

# zPDT 2016 Sysplex Extensions

Frank Kyne

Bill Ogden







International Technical Support Organization

**zPDT 2016 Sysplex Extensions**

May 2016

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

## **Second Edition (May 2016)**

| This edition applies to the December 2015 Edition of Application Developers Controlled Distributions (ADCD).

**© Copyright International Business Machines Corporation 2016. 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

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>IBM Redbooks promotions</b> .....	ix
<b>Preface</b> .....	xi
Authors .....	xi
Now you can become a published author, too! .....	xii
Comments welcome .....	xii
Stay connected to IBM Redbooks .....	xiii
<b>Summary of changes</b> .....	xv
May 2016, Second Edition .....	xv
<b>Chapter 1. Introduction</b> .....	1
1.1 What is sysplex? .....	2
1.2 Base sysplex versus Parallel Sysplex .....	2
1.3 Why did we create this document? .....	5
1.4 Who is this deliverable aimed at? .....	6
1.5 Differences between 2016 Sysplex Extensions and previous version .....	6
1.6 Contents of this document .....	7
<b>Chapter 2. System migration considerations</b> .....	9
2.1 System layout recommendations .....	10
2.1.1 Avoiding updates to ADCD-provided volumes if possible .....	10
2.1.2 Catalogs .....	12
2.1.3 Minimizing changes to provided parmlib members .....	13
2.1.4 Simplifying your system structure .....	14
2.1.5 Summary .....	16
<b>Chapter 3. Sysplex configurations</b> .....	17
3.1 zPDT sysplex configurations .....	18
3.2 Running a sysplex under zPDT .....	19
3.2.1 High-level concepts .....	20
3.2.2 Planning for CFs under zPDT .....	21
3.2.3 Hardware .....	22
3.2.4 Performance .....	23
3.3 Sysplex goals .....	25
3.3.1 General system characteristics .....	25
3.3.2 z/VM-related sysplex limitations .....	28
3.4 System logger .....	29
3.4.1 Resource Recovery Services .....	30
3.4.2 LOGREC .....	32
3.4.3 OPERLOG .....	32
3.4.4 System Management Facility .....	33
3.5 RACF sysplex usage .....	34
<b>Chapter 4. Installing the 2016 Sysplex Extensions</b> .....	35
4.1 Implementing a sysplex under zPDT .....	36
4.2 Implementation overview .....	36

4.3 Setup steps . . . . .	40
4.3.1 Installing the base ADCD system . . . . .	40
4.3.2 Installing z/VM . . . . .	41
4.3.3 Download and create volumes . . . . .	42
4.3.4 Establish a known restart point . . . . .	44
4.3.5 Updating devmap to add new volumes . . . . .	46
4.3.6 Configuring z/VM guests . . . . .	48
4.3.7 Initializing new volumes . . . . .	49
4.3.8 Importing user catalogs . . . . .	49
4.3.9 Creating system-specific system data sets . . . . .	50
4.3.10 Creating zFS file systems for system S0W2 . . . . .	53
4.3.11 Creating Health Checker persistent data data set for S0W2 . . . . .	53
4.3.12 Recataloging master catalog data sets . . . . .	53
4.3.13 SMS changes . . . . .	54
4.3.14 JES2 MAS configuration . . . . .	60
4.3.15 Creating PARMLIB members . . . . .	62
4.3.16 Network definitions . . . . .	66
4.3.17 Creating PROCLIB members . . . . .	69
4.3.18 Creating TCPPARMS . . . . .	70
4.3.19 Creating VTAMLST members . . . . .	71
4.3.20 Customization for Parallel Sysplex complete . . . . .	71
4.4 Bringing up your Parallel Sysplex . . . . .	72
4.4.1 Parallel sysplex implementation checklist . . . . .	75
4.5 Other steps for base sysplex under z/VM . . . . .	76
4.5.1 Changes to z/VM directory . . . . .	76
4.5.2 Changes in parmlib . . . . .	77
4.5.3 Bringing up your base sysplex . . . . .	78
4.6 Implementing a base sysplex without z/VM . . . . .	79
4.6.1 Setting up and starting STP . . . . .	80
4.6.2 Defining CTCs for XCF communication . . . . .	81
4.6.3 Sharing zPDT DASD across PCs . . . . .	82
4.6.4 Required parmlib changes . . . . .	83
4.6.5 Loading the systems . . . . .	85
4.7 Operating your sysplex . . . . .	85
4.7.1 Managing your x3270 sessions . . . . .	86
4.7.2 Post-loading messages . . . . .	86
4.7.3 SMF considerations . . . . .	87
4.7.4 Sysplex policies . . . . .	87
4.7.5 Shutdown . . . . .	88
4.7.6 Useful commands . . . . .	89
4.7.7 Defining new CDSs . . . . .	90
4.7.8 Adding 3390 volumes in Parallel Sysplex . . . . .	92
4.7.9 Adding 3270 terminals in Parallel Sysplex . . . . .	93
4.7.10 Scaling up in Parallel Sysplex . . . . .	94
<b>Chapter 5. Sample DB2 data sharing environment . . . . .</b>	<b>95</b>
5.1 Setting up the 2016 Sysplex Extensions DB2 data sharing environment . . . . .	96
5.1.1 Overview of the DB2 environment . . . . .	96
5.1.2 Implementation . . . . .	96
5.2 Controlling the DB2 data sharing group . . . . .	99
5.3 Maintaining the DB2 environment . . . . .	99
5.4 Sample batch job to test DB2 data sharing . . . . .	100

<b>Chapter 6. Sample CICSplex</b>	103
6.1 CICSplex overview	104
6.2 Setting up the 2016 Sysplex Extensions CICSplex environment	104
6.2.1 SMS changes	104
6.2.2 Parmlib changes for CICS	105
6.2.3 Adding CICS definitions to VTAMLST	105
6.2.4 Importing connect CICS user catalog	106
6.2.5 Defining CICS model log streams	106
6.2.6 Copying CICS procs to USER.PROCLIB	106
6.3 Controlling your CICSplex environment	106
6.4 Migration considerations	109
<b>Appendix A. Sample definitions</b>	111
Sample devmap file	112
Sample z/VM Directory entries	114
Devices that are defined in the ADCD-provided IODF	122
Sample IEASYMxx member for sysplex	123
Sample JCL to allocate system-specific data sets	123
Job to create UNIX System Services data sets for second system	125
zPDT Disk versioning sample script	126
<b>Appendix B. Sysplex couple data sets and policies</b>	127
ADCD CDSs	128
2016 Sysplex Extensions CDSs for Parallel Sysplex	128
Job to create CDSs	130
2016 Sysplex Extensions-provided policies	133
ARM policy	133
CFRM policy	134
SFM policy	139
<b>Appendix C. Alternative disk configuration</b>	141
Alternative configuration for base sysplex	142
<b>Appendix D. Sample IPL flow for sysplex under z/VM</b>	145
Alternative configuration for base sysplex	146
<b>Appendix E. Installation instructions for z/OS 2.2 base</b>	149
Implementing a sysplex under zPDT	150
Implementation overview	150
Setup steps	154
Installing the base ADCD system	154
Install z/VM	155
Download and create volumes	156
Establish a known restart point	158
Updating devmap to add new volumes	160
Configuring z/VM guests	162
Initializing new volumes	163
Importing user catalogs	164
Creating system-specific system data sets	164
Creating zFS file systems for system S0W2	167
Create mount points and symlinks for system-specific file systems	168
Creating Health Checker persistent data data set for S0W2	168
Recataloging master catalog data sets	168
SMS changes	168

JES2 MAS configuration . . . . .	175
Creating PARMLIB members . . . . .	177
Network definitions . . . . .	182
Creating PROCLIB members . . . . .	185
Creating TCPPARMS . . . . .	186
Creating VTAMLST members . . . . .	187
Customization for Parallel Sysplex complete . . . . .	187
Bringing up your Parallel Sysplex . . . . .	188
Parallel sysplex implementation checklist . . . . .	191
Other steps for base sysplex under z/VM . . . . .	192
Changes to z/VM directory . . . . .	192
Changes in parmlib . . . . .	193
Bringing up your base sysplex . . . . .	194
Implementing a base sysplex without z/VM . . . . .	195
Setting up and starting STP . . . . .	196
Defining CTCs for XCF communication . . . . .	197
Sharing zPDT DASD across PCs . . . . .	198
Required parmlib changes . . . . .	200
Loading the systems . . . . .	201
Operating your sysplex . . . . .	202
Managing your x3270 sessions . . . . .	202
Post-loading messages . . . . .	203
SMF considerations . . . . .	203
Sysplex policies . . . . .	204
Shutdown . . . . .	204
Useful commands . . . . .	205
Defining new CDSs . . . . .	206
Adding 3390 volumes in Parallel Sysplex . . . . .	209
Adding 3270 terminals in Parallel Sysplex . . . . .	210
Scaling up in Parallel Sysplex . . . . .	210
<b>Appendix F. Transmission Control Protocol considerations . . . . .</b>	<b>213</b>
TCP definitions . . . . .	214
<b>Appendix G. Additional material . . . . .</b>	<b>215</b>
Locating the Web material . . . . .	215
Using the Web material . . . . .	215
System requirements for downloading the Web material . . . . .	216
<b>Related publications . . . . .</b>	<b>217</b>
IBM Redbooks . . . . .	217
Other publications . . . . .	217
Online resources . . . . .	217
Help from IBM . . . . .	217
<b>Index . . . . .</b>	<b>219</b>



# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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

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

*IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US*

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

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

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

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

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


## COPYRIGHT LICENSE:

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

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

CICS®	IMS™	System z®
CICSplex®	MVS™	VTAM®
DB2®	Parallel Sysplex®	WebSphere®
GDPS®	PartnerWorld®	z Systems™
IBM®	RACF®	z/OS®
IBM z™	Rational®	z/VM®
IBM z Systems™	Redbooks®	z13™
IBM z13™	Redbooks (logo)  ®	zPDT®

The following terms are trademarks of other companies:

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

Linux is a trademark of Linus Torvalds 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.

2015 SUSE LLC. All rights reserved. SUSE and the SUSE logo are registered trademarks of SUSE LLC in the United States and other countries.

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

## Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get personalized notifications of new content
- ▶ Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App



iOS

Download  
Now

Android



## Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



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

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

# Preface

This IBM® Redbooks® publication describes the IBM zPDT® 2016 Sysplex Extensions, which is a package that consists of sample files and supporting documentation to help you get a functioning, data sharing, sysplex up and running with minimal time and effort.

This package is designed and tested to be installed on top of a standard ADCD environment. It provides the extra files that you need to create a two-way data sharing IBM z/OS® 2.1 or z/OS 2.2 sysplex that runs under ADCD in a zPDT environment.

This package differs from the previous zPDT sysplex package in that it provides working examples of more sysplex exploiters. It also is designed to adhere to IBM's sysplex best practice recommendations, in as far as is possible in a zPDT environment.

Although the package was not tested with IBM Rational® Development and Test for IBM System z® (RD&T), it might be used to reduce the effort to create a fully functional sysplex under RD&T.

Conceptually, the package might also be restored and used as a template to create a sysplex environment that is running on a real IBM z Systems™ CPC.

The target audience for this document is system programmers that are responsible for designing, creating, and maintaining IBM Parallel Sysplex® environments. It can also be beneficial to developers that currently maintain their own ADCD environments and want to extend them to add sysplex functions.

## Authors

This book was produced in the International Technical Support Center (ITSO), Poughkeepsie, NY, US.

**Frank Kyne** is the editor of Cheryl Watson's Tuning Letter. Before joining Watson and Walker, Frank was a project leader in the ITSO in Poughkeepsie for 15 years, responsible for ITSO deliverables that were related to sysplex, high availability, IBM GDPS®, and performance. In addition to editing the Tuning Letter, Frank teaches classes and provides technical consulting for z/OS customers globally.

**Bill Ogden** is a retired Senior Technical staff member who continues to work part-time with projects, including new mainframe users and entry-level systems.

Thanks to the following people for their environment and residency support contributions to this project:

**Bob Haimowitz** (Development Support Team [DST], Poughkeepsie Center) for his invaluable advice and assistance in setting up and maintaining the systems throughout the creation of this IBM Redbooks publication.

**Rich Conway** (DST, Poughkeepsie Center) for setting up and maintaining the systems, and providing valuable advice, guidance, and assistance throughout the creation of this IBM Redbooks publication.

Thanks to the following people for their contributions to this project:

**Stephen Anania**  
IBM US

**Mario Bezzi**  
IBM Italy

**Andy Clifton**  
IBM UK

**Alan Murphy**  
Tengri Consulting Ltd., Ireland

**Keith VanBenschoten**  
IBM US

**Marc Van der Meer**  
IBM Netherlands

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate in person or as a remote resident working from your home base.

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

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

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

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

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

- Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:  
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:  
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:  
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:  
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:  
<http://www.redbooks.ibm.com/rss.html>





# Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes  
for SG24-8315-01  
for zPDT 2016 Sysplex Extensions  
as created or updated on May 10, 2016.

## May 2016, Second Edition

This revision adds support for z/OS 2.2. All changes are marked with change bars. The revision includes the following new and changed information:

### New information

- ▶ Added information about jobs that are provided by Application Developers Controlled Distributions (ADCD) to help you migrate Resource Access Control Facility (RACF) to a new ADCD release in “System volumes” on page 11.
- ▶ Added Appendix E, “Installation instructions for z/OS 2.2 base” on page 149 to provide installation instructions for a z/OS 2.2 environment.
- ▶ Added a description about enabling System-Managed Duplexing between the two virtual Coupling Facility Virtual Machines.
- ▶ Added information about Transmission Control Protocol (TCP) and networking customization.
- ▶ The additional material for this document is updated to include the new data sets in support of z/OS 2.2.
- ▶ Added information about the importance of performing an orderly shutdown when you stop your systems.

### Changed information

- ▶ Updated several CICS-related jobs in the SYSPLEX.PARALLEL.CNTL data set to reflect the corrected CICS installation data set names.





# Introduction

This chapter introduces the concepts of base sysplex and Parallel Sysplex for readers who might not be familiar with these environments. In particular, it describes the zPDT and AD/CD support for base and Parallel Sysplex.

The contents of the remainder of this document also is described.

This chapter includes the following topics:

- ▶ 1.1, “What is sysplex?” on page 2
- ▶ 1.2, “Base sysplex versus Parallel Sysplex” on page 2
- ▶ 1.3, “Why did we create this document?” on page 5
- ▶ 1.4, “Who is this deliverable aimed at?” on page 6
- ▶ 1.5, “Differences between 2016 Sysplex Extensions and previous version” on page 6
- ▶ 1.6, “Contents of this document” on page 7

## 1.1 What is sysplex?

Consider that you are in the supermarket, in the queue waiting to pay for your groceries. Then, the single cash register breaks down. You now have two choices: wait until they repair the cash register, or return your items to the shelves and go to another supermarket.

In this example, you were a victim of a “single point of failure.” If there were two cashiers and two cash registers, it is still possible to pay for your groceries if the cash register broke down. The queue might move slower, but it *does* move.

Now, take the analogy one step further. Consider that in our first scenario, each cash register had its own queue of people waiting to check out. When the cash register broke down, you move and join the end of the other queue. At least, you do not have to return all of your groceries; however, all of the people that joined the other queue *after* you are now ahead of you and are served *before* you.

But what if there was a single queue and the person at the top of the queue always goes to whichever cashier was available? All customers are served in the sequence in which they joined the queue. If the business does well and the number of customers in the queue starts to grow, the store can easily add another checkout as needed. At quiet times, the store can reassign one of the checkout assistants to another role. Despite the changes or failures, the customers do not get disrupted and they continue to get the best possible service.

This example shows the basic concepts of sysplex and why successful businesses use it: to avoid single points of failure, deliver high availability, and have the scalability to dynamically grow and shrink in line with their business needs.

## 1.2 Base sysplex versus Parallel Sysplex

You might be familiar with the terms *base sysplex* and *Parallel Sysplex* and wondered what is the difference between the two.

A *base sysplex* is a group of up to 32 z/OS systems that have a common time source and share a set of control data sets called *couple data sets* (CDSs). z/OS provides a software component called Cross Systems Coupling Facility (XCF) that makes it easier for peer programs in the sysplex to communicate with each other. XCF also provides management and monitoring capabilities.

Figure 1-1 shows the basic components that make up a base sysplex.

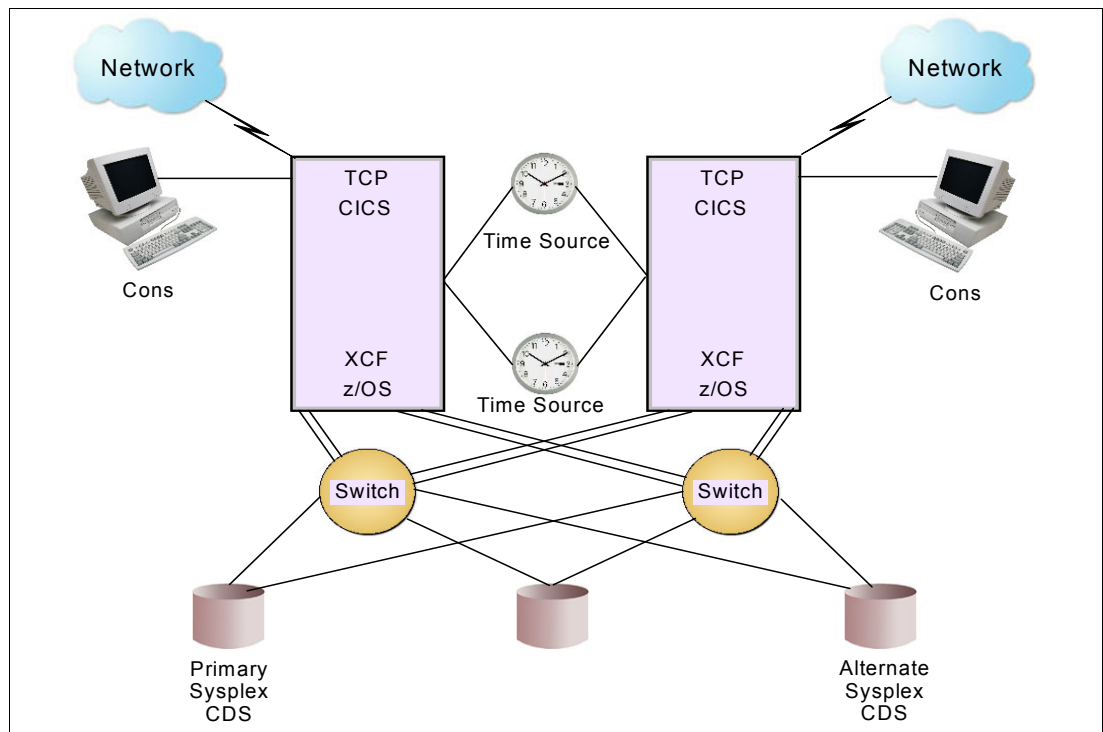


Figure 1-1 Components of a base sysplex

A *Parallel Sysplex* is a base sysplex that also contains one or more Coupling Facilities (CFs). CFs are special purpose logical partitions (LPARs) that provide for high-speed communication between z/OS LPARs. They also provide functions to actively share data with full data integrity and minimal overhead. They also provide the ability to have a single queue of work requests that can be shared and processed by multiple systems in the sysplex. Figure 1-2 shows the components of a Parallel Sysplex.

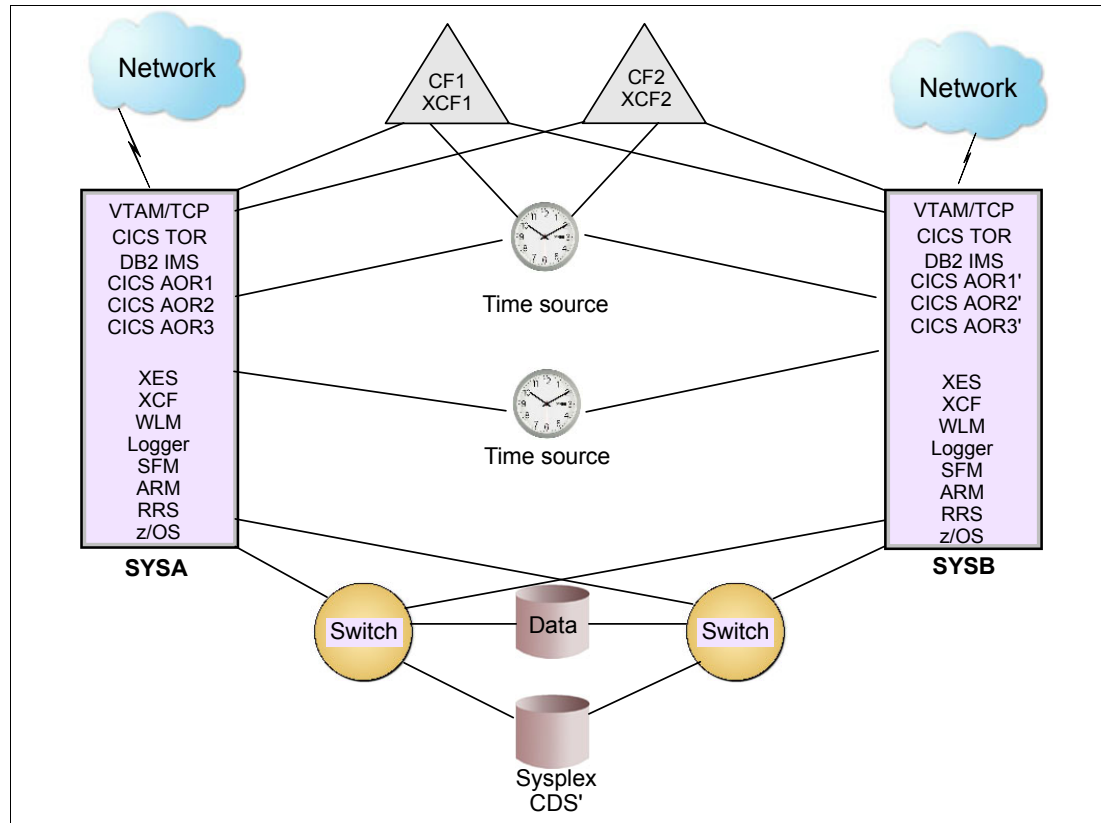


Figure 1-2 Components of a Parallel Sysplex

Although a Parallel Sysplex contains only one extra component compared to a base sysplex (the coupling facility), that one extra component enables many capabilities that are not possible in a base sysplex. Because of this feature, more components are available in the z/OS systems that are shown in Figure 1-2.

Base sysplex provides the infrastructure with which you share things, such as your disk farm, performance policy (Workload Manager), and network access. This feature is often referred to as *resource sharing*. Base sysplex often is considered a stepping stone on the way to Parallel Sysplex.

Parallel Sysplex builds on top of base sysplex and provides the infrastructure that enables data sharing, dynamic workload routing, and queue sharing. A sysplex that uses all of these capabilities is often referred to as a *Platinum Plex*. A Platinum Plex moves you to another level of sophistication and availability.

In a Parallel Sysplex environment, it might be possible to transparently add and remove resources, take systems in and out of the sysplex, automatically route work to whichever system is best able to deliver the required service levels, and even mask planned and unplanned outages from users.

For more information about the capabilities and requirements of a PlatinumPlex, see *Merging Systems into a Sysplex*, SG24-6818, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg246818.html?Open>

## 1.3 Why did we create this document?

Today's business environment is brutally competitive. A large part of that competitiveness is a result of so much business being conducted online.

Thirty years ago, if you wanted to buy something, you drove to the local shopping center, found the item that you wanted to purchase, and brought it to the checkout. There was no easy way to determine how much you might pay for the item elsewhere. Even if price comparison was possible, you were unlikely to travel far just to buy one item. Therefore, you selected your item and waited in the checkout queue. If the cash register stopped working for 10 seconds, you were unlikely to jump back into your car and drive to another store to make the purchase.

Today, you go online to research the item that you are interested in, select the vendor that provides an acceptable level of service and price, click your item, and select "go to check out". If you do not get a response within a few seconds, you abandon that purchase and go to the next website.

Enterprises that cannot provide continuously available service and short response times get into the death spiral of declining revenues, cost cutting (which makes the online service even worse), and further declines in income, which are followed by more cost cutting until the business is eventually taken over or closes down.

In this environment, the strengths of Parallel Sysplex (high availability and scalability) are more relevant than ever.

For software developers, a product that uses the capabilities of Parallel Sysplex can be much more attractive to prospective clients than one that does not. Rather than simply *existing* in the sysplex, a product that uses sysplex capabilities to contribute to the clients continuous availability requirements has real competitive advantages. However, developing such a product requires knowledge of the Sysplex Services Reference and a capability to test the new sysplex-supporting functions.

Many independent software vendors use zPDT and ADCD to develop and test their applications, but might not have the skills or experience to create a Parallel Sysplex environment of your own. These companies requested extensions to the previously available ADCD sysplex download that provide working examples of sysplex-specific functions, such as IBM CICS® Named Counters Server.

For application developers, the good news is that often there is nothing *extra* that you must do to get your application to work in a Parallel Sysplex. However, there are some things that an application should *not* do if it is to work well in a Parallel Sysplex environment.

We developed this package to enable mainframe customers and software developers to quickly and easily create a robust Parallel Sysplex that contains many examples of sysplex exploiters. Rather than many customers and vendors having to go through this process over and over, we hope that the base we provide means a significant time savings and encourages more customers and vendors to extend their support and use of Parallel Sysplex.

## 1.4 Who is this deliverable aimed at?

There are several target audiences that we believe can find value in this deliverable.

The first target is the software developer whose product uses Parallel Sysplex. People in this environment must test their functions to ensure that they work as expected. The Coupling Facility Control Code (that is, the “operating system” of the CF) that is provided with zPDT is the same code that runs on a real IBM z Systems™ mainframe. There are a few functions that can be performed on a real mainframe that are not possible with zPDT. However, generally speaking, zPDT can be used for developing and testing (including recovery testing) sysplex-exploiting products.

The next audience is application developers that use zPDT to develop programs for a production Parallel Sysplex environment. Before moving your application to the production environment, you want to test to ensure that the program is “well-behaved”. For example, it should not have any affinities to specific regions or subsystems or to other transactions.

One of the objectives of Parallel Sysplex is to improve resiliency. Therefore, if you have affinities to some resource in your code and that resource is not available, your application cannot be available. zPDT can be used to test your application to identify such affinities. zPDT does not provide the same level of activity and interaction between components that a test environment that is running on a real z Systems mainframe does, so we still recommend that pre-production testing should be carried out on a real mainframe.

Another audience is anyone that is interested in investigating the function and value of various Parallel Sysplex components. A good example might be IBM MQ Shared Queues. By using IBM MQ Shared Queues, you can place selected IBM MQ queues in a CF structure, meaning that they can now be accessed by all of the members of the queue sharing group. If one IBM MQ is down, the messages can still be placed on (and retrieved from) the queue by any of the other queue managers. This feature provides a significant availability improvement; however, you might want to test this feature in an isolated test environment (zPDT) before implementing it in your real test or production systems.

The challenge of navigating changes through ever-stricter change management processes is becoming a real bottleneck in some environments. Given the real security concerns and risks, this concern is understandable. However, it does slow the evaluation of new functions and new products. Because zPDT is isolated from your real mainframe data (test and production), it should be possible to make a case to exclude your zPDT from change management. Also, because ADCD comes with many products and functions that are already installed and enabled, it might be possible to evaluate new functions in much less time and with less effort than implementing that function in a real mainframe z/OS system.

## 1.5 Differences between 2016 Sysplex Extensions and previous version

Consider the following differences between the previous zPDT sysplex support and the support that is provided by the zPDT 2016 Sysplex Extensions:

- ▶ The management of parmlib members changed significantly. Rather than adhering to the naming convention that was established by ADCD, this package focused on making the sysplex easier to manage by using system symbols to minimize the number of members and the amount of duplication.



- ▶ This package has less differentiation between the base sysplex and Parallel Sysplex setup. The use of the same set of CDSs for base and Parallel sysplexes provides more flexibility to switch back and forth between base and Parallel Sysplex modes. It also reduces the amount of duplication and simplifies the installation. However, you do see some informational messages that result from the coupling facilities not being available if you are running in base sysplex mode, for example.
- ▶ Rather than creating and populating the CDSs during the installation, this package provides the CDSs with policies preinstalled.
- ▶ The previous version used the z/OS default WLM policy. This version uses the ADCD-supplied WLM policy.
- ▶ This version provides the Automatic Restart Manager (ARM) CDSs and an ARM policy. It also provides Sysplex Failure Management (SFM) CDSs and policies that adhere to IBM's best practices recommendations.
- ▶ This version provides a pre-configured IBM DB2® data sharing environment and batch jobs to use that environment.
- ▶ This version provides a pre-configured CICSplex environment, complete with TORs, AORs, CMASs, a CICS Web User Interface (WUI), and the CICS servers for CF Data Tables, Temporary Storage, and Named Counter Server.
- ▶ The XCF signaling infrastructure was reworked to adhere to IBM best practices recommendations.
- ▶ A WLM Scheduling Environment is defined, which can be used to control the systems in a data sharing environment that can run DB2 batch jobs.
- ▶ Advice is provided about ways to structure your system data sets and volumes to simplify the migration from one ADCD release to another.
- ▶ Information is provided about system logger in general, and system logger users in an ADCD sysplex environment in particular.

## 1.6 Contents of this document

This publication includes the following remaining chapters:

- ▶ Chapter 2, “System migration considerations” on page 9 provides hints and tips to help you manage your ADCD systems in a way that reduces the migration effort to move to a new ADCD release in the future.
- ▶ Chapter 3, “Sysplex configurations” on page 17 provides general information about the sysplex that is provided by this package and includes configuration planning information.
- ▶ Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35 provides detailed step-by-step instructions to help you install this package on a z/OS 2.1 base and manage the resulting sysplex environment:
  - Appendix E, “Installation instructions for z/OS 2.2 base” on page 149 contains the equivalent information for a z/OS 2.2 environment.
- ▶ Chapter 5, “Sample DB2 data sharing environment” on page 95 provides step-by-step information to help you get the provided DB2 data sharing environment up and running.
- ▶ Chapter 6, “Sample CICSplex” on page 103 provides information to help you implement and use the provided CICSplex environment.

Also, several appendixes provide supporting information, sample JCL, and so on.





## System migration considerations

As many different possible z/OS configurations exist as there are z/OS customers. As a result, it is impossible to provide a document that comprehensively describes the considerations for migrating from one Application Developers Controlled Distributions (ADCD) release to another for every customer.

However, certain “good housekeeping” practices exist that you can adhere to that make it easier whenever you are ready to move to a new ADCD release or a new release of the 2016 Sysplex Extensions in the future. This chapter describes those practices.

## 2.1 System layout recommendations

Although an ADCD system is not for use in production environments, there is still value in configuring it in a manner that minimizes the time and effort to migrate it to a newer release in the future.

In an ideal world, you download the volumes that are associated with a new ADCD release, initial program load (IPL) the new sysres, and everything works with no extra customization or effort. The other extreme is that you must start from scratch when you install a new release and perform the same customization process when you move to a new release.

Although it is unlikely that you can achieve the ideal scenario, there are things that you can do to bring you closer to that extreme than the other. This chapter provides advice to help you achieve this goal.

### 2.1.1 Avoiding updates to ADCD-provided volumes if possible

When users move to a new ADCD release, they often want to carry over as much of their existing customization as possible, but without having to manually reapply all of the changes to the new ADCD-supplied data sets. A good way to come close to this outcome is to do your utmost to avoid changing ADCD-provided volumes. It helps with this process if you can think of data sets as falling into one of the following categories:

- ▶ SMP-maintained software libraries. These data sets are treated as if they are read-only, with the only updates being applied by SMP when you are applying service or performing an upgrade.
- ▶ System data sets that contain information that is specific to that system or sysplex; for example, your IBM RACF® databases. These databases are provided by ADCD, but they contain information that is unique to your environment.
- ▶ Your own programs and data.

Roughly corresponding to these categories, you ideally have the following types of DASD volumes:

- ▶ Volumes that contain SMP-maintained software libraries, which are known as system volumes. These volumes are supplied as part of the ADCD package and are replaced in their entirety when you move to a new ADCD release.
- ▶ Volumes that contain data sets that are updated by the system or by the people maintaining the system (including the page data sets, RACF data sets, couple data sets (CDSs), parmlib, and proclib). Known as system work volumes, these volumes are also supplied by ADCD; however, you often must perform some migration activity to carry some of the information in these data sets forward to the next release.
- ▶ Volumes that contain your data, programs, databases, and so on. These volumes are known as user volumes and are not provided by ADCD.

In practice, several of the ADCD volumes are a mix of system volumes and system work volumes. ADCD volumes are structured in this way to reduce the number of volumes that are required for the complete ADCD environment. For example, the F1C521 volume contains CICS SMP-maintained libraries and CICS runtime data sets. Nevertheless, the *concept* of trying to segregate volumes that are replaced by a future ADCD release from volumes that you carry forward from one release to the next is still a valuable one.

## System volumes

The sysres is a good example of a system volume. Virtually all of the data sets on that volume should not be changed by you. Taking the May 2015 ADCD release sysres as an example, there are the following exceptions to the recommendation about treating that volume as if it was read-only:

- ▶ SYS1.RACFDS.BACKUP
- ▶ SYS1.PARMLIB
- ▶ SYS1.PROCLIB

Although SYS1.PARMLIB and SYS1.PROCLIB *can* be changed manually, you should try to place any parmlib or proclib changes in the USER.PARMLIB and USER.PROCLIB data sets. Those data sets are placed ahead of the SYS1 equivalents in their respective concatenations. For example, if there is a PROGBO in SYS1.PARMLIB and in USER.PARMLIB, the member from USER.PARMLIB is used.

If you can adhere to this guidance, the only data set on your sysres that is changed is the backup RACF database data set. When you migrate to a new release, you can start with a new ADCD-provided RACF database, or you can use a RACF utility to copy the RACF database from your old release over to the new release.

**Tip:** The RACFUNLD and DBSYNJCL jobs in the ADCD.\*.JCL data set can be used to unload the RACF databases and then compare the old database to the “new” one. The DBSYNJCL job also creates RACF commands to bring the two databases into line with each other. Refer to the DBSYND0C member of ADCD.\*.JCL for information about using the DBSYNC exec.

If you want to use a copy of the old RACF database, you can use the IRRUT200 utility to create an exact copy of a RACF database. For more information about the RACF database utilities, see the RACF database utilities section in *z/OS Security Server RACF System Programmers Guide*, SA23-2287.

The product volumes (F1PRD1, F1PRD2, and F1PRD3) are more examples of system volumes because the data sets that they contain must be updated only by SMP/E for z/OS.

The subsystem volumes (those volumes that are used by DB2, CICS, and so on) as provided by ADCD are something of a hybrid. They contain SMP-maintained data sets *and* data sets that are used by your subsystems.

## System work volumes

The system work volumes are provided by ADCD, but they contain data that is specific to your environment, for example, F1SYS1, F1PAGA/B/C, F1USS1/2, and F1CFG1. Unfortunately, no simple one-size-fits-all migration process exists for these volumes.

**Tip:** The volsers that are used in the bulk of this document (F1nnnn) reflect the May 2015 ADCD. If you install by using a different ADCD release, adjust the volsers. For example, the December 2015 ADCD uses volsers, such as A2nnnn.

Several of these volumes contain data that you might be willing to discard when you move to a new release, for example, your JES2 spool, SMF data sets, page data sets, Logrec, and DAE data sets.

Other volumes contain data that you likely want to merge with the version of that data set that is provided by the new release; for example, your master catalog<sup>1</sup>, RACF database, and SMS SCDS.

You should review the contents of these volumes and identify how you can handle each data set when the time comes to migrate to a new ADCD release.

The 2016 Sysplex Extensions provides DB2 and CICS subsystems that are tailored to run in a Parallel Sysplex environment. They use the ADCD-provided CICS and DB2 software libraries. All of the other data sets that you need to run the subsystems that are provided by the 2016 Sysplex Extensions are contained on a pair of SMS-managed volumes (one pair for CICS and one pair for DB2). Packaging the subsystems in this way makes it easier for you to add or remove those subsystems. It minimizes the number of steps that are required to add those subsystems and makes it easier for you to migrate these subsystems from one ADCD release to another. If you plan to add your own subsystems, you can use this model to build the subsystems in a way that eases the migration effort when you move to a new ADCD release.

We expect future releases of the 2016 Sysplex Extensions to follow this model (the volumes that contain the subsystem work data sets will be delivered with each new release). This configuration gives you the option of replacing your volumes with the new volumes (and cold starting your subsystems), or carrying your volumes forward. However, in the latter case, you must follow the migration process that is provided by the new release of CICS or DB2.

### User volumes

Perhaps the most important thing is to do your utmost to avoid adding *new* data sets to the ADCD-provided volumes. If you have data sets that you know you want to carry forward from one ADCD release to another (such as those containing your application programs, your own data, load libraries, and JCL), you should place those data sets on volumes (we call them *user volumes*) that are not replaced by the new ADCD volumes. Also, ensure that they are cataloged in user catalogs that are also on user volumes.

In as far as reasonable, any new data sets that you create should be placed on a user volume. These user volumes should not contain any system data sets. The objective is to maintain a clear delineation between ADCD-delivered volumes and data sets, and your own volumes and data sets. The fewer changes that you must make to anything on an ADCD-provided volume, the easier it is to migrate to a new ADCD release.

## 2.1.2 Catalogs

You should expect that every new ADCD release contains new data sets in support of new software products or new releases of existing products. These data sets often are cataloged in the master catalog that is provided with the new ADCD.

When you IPL the new ADCD release with your master catalog, you cannot access those data sets unless you specify the volume serial. Therefore, part of the migration to a new ADCD release includes creating a master catalog that contains all of the data sets on the new ADCD system *and* all of your own data sets that you were using on the previous release.

When you are deciding how to create this superset master catalog, you have the following choices:

- Identifying the new data sets and adding those to your current master catalog.

<sup>1</sup> There are several tools on the CBT tape that might help you carry over your catalog entries to a new master catalog. In particular, consider the MCNVT CAT tool. For more information, see this website: <http://www.cbttape.org/cbtdowns.htm>

- Identifying the entries (data sets, user catalogs, aliases, and so on) that are in your current master catalog that are not in the new master catalog. Then, identify the subset of entries that you want to carry forward to the new release and add them to the new master catalog.

In either case, the task is greatly simplified if you maintained a “clean” master catalog; that is, a catalog that has a minimum number of changes to the master catalog that was delivered with your current ADCD release. One of the best ways of achieving that is to catalog all of your user data sets in user catalogs and place those catalogs on your user volumes<sup>2</sup>.

There are a few restrictions that are related to the use of user catalogs. If a data set that is specified in any of the following parmlib members is cataloged in a user catalog, the volser that the data set is on must be specified in the corresponding parmlib member:

- COUPLExx
- IEAFIXcc
- IEALPAXx
- LNKLSTxx
- LPALSTxx
- MSTJCLxx
- LOGREC data sets
- VIO journaling data set (often called SYS1.STGINDEX)
- Data sets in the parmlib concatenation (for example, SYS1.PARMLIB, ADCD.Z21F.PARMLIB, and USER.PARMLIB)

Although this list is long, most of the affected data sets are system data sets. Therefore, they normally are cataloged in the master catalog.

A good way to ensure that you do not forget to create an ALIAS that points at a user catalog when you create a high-level qualifier is to severely restrict the number of users that have RACF update access to the master catalog. This restriction helps highlight situations where a data set might be cataloged in the master catalog because no corresponding ALIAS was defined.

Another useful tip is to use a consistent naming convention for your user catalogs; for example, UCAT.PROJECTA. This naming convention makes it easy to differentiate between aliases that are delivered with ADCD (the ADCD-provided user catalogs all have a high-level qualifier of USERCAT) and aliases that were created by you (and therefore point at a user catalog that is called UCAT.xxxxx).

### 2.1.3 Minimizing changes to provided parmlib members

Most (but not all) parmlib members support the ability to concatenate multiple members. For example, assume that the IGGCAT00 member contains a parameter that you want to override.

One way to achieve this override is to update the IGGCAT00 member in the ADCD.Z21F.PARMLIB member. Although this method works, the next ADCD release ships a new ADCD.anna.PARMLIB data set, which means that the change you made to the IGGCAT00 member in ADCD.Z21F.PARMLIB is lost.

<sup>2</sup> To support RLS management of user catalogs, we strongly recommend that you place any new user catalogs on SMS-managed volumes. Even if you do not intend to use this capability now, placing your user catalogs on SMS volumes now greatly simplifies the migration to RLS-managed catalogs in the future and does not have any downside in the interim.

Another option is to copy the entire IGGCAT00 member to your USER.PARMLIB member. Because USER.PARMLIB is ahead of ADCD.Z21F.PARMLIB in the parmlib concatenation, the USER.PARMLIB version is used and the copy in ADCD.Z21F.PARMLIB is not used.

**Important:** If a member with the same name is in two data sets in the parmlib concatenation, only the first member in the concatenation is read at IPL or if you issue a command that causes the member to be read.

This method is effective from the perspective that you can easily copy your USER.PARMLIB data set contents to the corresponding data set in the new ADCD release. However, it has the drawback that if IBM updates the 00 member, your system does not pick up that change (the system reads only the first member if there are multiple identically-named members).

For components that support multiple concatenated members, we believe that the ideal solution is to create a member with a meaningful suffix (something other than 00), and to populate that member with *only* the parameters that you are overriding. You then specify something such as, CATALOG=(00,PS) in your IEASYSxx member. This specification results in the 00 and the PS members being read during the IPL. If IBM makes changes to the 00 member, those changes are picked up during the IPL.

Unfortunately, not every component supports the use of concatenated parmlib members. Nevertheless, this methodology is a valuable for those components that do support this use.

For more information about effectively managing your parmlib members, see the section titled, “Description and use of the parmlib concatenation” in *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

## 2.1.4 Simplifying your system structure

Our last tip is that you try to keep your system structure as simple as possible. Simplicity makes the administration of your system easier and less prone to human error. It also significantly reduces the problem determination effort if something is not working as planned.

A good example of this guideline is to minimize the number of parmlib members through the intelligent use of system symbols. Example 2-1 shows an excerpt from the IEASYSB1 parmlib member. In the highlighted text, you can see that the page data set names contain the system name as the second qualifier.

*Example 2-1 IEASYSB1 parmlib member*

---

MLPA=00,	SELECT IEALPA00, MLPA PARAMETERS
MSTRJCL=00,	SELECT MSTJCLEX, MASTER JCL
OMVS=00,	SELECT BPXPRMCS
OPI=YES,	ALLOW OPERATOR OVERRIDE
PAGE=(SYS1. <b>SOW1</b> .PLPA.PAGE,	
SYS1. <b>SOW1</b> .COMMON.PAGE,	
SYS1. <b>SOW1</b> .LOCALA.PAGE,	
SYS1. <b>SOW1</b> .LOCALB.PAGE,L),	
PAK=00,	SELECT IEAPAK00
PLEXCFG=ANY,	

---

Example 2-2 on page 15 shows an excerpt from the IEASYSB2 parmlib member. In this example, you can see that the two members are identical, except for the page data set names.



*Example 2-2 IEASYSB2 parmlib member*

---

```
MLPA=00,                SELECT IEALPA00, MLPA PARAMETERS
MSTRJCL=00,             SELECT MSTJCLEX, MASTER JCL
OMVS=00,                SELECT BPXPRMCS
OPI=YES,                ALLOW OPERATOR OVERRIDE
PAGE=(SYS1.S0W2.PLPA.PAGE,
      SYS1.S0W2.COMMON.PAGE,
      SYS1.S0W2.LOCALA.PAGE,
      SYS1.S0W2.LOCALB.PAGE,L),
PAK=00,                 SELECT IEAPAK00
PLEXCFG=ANY,
```

---

An alternative is to maintain a single IEASYSxx member, which is similar to the one that is shown in Example 2-3. In this case, the system name in the page data set name is replaced with the system symbol &SYSNAME. When that member is read by system S0W1, the &SYSNAME in the page data sets is replaced with S0W1. When it is read by S0W2, it is replaced by S0W2.

*Example 2-3 IEASYSxx parmlib member*

---

```
MLPA=00,                SELECT IEALPA00, MLPA PARAMETERS
MSTRJCL=00,             SELECT MSTJCLEX, MASTER JCL
OMVS=00,                SELECT BPXPRMCS
OPI=YES,                ALLOW OPERATOR OVERRIDE
PAGE=(SYS1.&SYSNAME..PLPA.PAGE,
      SYS1.&SYSNAME..COMMON.PAGE,
      SYS1.&SYSNAME..LOCALA.PAGE,
      SYS1.&SYSNAME..LOCALB.PAGE,L),
PAK=00,                 SELECT IEAPAK00
PLEXCFG=ANY,
```

---

System symbols are a powerful tool for simplifying the task of managing your systems. Symbols are defined in the IEASYMxx member. In that member, you can specify values for symbols that are used on every system in the sysplex. You also can filter based on the LPAR name or virtual machine name (or zPDT host name) so that the same symbol resolves to different values, depending by which system they are referenced.

In addition to parmlib members, system symbols can be used in procedures, TSO logon procs, and system commands. Starting with z/OS 2.1, system symbols also (optionally) can be used in normal job JCL (including in SYSIN statements) and even in jobs that are submitted via the internal reader. A future update to this document will contain some real-world examples of how this capability might be used.

Also, starting with z/OS 2.1 is the ability to dynamically change system symbols by using a system command (**SETLOAD IEASYM,xx**). In z/OS 2.2, the maximum length of a system symbol was increased to 44 characters so that a system symbol can contain a value, such as a data set name or an IP address.

If you refer to the IEASYMPS member in the SYSPLEX.PARMLIB data set after you download the 2016 Sysplex Extensions volumes, you will see examples of how we used system symbols to simplify the members that are used for the sysplex.

### 2.1.5 Summary

This first release of the 2016 Sysplex Extensions is a significant change from the previously available ADCD sysplex support, so we do not want to make too many changes at one time.

However, future releases will expand on the examples and advice that is provided in this chapter. If you have any suggestions that you believe might benefit other zPDT users, send an email to: [ogden@us.ibm.com](mailto:ogden@us.ibm.com).



# Sysplex configurations

**Important:** This chapter is not intended as an introduction to z/OS sysplex concepts or operation. We assume that readers understand the basic concepts and terminology that is involved with z/OS sysplex use. For more information about Parallel Sysplex, see Chapter 1, “Introduction” on page 1.

This chapter describes the different types of sysplex that are available and the considerations for running a sysplex under zPDT.

This chapter includes the following topics:

- ▶ 3.1, “zPDT sysplex configurations” on page 18
- ▶ 3.2, “Running a sysplex under zPDT” on page 19
- ▶ 3.3, “Sysplex goals” on page 25
- ▶ 3.4, “System logger” on page 29
- ▶ 3.5, “RACF sysplex usage” on page 34

### 3.1 zPDT sysplex configurations

zPDT can be used in Parallel Sysplex and base sysplex configurations. Both configurations feature the following key requirements:

- ▶ Shared direct access storage device (DASD)
- ▶ Communication among the multiple z/OS systems in the sysplex
- ▶ Synchronized time-of-day clocks among the z/OS systems in the sysplex

Two conceptual implementations (one Parallel Sysplex, one base sysplex) of these three elements (for zPDT) are shown in Figure 3-1.

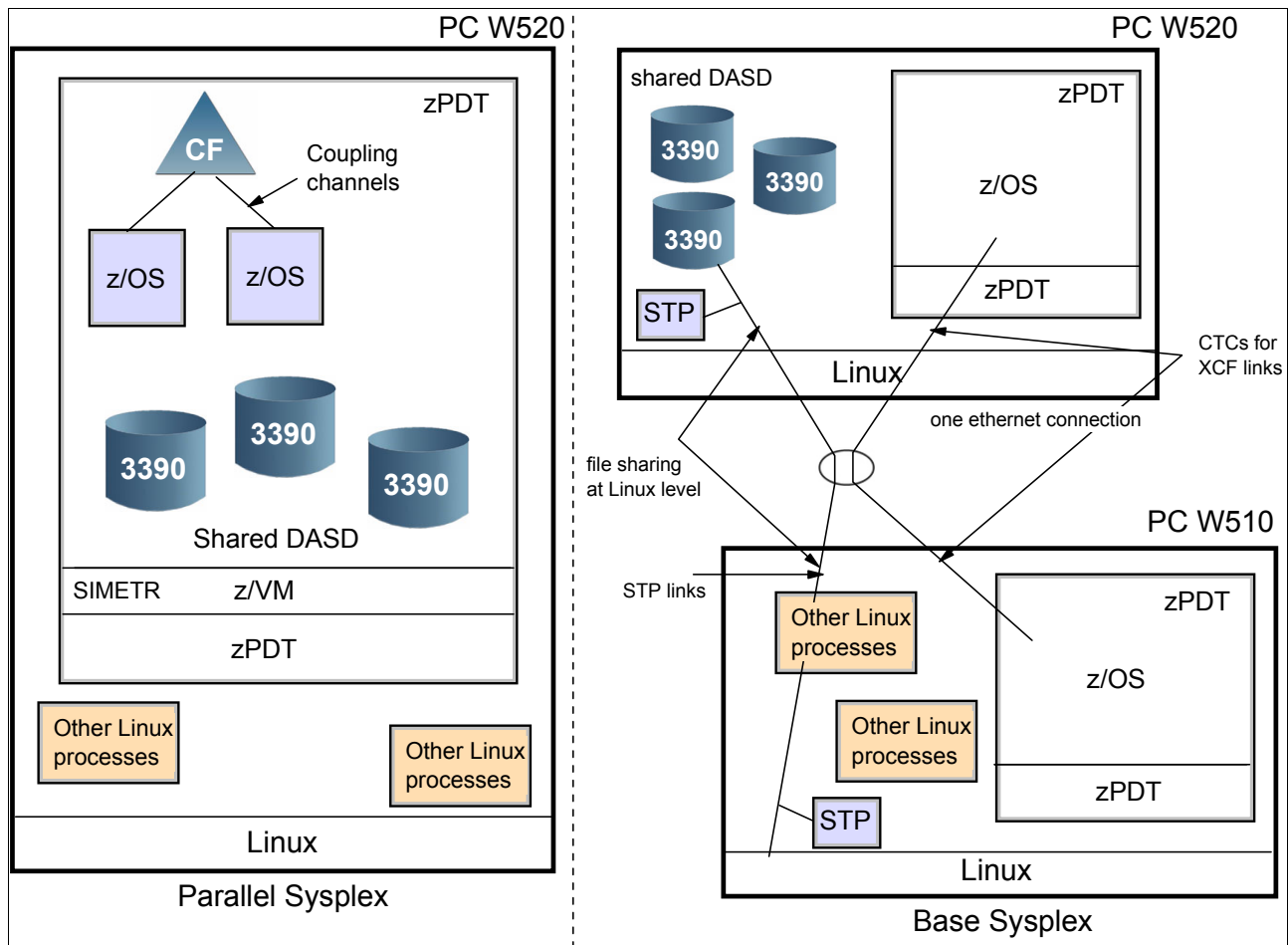


Figure 3-1 zPDT Parallel sysplex and base sysplex

The two implementations are different in the following ways:

- ▶ The Parallel Sysplex is solely within IBM z/VM in a single zPDT instance. z/VM accesses the coupling facility (CF) function<sup>1</sup> and emulates the required coupling channels. z/VM “owns” the emulated DASD and is configured to share them among the z/OS guests.

z/VM also provides a simulated external time reference (ETR)<sup>2</sup> clock that is used by all z/OS guests. z/OS uses global resource serialization (GRS) (via the GRS Star lock structure in the CF) to coordinate data set sharing and other serialization across the members of the sysplex.

<sup>1</sup> This code is the same CFCC licensed code that is used on larger System z machines.

<sup>2</sup> A synchronized time-of-day clock is required for all members of a sysplex.

- The base sysplex configuration involves multiple Linux machines.<sup>3</sup> A Linux file sharing function, such as a Network File System (NFS), provides coordinated DASD functions at the Linux cache level. z/OS uses GRS functions via the channel-to-channel (CTC) links among z/OS systems to coordinate data set sharing and other serialization at the z/OS level. A synchronized time-of-day function is provided by the zPDT Server Time Protocol (STP)<sup>4</sup> function (which is new in zPDT GA6).

In Figure 3-1 on page 18, all of the emulated DASD for all z/OS systems is owned by one Linux machine, which becomes the Linux file server. In practice, the emulated DASD can be spread over several Linux machines, each of which can work as file sharing servers and clients.

Although not shown in Figure 3-1 on page 18, a base sysplex can be under z/VM because z/VM supports virtual channel-to-channel adapters. However, if you have z/VM, it is more likely that you implement a Parallel Sysplex in practice because of the greater functionality it provides.

Conversely, it is *not* possible to implement a Parallel Sysplex across multiple PCs. The main reason is that the support for running the Coupling Facility Control Code is provided by z/VM; the virtual machine is called a Coupling Facility Virtual Machine (CFVM). z/VM requires that all of the systems that are communicating with a CFVM must be in the same z/VM system. A single z/VM system cannot span PCs; therefore, all systems in a zPDT Parallel Sysplex must be in the same PC and running in the same z/VM.

Many details are not shown in Figure 3-1 on page 18; it is intended to show only the general concepts that are involved.

## 3.2 Running a sysplex under zPDT

There are several reasons why you might want to configure your zPDT as a sysplex, including the following most common reasons:

- Developers of products that use sysplex facilities (especially CF operations) need a platform for testing.
- Developers in larger organizations often must share libraries or test data across multiple z/OS systems. Test data can be large and downloading it from larger System z machines to a single shared zPDT volume (or volumes) is more attractive than downloading to multiple separate zPDT machines.
- Application developers must test their programs in a sysplex environment to ensure that they do not do anything that might cause a problem in a production data sharing environment.
- If multiple developers are sharing the system, a Parallel Sysplex gives you the ability to shut down one system while the other developers use the other system, which gives you more flexibility to make changes during normal working hours without disrupting other users of that environment.

Sysplex functions are not apparent to Time Sharing Option (TSO) users or normal batch jobs. However, they are apparent to systems programmers or z/OS system operators. More skills are needed to configure and operate a sysplex system. In addition, configuring and operating a zPDT Parallel Sysplex system requires basic z/VM administration skills. The advantages of sysplex operation must be weighed against the extra required skills.

<sup>3</sup> Multiple zPDT instances in the same Linux machine can also be used.

<sup>4</sup> STP is another method to provide synchronized time-of-day clocks.

## Licenses

You must have the appropriate licenses to have a sysplex running under zPDT. Assuming you have the correct licenses for normal zPDT and z/OS use, 1090 users might need a separate license to use z/VM. Also, 1091 users might need a separate feature (license) to use z/VM and Parallel Sysplex functions. Consult your zPDT provider for more information.

### 3.2.1 High-level concepts

The following high-level concepts must be understood before sysplex is used in a zPDT environment:

- ▶ zPDT does not emulate coupling channels, which are needed for connections to CFs. z/VM provides the coupling channels emulation capability, which is why zPDT Parallel Sysplex systems must operate under z/VM. Multiple z/OS guests can be run under a single z/VM instance in a Parallel Sysplex configuration. Several zPDT instances *cannot* be linked (in the same or separate PCs) to create a Parallel Sysplex configuration.
- ▶ A zPDT instance that is running only the CFCC code cannot be created. The CFCC function is available *only* to guests that are running under z/VM.
- ▶ The zPDT product includes the following delivery streams (based on the token that is used):
  - The 1090 tokens<sup>5</sup> are used by independent software vendors (ISVs) that obtained zPDT through the IBM PartnerWorld® for Developers organization. These tokens can be used with CF functions (under z/VM) to enable Parallel Sysplex operation.<sup>6</sup> The same 1090 tokens are used by many IBM employees, with the same capabilities as the ISV users.
  - The 1091 tokens are used by Rational RD&T clients. These tokens can have optional features enabled. The use of System z CFs is an optional (priced) feature with 1091 tokens.
- ▶ Although our sysplex description is directed to 1090 token users, it also applies to 1091 tokens that have CFs enabled. The 1090 version of zPDT operates with 1090 tokens only, and the 1091 version of zPDT operates with 1091 tokens only.<sup>7</sup>
- ▶ A base sysplex does not involve a CF and can be created with 1090 or 1091 tokens (without extra license features). Several PCs can be connected (each running zPDT and z/OS) to form a base sysplex. If you plan on such a sysplex, see “Considerations for sharing DASD across multi-PC sysplex” on page 24 for information about the performance considerations for NFS.
- ▶ The zPDT plus z/VM system that is hosting the Parallel Sysplex is limited to the number of System z CPs that are allowed by one or more zPDT tokens that are in the PC, with an upper limit of eight CPs.<sup>8</sup>
- ▶ The multiple zPDTs that are creating a base sysplex across multiple PCs are each limited by the token (or tokens) that are used by each zPDT. That is, each zPDT has its own token (or tokens) and there is no collective limit for the number of CPs present in the overall base sysplex.

