=$CONFIG_DIR/ing.adapter.ssl.properties

#-----

```



```
# Customization section ends here.
```

```
#-----
```

The link list

We add the following libraries to the link list in our case study environment:

- ▶ SYS1.SCLBDLL2
- ▶ SYS1.SCEERUN
- ▶ SYS1.SCEERUN2
- ▶ SYS1.SCLBDLL

You can check that these libraries have been linked using the following command: `D PROG, LNKLST`

The End-to-end Automation Adapter master configuration file

The master configuration file for the End-to-end Automation Adapter is specified in the `ingadapter.sh` script as follows:

```
# Master config file for the automation adapter.  
# The suffix will be added automatically if required.  
ADAPTER_CONFIG=$CONFIG_DIR/ing.adapter${SUFFIX}.properties
```

Since we do not use a suffix in our environment, we have to configure the file `/usr/lpp/ing/adapter/config/ing.adapter.properties`.

We change the following parameters in the End-to-end Automation Adapter master configuration file for our case study scenario. Entries that we do not mention here in the following list were not modified and were left with their default values.

- ▶ The IP address of our z/OS system on which the End-to-end Automation Adapter receives requests from the End-to-end Automation Management Component. This is specified in the **eez-remote-contact-hostname** entry.
- ▶ We set **eez-operator-authentication** to `false` because we are using the default IBM Tivoli System Automation for z/OS V3.1 JAAS login modules for authentication.
- ▶ We specify the fully qualified hostname of the server on which the End-to-end Automation Management Component runs in our environment using the **eif-send-to-hostname** entry.

Example 6-75 shows the End-to-end Automation Adapter master configuration file for our case study scenario. Changed attributes are marked in bold.

Example 6-75 End-to-end Automation Adapter master configuration file

```
# --- Adapter Configuration -----
```

```

eez-remote-contact-hostname = 9.12.6.9
eez-remote-contact-port    = 2001
eez-remote-contact-over-ssl = false
eez-operator-authentication = false
eez-initial-contact        = true
eez-max-connections        = 3
eez-data-directory         = ./data

# --- EIF Configuration -----

eif-cache                = true
eif-cache-size           = 500
eif-retry-interval-seconds = 30
eif-send-to-hostname      = tsa011.itsc.austin.ibm.com
eif-send-to-port          = 2002
eif-receive-from-hostname = 127.0.0.1
eif-receive-from-port     = 5529

# --- Plugin Configuration -----

plugin-configfile-sa4zos = ./config/ing.adapter.plugin.properties

```

The End-to-end Automation Adapter plug-in configuration file

The master configuration file contains the location of the End-to-end Automation Adapter plug-in configuration file, as follows:

```
plugin-configfile-sa4zos = ./config/ing.adapter.plugin.properties
```

We change the following parameters in the End-to-end Automation Adapter plug-in configuration file for our case study scenario. Entries that are not mentioned in the following list were not modified and were left with their default values.

- We set the XCF group ID of the our z/OS as per configurations in INGXINIT using the **GRPID** entry.

Note: The Automation manager PARMLIB (HSAMPRM00) on z/OS must also be updated as GRPID=XY.

- We change the number of elements in the PPI queue with the **PPIBQL** entry.
- Since our z/OS system resides in a remote data center, we change the value of the **TIMEOUT** entry.

- We decided to change the **plugin-domain-name** entry to SC64N because this is our IBM Tivoli System Automation for z/OS NetView domain ID.

Here you can decide your own naming strategy or set it to ? and IBM Tivoli System Automation for z/OS V3.1 uses the XCF sysplex group name.

Example 6-76 shows the End-to-end Automation Adapter plug-in configuration file for our case study scenario. Changed attributes are marked in bold.

Example 6-76 End-to-end Automation Adapter plug-in configuration file

```
# --- Specific settings for the SA/Netview communication ---
# --- Modify these parameters to your needs ---

GRPID    = XY
PPIBQL  = 3000
AUTOPFN  = E2EOPER
TIMEOUT = 4447
CODEPAGE= Cp1047

# --- Domain name may be modified or omitted ---

plugin-domain-name = SC64N

# --- Do not modify these plugin settings ---
plugin-impl-class      = com.ibm.ing.sam.INGXPlugin
plugin-impl-class-singleton = true
plugin-event-classes   = SystemAutomation_Resource_Status_Change
                        SystemAutomation_Domain_Status_Change SystemAutomation_Resource_Conf
                        igation_Change SystemAutomation_Request_Configuration_Change
                        SystemAutomation_Relationship_Configuration_Change
plugin-auto-start      = true
```

The End-to-end Automation Adapter JAAS configuration file

The JAAS configuration file for the End-to-end Automation Adapter is specified in the ingadapter.sh script as follows:

```
# Other configuration files
JAAS_CONFIG=$CONFIG_DIR/ing.adapter.jaas.properties
```

This file specifies two entries for a login module that is required, because in our environment all incoming requests to the End-to-end Automation Adapter will be authenticated using JAAS.

We have not changed this file and used all default values. See Example 6-77. In our environment, the JAAS configuration file is located at:
`/usr/lpp/ing/adapter/config/ing.adapter.jaas.properties`

Example 6-77 The JAAS configuration file and adapter.jaas.properties

```
EEZAdapterDefaultLogin {  
  
    com.ibm.eez.adapter.EEZAdapterDefaultLoginModule required ;  
};  
  
EEZAdapterLogin {  
  
    com.ibm.security.auth.module.OS390LoginModule required;  
};
```

6.4.7 Perform configuration for security

When the End-to-end Automation Management Component issues a request to the End-to-end Automation Adapter, there is always a user ID and password associated with the request. You need to consider two security aspects:

Authentication

In our environment, the End-to-end Automation Adapter then performs authentication checks using RACF using the login modules specified in the JAAS configuration file, described in “The End-to-end Automation Adapter JAAS configuration file” on page 195.

This decision is made at End-to-end Automation Adapter startup time in the master configuration file `eez-operator-authentication=false`.

There are other options for setting up authentication security. Refer to the *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01, manual for details.

Authorization

In case the user ID authentication succeeds, the End-to-end Automation Adapter performs an authorization check for each of the commands in the request that are to be executed.

The authorization check is done by an authorization user exit, which is an external REXX program that *must* be named AOFEXE2E. If no authorization user exit exists, the user ID associated with the request is considered to be authorized for each request.

We decide not to create our own authorization user exit, but decided that any request to be considered granted would be a security hazard. We decided to use the sample that is provided by IBM Tivoli System Automation for z/OS V3.1 as the member AOFEXE2E in the sample data set SINGSAMP. Refer to the *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01, for details about how to create your own authentication user exit or to use the AOFEXE2E sample.

6.4.8 Verify startup of the Automation Adapter

Once we complete all the customization, we are ready to start the End-to-end Automation Adapter for z/OS.

Example 6-78 displays the status of E2E_EAS and E2E_ADPT resources on NetView. The status of these resources are in CTLDOWN.

We need to change the status of these resources to AUTODOWN, starting with the E2E_ADPT first and then E2E_EAS second. This will start both resources in the proper order.

Example 6-78 Status of resources E2E_EAS and E2E_ADPT

AOFKSTA5		SA z/OS - Command Dialogs		Line 1 of 18	
Domain ID = SC64N		----- DISPSTAT -----		Date = 09/15/05	
Operator ID = TIV001				Time = 13:59:46	
A ingauto B setstate C ingreq-stop D thresholds E explain F info G tree					
H trigger I service J all children K children L all parents M parents					
CMD	RESOURCE	STATUS	SYSTEM	JOB NAME	Activity
---	-----	-----	-----	-----	-----
	AOFASSI	UP	SC64	AOFASSI	--none--
	APPC	UP	SC64	APPC	--none--
	ASCH	UP	SC64	ASCH	--none--
	DB8Q	UP	SC64	DB8QMSTR	--none--
	DB8QDBM1	UP	SC64	DB8QDBM1	--none--
	DB8QDIST	UP	SC64	DB8QDIST	--none--
	DB8QIRLM	UP	SC64	DB8QIRLM	--none--
	E2E_ADPT	CTLDOWN	SC64	AOFAADPT	--none--
	E2E_EAS	CTLDOWN	SC64	AOFAEVNT	--none--
	HSAMPROC	UP	SC64	HSAMSC64	--none--
	JES2	UP	SC64	JES2	--none--
	LLA	UP	SC64	LLA	--none--
	RMF	UP	SC64	RMF	--none--

Command ==>

The name of our End-to-end Automation Adapter for our case study scenario is (AOFAADPT). This is fully automated on our IBM Tivoli System Automation for

z/OS V3.1. Refer to *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01, for details about how to automate the start of the End-to-end Automation Adapter.

Example 6-79 shows AOFAADPT application console output. We highlight key messages in bold.

Example 6-79 AOFAADPT application console output