This token description does not include the practical aspects of the use of a remote token license server. However, the underlying limitations of the number of CPs for a zPDT instance remain.

<sup>5</sup> It is more correct to refer to the *licenses* that are acquired through a token, but for brevity we refer to a token.

<sup>6</sup> Another license agreement covering z/VM usage might be required.

<sup>7</sup> This separation of license control is for zPDT releases dated 2Q13 or later.

<sup>8</sup> Up to eight CPs can be obtained by using multiple tokens or through a special large token.

- ▶ The importance and implementation of data set serialization across all z/OS instances that use shared DASD must be thoroughly understood. As a practical matter, this issue is handled by z/OS GRS. Regardless of whether the sysplex is a base sysplex or a Parallel Sysplex, sharing DASD across multiple z/OS images results in corrupted DASD unless GRS is properly configured and working.
- ▶ A base sysplex requires time-of-day clock synchronization among the z/OS members. To provide this feature, the zPDT STP function must be started before any zPDT instances are started. (z/VM provides a simulated time-of-day function that is used if the sysplex is run under z/VM.) For more information about STP, see the Server Time Protocol chapter in *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:  
<http://www.redbooks.ibm.com/abstracts/sg248205.html?open>
- ▶ We do not address z/OS system tuning in this book. Sysplex operation typically requires attention to various tuning tasks that are handled by z/OS systems programmers. Sysplex operation stresses various sharing functions that are related to ENQ and ENQ+RESERVE processing. IBM APAR II14297 addresses this area with suggestions that might be useful when operating on a zPDT base and is available at this website:  
<http://www.ibm.com/support/docview.wss?uid=isg1II14297>

### 3.2.2 Planning for CFs under zPDT

For 1090 users, no other zPDT device map (devmap) statements are required to use CF functions.

For 1091 users, an extra parameter *is* required in the devmap, as shown in the following example:

```
[system]
memory 10000m
3270port 3270
cpuopt zVM_CouplingFacility
processors 3
```

**Note:** The z/VM directory that is provided by ADCD contains all of the definitions that are necessary for your Parallel Sysplex. The following section is provided only for your information and to help you understand the configuration that is provided by ADCD. If you are familiar with running a sysplex under z/VM, you can skip to 3.2.3, “Hardware” on page 22.

For a Parallel Sysplex, at least one CF guest machine must be defined with a z/VM directory entry by using statements that are similar to the following example:

```
USER CFCC1 CFCC1 1200M 1200M G
XAUTOLOG CFCONSOL
OPTION CFVM TODENABLE
MACH ESA
CONSOLE 009 3125 T CFCONSOL
```

The CFCONSOL keyword on the CONSOLE statement in this example refers to the name of a z/VM virtual machine. In this case, any messages that are issued by the CF that is running in CFCC1 is sent to the CFCONSOL ID. This configuration allows you to consolidate the messages from all CFs to a single place. The console guest is defined as shown in the following example:

```
USER CFCONSOL CFCONSOL 4M 4M ABCDEFG
```

```
MACH ESA
CONSOLE 009 3215 T
```

The memory size for the CF (shown as 1200M in this example) depends on your requirements, including the size of your structures and how many are allocated concurrently.

The z/VM directory entry for each z/OS guest that uses a CF must contain `OPTION` and `SPECIAL` statements, as shown in the following example:

```
USER ADCD .....
OPTION CFUSER TODENABLE
.....
SPECIAL 1400 MSGP CFCC1           (CFCC1 is the name of a CFCC guest definition)
```

This `SPECIAL` statement defines a CF channel, which is emulated by z/VM. Also, the z/VM directory `MDISK` definition for volumes that contain z/OS couple data sets (CDSs) must contain a special parameter, as shown in the following example:

```
MDISK A9F 3390 DEVNO A9F MWV
DASDOPT WRKALLEG           <==this parameter needed for couple data set volumes
```

For more information about the `DASDOPT` statement, see 4.3.6, “Configuring z/VM guests” on page 48.

### 3.2.3 Hardware

No special base hardware (for the Intel compatible computers that are running Linux and zPDT) or Linux release is required. However, as a practical matter, a Parallel Sysplex configuration normally requires more memory than a simple z/OS configuration under z/VM.

z/VM runs in System z memory. System z memory (in a zPDT environment) is virtual memory that is created by Linux. In principle, Linux virtual memory is loosely related to the real memory of the PC only. In practice, the real memory of the PC should be large enough to avoid paging and swapping by Linux.

In a simple case, base sysplex members are normal z/OS systems that often require no extra memory above what might be normal for that z/OS system’s workload.

As an overall suggestion, our normal zPDT documentation states that the defined System z memory (specified on the `memory` parameter in the `devmap` file) should be *at least* 1 GB smaller than the real PC memory for a dedicated zPDT system. In the case of a Parallel Sysplex configuration, we suggest that 16 GB of real PC memory should be the minimum reasonable size, with more memory being better.<sup>9</sup>

With 16 GB of real memory, defining a 12 - 13 GB System z environment is reasonable. This environment can be used with two z/OS guests (at 4 - 6 GB each, for example), two CFs (1 GB each), and z/VM.

It is important that the memory value is sufficiently smaller than the real memory amount to leave memory for zPDT device managers, a reasonable disk cache, and other functions. More Linux workloads require more memory, as does the use of more or larger z/OS guests. If you plan to use a memory value larger than 14 GB, see the “Alter Linux files” section in *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205.

---

<sup>9</sup> These numbers are minimum numbers; much larger systems might be used.



Ignoring hypervisor situations, zPDT does not partition PC memory. The base Linux system has complete control of PC memory; zPDT runs in Linux virtual memory, as with any other Linux application. However, zPDT can be a major user of memory and runs faster if there is little paging at the Linux level and ample memory for the Linux disk cache.

Discussions that refer to, for example, an 8 GB notebook that can have a 6 GB zPDT *do not* imply that 6 GB is somehow partitioned solely for zPDT use. Rather, it is a discussion for obtaining best performance by ensuring that memory resources are sufficient. For more information about memory use in a zPDT environment, see the Memory section in the “Function, releases, content” chapter of *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205.

In principle, a 1090-L01 zPDT system (one CP) can be used on a base PC with one processor (“core”) to run a z/VM system with several Parallel Sysplex z/OS guests. In practice, this configuration is not practical and might result in various z/OS timeouts.

For any significant use of a Parallel Sysplex system under z/VM, we suggest *at least* a 1090-L02 model that is running on a PC with a four-way processor (four “cores”). The individual z/OS guests (under z/VM) might be defined with one logical CP each. The PC must have more cores than the number of zPDT processors in use in any instance of zPDT. For example, on a quad core PC, you should not define a zPDT instance with four processors. For more information, see *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205.

The amount of disk space that you need depends on which ADCD volumes you restore and on how many other volumes you have for your own programs and data. One suggestion that we make is that it does not make sense to try to skimp on disk space. At the time of this writing, a 1 TB internal hard disk drive (HDD) costs approximately \$75 US. Trying to run a system with insufficient disk space wastes time, results in abends because of a lack of space, and affects your ability to use the zPDT disk versioning capability.

**Tip:** Do *not* try to save money by using a small HDD in your zPDT PC. The extra cost for a 1 TB HDD compared to a 500 GB HDD (or a 2 TB HDD compared to a 1 TB HDD) is minimal. Providing plenty of disk space is one of the cheapest ways to improve operability of your zPDT sysplex.

### 3.2.4 Performance

A zPDT *Parallel Sysplex* environment is intended for basic development, self-education, minor proof-of-concept work, small demonstrations, and so on. It is not intended for any type of production or stressful work. It should *never* be used to gauge relative performance of any software. The primary performance limitations are in the following areas:

- ▶ Disk access, especially on a notebook with a single, relatively slow disk. Physical disk access is reduced by the normal Linux disk cache functions; the effectiveness of the Linux disk cache is improved if ample PC memory is available. When two or more z/OS systems are accessing the same PC disk or disks (as in a Parallel Sysplex under z/VM), I/O performance becomes a critical factor.
- ▶ Memory management (real and virtual). Paging (by Linux, z/VM, and z/OS guests) must be avoided as much as possible, mostly because of disk bottlenecks. An effective Linux disk cache is essential for reasonable zPDT performance.

- ▶ System z CP processing power. This issue is limited by the number of PC processors that are available, their speed, and the maximum number of CPs that can be defined under zPDT. In a zPDT Parallel Sysplex configuration, the available CPs are shared among all z/OS guests.
- ▶ z/VM overhead under zPDT. This overhead is greater than the z/VM overhead on a real System z machine.

Allowing for these considerations, we found the performance of our Parallel Sysplex to be reasonable when normal development workloads are involved. I/O performance, which is provided by the combination of the PC disk (or disks) and the Linux disk cache in memory, tends to be a limiting factor for more complex workloads. However, “reasonable” is subjective; we cannot predict the performance of *your* workloads on a sysplex system.

A zPDT *base sysplex* has different performance characteristics. In the simple case, only one z/OS (and no z/VM) is present in each base Linux PC. Each z/OS has the full power of its base machine and uses as many CPs as are provided in the zPDT token (or tokens) for that machine.

### Considerations for sharing DASD across multi-PC sysplex

The implementation of shared DASD across multiple Linux PCs has significant performance implications. We noted the following aspects in our sample configuration:

- ▶ In the simple configuration that is shown in Figure 3-1 on page 18, all of our emulated 3390 DASD were in one Linux PC. Our other Linux PC required all DASD access to go through the LAN. When read, active 3390 tracks might be contained in the local Linux caches.
- ▶ We used NFSv4 for our shared file operation.<sup>10</sup> zPDT (with the `-shared` option specified for the `awsckd` device manager in the `devmap` file) issues Linux file locks on file byte ranges that are the target of an operational CCW. NFSv4 then invalidates any matching cached ranges on other sharing machines. This configuration requires considerable communication overhead.
- ▶ We found that shared DASD performance that uses NFSv4 with 1 Gb Ethernet links and placing all emulated DASD on one PC had performance concerns.

We cannot predict whether the performance fits *your* requirements. We can suggest only that you try it and see whether it is acceptable to you.

Some common-sense techniques can help. For example, IPLing multiple sysplex members at the same time creates a much greater stress on I/O responsiveness than is encountered during more normal operation.

- ▶ We found that an environment with more sophisticated (and more expensive) HDDs had much better performance (combined with greater complexity). For more information about our experiences in this area, see Appendix C, “Alternative disk configuration” on page 141.

NFS is not always a high-performance shared file system. It is attractive because it is widely available and simple to configure. The following ways can be used to address the performance issues:

- ▶ Use an alternative Linux shared file system (several are available). The disadvantage is that they are not as easy to configure as NFS, extensive documentation is not generally available, and these alternatives were not exercised by the zPDT development group.

<sup>10</sup> We are told that NFS (as opposed to NFSv4) has an older locking design. We used NFSv4 for base sysplex operation. NFS or NFSv4 might be used for the read-only DASD sharing.

- Reduce the number of shared volumes. That is, provide each of the members of the sysplex with most or all of the standard z/OS volumes as local, unshared volumes. In the extreme case, only obvious functions, such as RACF and JES2, might use shared volumes. Volumes that contain development libraries and test data can be shared. This arrangement might appear obvious, but implementation is not trivial.

The IBM design of sysplex capability assumes that (almost) all volumes are shared and GRS locking is used to serialize access to data sets. Among other changes, extensive GRS parameter customization (in the GRSRNLxx member of PARMLIB) is needed to make this configuration effective. GRS locking is by data set name, not by volume name. Our sample implementation does not explore this approach.

## 3.3 Sysplex goals

Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35 describes an implementation of a base sysplex and a Parallel Sysplex system. The implementations are intended to be practical and you might follow the same steps to create your own sysplex systems.

In comparison to previous versions of the zPDT sysplex support, this version delivers a sysplex that uses more sysplex functions. You can choose which of those functions you want to use.

One of the reasons for delivering this more robust sysplex environment is to provide software vendors and customers with a configuration with which you can easily test software and applications in a realistic sysplex environment. Access to a more real world-like configuration can result in fewer problems making it through to a production environment. We also designed it with an aim to keep the installation and management as easy and flexible as possible, which makes it easier for you to create and test various configurations.

We designed the 2016 Sysplex Extensions to adhere to IBM’s sysplex best practices guidelines (or at least, those guidelines that are applicable to a zPDT environment). By providing a working example of such a configuration, we hope that you can more easily determine whether configuring your systems and subsystems in this manner delivers benefits in your production environments that you are not receiving today.

### 3.3.1 General system characteristics

Our specific examples of sysplex implementations have the following characteristics:

- As far as possible, the volumes that are included in the package contain everything that you need to add the zPDT 2016 Sysplex Extensions to an ADCD system, which results in a base or Parallel Sysplex. Some manual changes are required, but those changes are kept to a minimum and are described in Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35.
- The 2016 Sysplex Extensions package is based on the May 2015 ADCD z/OS 2.1 release. In principle, there is nothing unique in our implementation that is tied to this release, but various details (such as the name of the ADCD data sets) might change from release to release.

- ▶ In this release of the zPDT Sysplex Extensions, we added sample CICSplex and DB2 data sharing environments. These environments are the most widespread sysplex environments in z/OS customers, so we felt that these environments provide the most value to the largest number of users.

Our sysplex implementations include shared consoles, SMF log streams, and RRS log streams. We do not include VTAM or IP failover or pass-through. JES2 Multiple Access Spool (MAS) (or “shared spool”) is configured, even for “normal” z/OS where it does no harm. We also added other Parallel Sysplex users, such as Automatic Restart Manager, Sysplex Failure Manager, and Health Checker use of log streams, along with supporting documentation to help you use and evaluate them.

In future releases of the zPDT Sysplex Extensions, we might extend the provided Parallel Sysplex environments to include IBM IMS™, IBM MQ, and IBM WebSphere® Application Server. In this version, those other subsystems can be used, but the implementation is left as an exercise for the reader.

- ▶ For our Parallel Sysplex configuration, all DASD<sup>11</sup> is “owned” by z/VM and is shared by the z/OS guests. For our multi-PC base sysplex configuration, all DASD is placed on one of the PCs; the other system accesses the DASD through Linux shared file facilities.
- ▶ Only one set of z/OS volumes is involved. It is initially loaded with different parameters to run as one of the following systems:
  - The “normal” ADCD z/OS monoplex system, with no sysplex relationships. (All the standard ADCD z/OS 2.1 IPL parameters can be used in this mode. The system name is S0W1.)
  - The first system in a Parallel Sysplex. (IPL parameter PS<sup>12</sup> is used; the system name is S0W1.<sup>13</sup>)
  - The second system in a Parallel Sysplex. (IPL parameter PS is used; the system name is S0W2.)
  - The first system in a base sysplex. (IPL parameter BS is used; the system name is S0W1.)
  - The second system in a base sysplex. (IPL parameter BS is used; the system name is S0W2.)

An IPL of the system must be done in consistent ways. The “normal” z/OS cannot be used at the same time that one of the sysplex configurations is being used. The S0W1 and S0W2 systems can be loaded in Parallel Sysplex mode or in base sysplex mode, but you should not load one system in one sysplex mode and the other system in a different sysplex mode concurrently.

- ▶ z/VM 6.3 is used as the basis of the Parallel Sysplex system. There is nothing unique in our implementation that is tied to this release.
- ▶ The name of our sysplex is ADCDPL because it is the name of the standard ADCD monoplex and there is no reason to change it. The name remains the same for the monoplex, base sysplex, and Parallel Sysplex.
- ▶ There is no *cold start* and *warm start* distinction among the sysplex IPL parameters. Every load performs a CLPA.
- ▶ A JES2 cold start can be triggered only by loading a “normal” z/OS (ADCD monoplex) and selecting an IPL parameter that produces a cold start. (This parameter is IPL parameter CS in recent ADCD releases.)

<sup>11</sup> We refer to emulated System z DASD here. Other Linux files are in normal Linux locations.

<sup>12</sup> A zPDT IPL statement has the format `ip1 a80 parm 0a82xx`; for example, where the two xx characters are what is referenced as the “IPL parameter” in this description.

<sup>13</sup> The S0W1 name is set in the delivered ADCD z/OS system; we elected to not alter this name.

- The sysplex configurations are mostly determined by members in PARMLIB, PROCLIB, TCPPARMS, and VTAMLST. The ADCD system provides empty libraries, such as USER.PARMLIB, USER.PROCLIB. These USER libraries are concatenated before the standard ADCD and SYS1 libraries. Most of our altered members are placed in the USER libraries by completing the steps that are described in Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35.
- The Linux names (defined in the Linux system's /etc/HOSTNAME) help identify the personal computer that we describe in our examples. We use the name W520 for the first base sysplex machine (which is also running the STP server) and the name W510 for the second base sysplex machine. The Linux names are not meaningful in a Parallel Sysplex environment unless they are needed for external IP name resolution.

Our intended IPL parameters and IP addresses are listed in Table 3-1.

*Table 3-1 IPL parameters and addresses*

	System name	IPL parameters	IP address	OMVS data sets
Base ADCD z/OS	S0W1	CS, 00, and others	192.168.1.80 10.1.1.2	ZFS.ADCD21F
Base sysplex system 1	S0W1	BS or ST	192.168.1.80 10.1.1.2	ZFS.ADCD21F
Base sysplex system 2	S0W2	BS or ST	192.168.1.90 10.1.1.3 <sup>a</sup>	ZFS.BDCD21F
Parallel sysplex system 1	S0W1	PS	192.168.1.80 10.1.1.2	ZFS.ADCD21F
Parallel sysplex system 2	S0W2	PS	192.168.1.90 10.1.1.3	ZFS.BDCD21F

a. We can use 10.1.1.2 for this address because it is a different Linux than the other 10.1.1.2; however, the use of 10.1.1.3 avoids the confusion that it can cause.

Both members of the Parallel Sysplex are under one z/VM, which is running in one Linux system. The members of the base sysplex can be under z/VM (in which case the BS IPL parameter is used) or in two different Linux systems (in which case the ST parameter is used). When running across two PCs, they are sharing all of the 3390 volumes via Linux file sharing facilities. For more information, see Figure 3-1 on page 18.

Figure 3-2 on page 28 shows the DASD volumes that are involved in the sysplex systems. The z/VM volumes might not be used for the base sysplex and can be omitted. As noted in Figure 3-2 on page 28, there is a single IPL volume for z/OS. Different IPL parameters are used to start the different sysplex configurations we provide (monoplex, base sysplex under z/VM, base sysplex spread over two PCs, and Parallel Sysplex).

The small numbers in Figure 3-2 on page 28 are the addresses (device numbers) that we used in the zPDT devmap and in this documentation. There is no need to use these addresses in your sysplex system. The only semi-standard addresses are A80 (for the IPL volume) and A82 (for the IODF and master catalog) and these addresses are *conventions* that are used in ADCD documentation. If you build a sysplex that is based on a different ADCD release, your volsers will differ from those volsers that are shown in Figure 3-2 on page 28.

The CF0001, DB200n, and CICS0n volumes in Figure 3-2 are volumes that we provide as part of the zPDT 2016 Sysplex Extensions. The CF0001 volume contains CDSs and jobs that are related to sysplex setup. You can use other volumes for these functions. There is nothing special about the CF0001 volume except that it is mounted at an address that has the WRKALLEG option in the z/VM directory.

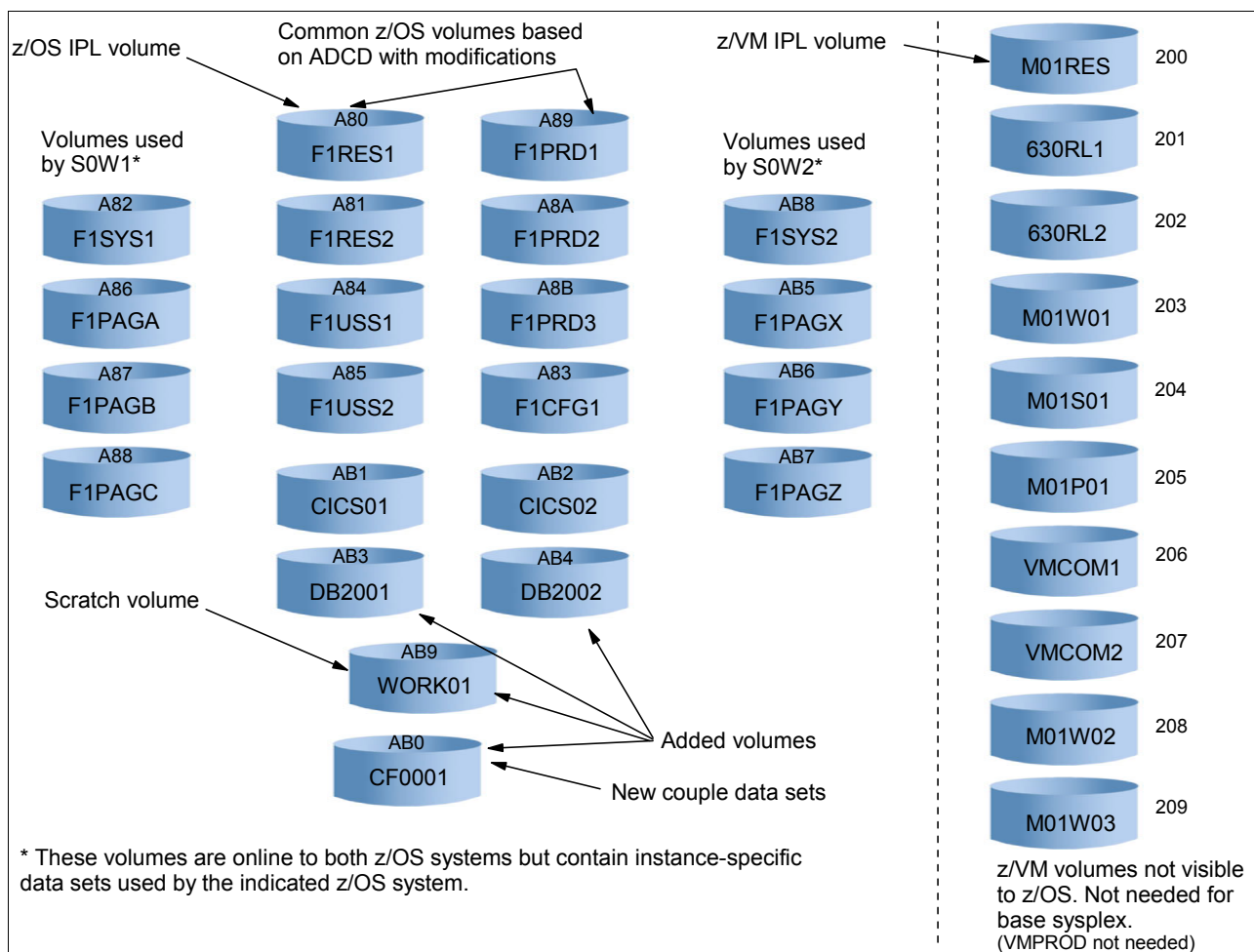


Figure 3-2 Total of 3390 volumes for sysplex

### 3.3.2 z/VM-related sysplex limitations

The following sysplex-related functions are not available when running under z/VM:

- Enhanced Catalog Sharing (ECS)

This function uses a CF cache structure to eliminate the need to read the sharing subcell in the VVDS for catalogs that are defined to use ECS. During system initialization, the ECS code checks to see whether z/OS is running as a virtual machine under z/VM. If it is, an IEC377I message is issued and ECS is not used by that system, even if you define the ECS CF structure and enable catalogs for ECS.

- ▶ BCPii

BCPii is a z/OS system function that provides the ability for a program that is running on z/OS to communicate with the Support Element and HMC of the CPC on which the system is running. This communication is a powerful capability and is used by the XCF System Status Detection Partitioning Protocol to more quickly and more accurately determine the state of a system that stopped updating its time stamp in the sysplex CDS. However, because z/OS is running under the control of z/VM rather than directly on the CPC, BCPii shuts down during the system start if it determines that it is running under z/VM. Message HWI010I (BCPII DOES NOT OPERATE ON A VM GUEST) is issued and the BCPii address space ends.

## 3.4 System logger

The system logger component of z/OS provides generalized logging services to products or functions, such as CICS, IMS, SMF, LOGREC, OPERLOG, and Health Checker. The installation defines “log streams”, which are conceptually similar to infinitely large VSAM linear data sets. Log streams can be shared by multiple systems, but often are dedicated to only one product or function.

When data (known as *log blocks*) is written to a log stream, it is initially placed in interim storage. To protect the availability of the data, system logger keeps two copies of the data while it is in interim storage. These copies are kept in the following locations:

- ▶ A structure in the CF
- ▶ A staging data set on DASD
- ▶ A data space in z/OS

Exactly which two of these locations system logger uses for any specific log stream depends on parameters that you specify when you define the log stream.

After some time, the interim storage portion of a log stream fills up, which triggers a process known as *offload*. Offload consists of the following parts:

- ▶ Any log blocks that are deleted by the application are physically deleted from interim storage.
- ▶ Remaining log blocks are moved to offload data sets until the log stream utilization drops to an installation-specified threshold (the LOWOFFLOAD threshold). The first offload data set is allocated when the log stream is defined (even if it is never used) and more offload data sets are allocated automatically as needed.

Eventually, the older log blocks are deleted by system logger based on a retention period you specify for each log stream, or by the application that owns the data.

A critical data set is the LOGR CDS. The LOGR CDS contains the definitions of all log streams. It also contains information about the staging and offload data sets and the range of log blocks in each offload data set. From that perspective, it can be viewed as being similar to a catalog.

Based on this configuration, you can see that there is a relationship between the following components:

- ▶ The LOGR CDS, which contains the log stream definitions and information about every staging and offload data set.
- ▶ One or more user catalogs that contain entries for the staging and offload data sets.

- The volumes that contain the staging and offload data sets.

If any of these components are unavailable, the log stream is unusable or at a minimum, it is in a LOST DATA status.

For the Sysplex Extensions to ship a predefined log stream, it must provide the following components:

- LOGR CDS that contains all of the definitions
- User catalog that contains the entries for the staging and offload data sets
- Staging and offload data sets

The CICS component of the Sysplex Extensions places the user catalog for the CICS log stream data sets, and the data sets themselves, on the SMS-managed volumes that are provided as part of the package. The LOGR CDS is contained on the CF0001 volume. This configuration means that the entire system logger environment for the CICSplex is provided as part of the package.

However, the data sets for the OPERLOG, LOGREC, RRS, SMF, and Health Checker log streams are not on a volume that is provided by the package. Fortunately, none of these log streams are required at IPL time, which means that the LOGR CDS that is provided by the Sysplex Extensions package does *not* contain the definitions for these log streams. Instead, jobs are provided to create all of the definitions. These jobs are run after the first load of the Parallel Sysplex. For more information, see 4.4, “Bringing up your Parallel Sysplex” on page 72.

Something that you must consider is whether you have any log streams today. Two LOGR CDSs cannot be merged. Therefore, if you move to the CDSs that are contained with this package, all of the data in your log streams is lost. You must review your log streams and determine how you can handle the loss of those log streams. For some log stream users, such as SMF, you can empty the log stream before you shut down in preparation for the move to the new CDSs. For other log streams, such as CICS or RRS, you must perform a cold start after the move.

For more information about system logger in general, see the following resources:

- *Systems Programmer's Guide to: z/OS System Logger*, SG24-6898:  
<http://www.redbooks.ibm.com/abstracts/sg246898.html?Open>
- *z/OS MVS Setting Up a Sysplex*, SA23-1399:  
<http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/>

### 3.4.1 Resource Recovery Services

Resource Recovery Services (RRS) is a z/OS component that provides two-phase commit support across different components and potentially across different systems. Common users of RRS are CICS, IMS, IBM MQ, DB2, and WebSphere Application Server.

RRS uses system logger log streams to track its actions and enable it to recover if there is a failure in RRS or of the system it is running on, or even of the entire sysplex. If it cannot retrieve its log records from system logger, it might be necessary to cold start RRS, which means that it loses information about any in-flight transactions that it was managing.



Depending on the type of work you are doing, this issue might be a factor. However, an important point is that the loss of log stream data in a zPDT environment is far more likely than in a “real” system, which is true because the CF and the connected z/OS systems are all in the one PC. If that PC fails or is shut down abruptly (RRS is stopped abruptly) and you did not define the log streams to use staging data sets, all of the log blocks that were in interim storage are lost.

If this issue is a problem for you, the following measures can be taken to reduce the likelihood of being affected by this issue:

- ▶ Define the RRS log streams to use staging data sets. You achieve this definition by updating the log stream definitions to say STG\_DUPLEX(YES). There is a related parameter, DUPLEXMODE, that controls whether a staging data set is *always* used for that log stream, or only if there is a single point of failure between the z/OS system and the CF containing the structure that is associated with that log stream. However, when running a Parallel Sysplex under z/VM, the CF is always in the same failure domain as all connected systems, so a staging data set always is used for that log stream if STG\_DUPLEX(YES) is specified.
- ▶ Always complete a controlled shutdown of the system. If the system is stopped in an orderly manner and RRS is stopped as part of that shutdown procedure, the log blocks in the CF are moved to an offload data set before RRS disconnects from the log stream. This process means that when the system is brought back up, any required log blocks are still accessible to system logger (and therefore, to RRS).

For more information, see *z/OS MVS Programming: Resource Recovery*, SA23-1395.

At one time, IBM advised customers not to define the RRS archive log stream on the basis that it can contain much information that is rarely required. However, since then, the **SETRRS ARCHIVELOGGING** command was added with which you can dynamically turn on and off the archive log stream.

For that reason, we define the archive log stream and then run the **SETRRS ARCHIVELOGGING,DISABLE** command from the VTAMPS member. If you require the data in the RRS archive log stream, remove that command from the VTAMPS member or run the **SETRRS ARCHIVELOGGING,ENABLE** command to enable the use of that log stream again.

## Considerations for base sysplex

By default, all RRSs in a sysplex are in the same RRS group, meaning that all of the RRSs write to the same set of log streams. In this case, the RRS group name matches the sysplex name.

However, if you are running in a base sysplex configuration, all log streams must be defined as DASDONLY log streams, and DASDONLY log streams cannot be shared between systems. If you want to use RRS in such an environment, you must change the command that is used to start RRS to include a unique group name (something other than the sysplex name). For our base sysplex, we used RRS group names that match the system name.

From a log stream perspective, the second qualifier of the log stream name is the RRS group name. Therefore, the following RRS log streams are defined in the LOGR CDS:

- ▶ RRS.ADCPL.RESTART: Restart log stream for use in Parallel Sysplex mode.
- ▶ RRS.S0W1.RESTART: Restart log stream for system S0W1 when in base sysplex mode.
- ▶ RRS.S0W2.RESTART: Restart log stream for system S0W2 when in base sysplex mode.

We set up the VTAMPS parmlib members so that the default group name is used if the systems are in Parallel Sysplex mode. If the system is started as a base sysplex, it automatically starts RRS in each system by using the system name as the group name. This process should not be apparent to you, except that if you move back and forth between Parallel Sysplex and base sysplex, RRS switches back and forth between different log streams. Therefore, information about any in-flight transactions that were being managed by RRS immediately before the switch are unavailable when the system comes up in the opposite sysplex mode.

For more information about RRS, see *Systems Programmer's Guide to Resource Recovery Services (RRS)*, SG24-6980, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg246980.html?Open>

### 3.4.2 LOGREC

One of the common problems with the LOGREC data sets is that they are a finite size, meaning that they can fill up, often at the time when you really need the information they contain.

Another challenge in a sysplex environment is that problems on one member of the sysplex can often be related to a problem or event on another member of the sysplex. However, if you have a separate LOGREC data set for each system, the relationship between what is happening across the sysplex might not be so obvious.

It is for these reasons that LOGREC was one of the first users of system logger. Because the log streams provide far more space than the LOGREC data sets, you do not have to worry about the data set filling exactly when you need it.

Because multiple systems can write to a single LOGREC log stream, you have a single repository for all LOGREC data, and the data is in chronological order across the whole sysplex. Therefore, a single report shows you what is happening (in the correct sequence) across all of your systems.

Before z/OS 2.2<sup>14</sup>, you used only LOGREC data sets if you specified the LOGREC data set name in the IEASYSxx member, which means that the system always came up in data set mode and you then used the **SETLOGRC** command to switch to logstream mode. The IEASYSPS member that is delivered by the Sysplex Extensions results in the system coming up in data set mode. (It specifies LOGREC=SYS1.&SYSNAME..LOGREC.) If you want to run in logstream mode, update the appropriate VTAMxx parmlib member to issue the **SETLOGRC LOGSTREAM** command during system initialization.

For more information about the use of LOGREC LOGSTREAM mode, see *S/390 Parallel Sysplex: Resource Sharing*, SG24-5666.

### 3.4.3 OPERLOG

Related to the LOGREC log stream is the OPERLOG log stream. OPERLOG is an application that records and merges the hardcopy message set from each system in a sysplex that activates the application. OPERLOG is helpful when you need a sysplex-wide view of system messages; for example, if you are investigating a system problem, it can be invaluable to also see what is happening on the other systems in the sysplex.

---

<sup>14</sup> Starting with z/OS 2.2, it is possible to IPL in LOGREC log stream mode and then switch to DATASET mode.

OPERLOG is automatically enabled at IPL time if you include the OPERLOG keyword on the HARDCOPY DEVNUM statement in the CONSOLxx member, as shown in the following example:

```
HARDCOPY
  DEVNUM(SYSLOG,OPERLOG)
  CMDLEVEL(CMDS)
  ROUTCODE(ALL)
```

If the OPERLOG log stream is not defined (as it is not when you load the Parallel Sysplex for the first time), you are presented with message CNZ4201E OPERLOG HAS FAILED. Job LOGROPR, which is described in 4.4, “Bringing up your Parallel Sysplex” on page 72, allocates the OPERLOG log stream. If you want to enable OPERLOG after the log stream is defined, run the **V OPERLOG,HARDCPY** command.

If you want to stop using OPERLOG and delete the OPERLOG log stream, you must complete the following steps:

1. Run the **V OPERLOG,HARDCPY,OFF** command on every system.  
Running this command stops the system from writing any more messages to the OPERLOG log stream.
2. Wait until everyone that might be using OPERLOG from TSO logs off (or at least, exited SDSF). Each system maintains a connection to the log stream until there is no longer anyone using OPERLOG on that system.

For more information about the use of OPERLOG, see *S/390 Parallel Sysplex: Resource Sharing*, SG24-5666 and *z/OS MVS Planning: Operations*, SA23-1390.

### 3.4.4 System Management Facility

The ability to write System Management Facility (SMF) data to log streams was introduced with z/OS 1.9. The original (and still default) way of running SMF is to write all SMF records that were created by a system to one VSAM data set.

This method made sense when most installations had only one MVS system and when there were only a few products that created SMF records. However, today there are few sites that have only a single z/OS system. There also are many products from IBM and other vendors that create SMF records. In many cases, SMF data is used by extracting a subset of SMF record types from the SMF data sets of every system, merging them, and then running reports that are based on that installation-wide subset of SMF records.

SMF's ability to write its records to one or more log streams is a far more suitable model in these cases. For example, you can create one log stream that contains performance-related SMF records from every member of the sysplex, another log stream that contains the CICS records from every member, and yet another one that contains the security violation SMF records. Each log stream can be defined with retention and availability characteristics that are appropriate for that SMF record type.

The SMFPRMxx member that is provided by the Sysplex Extensions package contains definitions for traditional SMF VSAM data sets (SYS1.MANx) and a single SMF log stream. However, the member also disables the writing of any SMF record, so, as delivered, this sysplex does not create any SMF records. You can easily change this configuration by updating the SMFPRMxx member in USER.PARMLIB to specify ACTIVE rather than NOACTIVE and activating the changed member.

If you make this change, the system starts writing to the SYS1.MANx data sets. If you want to use the log stream instead, update the member to specify RECORDING(LOGSTREAM) and activate the updated member.

For more information about the use of SMF LOGSTREAM mode, see the following resources:

- ▶ *SMF Logstream Mode: Optimizing the New Paradigm*, SG24-7919:  
<http://www.redbooks.ibm.com/abstracts/sg247919.html?Open>
- ▶ *z/OS MVS System Management Facility*, SA38-0667, which is available at this website:  
<http://publibz.boulder.ibm.com/epubs/pdf/iea3g211.pdf>

## 3.5 RACF sysplex usage

The recommended approach when RACF is used in a sysplex is to share the RACF database. If sysplex data sharing is not enabled, RACF uses hardware reserve/release processing on the RACF database to serialize access. This pre-sysplex approach is known as a *shared RACF database*. As more systems join the sysplex, I/O to the shared RACF database increases, which leads to more contention. Updates that are made on one system must be propagated to other systems' local buffers. This propagation requires the deletion of all database buffers because there is no way to identify which particular buffer is affected.

RACF sysplex data sharing allows RACF to invalidate only the buffers that are no longer valid, and uses the CF to provide each RACF with access to many more database buffers than traditional RACF database sharing. RACF sysplex communication, which uses XCF services for communication between RACF subsystems, is a prerequisite to RACF sysplex data sharing.

RACF sysplex communication communicates changes that you make on one system to the other RACFs in the sysplex. Specifically, the following commands are propagated to the other systems:

- ▶ RVARV SWITCH
- ▶ RVARV ACTIVE
- ▶ RVARV INACTIVE
- ▶ RVARV DATASHARE
- ▶ RVARV NODATASHARE
- ▶ SETROPTS RACLIST (classname)
- ▶ SETROPTS RACLIST (classname) REFRESH
- ▶ SETROPTS NORACLIST (classname)
- ▶ SETROPTS GLOBAL (classname)
- ▶ SETROPTS GLOBAL (classname) REFRESH
- ▶ SETROPTS GENERIC (classname) REFRESH
- ▶ SETROPTS WHEN(PROGRAM)
- ▶ SETROPTS WHEN(PROGRAM) REFRESH

Activating RACF sysplex database sharing and sysplex communication requires updates to the RACF data set name table (ICHRDSNT). Because you might have your own RACF databases, we did not provide an updated table because we do not know your RACF database data set names.

For more information about implementing this capability, see the “RACF Sysplex Data Sharing” chapter of *S/390 Parallel Sysplex: Resource Sharing*, SG24-5666.



# Installing the 2016 Sysplex Extensions

**Important:** This chapter is not intended as an introduction to z/OS sysplex concepts or operation. We assume that readers understand the basic concepts and terminology that is involved with z/OS sysplex use. For more information about Parallel Sysplex, see Chapter 1, “Introduction” on page 1.

This chapter describes the installation of the zPDT 2016 Sysplex Extensions package that can be downloaded from the Additional Materials section for this Redbooks document at the following ITSO website:

<ftp://www.redbooks.ibm.com/redbooks/SG248315/>

This document provides step-by-step instructions to help you use the package to turn your single-system ADCD into a Parallel Sysplex<sup>1</sup>.

**Reminder:** This release of the zPDT Sysplex Extensions is based on the May 2015 ADCD release. Although it might be possible to use it with other ADCD releases, small adjustments must be made to accommodate the changing volume names and other possible small changes between ADCD releases.

This chapter includes the following topics:

- ▶ 4.1, “Implementing a sysplex under zPDT” on page 36
- ▶ 4.2, “Implementation overview” on page 36
- ▶ 4.3, “Setup steps” on page 40
- ▶ 4.4, “Bringing up your Parallel Sysplex” on page 72
- ▶ 4.5, “Other steps for base sysplex under z/VM” on page 76
- ▶ 4.6, “Implementing a base sysplex without z/VM” on page 79
- ▶ 4.7, “Operating your sysplex” on page 85

<sup>1</sup> In theory, it should be possible to use the 2016 Sysplex Extensions with some small modifications to create a Parallel Sysplex that is based on an appropriately licensed Rational RD&T environment.

## 4.1 Implementing a sysplex under zPDT

**Important:** This chapter is based on the May 2015 ADCD z/OS 2.1 system. If your base is the December 2015 ADCD release, use Appendix E, “Installation instructions for z/OS 2.2 base” on page 149 instead of this chapter.

Having described in Chapter 3, “Sysplex configurations” on page 17 the reasons why you might want to use a sysplex under zPDT and several of the benefits of sysplex, this chapter helps you combine your ADCD monoplex system with the 2016 Sysplex Extensions package to create your sysplex.

The following configurations are supported:

- ▶ Parallel sysplex that is running under IBM z/VM®
- ▶ Base sysplex that is running under z/VM
- ▶ Base sysplex that is running “native” under z/PDT, spread over two PCs

We assume that most people that are interested in running a sysplex under zPDT want a Parallel Sysplex. Therefore, the primary installation path covers all of the required steps to turn your ADCD system into a two-way Parallel Sysplex.

However, that path also caters for most of the steps that are required for a base sysplex. Therefore, all users that are interested in any form of sysplex under zPDT should follow the installation steps that are described in 4.2, “Implementation overview” on page 36 and 4.3, “Setup steps” on page 40.

If your objective is to have a base sysplex, you also should complete the steps that are described in 4.5, “Other steps for base sysplex under z/VM” on page 76 or in 4.6, “Implementing a base sysplex without z/VM” on page 79, as appropriate.

## 4.2 Implementation overview

One of our objectives with this package was to make its installation as easy as possible. It is not intended as a vehicle to help you learn how to set up a sysplex; rather, it is intended as a tool to help you create a data sharing Parallel Sysplex environment quickly and with as little specialist knowledge as possible. We also wanted to make it easy for you to easily switch back to a standard ADCD environment if you wanted.

Therefore, we wanted to make the package as self-contained as possible and minimize the number of changes you must make to your environment to get this sysplex up and running. This approach also contributes to the objective of making it easier to revert to a standard ADCD environment.

**Note:** For more information about zPDT and ADCD, see *IBM zPDT Guide and Reference: System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

We strongly recommend that you review and are familiar with the contents of that book before you start the implementation of the 2016 Sysplex Extensions.

Before you start the installation, an overview of the start-to-end process is valuable. The implementation steps are summarized in this section. For more information about each step, see 4.3, “Setup steps” on page 40. Unless otherwise noted, all of the new PDS members that are described are created in USER.PARMLIB, USER.PROCLIB, and so forth.

The following summarized tasks are performed to install the 2016 Sysplex Extensions:

1. Install the base ADCD z/OS system.  
You must be familiar with the operation of a z/OS system before you move to a sysplex configuration. If you do not have an ADCD system up and running, you must implement a standard ADCD monoplex system.

**Note:** This chapter covers the basic z/OS environment only. The implementation of the DB2 data sharing group is described in Chapter 5, “Sample DB2 data sharing environment” on page 95 and the CICSplex environments implementation is described in Chapter 6, “Sample CICSplex” on page 103.

2. Install z/VM if your objective is to create a Parallel Sysplex or a base sysplex that is running under z/VM. To gain some experience with running z/OS under z/VM, you can run your ADCD system under z/VM before moving to sysplex mode.
3. Download the volumes that are provided as part of the zPDT 2016 Sysplex Extensions package. The volume serial numbers, virtual address that we used for them, and a brief description of the contents of each volume are listed in Table 4-1.

Table 4-1 Volumes that are delivered by zPDT 2016 Sysplex Extensions package

Volser	Device Number	Use
CF0001	AB0	Couple data sets (CDSs) for Base and Parallel Sysplex. PDSs containing sample JCL, parmlib, Proclib, and other required members.
CICS01, CICS02	AB1 AB2	SMS-managed volumes containing data sets required by the CICSplex subsystems that are provided by this package.
DB2001, DB2002	AB3 AB4	SMS-managed volumes containing the data sets required by the DB2 data sharing group that is provided by this package.

In addition to the volumes that you download, there are several volumes that you must create, initialize, and populate with system-specific data sets. Those volumes are listed in Table 4-2. In this step, you use the zPDT **a1cckd** command to create the Linux data sets that contain the z/OS volumes. In a later step, the volumes are initialized, and then the data sets are allocated on those volumes.

Table 4-2 Volumes that must be created and initialized

Volser	Device Number	Use
F1PAGX F1PAGY F1PAGZ	AB5 AB6 AB7	Page data sets for S0W2 system.
F1SYS2	AB8	System-specific data sets for the S0W2 system
WORK01	AB9	Work volume

**Tip:** If you experience PGT004 wait states in z/VM, it might be that you must add more paging volumes to z/VM. The z/VM IBM Knowledge Center contains the information that is needed to help you update z/VM to inform it of the new paging volumes.

We encountered HCPPGT401I and HCPPGT400I messages on the VM OPERATOR console when we started all of the CICS address spaces on both systems. These messages are pre-cursors to wait states that are caused by lack of paging space.

4. Before you change your working system, it is a good idea to create a known restart point. For more information about creating that point, see 4.3.4, “Establish a known restart point” on page 44.
5. After you download and create the 2016 Sysplex Extensions volumes, update your zPDT device map (devmap) file to add the new DASD volumes. We also suggest adding commands to your devmap file to automatically start more x3270 sessions. Because you are running z/VM and another z/OS instance, it is likely that you need more 3270 sessions than you have now. You also must set up the OSA definitions.
6. The Parallel Sysplex environment that we provide with this package uses two z/OS virtual machines that are called S0W1 and S0W2, and two Coupling Facility Virtual Machines that are called CFCC1 and CFCC2. All of these virtual machines are included in the VM directory that is provided with the ADCD z/VM package. A small change to the VM directory is required to identify the volumes that contain the couple data sets (CDSs).
7. When you use the zPDT `a1cckd` command to allocate the new DASD volumes, the system creates Linux files that are internally formatted to resemble a CKD DASD device. However, at this stage, they do not have a volser; the device resembles a new disk that was not yet initialized. In this step, you use ICKDSF to start the volumes that are listed in Table 4-2 on page 37.
8. Most of the data sets on the volumes that you download are cataloged in user catalogs. To make those catalogs and the data sets that they reference available to your systems (in monoplex mode and when running in sysplex mode), you must IMPORT CONNECT the new user catalogs and create ALIASes for those catalogs.
9. Create the system-specific system data sets (Paging, VIO, SMF, and LOGREC) for system S0W2.
10. The ADCD naming convention of the paging data sets is not supportive of a sysplex environment and the use of system symbols. However, they use the entire volume, which leaves no room to allocate new data sets with more sysplex-appropriate names. Therefore, in this step we delete the page data sets and replace them with smaller data sets that use the same names (for compatibility purposes), and a set that uses the new naming convention.
11. Clone HFS and zFS instance-specific file systems for the second (S0W2) system.
12. Starting with z/OS 2.1, the z/OS Health Checker is started automatically. It also uses a data set to carry information from one IPL forward to the next. In this step, we allocate the S0W2 copy of that data set to avoid a JCL ERROR when the Health Checker is started on S0W2.
13. The 2016 Sysplex Extensions package provides a new set CDSs to be used in place of those CDSs that are provided by the standard ADCD deliverable. Although CDSs that are not cataloged in the master catalog can be used, this option is not the preferred option. In this step, we cataloged again the 2016 Sysplex Extensions-provided CDSs in your master catalog.



14. The ADCD-provided SMS configuration only knows about the S0W1 system and is not aware of the sysplex name. In this step, we configure SMS (by using the ISMF panels) to recognize the sysplex and the second system name (S0W2). You also must define two data classes for System Logger (LOGR) data sets and one for DB2, a new storage class for CICS and DB2 runtime data sets (logs, journals, and so on), and new storage groups for the CICS and DB2 volumes. You also must update the ACS routines to assign the appropriate SMS objects to each data set.
15. The JES2 parms must be altered to define a Multi-Access Spool (MAS) system with two members (S0W1 and S0W2). In this step, you update a member that is used by your system, so you must be careful not to create any syntax errors that can stop your current system from starting.
16. Because we wanted to give you the flexibility to switch back and forth between sysplex mode and monoplex mode, we attempted to isolate all parmlib changes to a new set of members in USER.PARMLIB, rather than changing any of the members in ADCD.xxxx.PARMLIB or SYS1.PARMLIB. In this step, you copy the 2016 Sysplex Extensions-provided parmlib members to USER.PARMLIB or to SYS1.IPLPARM (in the case of the LOADxx member). In general, there is only one new member for each type. In cases where different values were required, we used the following suffixes:
  - BS: Base sysplex under z/VM
  - PS: Parallel sysplex under z/VM
  - ST: Base sysplex spread across two or more PCs
17. Create PROCLIB members for SHUTSYS, VTAMPS, and TCP/IP.
18. Create TCPPARMS members as described later.
19. Create VTAMLST members for ATCCONPS, ATCSTRPS, and OSATRLx.
20. IPL the first member of your new sysplex.
21. Submit a job to define the log streams for LOGREC, OPERLOG, Health Checker, RRS, and SMF.

Complete the following steps if your objective is to have a base sysplex environment that is running under z/VM:

1. Because a base sysplex does not have a coupling facility (CF), it uses CTCs for inter-system XCF communication. For ease of operation, define the CTCs in the VM directory.
2. Another aspect that is related to the lack of a coupling facility is GRS. Because there is no CF, GRS must run in Ring mode rather than Star mode. The IEASYSxx member must be updated to reflect this different mode.
3. When RRS is running in a Parallel Sysplex, all of the RRSs in the sysplex normally are in the same RRS group, meaning that they all share a set of log streams. In a base sysplex, it is not possible to share log streams across systems. Therefore, the Start command for RRS must be changed to specify a different group name for each system.

For a base sysplex that spans more than one PC, the following steps are required (in addition to those steps for a base sysplex running under z/VM):

1. Create a Linux shared file environment.
2. Create a second zPDT system (on another PC) with basic Linux and zPDT.
3. Create a devmap that points to the shared files on the “first” Linux system for all of the 3390 volumes.
4. Add CTC definitions to the devmaps on both systems.
5. Add STP definitions to the devmaps on both systems.

6. Remember to start the zPDT STP function on each Linux system before starting zPDT. zPDT fails if the devmap includes STP definitions and the STP function is not running when zPDT is started.

## 4.3 Setup steps

This section provides detailed descriptions of the steps that are required to use the 2016 Sysplex Extensions to create a Parallel or base sysplex. Most of the steps are common to both types of sysplexes (when run under z/VM). A few extra steps are required to enable the base sysplex under z/VM. Those steps are described in 4.5, “Other steps for base sysplex under z/VM” on page 76.

Before you start working on these steps, we recommend printing Table 4-5 on page 75. That table includes a checklist that you can use to track your progress through the implementation steps. It is also helpful in ensuring that you do not miss any steps.

### 4.3.1 Installing the base ADCD system

We assume that you are familiar with ADCD and likely have an ADCD-based z/OS system up and running in monoplex mode. If you do not, we strongly recommend that you do so now (by using the standard ADCD documentation that is available at this website:

<http://dtsc.dfw.ibm.com/MVSDS/'HTTPD2.ADCD.GLOBAL.SHTML%28READM21F%29'>

You also should use *IBM zPDT Guide and Reference: System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

You also should be familiar with the operation and configuration of that system before extending it (by using this package) to a sysplex.

We used the ADCD May 2015 z/OS 2.1 release as the base for this package. This release included the following volumes:

- ▶ F1RES1
- ▶ F1RES2
- ▶ F1SYS1
- ▶ F1USS1
- ▶ F1USS2
- ▶ F1PAGA
- ▶ F1PAGB
- ▶ F1PAGC
- ▶ F1PRD1
- ▶ F1PRD2
- ▶ F1PRD3
- ▶ F1BBN1
- ▶ F1CFG1

These volumes provide z/OS and the common products, such as compilers.<sup>2</sup> We also recommend that you install the single-pack z/OS system (volume SARES1). This system can prove invaluable if you make a mistake and end up with a system that does not IPL.

---

<sup>2</sup> The F1PRDn volumes contain various products and you might not need all of them; see the ADCD documentation for more information about the contents of each volume.

Also, if you expect to need to install service on z/OS, you should restore the two DLIB volumes: F1DIS1 and F1DIS2. We also installed the volumes that are used by CICS V5.2 (F1C521) and DB2 V11 (F1DBB1 and F1DBB2); however, they are not strictly necessary unless you want to use the associated subsystems.

For more information about all of the volumes that are provided by the ADCD deliverable, see the ADCD documentation that is available at this website:

<http://dtsc.dfw.ibm.com/MVSDS/'HTTPD2.ADCD.GLOBAL.SHTML%28READM21F%29'>

## 4.3.2 Installing z/VM

If your target environment is to run your sysplex under z/VM (it can be a base or a Parallel Sysplex), you should install the ADCD z/VM package now.

For our Parallel Sysplex, we used the ADCD z/VM 6.3 release<sup>3</sup>. We downloaded the following volumes:

- ▶ M01RES
- ▶ 630RL1
- ▶ 630RL2
- ▶ M01W01
- ▶ M01S01
- ▶ M01P01
- ▶ VMCOM1
- ▶ VMCOM2
- ▶ M01W02
- ▶ M01W03

The VMPROD volume (which is part of the z/VM 6.3 ADCD package) is not strictly needed for Parallel Sysplex operation.

Ensure that you update your devmap file with the statements for the z/VM volumes. You can find a sample devmap file on the download site where you obtain the ADCD z/VM volumes. However, to save you time, we used the following statements (for simplicity, we recommend that you also use these device numbers):

```
device 0200 3390 3990 M01RES
device 0201 3390 3990 630RL1
device 0202 3390 3990 630RL2
device 0203 3390 3990 M01W01
device 0204 3390 3990 M01S01
device 0205 3390 3990 M01P01
device 0206 3390 3990 VMCOM1
device 0207 3390 3990 VMCOM2
device 0208 3390 3990 M01W02
device 0209 3390 3990 M01W03
device 020A 3390 3990 VMPROD
```

For more information about the use of and maintaining the z/VM system, see the z/VM product documentation that is available at this website:

<http://www.vm.ibm.com/library/>

---

<sup>3</sup> If you are using zPDT release 6, you must use z/VM 6.3, or z/VM 6.2 with the APARs that are required to support IBM z13™. These APARs are required even though you are not running on a real z13.

For information about getting z/VM up and running under zPDT, see the chapter that is titled “Other System z Operating Systems” in *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

To give you some advance help, we provide the following zPDT commands that you must issue to load z/VM under zPDT (you can start VM now to get some experience with it if you want, but you must shut it down again to complete some of the later steps in this process):

- ▶ `LOADPARM 0700` is the command defines 3270 device 700 as the VM IPL console.
- ▶ `IPL 0200` is the device number of the M01RES volume. The VMCOM1 volume should be defined on device number 0206 in your devmap file).
- ▶ On the x3270 session that corresponds to device 700, press PF10 to proceed with loading z/VM.

You do not need to be a z/VM expert to use it to host your sysplex. However, you must be familiar with the following tasks:

- ▶ Loading and shutting down z/VM under zPDT
- ▶ Logging on and off virtual machines
- ▶ Autologging the CF virtual machines
- ▶ Updating the VM directory and using the `DIRECTXA` command to activate the updated directory
- ▶ DIALing to a virtual 3270 device so that you can log on to z/OS applications, such as TSO or CICS

If you are not familiar with z/VM, we recommend installing z/VM and running your ADCD z/OS system in monoplex mode under z/VM. This approach gives you an opportunity to get used the use of z/VM without the added complexity of running multiple z/OS guests. This experience is valuable later when you are ready to move to a Parallel Sysplex environment.<sup>4</sup>

### 4.3.3 Download and create volumes

Now that z/VM is installed and you are familiar with the mechanics of running z/OS under z/VM, the next step is to download the volumes that are provided by the 2016 Sysplex Extensions. There are also new volumes that you create and then populate with system data sets.

The zPDT 2016 Sysplex Extensions uses the following volumes:

<b>CF0001</b>	This volume is part of the zPDT 2016 Sysplex Extensions package. It contains CDSs, sample JCL, and sample definitions.
<b>CICS01</b>	This volume is also part of the zPDT 2016 Sysplex Extensions package. It contains the user catalog and data sets for the CICSplex regions.
<b>CICS02</b>	This volume is the second volume for the CICSplex data sets and also is included in the 2016 Sysplex Extensions package.

<sup>4</sup> All of the z/OS parts of the 2016 Sysplex Extensions installation can be done in a z/OS that is running natively under zPDT rather than under z/VM. However, that configuration eliminates the opportunity to gain experience with z/VM before starting your sysplex. It also means having to go back through all of the steps in this chapter and selecting the z/VM-related steps. However, if your target environment is a base sysplex spanning multiple PCs, it is not necessary to use z/VM.

	<b>DB2001</b>	This volume is part of the zPDT 2016 Sysplex Extensions package. It contains the data sets for the DB2 data sharing subsystems.
	<b>DB2002</b>	This volume is the second volume for the DB2 data sets and also is included in the 2016 Sysplex Extensions package.
	<b>F1PAGX</b>	This volume starts as an empty volume so it is <i>not</i> downloaded as part of the package. This volume contains page data sets for the second z/OS system (S0W2).
	<b>F1PAGY</b>	This volume is an empty volume that will contain page data sets for the second z/OS system.
	<b>F1PAGZ</b>	This volume is an empty volume that will contain page data sets for the second z/OS system.
	<b>F1SYS2</b>	This empty volume contains system and work data sets that are used by the second z/OS system.
	<b>WORK01</b>	This volume also starts as an empty volume. It is used to contain work data sets only and is mounted as a STORAGE volume in the VATLSTPS member.

CF0001 is defined as a 3390-1, and the CICS and DB2 volumes are 3390-3s. Because you are allocating the other volumes, you control their size. We recommend 3390-9s for the paging volumes. The F1SYS2 and WORK01 volumes should be at least 3390-3s.

The CF0001, CICS01, CICS02, DB2001, and DB2002 volumes should be downloaded from the following website:

<ftp://www.redbooks.ibm.com/redbooks/SG248315/>

There are five .gz files, one for each of the first five volumes. Download the following files into the directory that contains your ADCD z/OS files:

- ▶ cf0001.gz
- ▶ cics01.gz
- ▶ cics02.gz
- ▶ db2001.gz
- ▶ db2002.gz

After the downloads complete, open a window, browse to the directory that contains your z/OS volumes, and use the **gunzip** command (for example, **gunzip -c cf0001.gz > CF0001**) to extract each of the .gz files.

The other volumes contain only data sets that are defined during the installation process. Therefore, you only need to create the empty volumes at this stage in the process. Some of the data sets on these volumes must be cataloged in your master catalog. Therefore, we feel that providing you with instructions and jobs to create them is easier than providing the volumes pre-populated and then having to catalog again the VSAM data sets in your master catalog.

Use the following commands to create Linux files now that will contain the z/OS volumes later (this process can take some time, depending on the speed of your PC's hard disk drive):

- ▶ alckd F1PAGX -d3390-9
- ▶ alckd F1PAGY -d3390-9
- ▶ alckd F1PAGZ -d3390-9
- ▶ alckd F1SYS2 -d3390-3
- ▶ alckd WORK01 -d3390-3

You now have the five volumes that are provided by this package, and five volumes that you soon initialize by using ICKDSF. But before we can perform that initialization, we update the devmap file to add the definitions for the extra volumes.

### 4.3.4 Establish a known restart point

When you have a working z/OS system running under z/VM and before you start making any changes to z/VM or z/OS to add sysplex support, it is recommended that you create a clean restart point that you can fall back to if your changes result in a system that does not initialize successfully.

One way to create this restart point is to ensure that the zPDT environment (z/OS and z/VM) is stopped and then take a copy of the z/VM and z/OS Linux files. If there are problems, you can then restore those volumes (or starting zPDT by using a devmap file that references those copies).

An alternative is to use the disk versioning support that is provided by zPDT. This support is an attractive option if you expect to be performing repetitive testing and want to repeatedly return to a known restart point. For more information about this capability, see the chapter “CKD versioning” in *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

To help you get started, we included a basic Linux script to enable versioning support for our z/OS and z/VM disks. The script is shown in “zPDT Disk versioning sample script” on page 126. You can customize this script to include more volumes (your own z/OS volumes, for example), or create a corresponding script to accept the outstanding changes and create a restart point, or to discard the changes and return to your known restart point. The z/OS and z/VM systems must be shut down at the time these scripts are run.

If you do not want to back up all of your volumes, we recommend that you create a backup of the following data sets at a minimum:

- ▶ AD CD.Z21F.PARMLIB (or the corresponding data set for your level of AD CD)
- ▶ AD CD.Z21F.PROCLIB (or the corresponding data set for your level of AD CD)
- ▶ USER.PARMLIB
- ▶ USER.PROCLIB
- ▶ USER.TCPPARMS
- ▶ USER.VTAMLST

**Note:** Your z/VM and z/OS system likely is down at this point; therefore, make a note to run this job after you bring your z/OS system back up.

Apart from providing an ability to back out any changes, having a “before” copy of the data sets that you are changing helps you to more easily visualize the differences between the system that you started with (the AD CD monoplex) and the sysplex with which you end up.

Finally, backups are like umbrellas in that you likely need a backup only if you do not have one. Therefore, taking a few minutes to create them at this point is a good investment of your time. To make it even easier, you can use the JCL that is shown in Example 4-1.

*Example 4-1 Sample JCL to back up key system data sets*

```
//BACKUP JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//BACKUP EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
```

```

//SYSUT3 DD UNIT=SYSDA,SPACE=(3120,(20,10))
//SYSUT4 DD UNIT=SYSDA,SPACE=(3120,(20,10))
//*
//IN1 DD DISP=SHR,DSN=ADCD.Z21F.PARMLIB
//OT1 DD DISP=(NEW,CATLG),DSN=SYSPLEX.Z21F.PARMLIB.BKUP,
// UNIT=SYSDA,VOL=SER=F1SYS1,
// LIKE=ADCD.Z21F.PARMLIB
//IN2 DD DISP=SHR,DSN=ADCD.Z21F.PROCLIB
//OT2 DD DISP=(NEW,CATLG),DSN=SYSPLEX.Z21F.PROCLIB.BKUP,
// UNIT=SYSDA,VOL=SER=F1SYS1,
// LIKE=ADCD.Z21F.PROCLIB
//IN3 DD DISP=SHR,DSN=USER.PARMLIB
//OT3 DD DISP=(NEW,CATLG),DSN=SYSPLEX.USER.PARMLIB.BKUP,
// UNIT=SYSDA,VOL=SER=F1SYS1,
// LIKE=USER.PARMLIB
//IN4 DD DISP=SHR,DSN=USER.PROCLIB
//OT4 DD DISP=(NEW,CATLG),DSN=SYSPLEX.USER.PROCLIB.BKUP,
// UNIT=SYSDA,VOL=SER=F1SYS1,
// LIKE=USER.PROCLIB
//IN5 DD DISP=SHR,DSN=USER.TCPPARMS
//OT5 DD DISP=(NEW,CATLG),DSN=SYSPLEX.USER.TCPPARMS.BKUP,
// UNIT=SYSDA,VOL=SER=F1SYS1,
// LIKE=USER.TCPPARMS
//IN6 DD DISP=SHR,DSN=USER.VTAMLST
//OT6 DD DISP=(NEW,CATLG),DSN=SYSPLEX.USER.VTAMLST.BKUP,
// UNIT=SYSDA,VOL=SER=F1SYS1,
// LIKE=USER.VTAMLST
//*
//SYSIN DD *
COPY INDD=IN1,OUTDD=OT1
COPY INDD=IN2,OUTDD=OT2
COPY INDD=IN3,OUTDD=OT3
COPY INDD=IN4,OUTDD=OT4
COPY INDD=IN5,OUTDD=OT5
COPY INDD=IN6,OUTDD=OT6
/*

```

---

Ensure that the job ended with return code 0 before you start making changes to these data sets.

## Starting over

If you find yourself in a position where you want to go back to the beginning and start the installation of the 2016 Sysplex Extensions all over again, you should delete (or rename) *all* of the following volumes and run the **gunzip** command against the downloaded data sets again:

- ▶ CF0001
- ▶ CICS01
- ▶ CICS02
- ▶ DB2001
- ▶ DB2002

Additionally, you should delete *all* of the following volumes and create versions by using the **alckkd** command:

- ▶ F1PAGX

- ▶ F1PAGY
- ▶ F1PAGZ
- ▶ F1SYS2
- ▶ WORK01

### 4.3.5 Updating devmap to add new volumes

Before any volume can be used by a system that is running under zPDT, the device and the associated Linux file must be defined in the devmap file. This requirement means that you must update your devmap file by adding the new volumes to your definitions.

z/VM automatically generates a device for any device that you define in the devmap file. However, on the z/OS side, you must make sure that the device numbers that you use are defined in the IODF (for more information about the devices and associated device numbers that are defined in the ADCD-provided IODF, see Table A-1 on page 122).

You can use the following devmap statements as examples to help you add the new volumes to your devmap (all of these addresses are defined as 3390 disks in the ADCD z/OS IODF file):

```
device 0ab0 3390 3990 CF0001
device 0ab1 3390 3990 CICS01
device 0ab2 3390 3990 CICS02
device 0ab3 3390 3990 DB2001
device 0ab4 3390 3990 DB2002
device 0ab5 3390 3990 F1PAGX
device 0ab6 3390 3990 F1PAGY
device 0ab7 3390 3990 F1PAGZ
device 0ab8 3390 3990 F1SYS2
device 0ab9 3390 3990 WORK01
```

The last value in each line is the Linux file name for the respective volume. If your devmap file includes a directory list in the file name (for example, 'frank/z/F1RES1'), you must adjust the device statements accordingly.

When you are updating the devmap, note the device number that you assign to the new volumes (you can enter the information in Table 4-3). The devmap points only to a Linux file and there is not necessarily any correlation between the file name and the contents of the file. However, we *strongly* recommend that the name of the Linux file matches the volser of the volume that is in that file. For example, Linux file F1SYS2 should contain the z/OS volume that is called F1SYS2. Failing to use this naming convention can lead to confusion and much wasted time later on.

Table 4-3 zPDT 2016 Sysplex Extensions volumes

Volser	Linux file name	Device number	Initialized?
CF0001			N/A
CICS01			N/A
CICS02			N/A
DB2001			N/A
DB2002			N/A
F1PAGX			
F1PAGY			



Volser	Linux file name	Device number	Initialized?
F1PAGZ			
F1SYS2			
WORK01			

Now is a good time to add some commands to the devmap file to automatically start some x3270 sessions for you. When you are running z/OS under z/VM, you often find that you need the following x3270 sessions:

- ▶ One session for the z/VM console (OPERATOR normally is logged on that session)
- ▶ One session for each z/OS system console
- ▶ Two sessions with which you can log on to TSO without requiring TCP access
- ▶ Two sessions for logging on to various z/VM virtual machines

To support these sessions, consider adding the following statements after the processors statement in the [system] section of your devmap file now:

```
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
```

You also can update the devmap to add the definitions of the OSA devices to connect your systems to the network. For more information about using the statements, see 4.3.16, “Network definitions” on page 66. For now, add the following definitions:

```
[manager]
name awsosa 0023 --path=A0 --pathtype=OSD --tunnel_intf=y
device 400 osa osa
device 401 osa osa
device 402 osa osa
device 408 osa osa
device 409 osa osa
device 40A osa osa
```

```
[manager]
name awsosa 0024 --path=f0 --pathtype=OSD
device 404 osa osa
device 405 osa osa
device 406 osa osa
device 40C osa osa
device 40D osa osa
device 40E osa osa
```

**Note:** The device numbers and the distinction between which devices are used for a tunnel to the underlying Linux system are important. The values that are used here match the values in the default ADCD-provided VM directory. Those values in turn match the values in the IBM VTAM® and TCP parms. Unless you are experienced with VTAM and TCP, we suggest that you use these values exactly as they are provided here.

The exception is the value on the path= parameter, which must be tailored to match the value from your PC. For more information, see 4.3.16, “Network definitions” on page 66.

If your zPDT system is running, you should shut it down after you apply the changes to the devmap file. Then, run the **awsstop** command, followed by the **awsstart** command to load the new devmap information. Devices that are not defined in the devmap file cannot be added dynamically; therefore, you must stop and restart zPDT to make them known to zPDT.

When the new devmap loads successfully by using the **awsstart** command, the next step is to IPL z/VM. The VM IPL address is 0200 and the console address (which is specified on the loadparm) should be 0700. The console is written to quickly, but you might need to minimize some of the other x3270 windows to see the VM console.

When the system finishes the IPL process, verify that the device numbers that you assigned to your new volumes are known to the system (in z/VM, you can run the **Q nnnn** command where nnnn is the device number that you used in the devmap file) command from the OPERATOR ID. The volumes that were delivered by the zPDT 2016 Sysplex Extensions should be online. The output from the **Q** command displays something that is similar to DASD 0AB0 CF0001. The devices that contain the empty volumes (the last five volumes that are listed in Table 4-3 on page 46) are shown as 'FREE' until you initialize them.

### 4.3.6 Configuring z/VM guests

The Parallel Sysplex requires four virtual machines (also known as *guests*): two Coupling Facility Virtual Machines (called CFCC1 and CFCC2) and two z/OS virtual machines, called S0W1 and S0W2. If you want to run the ADCD system as a monoplex under VM, we recommend that you use a virtual machine called BASEAD. The z/VM 6.3 ADCD package includes the definitions for all these virtual machines in the supplied VM directory, which is stored in the file that is called USER DIRECT C on the MAINT user ID. The initial password for these user IDs matches the user IDs.

**Important:** The BASEAD guest should *not* be used at the same time as the S0W1 or S0W2 guests.

To make it easier to share the disks between multiple guests and to add guests if you need a sysplex with more than two members, all of the z/OS disks are defined as belonging to a virtual machine called MVSDUMMY. The S0W1, S0W2, and BASEAD user IDs are all defined to link to the MVSDUMMY-owned disks. (The VM directory statements that we used for these four IDs are described in “Sample z/VM Directory entries” on page 114.)

The sample definitions contain more DASD (MDISK and LINK) statements than you need for the basic ADCD systems. This configuration is used to make it easier to add volumes to your systems in the future.

You might need to make only one small change to the ADCD VM Directory. The WRKALLEG option after an MDISK statement controls how z/VM segments channel programs. If you specify WRKALLEG for a device, z/VM does not segment a channel program, which ensures that any working allegiance is not severed by z/VM ending the channel program “early.” (Anything that does atomic I/Os needs working allegiance.) In general, it should be defaulted to OFF because OFF allows z/VM to better manage channel programs from guests. However, the WRKALLEG option *must* be present for any volumes that contain CDSs.

Because we do not know which address you use for the CF0001 volume (the volume that contains the CDSs for the base and Parallel sysplexes), you must edit the VM directory and add the WRKALLEG statement at the appropriate location. For example, if you use the same addresses as our devmap file that was described in “Sample devmap file” on page 112, you insert a DASDOPT WRKALLEG statement after the MDISK AB0 statement.

After you make any required changes to the directory entries, run the **DIRECTXA USER DIRECT C** command. Running this command updates the VM directory from your source file. You must perform this update before you log on to the z/OS virtual machines.

### 4.3.7 Initializing new volumes

The next step is to use ICKDSF to initialize the new empty volumes by using your ADCD z/OS monoplex system.

Log on to the BASEAD user ID on z/VM. When you do, you see many 'DASD 0Axx offline' messages. Those messages relate to the spare devices that we referred to in 4.3.6, "Configuring z/VM guests" on page 48, so they can be ignored.

Now IPL z/OS by running the following commands:

```
TERM CONMODE 3270
IPL A80 LOADP 0A8200M1
```

Your system initially loads in monoplex mode, and uses the x3270 session in which you entered the commands as the z/OS console.

When the system comes up, issue **D U,,,nnnn,1** commands to verify that the new devices (the devices that you must initialize) are offline (they should be offline because they are not yet initialized).

The next step is to logon to TSO by using IBMUSER or ADCDMST. To get to the z/OS TSO logon window, enter DIAL BASEAD in the COMMAND area of any of the available VM screens.

Your ADCD.LIB.JCL data set should contain a member that is called DSFINIT. Use the information that you entered in Table 4-3 on page 46 to help customize the job to be sure that you are using the correct device number for each volume. Ensure that you initialize only the volumes that do *not* have "N/A" in the Initialized column. Update the PGM=ICKDSF statement to add REGION=OM<sup>5</sup> and tailor the SYSIN input as shown in the following example:

```
INIT UNIT(0AB5) NVFY VOLID(F1PAGX) VTOC(0,1,29) INDEX(2,0,15)
```

Perform this process for each of the five new volumes (F1PAGX, F1PAGY, F1PAGZ, F1SYS2, and WORK01). You must reply U to the ICK003D message on the z/OS console for each volume. After the initialization jobs complete (check to ensure that they ended with return code 0), run the **V nnnn,ONLINE** command to bring those devices online.

### 4.3.8 Importing user catalogs

Most of the data sets that are provided by the zPDT 2016 Sysplex Extensions are cataloged in user catalogs on the 2016 Sysplex Extensions-provided volumes. To make those user catalogs and the data sets they reference available to your system, you must run an IDCAMS IMPORT CONNECT to make them known to your master catalog.

**Note:** The 2016 Sysplex Extensions assumes that you use the same master catalog for the Parallel Sysplex, base sysplex, and your monoplex configurations. Having separate master catalogs complicates matters and provides no extra benefit.

Job IMPCONN in data set SYSPLEX.PARALLEL.CNTL on the CF0001 volume performs the IMPORT CONNECTs and defines the related aliases for you.

<sup>5</sup> If you do not adjust the region size, the job might end with a return code 12 and message ICK313271.

Run the IMPCONN job now and verify that it ends with return code 0.

### 4.3.9 Creating system-specific system data sets

There are several system data sets that should not be shared between systems. In this step, we allocate those data sets for the S0W2 system. We also make some changes to the S0W1 data sets to make them more amenable to a sysplex environment. Figure 4-1 shows the contents of the F1PAGA/B/C/X/Y/Z and the F1SYS1 and F1SYS2 volumes at the start of this step.



Figure 4-1 System volumes and system data sets

Figure 4-2 on page 51 shows the contents of those volumes *after* you complete all of the tasks in this step. You see that there are data sets for the S0W1 and S0W2 systems when running in sysplex mode, and data sets with the original names that can be used when running in monoplex mode.

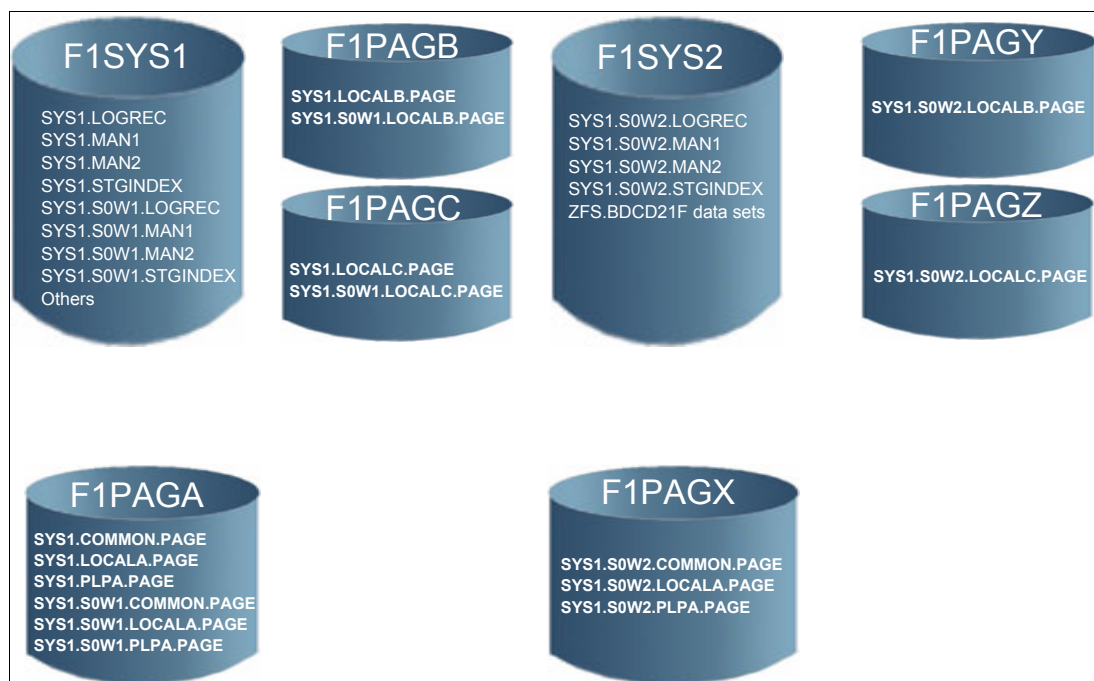


Figure 4-2 System volumes and system data sets

Data set `SYSplex.PARALLEL.CNTL` (you can access it by using the normal catalog search order following the `IMPCONN` job) contains a job that is called `VSAMS0W2` to define the Paging, VIO, SMF, and LOGREC data sets for system `S0W2`. The page data sets are allocated on the `F1PAGX/Y/Z` volumes. The other data sets are allocated on the `F1SYS2` volume. All of these data sets are cataloged in the master catalog.

**Note:** The `CF0001` volume contains two data sets that are used when you install the 2016 Sysplex Extensions with the December 2015 z/OS 2.2 AD CD system: `SYSplex.PARALLEL.CNTL.ZOS22` and `SYSplex.PARMLIB.ZOS22`. Both of these data sets need to be ignored if you are installing the 2016 Sysplex Extensions with z/OS 2.1.

Submit the `VSAMS0W2` job now. This job formats all of the page data sets. It might appear that nothing is occurring, but be patient. On our system, it took nearly 10 minutes and then ended with a return code 0.

### Reallocating page data sets for S0W1 system

The `S0W1` system (when it is part of the sysplex) *could* use the paging, SMF, VIO, and LOGREC data sets of the base (non-sysplex) z/OS system. However, to demonstrate the use of system symbols in system data set names, we provide a set of jobs that allocate the equivalent data sets for system `S0W1`.

Because the page data sets on your AD CD system take up the entire volume, the following process is used to allocate the new page data sets (this process seems long-winded, but the results are worth the effort):

**Tip:** This step and the following step (cloning the OMVS data sets) are I/O intensive. On a PC with a single hard disk drive, running multiple of these jobs in parallel elongates the overall elapsed time. Therefore, it is suggested that these steps are completed in sequence and wait for each step to complete before starting the next step.

1. Run the **D ASM** command. The response tells you how many local page data sets are in use; most likely, there are two sets: SYS1.LOCALA.PAGE and SYS1.LOCALB.PAGE. Make a note of those data set names.
2. If SYS1.LOCALC.PAGE was *not* in the list, do an ISPF 3.4 for SYS1.LOCALC.PAGE and delete that data set by entering DEL / PGSPC on the appropriate line in the data set list.

**Note:** If you cannot delete the page data set by using this method, customize and submit the DELPGSPC job in SYSPLEX.PARALLEL.CNTL to perform the delete by using a batch job.

3. If SYS1.LOCALC.PAGE *was* in the list, run a **PAGEDEL DELETE,PAGE=SYS1.LOCALC.PAGE** command. This command moves all active pages out of that data set. When the command completes (it often takes a few minutes), do a 3.4 for SYS1.LOCALC.PAGE and delete that data set (the **PAGEDEL** command empties only the data set, it does not delete it). You now have sufficient space on the F1PAGC volume to allocate two page data sets: one with the original ADCD name and one with a name that includes the system name.
4. Run the VSAM1A job from the SYSPLEX.PARALLEL.CNTL data set. This job allocates two local page data sets on F1PAGC: SYS1.SOW1.LOCALC.PAGE (for use by SOW1 when in sysplex mode), and SYS1.LOCALC.PAGE (for use by your monoplex system). The system formats the page data sets as part of the allocation process, so you should expect the job to run for a few minutes.
5. When the VSAM1A job completes, run a **PAGEADD PAGE=SYS1.LOCALC.PAGE** command to bring that page data set into service.
6. Run a **PAGEDEL DELETE,PAGE=SYS1.LOCALB.PAGE** command to empty the local page data set on F1PAGB. When the command completes (it often takes a few minutes and you see an IEE205I message that indicates that the page data set was “deleted”), do an ISPF 3.4 for SYS1.LOCALB.PAGE and delete that data set.
7. Run the VSAM1B job. This job allocates two local page data sets on F1PAGB: SYS1.SOW1.LOCALB.PAGE, and SYS1.LOCALB.PAGE.
8. After the VSAM1B job completes with return code 0, run a **PAGEADD PAGE=SYS1.LOCALB.PAGE** command to bring that page data set back into service.
9. Run a **PAGEDEL DELETE,PAGE=SYS1.LOCALA.PAGE** command to empty the local page data set on F1PAGA. When the command completes, do a 3.4 for SYS1.LOCALA.PAGE and delete that data set.
10. Run the VSAM1C job. This job allocates new SMF, VIO, and LOGREC data sets containing the system name for SOW1 on the F1SYS1 volume, plus two local page data sets on F1PAGA: SYS1.SOW1.LOCALA.PAGE, and SYS1.LOCALA.PAGE.
11. When the VSAM1C job completes, run the **PAGEADD PAGE=SYS1.LOCALA.PAGE** command to bring that page data set back into service.

This process can take up to 20 minutes (most of which is spent waiting for the jobs to finish initializing the new page data sets); however, at the end of this process, you have a set of page data sets that can be used by the standard ADCD z/OS monoplex system. you also have a set of page data sets that can be used by the base sysplex or Parallel Sysplex systems. You also have a job (VSAM1C) and a process that you can use if you want to create system-specific data sets for another system in the future.

You can use the IEASYSxx and IEASYMxx parmlib members that are provided with this package without having to modify them. Those members can be easily extended to support any number of systems if you use consistent naming for the system-specific data sets.

A bonus is that you rarely have an opportunity to remove and add page data sets to a running system, so this process gives you a chance to test and become familiar with the process if you ever need to use it on a production system.

When running in sysplex mode, you can save LOGREC and SMF data in a log stream and LOGREC and SMF data sets that are allocated by job VSAM1C are not used. However, by creating these data sets, you give yourself the option of running in DATASET or LOGSTREAM mode, and dynamically switching back and forth between the two.

#### **4.3.10 Creating zFS file systems for system S0W2**

A subset of the zFS data sets is system-specific. The 2016 Sysplex Extensions package provides a job that is called OMVSCLON in SYSPLEX.PARALLEL.CNTL to create a copy of these data sets for system S0W2 and place them on volume F1SYS2.

The naming convention of the ZFS data sets in our sample system is not ideal for a sysplex environment. Ideally, we would change the ADCD21F and BDCD21F qualifiers to reflect the names of the systems that are using the data set (S0W1 and S0W2). However, this process involved too many other changes in the base ADCD system. This is a change that might be made in a future release.

Our ADCD 2.1 system had the sysplex root and instance file systems created. If the sysplex root did not exist, SYS1.SAMPLIB(BPXISYZR) and SYS1.SAMPLIB(BPXISYZS) provide jobs to create it.

#### **4.3.11 Creating Health Checker persistent data data set for S0W2**

The z/OS Health Checker persistent data data set (called HZSPDATA) provides the Health Checker with the ability to carry information over from one IPL to the next. This capability is powerful because the Health Checker can identify changes that were introduced by the IPL of which you might otherwise be unaware.

The ADCD system includes a data set that is called ADCD.S0W1.HZSPDATA that is used by the Health Checker started task on system S0W1. In this step, we allocate a corresponding data set for the S0W2 system. If you do not run this step, Health Checker fails with a JCL error on system S0W2.

Submit the HZSALLCP job in SYSPLEX.PARALLEL.CNTL. to allocate the ADCD.S0W2.HZSPDATA data set.

#### **4.3.12 Recataloging master catalog data sets**

A few data sets (the new 2016 Sysplex Extensions-supplied CDSs) are provided on the 2016 Sysplex Extensions-supplied CF0001 volume that must be cataloged in the master catalog. Job RECATLG in SYSPLEX.PARALLEL.CNTL contains the IDCAMS statements to add those data sets to your master catalog.

The data sets that are cataloged in the master catalog all have a high-level qualifier of PARALLEL. Check to ensure that an alias of PARALLEL is not already defined in the master catalog. Then, run the RECATLG job and ensure that it completes with return code 0.

### 4.3.13 SMS changes

There is an increasing number of cases where software products require that data sets are placed on SMS-managed volumes. For this package, some of the DB2 data sets must be SMS-managed. For simplicity, we also packaged our entire CICSplex environment in two SMS-managed volumes (CICS01 and CICS02).

As a result, multiple changes are needed for SMS. You have an SMS SCDS and ACDS (they are provided by ADCD) and you likely made changes to them. For that reason, we did not want to ship replacement SMS control data sets because their use would regress your changes. Also, there is no mechanism to merge two sets of SMS control data sets. There is also no way to update the SMS definitions from a batch job, so we have no choice but to have you apply these changes by using ISPF.

However, the process is not as bad as it seems; it should take you only a few minutes. The following changes are required:

- ▶ The second z/OS system (S0W2) and the sysplex name (ADCDPL) must be added to the SMS configuration.
- ▶ Two data classes must be added for the automatic allocation of LOGR staging and offload data sets, and one data class is added for DB2 use.
- ▶ Two storage groups must be defined: one for the DB2 user volumes and one for the CICS user volumes.
- ▶ A new storage class must be defined for the data sets on the CICS or DB2 SMS-managed volumes.
- ▶ The SMS storage class and storage group ACS routines must be updated to reflect the new storage classes and storage groups.
- ▶ The ADCD-supplied SMS control data sets must have their share options changed to 3,3 to support cross-system sharing.

#### Adding S0W2 and ADCDPL to SMS

Complete the following steps:

1. Go to the ISMF panels in ISPF (option **M.2** in recent ADCD systems).
2. Ensure that you are in Administrator mode.  
If you can see option 8 (Control Data Set) in the panel, you are in Administrator mode.  
If not, select option **0** (Profile), then option **0** (User Mode), then option **2** (Administrator). Then, exit out of ISMF (by pressing F3) and restart it.
3. Select option **8** (Control Data Set).
4. Set the CDS name to SYS1.SCDS at the top of the panel.
5. Select option **3** (Alter).
6. Page down (F8) to the second panel.
7. Specify option **1** (Add), and System Name S0W2. Then, press Enter.
8. Specify option **1** (Add), and Sys Group Name ADCDPL. Then, press Enter.
9. Exit this panel (F3).
10. Select option **4** (Validate the SCDS) and look for the message SUCCESSFUL VALIDATION in the upper right corner of the window.
11. Press F3 to exit this panel.



12. Select option **5** to activate the CDS. Enter a forward slash (/) to request activation.

You should see an ACTIVATION SCHEDULED message and the IBM MVS™ console should have a NEW CONFIGURATION ACTIVATED message.

13. Press F3 to exit from ISMF.

## Creating data classes

The System Logger uses the following types of data sets for log streams:

- ▶ Staging data sets. These sets are optional, depending on how the log stream is defined. They *must* have a CI Size of 4KB.
- ▶ Offload data sets. Every log stream has one or more offload data sets. These data sets support any CI Size that is a multiple of 4KB; however, it is highly recommended that they have a CI Size of 24 KB for optimal performance.

Because of the different CI Sizes, two data classes are necessary for System Logger use: one for the staging data sets (named LOGR4K) and one for the offload data sets (named LOGR24K).

To define the LOGR4K data class, complete the following steps:

1. Go to the ISMF panels in ISPF.
2. Select option **4** (Data Class).
3. Set CDS name to SYS1.SCDS at the top of the panel.
4. Enter LOGR4K as the Data Class Name.
5. Select option **3** (Define) and press Enter.
6. Press F8 to scroll through several panels. Make *only* the following changes:
  - Description = Data class for Logger staging data sets (Page 1)
  - RECOrg = LS (Page 4)
  - CIsze Data = 4096 (Page 4)
  - Shareoptions Xregion = 3 (Page 6)
  - Xsystem = 3 (Page 6)
7. Press F3 to return to the main ISMF menu.
8. Repeat these steps for the LOGR24K data class, remembering to specify a CIsze of 24576 instead of 4096.

Also, DB2 now requires that some of its data sets are on SMS-managed volumes. For more information about this requirement, see this website:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-1302smsenvironment/>

To define the data class for DB2, complete the following steps:

1. Enter DPDGDC as the Data Class Name.
2. Select option **3** (Define).
3. Press F8 to scroll through a number of panels. Make *only* the following changes:
  - Description = Data class for DB2 (Page 1)
  - Data Set Name Type = EXT (Page 2)
  - If Ext = R (Page 2)
  - Extended Addressability = Y (Page 2)
  - Record Access Bias = U (Page 2)
4. Press Enter to save your changes and then, press F3 to return to the main ISMF menu.

5. Exit to the ISMF Primary Option Menu.
6. Select option **8** (Control Data Set) and option **5** (Activate the CDS) to activate the modified CDS.

### **Defining new storage groups**

Complete the following steps to define the two new storage groups:

1. Go to the ISMF panels in ISPF.
2. Select option **6** (Storage Group).
3. Set CDS name to SYS1.SCDS at the top of the panel.
4. Enter CICSFILES as the Storage Group Name.
5. Enter P00L as the Storage Group Type
6. Select option **3** (Define) and press Enter.
7. Change Auto Migrate and Auto Backup to N and press Enter.
8. Press F3 to save your changes.
9. Select option **5** (Volume) to define your CICS volumes.
10. Enter 2 (Define). Then, tab down to the section where you specify the volsers of the volumes that are in this storage group.
11. Enter CICS0 in the Prefix column, 1 in the From column, and 2 in the To column
12. Press Enter.
13. Press F3 to save your changes. Then, press F3 again to return to the first Storage Group panel.
14. Repeat these steps for the DB2FILES storage group, using volsers of DB2001 and DB2002.
15. Exit to the ISMF Primary Option menu.
16. Select option **8** (Control Data Set) and option **5** (Activate the CDS) to activate the modified CDS.

### **Defining new storage classes**

To be on a storage group volume, a data set must be SMS-managed. To be SMS-managed, the data set must have a storage class. Complete the following steps to define a simple storage class that is used for all data sets in the CICS or DB2 storage groups:

1. Go to the ISMF panels in ISPF.
2. Select option **5** (Storage Class).
3. Set CDS name to SYS1.SCDS at the top of the panel.
4. Enter PSADDON as the Storage Class Name.
5. Select option **3** (Define) and press Enter.
6. You do not need to change any of the settings. Press F3 to save the definition of the new storage class.
7. Press F3 to save your changes.
8. Select option **8** (Control Data Set) and option **5** (Activate the CDS) to activate the modified CDS.

## Updating ACS routines

The ADCD-provided SMS ACS routines are kept in data set SYS1.SMS.CNTL. The following sources are used for the active ACS routines:

- ▶ Data class: ACSSTORD
- ▶ Storage class: DB2STORC
- ▶ Storage group: DB2STORG

You can always find the location of the source of the currently active ACS routines by selecting option 7 (Automatic Class Selection) from the ISMF Primary Option menu, and then selecting option 5 (Display). You see a list of routines and their location, as shown in Figure 4-3.

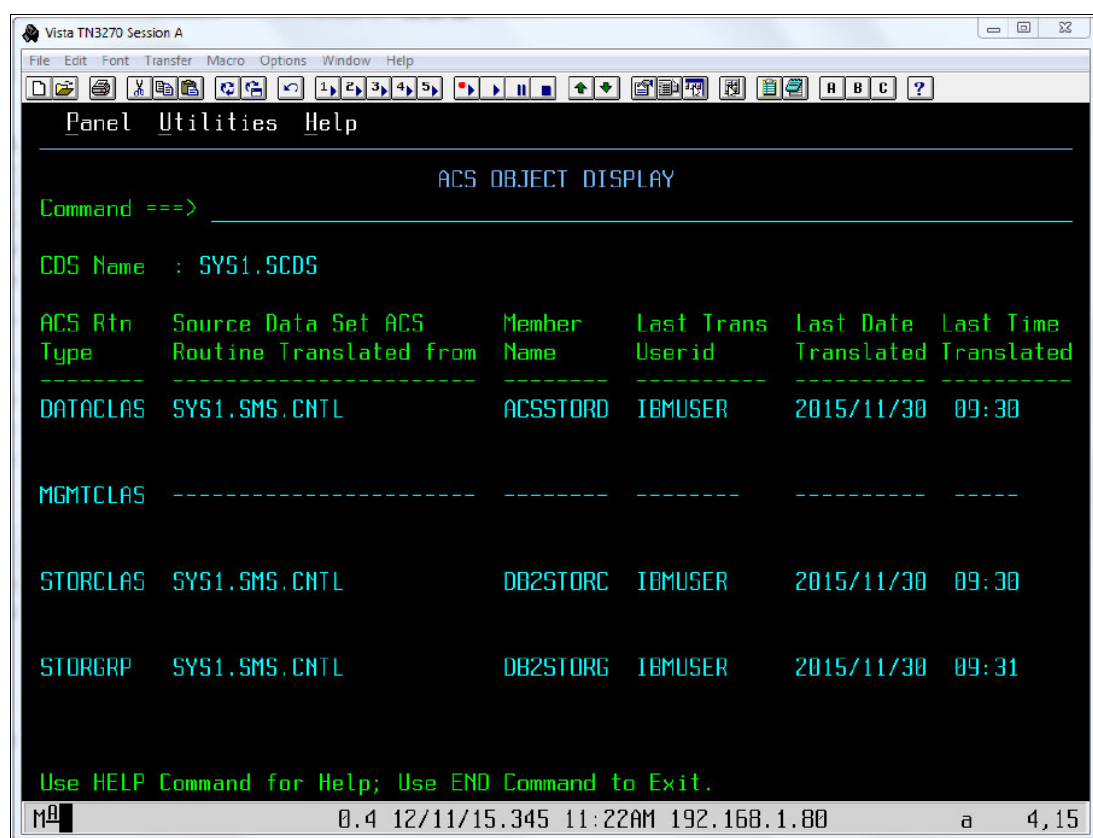


Figure 4-3 ISMF Display of ACS routines information

The data class ACS routine must be updated for the new DB2 subsystems that are provided with this package. Make the following changes:

1. Save a copy of the current data class member (ACSSTORD in this case).
2. Add the following lines near the end of the member:

```
IF &DSN(1) = 'DSNDPDG' THEN
DO
  IF &DSN(2) = 'DSNDBC' THEN
DO
  IF &DSN(3) = 'DSNDB01' THEN
DO
    SET &DATACLAS='DPDGC'
  END
  IF &DSN(3) = 'DSNDB06' THEN
```

```

        DO
            SET &DATACLAS='DPDGDC'
        END
    END
    IF &DSN(2) = 'DSNDBD' THEN
        DO
            IF &DSN(3) = 'DSNDB01' THEN
                DO
                    SET &DATACLAS='DPDGDC'
                END
            IF &DSN(3) = 'DSNDB06' THEN
                DO
                    SET &DATACLAS='DPDGDC'
                END
            END
        END
    END
END
END

```

These statements are provided in the ACSDB2DC member of SYSPLEX.PARALLEL.CNTL, so you can copy them from the member to save you from creating them manually. However, that member does *not* contain a full replacement for the data class ACS routine. Instead, it contains only the additions that are related to the 2016 Sysplex Extensions.

We found in our testing that the END statements in the provided ACSSTORD member are sometimes incorrect. We recommend reviewing the member after you add the DB2 data class statements to ensure that every DO statement has a corresponding END statement. For example, the ACSSTORD member of SYSPLEX.PARALLEL.CNTL contains the corrected member that we used for our testing.

3. Press F3 to save your changes.

The storage class ACS routine (DB2STORC) should be edited and the following changes made:

1. Save a copy of the current storage class member (DB2STORC in this case).
2. Add the following lines to the FILTLIST section at the top of the member:

```

FILTLIST DB2USER          INCLUDE('DSNDPDG')
FILTLIST DB2UCAT          INCLUDE('UCAT.DB2USER')
FILTLIST CICSUCAT         INCLUDE('UCAT.CICSUSER')
FILTLIST CICSUSER         INCLUDE('CTSLOGR', 'CTS52')

```

3. Add the following lines after the SELECT line:

```

    WHEN (&DSN = &CICSUCAT)
        DO
            SET &STORCLAS = 'PSADDON'
            EXIT
        END

    WHEN (&HLQ = &CICSUSER)
        DO
            SET &STORCLAS = 'PSADDON'
            EXIT
        END

    WHEN (&HLQ = &DB2USER)
        DO

```

```

        SET &STORCLAS = 'PSADDON'
        EXIT
    END

    WHEN (&DSN = &DB2UCAT)
        DO
            SET &STORCLAS = 'PSADDON'
            EXIT
        END
    END

```

These statements are also provided in the ACSDB2SC member of SYSPLEX.PARALLEL.CNTL. That member contains only the extra statements for the storage class ACS routine; it is *not* a complete replacement for the entire member.

4. Press F3 to save your changes.

Finally, the storage group ACS routine must be edited and the following changes made:

1. Save a copy of the current member (DB2STORG in this case).
2. Add the following lines to the FILTLIST section at the top of the member:

```

FILTLIST DB2USER          INCLUDE('DSNDPDG')
FILTLIST DB2UCAT          INCLUDE('UCAT.DB2USER')
FILTLIST CICSUCAT         INCLUDE('UCAT.CICSUSER')
FILTLIST CICSUSER         INCLUDE('CTSLOGR', 'CTS52')

```

3. Add the following lines after the SELECT line:

```

    WHEN (&DSN = &CICSUCAT)
        DO
            SET &STORGRP = 'CICFILES'
            EXIT
        END
    WHEN (&HLQ = &CICSUSER)
        DO
            SET &STORGRP = 'CICFILES'
            EXIT
        END
    WHEN (&DSN = &DB2UCAT)
        DO
            SET &STORGRP = 'DB2FILES'
            EXIT
        END
    WHEN (&HLQ = &DB2USER)
        DO
            SET &STORGRP = 'DB2FILES'
            EXIT
        END
    END

```

These statements are also provided in the ACSDB2SG member of SYSPLEX.PARALLEL.CNTL. That member contains only the extra statements for the storage group ACS routine; it is *not* a complete replacement for the entire member.

4. Press F3 to save your changes.

Having updated the ACS routines, we are now ready to translate them and then activate the new routines. Complete the following steps:

1. Go to the ISMF panels in ISPF.
2. Select option 7 (Automatic Class Selection).

3. Set CDS name to SYS1.SCDS.
4. Select option **2** (Translate) and press Enter.
5. On the ACS Source Data Set line, add SYS1.SMS.CNTL.
6. On the ACS Source Member line, enter the name of your updated data class member.
7. On the Listing Data Set line, enter a valid data set name (it is created if it does not exist).
8. Press Enter.
9. Scroll to the end of the listing to ensure that the translation process ended with return code of 0.
10. Repeat these steps for the other ACS members that you updated.
11. Press F3 to return to the primary ISMF menu.
12. Select option **8** (Control Data Set) and option **5** (Activate the CDS) to activate the modified CDS.

### Altering share options of SMS control data sets

Next, submit job ACDS5 in SYSPLEX.PARALLEL.CNTL to change the share options of several SMS data sets to allow them to be shared by the S0W1 and S0W2 systems. Sample JCL statements are shown in Example 4-2.

*Example 4-2 Sample JCL statements*

---

```
//ACDS5 JOB 1,OGDEN,MSGCLASS=X
// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  ALTER SYS1.ACDS.DATA SHAREOPTIONS(3 3)
  ALTER SYS1.SCDS.DATA SHAREOPTIONS(3 3)
  ALTER SYS1.COMMDS.DATA SHAREOPTIONS(3 3)
/*
```

---

The SMS-related changes are now complete.

## 4.3.14 JES2 MAS configuration

**Important:** The installation of the 2016 Sysplex Extensions makes only two changes that could affect your current ADCD system. The change that is described in this section is one of those changes. Therefore, you must be careful that the change we recommend does not conflict with any customization that you might performed on your JES2PARM member.

JES2 must be changed to a Multi-Access Spool (MAS) configuration with two members. This change does not affect the usability of the JES2 parameters for “normal” ADCD use.

We made the following change to member JES2PARM in the ADCD PARMLIB data set. Although we attempt to avoid making any changes to the ADCD data sets, making the changes in the USER data set resulted in several other changes in this case, which increased the complexity with little benefit in return.

Also, the change that we make is consistent with running the system in monoplex mode, which means that you can switch back and forth between monoplex and sysplex mode without making any JES-related changes.

**Tip:** Because you are changing a member that is used by your running system, ensure that you can create a backup of the member before it is changed.

The whole JES2PARM member is not shown here, only the changed lines that are related to the MAS definition are shown. (We also changed the number and classes of started initiators, but this change is optional.) There already is a MASDEF statement in the JES2PARM member, so you must scroll down to find that statement, delete it, and then insert the following statements:

```

/*                                     *-----*
*                                     * Multi-Access Spool *
*                                     *-----*
*/

MASDEF  SHARED=CHECK,
        RESTART=YES,
        CKPTLOCK=ACTION,
        DORMANCY=(25,300),
        HOLD=10,
        LOCKOUT=500

MEMBER(1) NAME=SOW1
MEMBER(2) NAME=SOW2

```

These statements are included in member JES2MASD in SYSPLEX.PARALLEL.CNTL if you want to copy them from there to the end of the JES2PARM member.

The first time you bring up JES2 after you make this change, you see message HASP870, which prompts you to confirm that you want to add member SOW2 to the JES2 MAS. Reply Y to this write to operator with reply (WTOR).

## JES2 Dynamic Proclib

While updating the JES2PARM, we implemented JES2 dynamic proclib support. This support consists of adding the JES2 proclib definitions to the JES2PARM member. The following statements are used:

```

/*****
/*                                     */
/*                                     */
/*          Dynamic JES2 proclib definitions          */
/*                                     */
/*                                     */
/*****
PROCLIB(PROC00) DD(1)=(DSN=USER.PROCLIB),
                DD(2)=(DSN=ADCD.&SYSVER..PROCLIB),
                DD(3)=(DSN=CEE.SCEEPROC),
                DD(4)=(DSN=CSQ800.SCSQPROC),
                DD(5)=(DSN=IOE.SIOEPROC),
                DD(6)=(DSN=EOY.SEOYPROC),
                DD(7)=(DSN=HLA.SASMSAM1),
                DD(8)=(DSN=CBC.SCCNPRC),
                DD(9)=(DSN=SYS1.PROCLIB)

```

These statements are included in member JES2PRCL in SYSPLEX.PARALLEL.CNTL if you want to copy them from there.

To confirm that your changes are successful, shut down and restart JES2 without removing the proclib definitions from the JES2 JCL. When you are satisfied that your change was successful (use the **\$D PROCLIB** command), you can remove the PROC00 DD statements from the JES2 PROC.

The use of this capability includes the following advantages:

- ▶ Simplified JES2 PROC JCL because all of the DD statements for your procedure libraries can be removed from the JES2 JCL.
- ▶ The proclib concatenation can be changed dynamically, with no need to stop and restart JES2.
- ▶ If you have a syntax error in your JES2 proc JCL, JES2 does not start. However, if you have a syntax error in the PROCLIB definitions in the JES2PARM member, you are presented with an HASP469 message, which gives you the option of correcting or bypassing the invalid statement.

If a proclib concatenation (PROC00, for example) is specified in the JES2 JCL and JES2 parm member, the parm definitions are used rather than those definitions in the JCL. Therefore, you can add the statements to your JES2PARM member without removing them from the JCL. For more information about dynamically changing your proclib concatenations, see the section “Using dynamic PROCLIB allocation” in *z/OS JES2 Initialization and Tuning Guide*, SA32-0991.

### 4.3.15 Creating PARMLIB members

Many of the attributes of how your system and sysplex work are controlled via parmlib members. When this deliverable was designed, we wanted to make the installation as simple and automated as possible. In particular for the parmlib member changes, we wanted to minimize your manual intervention and avoid situations in which we provide a parameter change that clashes with values that were used for your system. We also tried to optimize the use of system symbols to minimize the number of members and the administrative effort to keep multiple duplicate or near-duplicate members in sync.

Many (but not all) parmlib members support the ability to concatenate members. If you use concatenated members, the values that are obtained from the first member of the concatenation are overridden if the same parameter is encountered in a later member. For example, assume that you specify SYSPARM(00,PS) in your IEASYMxx member. That specification indicates that IEASYS00 is read first, followed by IEASYSPS. If any parameter is specified in both members, the value from IEASYSPS is used. Any parameters that are not specified in IEASYSPS use the value from IEASYS00.

This capability allows us, where possible, to provide parmlib members that contain only the parameters that must be overridden for the sysplex environment. We use a suffix of PS (for Parallel Sysplex), BS (for base sysplex under z/VM), or ST (for base sysplex spread across two PCs) our members. The members that we provided and the changes that are in each member are listed in Table 4-4 on page 63.



Table 4-4 Parmlib members supplied by zPDT 2016 Sysplex Extensions

Member	Supports concatenation?	Changed parms
CLOCKPS CLOCKST	Yes	Added SIMETRID value. Changed STPMODE to YES.
COMMNDPS	Yes	Changed VTAM start command to specify that ATCSTRPS should be used rather than ATCSTR00.  Changed VTAMAPPL started task name to VTAMPS.
CONSOLPS	No	Replaced entire member. Changes are to NAME parm on console definition and to enable OPERLOG.
COUPLEPS	No	Replaced entire member. Nearly all parameters changed.
DEVSUPPS	Yes	Added extended TIOT support for non-VSAM data sets.
GRSRNLPS	Yes	Convert all reserves to ENQs.
IEASYMPS IEASYMBS IEASYMST	Yes	Replaced entire member.  Adds IEASYSBS and IEASYSST sysparms.
IEASYSPS  IEASYSBS  IEASYSST	Yes	Updated to point at the members that require changes for the sysplex environment.  Changed GRS to TRYJOIN.  Changed CLOCK to enable STP.
IEFSSNPS	Yes	Added definitions for our CICSplex and data sharing DB2 subsystems.
LOADPS LOADBS LOADST	No	Set appropriate IEASYM parameter.
LPALSTPS	Yes	Add CICS 5.2 to LPALST.
PROGPS	Yes	Added SDSNEXIT library.
SHUTS0Wn	No	Add system-specific system shutdown commands for VTAMAPPL.
SMFPRMPS SMFPRMBS	No	Updated the SMF data set names System-specific log stream names.
VATLSTPS	Yes	Added work volume definition.
VTAMPS VTAMBS	No	Tailored VTAM00 for sysplex. Similar to VTAM00 except in how RRS is started.

To copy our parmlib members to the SYS1.IPLPARM and USER.PARMLIB data sets, submit job COPYPARM in SYSPLEX.PARALLEL.CNTL.

All of the parmlib members that are described in this section are provided as part of this package and do not require changes unless you must adjust to fit something that is specific to your configuration.

For more information about the specific changes we made to the parmlib members, see the next sections. Otherwise, see 4.3.16, “Network definitions” on page 66.

## LOADPS member

This member is copied to the SYS1.IPLPARM data set rather than USER.PARMLIB. However, we include it here because it is logically related to the members of the USER.PARMLIB data set.

We wanted to consolidate to a single IEASYMxx member that can be used by multiple systems when running in a Parallel Sysplex. We also wanted the ability to control the IEASYSxx concatenation at the system level without having to have system-specific LOADxx members. Therefore, we used the LOAD00 member as a base and changed the IEASYM parameter to point at IEASYMPS.

The LOADBS and LOADST and all of the other base sysplex-related members are described in 4.5.2, “Changes in parmlib” on page 77 and 4.6.4, “Required parmlib changes” on page 83.

**Important:** If you use a member other than LOAD00 when you load your ADCD system, you must update the supplied LOADPS member to reflect the changes that you made to LOAD00.

## IEASYMPS member

The original IEASYM00 member was designed for a single system environment. We wanted the ability to use a single IEASYMxx member for multiple systems, so we created a IEASYMPS member. Any symbols that are common to all members of the sysplex are at the top of the member in the common area. We then added filter statements that use the VM user ID that is associated with the z/OS system to set some system-specific symbols. There is one section for user ID S0W1, and one for S0W2. If you want to add systems later, you add sections that are based on these examples. For more information about our IEASYMPS member, see “Sample IEASYMxx member for sysplex” on page 123.

## IEASYSPS member

There are a few parmlib members that must be changed for the sysplex environment. Therefore, we created an IEASYSPS member that is concatenated to the standard IEASYS00 member. The IEASYSPS member contains *only* the parameters that we needed to override for the sysplex environment.

In addition to pointing at the other changed members, the IEASYSPS member performs the following tasks:

- ▶ Overrides the page data set names to point at system-specific data sets that were created in 4.3.9, “Creating system-specific system data sets” on page 50.
- ▶ Overrides the LOGREC parameter to point at the system-specific LOGREC data set.
- ▶ Overrides the VIO parameter to point at the system-specific STGINDEX data set.
- ▶ Overrides the GRS parameter to specify that GRS is used. The standard single-system ADCD environment specifies GRS=NONE because there is no other system with which it can share.

## **CLOCKPS member**

One of the requirements for systems in a sysplex is that they have a common time source. z/VM provides this capability, but the CLOCKxx member must be updated to inform z/OS of the ID of the simulated external time source. This feature required a new parameter, SIMETRID, to be added. No changes were made to the ADCD-provided CLOCKxx member.

## **COMMNDPS member**

The ADCD-provided COMMNDxx members start VTAM by using the ATCSTR00 member of USER.VTAMLST. We did not want to change that member (our objective is to make it easy to switch back to monoplex mode); therefore, we set up a new ATCSTRPS member. However, that change required a change to the **S VTAM** command to point at that member instead.

If we created a COMMNDPS member that contains only the **S VTAM** command and concatenated it to the COMMNDWS member, the **S VTAM** command is issued twice, once from the COMMNDWS member and again from the COMMNDPS member. Therefore, we copied the contents of COMMNDWS into COMMNDPS and changed only the **S VTAM** command and point at the ATCSTRPS member in VTAMLST.

## **CONSOLPS member**

One of the requirements of being in a sysplex is that every console must have a name that is unique within the sysplex. The ADCD-supplied CONSOL00 member uses a fixed name for its consoles: L700 and C908. If two systems are loaded with the CONSOL00 member, you have two consoles that are named L700 in the sysplex, which is not permitted.

Therefore, we copied the contents of CONSOL00 into CONSOLPS and changed the NAME keyword on the console definition to "&SYSNAME.L700". So, on system S0W1, the console is named S0W1L700 and on system S0W2, its console is named S0W2L700.

We also enabled OPERLOG by adding the OPERLOG keyword to the HARDCOPY DEVNUM parameter. We also added the HOLDMODE(YES) parameter to the DEFAULT statement so that you can hold the flow of messages on the console by pressing Enter. We felt that this change is useful if you are trying to debug errors during the early phases of an IPL before you can logon to TSO to review the syslog. To restart the flow of messages, press Enter again.

## **COUPLEPS member**

Because we are using unique CDSs for the sysplex environment, we needed a new COUPLExx member. We also took this opportunity to create a more robust XCF signaling infrastructure, so there are more transport classes and more signaling paths than in previous releases.

We also added definitions for the full set of CDSs (ARM, BPXMCDS, CFRM, LOGR, SFM, Sysplex, and WLM). We also included definitions for CTC connections between the systems. These connections are primarily intended for a base sysplex environment, where CF structures are not available. However, you can use the CTCs and the signaling structures in a Parallel Sysplex.