```
$HASP100 AOFAADPT ON STCINRDR
IEF695I START AOFAADPT WITH JOBNAME AOFAADPT IS ASSIGNED TO USER STC, GROUP SYS1
$HASP373 AOFAADPT STARTED
CNM493I INMSG02 : 00000029 : AOCFILT AOFAADPT ACTIVMSG JOBNAME=AOFAADPT
IEF403I AOFAADPT - STARTED - TIME=13.41.04 - ASID=0080 - SC64
INGX9704I Preparing the environment to start the automation adapter.
EEZA0100I The adapter has been started*
Initial contact was enabled and the connection to the management server has
been established*
EEZA0101I The adapter is active*
EEZA0111I The plug-in is starting: class com.ibm.ing.sam.INGXPlugin *
INGX9802I INGX9802I INGXLogger has successfully been initialized using
configuration file ing.adapter.jlog.properties from path /usr/lpp/ing/
adapter/config.*
INGX9902I INGXPluginLogger has successfully been initialized using con
figuration file ing.adapter.jlog.properties from path /usr/lpp/ing/ada
figuration file ing.adapter.jlog.properties from path /usr/lpp/ing/ada
pter/config.*
CNM493I INMSG02 : 00003489 : ACTIVMSG UP=YES
EEZA0112I The plug-in has been started: class com.ibm.ing.sam.INGXPlugin *
EEZA0102I The adapter is ready*
```

Note: You must define the automated operator function EVTOPER. It is used to forward events to the End-to-end Automation Adapter. If it is not defined, the initialization of the End-to-end Automation Adapter will fail.

Important: The End-to-end Automation Management Component automation manager has to be active for the End-to-end Automation Adapter to initialize with the above messages.

6.4.9 Solve timeout problems

Since we have a relatively small z/OS system, during the development of this redbook, we experienced timeout problems on the End-to-end Automation Management Component operations console when displaying IBM Tivoli System Automation for z/OS resource data from our SC64N automation domain.

As described in “End-to-end Automation Adapter configuration” on page 181, the End-to-end Automation Adapter receives the requests from the End-to-end Automation Management Component operations console.

These requests are then mapped to IBM Tivoli System Automation for z/OS commands that query the requested data from the End-to-end Automation Management Component automation manager. Before sending the query commands to the End-to-end Automation Management Component automation manager, the primary automation agent in z/OS checks the expiration time given to the requests. If the expiration time that remains is too short, the requested command will be rejected, resulting in error message **ING249E** that indicates that a task execution request timed out.

The manual *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01, provides a mapping of end-to-end automation requests to IBM Tivoli System Automation for z/OS commands. Most of these commands contain a WAIT parameter. The value of the WAIT parameter is calculated as the difference between the time when the IBM Tivoli System Automation for z/OS command was issued within the NetView environment and the expiration time given to the end-to-end automation request.

The expiration time of an end-to-end automation request is determined by both:

- ▶ The time when the corresponding end-to-end automation request was issued from the End-to-end Automation Management Component operations console or the End-to-end Automation Management Component automation manager.
- ▶ The timeout in seconds defined by an environment variable defined in the IBM WebSphere Application Server on which the End-to-end Automation Management Component automation manager runs. The environment variable must be named **com.ibm.eez.aab.invocation-timeout-seconds** and must be set to a value in seconds.

In our environment, we set the **com.ibm.eez.aab.invocation-timeout-seconds** to 2500 seconds as you can see in Figure 6-4.

WebSphere Variables

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

☒ Scope: Cell=**tsa011Node01Cell**, Node=**tsa011Node01**

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, [see the scope settings help](#).

Cell

→ Node

Server

☒ Preferences

Maximum rows

☐ Retain filter criteria.

Select	Name	Value	Scope
<input type="checkbox"/>	WPS_PATH_SEPARATOR	:	cells:tsa011Node01Cell:nodes:tsa011Node01
<input type="checkbox"/>	com.ibm.eez.aab.invocation-timeout-seconds	2500	cells:tsa011Node01Cell:nodes:tsa011Node01

Page: 8 of 8 Total 37

Figure 6-4 *com.ibm.eez.aab.invocation-timeout-seconds variable definition*

This solved our timeout issues. Set the value of the **com.ibm.eez.aab.invocation-timeout-seconds** variable to fit your environment.

Case study scenario: End-to-end automation domain

In this chapter, we discuss the high availability and end-to-end automation aspects of the sample application environment across its multiple first-level automation domains.

We do this by creating and configuring a IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component managing an end-to-end automation domain. The IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component is configured to manage all first-level domains of our case study scenario environment.

We describe the following topics:

- ▶ “End-to-end Automation Management Component installation” on page 203
- ▶ “Installation verification tasks” on page 211
- ▶ “Users and group management” on page 218
- ▶ “End-to-end Automation Management Component configuration” on page 228
- ▶ “Defining the end-to-end automation policy” on page 229

Figure 7-1 shows the portion of the entire case study scenario that this chapter covers.

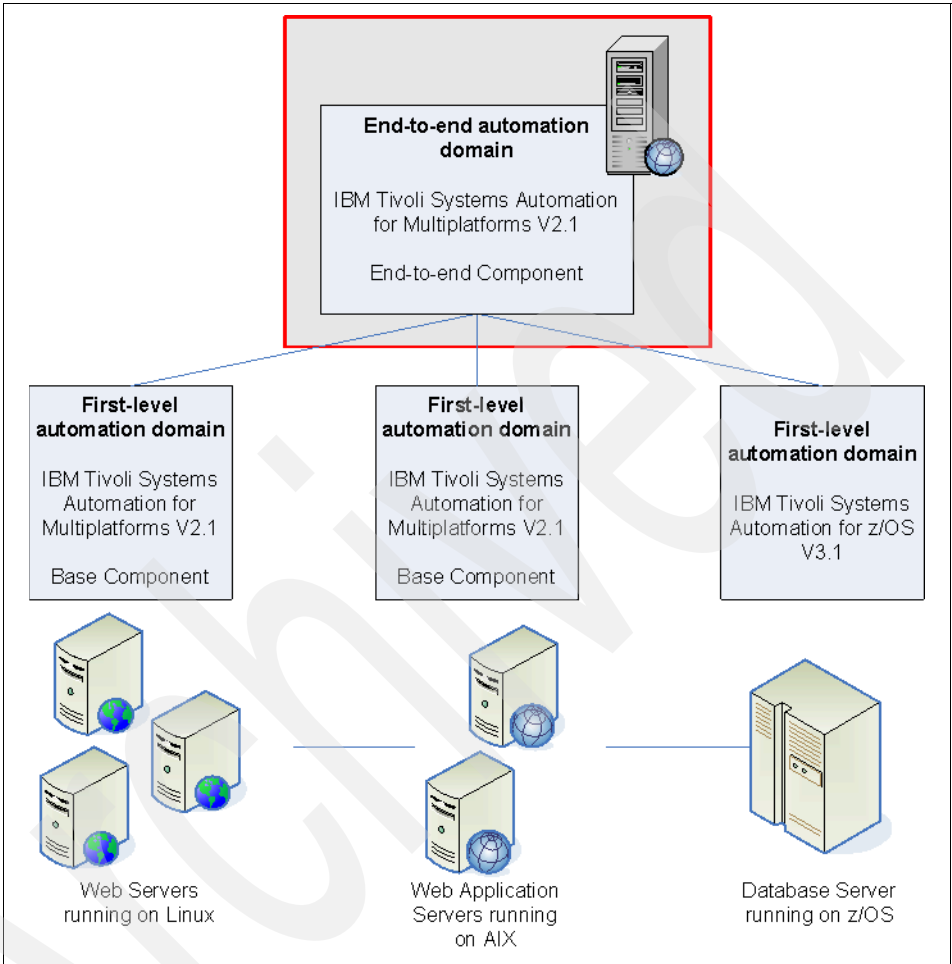


Figure 7-1 Our scenario's end-to-end automation domain

7.1 End-to-end Automation Management Component installation

The End-to-end Automation Management Component IBM Tivoli System Automation for Multiplatforms V2.1 partly runs on a J2EE Application Server and needs a database server for persisting data. The only supported J2EE Application Server is IBM WebSphere Application Server Version 6.0.0.2 with a fixed list of required interim fixes. This very particular IBM WebSphere® version is necessary because the Integrated Solutions Console (ISC) Version 6 is installed on top of IBM WebSphere.

Important: Using a different IBM WebSphere version, release, modification (refresh packs), fix level (fix packs), or even deviating interim fixes may lead to a non-working installation of the End-to-end Automation Management Component. Ensure you use the IBM WebSphere installation media shipped with IBM Tivoli System Automation for Multiplatforms V2.1.

You can summarize the installation of End-to-end Automation Management Component by the following steps:

1. Install IBM DB2 UDB Enterprise Server Edition V8.2
2. Install IBM DB2 V8 Fix Pack 10

Applying this Fix Pack will bring the IBM DB2 installation to an 8.2.3 level. You can obtain this Fix Pack from the following site:

<http://www.ibm.com/software/data/db2/udb/support/downloadv8.html>

3. Install IBM WebSphere Application Server V6
4. Install IBM WebSphere Application Server V6 Fix Pack 2

Applying this Fix Pack will bring the IBM WebSphere installation to a 6.0.0.2 level.

5. Install the following IBM WebSphere Application Server V6 interim fixes

- PK08802

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=PK08802&uid=swg24010206&loc=en_US&cs=utf-8&lang=en

- PK10066

No information on the IBM WebSphere support web site at the time of writing this IBM Redbook.

- PK01524

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=PK01524&uid=swg24009431&loc=en_US&cs=utf-8&lang=en

– PK00842

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=PK00842&uid=swg1PK00842&loc=en_US&cs=utf-8&lang=en

– PK05321

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=PK05321&uid=swg24009601&loc=en_US&cs=utf-8&lang=en

– PK06246

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=PK06246&uid=swg24009735&loc=en_US&cs=utf-8&lang=en

– PK04784

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=PK04784&uid=swg24009575&loc=en_US&cs=utf-8&lang=en

– PK06140

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=PK06140&uid=swg24009756&loc=en_US&cs=utf-8&lang=en

– PK00652

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=PK00652&uid=swg1PK00652&loc=en_US&cs=utf-8&lang=en

6. Install End-to-end Automation Management Component

All listed Interim Fixes apply to all operating systems and platforms supported by the IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component. Install all required Interim Fixes exactly in the sequence listed above.

Important: Install the interim fixes in the order presented here. Also, do not leave out any of the Interim Fixes listed above and do not install additional Interim Fixes.

For our scenario, we decided to use a single server for the End-to-end Automation Management Component. Figure 7-2 shows the components installed on our single End-to-end Automation Management Component server.

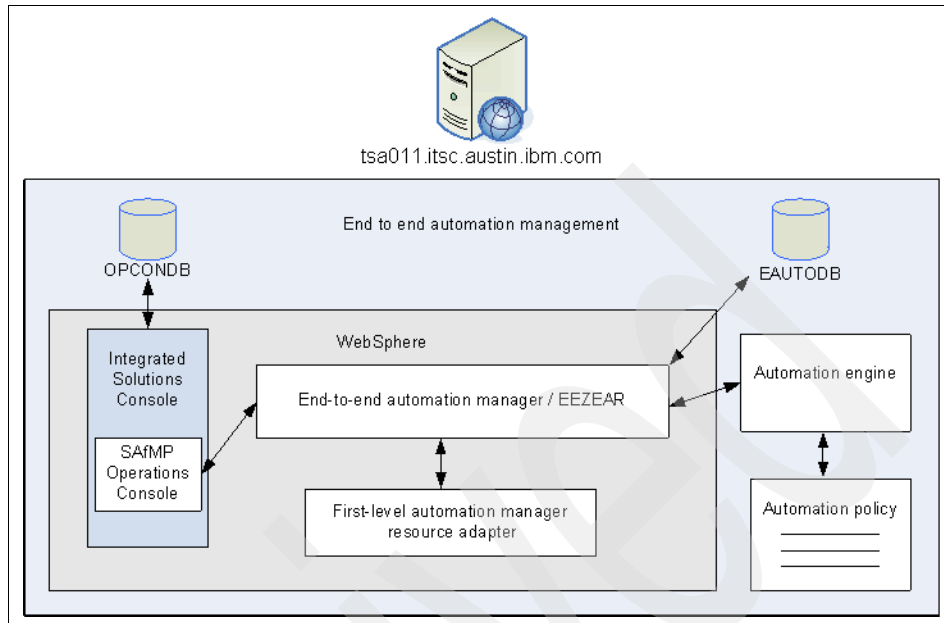


Figure 7-2 End-to-end Automation Management Component scenario server

We performed the installation of the above prerequisite software following the steps and guidelines provided in the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component User's Guide and Reference*, SC33-8211.

Important: Before proceeding with the End-to-end Automation Management Component installation, ensure that all the prerequisite software is at the required levels. Special attention is required to the IBM WebSphere Application Server Fix Packs and interim fixes.

Run the **genHistoryReport** command to generate a IBM WebSphere Application Server installation report. This tool generates an HTML file containing all changes to the IBM WebSphere Application Server product due to installation of Fix Packs and interim Fixes. Verify that the report contains exactly all the required Fix Packs and interim Fixes.

We installed the End-to-end Automation Management Component on a single server running Microsoft® Windows® 2003 Server with Service Pack 1. In this situation, this single server works as both the End-to-end Automation Management Component and the operations console server. The installation option we used in our scenario is **End-to-End Automation Management**

component, which includes the IBM Tivoli System Automation for Multiplatforms operations console.

During the installation process, we kept the default value for the end-to-end automation manager database: **EAUTODB**. We also provided the user ID and password of the IBM DB2 instance owner, the hostname, and port number of the IBM DB2 server.

We used the default configuration settings for IBM WebSphere Application Server such as the default profile and server1 application server.

For the Lightweight Third Party Authentication (LTPA) configuration settings, we specified the SSO domain name as shown in Figure 7-3.

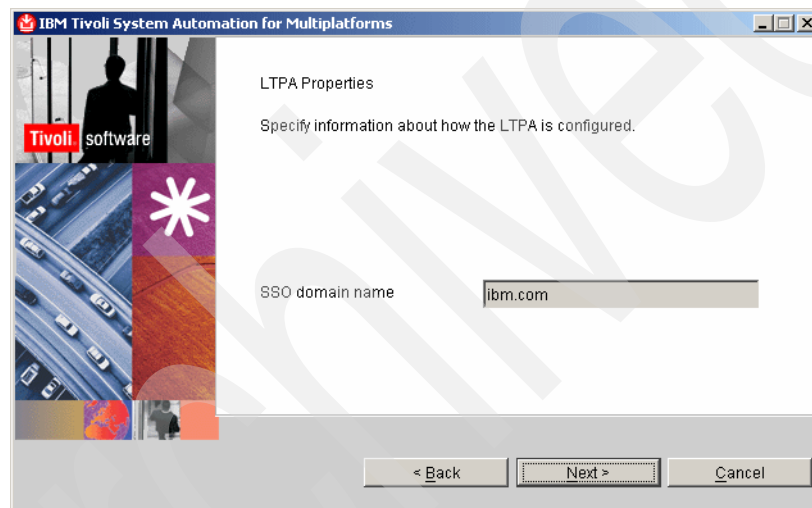


Figure 7-3 LTPA Properties: SSO domain name

We also kept the default value for the IBM Tivoli System Automation for Multiplatforms operations console database: **OPCONDB**, as shown in Figure 7-4 on page 207. Also, notice that if your database server is not local to the End-to-end Automation Management Component server, you must have already created both the operations console database on the database server and a connection to this database on the End-to-end Automation Management Component server. In our case, we use a database server local to the End-to-end Automation Management Component server and we create the operations console database by the install process.

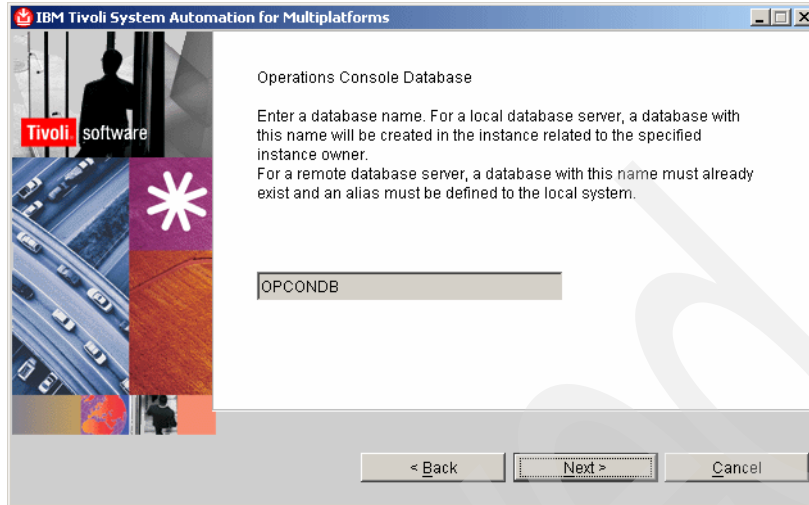


Figure 7-4 Operations Console Database

The installation proceeds to requesting the type of user registry to use for IBM WebSphere Application Server security. See Figure 7-5. In our environment, we opt for an IBM DB2 base user registry. Refer to the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component User's Guide and Reference*, SC33-8211, if you plan to use a LDAP user registry.

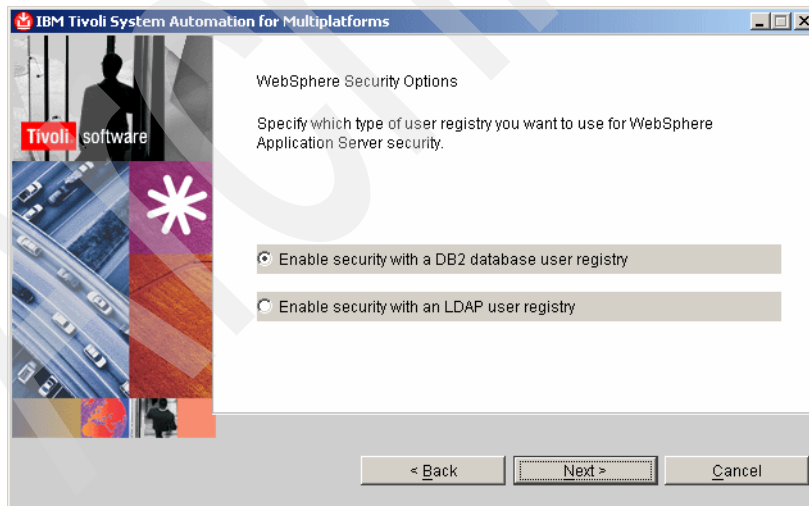
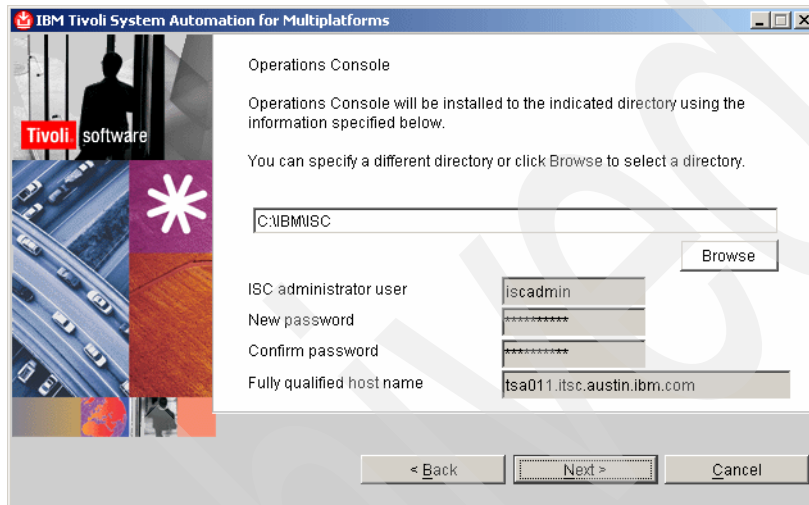


Figure 7-5 Security using IBM DB2 database user registry option

At this point, we have provided the installation process all the necessary information for the End-to-end Automation Management Component. It now starts collecting information for the IBM Tivoli System Automation for Multiplatforms operations console.

We specify the installation directory for Integrated Solutions Console (ISC), and set the credentials for the ISC administrator user, as presented in Figure 7-6.



The screenshot shows the 'Operations Console' installation window for IBM Tivoli System Automation for Multiplatforms. The window has a title bar with the IBM logo and the text 'IBM Tivoli System Automation for Multiplatforms'. On the left side, there is a vertical strip with a 'Tivoli software' logo and a decorative graphic. The main area contains the following text and fields:

- Operations Console**
- Operations Console will be installed to the indicated directory using the information specified below.
- You can specify a different directory or click Browse to select a directory.
- Directory field: C:\IBMISC (with a 'Browse' button to its right)
- ISC administrator user: iscadmin
- New password: (masked with asterisks)
- Confirm password: (masked with asterisks)
- Fully qualified host name: tsa011.itsc.austin.ibm.com

At the bottom of the window, there are three buttons: '< Back', 'Next >', and 'Cancel'.

Figure 7-6 ISC settings

Figure 7-7 shows the values for the ports used by ISC. The port numbers are registered within IBM and usually do not require modification. We use the specified default values.

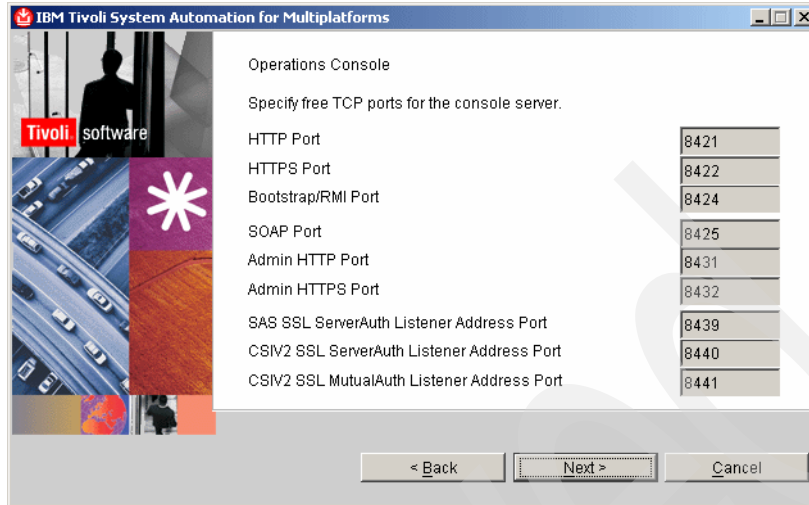


Figure 7-7 ISC port number settings

We also keep the default values for the Console Help Server port number. See Figure 7-8.

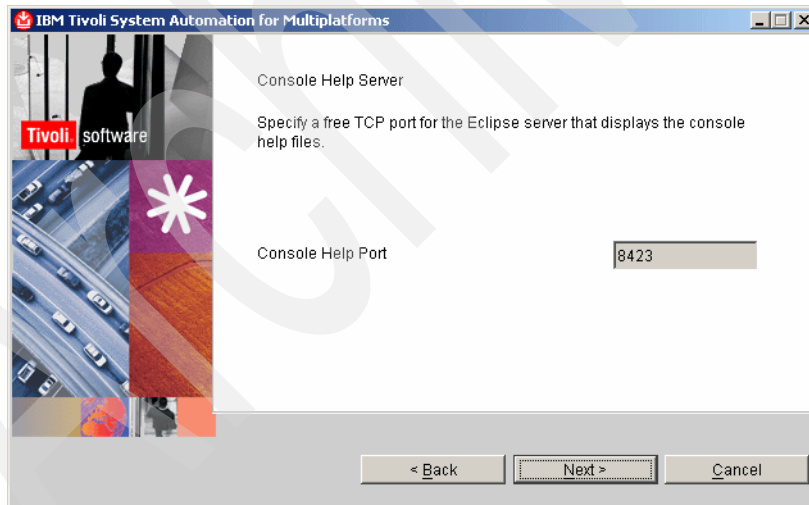


Figure 7-8 Console Help Server port number

Figure 7-9 shows the values used for the Console and Console Help Services. The Console Service is an application that runs on top of IBM WebSphere Application Server. Both Console and Console Help Services are registered as system services in our environment.

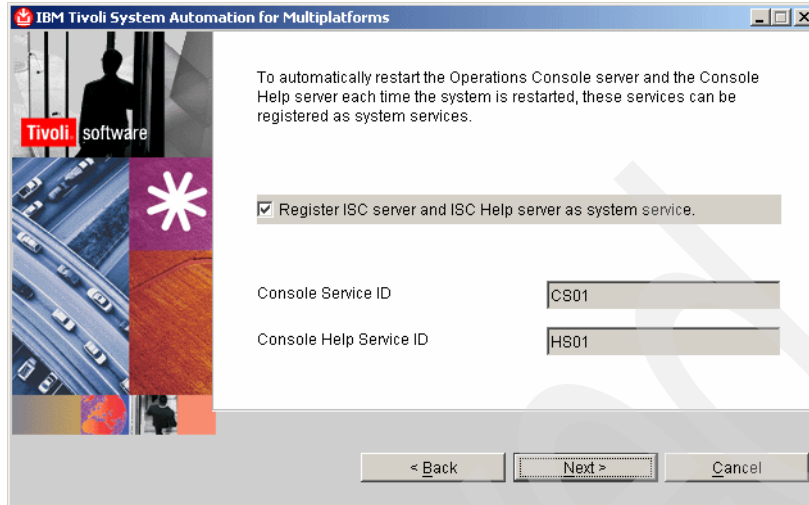


Figure 7-9 ISC application server names

As a last step before the actual installation process begins, we provide the name of our End-to-end Automation Management Component automation domain (Figure 7-10). We use this domain later in our environment to manage resources of our first-level automation domains.

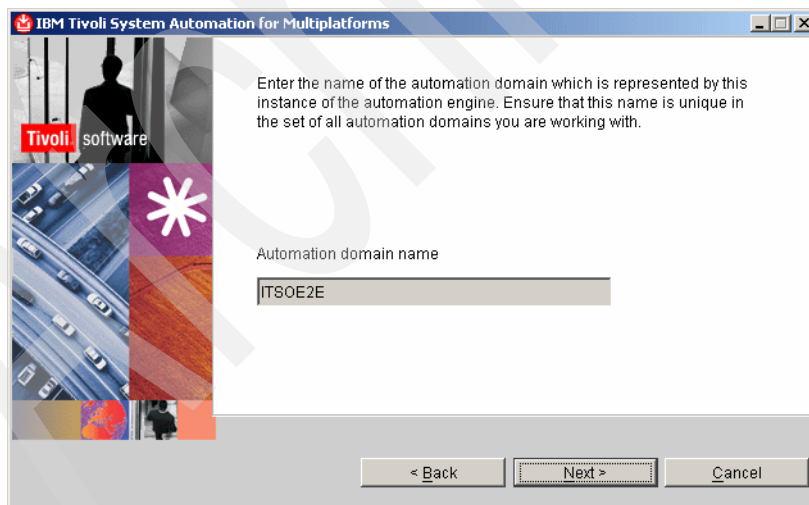


Figure 7-10 End-to-end automation domain name

The installation process finishes by showing an installation completion message as shown in Figure 7-11.

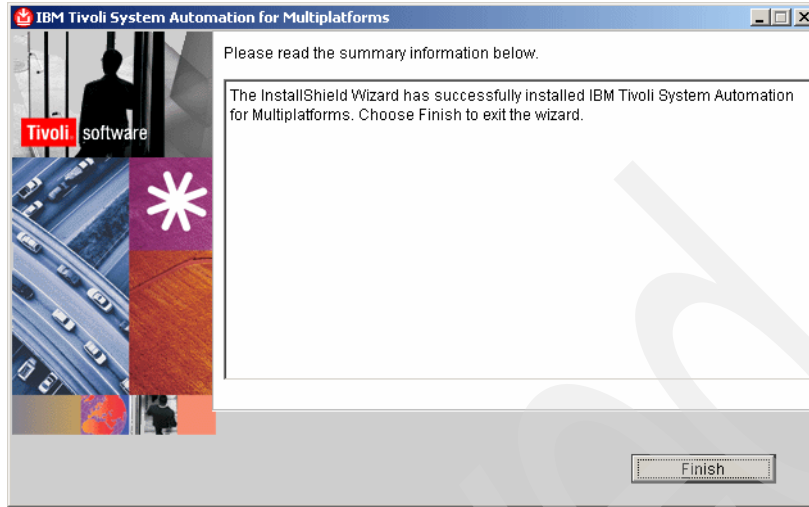


Figure 7-11 Installation successful message

7.2 Installation verification tasks

In this section, we provide a list of tasks to verify the installation of the End-to-end Automation Management Component. Perform the following tasks to verify a successful installation of the End-to-end Automation Management Component:

- ▶ Verify the creation of the EAUTODB and OPCONDB databases
- ▶ Start the End-to-end Automation Management Component automation engine
- ▶ Verify the status of the applications on IBM WebSphere Application Server
- ▶ Check the definition and connection of the JDBC providers in IBM WebSphere Application Server
- ▶ Start the ISC portal application
- ▶ Open the IBM Tivoli System Automation for Multiplatforms operations console

7.2.1 EAUTODB and OPCONDB databases

In order to verify the creation of the EAUTODB and OPCONDB databases, list the database directory on the IBM DB2 server and try a connection to the databases using the database instance owner specified during the End-to-end Automation Management Component installation as shown Example 7-1.

Example 7-1 Database verification

C:\>db2 list database directory

System Database Directory

Number of entries in the directory = 2

Database 1 entry:

Database alias	= EAUTODB
Database name	= EAUTODB
Database drive	= C:\DB2
Database release level	= a.00
Comment	=
Directory entry type	= Indirect
Catalog database partition number	= 0
Alternate server hostname	=
Alternate server port number	=

Database 2 entry:

Database alias	= OPCONDB
Database name	= OPCONDB
Database drive	= C:\DB2
Database release level	= a.00
Comment	=
Directory entry type	= Indirect
Catalog database partition number	= 0
Alternate server hostname	=
Alternate server port number	=

C:\>db2 connect to EAUTODB user db2admin

Enter current password for db2admin:

Database Connection Information

Database server	= DB2/NT 8.2.3
SQL authorization ID	= DB2ADMIN
Local database alias	= EAUTODB

C:\>db2 connect to OPCONDB user db2admin

Enter current password for db2admin:

Database Connection Information

Database server	= DB2/NT 8.2.3
SQL authorization ID	= DB2ADMIN

Local database alias = OPCONDB

Ensure that the tables used by the End-to-end Automation Management Component are created in the EAUTODB database, as presented in Example 7-2.

Example 7-2 EAUTODB database tables

```
C:\>db2 connect to EAUTODB user db2admin
Enter current password for db2admin:
```

Database Connection Information

```
Database server      = DB2/NT 8.2.3
SQL authorization ID = DB2ADMIN
Local database alias = EAUTODB
```

```
C:\>db2 select name from sysibm.systables where creator='EAUTOUSR'
```

```
NAME
```

```
-----
EEZAUTOMATIONACCESS
EEZAUTOMATIONRELATION
EEZDOMAINSUBSCRIPTION
EEZOPERATORDOMAINFILTER
EEZOPERATORDOMAINPREFERENCES
EEZOPERATORHIDDENDOMAIN
EEZRESOURCESUBSCRIPTION
```

```
7 record(s) selected.
```

7.2.2 End-to-end Automation Management Component automation engine startup

Start the End-to-end Automation Management Component automation engine using the **eezdmn** command. For details about the **eezdmn** command, refer to the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component User's Guide and Reference*, SC33-8211.

Example 7-3 shows the start of the End-to-end Automation Management Component automation engine in our environment.

Example 7-3 Automation engine status

```
C:\IBM\tsamp\eez\bin>eezdmn -?
IBM Tivoli System Automation end-to-end automation engine Version:
2.1.0.053601, NO_APAR
```

Usage:
eezdmn [option]

-START	Starts the automation engine
-SHUTDOWN -SHUTD	Stops the automation engine
-MONITOR -M	Displays the current state
-RECONFIG -R	Re-configures the automation engine
-CO	Starts only the EIF2JMS conversion thread
-XD ("*" "<RES_NAME>[,<RES_NAME>"])	<DUMPFIL>
	Dumps (all specific) resources to a file

When no option is specified, START is used

C:\IBM\tsamp\eez\bin>eezdmn -M

About to get current state of the automation engine
State of automation engine is: NOT AVAILABLE - the automation engine probably not started
Done

C:\IBM\tsamp\eez\bin>eezdmn.bat -START

Starting...
About to get current state of the automation engine
State of automation engine is: IDLE - no policy activated
Done

7.2.3 End-to-end Automation Management Component applications status

The End-to-end Automation Management Component installation process installs several applications on IBM WebSphere Application Server. For the End-to-end Automation Management Component automation engine, the following applications are installed:

- ▶ EEZEAR
- ▶ EventServer
- ▶ EventServerMdb

In order to check the status of the above listed applications, perform the following steps.

1. Open the IBM WebSphere Application Server administrative console using the following URL:

https://<SERVER_NAME>:9043/ibm/console/login.jsp

2. Provide the iscadmin user credentials.

3. Navigate to **Applications** → **Enterprise Applications**.
4. Verify the status of the applications. Start them in case they are stopped, as shown in Figure 7-12.

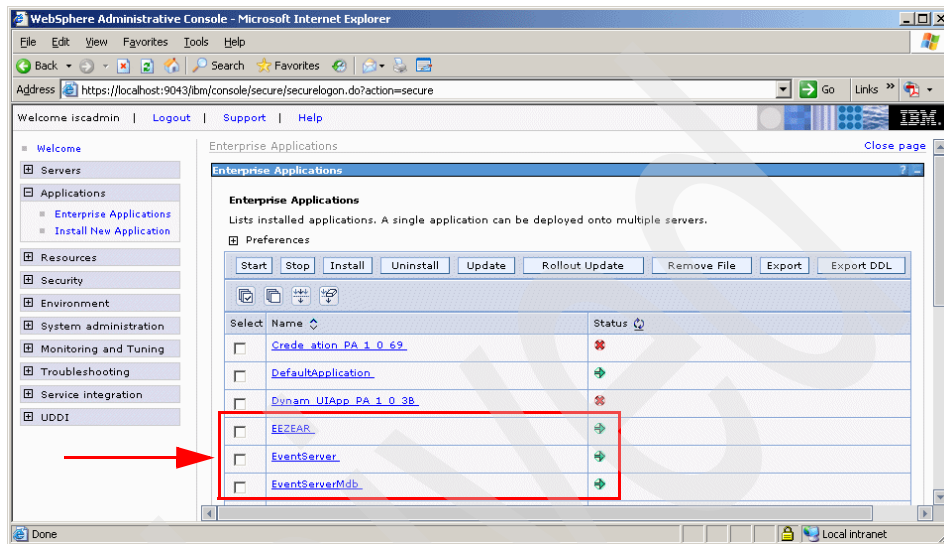


Figure 7-12 End-to-end Automation Management Component applications

7.2.4 JDBC providers connection

You can check the JDBC providers connection in IBM WebSphere Application Server by performing the following steps:

On the IBM WebSphere Application Server administrative console, navigate to **Resources** → **JDBC Providers**.

Select **DB2 Universal JDBC Driver (XA)** → **Data Sources**.

Select **EAUTODBDS** and click **Test Connection** as seen in Figure 7-13.

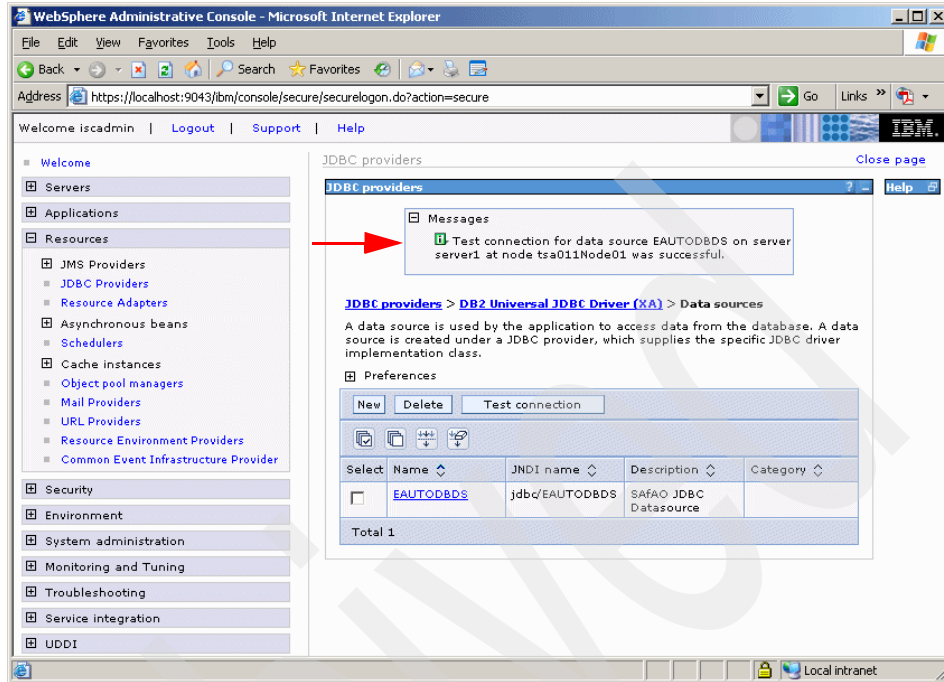


Figure 7-13 JDBC Providers connection

7.2.5 ISC portal application startup

To be able to use the IBM Tivoli System Automation for Multiplatforms operations console, both the Integrated Solutions Console (ISC) console server and the Eclipse Help system application must be up and running.

In our case, the Eclipse Help system runs as a Windows Service named HS01. We use the Windows Services panel to start it.

The ISC console server runs as a portal application on IBM WebSphere Application Server. Start it by using the **startISC ISC_PORTAL** command. In our environment, we run the **startISC** command as shown in Example 7-4.

Example 7-4 ISC startup

```
C:\IBM\ISC\PortalServer\bin>startISC.bat ISC_PORTAL
```

```
C:\IBM\ISC\PortalServer\bin>rem Licensed Materials - Property of IBM, (C)
Copyright IBM Corp. 2003,2004 - All Rights reserved.
```