## **DEVSUPPS member**

We added a DEVSUPPS member with a single statement to enable the use of extended TIOTs for non-VSAM data sets. This addition is in line with IBM preferred practice recommendations and eliminates an exception in the z/OS Health Checker.

### **GRSRNLPS member**

Because all z/OS systems that are sharing your zPDT z/OS volumes are in the same sysplex, it is better to use GRS than RESERVE/RELEASE to protect the integrity of those volumes. To avoid problems with RESERVEs locking out volumes, we updated the GRS Resource Names List to indicate that all RESERVEs must be converted to ENQs.

### **LPALSTPS member**

Because we restored only the volume for one release of CICS (CICS TS 5.2), we created an LPALSTPS member to add only the LPA data set for that one release of CICS.

### **PROGPS member**

The load library that contains the DB2 subsystem parameters (SDSNEXIT) must be APF-authorized, so we created a PROG member with only that one entry and concatenated it to the ADCD-provided PROG members.

### **SHUTS0W1 and SHUTS0W2 members**

To make it easier to stop the system, we created system-specific versions of the SHUTALL member that is provided by ADCD. To shut down a system, run the **S SHUTSYS** command on the console of the system you want to shut down.

### **SMFPRMPS member**

Unfortunately, the SMFPRMxx member does not support concatenation. Therefore, we copied the SMFPRM00 member, updated the SMF data set names, and added the SID (system ID) parameter.

### **VATLSTPS member**

Rather than update the ADCD-provided VATLST member, we created a member that contains one entry (for the WORK01 volume) and concatenated that to the ADCD-provided VATLST00 member.

### **VTAMPS member**

This member is used by the VTAMAPPL program to start selected address spaces after the IPL completes. The VTAMPS member is based on the ADCD-provided VTAM00 member, with the exception that it uses the &SYSNAME. system symbol where appropriate to tailor commands to the system on which they run.

## **4.3.16 Network definitions**

Before we describe PROCLIB, TCP, and VTAM changes, we describe our network setup and the relationships between the definitions.

Our configuration featured a single network adapter on our PC. Linux and the two z/OS guests use it to communicate to the outside world. For simplicity, we do not have TCP set up on z/VM. Therefore, our configuration is similar to the configuration that is described in the chapter that is titled, "Multiple guests in one instance" in *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

We use VNC to log on to the Linux desktop remotely (we can use this VNC to logon to VM from a remote window if necessary). As a result, we have not found the absence of TCP on z/VM to be a problem.

The various definitions feature the following relationships:

- ▶ The devmap file contains the definition for the OSA addresses and points the control units at the respective actual adapters (one for the tunnel to Linux and one for the connection to the outside world). For more information, see the sample devmap file that is shown in Example A-1 on page 112.
- ▶ The VM directory entries for each z/OS guest contain DEDICATE statements for three OSA addresses for each connection (three each for the tunnel and real network adapter). The addresses that are used for each system must be unique. Each system must have access to two sets of three consecutive OSA devices, and the first device number must be an even number. The device numbers must match those numbers that are specified in the devmap file. For more information, see the sample directory entries for S0W1 and S0W2 that are shown in Example A-4 on page 117 and Example A-5 on page 119.
- ▶ The VTAMLST data set contains one TRLE member for each system. Each member contains two TRLE definitions: one for the tunnel to Linux and one for the real network adapter. Each definition includes a portname.
- ▶ The TCPPARMS data set contains the TCP PROFILE members (one per z/OS). The DEVICE and LINK statements use the portnames that were defined for the OSA in the TRLE definitions in VTAMLST.

**Tip:** For this configuration to work, all of the systems that share a specific network adapter *MUST* use the *same* port name.

- ▶ To keep our VTAMLST simple, we use a system symbol for the name of the member that contains the OSA definitions. The value of the symbol is set in the IEASYMxx member and depends on the system. This configuration allows us to have a single ATCCONPS member that can be used by all of the systems in the sysplex.

The OSA section of our devmap file resembles the following example:

```
[manager]
name awsosa 0023 --path=A0 --pathtype=OSD --tunnel_intf=y
device 400 osa osa
device 401 osa osa
device 402 osa osa
device 408 osa osa
device 409 osa osa
device 40A osa osa
```

```
[manager]
name awsosa 0024 --path=f2 --pathtype=OSD
device 404 osa osa
device 405 osa osa
device 406 osa osa
device 40C osa osa
device 40D osa osa
device 40E osa osa
```

To get the value of the path parameter on the name awsosa statement, run the **find\_io** command. The output from the command on our PC resembles the following example:

Path	Interface Name	Current State	MAC Address	IPv4 Address	IPv6 Address
F2	enp13s0	UP, RUNNING	00:14:5e:57:c5:6a	192.168.1.99	*
F1	enp24s0	UP, NOT-RUNNING	00:14:5e:57:c5:6c	*	* .
A0	tap0	DOWN	02:a0:a0:a0:a0:a0	*	*
A1	tap1	DOWN	02:a1:a1:a1:a1:a1	*	*
A2	tap2	DOWN	02:a2:a2:a2:a2:a2	*	*
A3	tap3	DOWN	02:a3:a3:a3:a3:a3	*	*
A4	tap4	DOWN	02:a4:a4:a4:a4:a4	*	*
A5	tap5	DOWN	02:a5:a5:a5:a5:a5	*	*
A6	tap6	DOWN	02:a6:a6:a6:a6:a6	*	*
A7	tap7	DOWN	02:a7:a7:a7:a7:a7	*	*

End of FIND\_IO

In this example, you can see that path F2 is the path that includes a status of UP, RUNNING; therefore, F2 is specified on the path= statement in the devmap.

The VM directory entry for S0W1 contains the following statements:

```
DEDICATE 0400 0400
DEDICATE 0401 0401
DEDICATE 0402 0402
DEDICATE 0404 0404
DEDICATE 0405 0405
DEDICATE 0406 0406
```

The directory entry for S0W2 contains the following statements:

```
DEDICATE 0408 0408
DEDICATE 0409 0409
DEDICATE 040A 040A
DEDICATE 040C 040C
DEDICATE 040D 040D
DEDICATE 040E 040E
```

The VTAMLST TRLE definition for S0W1 contains the following statements:

```
OSATRL1 VBUILD TYPE=TRL
OSATRL1E TRLE LNCTL=MPC,READ=(0400),WRITE=(0401),DATAPATH=(0402),      X
                PORTNAME=PORTA,                                         X
                MPCLEVEL=QDIO
OSATRL2E TRLE LNCTL=MPC,READ=(0404),WRITE=(0405),DATAPATH=(0406),      X
                PORTNAME=PORTB,                                         X
                MPCLEVEL=QDIO
```

You can see that the device numbers from the VM directory entry are used for each entry; for example, 400 (READ), 401 (WRITE), and 402 (DATAPATH). The portname for that TRLE is PORTA. In this example, devices 400 - 402 are used for the tunnel to Linux. Devices 404 - 406 are used for the real network interface.

The TCP Profile member for S0W1 contains the following statements:

```
DEVICE PORTA MPCIPA
LINK ETH1 IPAQENET PORTA
HOME 10.1.1.2 ETH1
;
DEVICE PORTB MPCIPA
LINK ETH2 IPAQENET PORTB
HOME 192.168.1.81 ETH2
```

You can see that the same portname (for example, PORTA) that was specified on the TRLE definition is used on the DEVICE and LINK statements.

The TRLE definition for S0W2 contains the following statements:

```
OSATRL2 VBUILD TYPE=TRL
OSATRL2E TRLE LNCTL=MPC,READ=(0408),WRITE=(0409),DATAPATH=(040A),      X
                PORTNAME=PORTA,                                         X
                MPCLEVEL=QDIO
OSATRL3E TRLE LNCTL=MPC,READ=(040C),WRITE=(040D),DATAPATH=(040E),      X
                PORTNAME=PORTB,                                         X
                MPCLEVEL=QDIO
```

You see that although S0W2 uses different device numbers, *the portnames are the same as the definition for S0W1*. The Profile member for S0W2 contains the following statements:

```
DEVICE PORTA MPCIPA
LINK ETH1 IPAQENET PORTA
HOME 10.1.1.3 ETH1
;
; This second device is optional
DEVICE PORTB MPCIPA
LINK ETH2 IPAQENET PORTB
HOME 192.168.1.91 ETH2
```

If you use different portnames on the two systems, the adapter does *not* activate on the second system. To our knowledge, this information is not documented elsewhere.

### 4.3.17 Creating PROCLIB members

A few changes are required to PROCLIB members for the base system. There are many new procs for the 2016 Sysplex Extensions-provided CICS regions and DB2 subsystems; however, those changes are described in Chapter 6, “Sample CICSplex” on page 103, and Chapter 5, “Sample DB2 data sharing environment” on page 95. All changes and additions are made in the USER.PROCLIB data sets.

Run the COPYPROC job in SYSPLEX.PARALLEL.CNTL now to copy the procs. For more information about the changes that we made, continue reading this section. Otherwise, you can skip to 4.3.18, “Creating TCPPARMS” on page 70.

#### VTAMAPPL started tasks

VTAMAPPL is a basic automation tool that can be used to issue system commands and insert pauses when the commands are issued. The supplied VTAMAPPL proc provides the ability to override the member name, but not the data set name. We created a VTAMPS member in USER.PROCLIB because the ADCD-supplied VTAM00 member points at ADCD.Z21F.PARMLIB and we did not want to place any of our members in that data set.

VTAMPS is started by using a command, such as - **S VTAMPS,M=&VTAMAPPL**, where &VTAMAPPL is a system symbol that indicates which VTAMAPPL member of USER.PARMLIB is used.

The SHUTSYS proc also uses VTAMAPPL to shut down started tasks in an orderly and phased manner. The SHUTSYS proc uses the &SYSNAME. system symbol to select the correct SHUTxxxx parmlib member for that system.

The VTAMPS and the SHUTSYS procs are generic and can be used to start or stop either system.

### TCP/IP procedure

The ADCD-supplied TCP/IP procedure references the ADCD TCPPARMS data set, so we created a TCP/IP proc in USER.PROCLIB to point to the USER.TCPPARMS data set. This change also affects the non-sysplex AD system. The same PROFILE and DATA members are used by system S0W1, regardless of whether it is in monoplex or sysplex mode.

**Note:** The TCPIP member that is provided (and copied into your USER.PROCLIB data set) by the 2016 Sysplex Extensions is set up to use PROFILE and TCPDATA members with names that match the system name (S0W1P and S0W1D). The ADCD-provided members use a different naming convention (PROFILE and TCPDATA) because they were not intended for a sysplex environment.

## 4.3.18 Creating TCPPARMS

**Important:** We mentioned in 4.3.14, “JES2 MAS configuration” on page 60 that there are two steps in the implementation of the 2016 Sysplex Extensions in which you must make changes that might affect your current, running ADCD system. This step is the second of those instances.

Because network implementations can vary greatly from one system to another, it is not possible to provide a set of definitions that work in every system. This step copies the members that *we* used to get our systems to work in a sysplex configuration. However, it should be treated as an example; you likely must make adjustments that are based on your specific network setup.

If there is an error in your TCP definitions, you can still log on to the system by using one of the local z/VM windows and fix the problem from there.

The COPYTCPP job in SYSPLEX.PARALLEL.CNTL copies four members to your USER.TCPPARMS data set. You likely must adjust these parameters to match *your* LAN configuration. If you change the member names, make corresponding changes to the TCP/IP JCL in USER.PROCLIB.

The following examples show the TCP/IP DATA members for the two systems. If you connect your z/OS systems to an external network, the DOMAINORIGIN values in these members must be changed. If you intend to use a name server, the NSINTERADDR values must be provided.



### Example 1: MEMBER S0W1D

The S0W1D member is used by base and Parallel Sysplex systems, as shown in the following example:

```
TCPIPJOBNAME TCPIP
S0W1: HOSTNAME S0W1
DOMAINORIGIN XXX.YYY.COM
DATASETPREFIX TCPIP
;NSINTERADDR xx.xx.xx.xx
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
ALWAYSUTO NO
```

### Example 2: MEMBER S0W2D

The S0W2D member is used by base and Parallel sysplex systems, as shown in the following example:

```
TCPIPJOBNAME TCPIP
S0W2: HOSTNAME S0W2
DOMAINORIGIN XXX.YYY.COM
DATASETPREFIX TCPIP
;NSINTERADDR xx.xx.xx.xx
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
ALWAYSUTO NO
```

## 4.3.19 Creating VTAMLST members

We added several members to USER.VTAMLST. We provide sample VTAMLST members in SYSPLEX.VTAMLST. You can use job COPYVTAM in SYSPLEX.PARALLEL.CNTL to copy the sample members to USER.VTAMLST. We updated the ATCSTRPS, ATCCONPS, and OSATRLx members. You might need to customize these members, depending on which VTAM applications you want to automatically activate at VTAM startup, and on the device numbers that you used for your virtual OSA adapters.

**Tip:** If you change the VTAMLST members, the X symbols on the right side of some lines are continuation characters and *must* be in column 72 of the statement.

The naming convention (the two-letter suffix) is not carried into the TRL names.

We also added members for CICS and DB2. Those changes are described in Chapter 6, “Sample CICSplex” on page 103, and Chapter 5, “Sample DB2 data sharing environment” on page 95.

## 4.3.20 Customization for Parallel Sysplex complete

You now completed all of the changes that are required to start the systems in Parallel Sysplex mode. If this configuration is your target, proceed to 4.4, “Bringing up your Parallel Sysplex” on page 72. If your objective is to have a base sysplex under z/VM, there are some extra steps that must be completed. For more information about those steps, see 4.5, “Other steps for base sysplex under z/VM” on page 76. If your target environment is a base sysplex spread over two PCs, see 4.6, “Implementing a base sysplex without z/VM” on page 79.

## 4.4 Bringing up your Parallel Sysplex

At this point, you made all of the changes that are required to start the Parallel Sysplex.

However, to verify that you still can fall back to monoplex mode, we recommend that you shut down and then reload the z/OS system that is running under the BASEAD user ID. It should be possible to load your system in the same way as you did previously; that is, by using the same IPL address and load parm.

Having verified that your current system still initializes successfully, the next step is to start your Parallel Sysplex. This process includes the following steps:

1. Shut down the BASEAD system and log off that virtual machine.
2. Run the following commands from the OPERATOR or MAINT user IDs to initialize the two Coupling Facility Virtual Machines:

```
XAUTOLOG CFCC1  
XAUTOLOG CFCC2
```

If you want to see the messages that are issued by the CFs, log on to CFCONSOL. The messages from both CFVMs are directed to that user ID.

Wait a couple of minutes to give the CFs a chance to initialize.

3. Log on to the S0W1 virtual machine. Check that you see messages, such as “HCPMFC2804I Message devices 1400-1403 defined and coupled to CFCC1.” for each of the two CFs. These messages confirm that the CFVMs finished initializing. If you do not see these messages, log off and wait for a few minutes and then, log on again. After you see the messages about coupling to the CFs, run the following commands:

```
TERM CONMODE 3270  
IPL A80 LOADP 0A82PSM1
```

This IPL command assumes that the sysres is on device A80 and the F1SYS1 volume is on device A82. The ‘PS’ in the load parm refers to member LOADPS in SYS1.IPLPARM.

Depending on the speed of your PC, it might take a few minutes for the NIP messages to start appearing on the console. After the messages start, you might be prompted to confirm that you want to Initialize the sysplex (reply R 00,I to that WTOR). Sometime later, you might be prompted with WTOR IGGN505A, which indicates that the CICS TS 5.1 link library is not accessible. If you see that message and you do not want CICS TS 5.1, reply CANCEL.

The system should not present any more WTORs.

You might notice some messages about failures to connect to log streams (specifically for RRS and OPERLOG). The reason for these messages is described in 3.4, “System logger” on page 29. To define the log streams and eliminate those messages, run the following jobs in data set SYSPLEX.PARALLEL.CNTL:

► LOGREREP

This job defines the log stream for LOGREC data. For information about enabling the use of this log stream, see 3.4.2, “LOGREC” on page 32.

► LOGRHC

This job defines the log stream that can be used by the z/OS Health Checker.

► LOGROPR

This job defines the OPERLOG log stream. For information about activating OPERLOG, see 3.4.3, “OPERLOG” on page 32.

► LOGRSMF

This job defines the SMF log stream. For more information about the use of SMF log streams, see 3.4.4, “System Management Facility” on page 33.

► LOGRRRS

This job defines the RRS log streams for Parallel and base sysplex modes. For more information about RRS and considerations for its log streams, see 3.4.1, “Resource Recovery Services” on page 30.

**Tip:** All of these jobs end with a return code of 12 the first time they are run, which is acceptable. The jobs are set up to delete and redefine the respective log streams. Because the log streams do not exist the first time that the job is run, the DELETE LOGSTREAM statement fails, but the other statements complete successfully.

Because there are so many log streams, checking the output from this job can be time-consuming. The quickest way to check is to run the **D LOGGER, L** command and check that all of the log streams that are shown in Example 4-3 are defined in your system.

*Example 4-3 Log streams defined by LOGRxxx jobs*

ATR.ADCDPL.ARCHIVE	RRS_ARCHIVE_1	000001 IN USE
ATR.ADCDPL.DELAYED.UR	RRS_DELAYEDUR_1	000001 IN USE
ATR.ADCDPL.MAIN.UR	RRS_MAINUR_1	000001 IN USE
ATR.ADCDPL.RESTART	RRS_RESTART_1	000001 IN USE
ATR.ADCDPL.RM.DATA	RRS_RMDATA_1	000001 IN USE
ATR.ADCDPL.RM.METADATA	RRS_RM_META_1	000001 IN USE
ATR.SOW1.ARCHIVE	*DASDONLY*	000000 AVAILABLE
ATR.SOW1.DELAYED.UR	*DASDONLY*	000000 AVAILABLE
ATR.SOW1.MAIN.UR	*DASDONLY*	000000 AVAILABLE
ATR.SOW1.RESTART	*DASDONLY*	000000 AVAILABLE
ATR.SOW1.RM.DATA	*DASDONLY*	000000 AVAILABLE
ATR.SOW1.RM.METADATA	*DASDONLY*	000000 AVAILABLE
ATR.SOW2.ARCHIVE	*DASDONLY*	000000 AVAILABLE
ATR.SOW2.DELAYED.UR	*DASDONLY*	000000 AVAILABLE
ATR.SOW2.MAIN.UR	*DASDONLY*	000000 AVAILABLE
ATR.SOW2.RESTART	*DASDONLY*	000000 AVAILABLE
ATR.SOW2.RM.DATA	*DASDONLY*	000000 AVAILABLE
ATR.SOW2.RM.METADATA	*DASDONLY*	000000 AVAILABLE
HZS.HEALTH.CHECKER.HISTORY	HZS_HEALTHCHKLOG	000000 AVAILABLE
IFASMF.GENERAL	IFASMF_GENERAL	000000 AVAILABLE
IFASMF.SOW1	*DASDONLY*	000000 AVAILABLE
IFASMF.SOW2	*DASDONLY*	000000 AVAILABLE
SYSPLEX.LOGREC.ALLRECS	SYSTEM_LOGREC	000000 AVAILABLE
SYSPLEX.OPERLOG	SYSTEM_OPERLOG	000000 AVAILABLE

If a log stream is missing, refer to the output from the associated job and search for the corresponding DEFINE LOGSTREAM NAME(log\_stream\_name) statement. An example of the part of the output from the LOGRRRS job is shown in Example 4-4 on page 74. In this case, the IXG007E message indicates that an error was encountered while processing the previous statement (#116). In this case, statement 116 was a DEFINE LOGSTREAM for ATR.ADCDPL.RESTART. The IXG007E message indicated that the DATACLASS definition in the DEFINE LOGSTREAM referred to a data class that was not defined.

*Example 4-4 Sample LOGRRRS job output*

---

```
LINE #      CONTROL CARDS
116      DEFINE LOGSTREAM NAME(ATR.ADCDPL.RESTART)
117      STRUCTNAME(RRS_RESTART_1)
118      LS_DATACLAS(LOGR24K)
119      HLQ(LOGGER)
120      LS_SIZE(10240)
121      MODEL(NO)
122      LOWOFFLOAD(20)
123      HIGHOFFLOAD(80)
...
...
IXG005I LOGR POLICY PROCESSING LINE# 116
IXG007E A STORAGE MANAGEMENT SUBSYSTEM (SMS) ATTRIBUTE CLASS IS UNDEFINED.
IXG447I LOGR POLICY PROCESSING FOUND AN ERROR BUT CONTINUES.  RETCODE=00000008
RSNCODE=00000838
IXG003I LOGR POLICY PROCESSING ENCOUNTERED AN UNEXPECTED ERROR.  DIAGNOSIS
INFORMATION: 00000004 000003F6 0107001B 00000000
```

---

You also see many log streams the names of which start with START1. These streams are CICS log streams that are pre-defined in the LOGR CDS that is included with this package. Therefore, you do not need to set them up.

The CICS log stream staging and offload data sets are allocated in the CICS storage group (CICS01 and CICS02 volumes). The OPERLOG, LOGREC, Health Checker, SMF, and RRS log stream data sets (all with a high-level qualifier of LOGGER) are allocated on the volumes that are mounted with the storage attribute (F1SYS1 and WORK01, if you use the provided VATLST00 and VATLSTPS members).

It is suggested that the system is shut down at this point (run the **S SHUTSYS** command) and then, bring it back up.

After the restart, the system should automatically start using the OPERLOG and RRS log streams.

If you want to use the LOGREC log stream, run the **SETLOGRC LOGSTREAM** command or include that command in the VTAMPS member.

If you want to use the SMF log stream, edit the SMFPRMPS member and change the first line to say **ACTIVE** rather than **NOACTIVE**, and the second line to say **RECORDING(LOGSTREAM)** rather than **RECORDING(DATASET)**. Then, activate the updated member by running the **SET SMF=PS** command.

If you want Health Checker to send its output to a log stream, run the **F HZSPROC,LOGGER=ON,LOGSTREAMNAME=HZS.HEALTH.CHECKER.HISTORY** command. You can also update the HZSPRMAD member of the ADCD.Z21F.PARMLIB data set so that the log stream is enabled automatically. For more information about the use of the HZSPRMxx member, see *IBM Health Checker for z/OS User's Guide*, SC23-6843.

After system S0W1 initializes successfully, complete the following steps to start the other sysplex member:

1. Log on to S0W2 (the initial password is S0W2).
2. Run the following commands to load the S0W2 system:

```
TERM CONMODE 3270
IPL A80 LOADP 0A82PSM1
```

The S0W2 system comes up with no further intervention required.

#### 4.4.1 Parallel sysplex implementation checklist

Because the implementation of the 2016 Sysplex Extensions Parallel Sysplex requires many steps and it is important at the steps are completed in the correct sequence, we provide the checklist in Table 4-5. You can use this checklist track your progress and ensure that you do not accidentally omit or skip any steps.

*Table 4-5 2016 Sysplex Extensions Parallel Sysplex implementation checklist*

Task	Description	Done?
1	Install base ADCD z/OS system	
2	Download/Install base ADCD z/VM system	
3	Download and gunzip five 2016 Sysplex Extensions volumes	
4	Allocate five new volumes by using the <b>a1cckd</b> command	
5	Create backup of z/VM and z/OS volumes before making changes	
6	Add 2016 Sysplex Extensions volumes to devmap	
7	Add commands to devmap to start x3270 sessions	
8	Restart zPDT and IPL z/VM	
9	Update VM Directory to add WRKALLEG keyword	
10	Log on to BASEAD and IPL z/OS in monoplex mode	
11	Initialize five new z/OS volumes	
12	Run IMPCONN job to import connect user catalogs	
13	Run VSAMS0W2 job Create S0W2 System VSAM data sets	
14	Run VSAM1A, VSAM1B, and VSAM1C jobs to reallocate S0W1 page data sets	
15	Run OMVSCLON job to create S0W2 copies of OMVS file systems	
16	Run HZSALLCP job to allocate S0W2 Health Checker data set	
17	Run RECATLG job to recatalog 2016 Sysplex Extensions CDSs in master catalog	
18	Add S0W2 system name and sysplex name to SMS config	
19	Add the following SMS data classes for Logger and DB2: LOGR4K LOGR24K DPDGDC	

Task	Description	Done?
20	Create SMS storage class for CICS and DB2 data sets: PSADDON	
21	Create SMS storage groups for CICS and DB2 data sets: CICFILES DB2FILES	
22	Update ACS routines to assign new data and storage classes and storage groups	
23	Run ACDS5 job to change shareoptions of SMS CDSs	
24	Update MASDEF statements in JES2PARM member	
25	Run COPYPARM job to copy 2016 Sysplex Extensions parmlib members to USER.PARMLIB and SYS1.IPLPARM	
26	Run COPYPROC job to copy 2016 Sysplex Extensions Proclib members to USER.PROCLIB	
27	Create new TCP PROFILE and TCPDATA members in USER.TCPPARMS	
28	Run COPYVTAM job to copy VTAMLST members to USER.VTAMLST	
29	Reload BASEAD system in monoplex mode	
30	Logoff BASEAD	
31	AUTOLOG CF Virtual Machines from CFCONSOL user	
32	Log on to S0W1 and IPL by using new Parallel Sysplex LOADPS member	
33	Run LOGREREP job to define LOGREC log stream	
34	Run LOGROPR job to define OPERLOG log stream	
35	Run LOGRRRS job to define RRS log streams	
36	Run LOGRSMF job to define SMF log stream	
37	Run LOGRHC job to define Health Checker log stream	

## 4.5 Other steps for base sysplex under z/VM

If your target environment is a base sysplex under z/VM, there are some other steps required. There are also some messages that can be ignored when you start a base sysplex under z/VM. In this section, we list those messages and describe the reasons why you are receiving them.

### 4.5.1 Changes to z/VM directory

Add definitions for virtual CTCs that are used for XCF communication between the members of the sysplex.

You can define these dynamically in each virtual machine that is connected via the CTCs (S0W1 and S0W2 in this case) by running the VM **DEFINE** command.

However, we believe that it is less work to add the definitions to the VM directory. To add these definitions, add the following statements to the directory entries for S0W1 and S0W2 (use the same definitions for both virtual machines):

```
SPECIAL E20 CTCA
SPECIAL E21 CTCA
SPECIAL E40 CTCA
SPECIAL E41 CTCA
SPECIAL E42 CTCA
SPECIAL E43 CTCA
```

After you make these changes, save the `USER DIRECT C` file and run the **DIRECTXA USER DIRECT C** command to update the directory.

You notice that these statements do not provide information about which CTC devices are connected to which other devices. For more information, see 4.5.3, “Bringing up your base sysplex” on page 78.

## 4.5.2 Changes in parmlib

From a parmlib perspective, the only thing that *must* be different in a base sysplex compared to a Parallel Sysplex is that GRS Star cannot be used in a base sysplex. The GRS parameter in IEASYSxx must specify something other than “STAR”.

This section describes the members that must be used when running in base sysplex mode under z/VM.

### LOADxx member

Because you use the same z/VM virtual machines for base and Parallel Sysplex, you need some way to differentiate between base and Parallel Sysplex. This distinction is made by using a different LOADxx member when running in base sysplex mode. The LOADBS member points at IEASYMBS.

The LOADBS member is copied into `SYS1.IPLPARM` when you run the BASPARM job.

### IEASYMBS member

The IEASYMBS member is identical to the IEASYMPS member, with the following exceptions:

- ▶ The IEASYMBS member sets the VTAMAPPL symbol to be VTAMBS, which is used by the START VTAMPS command in COMMNDPS.
- ▶ The IEASYMBS member adds one more member (IEASYSBS) to the sysparm concatenation.

The IEASYMBS member is copied into `USER.PARMLIB` when you run the BASPARM job.

### IEASYSBS member

The IEASYSBS member contains one parameter (the GRS=TRYJOIN statement). All of the other IEASYSxx parms are picked up from IEASYS00 in ADCD.Z21F.PARMLIB or IEASYS00 in USER.PARMLIB.

## SMFPRMBS

The SMFPRMBS member is identical to the SMFPRMPS member with the exception that it specifies system-unique log streams rather than a single log stream that is shared by both systems.

The SMFPRMBS member is copied into USER.PARMLIB when you run the BASPARM job.

## VTAMBS member

The VTAMBS member starts each RRS with a different group name.

Now, run job BASPARM in SYSPLEX.PARALLEL.CNTL to copy these members into USER.PARMLIB and SYS1.IPLPARM.

### 4.5.3 Bringing up your base sysplex

There are some small changes to the IPL process if you are bringing up the systems in base sysplex mode.

Because the same virtual machines are used for Parallel and base sysplex modes, the systems automatically use the CFs if they are running and connected to the S0W1 and S0W2 machines. Therefore, you must ensure that the Coupling Facility Virtual Machines are *not* logged on. To do this, use the OPERATOR or MAINT ID and run a **Q N** command. If CFCC1 or CFCC2 appear in the list, run the following commands to stop them (having first shut down any z/OS systems that might be using them):

```
FORCE CFCC1
FORCE CFCC2
```

When running in a base sysplex under z/VM, the S0W1 and S0W2 systems communicate by using the CTCs that you defined in 4.5.1, “Changes to z/VM directory” on page 76. Before they can be used, you must connect the CTCs to each other. Although this connection must be done only from one of the virtual machines, S0W1 and S0W2 must be logged on before you can issue these commands. After both machines are logged on, run the following commands in S0W1:

```
COUPLE E20 S0W2 E21
COUPLE E21 S0W2 E20
COUPLE E40 S0W2 E41
COUPLE E41 S0W2 E40
COUPLE E42 S0W2 E43
COUPLE E43 S0W2 E42
```

You are now ready to start the systems in base sysplex mode. To do start the system in this mode, run the following command in one of the systems:

```
TERM CONMODE 3270
IPL A80 LOADP 0A82BSM1 (this loads the system by using the LOADBS member of
SYS1.IPLPARM)
```

As with any time you load the first system in a sysplex, you might receive a WTOR asking if you want to initialize the sysplex. If you see that message, reply **R 00,I**. You should expect to see messages when the system is initializing that indicate it cannot access CFs CFCC1 or CFCC2. You see these messages because the base sysplex is using the same CDSs (including the CFRM data set) as the Parallel Sysplex. Because you are coming up in base sysplex mode, you can ignore those messages.



When the first system finishes initializing, logon to TSO and browse back through syslog to verify that everything started as expected. If everything is OK, submit job LOGRRRS in SYSPLEX.PARALLEL.CNTL to allocate the DASDONLY log streams that are used by RRS on each system.

The OPERLOG and LOGREC log streams include fixed names. Base sysplex supports only DASDONLY log streams, and DASDONLY log streams cannot be shared across systems. In a base sysplex, a maximum of one system can use OPERLOG or LOGREC, which negates much of the value of those two functions. As a result, we did not provide jobs to create DASDONLY versions of the OPERLOG or LOGREC log streams.

Address any unexpected messages, then load the other system by using the same IPL address and load parameters. When that system is initializing, you should see messages that indicate that XCF is connecting to the other system by using the CTC addresses that you specified in the **COUPLE** commands.

## 4.6 Implementing a base sysplex without z/VM

The other option for running a base sysplex is more complex because it involves two PCs, cross-PC file sharing, Server Time Protocol (STP), and the use of a network to connect the PCs.

Base sysplex operation does not involve z/VM, but it does involve other details that must be set up correctly.

The base sysplex configuration involves multiple Linux machines. A Linux file sharing function, such as NFS, provides coordinated DASD functions at the Linux cache level. z/OS uses GRS functions, via the CTC links among z/OS systems, to coordinate data set sharing and other serialization. A synchronized time-of-day function is provided by the zPDT STP function<sup>6</sup>.

In the configuration that is shown Figure 4-4, all of the emulated DASD for all z/OS partners are placed on one Linux machine (PC2), which becomes the Linux file server. In practice, the emulated DASD can be spread over several Linux machines, all of which can work as file sharing servers and clients. In Figure 4-4, the solid lines between the PCs and the switch represent the physical connections between the PCs. The dotted lines represent logical connections. In practice, all of the traffic (STP, XCF, NFS, and so on) goes over the Ethernet connections.

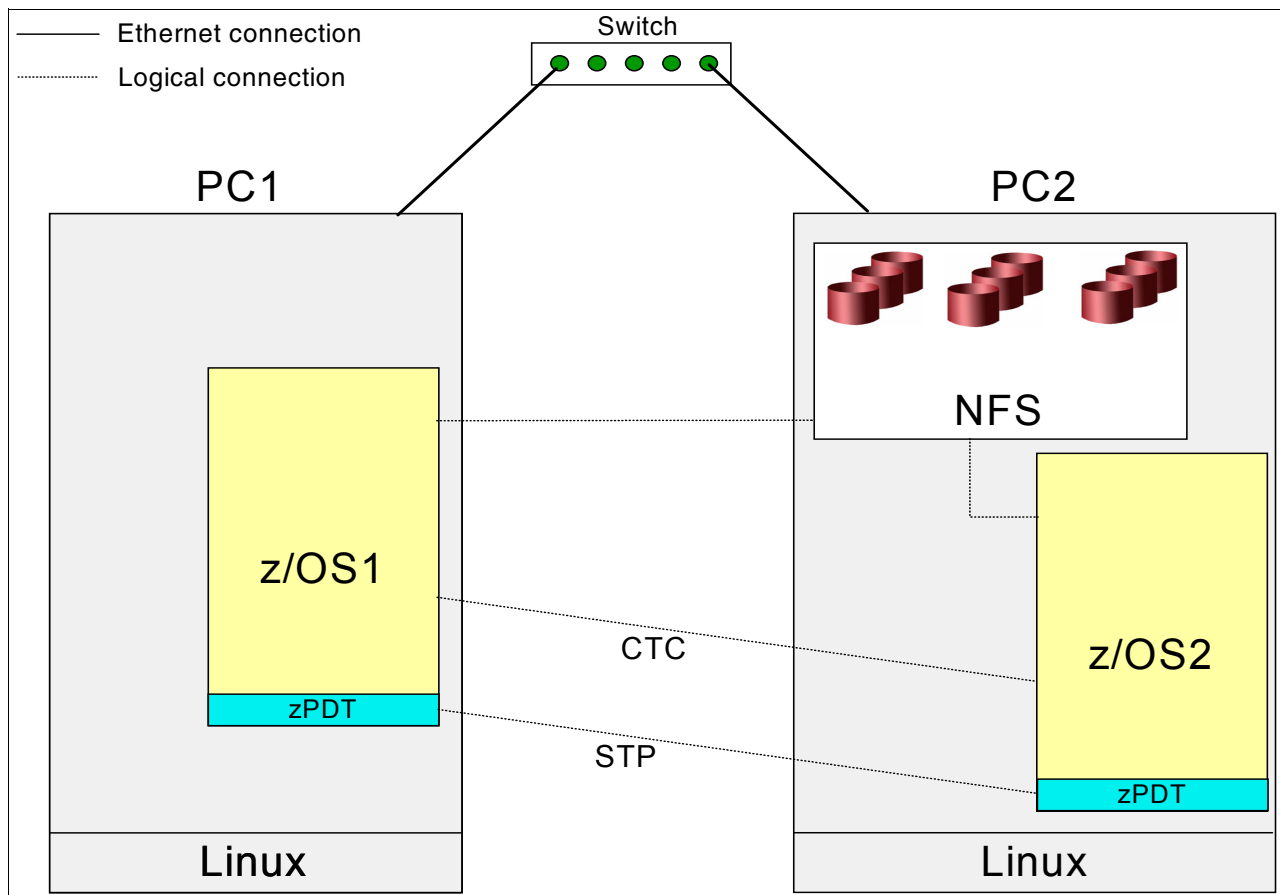


Figure 4-4 Base sysplex without z/VM

#### 4.6.1 Setting up and starting STP

One of the most basic requirements of any sysplex is that all of the systems in the sysplex must have a common time source. When running a parallel or base sysplex under z/VM, z/VM provides the common time source. However, if the sysplex is spread over more than one PC, you need some other mechanism for keeping the System z clocks in sync; namely, STP.

<sup>6</sup> The zPDT STP function requires zPDT GA6 or later.

The zPDT implementation of STP timer facilities is described in the STP chapter of *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

To use STP, you must edit and configure the `/usr/z1090/bin/CCT_data` file on each PC. This file differs slightly on each member of the base sysplex because the IP address (or domain name) differs for each member.

In addition to updating the `CCT_data` file, you must update the `devmap` file to add the `[stp]` section. The STP chapter of *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, includes clear, step-by-step, instructions for setting up these files. An example of the definitions from our `devmap` is shown in Example 4-5.

*Example 4-5 Sample stp definition statements from devmap file*

---

```
[stp]
ctn 00000000F1F0F9F0
node 1 W520 *      # For the W510 system, the asterisk is
node 2 W510      # moved to the second node statement
```

---

After configuring this file on each sysplex member, verify that all of the members of the base sysplex have TCP/IP connectivity to each other. This connection can be verified by using simple **ping** commands. Now, run the zPDT **stpserverstart** command on each member. In principle, running this command causes the STP functions to be started automatically whenever you start Linux. The **stpserverquery** command can be used to verify that STP is functioning as you expect.

## 4.6.2 Defining CTCs for XCF communication

Another requirement for a base or Parallel sysplex is that XCF in the systems in the sysplex must communicate with each other. If a system is trying to join the sysplex but cannot communicate over XCF to the other sysplex members, it is not allowed to join the sysplex.

Because a base sysplex does not have CF structures for XCF communication, it uses CTCs to communicate. In a zPDT sysplex that does not use z/VM, zPDT emulates the CTC function by using TCP/IP to communicate between the PCs. You can find all of the information that you need with sample configurations in the channel-to-channel chapter of *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

When z/VM is used to emulate the CTC function, you use the **COUPLE** command to tell which two CTC devices to connect. Similarly, with zPDT, you need to tell each zPDT the following information:

- ▶ Device number for each CTC device
- ▶ To which remote CTC it should connect

Because TCP/IP is used for the CTC communication, you use the IP address of the remote CTC by using the IP address of the remote Linux (you specify the IP address of the *Linux* system, *not* the address of the z/OS system). Sample definitions for two PCs are shown in Example 4-6 on page 82 and Example 4-7 on page 82.

#### Example 4-6 Sample devmap CTC definitions on PC1

---

```
[manager]          #CTC definitions for PC1 in base sysplex
name awsctc 0088
device E40 3088 3088 ctc://192.168.1.99:4000/E41
device E41 3088 3088 ctc://192.168.1.99:4001/E40
```

---

#### Example 4-7 Sample devmap CTC definitions on PC2

---

```
[manager]          #CTC definitions for PC2 in base sysplex
name awsctc 0088
device E40 3088 3088 ctc://192.168.1.230:4001/E41
device E41 3088 3088 ctc://192.168.1.230:4000/E40
```

---

As described in 4.5.3, “Bringing up your base sysplex” on page 78, the VM **COUPLE** commands connected CTC E40 in S0W1 to E41 in S0W2. XCF CTCs are unidirectional; that is, each CTC is a PATHIN (meaning that XCF messages are received on that device) or a PATHOUT (meaning that XCF messages are sent on that device). Therefore, you must always connect a PATHIN device to a PATHOUT device. The CTCs are defined by using the following statements in the COUPLEPS member:

```
/* DEFINE XCF TRANSPORT CLASS PATHS FOR BASE SYSPLEX */
PATHIN  DEVICE(E20)                                /* DEFAULT TC */
PATHIN  DEVICE(E40)                                /* DEF8K TC */
PATHIN  DEVICE(E42)                                /* DEF61K TC */
PATHOUT DEVICE(E21)                                CLASS(DEFAULT)
PATHOUT DEVICE(E41)                                CLASS(DEF8K)
PATHOUT DEVICE(E43)                                CLASS(DEF61K)
```

On system S0W1, you connect E20 (a PATHIN) to E21 (a PATHOUT) on S0W2, and E21 (a PATHOUT) to E20 (a PATHIN) on S0W2. Therefore, the devmap file should connect the CTCs devices to the opposite type of CTC on the target system. In the statements that are shown in Example 4-6, E40 on this system is connected to E41 on the remote system. Also, the port number for this connection is 4000. If you review the definition of device E41 on the other PC, you see that it specifies device number E40 as the target CTC device. It also specifies the same port number as the E40 definition on PC1. Similarly, E41 on PC1 is connected to E40 on PC2. But, this connection is using a different port number (4001) in this case. If you have multiple CTC connections, each one *must* use a different port number.

When you run the **awsstart** command on the second PC, you do not see any messages about the CTC connection unless there is a problem. If you want to check that the two PCs are connected, run the zPDT **awsstat** command.

### 4.6.3 Sharing zPDT DASD across PCs

Our base sysplex configuration requires shared DASD operation for all of the emulated 3390 volumes. Setting up Linux shared directories is beyond the scope of this book.

The process that is used to share zPDT DASD across PCs includes the following steps on an openSUSE 13.1 system:

1. On our openSUSE Linux system that holds all of the emulated 3390 volumes, we used YaST functions to create an NFS server with the following parameters:
  - NFS server: Start
  - Firewall: (firewall is disabled; you might request opening a port in your firewall)
  - Enable NFSv4: yes

- Enable GSE security: no
  - Directories to export: /z (this directory contained all our emulated volumes)
  - Host wild card: 192.168.1.230 (the address of our second system Linux)
  - Options: fsid=0,crossmnt,rw,root\_squash,sync,no\_subtree...
2. On our second Linux system, we defined mount point /z2 (owned by user ibmsys1) with read and write access by everyone. We selected this mount point to be different from the /z mount point that we normally use for local volumes (we did not want there to be any confusion about to which PC's disks we were referring). We then used YaST services to define an NFS client with the following parameters:
- NFS shares:
- | Server       | remote-directory | mount-point | NFS type | options     |
|--------------|------------------|-------------|----------|-------------|
| 192.168.1.99 | /z               | /z2         | nfs      | rw,defaults |
- Enable NFS4: yes
  - NFSv4 Domain Name: localdomain (use default value unless you know better)
  - Enable GSS Security: no
  - Firewall settings: (firewall is disabled)
3. In principle (perhaps after rebooting both systems), you should see the shared volumes from the second Linux system by running a command, such as `ls /z2`. In practice, we ran the following command on the second system (working as *root*) to see the shared volumes from the second system:
- ```
# mount -t nfs4 192.168.1.80:/ /z2
```
- (We are not NFS experts and considerable experimentation at times was needed to make the sharing work.) Be certain that the *rw* (read/write) option is present on the NFS server and client.
4. Be certain the `--shared` option is present on the `awsckd` device manager statements in the `devmaps` on both systems. (This statement enables RESERVE/RELEASE emulation.) Be certain the `devmap` for the second system contains the correct mount points (/z2 in our example) for all of the emulated disk volumes.

When the cross-PC file sharing is working, update the `devmap` file for the second PC to change the file names to include the `z2`.

**Note:** Although functional and sufficient for function testing, this configuration might not deliver the levels of performance that you require. If you can make a larger financial investment in improving the performance of a multi-PC zPDT configuration, see Appendix C, “Alternative disk configuration” on page 141.

#### 4.6.4 Required parmlib changes

In addition to the infrastructure changes (STP, CTCs, and shared DASD) that are required to span the base sysplex across more than one PC, a few changes to `parmlib` are required. The systems must be told to use STP for their time synchronization, meaning that the `CLOCKxx` member must be updated (the `ADCD` default is to disable `ETRMODE` and `STPMODE`). You also need to configure `GRS` for Ring mode rather than Star mode. Also, each `RRS` must be started with a different group name to stop them from trying to share log streams.

## LOADST member

You need some way to tell the system that it is to come up in base sysplex, not under z/VM mode. You also need a way to set system-specific system symbols. When running under z/VM, you can use the VM user ID as a filter in the IEASYMxx member. However, this sysplex is not running under VM so that method is not an option.

However, you *can* use the LPAR Name. When running under zPDT, the instance name (which is set to the Linux user ID that started zPDT) is loaded into the LPAR name field. Therefore, you can filter by using the LPARNAME field in your IEASYMxx member.

**Important:** When running a base sysplex that spans two PCs, you must start zPDT with a different Linux user ID on each system. Failing to use this configuration can result in problems because it appears that the same system is being loaded twice into the same sysplex.

Consider the following points:

- ▶ The LOADST member is identical to LOADPS, except that it points at IEASYMST.
- ▶ The LOADST member is copied into SYS1.IPLPARM when you run the STPPARM job.

## IEASYMST

IEASYMST is identical to IEASYMPS with the exceptions that it filters based on the system name, and it specifies that the system uses IEASYS00, IEASYSPS, and IEASYSST when initializing.

**Note:** It is likely that you must customize the IEASYMST member. It is set up on the assumption that the two z/OS systems include instance names of IBMSYS1 and IBMSYS2. If the Linux user IDs that you use to start zPDT have different names, you must adjust this member.

The IEASYMST member is copied into USER.PARMLIB when you run the STPPARM job.

## IEASYSST

The IEASYSST member contains only the following parameters:

- ▶ CLOCK: This parameter is set to ST to use the CLOCKST member.
- ▶ GRS: This parameter is set to TRYJOIN because it is a base sysplex.

The IEASYSST member is copied into USER.PARMLIB when you run the STPPARM job.

## CLOCKST member

The ADCD-provided CLOCK00 member specifies NO for STPMODE and ETRMODE. Because a sysplex that spans more than one PC requires STP, you must use a CLOCKxx member that specifies STPMODE YES. Consider the following points:

- ▶ The ADCDST member is a complete replacement for the CLOCK00 member.
- ▶ The CLOCKST member is copied into USER.PARMLIB when you run the STPPARM job.

## SMFPRMBS

The SMFPRMBS member is identical to the SMFPRMPS member with the exception that it specifies system-unique log streams rather than a single log stream that is shared by both systems.

The SMFPRMBS member is copied into USER.PARMLIB when you run the STPPARM job.

## VTAMBS

The VTAMBS member starts each RRS with a different group name, which avoids both RRSs trying to share the log stream (cross-system log stream sharing is not possible in a base sysplex).

Run job STPPARM in SYSPLEX.PARALLEL.CNTL to copy these members into USER.PARMLIB and SYS1.IPLPARM.

### 4.6.5 Loading the systems

All of the pieces that are required to start a base sysplex without z/VM should be in place now. We strongly suggest allowing the first system to complete the entire startup process before you load the second system. We also recommend that you load the system that is in the same PC as the NFS server first.

With Linux sharing enabled, some operations are slower. For example, starting the second system (NFS client) with no connection to the NFS server causes the start function to be *much* slower. Loading the local system first might load the Linux cache with data that is used by the IPL, which provides a small benefit for later loads by remote systems. Even allowing for that benefit, a load of the second system is much slower than a load of the first system.

Both systems can be loaded by using the `ipl a80 parm 0A82STM1` command.

As with any time you load the first system in a sysplex, you might receive a WTOR asking if you want to initialize the sysplex. If you receive that message, reply R 00,I. You can expect to see messages when the system is initializing that indicate that it cannot access CFs CFCC1 or CFCC2. You are seeing these message because the base sysplex is using the same CDSs (including the CFRM data set) as the Parallel Sysplex. Because you are coming up in base sysplex mode, you can ignore those messages.

When the first system finishes initializing, logon to TSO and browse back through the syslog to verify that everything started as expected. If everything is OK, submit job LOGRRRS in SYSPLEX.PARALLEL.CNTL to allocate the DASDONLY log streams that are used by RRS on each system.

The OPERLOG and LOGREC log streams have fixed names. Base sysplex supports only DASDONLY log streams, and DASDONLY log streams cannot be shared across systems. In a base sysplex, a maximum of one system can use OPERLOG or LOGREC, which negates a much of the value of those two functions. As a result, we did not provide jobs to create DASDONLY versions of the OPERLOG or LOGREC log streams.

Address any unexpected messages. Then, load the other system by using the same IPL address and load parameters. When that system is initializing, you should see messages that indicate that XCF is connecting to the other system by using the CTC addresses that you specified in the devmap file.

## 4.7 Operating your sysplex

After your sysplex finishes initializing, operating the systems should be similar to running in a normal monoplex environment, except for all of the extra functions that are available.

For example, when you have a single system, there always is a concern that the system might not come back up after a change is made. When you have a sysplex, you can stay logged on on one system while you load the other system. If there is a problem, you now have a running, fully functional system that you can use to address whatever is stopping the system from loading.

## 4.7.1 Managing your x3270 sessions

At least five 3270 sessions are needed for the sysplex. If you try to squeeze all sessions onto the one window, the window can be cluttered, which makes it easy to enter commands in the wrong 3270 session.

We suggest that two Linux desktop workspaces with the arrangement that is shown in Figure 4-5 is used.<sup>7</sup> The 3270 windows that are related to the S0W2 system are in the second workspace, which helps prevent confusion when the multiple 3270 windows are used. The 3270 addresses that are shown in Figure 4-5 correspond to the 3270 device addresses that are defined in the devmap file. An example of the interactions with the various x3270 sessions is provided in Appendix D, “Sample IPL flow for sysplex under z/VM” on page 145.

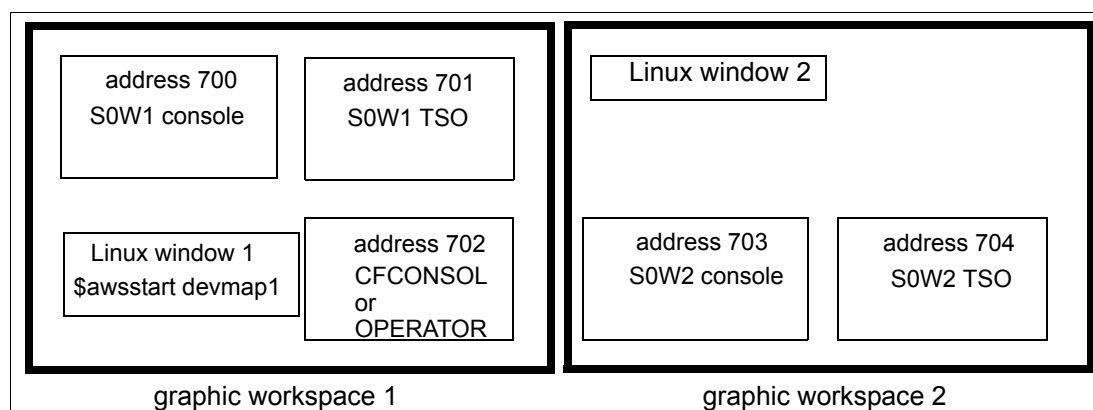


Figure 4-5 Sample lay out for x3270 sessions

## 4.7.2 Post-loading messages

If you use the S0W1 system in monoplex mode, you might see unusual messages during start because it appears to z/OS that systems in different LPARs are trying to use the same page data sets, as shown in the following example:

```

ILR030A PAGE DATA SET MAY BE IN USE
ILR031A REPLY 'U' TO PREVENT ACCESS, 'CONTINUE' TO ALLOW USE OF SYS1.....
r 00,continue <---your reply
  
```

In this case, ensure that only one system is using those page data sets. If it is, reply `continue`, as shown in the example.

<sup>7</sup> Another option is to use a separate PC for the 3270 sessions that are related to the S0W2 system.



### 4.7.3 SMF considerations

The default mode of operation for an ADCD system is for SMF collection to be disabled. In a purely development system (especially with one or a few users), you might find that you rarely use the SMF data. However, if you want to collect SMF records, the following options are available, depending on how you are running your systems:

- ▶ In a Parallel Sysplex, you can share a single log stream between both systems. This configuration is convenient because all of your SMF records are in a single location. You can leave the records in the log stream and have system logger automatically delete them after a retention period that is specified on the log stream (the SMF log stream that is delivered with the 2016 Sysplex Extensions has a retention period of 14 days, but you can change this setting at any time by updating the log stream definition). The log stream for Parallel Sysplex mode is called IFASMF.GENERAL.
- ▶ In a base sysplex, a log stream cannot be shared across multiple systems. However, you can still write the SMF data from each system to its own log stream. This configuration provides most of the benefits of the single log stream, with the exception that you no longer have a single repository of all SMF data. The log streams that we defined for base sysplex mode are called IFASMF.S0W1 and IFASMF.S0W2.
- ▶ In any mode (monoplex, base sysplex, or Parallel Sysplex), you can use the traditional SYS1.MANx VSAM data sets.

The 2016 Sysplex Extensions provides two SMFPRMxx members: SMFPRMBS and SMFPRMPS. Both members contain definitions for SYS1.MANx data sets with data set names that contain the system name (for example, SYS1.S0W1.MAN1). The SMFPRMBS member is used when the system is loaded by using a loadparm that indicates that it is to come up in base sysplex mode. That member contains definitions for the system-specific log stream name. The SMFPRMPS member is used when the system is brought up in Parallel Sysplex mode and it contains a definition for the shared log stream.

By default, SMF recording is disabled. If you enable recording, the default is to use the SYS1.MAN data sets. If you want to use the log streams instead, update the appropriate SMFPRMxx member to say RECORDING(LOGSTREAM) and run the **SET SMF=xx** command to get SMF to use the parms in the named member.

### 4.7.4 Sysplex policies

The sysplex policies that are provided by the 2016 Sysplex Extensions are sufficient to get the sample Parallel Sysplex (complete with CICSplex and DB2 data sharing) up and running. Nevertheless, it is likely that you will want to make changes to these policies over time. We recommend that you create a local copy of the provided jobs and make any changes that you want in those local copies. The sources for the provided policies are in the following members in SYSPLEX.PARALLEL.CNTL:

- ▶ POLARM1: Automatic Restart Manager policy
- ▶ POLCFRM1: Coupling Facility Resource Management policy
- ▶ POLSFM1: Sysplex Failure Management policy
- ▶ LOGRCICS: Defines CICS model log streams
- ▶ LOGREREP: Defines the LOGREC log stream
- ▶ LOGRHC: Defines the z/OS Health Checker log stream
- ▶ LOGROPR: Defines the OPERLOG log stream
- ▶ LOGRRRS: Defines the RRS log streams
- ▶ LOGRSMF: Defines the SMF log streams

If you want to change the ARM, CFRM, or SFM policies, we strongly recommend that you give the new policies a different name to the active policy. The policy name is specified on the NAME parameter on the DEFINE POLICY statement in the policy. When you activate the new policy, the policy name is specified in the SETXCF START,POLICY command. By using a different name, you can fall back to the old policy if there is a problem with the new policy.

The sources for the policies that are provided by the 2016 Sysplex Extensions are in Appendix B, “Sysplex couple data sets and policies” on page 127.

## 4.7.5 Shutdown

Although you might be tempted to shut down your system by ending it, this process can result in data loss and unexpected messages the next time the system is loaded. For these reasons, we strongly recommend taking the time to complete an orderly shutdown of your system. This suggestion applies to any system, but *especially* to a system in a zPDT Parallel Sysplex.

ADCD provides the VTAMAPPL tool to automate the start up and shut down of your system. The ADCD-supplied proc for this process is called SHUTALL. We created a slightly modified version called SHUTSYS. The SHUTSYS JCL is in USER.PROCLIB. It points at members SHUTSOW1 or SHUTSOW2 in USER.PARMLIB, depending on which system is being stopped.

Depending on which started tasks are running on your systems, you might want to customize these members to remove commands for started tasks that you do not use and add commands for your started tasks that are not included in the provided members.

The example shows how you perform an orderly shutdown of both systems (the numbers in parenthesis at the start of some lines refer to the device numbers that are used in Figure 4-5 on page 86):

```

logoff of all TSO sessions
(703)  S SHUTSYS                (start shutdown script for SOW2)
      (wait for the ALL AVAILABLE FUNCTIONS COMPLETE message)
(703)  $PJES2
      (wait for the JES2 ENDED message)
(700)  S SHUTSYS                (start shutdown script for SOW1)
      (wait for the ALL AVAILABLE FUNCTIONS COMPLETE message)
(700)  $PJES2
      (wait for the JES2 ENDED message)
(700)  V XCF,SOW2,OFFLINE
nn IXC371D CONFIRM REQUEST TO VARY SYSTEM SOW2 OFFLINE, REPLY
SYSNAME=SOW2 TO REMOVE SOW2 OR C TO CANCEL
(700)  nn,SYSNAME=SOW2
nn IXC102A XCF IS WAITING FOR SYSTEM SOW2 DEACTIVATION. REPLY DOWN
WHEN MVS ON SOW2 HAS BEEN SYSTEM RESET
      (If you simply wait for about 15 seconds, z/VM will end SOW2 so there
is no need to reply DOWN)
      (wait for console activity to stop; usually after a CONSOLE
PARTITION CLEANUP COMPLETE message.)
(703)  LOGOFF
      (to log off the SOW2 virtual machine)

(700)  V XCF,SOW1,OFFLINE      (this allows PDSE, etc, to stop cleanly)
(700)  nn,SYSNAME=SOW1
(700)  LOGOFF
      (to logoff the SOW1 virtual machine)

```

```

(CFCONSOL) FORCE CFCC1
(CFCONSOL) FORCE CFCC2
(CFCONSOL) SHUTDOWN (shutdown z/VM)
(Linux 1) $ awsstop

```

The V XCF command and the response to the WTOR can be issued on either system, but must name the system that is being stopped. It is important to complete this step. If you do not complete this step, the other members of the sysplex might think that you are loading another system with a duplicate name (because you never cleanly removed it from the sysplex) the next time you attempt to load the system.

## 4.7.6 Useful commands

A few commands can be entered directly on a CFCC console. Because we use a common z/VM interface to the CFCC virtual machines, we must use the z/VM **SEND** command. The following examples direct CFCC commands (these are entered on the CFCONSOL terminal window) can be used:

```

SEND CFCC1 DISPLAY MODE
SEND CFCC2 DISPLAY RESOURCES
SEND CFCC1 DISPLAY CHPIDS
SEND CFCC2 HELP

```

Several z/OS commands are directly related to CF usage. In principle, the following commands are issued through the MVS operator console. In practice, some of the commands produce too much output to be useful on the MVS console panel and are best used through the SDSF LOG interface:

|                           |                                             |
|---------------------------|---------------------------------------------|
| D CF                      | Best use SDSF interface                     |
| D XCF                     | Brief XCF status                            |
| D XCF,CF                  | More detailed information                   |
| D XCF,COUPLE              | Lots of output; use SDSF interface          |
| D XCF,POLICY              |                                             |
| D XCF,STR                 | List defined structures                     |
| D XCF,STR,STRNAME=ISGLOCK | Details about specific structure; use SDSF  |
| D LOGREC                  | LOGREC status                               |
| D LOGGER,L                | LOGGER status                               |
| V OPERLOG,HARDCPY         | If not already using LOGGER for some reason |
| SETLOGRC LOGSTREAM        | Change LOGREC recording                     |
| SET SMF=xx                | Switch to new SMFPRMxx member               |

The following z/OS commands are not directly related to CF usage, but are helpful in determining the state of the system:

|              |                            |
|--------------|----------------------------|
| D C,HC       | Hardcopy log status        |
| D ETR        | External timer mode status |
| D C          | MVS console status         |
| D IPLINFO    |                            |
| D M=CPU      |                            |
| D IOS,CONFIG |                            |
| D M=STOR     |                            |
| \$D MASDEF   |                            |
| \$D MEMBER   |                            |

The following Linux commands might be useful for base sysplex setup:

|                     |                              |
|---------------------|------------------------------|
| \$ cat /etc/exports | Used on an NFS server system |
| # showmounts -e     | Used on an NFS client system |

## 4.7.7 Defining new CDSs

There might be situations where you want to move to new CDSs. For example, the supplied CDSs are formatted for a sysplex that contains up to four systems. If you want to have more than four systems in your sysplex, you must create a set of CDSs that are formatted with an appropriate MAXSYSTEM value.

The following basic mechanisms are available for moving to a new set of CDSs:

- ▶ Define the new CDSs (by using the DEFCDSS job in SYSPLEX.PARALLEL.CNTL as a model), and use the SETXCF ACOUPLE and SETXCF PSWITCH commands to copy the contents of the CDSs to the new CDSs.
- ▶ Shut down the sysplex, delete the data sets, define new (empty) data sets (by using the DEFCDSS job in SYSPLEX.PARALLEL.CNTL as a model), and bring the sysplex back up again.

**Note:** Most CDSs contain information that is constantly updated by the systems in the sysplex. In addition, some CDSs contain information (policies) that you create and place in the appropriate CDS; for example, ARM, SMF, or CFRM. If you delete and redefine the primary and alternative CDSs, all of that information is lost.

The first mechanism is almost always used in a production environment or *any* environment in which a sysplex-wide outage is unacceptable. It also has the advantage that all of the information (policies and status information) in the CDSs is carried forward to the new CDSs.

If you decide to use the first mechanism, the process includes the following steps:

1. Create a copy of the DEFCDSS job that allocates the data sets you require, along with the changes that you want to make (for example, larger MAXSYSTEM value).
2. Run a **SETXCF ACOUPLE** command to replace the alternative data set with the data set that becomes the new primary.
3. Run a **SETXCF PSWITCH** command to replace the primary data set with your new primary data set.
4. Run another **SETXCF ACOUPLE** command to add the new alternative CDS.
5. Update the COUPLEPS member of USER.PARMLIB to specify the new CDS names. This step ensures that the sysplex uses the correct data sets the next time that you load a system or the entire sysplex.

At the end of this process, you are running on the new CDSs and all of the status and policy information from the corresponding old CDSs are carried forward.

However, if your objective is to discard all of the contents of the CDSs and start with a clean sheet, the process is more complex. It is especially important to understand that if you use the second mechanism to replace the Logger CDSs, all of the information in the existing log streams is inaccessible.

The process that is used to replace the CDSs by using the second mechanism includes the following steps:

1. Shut down the Parallel Sysplex and z/VM.<sup>8</sup>
2. Load the basic ADCD system (not under z/VM) with an IPL parameter, such as 0A82CS.

<sup>8</sup> This step also can be done by using the BASEAD guest under z/VM. When this approach is used, you must stop the two coupling facility virtual machines after you stop and log off the S0W1 and S0W2 virtual machines.

3. Using ISPF option 3.4, delete all of the Parallel Sysplex CDSs on volume CF0001.<sup>9</sup> Those data sets have names beginning with PARALLEL.ADCDPL. (Do *not* delete the CDSs that are used by the basic ADCD system. These data sets have names beginning with SYS1.ADCDPL and are on other volumes.)

You might also want to delete the logger offload data sets, as described in “Logger data sets” on page 92. Failing to delete these sets can result in confusing messages in the future because Logger attempts to allocate new staging and offload data sets with names that are duplicates of existing data sets.

4. Change whatever parameters you want in the coupling definitions in the DEFCDSS, and POLaaaa jobs in PARALLEL.SYSPLEX.CNTL.
5. Run job DEFCDSS to create the CDSs and verify that the job ends with return code zero.
6. Update the COUPLEPS member of parmlib to specify the new CDS names, if necessary.
7. Run the POLaaaa jobs to install your policies into the new CDSs.

**Important:** The default action when you run the IXCMIAPU utility to create a policy is to install that policy in the corresponding in-use CDS. However, you are running with the ADCD-provided CDSs in this case, but you want to install your new policies into the CDSs you just created. Therefore, it is *vital* to ensure that the POLaaaa jobs include the name of the target CDS on the DEFINE POLICY statement, as shown in Example 4-8.

This method can be used only with the ARM, CFRM, and SFM policies. It is not possible to update the LOGR policy in any data other than the active LOGR CDS.

*Example 4-8 Sample IXCMIAPU statements*

---

```
//POLARM1 JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID.  
//ARMPOL EXEC PGM=IXCMIAPU  
//SYSPRINT DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//SYSIN DD *  
DATA TYPE(ARM) REPORT(YES)  
DEFINE POLICY NAME(ARM1) DSN(SYSPLEX.ADCDPL.NEW.ARM.CDS01) REPLACE(YES)
```

---

8. If you created CFRM CDSs, update the COUPLEPS member to add the CFRMPOL(aaaaaa) statement. This step is necessary if you want to start in GRS Star mode and the CFRM CDSs being used for the IPL were not used previously.
9. Complete any other tasks that are easier in an unshared system.
10. Shut down the basic ADCD system.
11. Start z/VM, the two Coupling Facility Virtual Machines, and load the S0W1 system (with IPL parameter 0A82PS).

Several unexpected error messages might appear in the log. These messages occur because the various log streams are not yet defined.

12. Run the following jobs from the SYSPLEX.PARALLEL.CNTL data set:
  - LOGRCICS for the CICS model log streams
  - LOGREREP for the LOGREC log stream
  - LOGRHC for the z/OS Health Checker log stream
  - LOGROPR for the OPERLOG log stream
  - LOGRRRS for the RRS log streams
  - LOGRSMF for the SMF log streams

---

<sup>9</sup> Our installation example in this chapter uses this volser; other volumes can be used.

Use the z/VM command **DIAL SOW1** to access a VTAM session.

13. Edit the COUPLEPS parmlib member again and remove the CFRMPOL statement.

The extra CF structures are now available, but your SOW1 system is not using the structures that were added by the LOGRaaaa jobs. You can use the structures immediately by running the following MVS commands:

|                             |                                     |
|-----------------------------|-------------------------------------|
| V OPERLOG,HARDCPY           | (starts OPERLOG operation)          |
| SETLOGRC LOGSTREAM          | (moves LOGREC to the log stream)    |
| SETSMF RECORDING(LOGSTREAM) | (change SMF to use log stream mode) |
| S RRS,SUB=MSTR              | (restart RRS to use CF structures)  |
| D XCF,STR                   | (display the allocated structures)  |

The next time that you load (with IPL parameter PS), these log streams are used automatically.

### Logger data sets

Be aware that the CDSs might contain data that is needed across IPLs. The most obvious case involves LOGGER staging and offload data sets. In normal use, LOGGER automatically deletes its offloaded data sets when the retention period for all of the records in the data set expires. If you delete and re-create the CDSs, this retention period information for LOGGER data sets is lost and the data sets remain on your disks until they are explicitly deleted. These sets are VSAM data sets and are slightly more difficult to delete than simple sequential or partitioned data sets.

The LOGGER data sets normally have the high-level qualifier IXGLOGR (but have a HLQ of LOGGER or START1 for the CICS log streams in our parallel Sysplex system) and have full names, as shown in the following example:

|                                           |                                |
|-------------------------------------------|--------------------------------|
| IXGLOGR.IFASMF.GENERAL.A0000000           | <i>&lt;==base cluster name</i> |
| IXGLOGR.IFASMF.GENERAL.A0000000.DATA      |                                |
| IXGLOGR.SYSPLEX.LOGREC.ALLRECS.A0000000   | <i>&lt;==base cluster name</i> |
| IXGLOGR.SYSPLEX.LOGREC.ALLRECS.A0000000.D |                                |
| or                                        |                                |
| LOGGER.IFASMF.GENERAL.A0000000            | <i>&lt;==base cluster name</i> |
| LOGGER.IFASMF.GENERAL.A0000000.DATA       |                                |
| LOGGER.SYSPLEX.LOGREC.ALLRECS.A0000000    | <i>&lt;==base cluster name</i> |
| LOGGER.SYSPLEX.LOGREC.ALLRECS.A0000000.D  |                                |

The A0000000 qualifiers in these examples are for the first LOGGER data set in each group. This number is incremented when LOGGER data sets are created.

## 4.7.8 Adding 3390 volumes in Parallel Sysplex

You might want to add emulated 3390 volumes to one or both z/OS systems. The z/VM directory for the z/OS guests that is described in this chapter has directory definitions for z/OS 3390 volumes at addresses A80 - ABF, which might be sufficient for your extra volumes.

Complete the following steps:

1. Create (or restore) the volumes. Creating a volume involves the use of the zPDT **a1cckd** command as described in 4.3.3, "Download and create volumes" on page 42.

2. Add the new volume to the devmap. The devmap file is used in our examples is called devmapp. Use an address in the A80 - ABF range, if possible. If this range cannot be used, complete the following steps to modify the z/VM directory:
  - a. Start zPDT (run the **awsstart devmapp** command) and IPL z/VM (run the **ipl 200 parm 0700** command).
  - b. Log on as user maint (password is SSI1<sup>10</sup>).
  - c. Edit the VM directory (run the **xedit user direct c** command).
  - d. Scroll to the section that begins with user ID MVSDUMMY.
  - e. For more information about the devices that are defined in the ADCD-provided IODF, see “Devices that are defined in the ADCD-provided IODF” on page 122. If possible, use an address that is defined to avoid having to update the IODF.
  - f. By using the patterns for user IDs MVSDUMMY, BASEAD, S0W1, and S0W2, add lines for your new volumes, if needed. Add these lines to all four user IDs.
  - g. Exit from Xedit (run the **file** command).
  - h. Rebuild the active z/VM directory (run the **directxa user direct c** command).
3. Start one of the z/OS systems. If you created 3390 volumes (instead of restoring the volumes from a DVD or a download), you must submit an ICKDSF job from TSO to initialize the volumes. (z/OS automatically varies offline any uninitialized volumes it finds during IPL. After you initialize the volumes with ICKDSF, you can vary them online to z/OS.)

## 4.7.9 Adding 3270 terminals in Parallel Sysplex

You can add more 3270 sessions for z/VM or for each z/OS. Consider the following points:

- ▶ z/VM must know about the extra local 3270 devices. This process is completed by adding 3270 definitions in the devmap. z/VM recognizes 3270 terminals, regardless of the addresses that are used. That is, it is not necessary to use addresses in the 7xx range (although our z/VM customization requires a local 3270 session at address 700 for the initial operator session). Remember that a maximum of 32 terminals can be defined for the aws3274 device manager. You might need to run a z/VM **enable all** command to obtain a logon logo on the new terminals.
- ▶ After you define the extra z/VM 3270 sessions, you can use these sessions to DIAL to a z/VM guest. For example, DIAL S0W1 to connect to our first z/OS system.
- ▶ The z/VM user ID definition for a guest (S0W1, S0W2, and BASEAD) must have sufficient SPECIAL statements for all of the 3270 sessions (except the session at address 700 that is used for the MVS console). You can add SPECIAL statements for these user IDs by editing the z/VM directory, as described in 4.7.8, “Adding 3390 volumes in Parallel Sysplex” on page 92. These SPECIAL connections are hardware connections between a 3270 session and the guest z/OS virtual machine.
- ▶ z/OS must recognize the 3270 session. The current ADCD systems recognize local 3270 connections at addresses 701 - 73F. However, the A0600 member in VTAMLST allows only 10 TSO sessions through local 3270 connections. You must modify VTAMLST to increase this number; this task is a normal z/OS systems programming activity.

<sup>10</sup> This password is the password for the z/VM 6.3 ADCD system. Other z/VM implementations feature their own passwords.

- ▶ You can connect 3270 sessions to z/OS TCP/IP. The current ADCD systems allow up to 30 such connections (through definitions in the TCPIP PROFILE and in VTAMLST). These sessions<sup>11</sup> (directly to z/OS TCP/IP) do not require any modifications to z/VM or the devmap.

#### 4.7.10 Scaling up in Parallel Sysplex

Some of the limitations of the starter system that we described are important to understand. It is a relatively small configuration and various steps are required to expand it to a larger, more robust, configuration. In particular, consider the following points:

- ▶ Our two z/OS systems are defined (in the z/VM directory) with a modest amount of memory for each. Depending on the products that are used in the system and the volume of work that is processed, the system might require much more memory.
- ▶ Our sample devmap defines an 8 or 10 GB zPDT system. This devmap must be increased to accommodate the larger z/OS systems.
- ▶ Larger z/OS systems often require larger paging data sets. If large virtual memory applications (such as DB2 applications) are used and if SVC memory dumps might be expected, considerably larger paging areas are required for each z/OS. The MVS command **D ASM** is useful for determining the amount of paging space that is used. Under SDSF, the **DA** function can be used to informally sample dynamic paging rates.
- ▶ Larger z/VM and z/OS memory should have correspondingly larger PC server memory.
- ▶ Our z/VM system has one 3390-3 volume for paging. More z/VM paging volumes might be needed. Monitor for HCPPGT401I and HCPPGT400I messages on the VM OPERATOR console. These messages can precede a VM wait state; therefore, if you encounter these messages, add more paging volumes to z/VM before you add more work to z/OS.
- ▶ Our CFs (in the z/VM directory) are defined with 1200 MB each. Depending on the number and size of your coupling structures, these CFs might need to be larger. Run a **D CF** command on z/OS to determine how much free space you have in each CF. We strongly suggest that you aim for each CF not being more than 50% full to allow for moving all of the structures into one CF if you must restart the other CF.
- ▶ Our z/OS systems have limited STORAGE disk space (as defined in the VATLST00 members in PARMLIB). You might need to add disk volumes that are defined as STORAGE volumes.
- ▶ SVC memory dumps are directed to F1SYS1 and F1SYS2. You might want to direct them to larger volumes that you add.
- ▶ You should monitor paging and swap space in your base Linux. The **xosview** program is convenient for this process if you installed it with your Linux.

Paging (by the base Linux, z/VM, or z/OS) often is a poor use of the limited disk I/O resources in a zPDT system. Larger PC memory, which is combined with a small amount of monitoring to best balance memory usage, can be the most effective way of improving the performance of the zPDT systems. Despite these efforts, you are likely to see base Linux swap activity; this activity appears to be mostly because of a strong Linux preference to use real memory for the disk cache. An effective disk cache is important for zPDT performance.

---

<sup>11</sup> There is a limit of 30 because of TN3270 and VTAM definitions. These limits can be increased, if needed.





## Sample DB2 data sharing environment

The most common user of sysplex data sharing support is DB2. This chapter describes the implementation and operation of the DB2 data sharing configuration that is delivered by the 2016 Sysplex Extensions and includes the following topics:

- ▶ 5.1, “Setting up the 2016 Sysplex Extensions DB2 data sharing environment” on page 96
- ▶ 5.2, “Controlling the DB2 data sharing group” on page 99
- ▶ 5.3, “Maintaining the DB2 environment” on page 99
- ▶ 5.4, “Sample batch job to test DB2 data sharing” on page 100

## 5.1 Setting up the 2016 Sysplex Extensions DB2 data sharing environment

The ADCD system provides the environment to run a DB2 subsystem. However, there were many requests for a pre-packaged DB2 *data sharing* environment. When creating this package, we attempted to adhere to our overall objective of minimizing the time and effort to get this additional functionality up and running.

If we took the option of turning the ADCD DB2 subsystem into a data sharing group, the implementation effort is equivalent to a project to migrate from a single DB2 environment into a data sharing one. That is, there was minimal benefit from the use of the 2016 Sysplex Extensions.

Therefore, we decided to provide a self-contained DB2 data sharing environment based on DB2 V11. The 2016 Sysplex Extensions package consists of two new DB2 subsystems, two SMS-managed volumes containing all of the DB2 databases, DB2 catalogs, logs, archive logs, ICF user catalogs containing the entries for all these data sets, and so on.

A significant portion of the setup was completed as part of the standard 2016 Sysplex Extensions Parallel Sysplex implementation that is described in Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35. That is, much of the work is done. Nevertheless, we describe each of the implementation steps so that you understand the changes that were made for DB2 (we make it clear which steps were already completed).

### 5.1.1 Overview of the DB2 environment

The following information is helpful to understanding the DB2 environment that is delivered by the 2016 Sysplex Extensions:

- ▶ DB2 Group name: DPDG
- ▶ DB2 Location name: DPD1LOC
- ▶ DB2 catalog HLQ: DSNDPDG
- ▶ DB2 group members: DPD1 (normally on system S0W1) and DPD2 (normally on system S0W2)
- ▶ Data set name for tailored SDSNSAMP libraries: DSNDPDG.DPDx.SDSNSAMP

### 5.1.2 Implementation

As a starting point for this process, we assume that you downloaded the DB2001 and DB2002 volumes, and that you completed all of the Parallel Sysplex implementation steps that are described in 4.3, “Setup steps” on page 40. Also, because this package is based on DB2 V11, the ADCD DB2 V11 volumes F1DBB1 and F1DBB2 must also be available.

**Note:** The ADCD deliverable includes a F1DBAR volume and states that it must be placed on a specific device. That volume is used by the ADCD-provided DB2 subsystems. If you are using those subsystems, that volume must be available to your systems. If you are not using the ADCD-provided DB2 subsystems, it is *not* necessary to have that volume.

If you want to get your DB2 data sharing environment up and running as quickly as possible, you can skip to “RACF setup for DB2” on page 98. If you want to understand all of the steps that were necessary to prepare the sysplex for DB2 data sharing, see “SMS setup for DB2 data sharing”.

## SMS setup for DB2 data sharing

**Note:** All of the tasks in this section were completed during the installation of the Sysplex Extensions package, as described in Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35. This description is provided for your information only.

Starting with DB2 V10, DB2 requires that certain DB2 data sets are on SMS-managed volumes. Considering this requirement, we decided that it is simpler to spread all of the DB2 related data sets that are delivered by this package over two volumes, and to make both of them SMS-managed. Configuring the package in this way makes the initial setup a little more complex, but should reduce the operational complexity and management on an ongoing basis.

The following changes to SMS are required for DB2:

- ▶ Add a Storage Group that is called DB2FILES and add two volumes called DB2001 and DB2002 to the storage group.
- ▶ Add a Storage Class called PSADDON.
- ▶ Add a Data Class called DPDGDC:
  - The Data Set Name Type should be set to EXTENDED.
  - The Extended Addressability should be set to YES.
- ▶ Add new entries to your ACS routines by using information from the following members in SYSPLEX.PARALLEL.CNTL:
  - ACSDB2DC contains statements that are added to your Data Class ACS routine.
  - ACSDB2SG contains statements that are added to your Storage Group ACS routine.
  - ACSDB2SC contains statements that are added to your Storage Class ACS Routine.

## Add the DB2 user catalog

**Note:** All of the tasks in this section were completed during the installation of the Sysplex Extensions package, as described in Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35. This description is provided for your information only.

To access the DB2 related data sets by using the standard catalog search order, you must IMPORT CONNECT the DB2 user catalog into the ADCD master catalog. The IMPCONN job in SYSPLEX.PARALLEL.CNTL does this process for you (that job also imports the CICS user catalog and other user catalogs and aliases).

## Parmlib setup for DB2 data sharing

**Note:** All of the tasks in this section were completed during the installation of the Sysplex Extensions package, as described in Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35. This description is provided for your information only.

The two new DB2 subsystems must be defined to the system. The IEFSSNPS member that was copied into USER.PARMLIB contains definitions for two DB2 subsystems called DPD1 and DPD2.

In addition to the standard ADCD-provided DB2 software libraries that must be defined to APF, you must add the DSNDPDG.SDSNEXIT data set to APF.

The DB2 software libraries are included in the ADCD-provided PROGDB member so that member is included in the PROG concatenation in the IEASYSPS member. However, we created a PROGxx member for the SDSNEXIT library, so we called that PROGPS and included it in the PROG concatenation in IEASYSPS.

The DB2 V11 link library also must be added to the LNKLIST concatenation. Rather than creating a PROGxx member, we concatenated the ADCD-provided PROGBP member to our PROG concatenation in IEASYSPS.

**Note:** All of the steps up to this point were completed during the standard Sysplex Extensions installation that is described in Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35. The steps from this point forward were not completed; therefore, you must complete these steps.

## RACF setup for DB2

Several RACF permissions are required to grant access to the new DB2 subsystems. Rather than manually grant all of the accesses by using the RACF ISPF panels, we provided four jobs in SYSPLEX.PARALLEL.CNTL to issue the following required RACF commands:

- ▶ Run job RACFDB21 to permit IBMUSER to connect to DB2 from batch jobs.
- ▶ Run job RACFDB22 to permit started tasks and batch jobs to connect to DB2.
- ▶ Run jobs RACFDB23 and RACFDB24 to add RACF profiles for the WLM environment needed for DB2.
- ▶ Run job GRANT to grant shutdown authority to the VTAMAPPL user ID (to be used as part of the system shutdown process).

**Note:** The RACF profiles and accesses that are defined by these jobs are acceptable for a single-user development environment. However, if you have users on the system that should not have the provided level of access, you should adjust the sample jobs.

Take the time to review the job output to verify that all of the commands were processed successfully. A return code 0 for the job does *not* necessarily mean that all of the commands were processed successfully.

## Workload Manager set up for DB2

Although they are not required for the basic DB2 environment that is provided by the 2016 Sysplex Extensions, the DB2 ADMT component issues error messages if the WLM application environments that are used by DB2 are not defined.

These messages can be ignored; however, it is best to run job WLMDB21 to suppress them. This job defines and activates the application environments that are needed for our DB2. This job is a copy of the DB2 installation job DSNTIJRW.

## VTAM setup for DB2

The DB2 Distributed Data Facility requires an entry in VTAMLST and an update to the ATCCONPS member to automatically activate the DB2 LU.

The update to ATCCONPS was handled during the 2016 Sysplex Extensions installation, so no further action is required in relation to that member.

However, the DPD1LU and DPD2LU members were *not* copied to VTAMLST, so you must run job DB2VTAML in SYSPLEX.PARALLEL.CNTL now.

After the job completes successfully, issue a **RO \*ALL,V NET,ACT,ID=DPD&SYSNUM.LU** command to activate the DB2 VTAM definitions.

### DB2 started task procedures and other proclib members

DB2 and all its associated address spaces require their PROCs to be included in a data set in the JES proclib concatenation. In addition to the procs for starting all of the DB2 tasks, DSNPDG.PROCLIB contains the various DB2 procedures that are used for application development.

Run job DB2PROCS in SYSPLEX.PARALLEL.CNTL to copy the required members from DSNPDG.PROCLIB to USER.PROCLIB.

You might need to tailor the DB2 procedures that are used for compiling programs (DSNHASM, DSNHC, DSNHICOB, and DSNHPLI) depending on which releases of CICS and IMS you might be using. We included the CICS TS 5.2 libraries, but the IMS libraries are commented out.

**Note:** This release of the zPDT Sysplex Extensions does *not* include the enablement of the CICS to DB2 connection. All of the pieces are in place, and you can connect the CICS regions to the provided DB2 data sharing group. We hope to provide this connection and a sample CICS/DB2 workload in a future release of the Sysplex Extensions.

## 5.2 Controlling the DB2 data sharing group

**Note:** The ADMT part of DB2 uses RRS. Therefore, before you start DB2, ensure that you defined the RRS log streams and that RRS is up and running. Run the **D RRS,UR,S** command to verify that RRS is active.

To start DB2, run a **-DPDx START DB2** command, where x is 1 or 2 depending on which DB2 subsystem instance you are starting. Either DB2 can run on either system, but we assume that you normally run DPD1 on system S0W1, and DPD2 on S0W2.

To stop DB2, run the following commands (these commands are included in the SHUTS0Wn members of USER.PARMLIB):

```
F DPDxADMT,APPL=SHUTDOWN
-DPDx STOP DB2
```

To access the DB2 ISPF panels, log on to TSO by using the ADCD-supplied DBSPROCB logon proc. User IDs IBMUSER and ADCDMST were defined with SYSADM authority in DB2. You can grant authority to more IDs as required.

## 5.3 Maintaining the DB2 environment

DB2 was installed up to Step 20 in the DB2 installation guide (jobs DSNTIJRT and DSNTIJRV, which enable the DB2 supplied routines).

If you want to change or refer to the DB2 setup jobs, they are contained in data set `DSNDPDG.DPDx.SDSNSAMP`. `DPD1` was the first subsystem, so it has the full suite of setup jobs. The jobs that are required for `DPD2` were only the subset that is required to add a member to a data sharing group.

The `SDSNSAMP` data set contains the `ZPARM` jobs that are used for each DB2 subsystem.

If you must rerun the install/migration clist to make changes to the configuration, the output members from the initial setup are in `DSNDPDG.DPDx.SDSNSAMP`. The member for `DPD1` is called `DSNTIDXE`, and the member for `DPD2` is `DSNTIDXF`. You must copy these members back to the ADCCD-provided `DSNB10.SDSNSAMP` library before you can run the installation panels. Specify these members as the input on the initial panel.

## 5.4 Sample batch job to test DB2 data sharing

We assume that you have your own data and programs that you want to import into the data sharing group. However, if you want to validate the correct functioning of DB2 and DB2 data sharing, we included two jobs called `DB2IVP1` and `DB2IVP2` in `SYSplex.PARALLEL.CNTL`. Although these jobs are simple batch jobs that are based on the IBM-provided DB2 IVP jobs, they are an effective mechanism for verifying that all of the components of the DB2 data sharing environment are working.

To run the jobs, ensure that DB2 is started on both systems and then, submit the two jobs. You should then see the DB2 group buffer pool structure being allocated.

If you want to use WLM Scheduling Environments to control whether DB2 jobs can be started on a specific system, you can use a Scheduling Environment called DB2AVAIL that we defined in the WLM policy. To avail of this function, add the SCHENV parameter to the JOB card of your DB2 jobs and use the SDSF SE option to set the status of the DB2AVAIL resource to the wanted setting. An example of the SDSF panel that shows the status of the resource in each member of the sysplex is shown in Figure 5-1.

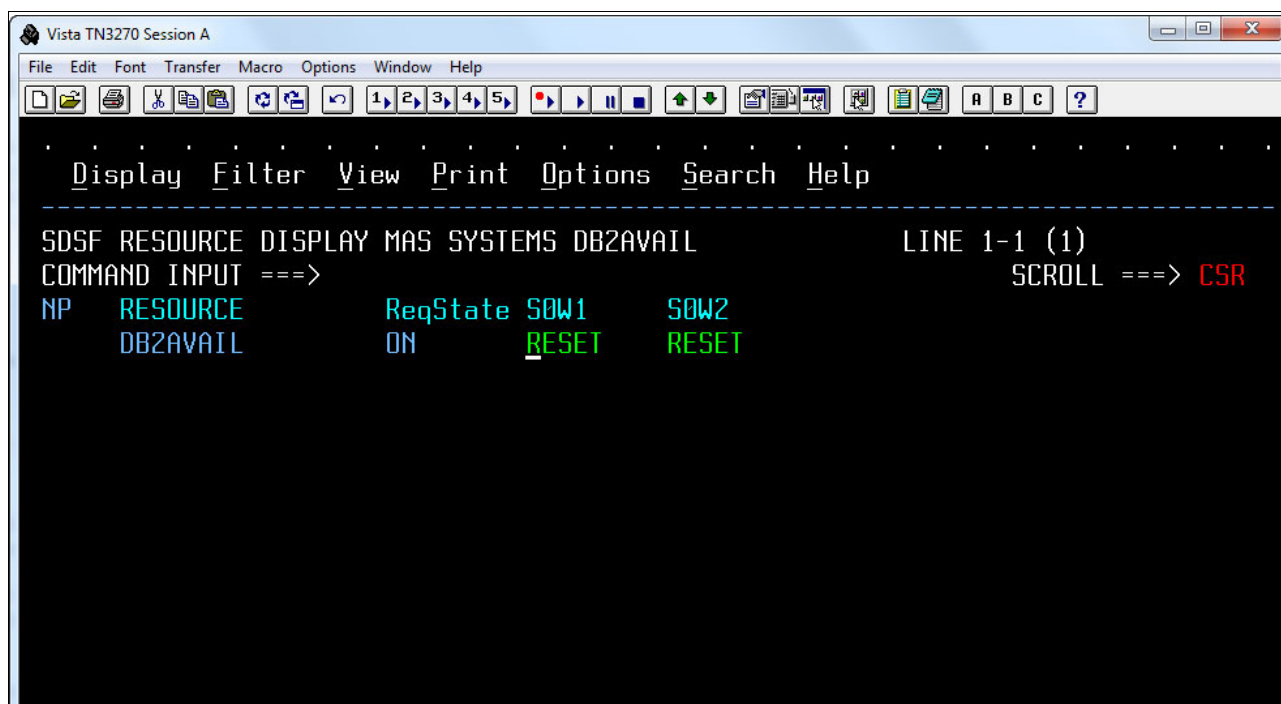


Figure 5-1 SDSF Scheduling Environment Resource Status window

In this example, you can see that the resource DB2AVAIL has a status of RESET on both systems. When you start DB2 on a system, you can access this panel and change the state of the resource on that system to ON. When you are preparing to stop DB2, you can change the state to OFF, which ensures that no new DB2 jobs are started on that system.

In a real-world system, you tie the use of scheduling environments to your system automation product. For example, when DB2 is started, automation uses a command, such as **F WLM,RESOURCE=DB2AVAIL,ON**, to enable running DB2 jobs on that system. Before stopping DB2, automation runs a **F WLM,RESOURCE=DB2AVAIL,OFF** command on that system to stop any other DB2 jobs from being started on that system.







## Sample CICSplex

This chapter describes the CICSplex that is provided by the 2016 Sysplex Extensions and the steps that are required to get it up and running.

This chapter includes the following topics:

- ▶ 6.1, “CICSplex overview” on page 104
- ▶ 6.2, “Setting up the 2016 Sysplex Extensions CICSplex environment” on page 104
- ▶ 6.3, “Controlling your CICSplex environment” on page 106
- ▶ 6.4, “Migration considerations” on page 109

## 6.1 CICSplex overview

The standard ADCD system includes a CICS environment. However, it does not include sysplex-specific functions. To address this requirement, the 2016 Sysplex Extensions provides a working example of a CICSplex in a full Parallel Sysplex environment, complete with a working CICSplex System Manager and the CICS Named Counter, Temporary Storage, and Data Tables servers.

The 2016 Sysplex Extensions also includes a DB2 data sharing environment. In this release, the CICS connection to DB2 was not implemented. That capability is an enhancement that might be added in future releases, along with VSAM Record Level Sharing.

## 6.2 Setting up the 2016 Sysplex Extensions CICSplex environment

Implementing the 2016 Sysplex Extensions-provided CICSplex consists of several steps. However, similar to the DB2 data sharing environment, many of these steps were completed as part of the process of implementing the z/OS Parallel Sysplex environment.

If you want to get your CICSplex up and running as quickly as possible, skip to 6.2.6, “Copying CICS procs to USER.PROCLIB” on page 106. However, if you want to understand all of the steps that were necessary to add the CICSplex to the Parallel Sysplex, see the topics in this section.

### 6.2.1 SMS changes

**Note:** All of the activities in this section were completed as part of the standard Sysplex Extensions installation as described in Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35. This description is provided for your information only.

To minimize the time and manual effort to install our CICSplex, we packaged the CICS environment to be as self-contained as possible. The 2016 Sysplex Extensions includes two SMS-managed volumes that contain the user catalog that is used for all of the CICS runtime data sets, the CICS parameter library, CICS sample VSAM files, and the CICS log streams.

Because the CICS volumes are SMS-managed, all data sets that are on those volumes must have a storage class that is assigned to them. This prerequisite requires setting up an SMS storage class, a storage group (for the CICS SMS volumes), and updates to the SMS ACS routines to assign the correct storage class and storage group to new CICS related data sets.

Specifically, the following changes must be made:

- ▶ Add a Storage Group that is called CICFILES and two volumes called CICS01 and CICS02 to the storage group.
- ▶ Add a Storage Class called PSADDON. This storage class is shared with the DB2 SMS-managed data sets.
- ▶ Update the SMS ACS routines. Sample filter list and SELECT section statements for the CICS data sets are contained in the ACSCICSG member of SYSPLEX.PARALLEL.CNTL.

## 6.2.2 Parmlib changes for CICS

**Note:** All of the activities in this section were completed. This description is provided for your information only.

A subset of CICS load modules must be placed in Link Pack Area (LPA). Rather than changing the LPALSTxx member in ADCD.Z21F.PARMLIB, we created an LPALSTPS member in USER.PARMLIB. That member contains a single entry for CICS TS 5.2 and was concatenated to LPALST01 in the IEASYSPS member.

To simplify the JCL for the CICS started tasks, we created a system symbol that is called &CTSLEVEL and set it to 520. This symbol is defined in the IEASYMPS member.

CICS has two libraries (DFH520.CICS.SDFHLINK and DFH520.CPSM.SEYULINK) that are placed in the system LNKLST. The ADCD-provided PROGxx members contain both APF and LNKLST definitions, which makes it difficult to add libraries to LNKLST without changing those members.

Because we did not want to make extensive changes to ADCD-provided members, we decided to use the ADCD PROGBP member (the IEASYSPS member includes 'BP' in its PROG= statement). A side effect of this change is that you receive a message IGGN505A during the IPL if the CICS TS 5.1 volume is not available in your sysplex. If you want to eliminate that message, you can edit the PROGBP member in ADCD.Z21F.PARMLIB and remove the following statement:

```
LNKLST ADD NAME(LNKLST00) DSN(DFH510.CICS.SDFHLINK) VOLUME(F1C511)
```

Two subsystems must be added to z/OS for CICS: one for the CICS regions and one for use by the extra CICS servers (Named Counter Server, Temp Storage Server, and so on). The 2016 Sysplex Extensions supplied IEFSSNPS member contains definitions for both of these subsystems.

CICS requires an SVC. The ADCD-provided IEASVCCI member contains the definition for that SVC, so we concatenated that member to IEASVC00 in our IEASYSPS member.

## 6.2.3 Adding CICS definitions to VTAMLST

**Note:** All of the activities in this section were completed during the standard Sysplex Extensions installation process that is described in Chapter 4, "Installing the 2016 Sysplex Extensions" on page 35. This description is provided for your information only.

Rather than having to provide separate VTAM definitions for each CICS region, we use model definitions. This method allows us to have a single VTAMLST member that can be used by all CICS regions. It also means that new CICS regions can be added without changing the VTAMLST.

The CICS VTAMLST member was copied as part of the COPYVTAM job that you ran during the Parallel Sysplex installation.

## 6.2.4 Importing connect CICS user catalog

**Note:** All of the activities in this section were completed. This description is provided for your information only.

The entire 2016 Sysplex Extensions CICS environment is self-contained on two volumes. All data sets on those volumes are cataloged in the CICS user catalog and the CICS user catalog is also contained on those volumes.

The CICS user catalog was connected to your master catalog in the IMPCONN job that you ran during the Parallel Sysplex installation.

## 6.2.5 Defining CICS model log streams

**Note:** All of the activities in this section were completed. This description is provided for your information only.

Every CICS region requires at least two log streams. Rather than pre-defining every log stream, you can define model log streams to system logger. A CICS region can then use those models to dynamically define new log streams if a log stream for that region does not exist.

Job LOGRCICS in SYSPLEX.PARALLEL.CNTL adds these model definitions.

## 6.2.6 Copying CICS procs to USER.PROCLIB

We provided a set of procs for each CICS region, including the CMASs and the Web User Interface (WUI). However, those procs were not copied into the USER.PROCLIB data set.

Run job CICSPROC in SYSPLEX.PARALLEL.CNTL now to add these procs.

## 6.3 Controlling your CICSplex environment

The CICSplex environment that is provided by the 2016 Sysplex Extensions consists of the following components:

- ▶ 2 CICS Terminal Owning Regions on each system
- ▶ 2 CICS Application Owning Regions on each system
- ▶ 1 CICS CMAS region per system
- ▶ 1 IBM CICSplex® Web User Interface (WUI) per sysplex
- ▶ 1 CICS CF Data Tables server per system
- ▶ 1 CICS Named Counter Server per system
- ▶ 1 CICS Temporary Storage Server per system

Two TORs and two AORs are provided per system because this configuration gives you the ability to restart one region without affecting the ability of that system to process CICS transactions during that time.

To make it easier to stop and start CICS, we provided the following procs in USER.PROCLIB:

- ▶ STRTCICS: Starts all CICS regions on both systems
- ▶ STRTCICn: Starts all CICS regions on one system
- ▶ STOPCICS: Stops all CICS regions on both systems
- ▶ STOPCICn: Stops all CICS regions on one system

The following naming convention for the CICS regions is used:

```
CICS1A11
  ||||
  |||----- Instance number (this is the first AOR in CICSplex 1 in system 1
  ||----- System number
  |----- Region type (A=AOR, T=TOR)
  ----- CICSplex identifier
```

The CMASs have a different naming convention. The normal recommendation is to have one CMAS per system, so our CMASs are called CMAS0W1 and CMAS0W2. If you have many CICS regions, you might want more than one CMAS per z/OS. In that case, you create a naming convention that communicates the address space type (CMAS), the system that it is associated with, and which CMAS instance it is. For example, you might use CMASW101 and CMASW102.

**Note:** In line with the recommendation that the CICS CMAS regions be placed in the WLM SYSSTC Service Class, we updated the ADCD WLM policy to add CMAS\* to the STCHI classification group.

If you want to start or stop a single CICS region, review the corresponding procedure to see the command that is used to stop or start each region. To minimize the effort when you must change CICS JCL and to reduce the opportunity for errors, our CICS regions are started by using a simple proc that refers to a master member that contains all of the JCL. For example, to start CICS region CICS1A11, you run the **S CICS1A11** command. The CICS1A11 proc in USER.PROCLIB contains the following elements:

```
//CICS1A11 EXEC CICS1A,REGNAM=1A11
```

CICS1A is the member of PROCLIB that contains all of the DD statements for the CICS data sets. The use of a mechanism such as this reduces the effort when you must make a JCL change for all CICS regions.

All the CICS parms, including those for the CICS server regions, are in data set CTS52.USER.SYSIN. The CICS servers use the following pool names:

- ▶ CF Data Tables: PDTCFDT1
- ▶ Named Counter Server: PDTNCS1
- ▶ Temporary Storage Server: PDTSTOR1

If you want to change the definitions of the CICS log streams or add new log streams, refer to member LOGRCICS in data set SYSPLEX.PARALLEL.CNTL. CICS is set up to use model log streams, which means that new log streams are defined dynamically if you start a new CICS region for the first time. The following model log stream names are used:

- ▶ S0W1.DFHLOG.MODEL
- ▶ S0W1.DFHSHUNT.MODEL
- ▶ S0W2.DFHLOG.MODEL
- ▶ S0W2.DFHSHUNT.MODEL

The VTAM APPLID for the CICS AORs and TORs is the same as the started task name. For example, to log on to CICS TOR CICS1T11, you enter L CICS1T11 in the VTAM logon panel. To get a list of the active CICS APPLIDs on a system, issue the **D NET,ID=CTSMODEL,E** command.

After the WUI (started task is called CICSWE11) finishes initializing, you can log on to it by browsing to <http://192.168.1.nn:3005/>. When you connect, you see a window similar to the window that is shown in Figure 6-1.

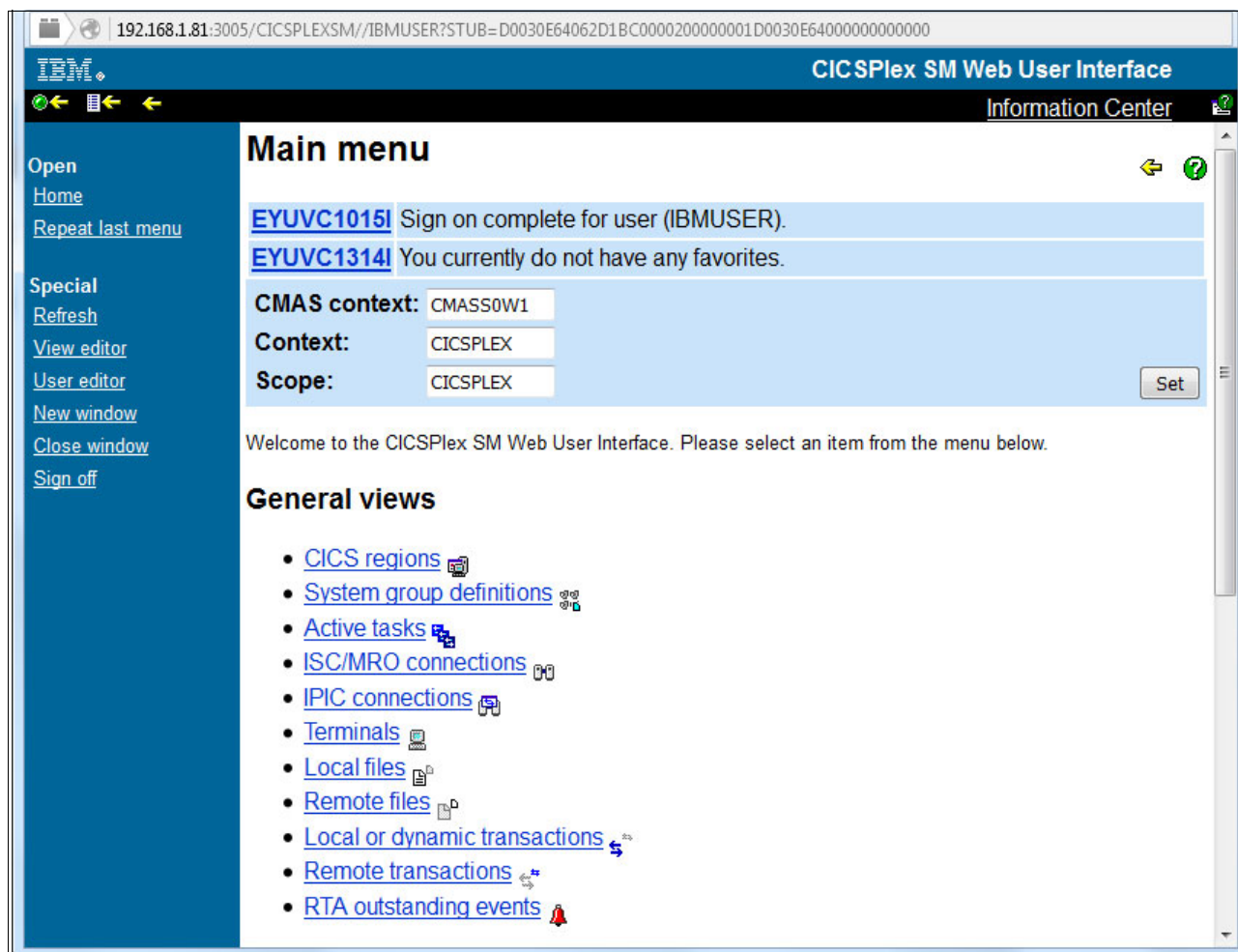


Figure 6-1 CICS WUI Home page

If you are not familiar with the CICS WUI and perform much of the CICS administration work by using the old CICS 3270 interface, we strongly encourage you to investigate the WUI as an easier to use alternative. In a future 2016 Sysplex Extensions release, we hope to provide examples of how the CICS WUI is used to monitor and manage various scenarios.

## 6.4 Migration considerations

In line with the process that is used by the ADCD deliverable, future releases of the 2016 Sysplex Extensions will provide full replacements for the two CICS SMS volumes (CICS01 and CICS02). If you want to add new CICS related data sets, we recommend that you set up new, private, volumes that are carried forward from one ADCD or Sysplex Extensions release to the next rather than placing the data sets in the CICFILES storage group.

For example, if you want to add your own application data files and load libraries, it is best to complete the following steps:

1. Set up a new user catalog.
2. Place that catalog on a private volume.
3. Create a high qualifier for the new data sets, define that high-level qualifier as an alias, and point it at your new user catalog.
4. Consider making all of those new data sets SMS-managed and place them in an SMS storage group *other than* CICFILES or DB2FILES. This method ensures that those data sets are not accidentally placed on a volume that is replaced when you move to the next release.







# A

## Sample definitions

This appendix provides sample definitions to help you get your sysplex up and running in a timely manner and with minimal manual effort. Each sample is described in the relevant section in this book.

## Sample devmap file

The devmap that is shown in Example A-1 contains definitions for the standard ADCD volumes (z/VM and z/OS) and the extra volumes that are provided or required by the zPDT 2016 Sysplex Extensions.

### *Example A-1 Sample devmap file*

---

```
#ADCD z/OS 2.1 May 2015 basic definition: TCP/IP QDIO definition
```

```
[system]
```

```
memory 7000m
```

```
3270port 3270
```

```
processors 1
```

```
command 2 x3270 localhost:3270
```

```
command 2 x3270 localhost:3270
```

```
command 2 x3270 localhost:3270
```

```
command 2 x3270 localhost:3270
```

```
command 2 x3270 localhost:3270
```

```
command 2 x3270 localhost:3270
```

```
command 2 x3270 localhost:3270
```

```
[manager]
```

```
name aws3274 1
```

```
device 0700 3279 3174 L700
```

```
device 0701 3279 3174 L701
```

```
device 0702 3279 3174 L702
```

```
device 0703 3279 3174 L703
```

```
device 0704 3279 3174 L704
```

```
device 0705 3279 3174 L705
```

```
device 0706 3279 3174 L706
```

```
device 0707 3279 3174 L707
```

```
device 0708 3279 3174 L708
```

```
device 0709 3279 3174 L709
```

```
device 070A 3279 3174 L70A
```

```
device 070B 3279 3174 L70B
```

```
device 070C 3279 3174 L70C
```

```
device 070D 3279 3174 L70D
```

```
device 070E 3279 3174 L70E
```

```
device 070F 3279 3174 L70F
```

```
device 0710 3279 3174 L710
```

```
device 0711 3279 3174 L711
```

```
device 0712 3279 3174 L712
```

```
device 0713 3279 3174 L713
```

```
device 0714 3279 3174 L714
```

```
device 0715 3279 3174 L715
```

```
device 0716 3279 3174 L716
```

```
device 0717 3279 3174 L717
```

```
device 0718 3279 3174 L718
```

```
device 0719 3279 3174 L719
```

```
device 071A 3279 3174 L71A
```

```
device 071B 3279 3174 L71B
```

```
device 071C 3279 3174 L71C
```

```
device 071D 3279 3174 L71D
```

```
device 071E 3279 3174 L71E
```

```
device 071F 3279 3174 L71F
```

```
[manager]
name awsckd 27
device 0200 3390 3990 M01RES
device 0201 3390 3990 630RL1
device 0202 3390 3990 630RL2
device 0203 3390 3990 M01W01
device 0204 3390 3990 M01S01
device 0205 3390 3990 M01P01
device 0206 3390 3990 VMCOM1
device 0207 3390 3990 VMCOM2
device 0208 3390 3990 M01W02
device 0209 3390 3990 M01W03
device 020A 3390 3990 VMPROD
```

```
[manager]
name awsckd 28
device 0a80 3390 3990 F1RES1
device 0a81 3390 3990 F1RES2
device 0a82 3390 3990 F1SYS1
device 0a83 3390 3990 F1CFG1
device 0a84 3390 3990 F1USS1
device 0a85 3390 3990 F1USS2
device 0a86 3390 3990 F1PAGA
device 0a87 3390 3990 F1PAGB
device 0a88 3390 3990 F1PAGC
device 0a89 3390 3990 F1PRD1
device 0a8a 3390 3990 F1PRD2
device 0a8b 3390 3990 F1PRD3
device 0a8c 3390 3990 F1DIS1
device 0a8d 3390 3990 F1DIS2
device 0a8e 3390 3990 SARES1
device 0a93 3390 3990 F1C521
device 0a97 3390 3990 F1DBB1
device 0a98 3390 3990 F1DBB2
device 0aa7 3390 3990 F1BBN1
device 0ab0 3390 3990 CF0001
device 0ab1 3390 3990 CICS01
device 0ab2 3390 3990 CICS02
device 0ab3 3390 3990 DB2001
device 0ab4 3390 3990 DB2002
device 0ab5 3390 3990 F1PAGX
device 0ab6 3390 3990 F1PAGY
device 0ab7 3390 3990 F1PAGZ
device 0ab8 3390 3990 F1SYS2
device 0ab9 3390 3990 WORK01
```

```
[manager]
name awsosa 0023 --path=A0 --pathtype=OSD --tunnel_intf=y
device 400 osa osa
device 401 osa osa
device 402 osa osa
device 408 osa osa
device 409 osa osa
device 40A osa osa
```

```

[manager]
name awsosa 0024 --path=f2 --pathtype=OSD
device 404 osa osa
device 405 osa osa
device 406 osa osa
device 40C osa osa
device 40D osa osa
device 40E osa osa

[manager]
name awsctc 100
device e40 3088 3088 ctc://192.168.1.99:4000/e41
device e41 3088 3088 ctc://192.168.1.99:4001/e40

```

---

The numbers on the name `awsckd` statements are random. They must be unique only within that devmap file.

Use the `zPDT find_io` command to determine the value to specify on the `path=` statement on the `awsosa` definitions. The `PATH` for the tunnel interface always appears to be `A0`, but the `PATH` for the network interface is impossible to predict. Therefore, you must check which is the active network interface on *your* PC and insert that value here.

## Sample z/VM Directory entries

The VM Directory statements in Example A-2, Example A-3 on page 116, Example A-4 on page 117, Example A-5 on page 119, Example A-6 on page 121, and Example A-7 on page 121 contain definitions for a VM user ID that acts as the owner of the z/OS volumes (MVSDUMMY), plus one non-sysplex guest (BASEAD), plus the two Parallel Sysplex systems (S0W1 and S0W2), and the two CFVMs (CFCC1 and CFCC2).

**Note:** All of these user IDs are defined in the ADCD-provided z/VM directory. They are *not* provided in the standard directory that is provided with z/VM.

*Example A-2 Our VM Directory statements for MVSDUMMY*

---

```

USER MVSDUMMY MVSDUMMY 1024M 1024M G
*****
*
* MVSDUMMY OWNS z/OS VOLUMES SO THEY CAN BE SHARED
* Addresses A9E and A9F are for coupling volumes
*
*****
MDISK 0A80 3390 DEVNO 0A80 MWV
MDISK 0A81 3390 DEVNO 0A81 MWV
MDISK 0A82 3390 DEVNO 0A82 MWV
MDISK 0A83 3390 DEVNO 0A83 MWV
MDISK 0A84 3390 DEVNO 0A84 MWV
MDISK 0A85 3390 DEVNO 0A85 MWV
MDISK 0A86 3390 DEVNO 0A86 MWV
MDISK 0A87 3390 DEVNO 0A87 MWV
MDISK 0A88 3390 DEVNO 0A88 MWV
MDISK 0A89 3390 DEVNO 0A89 MWV

```

MDISK 0A8A 3390 DEVNO 0A8A MWV  
 MDISK 0A8B 3390 DEVNO 0A8B MWV  
 MDISK 0A8C 3390 DEVNO 0A8C MWV  
 MDISK 0A8D 3390 DEVNO 0A8D MWV  
 MDISK 0A8E 3390 DEVNO 0A8E MWV  
 MDISK 0A8F 3390 DEVNO 0A8F MWV  
 MDISK 0A90 3390 DEVNO 0A90 MWV  
 MDISK 0A91 3390 DEVNO 0A91 MWV  
 MDISK 0A92 3390 DEVNO 0A92 MWV  
 MDISK 0A93 3390 DEVNO 0A93 MWV  
 MDISK 0A94 3390 DEVNO 0A94 MWV  
 MDISK 0A95 3390 DEVNO 0A95 MWV  
 MDISK 0A96 3390 DEVNO 0A96 MWV  
 MDISK 0A97 3390 DEVNO 0A97 MWV  
 MDISK 0A98 3390 DEVNO 0A98 MWV  
 MDISK 0A99 3390 DEVNO 0A99 MWV  
 MDISK 0A9A 3390 DEVNO 0A9A MWV  
 MDISK 0A9B 3390 DEVNO 0A9B MWV  
 MDISK 0A9C 3390 DEVNO 0A9C MWV  
 MDISK 0A9D 3390 DEVNO 0A9D MWV  
 MDISK 0A9E 3390 DEVNO 0A9E MWV  
 MDISK 0A9F 3390 DEVNO 0A9F MWV  
 MDISK 0AA0 3390 DEVNO 0AA0 MWV  
 MDISK 0AA1 3390 DEVNO 0AA1 MWV  
 MDISK 0AA2 3390 DEVNO 0AA2 MWV  
 MDISK 0AA3 3390 DEVNO 0AA3 MWV  
 MDISK 0AA4 3390 DEVNO 0AA4 MWV  
 MDISK 0AA5 3390 DEVNO 0AA5 MWV  
 MDISK 0AA6 3390 DEVNO 0AA6 MWV  
 MDISK 0AA7 3390 DEVNO 0AA7 MWV  
 MDISK 0AA8 3390 DEVNO 0AA8 MWV  
 MDISK 0AA9 3390 DEVNO 0AA9 MWV  
 MDISK 0AAA 3390 DEVNO 0AAA MWV  
 MDISK 0AAB 3390 DEVNO 0AAB MWV  
 MDISK 0AAC 3390 DEVNO 0AAC MWV  
 MDISK 0AAD 3390 DEVNO 0AAD MWV  
 MDISK 0AAE 3390 DEVNO 0AAE MWV  
 MDISK 0AAF 3390 DEVNO 0AAF MWV  
 MDISK 0AB0 3390 DEVNO 0AB0 MWV  
 DASDOPT WRKALLEG  
 MDISK 0AB1 3390 DEVNO 0AB1 MWV  
 MDISK 0AB2 3390 DEVNO 0AB2 MWV  
 MDISK 0AB3 3390 DEVNO 0AB3 MWV  
 MDISK 0AB4 3390 DEVNO 0AB4 MWV  
 MDISK 0AB5 3390 DEVNO 0AB5 MWV  
 MDISK 0AB6 3390 DEVNO 0AB6 MWV  
 MDISK 0AB7 3390 DEVNO 0AB7 MWV  
 MDISK 0AB8 3390 DEVNO 0AB8 MWV  
 MDISK 0AB9 3390 DEVNO 0AB9 MWV  
 MDISK 0ABA 3390 DEVNO 0ABA MWV  
 MDISK 0ABB 3390 DEVNO 0ABB MWV  
 MDISK 0ABC 3390 DEVNO 0ABC MWV  
 MDISK 0ABD 3390 DEVNO 0ABD MWV

```
MDISK 0ABE 3390 DEVNO 0ABE MWV
MDISK 0ABF 3390 DEVNO 0ABF MWV
```

---

Example A-3 contains the definition for the monoplex z/OS system.