```
ADMU7701I: Because ISC_PORTAL is registered to run as a Windows Service, the
request to start this server will be completed by starting the
```


associated Windows Service.
ADMU0116I: Tool information is being logged in file
C:\IBM\WebSphere\AppServer\profiles\default\logs\ISC_Portal\startServer.log
ADMU0128I: Starting tool with the default profile
ADMU3100I: Reading configuration for server: ISC_Portal
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server ISC_Portal open for e-business; process id is 33680

You can stop the ISC console server by running the **stopISC** command as follows: `stopISC ISC_PORTAL iscadmin <iscadmin_password>`. Refer to the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component User's Guide and Reference*, SC33-8211, for details about the **startISC** and **stopISC** commands.

7.2.6 System Automation operations console

You can verify the IBM Tivoli System Automation for Multiplatforms operations console by performing the following tasks.

1. Open the ISC administrative console using the following URL:
http://<SERVER_NAME>:8421/ibm/console
2. Provide the **iscadmin** user ID credentials.
3. Navigate to **IBM Tivoli System Automation for Multiplatforms** → **SA Operations Console**. The IBM Tivoli System Automation for Multiplatforms operations console panel displays as shown in Figure 7-14.

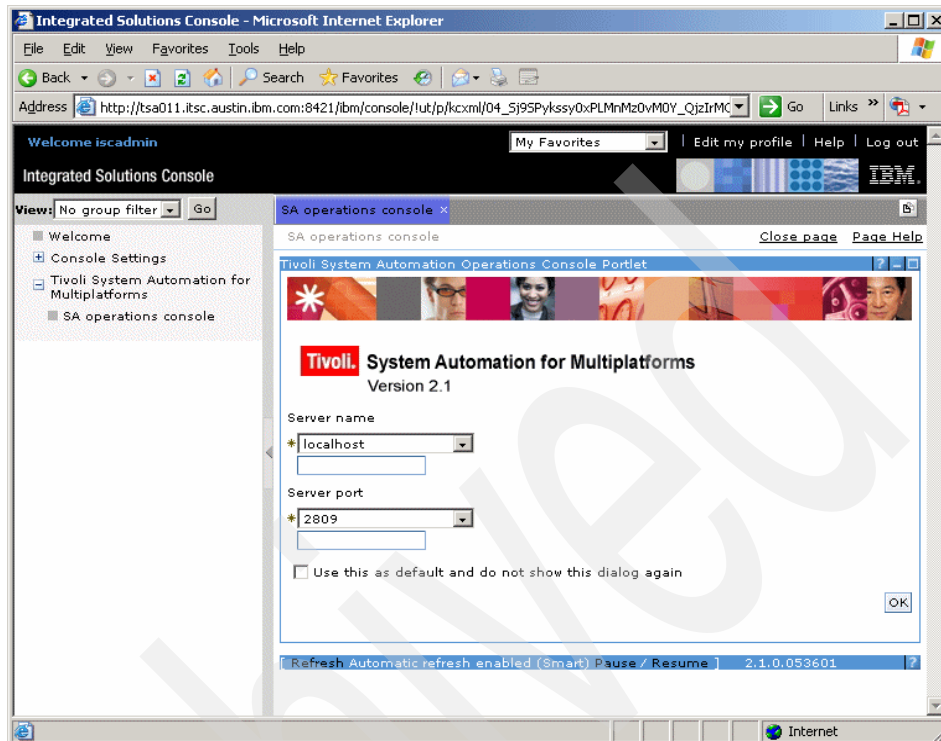


Figure 7-14 End-to-end Automation Management Component operations console

Refer to the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component User's Guide and Reference*, SC33-8211, for troubleshooting common installation problems.

7.3 Users and group management

This section provides the tasks for users, user groups, and their associated roles for End-to-end Automation Management Component. We provide an introduction to the roles used by End-to-end Automation Management Component and how to associate user groups with these roles.

The sections below contain the tasks you need to perform to authorize users to work with End-to-end Automation Management Component. You must perform these tasks immediately after the End-to-end Automation Management Component installation.

7.3.1 Creating users

You must create users and grant them proper authorization to the End-to-end Automation Management Component operations console. In our case study scenario, we use the default ISC administrator account `iscadmin` and have no need to define new users. We do not recommend this decision for production environments.

You can define users for the End-to-end Automation Management Component operations console by accounts with administrative authority, such as the `iscadmin` user ID. You perform this task using the ISC administration console, under **Console Settings** → **User and Group Management** → **All Authenticated Portal Users** → **New User**.

Refer to the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component User's Guide and Reference*, SC33-8211, for details.

7.3.2 Creating user groups

There are End-to-end Automation Management Component access roles defined in IBM WebSphere Application Server during the End-to-end Automation Management Component installation process. These roles determine the actions that users of the End-to-end Automation Management Component operations console are allowed to perform. The defined roles are:

EEZMonitor	This role provides authorization for query-type operations against defined resources.
EEZOperator	This role allows users to issue requests against first-level automation domain resources, but does not allow users to activate or deactivate policies.
EEZConfigurator	Extends the EEZOperator role by allowing users to work with policies.
EEZAdministrator	This role allows users to perform all operations for resources defined in the first-level domain.
EEZEndToEndAccess	This role must be given to users responsible for managing the end-to-end automation resources. This role does not determine which actions users can perform in the end-to-end automation domain. Users must be granted other roles defined above for the type of actions they are allowed to perform.
EEZAsync	This role is used for all internal IBM WebSphere Application Server methods of End-to-end Automation Management Component. This role is granted to the

IBM WebSphere Application Server user ID during the End-to-end Automation Management Component installation.

Table 7-1 shows the required user groups and their associated roles for the End-to-end Automation Management Component.

Table 7-1 End-to-end Automation Management Component required groups

End-to-end Automation Management Component role	Group name
EEZMonitor	EEZMonitorGroup
EEZOperator	EEZOperatorGroup
EEZConfigurator	EEZConfiguratorGroup
EEZAdministrator	EEZAdministratorGroup
EEZEndToEndAccess	EEZEndToEndAccessGroup
EEZAsync	No group definition required

In order to define these user groups, perform the following tasks:

1. Log in to the ISC administration console, providing the iscadmin user ID credentials.
2. Navigate to **Console Settings** → **User and Group Management** and select **New Group**.
3. Enter the name of the group and click **OK**. For example, EEZAdministratorGroup.
4. Repeat the above tasks for all required groups presented in Table 7-1 on page 220.

Figure 7-15 shows all the required groups defined in our environment.

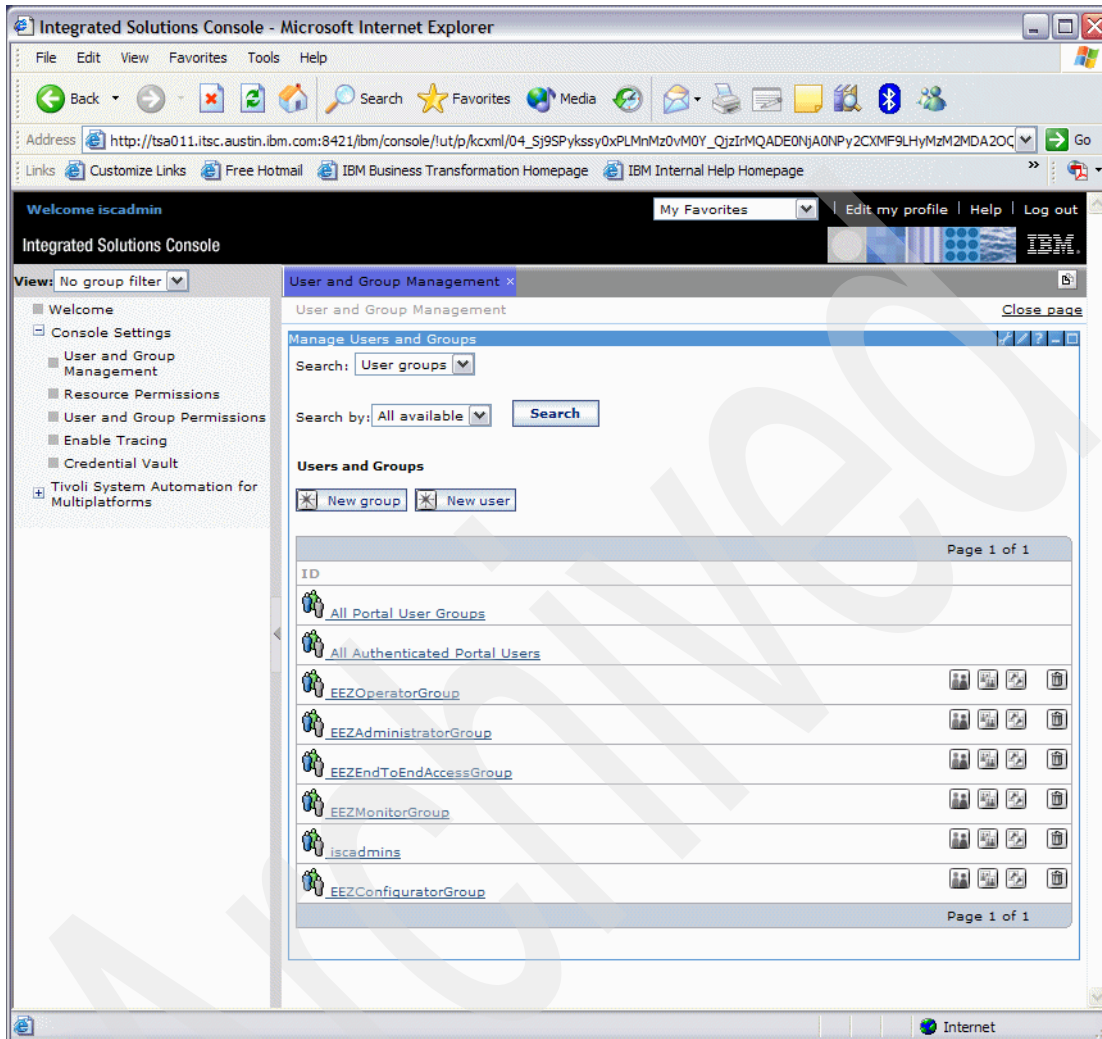


Figure 7-15 Required groups definition

7.3.3 Assigning access permissions to user groups

In order to enable End-to-end Automation Management Component users to work with ISC administrative console and the End-to-end Automation Management Component operations console, you must grant the following to the user groups defined in the previous section:

- Access permissions to the pages of ISC administrative console

- Access permissions to the End-to-end Automation Management Component operations console

These tasks must be performed once after the installation of the End-to-end Automation Management Component. When you have performed both tasks described in this section, users who belong to the user groups can access the pages of ISC administrative console and the End-to-end Automation Management Component operations console. However, users can only perform tasks when the access roles have been assigned to the user groups, as described in “Assigning access roles to user groups” on page 226.

Important: Once you have performed the above tasks, you must restart the Integrated Solutions Console. You do this by issuing the following command:

```
stopISC ISC_PORTAL -username iscadmin -password <iscadminPW>
```

where iscadminPW is the password for the iscadmin user ID.

Access to the pages of ISC administrative console

In order to grant user groups access permissions to the pages of ISC administrative console, perform the following tasks.

1. Log in to the ISC administration console providing the iscadmin user ID credentials.
2. Navigate to **Console Settings** → **Resource Permissions**.
3. In the Resource Types list, select **Pages**.
4. Click the icon representing **Assign Access** for **Content Root**.
5. Click the icon representing **Edit Role** for **Users**. Ensure that **Allow Propagation** and **Allow Inheritance** boxes are checked.
6. Select **Add**.
7. Select **User groups** in the **Search for Users or User Groups** pull-down menu and click **Search**.
8. Select all of the End-to-end Automation Management Component user groups defined in the previous section and click **OK**.
9. Ensure that the EJPA04003I: Members successfully added to the role message displays, as shown in Figure 7-16. For changes to take effect, you must restart Integrated Solutions Console.

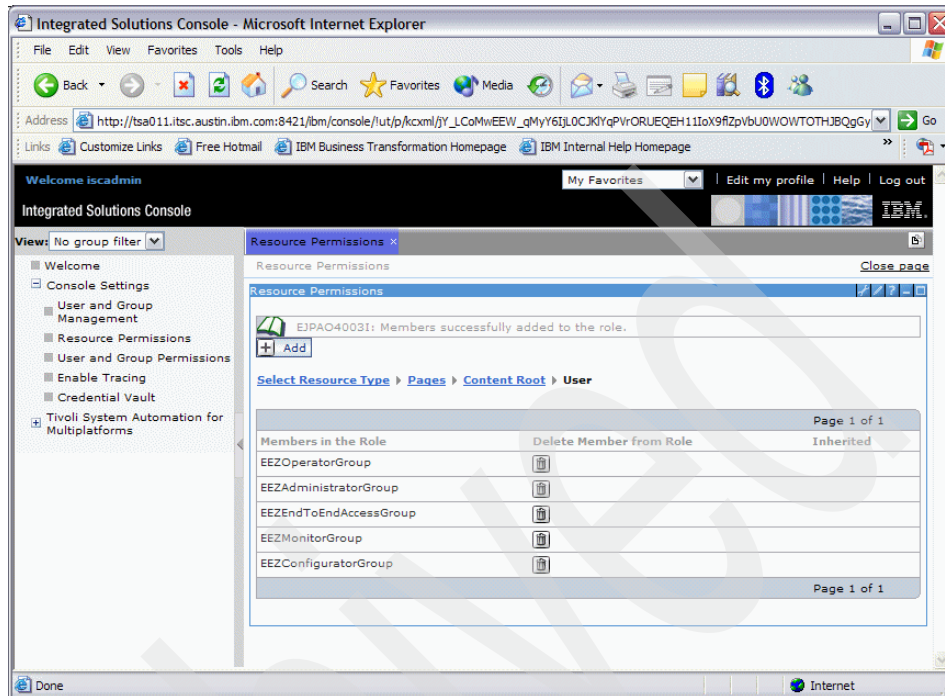


Figure 7-16 Granting access to the Integrated Solution Console pages

Access to the operations console

In order to grant user groups access permissions to the End-to-end Automation Management Component operations console, perform the following tasks.

1. Log in to the ISC administration console, providing the iscadmin user ID credentials.
2. Navigate to **Console Settings** → **Resource Permissions**.
3. In the Resource Types list, select **Portlet Applications**.
4. Click the icon representing **Assign Access** for **Tivoli System Automation for Multiplatforms Operations Console**.
5. Click the icon representing **Edit Role** for **Users**. Ensure that **Allow Propagation** and **Allow Inheritance** boxes are checked.
6. Select **Add**.
7. Select **User groups** in the **Search for Users or User Groups** pull-down menu and click **Search**.
8. Select all of the End-to-end Automation Management Component user groups defined in the previous section and click **OK**.

9. Ensure that the EJPA04003I: Members successfully added to the role message displays, as shown in Figure 7-17. For changes to take effect, you must restart the Integrated Solutions Console operations console.

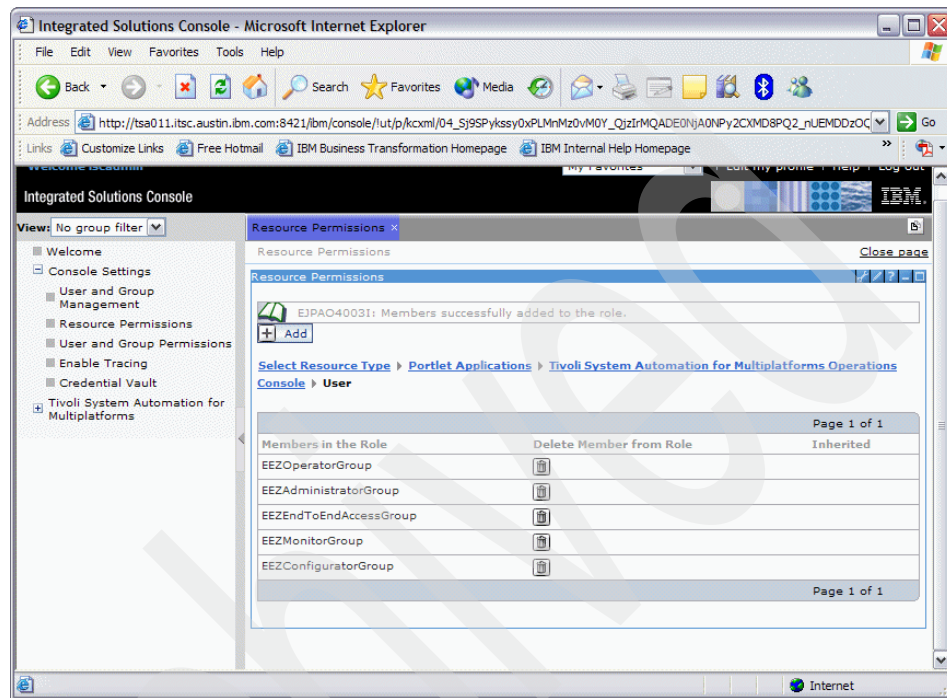


Figure 7-17 Granting access to ISC operations console

7.3.4 Assigning users to user groups

All users we defined in “Creating users” on page 219 must be members of a user group. This allows users to perform operations according to the roles associated to the user groups of which they are members.

The iscadmin user ID must be a member of the following groups:

- ▶ EEZAdministratorGroup
- ▶ EEZEndToEndAccessGroup

In our case study scenario environment, we use the default ISC administrator account iscadmin for all tasks for both ISC administrative console and End-to-end Automation Management Component operations console.

In order to add a user ID as a member of a user group, perform the following tasks.

1. Log in to the ISC administrative console providing the iscadmin user ID credentials.
2. Navigate to **Console Settings** → **User and Group Management**.
3. In the Manage Users and Groups, select **All Portal User Groups**.
4. Select the End-to-end Automation Management Component user groups we defined in the previous section, for example, EEZAdministratorGroup.
5. Click **Add Member**.
6. Select the user ID from the user list. In our case, we select iscadmin. Click **OK**.
7. Repeat the above steps to add the user ID iscadmin to the EEZEndToEndAccessGroup. Ensure you receive the EJPAL0140I: User or user group has been successfully added to the selected group message, as shown in Figure 7-18.

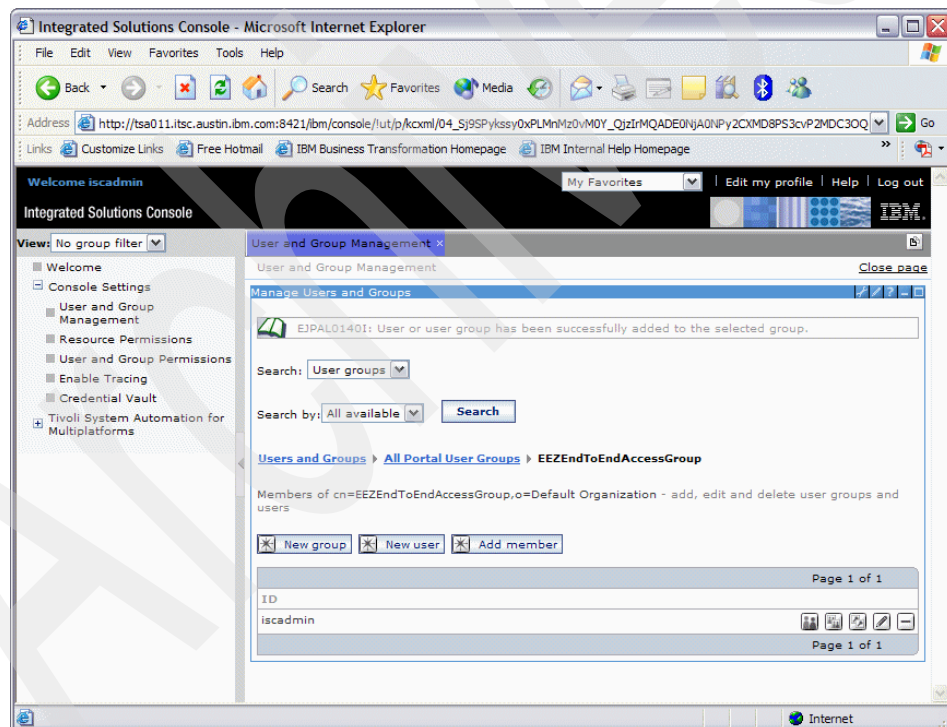


Figure 7-18 User association to a user group

You must perform the above steps for each defined user ID.

7.3.5 Assigning access roles to user groups

After you create the End-to-end Automation Management Component user groups using the ISC administrative console, you must assign access roles to these user groups in the IBM WebSphere Application Server administrative console. This will grant the users who are members of a user group permission to perform tasks associated to the role.

In order to assign access roles to user groups, perform the following tasks.

1. Log in to the IBM WebSphere Application Server administration console, providing the iscadmin user ID credentials.
2. Expand **Applications** → **Enterprise Applications** and select **EEZEAR** application.
3. Select **Map security roles to users/groups**.
4. Select one of the End-to-end Automation Management Component predefined roles, for example, EEZAdministrator.
5. Deselect both check boxes **Everyone?** and **All authenticated?** and Click **Look up groups**.
6. Click **Search**.
7. Select the group to be associated with this role. In our case, EEZAdministratorGroup. The proper entry is cn=EEZAdministratorGroup, o=Default Organization.
8. Move the selected group to the **Selected** column and click **OK**.
9. Repeat the above steps for all End-to-end Automation Management Component predefined roles and associated user groups.

Figure 7-19 shows the roles and user group associations for our environment.

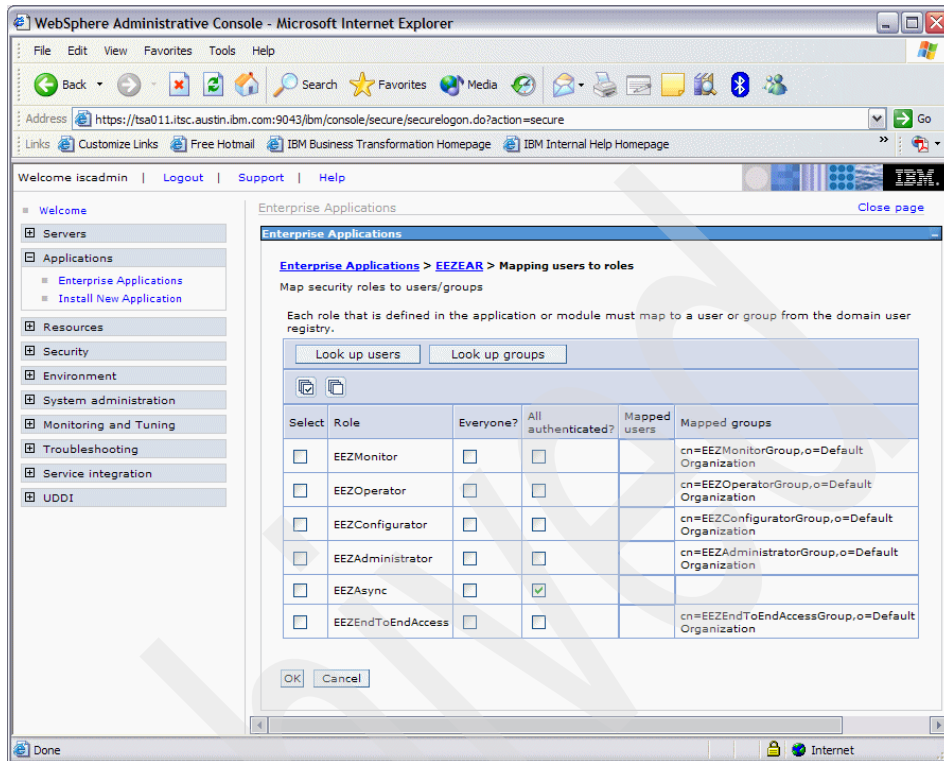


Figure 7-19 Mapping roles to user groups

10. Once you complete all associations, save the changes in the IBM WebSphere Application Server.
11. Expand **Applications** → **Enterprise Applications** and select **EEZEAR** application.
12. Select **Map RunAs roles to users**.
13. Enter the credentials of the iscadmin user ID and select the role **EEZAsync**.
14. Click **Apply**.
15. Click **OK**.
16. Once you complete all definitions, save the changes in the IBM WebSphere Application Server.
17. You must restart IBM WebSphere Application Server for the changes to take effect.

7.4 End-to-end Automation Management Component configuration

The basic configuration of the End-to-end Automation Management Component is performed during its installation time. The End-to-end Automation Management Component installation process creates several configuration files in the <E2E_INSTALL>/cfg directory, where <E2E_INSTALL> is the installation directory of the End-to-end Automation Management Component. You *must not manipulate* these configuration files manually.

End-to-end Automation Management Component provides a configuration tool to perform changes in the configuration files. The name of the End-to-end Automation Management Component configuration tool is **cfgeezdmn** and it is located in the <E2E_INSTALL>/bin directory.

For an overview of the End-to-end Automation Management Component configuration tool functionality, refer to the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-End Automation Management User's Guide and Reference*, SC33-8211.

For our case study scenario, we have to inform the End-to-end Automation Management Component automation domain how to authenticate to the first-level automation domains. We do this by providing the user credentials specific to our first-level automation domains. Whenever End-to-end Automation Management Component needs to perform actions against resources defined in the first-level automation domains, it collects the proper credentials in the `eez.automation.engine.dif.properties` file. All passwords are stored using an encryption mechanism.

In order to set the first-level automation domain user credentials, perform the following tasks.

1. Start the End-to-end Automation Management Component configuration tool by issuing the **cfgeezdmn** command (`cfgeezdmn.bat` in our environment).
2. Select the **User credentials** tab.
3. Under the Credentials for accessing specific first-level automation domains area, click **Add**.
4. Provide the Domain's name, User ID, and password.

Repeat this step for each first-level automation domain. Figure 7-20 shows our settings.

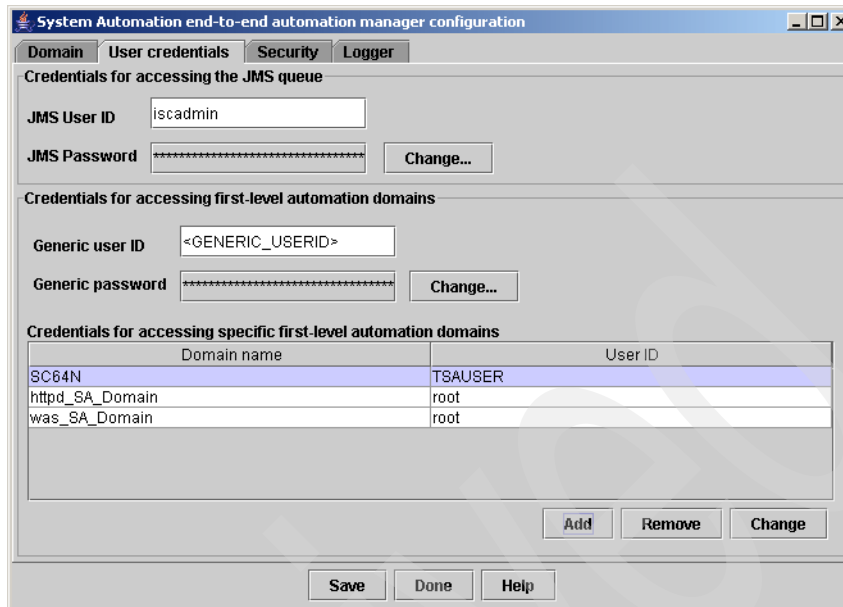


Figure 7-20 First-level automation domain credentials

- Click **Save** to commit the changes.
- You must restart the End-to-end Automation Management Component automation engine or re-configure it for the changes to take effect. In order to re-configure the End-to-end Automation Management Component automation engine, issue the **eezadm** command as follows:

```
C:\IBM\tsamp\eez\bin>eezdmn.bat -RECONFIG
About to re-configure automation engine
The automation engine has been re-configured
```

7.5 Defining the end-to-end automation policy

End-to-end Automation provides the capability to automate the operation of resources within heterogeneous environments. As presented in Chapter 3, “Case study scenario overview” on page 47, our scenario environment depicts a multi-tiered business application that has components running in heterogeneous platforms.

Each tier of this application runs on a dedicated infrastructure that has been made high available using IBM Tivoli System Automation for Multiplatforms V2.1 first-level automation domains. Now, the End-to-end Automation Management

Component is able to ensure high availability of the entire infrastructure used by the business application.

We accomplish this by defining an end-to-end automation policy that contains resource definitions and logical relationships among them.

7.5.1 Automation requirements and policy overview

In this section, we create an end-to-end automation policy which will fulfill the following automation requirements:

- ▶ Each tier of our application should have a desired state of online.
- ▶ In case of a failure of the database environment, the application environment must be brought offline.
- ▶ In case of a failure of either the application or database environment, the HTTP server environment must also be brought offline.
- ▶ The start and stop process of the various elements making up the application environment must obey a predefined order to ensure that required resources are available when needed. Specifically:
 - The database environment must be up and running for the Application environment start process.
 - Both the application and database environments must be up and running for the HTTP server environment start process.
 - Both the HTTP server environment and the application environment must be stopped so that the database environment can be properly stopped.
 - The HTTP server environment must be stopped so that the application environment can be stopped.

In order to achieve the automation requirements we describe above, we define an end-to-end automation policy that contains resource definitions that refer to resources on our first-level automation domains, resource groups organizing our end-to-end resources in a hierarchical structure, and relationship definitions.

Figure 7-21 provides an overview of the elements and their relationships for the end-to-end automation policy that fulfills the above automation requirements.

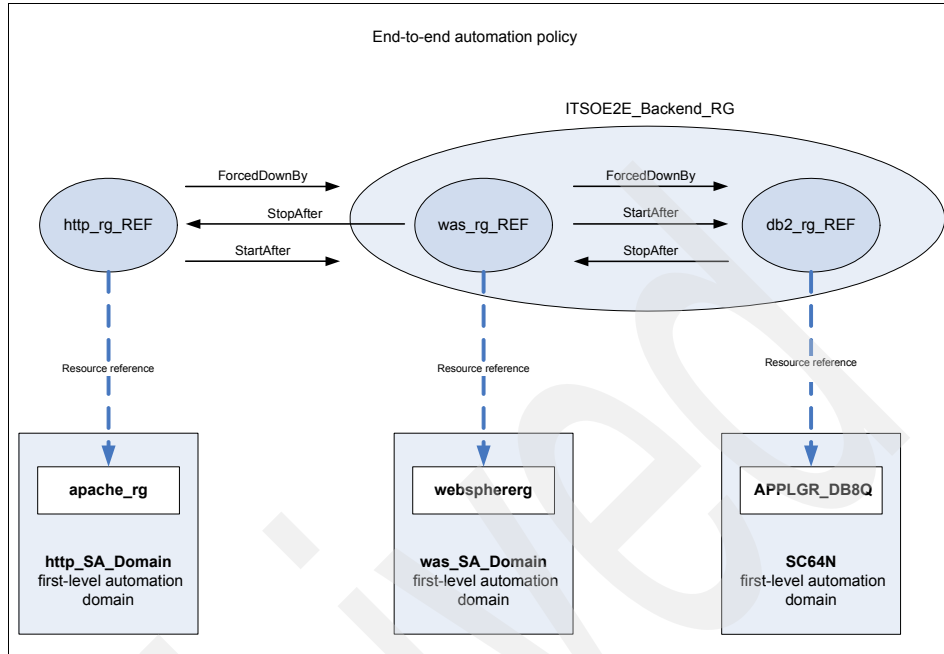


Figure 7-21 Case study scenario end-to-end automation policy

The following sections go into detail about how we define the end-to-end automation policy depicted in Figure 7-21.

7.5.2 Creating the end-to-end automation policy file

We create end-to-end automation policies by defining XML-formatted elements. IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component provides a schema definition file. Before defining the end-to-end automation policy, you should first review and gain a basic understanding of the schema definition file `EEZPolicy.xsd`. This file is located in the `<TSA_E2E_install>/policyPool` directory, where `<TSA_E2E_install>` is the installation directory of the End-to-end Automation Management Component.

Refer to the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-End Automation Management User's Guide and Reference*, SC33-8211-00, for detailed instructions about creating the end-to-end automation policy.

Note: Although you can create the end-to-end automation policy using any text editor, we highly recommend the use of an XML editor.

Once created, you must check the end-to-end automation policy XML file against the provided schema and place the end-to-end automation policy XML file in the policyPool directory prior to activation.

Based on the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-End Automation Management User's Guide and Reference*, SC33-8211-00, guidelines, we create the following XML element definitions:

- ▶ Standard end-to-end automation policy header and documentation information
- ▶ Resource references
- ▶ Resource groups
- ▶ Relationships

There is an additional element definition that our scenario did not require: *choice group*. Refer to the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-End Automation Management User's Guide and Reference*, SC33-8211-00, for details.

Automation policy header and documentation information

Our XML header information uses the sample template provided with the installation of IBM Tivoli System Automation for Multiplatforms End-to-end Automation Management Component with changes to the documentation fields. The XML policy sample template file is located in the <TSA_E2E_install>/policyPool/template.xml file, where <TSA_E2E_install> is the installation directory of the End-to-end Automation Management Component.

All entries in the end-to-end automation policy XML file must be enclosed by the <AutomationPolicy> element.

Based on the sample template, we create the following entries for the <PolicyInformation> XML element:

- ▶ PolicyName
- ▶ AutomationDomainName
- ▶ PolicyToken
- ▶ PolicyAuthor
- ▶ PolicyDescription

Example 7-5 shows the above definitions in XML format:

Example 7-5 PolicyInformation element definition

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<AutomationPolicy version="1.0"
```



```

xmlns="http://www.ibm.com/TSA/Policy.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/TSA/Policy.xsd EEZPolicy.xsd ">
  <PolicyInformation>
    <PolicyName>ITSOPolicyE2E</PolicyName>
    <AutomationDomainName>ITS0E2E</AutomationDomainName>
    <PolicyToken> 5.0 </PolicyToken>
    <PolicyAuthor>ITS0 Redbook Team</PolicyAuthor>
    <PolicyDescription>
      ITS0 E2E automation for Redbook Case Study Scenario
    </PolicyDescription>
  </PolicyInformation>

```

Resource definitions

Here we define the resources of the end-to-end automation domain by referencing resources of our first-level automation domains. The guidelines provided in the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-End Automation Management User's Guide and Reference*, SC33-8211-00, as well as the layout of our first-level automation domains made the selection resources for use in our end-to-end automation policy simple.

Having in mind the automation requirements described in “Automation requirements and policy overview” on page 230, we decide to create end-to-end automation resources that refer to the top-level or outermost first-level automation domain resource group.

Figure 7-22 on page 234 shows the end-to-end resource references to the selected resource group defined in our first-level automation domains.