*Example A-3 Our VM Directory statements for BASEAD*

---

```
USER BASEAD BASEAD 4000M 4000M G
*****
*
* The BASEAD guest allows IPLing z/OS in non-sysplex mode
*
*
*****
CPU 0
MACH ESA
DEDICATE 0400 0400
DEDICATE 0401 0401
DEDICATE 0402 0402
CONSOLE 0700 3215 T
SPECIAL 0701 3270
SPECIAL 0702 3270
SPECIAL 0703 3270
SPECIAL 0704 3270
SPECIAL 0705 3270
SPECIAL 0706 3270
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
LINK MVSDUMMY 0A80 0A80 MW
LINK MVSDUMMY 0A81 0A81 MW
LINK MVSDUMMY 0A82 0A82 MW
LINK MVSDUMMY 0A83 0A83 MW
LINK MVSDUMMY 0A84 0A84 MW
LINK MVSDUMMY 0A85 0A85 MW
LINK MVSDUMMY 0A86 0A86 MW
LINK MVSDUMMY 0A87 0A87 MW
LINK MVSDUMMY 0A88 0A88 MW
LINK MVSDUMMY 0A89 0A89 MW
LINK MVSDUMMY 0A8A 0A8A MW
LINK MVSDUMMY 0A8B 0A8B MW
LINK MVSDUMMY 0A8C 0A8C MW
LINK MVSDUMMY 0A8D 0A8D MW
LINK MVSDUMMY 0A8E 0A8E MW
LINK MVSDUMMY 0A8F 0A8F MW
LINK MVSDUMMY 0A90 0A90 MW
LINK MVSDUMMY 0A91 0A91 MW
LINK MVSDUMMY 0A92 0A92 MW
LINK MVSDUMMY 0A93 0A93 MW
LINK MVSDUMMY 0A94 0A94 MW
LINK MVSDUMMY 0A95 0A95 MW
LINK MVSDUMMY 0A96 0A96 MW
LINK MVSDUMMY 0A97 0A97 MW
```

```

LINK MVSDUMMY 0A98 0A98 MW
LINK MVSDUMMY 0A99 0A99 MW
LINK MVSDUMMY 0A9A 0A9A MW
LINK MVSDUMMY 0A9B 0A9B MW
LINK MVSDUMMY 0A9C 0A9C MW
LINK MVSDUMMY 0A9D 0A9D MW
LINK MVSDUMMY 0A9E 0A9E MW
LINK MVSDUMMY 0A9F 0A9F MW
LINK MVSDUMMY 0AA0 0AA0 MW
LINK MVSDUMMY 0AA1 0AA1 MW
LINK MVSDUMMY 0AA2 0AA2 MW
LINK MVSDUMMY 0AA3 0AA3 MW
LINK MVSDUMMY 0AA4 0AA4 MW
LINK MVSDUMMY 0AA5 0AA5 MW
LINK MVSDUMMY 0AA6 0AA6 MW
LINK MVSDUMMY 0AA7 0AA7 MW
LINK MVSDUMMY 0AA8 0AA8 MW
LINK MVSDUMMY 0AA9 0AA9 MW
LINK MVSDUMMY 0AAA 0AAA MW
LINK MVSDUMMY 0AAB 0AAB MW
LINK MVSDUMMY 0AAC 0AAC MW
LINK MVSDUMMY 0AAD 0AAD MW
LINK MVSDUMMY 0AAE 0AAE MW
LINK MVSDUMMY 0AAF 0AAF MW
LINK MVSDUMMY 0AB0 0AB0 MW
LINK MVSDUMMY 0AB1 0AB1 MW
LINK MVSDUMMY 0AB2 0AB2 MW
LINK MVSDUMMY 0AB3 0AB3 MW
LINK MVSDUMMY 0AB4 0AB4 MW
LINK MVSDUMMY 0AB5 0AB5 MW
LINK MVSDUMMY 0AB6 0AB6 MW
LINK MVSDUMMY 0AB7 0AB7 MW
LINK MVSDUMMY 0AB8 0AB8 MW
LINK MVSDUMMY 0AB9 0AB9 MW
LINK MVSDUMMY 0ABA 0ABA MW
LINK MVSDUMMY 0ABB 0ABB MW
LINK MVSDUMMY 0ABC 0ABC MW
LINK MVSDUMMY 0ABD 0ABD MW
LINK MVSDUMMY 0ABE 0ABE MW
LINK MVSDUMMY 0ABF 0ABF MW

```

---

Example A-4 contains the definition for the first z/OS system in the sysplex.

*Example A-4 Our VM Directory statements for S0W1*

---

|                                          |          |
|------------------------------------------|----------|
| USER S0W1 S0W1 5000M 5000M G             | 12051039 |
| *****                                    | 12051039 |
| *                                        | 12051039 |
| * The S0W1 guest is the "1" SYSPLEX z/OS | 12051039 |
| *                                        | 12051039 |
| *****                                    | 12051039 |
| CPU 0                                    | 12051039 |
| MACH ESA                                 | 12051039 |
| OPTION CFUSER TODENABLE                  | 12051039 |
| DEDICATE 0400 0400                       | 12051039 |
| DEDICATE 0401 0401                       | 12051039 |

|                            |          |
|----------------------------|----------|
| DEDICATE 0402 0402         | 12051039 |
| DEDICATE 0404 0404         | 12051039 |
| DEDICATE 0405 0405         | 12051039 |
| DEDICATE 0406 0406         | 12051039 |
| CONSOLE 0700 3215 T        | 12051039 |
| SPECIAL 1400 MSGP CFCC1    | 12051039 |
| SPECIAL 1500 MSGP CFCC2    | 12051039 |
| SPECIAL 0701 3270          | 12051039 |
| SPECIAL 0702 3270          | 12051039 |
| SPECIAL 0703 3270          | 12051039 |
| SPECIAL 0704 3270          | 12051039 |
| SPECIAL 0705 3270          | 12051039 |
| SPECIAL 0706 3270          | 12051039 |
| SPOOL 000C 2540 READER *   | 12051039 |
| SPOOL 000D 2540 PUNCH A    | 12051039 |
| SPOOL 000E 1403 A          | 12051039 |
| LINK MAINT 0190 0190 RR    | 12051039 |
| LINK MAINT 019D 019D RR    | 12051039 |
| LINK MAINT 019E 019E RR    | 12051039 |
| LINK MVSDUMMY 0A80 0A80 MW | 12051039 |
| LINK MVSDUMMY 0A81 0A81 MW | 12051039 |
| LINK MVSDUMMY 0A82 0A82 MW | 12051039 |
| LINK MVSDUMMY 0A83 0A83 MW | 12051039 |
| LINK MVSDUMMY 0A84 0A84 MW | 12051039 |
| LINK MVSDUMMY 0A85 0A85 MW | 12051039 |
| LINK MVSDUMMY 0A86 0A86 MW | 12051039 |
| LINK MVSDUMMY 0A87 0A87 MW | 12051039 |
| LINK MVSDUMMY 0A88 0A88 MW | 12051039 |
| LINK MVSDUMMY 0A89 0A89 MW | 12051039 |
| LINK MVSDUMMY 0A8A 0A8A MW | 12051039 |
| LINK MVSDUMMY 0A8B 0A8B MW | 12051039 |
| LINK MVSDUMMY 0A8C 0A8C MW | 12051039 |
| LINK MVSDUMMY 0A8D 0A8D MW | 12051039 |
| LINK MVSDUMMY 0A8E 0A8E MW | 12051039 |
| LINK MVSDUMMY 0A8F 0A8F MW | 12051039 |
| LINK MVSDUMMY 0A90 0A90 MW | 12051039 |
| LINK MVSDUMMY 0A91 0A91 MW | 12051039 |
| LINK MVSDUMMY 0A92 0A92 MW | 12051039 |
| LINK MVSDUMMY 0A93 0A93 MW | 12051039 |
| LINK MVSDUMMY 0A94 0A94 MW | 12051039 |
| LINK MVSDUMMY 0A95 0A95 MW | 12051039 |
| LINK MVSDUMMY 0A96 0A96 MW | 12051039 |
| LINK MVSDUMMY 0A97 0A97 MW | 12051039 |
| LINK MVSDUMMY 0A98 0A98 MW | 12051039 |
| LINK MVSDUMMY 0A99 0A99 MW | 12051039 |
| LINK MVSDUMMY 0A9A 0A9A MW | 12051039 |
| LINK MVSDUMMY 0A9B 0A9B MW | 12051039 |
| LINK MVSDUMMY 0A9C 0A9C MW | 12051039 |
| LINK MVSDUMMY 0A9D 0A9D MW | 12051039 |
| LINK MVSDUMMY 0A9E 0A9E MW | 12051039 |
| LINK MVSDUMMY 0A9F 0A9F MW | 12051039 |
| LINK MVSDUMMY 0AA0 0AA0 MW | 12051039 |
| LINK MVSDUMMY 0AA1 0AA1 MW | 12051039 |
| LINK MVSDUMMY 0AA2 0AA2 MW | 12051039 |
| LINK MVSDUMMY 0AA3 0AA3 MW | 12051039 |



|                            |          |
|----------------------------|----------|
| LINK MVSDUMMY 0AA4 0AA4 MW | 12051039 |
| LINK MVSDUMMY 0AA5 0AA5 MW | 12051039 |
| LINK MVSDUMMY 0AA6 0AA6 MW | 12051039 |
| LINK MVSDUMMY 0AA7 0AA7 MW | 12051039 |
| LINK MVSDUMMY 0AA8 0AA8 MW | 12051039 |
| LINK MVSDUMMY 0AA9 0AA9 MW | 12051039 |
| LINK MVSDUMMY 0AAA 0AAA MW | 12051039 |
| LINK MVSDUMMY 0AAB 0AAB MW | 12051039 |
| LINK MVSDUMMY 0AAC 0AAC MW | 12051039 |
| LINK MVSDUMMY 0AAD 0AAD MW | 12051039 |
| LINK MVSDUMMY 0AAE 0AAE MW | 12051039 |
| LINK MVSDUMMY 0AAF 0AAF MW | 12051039 |
| LINK MVSDUMMY 0AB0 0AB0 MW | 12051039 |
| LINK MVSDUMMY 0AB1 0AB1 MW | 12051039 |
| LINK MVSDUMMY 0AB2 0AB2 MW | 12051039 |
| LINK MVSDUMMY 0AB3 0AB3 MW | 12051039 |
| LINK MVSDUMMY 0AB4 0AB4 MW | 12051039 |
| LINK MVSDUMMY 0AB5 0AB5 MW | 12051039 |
| LINK MVSDUMMY 0AB6 0AB6 MW | 12051039 |
| LINK MVSDUMMY 0AB7 0AB7 MW | 12051039 |
| LINK MVSDUMMY 0AB8 0AB8 MW | 12051039 |
| LINK MVSDUMMY 0AB9 0AB9 MW | 12051039 |
| LINK MVSDUMMY 0ABA 0ABA MW | 12051039 |
| LINK MVSDUMMY 0ABB 0ABB MW | 12051039 |
| LINK MVSDUMMY 0ABC 0ABC MW | 12051039 |
| LINK MVSDUMMY 0ABD 0ABD MW | 12051039 |
| LINK MVSDUMMY 0ABE 0ABE MW | 12051039 |
| LINK MVSDUMMY 0ABF 0ABF MW | 12051039 |

---

Example A-5 contains the definition for the second z/OS system in the sysplex.

*Example A-5 Our VM Directory statements for S0W2*

|                                            |          |
|--------------------------------------------|----------|
| UUSER S0W2 S0W2 5000M 5000M G              | 12051040 |
| *****                                      | 12051040 |
| *                                          | 12051040 |
| * The S0W2 guest is the sysplex "2" system | 12051040 |
| *                                          | 12051040 |
| *****                                      | 12051040 |
| CPU 0                                      | 12051040 |
| MACH ESA                                   | 12051040 |
| OPTION CFUSER TODENABLE                    | 12051040 |
| DEDICATE 0408 0408                         | 12051040 |
| DEDICATE 0409 0408                         | 12051040 |
| DEDICATE 040A 040A                         | 12051040 |
| DEDICATE 040C 040C                         | 12051040 |
| DEDICATE 040D 040D                         | 12051040 |
| DEDICATE 040E 040E                         | 12051040 |
| CONSOLE 0700 3215 T                        | 12051040 |
| SPECIAL 1400 MSGP CFCC1                    | 12051040 |
| SPECIAL 1500 MSGP CFCC2                    | 12051040 |
| SPECIAL 0701 3270                          | 12051040 |
| SPECIAL 0702 3270                          | 12051040 |
| SPECIAL 0703 3270                          | 12051040 |
| SPECIAL 0704 3270                          | 12051040 |
| SPECIAL 0705 3270                          | 12051040 |

|                            |          |
|----------------------------|----------|
| SPECIAL 0706 3270          | 12051040 |
| SPOOL 000C 2540 READER *   | 12051040 |
| SPOOL 000D 2540 PUNCH A    | 12051040 |
| SPOOL 000E 1403 A          | 12051040 |
| LINK MAINT 0190 0190 RR    | 12051040 |
| LINK MAINT 019D 019D RR    | 12051040 |
| LINK MAINT 019E 019E RR    | 12051040 |
| LINK MVSDUMMY 0A80 0A80 MW | 12051040 |
| LINK MVSDUMMY 0A81 0A81 MW | 12051040 |
| LINK MVSDUMMY 0A82 0A82 MW | 12051040 |
| LINK MVSDUMMY 0A83 0A83 MW | 12051040 |
| LINK MVSDUMMY 0A84 0A84 MW | 12051040 |
| LINK MVSDUMMY 0A85 0A85 MW | 12051040 |
| LINK MVSDUMMY 0A86 0A86 MW | 12051040 |
| LINK MVSDUMMY 0A87 0A87 MW | 12051040 |
| LINK MVSDUMMY 0A88 0A88 MW | 12051040 |
| LINK MVSDUMMY 0A89 0A89 MW | 12051040 |
| LINK MVSDUMMY 0A8A 0A8A MW | 12051040 |
| LINK MVSDUMMY 0A8B 0A8B MW | 12051040 |
| LINK MVSDUMMY 0A8C 0A8C MW | 12051040 |
| LINK MVSDUMMY 0A8D 0A8D MW | 12051040 |
| LINK MVSDUMMY 0A8E 0A8E MW | 12051040 |
| LINK MVSDUMMY 0A8F 0A8F MW | 12051040 |
| LINK MVSDUMMY 0A90 0A90 MW | 12051040 |
| LINK MVSDUMMY 0A91 0A91 MW | 12051040 |
| LINK MVSDUMMY 0A92 0A92 MW | 12051040 |
| LINK MVSDUMMY 0A93 0A93 MW | 12051040 |
| LINK MVSDUMMY 0A94 0A94 MW | 12051040 |
| LINK MVSDUMMY 0A95 0A95 MW | 12051040 |
| LINK MVSDUMMY 0A96 0A96 MW | 12051040 |
| LINK MVSDUMMY 0A97 0A97 MW | 12051040 |
| LINK MVSDUMMY 0A98 0A98 MW | 12051040 |
| LINK MVSDUMMY 0A99 0A99 MW | 12051040 |
| LINK MVSDUMMY 0A9A 0A9A MW | 12051040 |
| LINK MVSDUMMY 0A9B 0A9B MW | 12051040 |
| LINK MVSDUMMY 0A9C 0A9C MW | 12051040 |
| LINK MVSDUMMY 0A9D 0A9D MW | 12051040 |
| LINK MVSDUMMY 0A9E 0A9E MW | 12051040 |
| LINK MVSDUMMY 0A9F 0A9F MW | 12051040 |
| LINK MVSDUMMY 0AA0 0AA0 MW | 12051040 |
| LINK MVSDUMMY 0AA1 0AA1 MW | 12051040 |
| LINK MVSDUMMY 0AA2 0AA2 MW | 12051040 |
| LINK MVSDUMMY 0AA3 0AA3 MW | 12051040 |
| LINK MVSDUMMY 0AA4 0AA4 MW | 12051040 |
| LINK MVSDUMMY 0AA5 0AA5 MW | 12051040 |
| LINK MVSDUMMY 0AA6 0AA6 MW | 12051040 |
| LINK MVSDUMMY 0AA7 0AA7 MW | 12051040 |
| LINK MVSDUMMY 0AA8 0AA8 MW | 12051040 |
| LINK MVSDUMMY 0AA9 0AA9 MW | 12051040 |
| LINK MVSDUMMY 0AAA 0AAA MW | 12051040 |
| LINK MVSDUMMY 0AAB 0AAB MW | 12051040 |
| LINK MVSDUMMY 0AAC 0AAC MW | 12051040 |
| LINK MVSDUMMY 0AAD 0AAD MW | 12051040 |
| LINK MVSDUMMY 0AAE 0AAE MW | 12051040 |
| LINK MVSDUMMY 0AAF 0AAF MW | 12051040 |

|                            |          |
|----------------------------|----------|
| LINK MVSDUMMY OAB0 OAB0 MW | 12051040 |
| LINK MVSDUMMY OAB1 OAB1 MW | 12051040 |
| LINK MVSDUMMY OAB2 OAB2 MW | 12051040 |
| LINK MVSDUMMY OAB3 OAB3 MW | 12051040 |
| LINK MVSDUMMY OAB4 OAB4 MW | 12051040 |
| LINK MVSDUMMY OAB5 OAB5 MW | 12051040 |
| LINK MVSDUMMY OAB6 OAB6 MW | 12051040 |
| LINK MVSDUMMY OAB7 OAB7 MW | 12051040 |
| LINK MVSDUMMY OAB8 OAB8 MW | 12051040 |
| LINK MVSDUMMY OAB9 OAB9 MW | 12051040 |
| LINK MVSDUMMY OABA OABA MW | 12051040 |
| LINK MVSDUMMY OABB OABB MW | 12051040 |
| LINK MVSDUMMY OABC OABC MW | 12051040 |
| LINK MVSDUMMY OABD OABD MW | 12051040 |
| LINK MVSDUMMY OABE OABE MW | 12051040 |
| LINK MVSDUMMY OABF OABF MW | 12051040 |

---

Example A-6 contains the definition for the first coupling facility (CF) virtual machine. The SPECIAL 1600 MSGP statement defines a virtual coupling link between CFCC1 and CFCC2, which is required if you want to use System Managed Duplexing between the two CFs.

*Example A-6 Our VM Directory statements for CFCC1*

---

```

USER CFCC1 CFCC1 1200M 1200M G
*****
*
* CFCC1 is the first coupling facility
*
*****
MACH ESA
OPTION CFVM TODENABLE
XAUTOLOG CFCONSOL
CONSOLE 0009 3215 T CFCONSOL
SPECIAL 1600 MSGP CFCC2

```

---

Example A-7 contains the definition for the second coupling facility virtual machine. The SPECIAL 1600 MSGP statement defines the CFCC2 end of the virtual coupling link to CFCC1.

*Example A-7 Our VM Directory statements for CFCC2*

---

```

USER CFCC2 CFCC2 1200M 1200M G
*****
*
* CFCC2 is the first coupling facility
*
*****
MACH ESA
OPTION CFVM TODENABLE
XAUTOLOG CFCONSOL
CONSOLE 0009 3215 T CFCONSOL
SPECIAL 1600 MSGP CFCC1

```

---

**Tip:** When we added the CF-to-CF links to our CFs, we had to shut down the entire sysplex, including logging off the CF virtual machines, in order to get the virtual links between the two CFs to work. This issue might be unique to our environment, but performing a restart in this manner might be faster in the end than trying to add the links dynamically.

## Devices that are defined in the ADCD-provided IODF

Table A-1 lists the device numbers and device types that are defined in the ADCD-provided input/output definition file (IODF).

*Table A-1 Devices defined in z/OS IODF*

| Device number | Device type |
|---------------|-------------|
| 00C           | 2540        |
| 00E           | 1403        |
| 00F           | 1403        |
| 0120-015F     | 3380        |
| 0300-0318     | 3390        |
| 0400-040F     | OSA         |
| 0550-055F     | 3400        |
| 0560-056F     | 3480        |
| 0580-058F     | 3490        |
| 0590-059F     | 3590        |
| 0600-060F     | 3390        |
| 0700-073F     | 3277        |
| 0900-091F     | 3277        |
| 0A80-0AFF     | 3390        |
| 0E20-0E23     | CTC         |
| 0E40-0E43     | CTC         |
| 1A00-1AFF     | 3390        |
| 2A00-2AFF     | 3390        |
| 3A00-3AFF     | 3390        |

## Sample IEASYMxx member for sysplex

The ADCD-provided IEASYM00 member is designed for a single-system environment. Because there are some values in a sysplex that are system-specific (such as the names of the page data sets), we created an IEASYMPS member that sets system-specific system symbols that are based on the virtual machine user ID as an example of how a single IEASYMxx member can support many systems. Our IEASYMPS member is shown in Example A-8.

*Example A-8 2016 Sysplex Extensions provided IEASYMPS member*

---

```
SYSDEF  SYMDEF(&UNIXVER='Z21F')
        SYMDEF(&CTSLEVEL='520')
        SYMDEF(&SYSVER='Z21F')
        SYMDEF(&SYSP1.='F1PRD1')
        SYMDEF(&SYSP2.='F1PRD2')
        SYMDEF(&SYSP3.='F1PRD3')
        SYMDEF(&SYSR2.='F1RES2')
        SYMDEF(&SYSS1.='F1SYS1')
        SYMDEF(&SYSC1.='F1CFG1')
        SYMDEF(&SYSNUM='&SYSNAME(4:1)')
        SYMDEF(&VTAMAPPL='VTAMPS')
        SYMDEF(&VTAMID='&SYSNAME(4:1)')
        SYMDEF(&TRLNAM='OSATRL&VTAMID.')
        SYSPARM(PS) /* CONCATENATES IEASYSPTS TO WHATEVER IS */
                   /* SPECIFIED ON SYSPARM IN LOADXX MEMBER */

SYSDEF  VMUSERID(SOW1)
        SYSNAME(SOW1)
        SYSCONE(1A)
        SYMDEF(&ADCDLVL='ADCD21F')
        SYMDEF(&VTAMSA='6')

SYSDEF  VMUSERID(SOW2)
        SYSNAME(SOW2)
        SYSCONE(1B)
        SYMDEF(&ADCDLVL='BDCD21F')
        SYMDEF(&VTAMSA='7')
```

---

We also provide two members for base sysplex mode: one for running under z/VM (called IEASYMBS), and one for running the sysplex across two PCs (called IEASYMST). The two base sysplex members are nearly identical to the Parallel Sysplex version, except that they concatenate another IEASYSxx member and the IEASYMST member filters on the Linux user ID rather than on the VM user name.

## Sample JCL to allocate system-specific data sets

Although the design point of a sysplex is that as much as possible should be shared, there are some data sets that must be unique to each system. The VSAMSOW2 member in the SYSPLEX.PARALLEL.CNTL data set allocates the system-specific data sets that are required by system SOW2. The JCL that is contained in VSAMSOW2 is shown in Example A-9 on page 124.

*Example A-9 Job VSAMS0W2 JCL*

---

```
//VSAMS0W2 JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID.
//S1      EXEC  PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DD1     DD VOL=SER=F1PAGX,UNIT=3390,DISP=SHR
//DD2     DD VOL=SER=F1PAGY,UNIT=3390,DISP=SHR
//DD3     DD VOL=SER=F1PAGZ,UNIT=3390,DISP=SHR
//SYSIN   DD *
DEFINE PGSPC(NAME(SYS1.S0W2.PLPA.PAGE) -
  FILE(DD1) CYLINDERS(100) VOLUME(F1PAGX))
DEFINE PGSPC(NAME(SYS1.S0W2.COMMON.PAGE) -
  FILE(DD1) CYLINDERS(50) VOLUME(F1PAGX))
DEFINE PGSPC(NAME(SYS1.S0W2.LOCALA.PAGE) -
  FILE(DD1) CYLINDERS(9000) VOLUME(F1PAGX))
DEFINE PGSPC(NAME(SYS1.S0W2.LOCALB.PAGE) -
  FILE(DD2) CYLINDERS(9000) VOLUME(F1PAGY))
DEFINE PGSPC(NAME(SYS1.S0W2.LOCALC.PAGE) -
  FILE(DD3) CYLINDERS(9000) VOLUME(F1PAGZ))
/*
//S2      EXEC  PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DD1     DD VOL=SER=F1SYS2,UNIT=3390,DISP=SHR
//SYSIN   DD *
DEFINE CLUSTER(NAME(SYS1.S0W2.MAN1) REUSE NIXD SPND SPEED SHR(2) -
  VOLUME(F1SYS2) CYLINDERS(40) RECSZ(4086,32767) CNVSZ(4096))
DEFINE CLUSTER(NAME(SYS1.S0W2.MAN2) REUSE NIXD SPND SPEED SHR(2) -
  VOLUME(F1SYS2) CYLINDERS(40) RECSZ(4086,32767) CNVSZ(4096))
DEFINE CL(NAME(SYS1.S0W2.STGINDEX) VOLUME(F1SYS2) CYLINDERS(10) -
  KEYS(12 8) BUFSPC(20480) RECSZ(2041 2041) FILE(DD1) REUSE )-
  DATA (CNVSZ(2048)) INDEX (CNVSZ(4096))
/*
//S3      EXEC  PGM=IFCDIP00
//SERERDS DD DSN=SYS1.S0W2.LOGREC,UNIT=3390,VOL=SER=F1SYS2,
//          DISP=(NEW,CATLG),SPACE=(TRK,(90),,CONTIG)
//SYSPRINT DD SYSOUT=*
/*
```

---

Note the use of the system name as the second qualifier in all of the data set names. By using this convention, you can have a single Parmlib member that is shared by all systems in the sysplex. For example, the following IEASYSxx statements define unique page data sets for each system:

```
PAGE=(SYS1.&SYSNAME..PLPA.PAGE,
      SYS1.&SYSNAME..COMMON.PAGE,
      SYS1.&SYSNAME..LOCALA.PAGE,
      SYS1.&SYSNAME..LOCALB.PAGE,L),
```

The &SYSNAME is replaced by the system name when this member is read at IPL time. As a result, on system S0W1, these statements define page data sets that are called SYS1.S0W1.PLPA.PAGE, SYS1.S0W1.COMMON.PAGE, and so on. On system S0W2, the same statements define data sets SYS1.S0W2.PLPA.PAGE, SYS1.S0W2.COMMON.PAGE, and so on. This configuration reduces the number of unique definitions that must be maintained, promotes good naming conventions, simplifies system operation, and reduces the time and effort required to add a new system to the sysplex.

## Job to create UNIX System Services data sets for second system

Certain UNIX System Services data sets can be shared by all systems in the sysplex. However, there are a subset of data sets that must exist for each system. This job creates a copy of those data sets for the SOW2 system and places them on the F1SYS2 volume, as shown in Example A-10.

*Example A-10 Cloning system-specific zFS data sets*

---

```
//OMVSCON JOB 1,OGDEN,MSGCLASS=X,REGION=OK,NOTIFY=&SYSUID.
//*
/* COPY SYSTEM-SPECIFIC ZFS DATASETS
/*
//CLONE EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//FROMDASD DD UNIT=3390,DISP=SHR,VOL=SER=F1USS2
//TODASD DD UNIT=3390,DISP=SHR,VOL=SER=F1SYS2
//SYSIN DD *
COPY DATASET(INCLUDE(ZFS.ADCD21F.DEV, -
                     ZFS.ADCD21F.VAR, -
                     ZFS.ADCD21F.ETC, -
                     ZFS.ADCD21F.SYSTEM, -
                     ZFS.ADCD21F.VARWBEM, -
                     ZFS.ADCD21F.WEB, -
                     ZFS.ADCD21F.WEB.HTTPD1.CONFIG, -
                     ZFS.ADCD21F.WEB.HTTPD1.V53.CONFIG, -
                     ZFS.ADCD21F.USERS, -
                     ZFS.ADCD21F.USR.MAIL, -
                     ZFS.ADCD21F.TMP )) -
LOGINDDNAME(FROMDASD) OUTDD(TODASD) -
TOLERATE(ENQF) -
RENAMEU((ZFS.ADCD21F.DEV,ZFS.BDCD21F.DEV), -
        (ZFS.ADCD21F.VAR,ZFS.BDCD21F.VAR), -
        (ZFS.ADCD21F.ETC,ZFS.BDCD21F.ETC), -
        (ZFS.ADCD21F.SYSTEM,ZFS.BDCD21F.SYSTEM), -
        (ZFS.ADCD21F.VARWBEM,ZFS.BDCD21F.VARWBEM), -
        (ZFS.ADCD21F.WEB,ZFS.BDCD21F.WEB), -
        (ZFS.ADCD21F.WEB.HTTPD1.CONFIG, -
         ZFS.BDCD21F.WEB.HTTPD1.CONFIG), -
        (ZFS.ADCD21F.WEB.HTTPD1.V53.CONFIG, -
         ZFS.BDCD21F.WEB.HTTPD1.V53.CONFIG), -
        (ZFS.ADCD21F.USERS,ZFS.BDCD21F.USERS), -
        (ZFS.ADCD21F.USR.MAIL,ZFS.BDCD21F.USR.MAIL), -
        (ZFS.ADCD21F.TMP,ZFS.BDCD21F.TMP)) -
SHARE ALLEXCP ALLDATA(*) CANCELERROR CATALOG /*
```

---

## zPDT Disk versioning sample script

The sample Linux script that is shown in Example A-11 demonstrates how you enable disk versioning for your z/VM and z/OS disks. This example is only an example; you must customize it to include all of your z/OS and z/VM disks (or, at least, all those that you want to restore to a known restart point). Do not forget that you must run the **chmod 755** command to grant yourself authority to run the script.

*Example A-11 Script to enable zPDT disk versioning*

---

```
#!/bin/bash
# Script to ENable disk versioning for ADCD volumes

alckd F1BBN1 -ve
alckd F1C521 -ve
alckd F1CFG1 -ve
alckd F1DBAR -ve
alckd F1DBB1 -ve
alckd F1DBB2 -ve
alckd F1DIS1 -ve
alckd F1DIS2 -ve
alckd F1PAGA -ve
alckd F1PAGB -ve
alckd F1PAGC -ve
alckd F1PRD1 -ve
alckd F1PRD2 -ve
alckd F1PRD3 -ve
alckd F1RES1 -ve
alckd F1RES2 -ve
alckd F1SYS1 -ve
alckd F1USS1 -ve
alckd F1USS2 -ve

alckd CF0001 -ve
alckd CICS01 -ve
alckd CICS02 -ve
alckd DB2001 -ve
alckd DB2002 -ve
alckd F1PAGX -ve
alckd F1PAGY -ve
alckd F1PAGZ -ve
alckd F1SYS2 -ve
alckd WORK01 -ve

alckd M01P01 -ve
alckd M01RES -ve
alckd M01S01 -ve
alckd M01W01 -ve
alckd M01W02 -ve
alckd M01W03 -ve
alckd VMCOM1 -ve
alckd VMCOM2 -ve
alckd VMPROD -ve
alckd 630RL1 -ve
alckd 630RL2 -ve
```

---





## Sysplex couple data sets and policies

The standard ADCD deliverable includes several couple data sets (CDSs). The 2016 Sysplex Extensions provides more CDSs and makes some changes to the ADCD-provided sets. This appendix describes the differences between what is delivered by ADCD and the 2016 Sysplex Extensions and includes the following topics:

- ▶ “ADCD CDSs” on page 128
- ▶ “2016 Sysplex Extensions CDSs for Parallel Sysplex” on page 128
- ▶ “Job to create CDSs” on page 130
- ▶ “2016 Sysplex Extensions-provided policies” on page 133

## ADCD CDSs

The ADCD system is delivered with the CDSs shown in Table B-1. The level of support is basic, as is fitting for a monoplex aimed at single-user environments.

*Table B-1 ADCD-provided CDSs*

| CDS type | Policy                                   | Notes                                                                                                 |
|----------|------------------------------------------|-------------------------------------------------------------------------------------------------------|
| BPXMCDS  | N/A                                      | Up to date                                                                                            |
| CFRM     | No policies provided in shipped CFRM CDS | CDS format is back-level                                                                              |
| LOGR     | Only one policy supported.               | Only RRS log streams are defined in the supplied CDS. Format level of the supplied CDS is back-level. |
| WLM      | ETPBASE                                  |                                                                                                       |
| Sysplex  | N/A                                      | CDS format is back-level                                                                              |

As you see in the Notes column, some of the CDSs are back-level. The format of most CDSs is determined based on the functions or attributes that you specify when you create the data set by using the IXCL1DSU utility. Because the level of sysplex usage in the standard ADCD monoplex is minimal, the CDSs are not formatted to support the most recent enhancements.

## 2016 Sysplex Extensions CDSs for Parallel Sysplex

Because of the 2016 Sysplex Extensions' dual role of delivering a working data sharing Parallel Sysplex and also acting as a platform to demonstrate the benefits of sysplex, the Parallel Sysplex CDSs that are delivered with this package contain much more information than the standard ADCD CDSs and are also formatted to support more functions. The CDSs that are delivered with the 2016 Sysplex Extensions are listed in Table B-2.

*Table B-2 2016 Sysplex Extensions-provided Parallel Sysplex CDSs*

| CDS type | Policy                     | Notes                                                                             |
|----------|----------------------------|-----------------------------------------------------------------------------------|
| ARM      |                            | Latest format level<br>Updated to support longer system symbol names in z/OS 2.2. |
| BPXMCDS  | N/A                        | Up to date                                                                        |
| CFRM     | CFRM1                      | Latest format level                                                               |
| LOGR     | Only one policy supported. | Includes CICS, RRS, Logrec. OPERLOG, SMF, and z/OS Health Checker.                |
| SFM      | SFM1                       | All in line with IBM Best Practices                                               |
| WLM      | ETPBASE                    | Standard ADCD policy                                                              |
| Sysplex  | N/A                        | Latest format level                                                               |

For more information about CDS format levels, see the IBM Redbooks document *System z Parallel Sysplex Best Practices*, SG24-7817. For information about how to create CDSs and control the functions that they support, see *z/OS MVS Setting Up a Sysplex*, SA23-1399.

To determine the format information about most CDSs, run a **D XCF,C,TYPE=cdstype** command. For example, the response to a **D XCF,C,TYPE=SYSPLEX** command is shown in Example B-1. In this example, the highlighted lines show that the CDS was formatted to support the System Status Detection Protocol.

*Example B-1 Displaying the format of a CDS*

---

```

D XCF,C,TYPE=SYSPLEX
IXC358I 02.30.02 DISPLAY XCF 691
SYSPLEX COUPLE DATA SETS
PRIMARY DSN: PARALLEL.ADCDPL.XCF.CDS01
VOLSER: CF0001 DEVN: 0A8B
FORMAT TOD MAXSYSTEM MAXGROUP(PEAK) MAXMEMBER(PEAK)
12/22/2014 18:00:23 4 48 (33) 35 (11)
ADDITIONAL INFORMATION:
ALL TYPES OF COUPLE DATA SETS ARE SUPPORTED
GRS STAR MODE IS SUPPORTED
SYSTEM STATUS DETECTION PROTOCOL IS SUPPORTED
ALTERNATE DSN: PARALLEL.ADCDPL.XCF.CDS02
VOLSER: CF0001 DEVN: 0A8B
FORMAT TOD MAXSYSTEM MAXGROUP MAXMEMBER
12/22/2014 18:00:23 4 48 35
ADDITIONAL INFORMATION:
ALL TYPES OF COUPLE DATA SETS ARE SUPPORTED
GRS STAR MODE IS SUPPORTED
SYSTEM STATUS DETECTION PROTOCOL IS SUPPORTED

```

---

For CDSs that support multiple policies, you can determine the name of the currently active policy by running the **D XCF,POL,TYPE=cdstype** command. An example for the CFRM CDS is shown in Example B-2.

*Example B-2 Displaying the name of the in-use policy*

---

```

RESPONSE=SOW1
IXC364I 02.31.58 DISPLAY XCF 693
TYPE: CFRM
POLNAME: CFRM1
STARTED: 12/01/2015 10:31:07
LAST UPDATED: 12/01/2015 10:30:54

```

---

The source of the policies that are delivered in the 2016 Sysplex Extensions Parallel Sysplex CDSs is contained in members called POLaaa in data set SYSPLEX.PARALLEL.CNTL. You can also print the contents of the active policy at any time by using the following process:

1. Run the **D XCF,POL,TYPE=cdstype** command. This command produces the name of the in-use policy.
2. The SYSPLEX.PARALLEL.CNTL data set contains a set of members called LSTxxx. Select the set that corresponds to the CDS that you are interested in and submit that job. Some CDSs support multiple policies, so the corresponding LSTxxx job prints *all* of the policies in that CDS. When reviewing the output, ensure that you select the output for the in-use policy.

## Job to create CDSs

The job that was used to create the CDSs on the CF0001 volume is shown in Example B-3. If you want to create your own CDSs, use this job as a starting point.

*Example B-3 Job to create CDSs*

---

```
//DEFCDSS JOB 1,OGDEN,MSGCLASS=X,REGION=40M
//DEFCDS EXEC PGM=IXCL1DSU
//STEPLIB DD DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD DATA,DLM='##'

/* SYSPLEX COUPLE DATASETS */

DEFINEDS SYSPLEX(ADCDPL)
  MAXSYSTEM(4)
  DSN(PARALLEL.ADCDPL.XCF.CDS01) VOLSER(CF0001)
  CATALOG
  DATA TYPE(SYSPLEX)
  ITEM NAME(GRS) NUMBER(1)
  ITEM NAME(GROUP) NUMBER(48)
  ITEM NAME(MEMBER) NUMBER(32)
  ITEM NAME(SSTATDET) NUMBER (1)

DEFINEDS SYSPLEX(ADCDPL)
  MAXSYSTEM(4)
  DSN(PARALLEL.ADCDPL.XCF.CDS02) VOLSER(CF0001)
  CATALOG
  DATA TYPE(SYSPLEX)
  ITEM NAME(GRS) NUMBER(1)
  ITEM NAME(GROUP) NUMBER(48)
  ITEM NAME(MEMBER) NUMBER(32)
  ITEM NAME(SSTATDET) NUMBER (1)

/* ARM COUPLE DATASETS */

DEFINEDS SYSPLEX(ADCDPL)
  MAXSYSTEM(4)
  DSN(PARALLEL.ADCDPL.ARM.CDS01) VOLSER(CF0001)
  CATALOG
  DATA TYPE(ARM)
  ITEM NAME(POLICY) NUMBER(4)
  ITEM NAME(MAXELEM) NUMBER(50)
  ITEM NAME(TOTELEM) NUMBER(200)

DEFINEDS SYSPLEX(ADCDPL)
  MAXSYSTEM(4)
  DSN(PARALLEL.ADCDPL.ARM.CDS02) VOLSER(CF0001)
  CATALOG
  DATA TYPE(ARM)
  ITEM NAME(POLICY) NUMBER(4)
  ITEM NAME(MAXELEM) NUMBER(50)
  ITEM NAME(TOTELEM) NUMBER(200)
```

```

/*    CFRM COUPLE DATASETS    */

DEFINEDS SYSPLEX(ADCDPL)
    MAXSYSTEM(4)
    DSN(PARALLEL.ADCDPL.CFRM.CDS01) VOLSER(CF0001)
    CATALOG
    DATA TYPE(CFRM)
    ITEM NAME(POLICY) NUMBER(4)
    ITEM NAME(CF) NUMBER(4)
    ITEM NAME(STR) NUMBER(100)
    ITEM NAME(CONNECT) NUMBER(8)
    ITEM NAME(SMREBLD) NUMBER(1)
    ITEM NAME(SMDUPLEX) NUMBER(1)
    ITEM NAME(MSGBASED) NUMBER (1)

DEFINEDS SYSPLEX(ADCDPL)
    MAXSYSTEM(4)
    DSN(PARALLEL.ADCDPL.CFRM.CDS02) VOLSER(CF0001)
    CATALOG
    DATA TYPE(CFRM)
    ITEM NAME(POLICY) NUMBER(4)
    ITEM NAME(CF) NUMBER(4)
    ITEM NAME(STR) NUMBER(100)
    ITEM NAME(CONNECT) NUMBER(8)
    ITEM NAME(SMREBLD) NUMBER(1)
    ITEM NAME(SMDUPLEX) NUMBER(1)
    ITEM NAME(MSGBASED) NUMBER (1)

/*    OMVS COUPLE DATASETS    */

DEFINEDS SYSPLEX(ADCDPL)
    MAXSYSTEM(4)
    DSN(PARALLEL.ADCDPL.OMVS.CDS01) VOLSER(CF0001)
    CATALOG
    DATA TYPE(BPXMCDs)
    ITEM NAME(MOUNTS) NUMBER(100)
    ITEM NAME(AMTRULES) NUMBER(50)

DEFINEDS SYSPLEX(ADCDPL)
    MAXSYSTEM(4)
    DSN(PARALLEL.ADCDPL.OMVS.CDS02) VOLSER(CF0001)
    CATALOG
    DATA TYPE(BPXMCDs)
    ITEM NAME(MOUNTS) NUMBER(100)
    ITEM NAME(AMTRULES) NUMBER(50)

/*    LOGR COUPLE DATASETS    */

DEFINEDS SYSPLEX(ADCDPL)
    MAXSYSTEM(4)
    DSN(PARALLEL.ADCDPL.LOGR.CDS01) VOLSER(CF0001)
    CATALOG
    DATA TYPE(LOGR)
    ITEM NAME(LSR) NUMBER(200)
    ITEM NAME(LSTRR) NUMBER(100)

```

```

ITEM NAME(DSEXTENT) NUMBER(20)
ITEM NAME(SMDUPLEX) NUMBER(1)

DEFINEDS SYSPLEX(ADCDPL)
MAXSYSTEM(4)
DSN(PARALLEL.ADCDPL.LOGR.CDS02) VOLSER(CF0001)
CATALOG
DATA TYPE(LOGR)
ITEM NAME(LSR) NUMBER(200)
ITEM NAME(LSTRR) NUMBER(100)
ITEM NAME(DSEXTENT) NUMBER(20)
ITEM NAME(SMDUPLEX) NUMBER(1)

/*      SFM COUPLE DATASETS      */

DEFINEDS SYSPLEX(ADCDPL)
MAXSYSTEM(4)
DSN(PARALLEL.ADCDPL.SFM.CDS01) VOLSER(CF0001)
CATALOG
DATA TYPE(SFM)
ITEM NAME(POLICY) NUMBER(10)
ITEM NAME(SYSTEM) NUMBER(4)
ITEM NAME(RECONFIG) NUMBER(10)

DEFINEDS SYSPLEX(ADCDPL)
MAXSYSTEM(4)
DSN(PARALLEL.ADCDPL.SFM.CDS02) VOLSER(CF0001)
CATALOG
DATA TYPE(SFM)
ITEM NAME(POLICY) NUMBER(10)
ITEM NAME(SYSTEM) NUMBER(4)
ITEM NAME(RECONFIG) NUMBER(10)

/*      WLM COUPLE DATASETS      */

DEFINEDS SYSPLEX(ADCDPL)
MAXSYSTEM(4)
DSN(PARALLEL.ADCDPL.WLM.CDS01) VOLSER(CF0001)
CATALOG
DATA TYPE(WLM)
ITEM NAME(POLICY) NUMBER(10)
ITEM NAME(WORKLOAD) NUMBER(35)
ITEM NAME(SRVCLASS) NUMBER(200)
ITEM NAME(APPLENV) NUMBER(200)
ITEM NAME(SCHENV) NUMBER(100)
ITEM NAME(SVDEFEXT) NUMBER(100)
ITEM NAME(SVDCREXT) NUMBER(100)
ITEM NAME(SVAEAEXT) NUMBER(100)
ITEM NAME(SVSEAEXT) NUMBER(100)

DEFINEDS SYSPLEX(ADCDPL)
MAXSYSTEM(4)
DSN(PARALLEL.ADCDPL.WLM.CDS02) VOLSER(CF0001)
CATALOG
DATA TYPE(WLM)

```

```

ITEM NAME(POLICY) NUMBER(10)
ITEM NAME(WORKLOAD) NUMBER(35)
ITEM NAME(SRVCLASS) NUMBER(200)
ITEM NAME(APPLENV) NUMBER(200)
ITEM NAME(SCHENV) NUMBER(100)
ITEM NAME(SVDEFEXT) NUMBER(100)
ITEM NAME(SVDCREXT) NUMBER(100)
ITEM NAME(SVAEAEXT) NUMBER(100)
ITEM NAME(SVSEAEXT) NUMBER(100)

```

##

**Note:** If you are running the z/OS 2.2 level of 2016 Sysplex Extensions, two versions of the ARM CDSs are provided:

- ▶ The PARALLEL.ADCDPL.ARM.CDS01/02 version is formatted at the z/OS 2.1 level. In a z/OS 2.1 system, the COUPLEPS member points at this data set.
- ▶ The PARALLEL.ADCDPL.ARM22.CDS01/02 version is formatted at the z/OS 2.2 level. In a z/OS 2.2 system, the COUPLEPS member points at this data set.

## 2016 Sysplex Extensions-provided policies

The sysplex policies that are provided by the Sysplex Extensions are available in the SYSPLEX.PARALLEL.CNTL data set. For more information about the member names for each policy, see in 4.7.4, “Sysplex policies” on page 87. We provide a copy of each policy here purely for reference.

### ARM policy

The ARM policy is shown in Example B-4

*Example B-4 ARM policy*

```

//POLARM1 JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID.
//ARMPOL EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
DATA TYPE(ARM) REPORT(YES)
DEFINE POLICY NAME(ARM1) REPLACE(YES)

/* ALL RESTART GROUPS THAT DO NOT SPECIFY TARGET_SYSTEM
CAN BE RESTARTED ON ANY SYSTEM IN THE PLEX IN THE SAME
JES2 MAS */

RESTART_GROUP(*)
TARGET_SYSTEM(*)

/* THE FOLLOWING RESTARTS CICS */

RESTART_GROUP(CICSPDT)
ELEMENT(SYSCICS_CICS1A11)
TERMTYPE(ELEMTERM)
ELEMENT(SYSCICS_CICS1A12)

```

```

        TERMTYPE(ELEMTerm)
    ELEMENT(SYSCICS_CICS2A11)
        TERMTYPE(ELEMTerm)
    ELEMENT(SYSCICS_CICS2A12)
        TERMTYPE(ELEMTerm)
    ELEMENT(SYSCICS_CICS1T11)
        TERMTYPE(ELEMTerm)
    ELEMENT(SYSCICS_CICS1T12)
        TERMTYPE(ELEMTerm)
    ELEMENT(SYSCICS_CICS2T11)
        TERMTYPE(ELEMTerm)
    ELEMENT(SYSCICS_CICS2T12)
        TERMTYPE(ELEMTerm)
    ELEMENT(SYSCICS_CICSWE11)
        TERMTYPE(ELEMTerm)
    ELEMENT(SYSCICS_CMAS)
        TERMTYPE(ELEMTerm)
/* THE FOLLOWING RESTARTS DB2 */

RESTART_GROUP(DB2DS1)
    ELEMENT(DSNDPDGDPD1)
    ELEMENT(DXRDPDGDJD1001)

RESTART_GROUP(DB2DS2)
    ELEMENT(DSNDPDGDPD2)
    ELEMENT(DXRDPDGDJD2002)

/* NO OTHER ARM ELEMENTS WILL BE RESTARTED */

RESTART_GROUP(DEFAULT)
    ELEMENT(*)
        RESTART_ATTEMPTS(0)
/*

```

---

## CFRM policy

This version of the zPDT 2016 Sysplex Extensions contains definitions for more CF structures than previous releases. The extra structures are required if you want to use the new functions that we included in this version.

The CFRM policy is loaded in the CFRM CDS that is provided on the CF0001 volume; however, we include it in Example B-5 for reference. This policy is a copy of the POLCFRM1 member from the SYSPLEX.PARALLEL.CNTL data set.

### *Example B-5 CFRM policy*

---

```

//POLCFRM1 JOB 1,OGDEN,MSGCLASS=X,NOTIFY=&SYSUID
//CFRMPOL EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
DATA TYPE(CFRM) REPORT(YES)
DEFINE POLICY NAME(CFRM1) REPLACE(YES)

CF NAME(CFCC1)

```



```

TYPE(SIMDEV)
MFG(IBM)
PLANT(EN)
SEQUENCE(0000000CFCC1)
PARTITION(0)
CPCID(00)
DUMPSPACE(10000)

CF NAME(CFCC2)
TYPE(SIMDEV)
MFG(IBM)
PLANT(EN)
SEQUENCE(0000000CFCC2)
PARTITION(0)
CPCID(00)
DUMPSPACE(10000)

/*
/* ALL STRUCTURE SIZES IN THIS POLICY ARE BASED ON A 4-WAY
/* SYSPLEX AND THE LOW REQUEST RATES THAT YOU WOULD EXPECT
/* IN A ZPDT ENVIRONMENT.
/* FOR ACCURATE SIZES FOR YOUR ENVIRONMENT USE THE CFSIZER
/* AVAILABLE ON THE WEB AT:
/*   HTTP://WWW-947.IBM.COM/SYSTEMS/SUPPORT/Z/CFSIZER/
/*
/* DEFINE XCF TRANSPORT CLASS STRUCTURES
STRUCTURE NAME(IXC_DEFAULT_1)
INITSIZE(10M)
SIZE(15M)
PREFLIST(CFCC1,CFCC2)

STRUCTURE NAME(IXC_DEFAULT_2)
INITSIZE(10M)
SIZE(15M)
PREFLIST(CFCC2,CFCC1)

STRUCTURE NAME(IXC_DEF8K_1)
INITSIZE(15M)
SIZE(22M)
PREFLIST(CFCC1,CFCC2)

STRUCTURE NAME(IXC_DEF8K_2)
INITSIZE(15M)
SIZE(22M)
PREFLIST(CFCC2,CFCC1)

STRUCTURE NAME(IXC_DEF61K_1)
INITSIZE(25M)
SIZE(40M)
PREFLIST(CFCC1,CFCC2)

STRUCTURE NAME(IXC_DEF61K_2)
INITSIZE(25M)
SIZE(40M)

```

```

        PREFLIST(CFCC2,CFCC1)

/* DEFINE OPERLOG STRUCTURE                                */

        STRUCTURE NAME(SYSTEM_OPERLOG)
        INITSIZE(10M)
        SIZE(15M)
        PREFLIST(CFCC2,CFCC1)
        FULLTHRESHOLD(90)

/* DEFINE LOGREC STRUCTURE                                  */

        STRUCTURE NAME(SYSTEM_LOGREC)
        INITSIZE(10M)
        SIZE(15M)
        PREFLIST(CFCC1,CFCC2)

/* DEFINE HEALTH CHECKER STRUCTURE                          */

        STRUCTURE NAME(HZS_HEALTHCHKLOG)
        INITSIZE(10M)
        SIZE(15M)
        PREFLIST(CFCC2,CFCC1)

/* DEFINE GRS STAR STRUCTURE                                */

        STRUCTURE NAME(ISGLOCK)
        SIZE(17M)
        PREFLIST(CFCC1,CFCC2)

/* DEFINE WLM ENCLAVES STRUCTURE                            */

        STRUCTURE NAME(SYSZWLM_WORKUNIT)
        INITSIZE(10M)
        SIZE(15M)
        PREFLIST(CFCC2,CFCC1)

/* DEFINE RRS LOGSTREAM STRUCTURES                          */

        STRUCTURE NAME(RRS_ARCHIVE_1)
        INITSIZE(20M)
        SIZE(30M)
        PREFLIST(CFCC1,CFCC2)

        STRUCTURE NAME(RRS_RMDATA_1)
        INITSIZE(20M)
        SIZE(30M)
        PREFLIST(CFCC1,CFCC2)

        STRUCTURE NAME(RRS_MAINUR_1)
        INITSIZE(20M)
        SIZE(30M)
        PREFLIST(CFCC2,CFCC1)

        STRUCTURE NAME(RRS_DELAYEDUR_1)

```

```

    INITSIZE(20M)
    SIZE(30M)
    PREFLIST(CFCC1,CFCC2)

    STRUCTURE NAME(RRS_RESTART_1)
    INITSIZE(20M)
    SIZE(30M)
    PREFLIST(CFCC2,CFCC1)

    STRUCTURE NAME(RRS_RM_META_1)
    INITSIZE(20M)
    SIZE(30M)
    PREFLIST(CFCC2,CFCC1)

/* DEFINE SMF LOGSTREAM STRUCTURE                                */

    STRUCTURE NAME(IFASMF_GENERAL)
    INITSIZE(100M)
    SIZE(200M)
    PREFLIST(CFCC1,CFCC2)

/* DEFINE JES2 CHECKPOINT STRUCTURE                                */

    STRUCTURE NAME(JES_CKPT_1)
    INITSIZE(12M)
    SIZE(18M)
    PREFLIST(CFCC1,CFCC2)

/* DEFINE HEALTH CHECKER LOGSTREAM STRUCTURE                      */

    STRUCTURE NAME(HZS_HLTHCHKER)
    INITSIZE(10M)
    SIZE(15M)
    PREFLIST(CFCC1,CFCC2)

/* DEFINE RACF STRUCTURE  */

    STRUCTURE NAME(IRRXCFO0_P001)
    INITSIZE(10M)
    SIZE(15M)
    PREFLIST(CFCC1,CFCC2)

    STRUCTURE NAME(IRRXCFO0_B001)
    INITSIZE(10M)
    SIZE(15M)
    PREFLIST(CFCC1,CFCC2)

/* DEFINE VTAM GR STRUCTURE                                       */

    STRUCTURE NAME(ISTGENERIC)
    INITSIZE(10M)
    SIZE(15M)
    PREFLIST(CFCC1,CFCC2)

```

```

/* DEFINE CICS LOG STREAM STRUCTURES */

STRUCTURE NAME(CIC_DFHLOG_AOR1)
  INITSIZE(10M)
  SIZE(15M)
  PREFLIST(CFCC1,CFCC2)

STRUCTURE NAME(CIC_DFHLOG_AOR2)
  INITSIZE(10M)
  SIZE(15M)
  PREFLIST(CFCC2,CFCC1)

STRUCTURE NAME(CIC_DFHLOG_TOR1)
  INITSIZE(10M)
  SIZE(15M)
  PREFLIST(CFCC1,CFCC2)

STRUCTURE NAME(CIC_DFHLOG_TOR2)
  INITSIZE(10M)
  SIZE(15M)
  PREFLIST(CFCC2,CFCC1)

STRUCTURE NAME(DFHCFLS_PDTCFDT1)
  INITSIZE(10M)
  SIZE(15M)
  PREFLIST(CFCC1,CFCC2)

STRUCTURE NAME(DFHNCLS_PDTCNCS1)
  INITSIZE(10M)
  SIZE(15M)
  PREFLIST(CFCC2,CFCC1)

STRUCTURE NAME(DFHXQLS_PDTSTOR1)
  INITSIZE(10M)
  SIZE(15M)
  PREFLIST(CFCC1,CFCC2)

/* DEFINE DB2 STRUCTURES */

STRUCTURE NAME(DSNDPDG_LOCK1)
  INITSIZE(33M)
  SIZE(60M)
  PREFLIST(CFCC1,CFCC2)

STRUCTURE NAME(DSNDPDG_SCA)
  INITSIZE(16M)
  SIZE(32M)
  PREFLIST(CFCC2,CFCC1)

STRUCTURE NAME(DSNDPDG_GBPO)
  INITSIZE(30M)
  SIZE(45M)
  DUPLEX(ENABLED)
  PREFLIST(CFCC1,CFCC2)

```

```

STRUCTURE NAME(DSNDPDG_GBP1)
INITSIZE(30M)
SIZE(45M)
DUPLEX(ENABLED)
PREFLIST(CFCC2,CFCC1)

STRUCTURE NAME(DSNDPDG_GBP2)
INITSIZE(30M)
SIZE(45M)
DUPLEX(ENABLED)
PREFLIST(CFCC1,CFCC2)
/* The GBP8K0 structure is used by buffer pool 3 (BP3) */
STRUCTURE NAME(DSNDPDG_GBP8K0)
INITSIZE(30M)
SIZE(45M)
DUPLEX(ENABLED)
PREFLIST(CFCC1,CFCC2)

/*

```

---

## SFM policy

The SFM policy is shown in Example B-6.