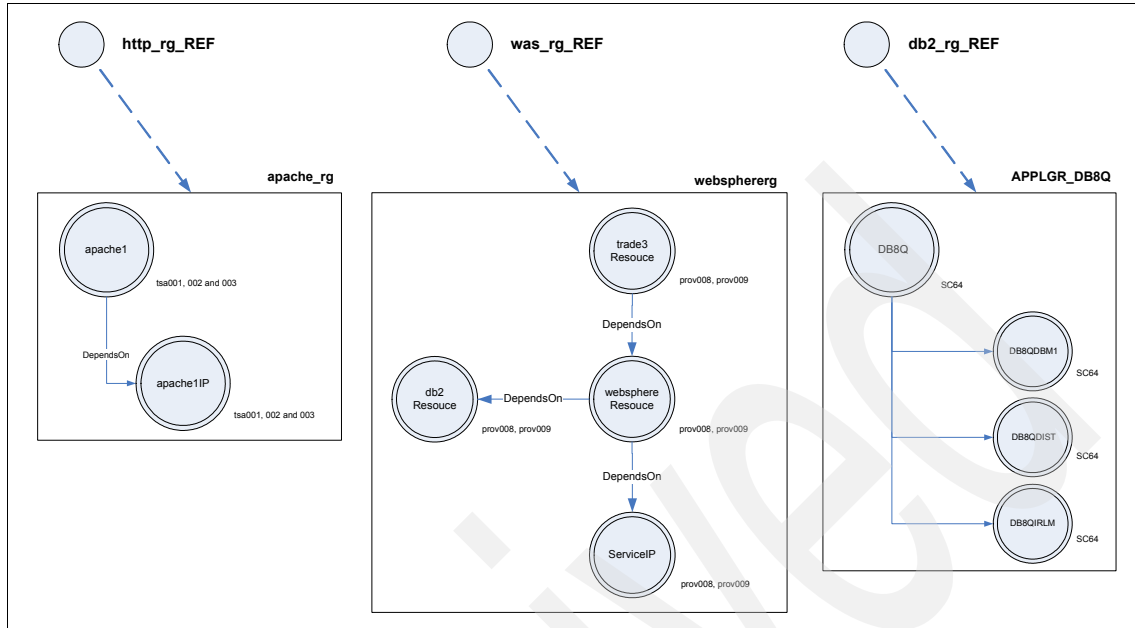


Figure 7-22 Resource reference selection

As you see in Figure 7-22, the selected resource groups are as follows:

- ▶ The resource reference `db2_rg_REF` refers to the `APPLGR_DB8Q` application group defined in the `SC64N IBM Tivoli System Automation for z/OS V3.1 z/OS first-level automation domain`.
- ▶ The resource reference `was_rg_REF` refers to the `websphererg` resource group defined in the `was_SA_Domain IBM Tivoli System Automation for Multiplatforms V2.1 first-level automation domain`.
- ▶ The resource reference `http_rg_REF` refers to the `apache_rg` resource group defined in the `http_SA_Domain IBM Tivoli System Automation for Multiplatforms V2.1 first-level automation domain`.

Resource references are defined in the end-to-end automation policy using the `<ResourceReference>` element. For each `<ResourceReference>` element definition, we use the following entries and sub-entries:

- ▶ `Description`
- ▶ `Owner`
- ▶ `DesiredState`, except for resource references that will be part of a resource group
- ▶ `ReferencedResource`
 - `AutomationDomain`

- Name
- Class
- Node

Example 7-6 shows the resource reference definitions for our environment in XML format.

Example 7-6 ResourceReference element definitions

```
<!-- Resource Reference for the HTTP tier -->
  <ResourceReference name="http_rg_REF">

    <Description>
      Reference to apache_rg resource group defined in
      the httpd_SA_Domain first level domain running the
      HTTP servers for our sample environment.
    </Description>

    <Owner>ITS0 Redbook Team</Owner>

    <DesiredState>Online</DesiredState>

    <ReferencedResource>
      <AutomationDomain>httpd_SA_Domain</AutomationDomain>
      <Name>apache_rg</Name>
      <Class>IBM.ResourceGroup</Class>
    </ReferencedResource>

  </ResourceReference>

<!-- Resource Reference for the application tier -->
  <ResourceReference name="was_rg_REF">

    <Description>
      Reference to websphererg resource group defined in
      the was_SA_Domain first level domain running IBM
      WebSphere and our sample J2EE application.
    </Description>

    <Owner>ITS0 Redbook Team</Owner>

    <ReferencedResource>
      <AutomationDomain>was_SA_Domain</AutomationDomain>
      <Name>websphererg</Name>
      <Class>IBM.ResourceGroup</Class>
    </ReferencedResource>

  </ResourceReference>
```

```

<!-- Resource Reference for the database tier on z/OS      -->
  <ResourceReference name="db2_rg_REF">

    <Description>
      Reference to APPLGR_DB8Q resource group defined in
      the SC64N first level domain running IBM DB2 on z/OS
      hosting the database for our sample application.
    </Description>

    <Owner>ITS0 Redbook Team</Owner>

    <ReferencedResource>
      <AutomationDomain>SC64N</AutomationDomain>
      <Name>APPLGR_DB8Q</Name>
      <Class>APG</Class>
      <Node>SC64</Node>
    </ReferencedResource>

  </ResourceReference>

```

Group definitions

Based on the automation requirements described in “Automation requirements and policy overview” on page 230, we group resources from different first-level domains that, in our scenario, must be managed and monitored as one unit. This also makes creating the definition of relationships between these resource references easier later.

For our scenario, we create a resource group named ITS0E2E_Backend_RG that groups the resource references for our application’s middleware and database tier, was_rg_REF and db2_rg_REF resources, as seen in Figure 7-21 on page 231.

Note: You can nest resource groups. However, a resource group can only be a member of a single resource group.

You define resource groups in the end-to-end automation policy using the <ResourceGroup> element. For each <ResourceGroup> element definition, use the following entries and sub-entries:

- ▶ DesiredState
- ▶ Description
- ▶ Members
 - ResourceReference

Example 7-7 shows the resource group definition for our environment in XML format.

Example 7-7 ResourceGroup element definitions

```
<!-- Resource Group definition -->
<ResourceGroup name="ITSOE2E_Backend_RG" >

    <DesiredState>Online</DesiredState>

    <Description>
        This is a Resource Group for the WAS and DB2 resource references.
    </Description>

    <Members>
        <ResourceReference name="was_rg_REF"/>
        <ResourceReference name="db2_rg_REF"/>
    </Members>

</ResourceGroup>
```

Relationship definitions

Now we are ready to define the relationships and dependencies between the resources and resource groups in our end-to-end automation policy. These definitions provide automation control over starting, stopping, and operations of our heterogeneous application environments. The available relationship definitions are as follows:

- ▶ StartAfter
- ▶ StopAfter
- ▶ ForcedDownBy

Based on the automation requirements we describe in “Automation requirements and policy overview” on page 230, we decide on the following relationships:

http_rg_REF StartAfter ITSOE2E_Backend_RG

In case of a start request is issued for http_rg_REF, the resources of the ITSOE2E_Backend_RG group must be started first.

was_rg_REF StartAfter db2_rg_REF

In case of a start request is issued for was_rg_REF, the db2_rg_REF resource must be started first.

was_rg_REF StopAfter http_rg_REF

In case of a stop request is issued for was_rg_REF, the http_rg_REF resource must be stopped first.

db2_rg_REF StopAfter was_rg_REF

In case of a stop request is issued for db2_rg_REF, the was_rg_REF resource must be stopped first.

http_rg_REF ForcedDownBy ITSOE2E_Backend_RG

The http_rg_REF is brought offline in case of a failure of any resource in the ITSOE2E_Backend_RG resource group.

was_rg_REF ForcedDownBy db2_rg_REF

The was_rg_REF is brought offline in case of a failure of the db2_rg_REF resource.

Figure 7-23 shows the end-to-end relationship definitions described above.

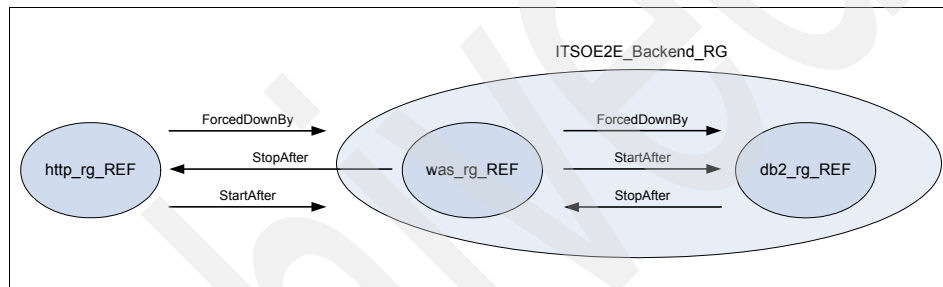


Figure 7-23 Relationship definitions

Relationships are defined in the end-to-end automation policy using the <Relationship> element. For each <Relationship> element definition, we use the following entries and sub-entries:

- ▶ Source
- ▶ Type
- ▶ Target

Example 7-8 shows the relationship definitions for our environment in XML format.

Example 7-8 Relationship element definitions

```
<!-- StartAfter Relationships -->
<Relationship>
  <Source>
    <ResourceReference name="http_rg_REF"/>
  </Source>
  <Type>StartAfter</Type>
  <Target>
    <ResourceGroup name="ITSOE2E_Backend_RG"/>
  </Target>
</Relationship>
```

```

    </Target>
</Relationship>

<Relationship>
  <Source>
    <ResourceReference name="was_rg_REF"/>
  </Source>
  <Type>StartAfter</Type>
  <Target>
    <ResourceReference name="db2_rg_REF"/>
  </Target>
</Relationship>

<!-- StopAfter Relationships -->
<Relationship>
  <Source>
    <ResourceReference name="was_rg_REF"/>
  </Source>
  <Type>StopAfter</Type>
  <Target>
    <ResourceReference name="http_rg_REF"/>
  </Target>
</Relationship>

<Relationship>
  <Source>
    <ResourceReference name="db2_rg_REF"/>
  </Source>
  <Type>StopAfter</Type>
  <Target>
    <ResourceReference name="was_rg_REF"/>
  </Target>
</Relationship>

<!-- ForcedDownBy Relationships -->
<Relationship>
  <Source>
    <ResourceReference name="http_rg_REF"/>
  </Source>
  <Type>ForcedDownBy</Type>
  <Target>
    <ResourceGroup name="ITSOE2E_Backend_RG"/>
  </Target>
</Relationship>

<Relationship>
  <Source>
    <ResourceReference name="was_rg_REF"/>
  </Source>

```

```
<Type>ForcedDownBy</Type>
<Target>
  <ResourceReference name="db2_rg_REF"/>
</Target>
</Relationship>
```

7.5.3 Verifying the end-to-end automation policy file

Once you create the end-to-end automation policy file, you must copy it to the <TSA_E2E_install>/policyPool directory, where <TSA_E2E_install> is the installation directory of the End-to-end Automation Management Component, so that it can be activated.

Prior to activating the end-to-end automation policy, you can verify the policy definitions against the provided policy schema definition and other logical verifications using the IBM Tivoli System Automation for Multiplatforms V2.1 policy checker tool.

In order to perform verification tasks in our newly defined end-to-end automation policy file, we use the policy checker tool by issuing the following command (Example 7-9):

Example 7-9 Policy checker tool

```
C:\IBM\tsamp\eez\bin>eezpolicychecker.bat C:\IBM\tsamp\eez\policyPool\ITS0E2E_v5.xml
CJL0044E The TCP/IP port 9,992 is already in use. The Log Manager cannot start a log command
server.
```

POLICY CHECKER

This program verifies a policy document (*.xml) for use with IBM Tivoli System Automation for Multiplatforms.

(C) COPYRIGHT International Business Machines Corp. 2005
All Rights Reserved.

Policy has been verified.

7.5.4 Activating the end-to-end automation policy file

Once you have created, verified, and placed the end-to-end automation policy file in the <TSA_E2E_install>/policyPool directory, activate it using the IBM Tivoli System Automation for Multiplatforms V2.1 Operations Console.

Note: Before performing the policy activation tasks, ensure that the domain name specified in the <PolicyInformation> element definition matches the name of the end-to-end automation domain showing on the Operations Console.

In order to activate the end-to-end automation policy defined in the policy definition XML file, perform the following steps:

1. Log in to the IBM Tivoli System Automation for Multiplatforms V2.1 Operations Console
2. In the Topology section, select the end-to-end automation domain. In our case, the ITSOE2E domain.
3. On the information area section, select **Policy** → **Activate new policy**, as shown in Figure 7-24.

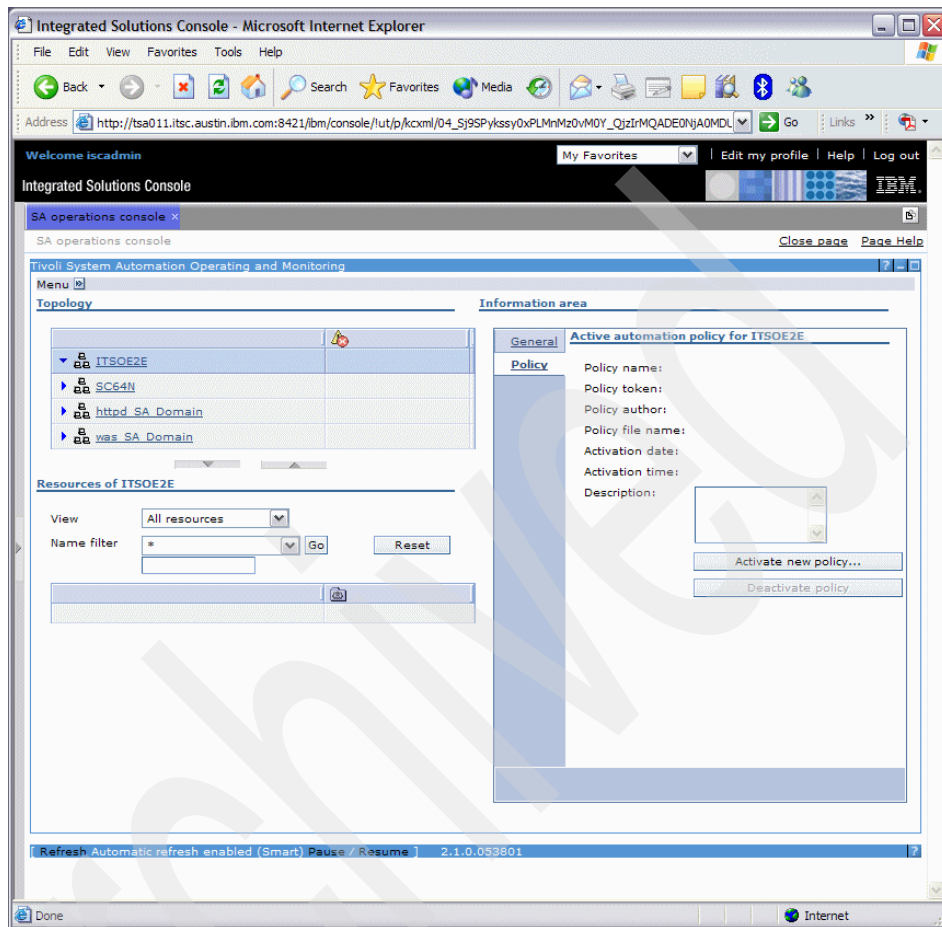


Figure 7-24 Operations Console: Policy information

4. Select the policy in the provided list, as shown in Figure 7-25. Once you select the policy file, a validity check is performed automatically and any warnings display on the same window.

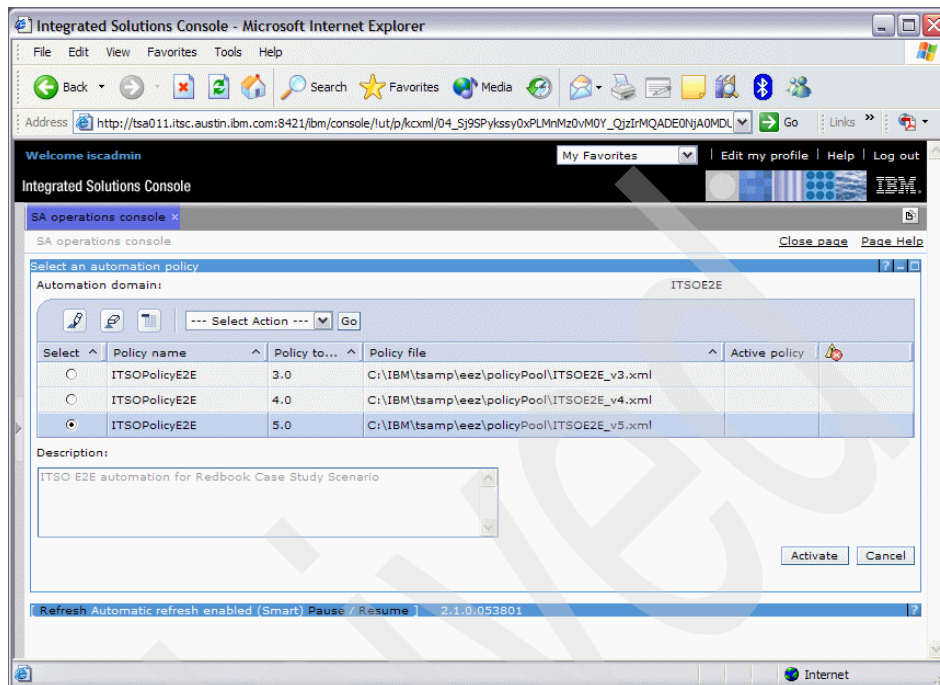


Figure 7-25 Operations Console: Policy selection

5. Select **Activate** to activate the end-to-end automation policy.
6. IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management Component performs policy activation tasks, such as issuing requests to first-level automation domains for obtaining the status, and for changing the desired status of referenced resources.
7. Once active, the policy information displays in the operations console, as in Figure 7-26.

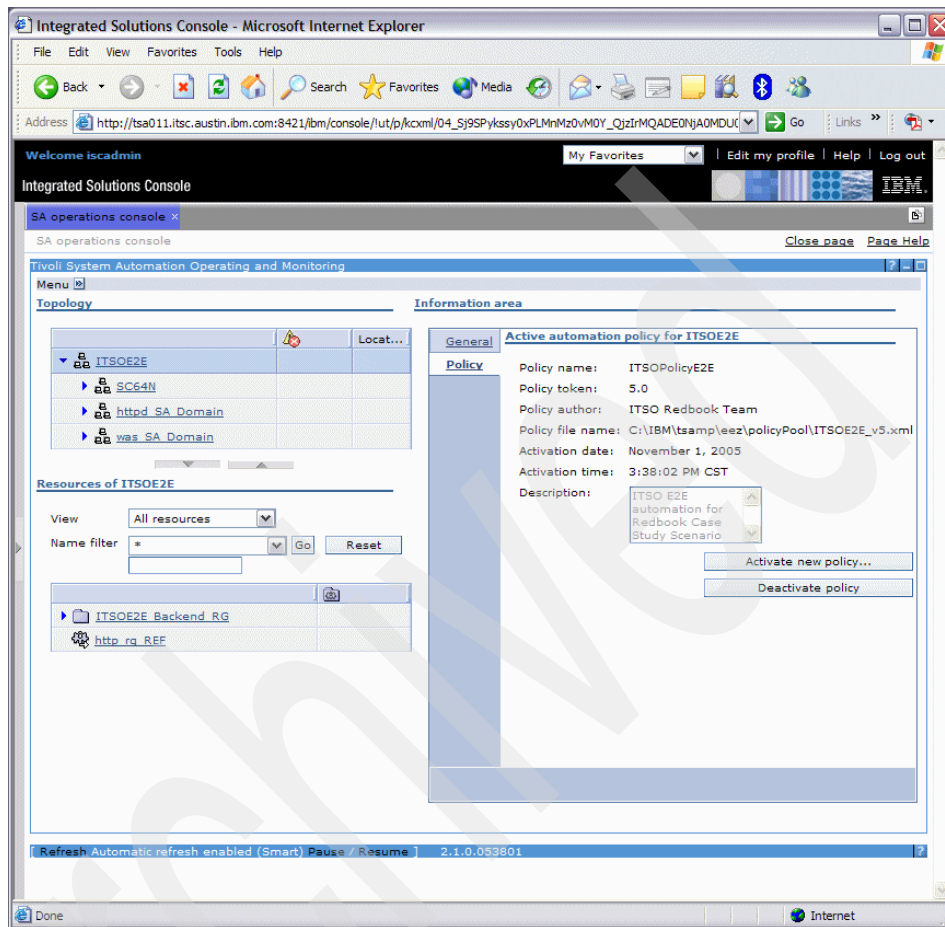


Figure 7-26 Operations Console: Populated policy information

8. Also, verify that the End-to-end Automation Management Component automation engine has picked up the policy activation by using the **eezadm** command as follows:

```
C:\IBM\tsamp\eez\bin>eezadm -M
```

```
About to get current state of the automation engine
```

```
State of automation engine is: RUNNING - policy is activated
```

```
Done
```



Part 3

Appendixes



Troubleshooting overview

This appendix provides an overview of tools, techniques, and pointers for troubleshooting the end-to-end automation environment.

For detailed information, refer to *the IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management User's Guide and Reference*, SC33-8211, and *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04, manuals.

Communication between end-to-end components

A key tool when performing troubleshooting tasks in the end-to-end automation environment is the understanding of the components that make up the end-to-end automation environment and how they interact with each other. Chapter 1, “IBM Tivoli System Automation for Multiplatforms V2.1” on page 3 provides a great deal of information about how elements of the IBM Tivoli System Automation for Multiplatforms V2.1 Base and End-to-end Automation Management Component communicate.

Refer also to the *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management User's Guide and Reference*, SC33-8211, because it provides additional communication scenarios.

Understanding these communication flows is important in order to later be able to understand how to work with logs and trace files, and then to be able to analyze and find the core problems in the end-to-end automation environment.

Location of the root directories

Since the end-to-end automation environment is composed of a variety of applications, here we provide a list of the root directories as reference.

► WAS_INST_ROOT

This is the recommended directory used to install IBM WebSphere Application Server, for example:

Windows	C:\IBM\WebSphere\AppServer
AIX	/usr/opt/IBM/WebSphere/AppServer
Linux	/opt/IBM/WebSphere/AppServer

► EEZ_INST_ROOT

This is the recommended directory used to install the End-to-end Automation Management Component automation engine, for example:

Windows	C:\IBM\tsamp\eez
AIX	/usr/IBM/tsamp/eez
Linux	/opt/IBM/eez/tsamp

► ISC_ROOT

This is the directory used as root to install the Integrated Solutions Console hosting the IBM Tivoli System Automation for Multiplatforms operations console, for example:

Windows	C:\IBM\ISC
AIX and Linux	/opt/IBM/ISC

► EEZ_CONF_ROOT

This is the directory used as root for all configuration files of the End-to-end Automation Management Component automation engine:

Windows	<EEZ_INST_ROOT>\cfg
AIX and Linux	/etc/<EEZ_INST_ROOT>/cfg

► TIVOLI_COM_DIR

This is the Tivoli common directory used by the End-to-end Automation Management Component automation engine and the End-to-end Automation Adapter to place logs and trace files, for example:

Windows	C:\Program Files\IBM\Tivoli\common
AIX and Linux	/var/ibm/tivoli/common

Tivoli common directory

The End-to-end Automation Management Component automation engine and the End-to-end Automation Adapter running on first-level automation domains write their traces and log information in sub-directories of the Tivoli common directory as follows:

► <TIVOLI_COMMON_DIR>/EEZ/logs

This is the directory to find End-to-end Automation Management Component automation engine log and trace files, or End-to-end Automation Adapter logs and trace files, or both if the mentioned components are installed on the same system.

► <TIVOLI_COMMON_DIR>/EEZ/ffdc

This is the directory to find First Failure Data Capture (FFDC) traces of the End-to-end Automation Management Component automation engine and End-to-end Automation Adapter.

► <TIVOLI_COMMON_DIR>/EEZ/lic (Windows only)

This is the directory where the nodelocked license file of the End-to-end Automation Management Component automation engine can be found.

It is possible to change the location of the TIVOLI_COMMON_DIR by changing the log.properties file, which you can find at:

Windows	<TIVOLI_COMMON_DIR>\cfg
---------	-------------------------

Log and trace files

This section provides details of the message log and trace files of the following components:

- ▶ End-to-end Automation Management Component automation engine
- ▶ End-to-end Automation Management Component automation manager
- ▶ End-to-end Automation Adapter
- ▶ IBM Tivoli System Automation for Multiplatforms Operations Console

End-to-end Automation Management Component automation engine

The message log files and trace files of the End-to-end Automation Management Component automation engine are available in the directory `<TIVOLI_COMMON_DIR>/eez/logs`. The message log and trace file are as follows:

- ▶ `msgengine.log` message log file
- ▶ `traceengine.log` message trace file

You can display the content of the `msgengine.log` file in the IBM Tivoli System Automation for Multiplatforms Operations Console for the respective selected automation domain.

Trace and log settings for the End-to-end Automation Management Component automation engine can be observed and changed with help of the End-to-end Automation Management Component configuration tool **cfgeezdmn**.

The message levels for log information are:

- ▶ Error
- ▶ Warning
- ▶ Information (default)

The message levels for trace information, as well as for FFDC, are:

- ▶ Off (no trace information recording)
- ▶ Minimum
- ▶ Medium
- ▶ Maximum

End-to-end Automation Management Component automation manager

Since the End-to-end Automation Management Component automation manager is a J2EE application running on top of IBM WebSphere Application Server, it uses the standard log files and the tracing function of IBM WebSphere Application Server.

The location of the log and trace files of the End-to-end Automation Management Component automation manager is

<WAS_INST_ROOT>\profiles\<profileName>\logs\<serverName>, where:
<profileName> is the name of the profile specified at installation time (for example, default) and <serverName> is the internal name of the IBM WebSphere Application Server server used during installation (for example, server1).

The log and trace files are as follows:

- ▶ SystemOut.log
- ▶ SystemErr.log
- ▶ trace.log

End-to-end Automation Adapter

The message log files and trace files of the End-to-end Automation Adapter are available in the directory <TIVOLI_COMMON_DIR>/eez/logs. The message log and trace file are as follows:

- ▶ msgAdapter.log and msgFlatAdapter.log message log files
- ▶ Several trace files are used. They all have file names as trace*.

Trace and log settings for the End-to-end Automation Adapter can be observed and changed with the help of the End-to-end Automation Adapter configuration tool **cfigsamadapter**.

The message levels for log information are:

- ▶ Error
- ▶ Warning
- ▶ Information (default)

The message levels for trace information as well as for FFDC are:

- ▶ Off (no trace information recording)
- ▶ Minimum
- ▶ Medium
- ▶ Maximum

IBM Tivoli System Automation for Multiplatforms Operations Console

Since the IBM Tivoli System Automation for Multiplatforms Operations Console runs on top of IBM WebSphere Portal Server, it uses the standard log files and the tracing function of IBM WebSphere Portal Server as follows:

- ▶ <WAS_INST_ROOT>\profiles\<profileName>\logs\ISC_Portal
- ▶ <WAS_INST_ROOT>\profiles\<profileName>\csa\logs
- ▶ <ISC_ROOT>/PortalServer/log/*.*

where <profileName> is the name of the profile specified at installation time (for example default).

The log viewer tool

IBM Tivoli System Automation for Multiplatforms V2.1 provides a tool that facilitates the display of the content of log files.

The installation package of the log viewer tool is located in the <EEZ_INST_ROOT>/install directory.

To install the log viewer tool, perform the following steps:

1. Unzip the content of the logviewer214_basics.zip file.
2. Adjust the file viewer.bat (for Windows) or viewer.sh (for Linux or AIX) with the path where an appropriate version of the JVM™.
3. Create a text file named **stdtrace** containing only the following line:

```
select Time,SourceFile,SourceMethod,MessageId,LogText,Exception,Thread
where (ProductId=SAMP)
```

Use the **viewer** command to start the conversion process. For example, to convert the traceengine.xml log file, issue the following command:

```
viewer -f stdtrace traceengine.xml > traceengine.htm
```

The TraceWizard utility

As described earlier, the IBM Tivoli System Automation for Multiplatforms Operations Console writes trace statements to <ISC_ROOT>/PortalServer/log/wps_*.log files. These are Portal Server trace files containing the trace statements of all user sessions since server start time.

With the help of the TraceWizard utility, this trace information can be split into different files which are easier for you to read for a better diagnosis of problems.

The TraceWizard utility is packaged in the <EEZ_INST_ROOT>/lib/eezutils.jar library.

Use the following command to generate readable formatted files.

```
java -classpath <EEZ_INST_ROOT>/lib/eezutils.jar  
com.ibm.eez.ui.trace.TraceWizard wps_<TIMESTAMP>.log
```

The generated files are organized by session IDs. This is especially useful when multiple users are accessing the operations console at the same time, because each of them has a different session ID. Organizing the output files by session ID has the great advantage of following the events and actions in the context of a single user session.

In addition to the full trace for each user session, the tool generates an overview trace that only contains the most important entries, such as user actions and events.

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG247117>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG247117.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
SG247117.zip	Zipped sample files and scripts

System requirements for downloading the Web material

A functional IBM Tivoli System Automation for Multiplatforms V2.1 Base Component environment running on either Linux or AIX is required.

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Abbreviations and acronyms

ACF	Automation Control File	NMC	NetView Management Console
AT	Automation Table	PAM	primary automation manager
EAS	Event Automation Service	PDB	Policy Data Base
EIF	Event Integration Facility	PPI	Program to Program Interface
ESCON	enterprise systems connection	PROCOPS	Processor Operations
FFDC	First Failure Data Capture	RM	resource managers
FLA	First-Level Automation Domains	RODM	Resource Object Data Manager
GDPS	Geographically Dispersed Parallel Sysplex	RSCT	Reliable Scalable Cluster Technology
GRT	Global Response Team	SDF	Status Display Facility
HFS	hierarchical file system	SDK	Software Development Kit
IBM	International Business Machines Corporation	SSH	Secure shell
IMS	IBM Information Management Systems	SSI	Subsystem Interface
IPL	initial program load	SYSOPS	System Operations
ISC	Integrated Solutions Console	TSCF	Target System Control Facility
ISST	IBM Software Services Tivoli	TSO	Time Sharing Option
ITSO	International Technical Support Organization	TWS	Tivoli Work Scheduler
JAR	Java archive	USS	UNIX System Services
JCL	job control language	VM	virtual machine
JDBC	Java Database Connectivity	VSAM	Virtual Storage Access Method
JES	Job Entry Subsystem	VSE	Virtual Storage Extended
JMS	Java Message Service	VTAM	Virtual Telecommunications Access Method
JRE	Java Runtime Environment	XML	Extensible Markup Language
JVM	Java virtual machine		
LDAP	Lightweight Directory Access Protocol		
LTPA	Lightweight Third Party Authentication		

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 260. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Security Configuration in a TCP/IP Sysplex Environment*, SG24-6527-00

Other publications

These publications are also relevant as further information sources:

- ▶ *BM Reliable Scalable Cluster Technology Administration Guide*, SA22-7889
- ▶ *IBM Tivoli System Automation for Multiplatforms V2.1 End-to-end Automation Management User's Guide and Reference*, SC33-8211
- ▶ *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component User's Guide*, SC33-8210-04
- ▶ *IBM Tivoli System Automation for Multiplatforms V2.1 Base Component Reference*, SC33-8212-00
- ▶ *IBM Tivoli System Automation for z/OS V3.1 Planning and Installation*, SC33-8261
- ▶ *IBM Tivoli System Automation for z/OS V3.1 Defining Automation Policy*, SC33-8262-01
- ▶ *IBM Tivoli System Automation for z/OS V3.1 Customizing and Programming*, SC33-8260-01
- ▶ *IBM Tivoli System Automation for z/OS V3.1 End-to-end Automation Adapter*, SC33-8271-01
- ▶ *Tivoli NetView for z/OS V5.1 Installation: Getting Started*, SC31-8872
- ▶ *Tivoli NetView for z/OS Installation: Configuring Additional Components*, SC31-8874

Online resources

These Web sites are also relevant as further information sources:

- ▶ IBM Tivoli System Automation for Multiplatforms product Web site
<http://www-306.ibm.com/software/tivoli/products/sys-auto-linux/>
- ▶ IBM Tivoli System Automation for z/OS product Web site
<http://www-306.ibm.com/software/tivoli/products/system-automation-390/>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

`${SUFFIX}` 193
`$CONFIG_DIR` 192
`*DEFAULT` sample policy 147
`/usr/sbin/cluster/netmon.cf` 118
`<AutomationPolicy>` 232
`<PolicyInformation>` 232
`<Relationship>` 238
`<ResourceGroup>` 236
`<ResourceReference>` 234

Numerics

390-CMOS 34

A

access permissions 221
access roles 219
access roles and group association 220
ACF 176
activating end-to-end policy definitions 240
active log 155
ActivePeerDomain 61, 99
adapter communication 182
adapter configuration 75, 77
adapter directory structure z/OS 190
adapter installation 190
adapter master file 193
adapter script 191
addrgmbr command 66, 108
Affinity 31
aggregate resource 98
AggregateResource 98
AntiAffinity 31
AntiCollocation 31
AOC command 178
AOF603D message 176
AOF AEVNT 186
AOFAPPL 142, 176
AOF CMDSO 133
AOF EXE2E sample exit 197
AOF EXE2E user exit 196
AOF MSGSY 133

APL relationships 147
APPLGR_DB8Q members 180
application group creation 170
application group z/OS 170
application movement 5
ApplicationGroups 167
assign access roles to groups 226
Asynchronous Communication 42
asynchronous communication 16
attributes
 dynamic 58
 persistent 58
authenticate to first-level automation domains 228
authorization user exit 196
Automated operator functions 185
Automated resources prefix 123
Automatic movement of applications 5
Automatic recovery 5
automation control file 176
Automation database 14
automation domain configuration 131
Automation Engine 11
automation engine startup 213
automation goal 112
Automation Manager cold start 140
Automation Manager configuration 135
Automation policy 11, 14
automation policy 12, 61, 67, 109, 120
automation policy example 72
automation policy item 146
Automation requirements 56, 92
automation scripts 61
automation subplex. 135
AutomationDomain entry 234
AutomationDomainName element 232
AutomationPolicy element 232
AUTOS ES 133