### *Example B-6 SFM policy*

---

```

//POLSFMI JOB 1,OGDEN,MSGCLASS=X
//SFMPOL EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
DATA TYPE(SFM) REPORT(YES)
DSN(PARALLEL.ADCDPL.SFM.CDS01)
DEFINE POLICY NAME(SFM1) REPLACE(YES)

CONNFAIL(YES)
SYSTEM NAME(*)
CFSTRHANGTIME(900)
ISOLATETIME(0)
MEMSTALLTIME(300)
SSUMLIMIT(720)
SYSTEM NAME(SOW1)
WEIGHT(80)
SYSTEM NAME(SOW2)
WEIGHT(20)

```

---





## **Alternative disk configuration**

The Network File System (NFS) file sharing configuration that is described in 4.6.3, “Sharing zPDT DASD across PCs” on page 82 does not deliver response times that you often experience if all of the systems that are accessing the disk were in the same PC. If the performance effect of our sample configuration is not acceptable, this appendix describes a more robust alternative.

## Alternative configuration for base sysplex

**Note:** This configuration is appropriate for a base sysplex configuration only. zPDT does not support a Parallel Sysplex that spans more than one PC, meaning that there is no need for a disk sharing solution in a zPDT Parallel Sysplex environment.

We built our base sysplex with two Linux PCs, as shown in Figure 4-4 on page 80. This configuration is probably the simplest configuration possible and we use it for descriptive purposes. However, the performance of this configuration might not suit many users. An alternative base sysplex configuration is shown in Figure C-1. This configuration involves a separate “device” for network-attached storage (NAS).

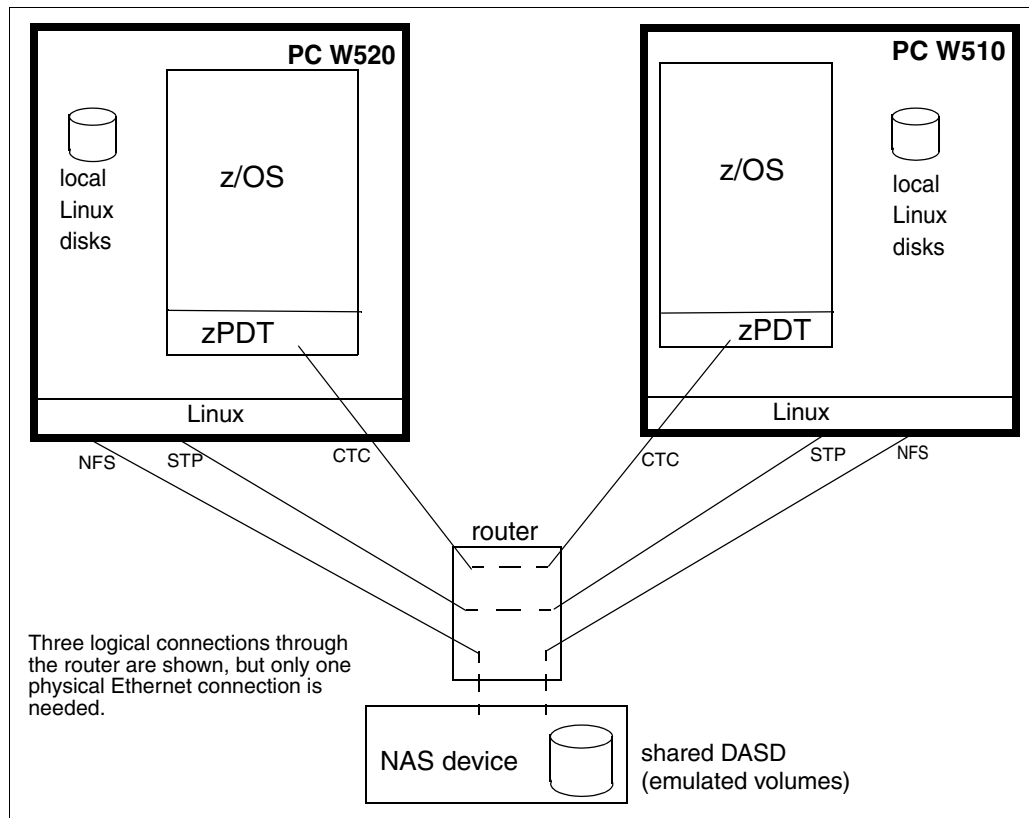


Figure C-1 NAS storage for base sysplex

We tested two NAS devices. One was a more sophisticated (and more expensive) unit with four drives (2 TB each) and four network interfaces (of which we used a single interface). The other NAS device was a much lower-priced unit with one 2 TB drive and one network interface. In both cases, our LAN connections were through 1 Gb switches or routers. All our emulated 3390 volumes were placed on the NAS device. All of the normal Linux files (including the zPDT programs, devmaps, and so forth) were retained on the separate PC systems.

There was a noticeable difference in performance between the more expensive and the less expensive NAS devices. The more expensive unit supported NFSv4. The less expensive unit supported only the older NFS operation.



We connected three base sysplex z/OS systems by using the more expensive NAS device. Performance was fairly good, although not as good as with locally resident PC disks. With the less expensive NAS device, we connected two base sysplex z/OS systems and performance was acceptable for many purposes. Performing an initial program load (IPL) shows the most noticeable slowdown, especially with the less expensive NAS device. When z/OS was ready, the systems were usable for our particular tasks.

These configurations involve an extra expense (the NAS device, with disk drives) and the extra complexity of integrating it into your configuration.<sup>1</sup> Many different NAS devices are available from multiple vendors. You must determine which, if any, meet your performance requirements.

Some common sense rules, such as not loading multiple z/OS systems within seconds of each other, can be helpful. The disk performance of zPDT is largely dependent on the effectiveness of the local Linux disk cache in each Linux machine. Operations that “drag” large numbers of modules from a remotely mounted disk, such as during IPL or the first TSO logon, can be considerably slower than later operations that benefit from having modules and data in the local cache.

The Linux and zPDT setup for using NAS storage is almost the same as for our base sysplex implementation as described in 4.6.3, “Sharing zPDT DASD across PCs” on page 82.

As their name implies, NAS devices are network-attached. If your network is the Internet, you must consider security and performance implications. Most vendor products offer VPN operation for better security, although a VPN might be slightly more difficult to configure. The local networks we used provided response times in the 0.15 millisecond range for simple **ping** operations.

---

<sup>1</sup> We noticed that many NAS devices are marketed as “cloud” solutions for Microsoft Windows systems and the setup process is optimized for this environment. Setup in a pure Linux environment can be more of a challenge in these cases, even though the NAS device often is built on a special-purpose Linux.





# D

## **Sample IPL flow for sysplex under z/VM**

If you are not familiar with the use of z/VM or with running z/OS as a guest under z/VM, the information in this appendix might prove helpful.

## Alternative configuration for base sysplex

If you are not familiar with z/VM or with running z/OS under z/VM, this example of the commands that we issued to start both systems under z/VM might be helpful. This example is based on the use of two Linux workspaces (named Linux1 and Linux 2 in this example) that we described in 4.7.1, “Managing your x3270 sessions” on page 86. The parenthesis at the beginning of each of the following lines indicate which window is used for the command.

```
(Linux 1) $ awsstart devmap1 (start zPDT)
(Linux 1) $ x3270 -port 3270 localhost & (starts 3270 as 700)
(Linux 1) $ x3270 -port 3270 localhost & (starts 3270 as 701)
(Linux 1) $ x3270 -port 3270 localhost & (starts 3270 as 702)
(Linux 2) $ x3270 -port 3270 localhost & (starts 3270 as 703)
(Linux 2) $ x3270 -port 3270 localhost & (starts 3270 as 704)
      (Position the windows as shown in Figure 4-5 on page 86)
      (Check for zPDT “License Obtained” messages)
(Linux 1) $ ipl 0200 parm 0700 (IPL z/VM)
      (See “z/VM startup comments” below about the z/VM stand-alone loader.)
      (Once z/VM starts you may need to clear the screen several
      times until normal z/VM startup is complete.)
(700) DISC (disconnect VM operator)

(702) logon as user CFCONSOL (password CFCONSOL)
(702) XAUTOLOG CFCC1 (start first CF)
(702) XAUTOLOG CFCC2 (start second CF)
(702) (Clear screen, as needed)

(700) logon to z/VM as user S0W1 (password S0W1)
(700) TERM CONMODE 3270
(700) ipl 0a80 loadparm 0a82ps (IPL S0W1 system)
      (Wait for z/OS messages. IPLing z/OS under z/VM is
      slower than when running natively under zPDT. BE PATIENT!)
(700) (Respond, as usual, to z/OS messages)
      (Wait until z/OS startup processing is complete)
(701) Command==> DIAL S0W1 (connect for TSO session)
(701) (The VTAM logon screen should appear)
(701)) logon ibmuser (logon to TSO)
(703) logon to z/VM as user S0W2 (password S0W2)
(703) TERM CONMODE 3270
(703) ipl 0a80 loadparm 0a82ps (IPL S0W2 system)
      (Wait for z/OS messages. Be patient.)
(703) (Respond, as usual, to z/OS messages)
      (Wait until z/OS startup is complete)
(704) Command==> DIAL S0W2 (connect for TSO session)
(704) (The VTAM logon screen should appear)
(704) logon adcdmst (logon to TSO)
```

## z/VM startup comments

Depending on your z/VM configuration, you might see the Stand Alone Program Loader when you load z/VM, as shown in Figure D-1.

```
STAND ALONE PROGRAM LOADER: z/VM VERSION x RELEASE 6.3

DEVICE NUMBER: 0200  MINIDISK OFFSET: 000000000  EXTENT: 1

MODULE NAME:  CPLOAD  LOAD ORIGIN: 10000

----- IPL PARAMETERS -----
fn=system ft=config pdnum=1 pdvol=0206 cons=0700
.
.
.
.
9= FILELIST 10= LOAD 11= TOGGLE EXTENT/OFFSET
```

*Figure D-1 Stand Alone Loader panel*

If the `pdvol` address (for the VMCOM1 volume) and the `cons` address (for the operator 3270 session) are correct, press PF10. You can type over the IPL parameters, if necessary.

We suggest you do not cold start z/VM unless you have a specific reason for doing so. If the z/VM logo does not appear on your 3270 sessions, enter an **ENABLE ALL** command in the z/VM OPERATOR session.





## **Installation instructions for z/OS 2.2 base**

This appendix is equivalent to Chapter 4, “Installing the 2016 Sysplex Extensions” on page 35, but it was updated to reflect an installation that is based on the Application Developers Controlled Distributions (ADCD) December 2015 z/OS 2.2 deliverable.

# Implementing a sysplex under zPDT

We described in Chapter 3, “Sysplex configurations” on page 17 the reasons why you might want to use a sysplex under zPDT and several of the benefits of sysplex. This chapter helps you combine your ADCD z/OS 2.2 monoplex system with the Sysplex Extensions package to create your multisystem sysplex.

The following configurations are supported:

- ▶ Parallel Sysplex that is running under IBM z/VM®
- ▶ Base sysplex that is running under z/VM
- ▶ Base sysplex that is running “native” under z/PDT, spread over two personal computers

We assume that most people that are interested in running a sysplex under zPDT want a Parallel Sysplex. Therefore, the primary installation path covers all of the required steps to turn your ADCD system into a two-way Parallel Sysplex.

However, that path also caters for most of the steps that are required for a base sysplex. Therefore, all users that are interested in any form of sysplex under zPDT need to follow the installation steps that are described in “Implementation overview” on page 150 and “Setup steps” on page 154.

If your objective is to have a base sysplex, you also need to complete the steps that are described in “Other steps for base sysplex under z/VM” on page 192 or in “Implementing a base sysplex without z/VM” on page 195, as appropriate.

## Implementation overview

One of our objectives with this package was to make its installation as easy as possible, while also delivering significant sysplex-unique functions. It is not intended as a vehicle to help you learn how to set up a sysplex; rather, it is intended as a tool to help you create a data sharing Parallel Sysplex environment quickly and with as little specialist knowledge as possible. We also wanted to make it easy for you to switch back to a standard ADCD environment easily if you wanted.

Therefore, we wanted to make the package as self-contained as possible and minimize the number of changes that you must make to your environment to get this sysplex up and running. This approach also contributes to the objective of making it easier to revert to a standard ADCD environment.

**Note:** For more information about zPDT and ADCD, see *IBM zPDT Guide and Reference: System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

We strongly advise that you review and are familiar with the contents of that book before you start the implementation of the 2016 Sysplex Extensions.



Before you start the installation, an overview of the start-to-end process is valuable. The implementation steps are summarized in this section. For more information about each step, see “Setup steps” on page 154.

The following list is a summary of the tasks that will be performed to install the 2016 Sysplex Extensions:

1. Install the base ADCD z/OS system.  
You must be familiar with the operation of a z/OS system before you move to a sysplex configuration. If you do not have an ADCD system up and running, you must implement a standard ADCD monoplex system.

**Note:** This chapter covers the basic z/OS environment only. The implementation of the DB2 data sharing group is described in Chapter 5, “Sample DB2 data sharing environment” on page 95 and the CICSplex environment implementation is described in Chapter 6, “Sample CICSplex” on page 103.

2. Install z/VM if your objective is to create a Parallel Sysplex or a base sysplex that is running under z/VM. To gain experience with running z/OS under z/VM, you can run your ADCD system under z/VM before you move to sysplex mode.
3. Download the volumes that are provided as part of the zPDT 2016 Sysplex Extensions package. The volume serial numbers, virtual address that we used for them, and a brief description of the contents of each volume are listed in Table E-1.

Table E-1 Volumes that are delivered by zPDT 2016 Sysplex Extensions package

| Volser         | Device number | Use                                                                                                                                                       |
|----------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| CF0001         | AB0           | Couple data sets (CDSs) for Base and Parallel Sysplex. PDSs that contain sample JCL, parmlib, Proclib, and other required members.                        |
| CICS01, CICS02 | AB1 AB2       | Storage management subsystem (SMS)-managed volumes that contain data sets that are required by the CICSplex subsystems that are provided by this package. |
| DB2001, DB2002 | AB3 AB4       | SMS-managed volumes that contain the data sets that are required by the DB2 data sharing group that is provided by this package.                          |

In addition to the volumes that you download, several volumes must be created, initialized, and populated with system-specific data sets. Those volumes are listed in Table E-2. In this step, you will use the zPDT **a1cckd** command to create the Linux data sets that contain the z/OS volumes. In a later step, the volumes are initialized, and then the data sets are allocated on those volumes.

Table E-2 Volumes that must be created and initialized

| Volser                     | Device number     | Use                                            |
|----------------------------|-------------------|------------------------------------------------|
| A2PAGX<br>A2PAGY<br>A2PAGZ | AB5<br>AB6<br>AB7 | Page data sets for S0W2 system.                |
| A2SYS2                     | AB8               | System-specific data sets for the S0W2 system. |
| WORK01                     | AB9               | Work volume.                                   |

**Tip:** We occasionally encountered HCPPGT401I and HCPPGT400I messages on the VM OPERATOR console when we started all of the CICS address spaces on both systems. These messages are precursors to z/VM PGT004 wait states that are caused by lack of paging space.

If you experience PGT004 wait states in z/VM, you might need more paging volumes in z/VM. The z/VM IBM Knowledge Center contains the information that is needed to help you update z/VM to define the new paging volumes.

4. Before you change your working system, it is a good idea to create a known restart point. For more information about creating that point, see “Establish a known restart point” on page 158.
5. After you download and create the 2016 Sysplex Extensions volumes, update your zPDT device map (devmap) file to add the new DASD volumes. We also suggest adding commands to your devmap file to automatically start more x3270 sessions. Because you are running z/VM and another z/OS instance, you likely will need more 3270 sessions than you have now. You also must set up the Open Systems Adapter (OSA) definitions that will be used to communicate between the z/OS guests and Linux, and between the z/OS guests and the rest of your network.
6. The Parallel Sysplex environment that we provide with this package uses two z/OS virtual machines that are called S0W1 and S0W2, and two coupling facility virtual machines (CFVMs) that are called CFCC1 and CFCC2. All of these virtual machines are included in the VM directory that is provided with the ADCD z/VM package. A small change to the VM directory is required to identify the volumes that contain the CDSs.
7. When you use the zPDT `alckkd` command to allocate the new DASD volumes, the system creates Linux files that are internally formatted to resemble a count key data (CKD) DASD device. However, at this stage, they do not have a volser; the device resembles a new disk that is not yet initialized. In this step, you use Device Support Facilities (ICKDSF) to initialize the volumes that are listed in Table E-2 on page 151.
8. Most of the data sets on the volumes that you download are cataloged in user catalogs. To make those catalogs and the data sets that they reference available to your systems (in monoplex mode and when you run in sysplex mode), you must IMPORT CONNECT the new user catalogs and create ALIASes for the data sets in those catalogs.
9. Create the system-specific system data sets (Paging, VIO, SMF, and LOGREC) for system S0W2.
10. The ADCD naming convention of the paging data sets does not support a sysplex environment and the use of system symbols. They also use the entire volume, which leaves no room to allocate new data sets with more sysplex-appropriate names. Therefore, in this step, we delete the ADCD-provided page data sets and replace them with smaller data sets that use the same names (for compatibility purposes), and a set that uses the new naming convention.
11. Clone zFS system-specific file systems for the second (S0W2) system.
12. Create a mount point for S0W2 system and modify the BPXPRMxx member to use mount points and attributes that are more suitable for a sysplex environment.
13. Starting with z/OS 2.1, the z/OS Health Checker starts automatically. It also uses a data set to carry information from one IPL forward to the next. In this step, we allocate the S0W2 copy of that data set to avoid a JCL ERROR when the Health Checker starts on S0W2.

14. The 2016 Sysplex Extensions package provides a new set of CDSs to use in place of the CDSs that are provided by the standard ADCD deliverable. Although CDSs that are not cataloged in the master catalog can be used, this option is not the preferred option. In this step, we recatalog the 2016 Sysplex Extensions-provided CDSs in your master catalog.
15. The ADCD-provided SMS configuration knows about only the S0W1 system and is not aware of the sysplex name. In this step, we configure SMS (by using the Interactive Storage Management Facility (ISMF) panels) to recognize the sysplex and the second system name (S0W2). You also must define two data classes for System Logger (LOGR) data sets and one for DB2, a new storage class for CICS and DB2 runtime data sets (logs, journals, and so on), and new storage groups for the CICS and DB2 volumes. You also must update the automatic class selection (ACS) routines to assign the appropriate SMS objects to each data set.
16. The JES2 parms must be altered to define a Multi-Access Spool (MAS) system with two members (S0W1 and S0W2). In this step, you update a member that is used by your current system, so you must be careful not to create any syntax errors that can stop that system from starting.
17. Because we wanted to give you the flexibility to switch back and forth between sysplex mode and monoplex mode, we attempted to isolate all parmlib changes to a new set of members in USER.PARMLIB, rather than change any of the members in ADCD.xxxx.PARMLIB or SYS1.PARMLIB. In this step, you copy the 2016 Sysplex Extensions-provided parmlib members to USER.PARMLIB or to SYS1.IPLPARM (in the case of the LOADxx member). In general, only one new member exists for each type. In cases where different values were required, we used the following suffixes:
 

|           |                                                    |
|-----------|----------------------------------------------------|
| <b>BS</b> | Base sysplex under z/VM                            |
| <b>PS</b> | Parallel Sysplex under z/VM                        |
| <b>ST</b> | Base sysplex that is spread across two or more PCs |
18. Create PROCLIB members for SHUTSYS, VTAMPS, and TCP/IP.
19. Create TCPPARMS members as described later.
20. Create VTAMLST members for ATCCONPS, ATCSTRPS, and OSATRLx.
21. Create mount points and symlinks for new UNIX System Services data sets.
22. IPL the first member of your new sysplex.
23. Submit a job to define the log streams for LOGREC, OPERLOG, Health Checker, Resource Recovery Services (RRS), and System Management Facilities (SMF).

Complete the following additional steps if your objective is a base sysplex environment that runs under z/VM:

1. Because a base sysplex does not have a coupling facility (CF), it uses channel-to-channels (CTCs) for intersystem cross-system coupling facility (XCF) communication. Therefore, you need to define the CTCs in the VM directory.
2. Another aspect that relates to the lack of a coupling facility is global resource serialization (GRS). Because no CF exists, GRS must run in Ring mode rather than Star mode. The IEASYSxx member must be updated to reflect this different mode.
3. When RRS is running in a Parallel Sysplex, all of the RRSs in the sysplex normally are in the same RRS group, which means that they all share a set of log streams. In a base sysplex, it is not possible to share log streams across systems. Therefore, the **Start** command for RRS must be changed to specify a different group name for each system.

For a base sysplex that spans more than one PC, the following steps are required (in addition to the steps for a base sysplex that is running under z/VM):

1. Create a Linux shared file environment.
2. Create a second zPDT system (on another PC) with basic Linux and zPDT.
3. Create a devmap that points to the shared files on the “first” Linux system for all of the 3390 volumes.
4. Add CTC definitions to the devmaps on both systems.
5. Add Server Time Protocol (STP) definitions to the devmaps on both systems.
6. Remember to start the zPDT STP function on each Linux system before you start zPDT. zPDT fails if the devmap includes STP definitions and the STP function is not running when zPDT is started.

## Setup steps

This section provides detailed descriptions of the steps that are required to use the 2016 Sysplex Extensions to create a Parallel Sysplex or base sysplex. Most of the steps are common to both types of sysplexes (when run under z/VM). A few extra steps are required to enable the base sysplex under z/VM. Those steps are described in “Other steps for base sysplex under z/VM” on page 192.

Before you begin these implementation steps, we suggest that you print Table E-5 on page 191. The table includes a checklist that you can use to track your progress through the implementation steps. It is also helpful to ensure that you do not miss any steps.

## Installing the base ADCD system

We assume that you are familiar with ADCD and you already have an ADCD-based z/OS system up and running in monoplex mode. If you do not, we strongly suggest that you implement and run an ADCD-based z/OS system in monoplex mode now, by using the standard ADCD documentation that is available at this website:

<http://dtsc.dfw.ibm.com/MVSDS/'HTTPD2.ADCD.GLOBAL.SHTML%28READM22A%29'>

You also need to use *IBM zPDT Guide and Reference: System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

You also need to be familiar with the operation and configuration of the ADCD system before you extend it (by using this package) to a sysplex.

We used the ADCD December 2015 z/OS 2.2 release as the base for this package. We used the following volumes. Depending on which products you use, you might download additional volumes:

- ▶ A2RES1
- ▶ A2RES2
- ▶ A2SYS1
- ▶ A2USS1
- ▶ A2USS2
- ▶ A2PAGA
- ▶ A2PAGB
- ▶ A2PAGC

- ▶ A2PRD1
- ▶ A2PRD2
- ▶ A2PRD3
- ▶ A2CFG1

These volumes provide z/OS and the common products, such as compilers.<sup>1</sup> We also suggest that you install the single-pack z/OS system (volume SARES1). This system can prove invaluable if you make a mistake and end up with a system that does not IPL.

Also, if you expect to need to install service on z/OS, you need to restore the two DLIB volumes: A2DIS1 and A2DIS2. We installed the volumes that are used by CICS V5.2 (A2C521) and DB2 V11 (A2DBB1 and A2DBB2). However, they are not strictly necessary unless you want to use the associated subsystems.

For more information about all of the volumes that are provided by the ADCD deliverable, see the ADCD documentation that is available at this website:

<http://dtsc.dfw.ibm.com/MVSDS/'HTTPD2.ADCD.GLOBAL.SHTML%28READM22A%29'>

## Install z/VM

If your target environment is to run your sysplex under z/VM (either as a base sysplex or a Parallel Sysplex), you need to install the ADCD z/VM package now.

For our Parallel Sysplex, we used the ADCD z/VM 6.3 release<sup>2</sup>. We downloaded the following volumes:

- ▶ 630RL1
- ▶ 630RL2
- ▶ M01RES
- ▶ M01P01
- ▶ M01S01
- ▶ M01W01
- ▶ M01W02
- ▶ M01W03
- ▶ VMCOM1
- ▶ VMCOM2

The VMPROD volume, which is part of the z/VM 6.3 ADCD package, is not strictly needed for Parallel Sysplex operation.

Ensure that you update your devmap file with the statements for the z/VM volumes. You can find a sample devmap file on the download site where you obtain the ADCD z/VM volumes. However, to save time, we used the following statements. For simplicity, we suggest that you also use these device numbers:

```
device 0200 3390 3990 M01RES
device 0201 3390 3990 630RL1
device 0202 3390 3990 630RL2
device 0203 3390 3990 M01W01
device 0204 3390 3990 M01S01
device 0205 3390 3990 M01P01
```

<sup>1</sup> The A2PRDn volumes contain various products and you might not need all of them. For more information about the contents of each volume, see the ADCD documentation.

<sup>2</sup> If you are using zPDT release 6, you must use z/VM 6.3, or z/VM 6.2 with the authorized program analysis reports (APARs) that are required to support IBM z13™. These APARs are required even though you are not running on a real z13.

```
device 0206 3390 3990 VMCOM1
device 0207 3390 3990 VMCOM2
device 0208 3390 3990 M01W02
device 0209 3390 3990 M01W03
device 020A 3390 3990 VMPROD
```

For more information about the use and maintenance of the z/VM system, see the z/VM product documentation that is available at this website:

<http://www.vm.ibm.com/library/>

For information about getting z/VM up and running under zPDT, see the chapter titled “Other System z Operating Systems” in *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

To help you in advance, we provide the following zPDT commands that you must issue to load z/VM under zPDT. You can start VM now to get experience with it if you want, but you must shut it down again later to complete several of the later steps in this process:

- ▶ **LOADPARM 0700** is the command that defines 3270 device 700 as the VM IPL console.
- ▶ **IPL 0200** is the device number of the M01RES volume. The VMCOM1 volume must be defined on device number 0206 in your devmap file.
- ▶ On the x3270 session that corresponds to device 700, press PF10 to load z/VM.

You do not need to be a z/VM expert to use it to host your sysplex. However, you must be familiar with the following tasks:

- ▶ Loading and shutting down z/VM under zPDT
- ▶ Logging on and logging off virtual machines
- ▶ Autologging the CF virtual machines
- ▶ Updating the VM directory and using the **DIRECTXA** command to activate the updated directory
- ▶ **DIAL** to a virtual 3270 device so that you can log on to z/OS applications, such as Time Sharing Option (TSO) or CICS

If you are not familiar with z/VM, we suggest that you install z/VM and run your ADCD z/OS system in monoplex mode under z/VM. This approach gives you an opportunity to get used to using z/VM without the added complexity of running multiple z/OS guests. This experience is valuable later when you are ready to move to a Parallel Sysplex environment.<sup>3</sup>

## Download and create volumes

Now that z/VM is installed and you are familiar with the mechanics of running z/OS under z/VM, the next step is to download the volumes that are provided by the 2016 Sysplex Extensions. You create new volumes and then populate them with system data sets.

<sup>3</sup> All of the z/OS parts of the 2016 Sysplex Extensions installation can be implemented in a z/OS that is running natively under zPDT rather than under z/VM. However, that configuration eliminates the opportunity to gain experience with z/VM before you start your sysplex. It also means that you must go back through all of the steps in this chapter and select the steps that relate to z/VM. However, if your target environment is a base sysplex that spans multiple PCs, it is not necessary to use z/VM.

The zPDT 2016 Sysplex Extensions uses the following volumes:

|               |                                                                                                                                                                         |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>CF0001</b> | This volume is part of the zPDT 2016 Sysplex Extensions package. It contains CDSs, sample JCL, and sample definitions.                                                  |
| <b>CICS01</b> | This volume is also part of the zPDT 2016 Sysplex Extensions package. It contains the user catalog and data sets for the CICSplex regions.                              |
| <b>CICS02</b> | This volume is the second volume for the CICSplex data sets, and it also is included in the 2016 Sysplex Extensions package.                                            |
| <b>DB2001</b> | This volume is part of the zPDT 2016 Sysplex Extensions package. It contains the data sets for the DB2 data sharing subsystems.                                         |
| <b>DB2002</b> | This volume is the second volume for the DB2 data sets, and it also is included in the 2016 Sysplex Extensions package.                                                 |
| <b>A2PAGX</b> | This volume starts as an empty volume so it is <i>not</i> downloaded as part of the package. This volume will contain page data sets for the second z/OS system (S0W2). |
| <b>A2PAGY</b> | This volume is an empty volume that will contain page data sets for the second z/OS system.                                                                             |
| <b>A2PAGZ</b> | This volume is an empty volume that will contain page data sets for the second z/OS system.                                                                             |
| <b>A2SYS2</b> | This empty volume will contain system and work data sets that are used by the second z/OS system.                                                                       |
| <b>WORK01</b> | This volume also starts as an empty volume. It is used to contain work data sets only, and it is mounted as a STORAGE volume in the VATLSTPS member.                    |

CF0001 is defined as a 3390-1, and the CICS and DB2 volumes are 3390-3s. Because you are allocating the other volumes, you control their size. We suggest 3390-9s for the paging volumes. The A2SYS2 and WORK01 volumes need to be at least 3390-3s.

The CF0001, CICS01, CICS02, DB2001, and DB2002 volumes need to be downloaded from the following website:

<ftp://www.redbooks.ibm.com/redbooks/SG248315/>

Five .gz files exist, one for each of the first five volumes. Download the following files into the directory that contains your ADCD z/OS files:

- ▶ cf0001.gz
- ▶ cics01.gz
- ▶ cics02.gz
- ▶ db2001.gz
- ▶ db2002.gz

After the downloads complete, open a command prompt window, browse to the directory that contains your z/OS volumes, and use the **gunzip** command (for example, **gunzip -c cf0001.gz > CF0001**) to extract the volumes from the .gz files.

The other volumes contain only data sets that are defined during the installation process. Therefore, you need to create only the empty volumes at this stage in the process. Several of the data sets on these volumes must be cataloged in your master catalog. Therefore, we think that providing you with instructions and jobs to create them is easier than providing the pre-populated volumes and then needing to re-catalog the VSAM data sets in your master catalog.

Now, use the following commands to create Linux files that will contain the z/OS volumes later (this process can take some time, depending on the speed of your PC's hard disk drive):

- ▶ `a1cckd A2PAGX -d3390-9`
- ▶ `a1cckd A2PAGY -d3390-9`
- ▶ `a1cckd A2PAGZ -d3390-9`
- ▶ `a1cckd A2SYS2 -d3390-3`
- ▶ `a1cckd WORK01 -d3390-3`

You now have the five volumes that are provided by this package, plus five new volumes that you soon initialize with Device Support Facilities (ICKDSF). But before we can perform that initialization, we must update the devmap file to add the definitions for the extra volumes.

## Establish a known restart point

When you have a working z/OS system that is running under z/VM and before you change z/VM or z/OS to add sysplex support, we suggest that you create a clean restart point that you can fall back to if your changes result in a system that does not initialize successfully.

*One* way to create this restart point is to ensure that the zPDT environment (z/OS and z/VM) is stopped and then take a copy of the z/VM and z/OS Linux files. If problems occur, you can then restore those volumes (or start zPDT by using a devmap file that references those copies directly).

An alternative is to use the disk versioning support that is provided by zPDT. This support is an attractive option if you expect to perform repetitive testing and want to repeatedly return to a known restart point. For more information about this capability, see the chapter “CKD versioning” in *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

To help you start with the disk versioning function, we included a basic Linux script to enable versioning for our z/OS and z/VM disks. The script is shown in “zPDT Disk versioning sample script” on page 126. You can customize this script to include more volumes (your own z/OS volumes, for example), or create a corresponding script to accept the outstanding changes and create a restart point, or to discard the changes and return to your known restart point. *The z/OS and z/VM systems must be shut down at the time that these scripts are run.*

If you do not want to back up all of your volumes, we suggest that you create a backup of the following data sets at a minimum:

- ▶ `ADCD.Z22A.PARMLIB` (or the corresponding data set for your level of ADCD)
- ▶ `ADCD.Z22A.PROCLIB` (or the corresponding data set for your level of ADCD)
- ▶ `USER.PARMLIB`
- ▶ `USER.PROCLIB`
- ▶ `USER.TCPPARMS`
- ▶ `USER.VTAMLST`

**Note:** Your z/VM and z/OS system likely is down at this point; therefore, make a note to run this job after you start your z/OS system.

Apart from providing an ability to back out any changes, having a “before” copy of the data sets that you are changing helps you more easily visualize the differences between the system that you started with (the ADCD monoplex) and the sysplex with which you end up.



Finally, backups are like umbrellas in that you likely need a backup only if you do not have one. Therefore, taking a few minutes to create them at this point is a good investment of your time. To make it even easier, you can use the JCL that is shown in Example E-1.

*Example E-1 Sample JCL to back up key ADCD-provided system data sets*

---

```
//BACKUP JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//BACKUP EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT3 DD UNIT=SYSDA,SPACE=(3120,(20,10))
//SYSUT4 DD UNIT=SYSDA,SPACE=(3120,(20,10))
//*
//IN1 DD DISP=SHR,DSN=ADCD.Z22A.PARMLIB
//OT1 DD DISP=(NEW,CATLG),DSN=SYSPLEX.Z22A.PARMLIB.BKUP,
// UNIT=SYSDA,VOL=SER=A2SYS1,
// LIKE=ADCD.Z22A.PARMLIB
//IN2 DD DISP=SHR,DSN=ADCD.Z22A.PROCLIB
//OT2 DD DISP=(NEW,CATLG),DSN=SYSPLEX.Z22A.PROCLIB.BKUP,
// UNIT=SYSDA,VOL=SER=A2SYS1,
// LIKE=ADCD.Z22A.PROCLIB
//IN3 DD DISP=SHR,DSN=USER.PARMLIB
//OT3 DD DISP=(NEW,CATLG),DSN=SYSPLEX.USER.PARMLIB.BKUP,
// UNIT=SYSDA,VOL=SER=A2SYS1,
// LIKE=USER.PARMLIB
//IN4 DD DISP=SHR,DSN=USER.PROCLIB
//OT4 DD DISP=(NEW,CATLG),DSN=SYSPLEX.USER.PROCLIB.BKUP,
// UNIT=SYSDA,VOL=SER=A2SYS1,
// LIKE=USER.PROCLIB
//IN5 DD DISP=SHR,DSN=USER.TCPPARMS
//OT5 DD DISP=(NEW,CATLG),DSN=SYSPLEX.USER.TCPPARMS.BKUP,
// UNIT=SYSDA,VOL=SER=A2SYS1,
// LIKE=USER.TCPPARMS
//IN6 DD DISP=SHR,DSN=USER.VTAMLST
//OT6 DD DISP=(NEW,CATLG),DSN=SYSPLEX.USER.VTAMLST.BKUP,
// UNIT=SYSDA,VOL=SER=A2SYS1,
// LIKE=USER.VTAMLST
//*
//SYSIN DD *
COPY INDD=IN1,OUTDD=OT1
COPY INDD=IN2,OUTDD=OT2
COPY INDD=IN3,OUTDD=OT3
COPY INDD=IN4,OUTDD=OT4
COPY INDD=IN5,OUTDD=OT5
COPY INDD=IN6,OUTDD=OT6
/*
```

---

Ensure that the job ended with return code 0 before you change these data sets.

## Starting over

If you find yourself in a position where you want to go back to the beginning and start the installation of the 2016 Sysplex Extensions all over again, you need to delete (or rename) *all* of the following volumes and run the **gunzip** command against the downloaded data sets again:

- ▶ CF0001
- ▶ CICS01

- ▶ CICS02
- ▶ DB2001
- ▶ DB2002

Additionally, you need to delete *all* of the following volumes and create new versions by using the **a1cckd** command:

- ▶ A2PAGX
- ▶ A2PAGY
- ▶ A2PAGZ
- ▶ A2SYS2
- ▶ WORK01

## Updating devmap to add new volumes

Before any volume can be used by a system that is running under zPDT, the device and the associated Linux file must be defined in the devmap file. This requirement means that you must update your devmap file by adding the new volumes to your definitions.

z/VM automatically generates a device control block for any device that you define in the devmap file. However, on the z/OS side, you must ensure that the device numbers that you use are defined in the IODF. (For more information about the devices and associated device numbers that are defined in the ADCD-provided IODF, see Table A-1 on page 122.)

You can use the following devmap statements as examples to help you add the new volumes to your devmap (all of these addresses are defined as 3390 disks in the ADCD z/OS IODF file):

```
device 0ab0 3390 3990 CF0001
device 0ab1 3390 3990 CICS01
device 0ab2 3390 3990 CICS02
device 0ab3 3390 3990 DB2001
device 0ab4 3390 3990 DB2002
device 0ab5 3390 3990 A2PAGX
device 0ab6 3390 3990 A2PAGY
device 0ab7 3390 3990 A2PAGZ
device 0ab8 3390 3990 A2SYS2
device 0ab9 3390 3990 WORK01
```

The last value in each line is the Linux file name for the volume. If your devmap file includes a directory list in the file name (for example, 'frank/z/A2RES1'), you must adjust the device statements.

When you update the devmap, note the device number that you assign to the new volumes and the name of the Linux file that contains each volume. (You can enter the information in Table E-3.) The devmap points only to a Linux file. The file name and the contents of the file are not necessarily correlated. However, we *strongly* advise that the name of the Linux file matches the volser of the volume that is in that file. For example, Linux file A2SYS2 needs to contain the z/OS volume that is called A2SYS2. Failing to use this naming convention can lead to confusion and wasted time later.

Table E-3 zPDT 2016 Sysplex Extensions volumes

| Volser | Linux file name | Device number | Initialized? |
|--------|-----------------|---------------|--------------|
| CF0001 |                 |               | N/A          |
| CICS01 |                 |               | N/A          |
| CICS02 |                 |               | N/A          |
| DB2001 |                 |               | N/A          |
| DB2002 |                 |               | N/A          |
| A2PAGX |                 |               |              |
| A2PAGY |                 |               |              |
| A2PAGZ |                 |               |              |
| A2SYS2 |                 |               |              |
| WORK01 |                 |               |              |

Now is a good time to add several commands to the devmap file to automatically start x3270 sessions for you. When you are running z/OS under z/VM, you will need the following x3270 sessions:

- ▶ One session for the z/VM console (OPERATOR normally is logged on to that session)
- ▶ One session for each z/OS system console
- ▶ Two sessions with which you can log on to TSO without requiring Transmission Control Protocol (TCP) access
- ▶ Two sessions for logging on to various z/VM virtual machines

To support these sessions, consider adding the following statements after the processors statement in the [system] section of your devmap file now:

```
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
command 2 x3270 localhost:3270
```

You also need to update the devmap to add the definitions of the OSA devices to connect your systems to the network. For more information about using the statements, see “Network definitions” on page 182. For now, add the following definitions:

```
[manager]
name awsosa 0023 --path=A0 --pathtype=OSD --tunnel_intf=y
device 400 osa osa
```

```
device 401 osa osa
device 402 osa osa
device 408 osa osa
device 409 osa osa
device 40A osa osa
```

```
[manager]
name awsosa 0024 --path=f0 --pathtype=OSD
device 404 osa osa
device 405 osa osa
device 406 osa osa
device 40C osa osa
device 40D osa osa
device 40E osa osa
```

**Note:** The device numbers and the distinction between the devices that are used for a tunnel to the underlying Linux system are important. The values that are used here match the values in the default ADCD-provided VM directory. Those values in turn match the values in the VTAMLST and TCPPARMS members. Unless you are experienced with VTAM and TCP, we suggest that you use these values exactly as they are provided here.

The exception is the value on the path= parameter, which must be tailored to match the value from *your* PC. For more information, see “Network definitions” on page 182.

If your zPDT system is running, you need to shut it down after you apply the changes to the devmap file. Then, run the **awsstop** command, followed by the **awsstart** command to load the new devmap information. Devices that are not defined in the devmap file cannot be added dynamically. Therefore, you must stop and restart zPDT to make them known to zPDT.

When the new devmap loads successfully by using the **awsstart** command, the next step is to IPL z/VM. The VM IPL address is 0200 and the console address, which is specified on the loadparm, needs to be 0700. The console is written to quickly, but you might need to minimize several of the other x3270 windows to see the VM console.

When the system finishes the IPL process, verify that the device numbers that you assigned to your new volumes are known to the system. From the z/VM OPERATOR ID, enter the **Q nnnn** command where **nnnn** is the device number that you used in the devmap file. The volumes that were delivered by the zPDT 2016 Sysplex Extensions need to be online. The output from the **Q** command displays information that is similar to DASD 0AB0 CF0001. The devices that contain the empty volumes (the last five volumes that are listed in Table E-3 on page 161) are shown as 'FREE' until you initialize them.

## Configuring z/VM guests

The Parallel Sysplex requires four virtual machines (also known as *guests*): two coupling facility virtual machines, which are called CFCC1 and CFCC2, and two z/OS virtual machines, which are called S0W1 and S0W2. If you want to run the ADCD system as a monoplex under VM, we suggest that you use a virtual machine that is called BASEAD. The z/VM 6.3 ADCD package includes the definitions for all of these virtual machines in the supplied VM directory, which is stored in the file that is called USER DIRECT C on the MAINT user ID. The initial password for these user IDs matches the user IDs.

**Important:** The BASEAD guest must *not* be used at the same time as the S0W1 guest or the S0W2 guest.

To make it easier to share the disks between multiple guests and to add guests easily if you need a sysplex with more than two members, all of the z/OS disks are defined as belonging to a virtual machine that is called MVSDUMMY. The S0W1, S0W2, and BASEAD user IDs are all defined to link to the MVSDUMMY-owned disks. (The VM directory statements that we used for these four IDs are described in “Sample z/VM Directory entries” on page 114.)

The sample definitions contain more DASD (MDISK and LINK) statements than you need for the basic ADCD systems. This configuration is used to make it easier to add volumes to your systems later.

You might need to make only one small change to the ADCD VM directory. The WRKALLEG option after an MDISK statement controls how z/VM segments channel programs. If you specify WRKALLEG for a device, z/VM does not segment a channel program, which ensures that any working allegiance is not severed by z/VM ending the channel program “early.” (Anything that performs atomic I/Os needs working allegiance.) In general, it needs to default to OFF because OFF allows z/VM to manage channel programs from guests better. However, the WRKALLEG option *must* be present for any volumes that contain CDSs.

Because we do not know the address that you use for the CF0001 volume (the volume that contains the CDSs for the base and Parallel Sysplexes), you must edit the VM directory and add the WRKALLEG statement at the correct location. For example, if you use the same addresses as our devmap file that was described in “Sample devmap file” on page 112, you insert a DASDOPT WRKALLEG statement immediately after the MDISK AB0 statement.

If you plan to use the system-managed duplexing function, you need to define links between the two coupling facility virtual machines. Add the following line to the entries for the CFCC1 and CFCC2 user IDs, immediately after the CONSOLE statements:

```
SPECIAL 1600 MSGP targid
```

The *targid* is the name of the CFVM that you want to connect to. So, for example, when you add this line to the CFCC1 entry, you specify SPECIAL 1600 MSGP CFCC2.

After you make any required changes to the directory entries, run the **DIRECTXA USER DIRECT C** command. This command updates the VM directory from your source file. You must perform this update *before* you log on to the z/OS virtual machines.

## Initializing new volumes

The next step is to use ICKDSF to initialize the new empty volumes by using your ADCD z/OS monoplex system.

Log on to the BASEAD user ID on z/VM. You will see many “DASD 0Axx offline” messages. Those messages relate to the spare devices that we referred to in “Configuring z/VM guests” on page 162, so they can be ignored.

Now, IPL z/OS by using the following commands:

```
TERM CONMODE 3270
IPL A80 LOADP 0A8200M1
```

Your system will IPL in monoplex mode by using the x3270 session in which you entered the commands as the z/OS console.

When the system comes up, issue **D U, , ,nnnn,1** commands to verify that the new devices (the devices that you must initialize) are offline. (They *must* be offline because they are not initialized yet.)

The next step is to log on to the IBMUSER or ADCDMST TSO user IDs. To get to the z/OS TSO logon window, enter DIAL BASEAD in the COMMAND area of any of the available VM panels.

Your ADCD.LIB.JCL data set needs to contain a member that is called DSFINIT. Use the information that you entered in Table E-3 on page 161 to help customize the job to ensure that you use the correct device number for each volume. Ensure that you initialize only the volumes that do *not* have “N/A” in the Initialized column. Update the PGM=ICKDSF statement to add REGION=OM<sup>4</sup> and tailor the SYSIN input as shown in the following example:

```
INIT UNIT(0AB5) NVFY VOLID(A2PAGX) VTOC(0,1,29) INDEX(2,0,15)
```

Perform this process for each of the five new volumes (A2PAGX, A2PAGY, A2PAGZ, A2SYS2, and WORK01). You must reply U to the ICK003D message on the z/OS console for each volume. After the initialization jobs complete (check to ensure that they ended with return code 0), run the **V nnnn,ONLINE** command to bring those devices online.

## Importing user catalogs

Most of the data sets that are provided by the zPDT 2016 Sysplex Extensions are cataloged in user catalogs on the 2016 Sysplex Extensions-provided volumes. To make those user catalogs and the data sets they reference available to your system, you must run an IDCAMS IMPORT CONNECT to make them known to your master catalog.

**Note:** The 2016 Sysplex Extensions assumes that you use the same master catalog for the Parallel Sysplex, base sysplex, and your monoplex configurations. Using separate master catalogs complicates matters and provides no extra benefit.

Job IMPCONN in data set SYSPLEX.PARALLEL.CNTL.ZOS22 on the CF0001 volume performs the IMPORT CONNECTs and defines the related aliases for you.

Run the IMPCONN job now and verify that it ends with return code 0.

## Creating system-specific system data sets

Certain system data sets must not be shared between systems. In this step, we allocate those data sets for the S0W2 system. We also change the S0W1 data sets to adapt them to a sysplex environment.

---

<sup>4</sup> If you do not adjust the region size, the job might end with a return code 12 and message ICK31327I.

Figure E-1 shows the contents of the A2PAGA/B/C/X/Y/Z and the A2SYS1 and A2SYS2 volumes at the start of this step.



Figure E-1 System volumes and system data sets

Figure E-2 shows the contents of those volumes *after* you complete all of the tasks in this step. Data sets exist for the S0W1 and S0W2 systems when they run in sysplex mode, and data sets with the original names exist that can be used when the system runs in monoplex mode.

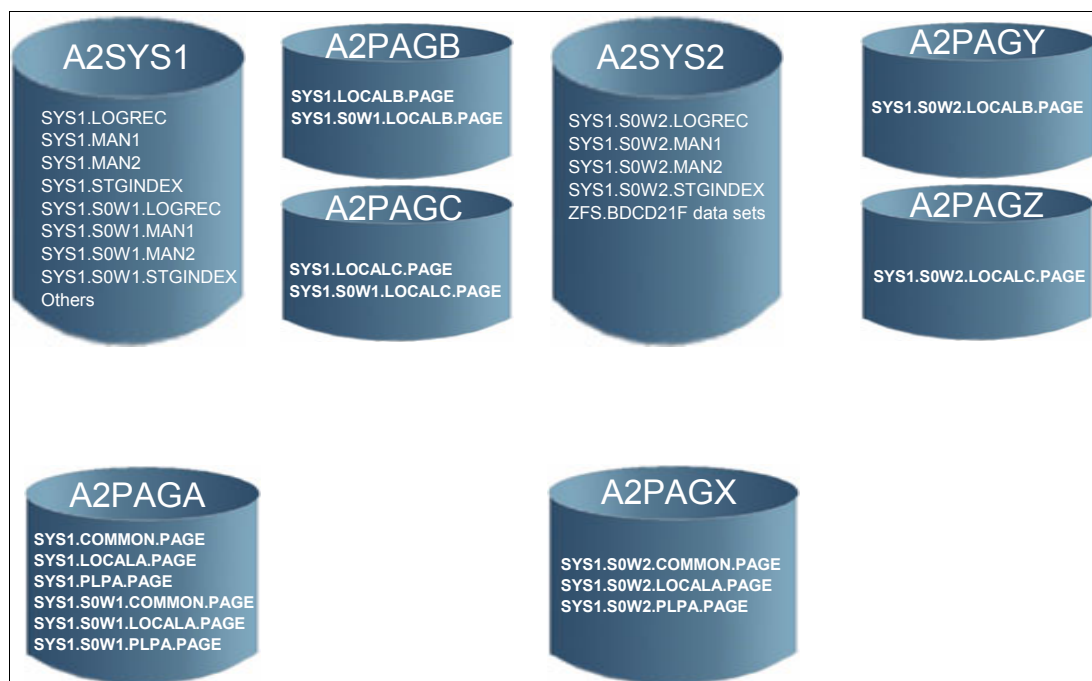


Figure E-2 System volumes and system data sets

Data set `SYSplex.PARALLEL.CNTL.ZOS22` contains a job that is called `VSAMS0W2` to define the Paging, VIO, SMF, and LOGREC data sets for system `S0W2`. (You can access data set `SYSplex.PARALLEL.CNTL.ZOS22` by using the normal catalog search order after you run the `IMPCONN` job.) The page data sets are allocated on the `A2PAGX/Y/Z` volumes. The other data sets are allocated on the `A2SYS2` volume. All of these data sets are cataloged in the master catalog.

Submit the `VSAMS0W2` job now. This job formats all of the page data sets. It might appear that nothing is happening, but be patient. On our system, it took nearly 10 minutes. Then, it ended with a return code 0.

## Reallocating page data sets for the `S0W1` system

The `S0W1` system (when it is part of the `sysplex`) *can* use the paging, SMF, VIO, and LOGREC data sets of the base (monoplex) `z/OS` system. However, to demonstrate the use of system symbols in system data set names, we provide a set of jobs to delete and reallocate the corresponding data sets for system `S0W1` by using the same naming convention that we use for the `S0W2` data sets.

Because the page data sets on your ADCD-provided monoplex system take up the entire volume, the following process is used to allocate the new page data sets. (This process seems long, but the results are worth the effort.)

**Tip:** This step and the following step (cloning the OMVS data sets) are I/O intensive. On a PC with a single hard disk drive, running multiples of these jobs in parallel elongates the overall elapsed time. Therefore, we suggest that these steps are submitted in sequence and that you wait for each job to complete before you start the next job.

Follow these steps:

1. Run the **D ASM** command. The response tells how many *local* page data sets are in use. Most likely, two sets are in use: `SYS1.LOCALA.PAGE` and `SYS1.LOCALB.PAGE`. Make a note of those data set names.
2. If `SYS1.LOCALC.PAGE` was *not* in the list, perform an Interactive System Productivity Facility (ISPF) 3.4 for `SYS1.LOCALC.PAGE` and delete that data set by entering `DEL / PGSPC` on the correct line in the data set list.

**Note:** If you cannot delete the page data set by using this method, customize and submit the `DELPGSPC` job in `SYSplex.PARALLEL.CNTL.ZOS22` to perform the delete by using a batch job.

3. If `SYS1.LOCALC.PAGE` *was* in the list, run a **PAGEDEL DELETE,PAGE=SYS1.LOCALC.PAGE** command. This command moves all active pages out of that data set. When the command completes (often takes a few minutes), perform a 3.4 for `SYS1.LOCALC.PAGE` and delete that data set. (The **PAGEDEL** command empties only the data set, it does not delete it.) You now have sufficient space on the `A2PAGC` volume to allocate two page data sets: one with the original ADCD name and one with a name that includes the system name.
4. Run the `VSAM1A` job from the `SYSplex.PARALLEL.CNTL.ZOS22` data set. This job allocates two local page data sets on `A2PAGC`: `SYS1.S0W1.LOCALC.PAGE` (for use by `S0W1` when in `sysplex` mode) and `SYS1.LOCALC.PAGE` (for use by your monoplex system). The system formats the page data sets as part of the allocation process, so expect the job to run for a few minutes.
5. When the `VSAM1A` job completes, run a **PAGEADD PAGE=SYS1.LOCALC.PAGE** command to bring that page data set into service.



6. Run a **PAGEDEL DELETE,PAGE=SYS1.LOCALB.PAGE** command to empty the local page data set on A2PAGB. When the command completes, perform an ISPF 3.4 for SYS1.LOCALB.PAGE and delete that data set. (It often takes a few minutes before you see an IEE205I message that indicates that the page data set was “deleted”.)
7. Run the VSAM1B job. This job allocates two local page data sets on A2PAGB: SYS1.SOW1.LOCALB.PAGE and SYS1.LOCALB.PAGE.
8. After the VSAM1B job completes with return code 0, run a **PAGEADD PAGE=SYS1.LOCALB.PAGE** command to bring that page data set back into service.
9. Run a **PAGEDEL DELETE,PAGE=SYS1.LOCALA.PAGE** command to empty the local page data set on A2PAGA. When the command completes, perform a 3.4 for SYS1.LOCALA.PAGE and delete that data set.
10. Run the VSAM1C job. This job allocates new SMF, VIO, and LOGREC data sets that contain the system name for SOW1 on the A2SYS1 volume, plus two local page data sets on A2PAGA: SYS1.SOW1.LOCALA.PAGE and SYS1.LOCALA.PAGE.
11. When the VSAM1C job completes, run the **PAGEADD PAGE=SYS1.LOCALA.PAGE** command to bring that page data set back into service.

This process can take up to 20 minutes (most of which is spent waiting for the jobs to finish initializing the new page data sets). However, at the end of this process, you have a set of page data sets that can be used by the standard ADCD z/OS monoplex system. You also have a set of page data sets that can be used by the base sysplex or Parallel Sysplex systems. And you also have a job (VSAMSOW2) and a process that you can use if you want to create system-specific data sets for another system in the future.

You can use the IEASYSxx and IEASYMxx parmlib members that are provided with this package without needing to modify them. Those members can be extended easily to support any number of systems if you use consistent naming for the system-specific data sets.

A bonus is that you rarely have an opportunity to remove and add page data sets to a running system, so this process gives you a chance to test and become familiar with the process if you ever need to use it on a production system.

When you run in sysplex mode, you can save LOGREC and SMF data in a log stream, so the LOGREC and SMF data sets that are allocated by job VSAM1C are not used. However, by creating these data sets, you give yourself the option of running in DATASET or LOGSTREAM mode, and dynamically switching back and forth between the two.

## Creating zFS file systems for system SOW2

A subset of the zFS data sets is system-specific. The 2016 Sysplex Extensions package provides a job that is called OMVSCLOM in SYSPLEX.PARALLEL.CNTL.ZOS22 to create a copy of these data sets for system SOW2 and place them on volume A2SYS2.

The naming convention of the zFS data sets that are in our sample system is not ideal for a sysplex environment. Ideally, we will change the ADCD22A and BDCD22A qualifiers to reflect the names of the systems that use the data set (SOW1 and SOW2). However, this process involved too many other changes in the base ADCD system. This change might be made in a future release.

The sysplex root and instance file systems were created in our ADCD 2.2 system. If the sysplex root did not exist, SYS1.SAMPLIB(BPXISYR) and SYS1.SAMPLIB(BPXISYZS) provide jobs to create it.

## Create mount points and symlinks for system-specific file systems

The standard ADCD system provides a S0W1 directory and mounts many, but not all, system-specific file systems at that directory. Job MKDIR in SYSPLEX.PARALLEL.CNTL.ZOS22 adds an S0W2 directory for the S0W2 file systems that were created by job OMVSCLOM. It also creates a symlink for /web and moves /web from the root into the S0Wx directories.

Ensure that the HTTPD1 address space is not running. Then, run job MKDIR and use the ishell to verify that the changes were completed successfully.

## Creating Health Checker persistent data data set for S0W2

The z/OS Health Checker persistent data data set, which is called HZSPDATA, provides the Health Checker with the ability to carry information over from one IPL to the next. This capability is powerful because the Health Checker can identify changes that were introduced by an IPL that you might otherwise not be aware of.

The ADCD system includes a data set that is called ADCD.S0W1.HZSPDATA that is used by the Health Checker started task on system S0W1. In this step, we allocate a corresponding data set for the S0W2 system. If you do not run this step, Health Checker fails with a JCL error on system S0W2.

Submit the HZSALLCP job in SYSPLEX.PARALLEL.CNTL.ZOS22 to allocate the ADCD.S0W2.HZSPDATA data set.

## Recataloging master catalog data sets

A few data sets (the new 2016 Sysplex Extensions-supplied CDSs) are provided on the 2016 Sysplex Extensions-supplied CF0001 volume that must be cataloged in the master catalog. Job RECATLG in SYSPLEX.PARALLEL.CNTL.ZOS22 contains the IDCAMS statements to add those data sets to your master catalog.

The data sets that are cataloged in the master catalog all have a high-level qualifier (HLQ) of PARALLEL. Check to ensure that an alias of PARALLEL is not already defined in the master catalog. Then, run the RECATLG job and ensure that it completes with return code 0.

## SMS changes

In an increasing number of cases, software products require that data sets are placed on SMS-managed volumes. For this package, several of the DB2 data sets must be SMS-managed. For simplicity, we also packaged our entire CICSplex environment in two SMS-managed volumes (CICS01 and CICS02).

As a result, multiple changes are needed for SMS. You have an SMS SCDS and ACDS, which are provided by ADCD, and you likely changed them. For that reason, we did not want to ship replacement SMS control data sets because their use will regress your changes. Also, no mechanism exists to merge two sets of SMS control data sets. Also, no way exists to update the SMS definitions from a batch job, so we have no choice but to ask you to apply these changes by using ISPF.

However, the process is not as bad as it seems. It takes only a few minutes. The following changes are required:

- ▶ The second z/OS system (S0W2) and the sysplex name (ADCDPL) must be added to the SMS configuration.
- ▶ Two data classes must be added for the automatic allocation of LOGR staging and offload data sets, and one data class is added for DB2 use.
- ▶ Two storage groups must be defined: one for the DB2 user volumes and one for the CICS user volumes.
- ▶ A new storage class must be defined for the data sets on the CICS or DB2 SMS-managed volumes.
- ▶ The SMS storage class and storage group ACS routines must be updated to reflect the new storage classes and storage groups.
- ▶ You must change the ADCD-supplied SMS control data sets' share options to 3,3 to support cross-system sharing.

## Adding S0W2 and ADCDPL to SMS

Complete the following steps:

1. Go to the ISMF panels in ISPF (option **M.2** in recent ADCD systems).
2. Ensure that you are in Administrator mode.  
If you can see option 8 (Control Data Set) in the panel, you are in Administrator mode.  
If not, select option **0** (Profile), then option **0** (User Mode), then option **2** (Administrator). Then, exit out of ISMF (by pressing F3) and restart it.
3. Select option **8** (Control Data Set).
4. Set the CDS name to SYS1.SCDs at the top of the panel.
5. Select option **3** (Alter).
6. Page down (F8) to the second panel.
7. Specify option **1** (Add), and System Name S0W2. Then, press Enter.
8. Specify option **1** (Add), and Sys Group Name ADCDPL. Then, press Enter.
9. Exit this panel (F3).
10. Select option **4** (Validate the SCDS) and look for the message "SUCCESSFUL VALIDATION" in the upper-right corner of the window.
11. Press F3 to exit this panel.
12. Select option **5** to activate the CDS. Enter a forward slash (/) to request activation.  
You will see an "ACTIVATION SCHEDULED" message and the IBM MVS™ console shows a "NEW CONFIGURATION ACTIVATED" message.
13. Press F3 to exit from ISMF.

## Creating data classes

The System Logger uses the following types of data sets for log streams:

- ▶ Staging data sets. These sets are optional, depending on how the log stream is defined. They *must* have a control interval (CI) size of 4 KB.
- ▶ Offload data sets. Every log stream has one or more offload data sets. These data sets support any CI size that is a multiple of 4 KB. However, we highly advise that these data sets have a CI size of 24 KB for optimal performance.

Because of the different CI sizes, two data classes are necessary for System Logger use: one for the staging data sets (named LOGR4K) and one for the offload data sets (named LOGR24K).

To define the LOGR4K data class, complete the following steps:

1. Go to the ISMF panels in ISPF.
2. Select option **4** (Data Class).
3. Set the CDS name to SYS1.SCDS at the top of the panel.
4. Enter LOGR4K as the Data Class Name.
5. Select option **3** (Define) and press Enter.
6. Press F8 to scroll through several panels. Make *only* the following changes:
  - Description = Data class for Logger staging data sets (Page 1)
  - RECORG = LS (Page 4)
  - CIsze Data = 4096 (Page 4)
  - Shareoptions Xregion = 3 (Page 6)
  - Xsystem = 3 (Page 6)
7. Press F3 to return to the main ISMF menu.
8. Repeat these steps for the LOGR24K data class, remembering to specify a CISIZE of 24576 instead of 4096.

Also, DB2 now requires that certain data sets are on SMS-managed volumes. For more information about this requirement, see this website:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-1302smsenvironment/>

To define the data class for DB2, complete the following steps:

1. Enter DPDGDC as the Data Class Name.
2. Select option **3** (Define).
3. Press F8 to scroll through a number of panels. Make *only* the following changes:
  - Description = Data class for DB2 (Page 1)
  - Data Set Name Type = EXT (Page 2)
  - If Ext = R (Page 2)
  - Extended Addressability = Y (Page 2)
  - Record Access Bias = U (Page 2)
4. Press Enter to save your changes. Then, press F3 to return to the main ISMF menu.
5. Exit to the ISMF Primary Option Menu.
6. Select option **8** (Control Data Set) and option **5** (Activate the CDS) to activate the modified CDS.

## Defining new storage groups

Complete the following steps to define the two new storage groups:

1. Go to the ISMF panels in ISPF.
2. Select option **6** (Storage Group).
3. Set the CDS name to SYS1.SCDS at the top of the panel.
4. Enter CICFILES as the Storage Group Name.
5. Enter P00L as the Storage Group Type.
6. Select option **3** (Define) and press Enter.

7. Change Auto Migrate and Auto Backup to N and press Enter.
8. Press F3 to save your changes.
9. Select option **5** (Volume) to define your CICS volumes.
10. Enter **2** (Define). Then, tab down to the section where you specify the volsers of the volumes that are in this storage group.
11. Enter CICS0 in the Prefix column, 1 in the From column, and 2 in the To column.
12. Press Enter.
13. Press F3 to save your changes. Then, press F3 again to return to the first Storage Group panel.
14. Repeat these steps for the DB2FILES storage group and use volsers of DB2001 and DB2002.
15. Exit to the ISMF Primary Option menu.
16. Select option **8** (Control Data Set) and option **5** (Activate the CDS) to activate the modified CDS.

### Defining new storage classes

To be on a storage group volume, a data set must be SMS-managed. To be SMS-managed, the data set must have a storage class. Complete the following steps to define a simple storage class that is used for all data sets in the CICS or DB2 storage groups:

1. Go to the ISMF panels in ISPF.
2. Select option **5** (Storage Class).
3. Set the CDS name to SYS1.SCDS at the top of the panel.
4. Enter PSADDON as the Storage Class Name.
5. Select option **3** (Define) and press Enter.
6. You do not need to change any of the settings. Press F3 to save the definition of the new storage class.
7. Press F3 to save your changes.
8. Select option **8** (Control Data Set) and option **5** (Activate the CDS) to activate the modified CDS.

### Updating ACS routines

The ADCD-provided SMS ACS routines are kept in data set SYS1.SMS.CNTL. The following members contain the sources for the ADCD-supplied ACS routines:

|                      |          |
|----------------------|----------|
| <b>Data class</b>    | ACSSTORD |
| <b>Storage class</b> | DB2STORC |
| <b>Storage group</b> | DB2STORG |

You can always find the location of the source of the currently active ACS routines by selecting option 7 (Automatic Class Selection) from the ISMF Primary Option menu. Then, select option 5 (Display). You see a list of routines and their locations, as shown in Figure E-3.

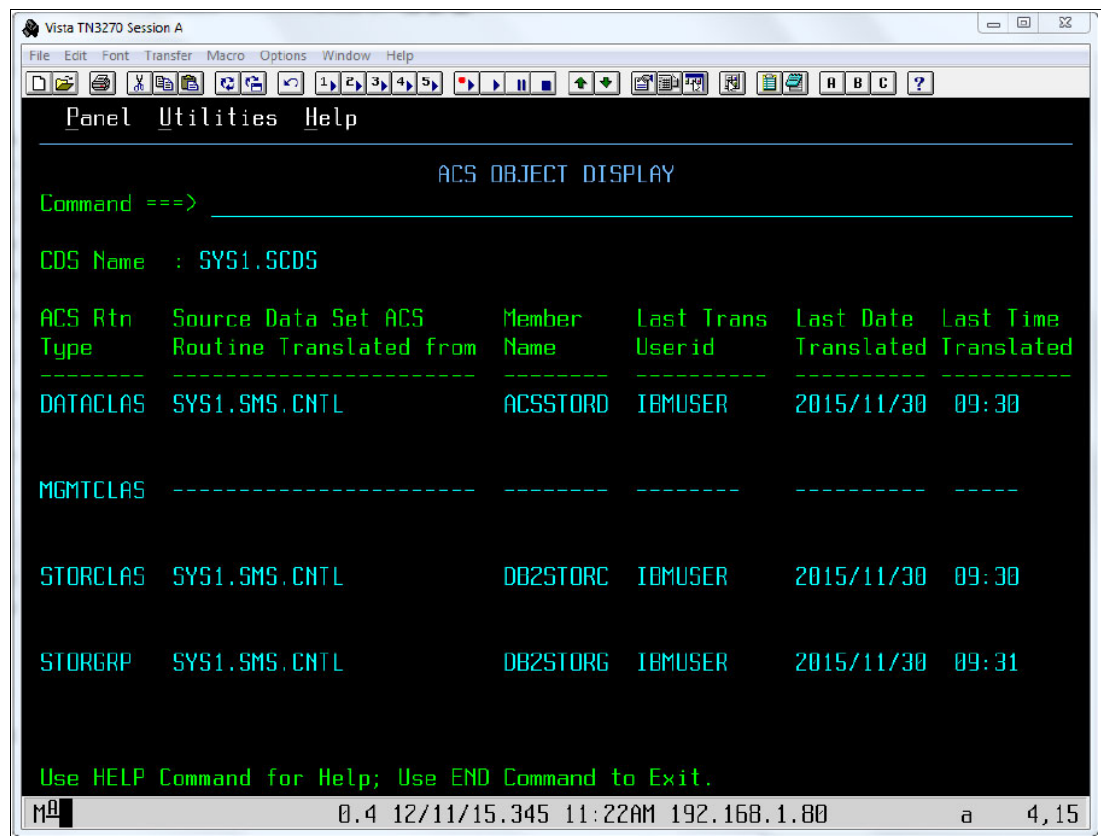


Figure E-3 ISMF display of information about ACS routines

The data class ACS routine must be updated for the new DB2 subsystems that are provided with this package. Make the following changes:

1. Save a copy of the current data class member (ACSSTORD in this case).
2. Add the following lines near the end of the member:

```
IF &DSN(1) = 'DSNPDG' THEN
DO
  IF &DSN(2) = 'DSNDBC' THEN
DO
  IF &DSN(3) = 'DSNDB01' THEN
DO
    SET &DATACLAS='DPDGDC'
  END
  IF &DSN(3) = 'DSNDB06' THEN
DO
    SET &DATACLAS='DPDGDC'
  END
END
IF &DSN(2) = 'DSNDBD' THEN
DO
  IF &DSN(3) = 'DSNDB01' THEN
DO
    SET &DATACLAS='DPDGDC'
```

```

        END
    IF &DSN(3) = 'DSNDB06' THEN
        DO
            SET &DATACLAS='DPDGDC'
        END
    END
END
END
END

```

These statements are provided in the ACSDB2DC member of SYSPLEX.PARALLEL.CNTL.Z0S22, so you can copy them from the member so that you do not need to create them manually. However, that member does *not* contain a full replacement for the data class ACS routine. Instead, it contains only the additions that relate to the 2016 Sysplex Extensions.

We discovered in our testing that the END statements in the provided ACSSTORD member are sometimes incorrect. We advise that you review the member after you add the DB2 data class statements to ensure that every DO statement has a corresponding END statement. For example, the ACSSTORD member of SYSPLEX.PARALLEL.CNTL.Z0S22 contains the corrected member that we used for our testing.

3. Press F3 to save your changes.

You need to edit the storage class ACS routine (DB2STORC) and change the following information:

1. Save a copy of the current storage class member (DB2STORC in this case).
2. Add the following lines to the FILTLIST section at the top of the member:

```

FILTLIST DB2USER          INCLUDE('DSNDPDG')
FILTLIST DB2UCAT          INCLUDE('UCAT.DB2USER')
FILTLIST CICSUCAT         INCLUDE('UCAT.CICSUSER')
FILTLIST CICSUSER         INCLUDE('CTSLOGR', 'CTS52')

```

3. Add the following lines after the SELECT line:

```

WHEN (&DSN = &CICSUCAT)
    DO
        SET &STORCLAS = 'PSADDON'
        EXIT
    END

WHEN (&HLQ = &CICSUSER)
    DO
        SET &STORCLAS = 'PSADDON'
        EXIT
    END

WHEN (&HLQ = &DB2USER)
    DO
        SET &STORCLAS = 'PSADDON'
        EXIT
    END

WHEN (&DSN = &DB2UCAT)
    DO
        SET &STORCLAS = 'PSADDON'
        EXIT
    END

```

These statements are also provided in the ACSDB2SC member of SYSPLEX.PARALLEL.CNTL.Z0S22. That member contains only the extra statements for the storage class ACS routine. It is *not* a complete replacement for the entire member.

4. Press F3 to save your changes.

Finally, the storage group ACS routine must be edited and the following changes must be made:

1. Save a copy of the current member (DB2STORG in this case).
2. Add the following lines to the FILTLIST section at the top of the member:

```
FILTLIST DB2USER          INCLUDE('DSNDPDG')
FILTLIST DB2UCAT          INCLUDE('UCAT.DB2USER')
FILTLIST CICSUCAT         INCLUDE('UCAT.CICSUSER')
FILTLIST CICSUSER         INCLUDE('CTSLOGR', 'CTS52')
```

3. Add the following lines after the SELECT line:

```
WHEN (&DSN = &CICSUCAT)
  DO
    SET &STORGRP = 'CICFILES'
    EXIT
  END
WHEN (&HLQ = &CICSUSER)
  DO
    SET &STORGRP = 'CICFILES'
    EXIT
  END
WHEN (&DSN = &DB2UCAT)
  DO
    SET &STORGRP = 'DB2FILES'
    EXIT
  END
WHEN (&HLQ = &DB2USER)
  DO
    SET &STORGRP = 'DB2FILES'
    EXIT
```

These statements are also provided in the ACSDB2SG member of SYSPLEX.PARALLEL.CNTL.Z0S22. That member contains only the extra statements for the storage group ACS routine. It is *not* a complete replacement for the entire member.

4. Press F3 to save your changes.

After the update of the ACS routines, we are now ready to translate them and then activate the new routines. Complete the following steps:

1. Go to the ISMF panels in ISPF.
2. Select option **7** (Automatic Class Selection).
3. Set the CDS name to SYS1.SCDS.
4. Select option **2** (Translate) and press Enter.
5. On the ACS Source Data Set line, add SYS1.SMS.CNTL.
6. On the ACS Source Member line, enter the name of your updated data class member.
7. On the Listing Data Set line, enter a valid data set name. (It will be created if it does not exist.)
8. Press Enter.



9. Scroll to the end of the listing to ensure that the translation process ended with return code 0.
10. Repeat these steps for the other ACS members that you updated.
11. Press F3 to return to the primary ISMF menu.
12. Select option **8** (Control Data Set) and option **5** (Activate the CDS) to activate the modified CDS.

### Altering share options of SMS control data sets

Next, submit job ACDS5 in SYSPLEX.PARALLEL.CNTL.Z0S22 to change the share options of several SMS data sets to allow them to be shared by the S0W1 and S0W2 systems. Sample JCL statements are shown in Example E-2.

#### *Example E-2 Sample JCL statements*

---

```
//ACDS5 JOB 1,OGDEN,MSGCLASS=X
// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ALTER SYS1.ACDS.DATA SHAREOPTIONS(3 3)
ALTER SYS1.SCDS.DATA SHAREOPTIONS(3 3)
ALTER SYS1.COMMDS.DATA SHAREOPTIONS(3 3)
/*
```

---

The SMS-related changes are now complete.

## JES2 MAS configuration

**Important:** The installation of the 2016 Sysplex Extensions makes only two changes that can affect your current ADCD system. The change that is described in this section is one of those changes. Therefore, you must be careful that the change that we suggest does not conflict with any customization that you performed on your JES2PARM member, and that the changes that you make do not inject any errors into the member.

JES2 must be changed to a Multi-Access Spool (MAS) configuration with two members. This change does not affect the usability of the JES2 parameters for “normal” ADCD use.

We made the following change to member JES2PARM in the ADCD PARMLIB data set. Although we attempt to avoid changing the ADCD data sets, moving the JES2PARM member to the USER.PARMLIB data set will require several other changes, which increase the complexity with little benefit in return.

Also, the change that we make is consistent with running the system in monoplex mode, which means that you can switch back and forth between monoplex and sysplex mode without making any JES-related changes.

**Tip:** Because you are changing a member that is used by your running system, ensure that you can create a backup of the member before it is changed.

The whole JES2PARM member is not shown here, only the changed lines that relate to the MAS definition are shown. (We also changed the number and classes of started initiators, but this change is optional.) A MASDEF statement is already in the JES2PARM member, so you must scroll down to find that statement, delete it, and then insert the following statements:

```

/*
*-----*
*   Multi-Access Spool
*-----*
*/

MASDEF  SHARED=CHECK,
        RESTART=YES,
        CKPTLOCK=ACTION,
        DORMANCY=(25,300),
        HOLD=10,
        LOCKOUT=500

MEMBER(1) NAME=SOW1
MEMBER(2) NAME=SOW2

```

These statements are included in member JES2MASD in SYSPLEX.PARALLEL.CNTL.Z0S22 if you want to copy them from there to the end of the JES2PARM member.

The first time that you start JES2 after you make this change, you see message HASP870, which prompts you to confirm that you want to add member SOW2 to the JES2 MAS. Reply Y to this write to operator with reply (WTOR).

## JES2 Dynamic Proclib

While we updated the JES2PARM, we implemented JES2 dynamic proclib support. This support consists of adding the JES2 proclib definitions to the JES2PARM member. The following statements are used:

```

/*****
/*
*
*   Dynamic JES2 proclib definitions
*
*
*****/
PROCLIB(PROC00) DD(1)=(DSN=USER.PROCLIB),
                DD(2)=(DSN=ADCD.&SYSVER..PROCLIB),
                DD(3)=(DSN=CEE.SCEEPROC),
                DD(4)=(DSN=CSQ800.SCSQPROC),
                DD(5)=(DSN=IOE.SIOEPROC),
                DD(6)=(DSN=EOY.SEOYPROC),
                DD(7)=(DSN=HLA.SASMSAM1),
                DD(8)=(DSN=CBC.SCCNPRC),
                DD(9)=(DSN=SYS1.PROCLIB)

```

These statements are included in member JES2PRCL in SYSPLEX.PARALLEL.CNTL.Z0S22 if you want to copy them from there.

To confirm that your changes are successful, shut down and restart JES2 without removing the proclib definitions from the JES2 JCL. When you are satisfied that your change was successful (use the **\$D PROCLIB** command), you can remove the PROC00 data definition (DD) statements from the JES2 PROC.

The use of this capability includes the following advantages:

- ▶ JES2 PROC JCL is simplified because all of the DD statements for your procedure libraries can be removed from the JES2 JCL.
- ▶ The proclib concatenation can be changed dynamically with no need to stop and restart JES2.
- ▶ If you have a syntax error in your JES2 proc JCL, JES2 does not start. However, if you have a syntax error in the PROCLIB definitions in the JES2PARM member, you are presented with an HASP469 message, which gives you the option of correcting or bypassing the invalid statement.

If a proclib concatenation (PROC00, for example) is specified in the JES2 JCL and JES2 parm member, the parm definitions are used rather than those definitions in the JCL. Therefore, you can add the statements to your JES2PARM member without removing them from the JCL. For more information about dynamically changing your proclib concatenations, see the section “Using dynamic PROCLIB allocation” in *z/OS JES2 Initialization and Tuning Guide*, SA32-0991.

## Creating PARMLIB members

Many of the attributes of how your system and sysplex work are controlled through parmlib members. When this deliverable was designed, we wanted to make the installation as simple and automated as possible. In particular for the parmlib member changes, we wanted to minimize your manual intervention and avoid situations in which we provide a parameter change that clashes with values that were used for your ADCD monoplex system. We also tried to optimize the use of system symbols to minimize the number of members and the administrative effort to keep multiple duplicate or near-duplicate members in sync.

Many (but not all) parmlib members support the ability to concatenate members. If you use concatenated members, the values that are obtained from the first member of the concatenation are overridden if the same parameter is encountered in a later member. For example, assume that you specify SYSPARM(00,PS) in your IEASYMxx member. That specification indicates that IEASYS00 is read first, followed by IEASYSPS. If any parameter is specified in both members, the value from IEASYSPS is used. Any parameters that are not specified in IEASYSPS use the value from IEASYS00.

This capability allows us, where possible, to provide parmlib members that contain only the parameters that must be overridden for the sysplex environment. We use a suffix of PS (for Parallel Sysplex), BS (for base sysplex under z/VM), or ST (for base sysplex spread across two PCs) on our members. The members that we provided and the changes that are in each member are listed in Table E-4 on page 178.

Table E-4 Parmlib members that are supplied by zPDT 2016 Sysplex Extensions

| Member                                   | Supports concatenation? | Changed parms                                                                                                                                                                                                                       |
|------------------------------------------|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BPXPRMPS                                 | Yes                     | Changed sysplex Poroot to be mounted read only.<br>Point both systems at the same USERS file system.<br>Changed /web and /web/httpd1 to mount in the \$SYSNAME directory.<br>Adjusted the AUTOMOVE setting of certain file systems. |
| CLOCKPS<br>CLOCKST                       | Yes                     | Added SIMETRID value.<br>Changed STPMODE to YES.                                                                                                                                                                                    |
| COMMNDPS                                 | Yes                     | Changed VTAM <b>start</b> command to specify that ATCSTRPS needs to be used rather than ATCSTR00.<br>Changed VTAMAPPL started task name to VTAMPS.                                                                                  |
| CONSOLPS                                 | No                      | Replaced entire member.<br>Changes are to NAME parm on the console definition and to enable OPERLOG.                                                                                                                                |
| COUPLEPS                                 | No                      | Replaced entire member.<br>Nearly all parameters changed.                                                                                                                                                                           |
| DEVSUPPS                                 | Yes                     | Added extended TIOT support for non-VSAM data sets.                                                                                                                                                                                 |
| GRSRNLPS                                 | Yes                     | Convert all reserves to ENQs.                                                                                                                                                                                                       |
| IEASYMPS<br>IEASYMBS<br>IEASYMST         | Yes                     | Replaced entire member.<br><br>Adds IEASYSBS and IEASYSST sysparms.                                                                                                                                                                 |
| IEASYSPS<br><br>IEASYSBS<br><br>IEASYSST | Yes                     | Updated to point at the members that require changes for the sysplex environment.<br><br>Changed GRS to TRYJOIN.<br><br>Changed CLOCK to enable STP.                                                                                |
| IEFSSNPS                                 | Yes                     | Added definitions for our CICSplex and data sharing DB2 subsystems.                                                                                                                                                                 |
| LOADPS<br>LOADBS<br>LOADST               | No                      | Set appropriate IEASYM parameter.                                                                                                                                                                                                   |
| LPALSTPS                                 | Yes                     | Add CICS 5.2 to LPALST.                                                                                                                                                                                                             |
| PROGPS                                   | Yes                     | Added SDSNEXIT library.                                                                                                                                                                                                             |

| Member               | Supports concatenation? | Changed parms                                                                   |
|----------------------|-------------------------|---------------------------------------------------------------------------------|
| SHUTS0Wn             | No                      | Add system-specific system shutdown commands for VTAMAPPL.                      |
| SMFPRMPS<br>SMFPRMBS | No                      | Updated the SMF data set names.<br>System-specific log stream names.            |
| VATLSTPS             | Yes                     | Added work volume definition.                                                   |
| VTAMPS<br>VTAMBS     | No                      | Tailored VTAM00 for sysplex.<br>Similar to VTAM00 except in how RRS is started. |

To copy our parmlib members to the SYS1.IPLPARM and USER.PARMLIB data sets, submit job COPYPA22 in SYSPLEX.PARALLEL.CNTL.ZOS22.

All of the parmlib members that are described in this section are provided as part of this package and do not require changes unless you must adjust to fit something that is specific to your configuration.

For more information about the specific changes that we made to the parmlib members, see the next sections. Otherwise, see “Network definitions” on page 182.

### LOADPS member

This member is copied to the SYS1.IPLPARM data set rather than USER.PARMLIB. However, we include it here because it is logically related to the members of the USER.PARMLIB data set.

We wanted to consolidate to a single IEASYMxx member that can be used by multiple systems when you are running in a Parallel Sysplex. We also wanted the ability to control the IEASYSxx concatenation at the system level without needing to have system-specific LOADxx members. Therefore, we used the LOAD00 member as a base and changed the IEASYM parameter to point at IEASYMPS.

The LOADBS and LOADST and all of the other base sysplex-related members are described in “Changes in parmlib” on page 193 and “Required parmlib changes” on page 200.

**Important:** If you use a member other than LOAD00 when you load your ADCD system, you must update the supplied LOADPS member to reflect the changes that you made to LOAD00.

### IEASYMPS member

The original IEASYM00 member was designed for a single system environment. We wanted the ability to use a single IEASYMxx member for multiple systems, so we created a IEASYMPS member. Any symbols that are common to all members of the sysplex are at the top of the member in the common area. Then, we added filter statements that use the VM user ID that is associated with the z/OS system to set several system-specific symbols. One section is for user ID S0W1, and one section is for S0W2. If you want to add more systems later, you can add sections that are based on these examples. For more information about our IEASYMPS member, see “Sample IEASYMxx member for sysplex” on page 123.

Small adjustments were made to this member to reflect the data names and volume names from the December 2015 ADCD system.

## IEASYSPS member

A few parmlib members that must be changed for the sysplex environment. Therefore, we created an IEASYSPS member that is concatenated to the standard IEASYS00 member. The IEASYSPS member contains *only* the parameters that we needed to override for the sysplex environment.

In addition to pointing at the other changed members, the IEASYSPS member performs the following tasks:

- ▶ Overrides the page data set names to point at system-specific data sets that were created in “Creating system-specific system data sets” on page 164.
- ▶ Overrides the LOGREC parameter to point at the system-specific LOGREC data set.
- ▶ Overrides the VIO parameter to point at the system-specific STGINDEX data set.
- ▶ Overrides the GRS parameter to specify that GRS is used. The standard single-system ADCD environment specifies GRS=NONE because it cannot share with any other system.

## BPXPRMPS member

Various changes were made to this member to make the mount points and modes more supportive of a sysplex environment.

Compared to the z/OS 2.1 level of 2016 Sysplex Extensions, now a single sysplex-wide USERS file system exists, rather than a separate one for each system. Therefore, you can access your UNIX file system files transparently, regardless of which system you are logged on to.

For a complete list of the changes, compare the BPXPRMPS member of SYSPLEX.PARMLIB.ZOS22 to the BPXPRM00 member of ADCD.Z22A.PARMLIB.

## CLOCKPS member

One of the requirements for systems in a sysplex is that they have a common time source. z/VM provides this capability, but the CLOCKxx member must be updated to inform z/OS of the ID of the simulated external time source. This feature required a new parameter, SIMETRID, to be added. No changes were made to the ADCD-provided CLOCKxx member.

## COMMNDPS member

The ADCD-provided COMMNDxx members start VTAM by using the ATCSTR00 member of USER.VTAMLST. We did not want to change that member. (Our objective is to make it easy to switch back to monoplex mode.) Therefore, we set up a new ATCSTRPS member. However, that change required a change to the **S VTAM** command to point at that member instead.

If we created a COMMNDPS member that contains only the **S VTAM** command and concatenated it to the COMMNDWS member, the **S VTAM** command is issued twice, one time from the COMMNDWS member and again from the COMMNDPS member. Therefore, we copied the contents of COMMNDWS into COMMNDPS and changed only the **S VTAM** command to point at the ATCSTRPS member in VTAMLST.

## CONSOLPS member

One of the requirements of being in a sysplex is that every console must have a name that is unique within the sysplex. The ADCD-supplied CONSOL00 member uses a fixed name for its consoles: L700 and C908. If two systems are loaded with the CONSOL00 member, you have two consoles that are named L700 in the sysplex, which is not permitted.

Therefore, we copied the contents of CONSOL00 into CONSOLPS and changed the NAME keyword on the console definition to "&SYSNAME.L700". So, on system S0W1, the console is named S0W1L700, and on system S0W2, its console is named S0W2L700.

We also enabled OPERLOG by adding the OPERLOG keyword to the HARDCOPY DEVNUM parameter. We also added the HOLDMODE(YES) parameter to the DEFAULT statement, which allows you to hold the flow of messages on the console by pressing Enter. We felt that this change is useful if you are trying to debug errors during the early phases of an IPL before you can log on to TSO to review the syslog. To restart the flow of messages, press Enter again.

## **COUPLEPS member**

Because we are using unique CDSs for the sysplex environment, we needed a new COUPLExx member. We also took this opportunity to create a more robust XCF signaling infrastructure, so more transport classes and more signaling paths exist than in previous releases.

Also, we added definitions for the full set of CDSs (ARM, BPXMCDS, CFRM, LOGR, SFM, Sysplex, and WLM). Also, we included definitions for CTC connections between the systems. These connections are primarily intended for a base sysplex environment, where CF structures are not available. However, you can use the CTCs and the signaling structures in a Parallel Sysplex.

**Note:** Due to the longer system symbol names that are supported by z/OS 2.2, a new format ARM CDS is required. The 2016 Sysplex Extensions provides two sets of ARM CDSs: one set that will work with the May 2015 z/OS 2.1 level of AD CD and one set that will work with the z/OS 2.2 AD CD systems. The COUPLEPS member uses the z/OS 2.2 version of the CDSs. If you need to revert to the previous level, update the COUPLEPS member and perform a sysplex IPL.

## **DEVSUPPS member**

We added a new DEVSUPPS member with a single statement to enable the use of extended TIOTs for non-VSAM data sets. This addition is in line with the IBM preferred practice suggestions and eliminates an exception in the z/OS Health Checker.

## **GRSRNLPS member**

Because all z/OS systems that share your zPDT z/OS volumes are in the same sysplex, it is better to use GRS than RESERVE/RELEASE to protect the integrity of those volumes. To avoid problems with RESERVEs locking out volumes, we updated the GRS Resource Names List to indicate that all RESERVEs must be converted to ENQs.

## **LPALSTPS member**

Because we restored only the volume for one release of CICS (CICS TS 5.2), we created an LPALSTPS member to add only the link pack area (LPA) data set for that one release of CICS.

## **PROGPS member**

The load library that contains the DB2 subsystem parameters (SDSNEXIT) must be APF-authorized, so we created a PROG member with only that one entry and concatenated it to the AD CD-provided PROG members.

## SHUTS0W1 and SHUTS0W2 members

To make it easier to stop the system, we created system-specific versions of the SHUTALL member that is provided by ADCD. To shut down a system, run the **S SHUTSYS** command on the console of the system that you want to shut down.

## SMFPRMPS member

Unfortunately, the SMFPRMxx member does not support concatenation. Therefore, we copied the SMFPRM00 member, updated the SMF data set names, and added the system ID (SID) parameter.

## VATLSTPS member

Rather than update the ADCD-provided VATLST member, we created a member that contains one entry (for the WORK01 volume) and concatenated that member to the ADCD-provided VATLST00 member.

## VTAMPS member

This member is used by the VTAMAPPL program to start selected address spaces after the IPL completes. The VTAMPS member is based on the ADCD-provided VTAM00 member, except that it uses the &SYSNAME. system symbol where appropriate to tailor commands to the system on which they run.

## Network definitions

Before we describe PROCLIB, TCP, and VTAM changes, we describe our network setup and the relationships between the definitions.

Our configuration featured a single network adapter on our PC. Linux and the two z/OS guests use it to communicate to the outside world. For simplicity, we did not set up TCP on z/VM. Therefore, our configuration is similar to the configuration that is described in the chapter titled “Multiple guests in one instance” in *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

We use Virtual Network Computing (VNC) to log on to the Linux desktop remotely. (We can use this VNC to log on to VM from a remote window, if necessary.) As a result, the absence of TCP on z/VM was not a problem for us.

The various definitions feature the following relationships:

- ▶ The devmap file contains the definition for the OSA addresses and points the control units at the corresponding actual adapters (one for the tunnel to Linux, and one for the connection to the outside world). For more information, see the sample devmap file that is shown in Example A-1 on page 112.
- ▶ The VM directory entries for each z/OS guest contain DEDICATE statements for three OSA addresses for each connection (three each for the tunnel and the real network adapter). The addresses that are used for each system must be unique. Each system must have access to two sets of three consecutive OSA devices, and the first device number must be an even number. The device numbers must match the numbers that are specified in the devmap file. For more information, see the sample directory entries for S0W1 and S0W2 that are shown in Example A-4 on page 117 and Example A-5 on page 119.



- ▶ The VTAMLST data set contains one TRLE member for each system. Each member contains two TRLE definitions: one for the tunnel to Linux, and one for the real network adapter. Each definition includes a *portname*.
- ▶ The TCPPARMS data set contains the TCP PROFILE members (one for each z/OS). The DEVICE and LINK statements use the *portnames* that were defined for the OSA in the TRLE definitions in VTAMLST.

**Tip:** For this configuration to work, all of the systems that share a specific network adapter **MUST** use the *same portname*.

- ▶ To keep our VTAMLST simple, we use a system symbol for the name of the member that contains the OSA definitions. The value of the symbol is set in the IEASYMxx member and depends on the system. With this configuration, a single ATCCONPS member can be used by all of the systems in the sysplex.

The OSA section of our devmap file resembles the following example:

```
[manager]
name awsosa 0023 --path=A0 --pathtype=OSD --tunnel_intf=y
device 400 osa osa
device 401 osa osa
device 402 osa osa
device 408 osa osa
device 409 osa osa
device 40A osa osa

[manager]
name awsosa 0024 --path=f2 --pathtype=OSD
device 404 osa osa
device 405 osa osa
device 406 osa osa
device 40C osa osa
device 40D osa osa
device 40E osa osa
```

To determine the correct value of the path parameter on the name *awsosa* statement for *your* PC, run the **find\_io** command. The output from the command on our PC resembles the following example:

| Path | Interface Name | Current State   | MAC Address       | IPv4 Address | IPv6 Address |
|------|----------------|-----------------|-------------------|--------------|--------------|
| F2   | enp13s0        | UP, RUNNING     | 00:14:5e:57:c5:6a | 192.168.1.99 | *            |
| F1   | enp24s0        | UP, NOT-RUNNING | 00:14:5e:57:c5:6c | *            | * .          |
| A0   | tap0           | DOWN            | 02:a0:a0:a0:a0:a0 | *            | *            |
| A1   | tap1           | DOWN            | 02:a1:a1:a1:a1:a1 | *            | *            |
| A2   | tap2           | DOWN            | 02:a2:a2:a2:a2:a2 | *            | *            |
| A3   | tap3           | DOWN            | 02:a3:a3:a3:a3:a3 | *            | *            |
| A4   | tap4           | DOWN            | 02:a4:a4:a4:a4:a4 | *            | *            |
| A5   | tap5           | DOWN            | 02:a5:a5:a5:a5:a5 | *            | *            |
| A6   | tap6           | DOWN            | 02:a6:a6:a6:a6:a6 | *            | *            |
| A7   | tap7           | DOWN            | 02:a7:a7:a7:a7:a7 | *            | *            |

End of FIND\_IO

In this example, you can see that path F2 is the path that includes a status of UP, RUNNING. Therefore, F2 is specified on the path= statement in the devmap.

The VM directory entry for S0W1 contains the following statements:

```
DEDICATE 0400 0400
DEDICATE 0401 0401
DEDICATE 0402 0402
DEDICATE 0404 0404
DEDICATE 0405 0405
DEDICATE 0406 0406
```

The directory entry for S0W2 contains the following statements:

```
DEDICATE 0408 0408
DEDICATE 0409 0409
DEDICATE 040A 040A
DEDICATE 040C 040C
DEDICATE 040D 040D
DEDICATE 040E 040E
```

The VTAMLST TRLE definition for S0W1 contains the following statements:

```
OSATRL1 VBUILD TYPE=TRL
OSATRL1E TRLE LNCTL=MPC, READ=(0400), WRITE=(0401), DATAPATH=(0402),      X
                PORTNAME=PORTA,  X
                MPCLEVEL=QDIO
OSATRL2E TRLE LNCTL=MPC, READ=(0404), WRITE=(0405), DATAPATH=(0406),      X
                PORTNAME=PORTB,  X
                MPCLEVEL=QDIO
```

You can see that the device numbers from the VM directory entry are used for each entry, for example, 400 (READ), 401 (WRITE), and 402 (DATAPATH). The *portname* for that TRLE is PORTA. In this example, devices 400 - 402 are used for the tunnel to Linux. Devices 404 - 406 are used for the real network interface.

The TCP Profile member for S0W1 contains the following statements:

```
DEVICE PORTA MPCIPA
LINK ETH1 IPAQENET PORTA
HOME 10.1.1.2 ETH1
;
DEVICE PORTB MPCIPA
LINK ETH2 IPAQENET PORTB
HOME 192.168.1.81 ETH2
```

You can see that the same *portname* (for example, PORTA) that was specified on the TRLE definition is used on the DEVICE and LINK statements.

The TRLE definition for S0W2 contains the following statements:

```
OSATRL2 VBUILD TYPE=TRL
OSATRL2E TRLE LNCTL=MPC, READ=(0408), WRITE=(0409), DATAPATH=(040A),      X
                PORTNAME=PORTA,  X
                MPCLEVEL=QDIO
OSATRL3E TRLE LNCTL=MPC, READ=(040C), WRITE=(040D), DATAPATH=(040E),      X
                PORTNAME=PORTB,  X
                MPCLEVEL=QDIO
```

You see that although S0W2 uses different device numbers, *the portnames are the same as the definition for S0W1*. The Profile member for S0W2 contains the following statements:

```
DEVICE PORTA MPCIPA
LINK ETH1 IPAQENET PORTA
HOME 10.1.1.3 ETH1
;
; This second device is optional
DEVICE PORTB MPCIPA
LINK ETH2 IPAQENET PORTB
HOME 192.168.1.91 ETH2
```

**Important:** If you use different *portnames* on the two systems, the adapter will *not* activate on the second system. To our knowledge, this information is not documented elsewhere.

## Creating PROCLIB members

A few changes are required to PROCLIB members for the base system. Many procs are new for the 2016 Sysplex Extensions-provided CICS regions and DB2 subsystems. However, those changes are described in Chapter 5, “Sample DB2 data sharing environment” on page 95 and Chapter 6, “Sample CICSplex” on page 103. All changes and additions are made in the USER.PROCLIB data sets.

Now, run the COPYPROC job in SYSPLEX.PARALLEL.CNTL.Z0S22 to copy the procs. For more information about the changes that we made, continue reading this section. Otherwise, you can skip to “Creating TCPPARMS” on page 186.

### VTAMAPPL started tasks

VTAMAPPL is a basic automation tool that can be used to issue system commands and insert pauses when the commands are issued. The supplied VTAMAPPL proc provides the ability to override the member name, but not the data set name. We created a VTAMPS member in USER.PROCLIB because the ADCD-supplied VTAM00 member points at ADCD.Z22A.PARMLIB and we did not want to place any of our members in that data set. VTAMPS is started by using a command, such as - S VTAMPS,M=&VTAMAPPL, where &VTAMAPPL is a system symbol that indicates the VTAMAPPL member of USER.PARMLIB that is used.

The SHUTSYS proc also uses VTAMAPPL to shut down started tasks in an orderly and phased manner. The SHUTSYS proc uses the &SYSNAME. system symbol to select the correct SHUTxxxx parmlib member for that system.

The VTAMPS and the SHUTSYS procs are generic and can be used to start or stop either system.

### TCP/IP procedure

The ADCD-supplied TCP/IP procedure references the ADCD TCPPARMS data set, so we created a TCP/IP proc in USER.PROCLIB to point to the USER.TCPPARMS data set. This change also affects the non-sysplex AD system. The same PROFILE and DATA members are used by system S0W1, regardless of whether it is in monoplex or sysplex mode.

**Note:** The TCPIP member that is provided (and copied into your USER.PROCLIB data set) by the 2016 Sysplex Extensions is set up to use PROFILE and TCPDATA members with names that match the system name (S0W1P and S0W1D). The ADCD-provided members use a different naming convention (PROFILE and TCPDATA) because they were not intended for a sysplex environment.

## Creating TCPPARMS

**Important:** We mentioned in “JES2 MAS configuration” on page 175 that two steps are part of the implementation of the 2016 Sysplex Extensions where you must make changes that might affect your current, running ADCD system. This step is the second of those instances.

Because network implementations can vary greatly from one system to another, it is not possible to provide a set of definitions that work in every system. This step copies the members that *we* used to get our systems to work in a sysplex configuration. However, it must be treated as an example. You likely must make adjustments that are based on your specific network setup.

If an error occurs in your TCP definitions, you can still log on to the system by using one of the local z/VM windows and fix the problem from there.

The COPYTCPP job in SYSPLEX.PARALLEL.CNTL.Z0S22 copies four members to your USER.TCPPARMS data set. You likely need to adjust these parameters to match *your* LAN configuration. If you change the member names, make the corresponding changes to the TCP/IP JCL in USER.PROCLIB.

The following examples show the TCP/IP DATA members for the two systems. If you connect your z/OS systems to an external network, the DOMAINORIGIN values in these members must be changed. If you intend to use a name server, the NSINTERADDR values must be provided.

### Example 1: MEMBER S0W1D

The S0W1D member is used by base and Parallel Sysplex systems, as shown in the following example:

```
TCPIPJOBNAME TCPIP
S0W1: HOSTNAME S0W1
DOMAINORIGIN XXX.YYY.COM
DATASETPREFIX TCPIP
;NSINTERADDR xx.xx.xx.xx
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
ALWAYSUTO NO
```

### Example 2: MEMBER S0W2D

The S0W2D member is used by base and Parallel sysplex systems, as shown in the following example:

```
TCPIPJOBNAME TCPIP
S0W2: HOSTNAME S0W2
DOMAINORIGIN XXX.YYY.COM
DATASETPREFIX TCPIP
;NSINTERADDR xx.xx.xx.xx
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
ALWAYSUTO NO
```

## Other network-related changes

Certain products use the TCP Resolver service to determine the system IP name and IP address. For more information, see the section titled “z/OS Resolver” in *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205.

The ADCD December 2015 system does not supply a `/etc/hosts` file or a `/etc/resolver` file. In the absence of a resolver file, the Resolver service gets its information from the members pointed to by the `GBLRESOL` member of `ADCD.Z22A.TCPPARMS`. For simplicity, we replaced the `GBLIPNOD` member of that data set with the identically named member from the `SYSplex.TCPPARMS` data set. That member was modified to reflect the IP addresses that we use for our two systems, 192.168.1.81 and 192.168.1.91, which means that both systems can use the same member. If you use products that require IP name resolution, use our sample `GBLIPNOD` member as an *example* of the changes that are required.

Another change that you might need to make, depending on the IP address that you use for your system, is to update the `TCPIP.HOSTS.LOCAL` data set to reflect *your* IP names and addresses. Then, use the **MAKESITE** command to update the `SITEINFO` and `ADDRINFO` data sets from the information that you enter in the `HOSTS.LOCAL` data set.

## Creating VTAMLST members

We added several members to `USER.VTAMLST`. We provide sample `VTAMLST` members in `SYSplex.VTAMLST`. You can use the job `COPYVTAM` in `SYSplex.PARALLEL.CNTL.ZOS22` to copy the sample members to `USER.VTAMLST`. We updated the `ATCSTRPS`, `ATCCONPS`, and `OSATRLx` members. You might need to customize these members, depending on the VTAM applications that you want to activate automatically at VTAM startup, and on the device numbers that you used for your virtual OSA adapters.

**Tip:** If you change the `VTAMLST` members, the X symbols on the right side of certain lines are continuation characters and *must* be in column 72 of the statement.

The naming convention (the two-letter suffix) is not carried into the `TRL` names.

Also, we added members for CICS and DB2. Those changes are described in Chapter 6, “Sample CICSplex” on page 103 and Chapter 5, “Sample DB2 data sharing environment” on page 95.

## Customization for Parallel Sysplex complete

Now, all of the changes that are required to start the systems in Parallel Sysplex mode are complete. If this configuration is your target, proceed to “Bringing up your Parallel Sysplex” on page 188. If your objective is a base sysplex under z/VM, extra steps must be completed. For more information about those steps, see “Other steps for base sysplex under z/VM” on page 192. If your target environment is a base sysplex that is spread over two PCs, see “Implementing a base sysplex without z/VM” on page 195.

## Bringing up your Parallel Sysplex

At this point, all of the changes that are required to start the Parallel Sysplex are made.

However, to verify that you still can fall back to monoplex mode, we suggest that you shut down and then reload the z/OS system that is running under the BASEAD user ID. It is possible to load your system in the same way that you did previously, that is, by using the same IPL address and load parm.

After you verify that your current system still initializes successfully, the next step is to start your Parallel Sysplex. This process includes the following steps:

1. Shut down the BASEAD system and log off that virtual machine.
2. Run the following commands from the OPERATOR or MAINT user IDs to initialize the two coupling facility virtual machines:

```
XAUTOLOG CFCC1  
XAUTOLOG CFCC2
```

If you want to see the messages that are issued by the CFs, log on to CFCONSOL. The messages from both CFVMs are directed to that user ID.

Wait a couple of minutes to give the CFs a chance to initialize.

3. Log on to the S0W1 virtual machine. Check that you see messages, such as “HCPMFC2804I Message devices 1400-1403 defined and coupled to CFCC1” for each of the two CFs. These messages confirm that the CFVMs finished initialization. If you do not see these messages, log off and wait for a few minutes and then, log on again. After you see the messages about coupling to the CFs, run the following commands:

```
TERM CONMODE 3270  
IPL A80 LOADP OA82PSM1
```

This IPL command assumes that the sysres is on device A80 and that the A2SYS1 volume is on device A82. The “PS” in the load parm refers to member **LOADPS** in **SYS1.IPLPARM**.

Depending on the speed of your PC, it might take a few minutes for the nucleus initialization program (NIP) messages to appear on the console. After the messages start, you might be prompted to confirm that you want to initialize the sysplex (reply R 00,I to that WTOR). Sometime later, you might be prompted with WTOR IGGN505A, which indicates that the CICS TS 5.1 link library is not accessible. If you see that message and you do not want CICS TS 5.1, reply CANCEL.

The system must not present any more WTORs.

You might notice messages about failures to connect to log streams (specifically for RRS and OPERLOG). The reason for these messages is described in 3.4, “System logger” on page 29. To define the log streams and eliminate those messages, run the following jobs in data set **SYSplex.PARALLEL.CNTL.Z0S22**:

► **LOGREREP**

This job defines the log stream for LOGREC data. For information about enabling the use of this log stream, see 3.4.2, “LOGREC” on page 32.

► **LOGRHC**

This job defines the log stream that can be used by the z/OS Health Checker.

► LOGROPR

This job defines the OPERLOG log stream. For information about activating OPERLOG, see 3.4.3, “OPERLOG” on page 32.

► LOGRSMF

This job defines the SMF log stream. For more information about the use of SMF log streams, see 3.4.4, “System Management Facility” on page 33.

► LOGRRRS

This job defines the RRS log streams for Parallel Sysplex and base sysplex modes. For more information about RRS and considerations for its log streams, see 3.4.1, “Resource Recovery Services” on page 30.

**Tip:** All of these jobs end with a return code of 12 the first time that they are run, which is acceptable. The jobs are set up to delete and redefine the corresponding log streams. Because the log streams do not exist the first time that the job is run, the DELETE LOGSTREAM statement fails, but the other statements complete successfully.

Because so many log streams exist, checking the output from this job can be time-consuming. The quickest way to check the output is to run the **D LOGGER, L** command and check that all of the log streams that are shown in Example E-3 are defined in your system.

*Example E-3 Log streams that are defined by LOGRxxx jobs*

|                            |                  |                  |
|----------------------------|------------------|------------------|
| ATR.ADCDPL.ARCHIVE         | RRS_ARCHIVE_1    | 000001 IN USE    |
| ATR.ADCDPL.DELAYED.UR      | RRS_DELAYEDUR_1  | 000001 IN USE    |
| ATR.ADCDPL.MAIN.UR         | RRS_MAINUR_1     | 000001 IN USE    |
| ATR.ADCDPL.RESTART         | RRS_RESTART_1    | 000001 IN USE    |
| ATR.ADCDPL.RM.DATA         | RRS_RMDATA_1     | 000001 IN USE    |
| ATR.ADCDPL.RM.METADATA     | RRS_RM_META_1    | 000001 IN USE    |
| ATR.SOW1.ARCHIVE           | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW1.DELAYED.UR        | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW1.MAIN.UR           | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW1.RESTART           | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW1.RM.DATA           | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW1.RM.METADATA       | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW2.ARCHIVE           | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW2.DELAYED.UR        | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW2.MAIN.UR           | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW2.RESTART           | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW2.RM.DATA           | *DASDONLY*       | 000000 AVAILABLE |
| ATR.SOW2.RM.METADATA       | *DASDONLY*       | 000000 AVAILABLE |
| HZS.HEALTH.CHECKER.HISTORY | HZS_HEALTHCHKLOG | 000000 AVAILABLE |
| IFASMF.GENERAL             | IFASMF_GENERAL   | 000000 AVAILABLE |
| IFASMF.SOW1                | *DASDONLY*       | 000000 AVAILABLE |
| IFASMF.SOW2                | *DASDONLY*       | 000000 AVAILABLE |
| SYSPLEX.LOGREC.ALLRECS     | SYSTEM_LOGREC    | 000000 AVAILABLE |
| SYSPLEX.OPERLOG            | SYSTEM_OPERLOG   | 000000 AVAILABLE |

If a log stream is missing, see the output from the associated job and search for the corresponding `DEFINE LOGSTREAM NAME(log_stream_name)` statement. An example of part of the output from the LOGRRRS job is shown in Example E-4. In this case, the IXG007E message indicates that an error was encountered while the system processed the previous statement (LINE # 116). In this case, statement 116 was a `DEFINE LOGSTREAM` for `ATR.ADCDPL.RESTART`. The IXG007E message indicated that the `DATACLAS` definition in the `DEFINE LOGSTREAM` referred to a data class that was not defined.

*Example E-4 Sample LOGRRRS job output*

---

```

LINE #      CONTROL CARDS
  116      DEFINE LOGSTREAM NAME(ATR.ADCDPL.RESTART)
  117      STRUCTNAME(RRS_RESTART_1)
  118      LS_DATACLAS(LOGR24K)
  119      HLQ(LOGGER)
  120      LS_SIZE(10240)
  121      MODEL(NO)
  122      LOWOFFLOAD(20)
  123      HIGHOFFLOAD(80)
...
...
IXG005I LOGR POLICY PROCESSING LINE# 116
IXG007E A STORAGE MANAGEMENT SUBSYSTEM (SMS) ATTRIBUTE CLASS IS UNDEFINED.
IXG447I LOGR POLICY PROCESSING FOUND AN ERROR BUT CONTINUES.  RETCODE=00000008
RSNCODE=00000838
IXG003I LOGR POLICY PROCESSING ENCOUNTERED AN UNEXPECTED ERROR.  DIAGNOSIS
INFORMATION: 00000004 000003F6 0107001B 00000000

```

---

Also, you see many log streams whose names start with `START1`. These streams are CICS log streams that are predefined in the LOGR CDS that is included with this package. Therefore, you do not need to set them up.

The CICS log stream staging and offload data sets are allocated in the CICS storage group (CICS01 and CICS02 volumes). The OPERLOG, LOGREC, Health Checker, SMF, and RRS log stream data sets (all with a high-level qualifier of `LOGGER`) are allocated on the volumes that are mounted with the storage attribute (`A2SYS1` and `WORK01`, if you use the provided `VATLST00` and `VATLSTPS` members).

We suggest that you shut down the system (run the **S SHUTSYS** command) and then, start it.

After the restart, the system automatically starts by using the OPERLOG and RRS log streams.

If you want to use the LOGREC log stream, run the **SETLOGRC LOGSTREAM** command or include that command in the VTAMPS member.

If you want to use the SMF log stream, edit the SMFPRMPS member and change the first line to say `ACTIVE` rather than `NOACTIVE`, and the second line to say `RECORDING(LOGSTREAM)` rather than `RECORDING(DATASET)`. Then, activate the updated member by issuing the **SET SMF=PS** command.

If you want Health Checker to send its output to a log stream, run the **F HZSPROC,LOGGER=ON,LOGSTREAMNAME=HZS.HEALTH.CHECKER.HISTORY** command. Also, you can update the HZSPRMAD member of the ADCD.Z22A.PARMLIB data set so that the log stream is enabled automatically. For more information about the use of the HZSPRMxx member, see *IBM Health Checker for z/OS User's Guide*, SC23-6843.



After system S0W1 initializes successfully, complete the following steps to start the other sysplex member:

1. Log on to S0W2 (the initial password is S0W2).
2. Run the following commands to load the S0W2 system (0A82PSM1):

```
TERM CONMODE 3270
IPL A80 LOADP 0A82PSM1
```

The S0W2 system is activated without requiring further intervention.

## Parallel sysplex implementation checklist

Because the implementation of the 2016 Sysplex Extensions Parallel Sysplex requires many steps and it is important that the steps are completed in the correct sequence, we provide the checklist in Table E-5. You can use this checklist to track your progress and ensure that you do not accidentally omit or skip any