B

Base Component overview 6

C

C_DB2_MSTR class 151

- C64STGEN 144
- cfgeezdmn 250
- cfgeezdmn configuration tool 228
- cfigsamadapter 251
- cfigsamadapter command 75, 120
- choice group 232
- Choice groups 30
- chrg command 69, 112
- chrsrc command 71, 118
- Clusters 25
- cn=EEZAdministratorGroup 226
- CNMLINK library 141
- CNMPSSI 141
- CNMSJ010 member 141
- CNMSTYLE 133
- cold start 140
- Collocated relationship 110
- Collocation 31
- com.ibm.eez.aab.invocation-timeout-seconds 199
- CommGroup 106
- COMMNDxx 134
- communication between components 15, 248
- Communication overview 16
- Compound State 30
- COMSTNXT 144
- concepts 24
- Config Quorum 115
- configuration files 120, 228
- Configuration quorum 26
- Configuration RM 8
- configuring End-to-end Automation Management Componen 228
- configuring the adapter 77
- ConfigValidity 61
- connection_oriented mode 187
- Console Help Server 209
- Console Help Server port number 209
- Console Help Service ID 209
- Console Service ID 209
- control policy item 146
- creating user groups 219
- creating users 219
- critical resource 29
- CTLDOWN status 197
- Current runtime status of a resource, online, offline, etc. Desired State 30
- CxxSTGEN 133

D

- DB2 automation 131
- DB2 CONTROL 153
- default gateway 117
- Defining group membership 173
- dependency definitions on z/OS 153
- DependsOn 31
- DependsOn relationship 110
- DependsOnAny 31
- Desired State 30
- DesiredState 234
- Direct access mode 13
- DLOGMOD 133
- domain ID 132
- DSICMSYS 133
- DSIDNMK 133
- DSIPARM 133, 184
- dynamic resources attributes 58
- dynamic select string method.network equivalency
- dynamic string 106

E

- E2E_AUTOOPS 185
- E2EOPER 185
- E2EOPER function 42
- E2EOPRNN 185
- EAS 184
- EAS resource name 186
- EAUTODB 206
- EAUTOUSR tables creator 213
- Eclipse Help system 216
- eez.automation.engine.dif.properties 228
- EEZAdministrator 219
- EEZAsync 219
- EEZAUTOMATIONACCESS 213
- EEZConfigurator 219
- eezdmn command 213
- eezdmn.bat -M 244
- eezdmn.bat -RECONFIG 229
- EEZDOMAINSUBSCRIPTION 213
- EEZEAR 214
- EEZEndToEndAccess 219
- EEZMonitor 219
- EEZOperator 219
- eez-operator-authentication 193, 196
- EEZPolicy.xsd 231
- eezpolicychecker tool 240
- eez-remote-contact-hostname 193

- eezutils.jar library 253
- EIF 15
- eif-send-to-hostname 193
- EJPAL0140I message 225
- EJPAO4003I message 222, 224
- elements in PPI queue 194
- End-to-end Automation Adapter 7
- End-to-end Automation Adapter configuration 118
- End-to-end Automation Adapter shell script 191
- End-to-end automation management 5
- End-to-end Automation Management Component
 - access roles 219
 - End-to-end Automation Management Component applications
 - EEZEAR 214
 - EventServer 214
 - EventServerMdb 214
- End-to-end Automation Management Component automation engine 211
- End-to-end Automation Management Component configuration 228
- End-to-end Automation Management Component configuration files 228
- End-to-end Automation Management Component installation 203
- End-to-end Automation Management Component startup 213
- End-to-end Automation Management Component Tables
 - EEZAUTOMATIONACCESS 213
 - EEZAUTOMATIONRELATION 213
 - EEZDOMAINSUBSCRIPTION 213
 - EEZOPERATORDOMAINFILTER 213
 - EEZOPERATORDOMAINPREFERENCES 213
 - EEZOPERATORHIDDENDOMAIN 213
 - EEZRESOURCESUBSCRIPTION 213
- End-to-end Automation Manager 11–12
- end-to-end automation manager database 206
- End-to-end automation mode 13
- Ene-to-end automation domain name 210
- environment variable 199
- equivalency definition 60
- Event Automation Service 184
- Event Integration Facility 15
- Event port number 121
- Event Publisher 185
- Event Response RM 9
- EventServer 214

- EventServerMdb 214
- EVT_PUBLISHER 185
- EVTOPER function 185, 198
- execution timeout error 199

F

- features 4
- FFDC 249
- First Failure Data Capture 249
- First-level automation mode 13
- floating resource 59, 96, 98
- ForcedDownBy 31
- ForcedDownBy relationship 237

G

- GDPS 37, 39
- GenerateSampleKeys function 191
- genHistoryReport command 205
- Geographically Dispersed Parallel Sysplex 39
- Global initialization file 186
- Global Resource RM 8
- granting access permissions 221
- group management 218
- group membership on z/OS 173
- GRPID 194
- GRPID=XY 184

H

- HAGS 8
- harvesting function 60
- harvesting functionality 29
- HASPARENT relationship type 159, 163
- HATS 8
- HCD trace file 134
- HealthCommand 99
- HealthCommandPeriod 99
- HealthCommandTimeout 65
- HeartbeatPeriod 117
- HFS 191
- hierarchical file system 191
- High availability 24
- High availability and resource monitoring 4
- High Availability Group Services 8
- High Availability Topology Services 8
- high available End-to-end Automation Adapter 119
- HS01 windows service 216
- HSAPIPLC 138

HSAPLIB DD 135
HSAPRMxx member 135

I

I/O OPS 134
IBM DB2 Active Log Dataset 155
IBM DB2 high availability 131
IBM DB2 UDB Enterprise Server Edition V 8.2 203
IBM DB2 V8 Fixpack 10 203
IBM Processor Operations 39
IBM WAS interim fixes 203
 PK00652 204
 PK00842 204
 PK01524 203
 PK04784 204
 PK05321 204
 PK06140 204
 PK06246 204
 PK08802 203
 PK10066 203
IBM WebSphere Application Server V 6 203
IBM WebSphere Application Server V 6 fixpack 2 203
IBM.Application 63, 96, 126
IBM.ConfigRM 8
IBM.Equivalency 126
IBM.ERRM 9
IBM.GblResRM 8
IBM.ManagedRelationship 126
IBM.NetworkInterface 60
IBM.RecoveryRM 8, 95
IBM.ResourceGroup 126
IBM.ServiceIP 59, 126
IBM.TieBreaker class 116
IEASYSxx 134
IEFSSNxx 141
ifconfig command 87
IHSAEVNT 186
IHSAINIT 186
IHSAMCFG 187
IHSSMP3 186
importing policy database 165
ing.PARMLIB 134
ING249E error message 199
ingadapter.sh 191
INGEAMSA sample file 138, 176
INGEBBLD sample job 145
INGECOM 134

INGEDLGA 136
INGEMPF member 137
INGENVSA 141
INGEVE2E 187
INGLIST 180
INGMSG01 133
INGPHOM 138
INGPIPLC 138
INGPIXCU 138
INGPXDST communication task 184
INGXDEBUG=1,2,1 192
INGXINIT 133
INGXINIT initialization member 184
initial program load 35
Installation verification tasks 211
installing End-to-end Automation Management Component 203
InstanceLocation 65
Integrated Solutions Console 208
IPL 35
IPL data collection 134
ISC administrator user 208
ISC port numbers 208
ISC portal application 211
ISC portal application startup 216
ISC settings 208
ISC user ID 217
ISC_PORTAL 216
iscadmin group membership 224
iscadmin user ID 217
ISPF Dialog Panels 44
IsStartable 31
ISSUECMD routine 146

J

J2EE application environment 12
JAAS login modules 193
JACL scripts 102
JAVA_KEYTOOL 192
JDBC Driver (XA) 215
JDBC providers 211
JDBC providers connections 215
JES 131
JRE on z/OS 184

L

LDAP user registry 207
Lightweight Third Party Authentication 206

- link list 193
- Liveness 24
- LNKLST 193
- Location relationships 31
- log data set 155
- log viewer tool 252
- Logger 123
- logviewer214_basics.zip 252
- lseu command 61, 107
- lsrel command 111
- lsrg command 108
- lsrg -m command 66
- lsrpdomain command 57–58, 94
- lsrpnod command 94
- lsrsrc command 98
- lssrc command 94
- LTPA 206

M

- majnode 132
- managing users and groups 218
- mapping access roles to groups 226
- master configuration file 193
- Master Node 95
- Message adapter configuration file 186
- Message Adapter Service 184
- mkequ command 60
- mkrel command 110
- mkrg command 108
- mkrdomain command 57, 93
- mkrsrc command 58, 63, 97
- model policy database 147
- MODETAB 133
- MonitorCommandPeriod 98
- MonitorCommandTimeout 98
- movement of applications 5
- MPFLSTxx member 136
- msgAdapter.log 251
- msgFlatAdapter.log 251

N

- netequ 107
- netmon.cf configuration file 118
- NETUSER.SCNMUXMS 186–187
- NetView domain ID 132
- NetView logon screen 177
- NetView Management Console 36
- NetView system name 132

- NETVIEW.V5R1M0.SCNMUXMS 186
- network equivalency 106
- NMC 36
- node preparation 93
- NodeNameList 98, 103
- NodeNameList attribute 98
- NodeNameList parameter 59
- nominalState 70
- NON-APF authorized task library 138
- NOSTART TASK=MESSAGEA 187

O

- Observed State 30
- OMEGAMON 36
- OPCONDB 206
- openssh 120
- openssl 120
- Operational Quorum 115
- Operational quorum 26, 115
- Operational State 30
- operational state 58
- Operations Console 11–12
- operations console database 206
- operator group 185
- OpState 58, 94, 102

P

- Parallel Sysplex 25
- PARMLIB 134
- PARMLIB data sets 136
- PDB 147
- Peerdomain 25
- Pending online. 112
- persistent resource attributes 58
- pid file 61
- PIDFILE 63
- PK00652 204
- PK00842 204
- PK01524 203
- PK04784 204
- PK05321 204
- PK06140 204
- PK06246 204
- PK08802 203
- PK10066 203
- plug-in configuration file 194
- plugin-domain-name 195
- policies backup 127

- policies maintenance 127
- policy checker tool 240
- policy database 144
- policy database import 165
- policy definitions 70, 113
- policy file activation 240
- PolicyAuthor element 232
- PolicyDescription element 232
- PolicyInformation element 232
- PolicyName element 232
- policyPool 231
- PolicyToken element 232
- POR 35
- port number for Console Help Server 209
- port numbers for ISC 208
- Portal Server log files 252
- Portlet Applications 223
- PostReserveWaitTime 117
- power-on reset 35
- PPI 141
- PPI queue 194
- PPI=INGEVE2E 187
- PPIBQL 184, 194
- preprnode command 57, 93
- Prerequisites 43
- PreReserveWaitTime 117
- Primary Automation Agent 43
- primary automation agent 181
- Primary Automation Manager 43
- Processor Operations facility 36
- proclib 132
- PROCOPS 39
- ProcOps 36
- PROG=xx 137
- Program Directory 131
- Program to Program Interface 141
- PROGxx member 136
- ProtectionMode 65, 104
- pSeries 120

Q

- QUERYSTAT 39
- Quorum 25

R

- RACF authorization 190
- recovery example 71
- Recovery RM 8

- Redbooks Web site 260
 - Contact us xxii
- REDIRSTDERR 191
- REDIRSTDOUT 191
- ReferencedResource entry 234
- relationship definition 157
- relationship definitions 67, 109
- Relationship element definition 238
- Relationships 30
- relationships definitions on z/OS 153
- ReleaseRetryPeriod 117
- Reliable Scalable Cluster Technology 7, 57, 96
- ReprobeData 117
- Request port number 121
- Resource 29
- resource attributes 58
- resource group 112
- Resource group XML definition 127
- Resource grouping 5
- Resource groups 30
- resource manager 8
- Resource Managers 7
- resource monitoring 4
- Resource Monitoring and Contro 8
- Resource Object Data Manager 36
- Resource Reference 30
- Resource reference definition 232
- resource references definitions 233
- Resource States 30
- ResourceGroup element definition 236
- ResourceReference element definition 234
- ResourceType attribute 100
- ResourceType=1 59, 98
- REXX program 196
- RMC 8
- RODM 36
- root directories 248
- RSCT 57, 96
- RSCT harvesting function 60
- RSCT resources 96
- RSCT resources groups 66
- RunCommandsSync 65

S

- Safety 24
- sample policy database 147
- sample policy databases 37
- sampolicy command 127

- samtb_net 116
- SCEERUN2 193
- schema definition file 231
- SCLBDLL 193
- SCLBDLL2 193
- SCNMUXMS 186
- scripts repository 61
- SDF 178
- SDK on z/OS 184
- Secure shell 124
- SelectString 107
- serial floating resource 98
- Service IP Address 58
- ServiceIP 104
- ServiceIP OpState 105
- session ID 253
- shared HFS 191
- shell script ingadapter.sh 191
- SINGMODX 137
- SINGSAMP 136, 138
- SINGSAMP data set 134
- SINGSAMP library 134
- SMP/E 131
- SSH 124
- SSI address space 184
- SSL_PASSWD parameter 191
- SSO domain name 206
- start cache 140
- StartAfter 31
- StartAfter relationship 109, 237
- StartCommandTimeout 98
- starting End-to-end Automation Management Component 213
- startISC command 216
- starttrpdomain command 58, 94
- StartsMeAndStopsMe 163
- startsrc command 95
- Start-stop relationships 31
- Startup Quorum 115
- STATOPT 133
- Status Display Facility 178
- steplib 138
- StopAfter 31
- StopAfter relationship 237
- StopCommandTimeout 98
- stopISC command 217
- stopsrc command 95
- Subclusters 25
- Synchronous Communication 42

- SYS1.SCEERUN 193
- SYS1.SCLBDLL2 193
- SYSOPS 138
- SysOps 36
- Sysplex 25
- sysplex 181
- System Automation operations console 217
- System Operations 36
- system service 209
- SystemErr.log 251
- SystemOut.log 251

T

- Target System Control Facility 36
- TCP/IP port numbers for ISC 208
- terminology 24
- tie breaker 115
- tie breaker definition 115
- tie breaker type EXEC 116
- Tiebreaker 25, 27
- timeout problems 198
- Tivoli common directory 249
- TOWER.SA 144
- TraceWizard utility 252
- troubleshooting 247
- trr troubleshooting 247
- TSCF 36
- TSO 36, 131

U

- UNIX System Services 184
- UNIXPRIV class 190
- user access roles 219
- user exit AOFEXE2E 197
- user group roles
 - EEZAdministrator 219
 - EEZAsync 219
 - EEZConfigurator 219
 - EEZEndToEndAccess 219
 - EEZMonitor 219
 - EEZOperator 219
- user groups 218
- user ID for ISC 217
- users management 218
- USS 184

V

Resource

Node 107

viewer command 252

VTAM 131

VTAM majnode 132

VTAM Major Node 132

W

WAIT parameter 199

wsadmin command 102

WTORs 137

X

XA JDBC Driver 215

XCF group ID 184, 194

XCF sysplex group name 195

XML definition files 4

XML elements definitions 232

XML formatted elements 231

XML policy definition 127

XML schema definition file 231

xxxSTGEN member 144

Z

z/OS adapter file structure 190

z/OS application group 170

z/OS automation domain 131

z/OS group membership 173

z/OS ingadapter.sh 191

z/OS UNIX privileges 190

z/OS UNIX superuser authority 190



End-to-end Automation with IBM Tivoli System Automation for Multipatforms

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Redbooks

End-to-end Automation with IBM Tivoli System Automation for Multiplatforms

**Achieve proactive
high availability of
heterogeneous
environments**

**Covering
multiplatforms,
Linux, AIX, and z/OS**

**Includes real world
case study scenarios**

IBM Tivoli System Automation for Multiplatforms monitors and automates applications distributed across Linux, AIX, and z/OS operating systems by introducing a new product structure with two major components: IBM Tivoli System Automation for Multiplatforms Base Component and the End-to-end Automation Management Component.

IBM Tivoli System Automation for Multiplatforms utilizes an adapter infrastructure to integrate with IBM Tivoli System Automation for z/OS, allowing for more effective high availability, automation, and management of multi-tier applications.

This IBM Redbook introduces the new versions of the IBM Tivoli Systems Automation product family and gives you a broad understanding of the new architecture and components of both IBM Tivoli System Automation for Multiplatforms V2.1 and IBM Tivoli System Automation for z/OS V3.1 using a scenario-based approach.

The instructions given in this redbook are meant to be followed by anyone to successfully install, configure, and set up end-to-end automation management using IBM Tivoli System Automation for Multiplatforms V2.1 and IBM Tivoli System Automation for z/OS V3.1 in environments of any size.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks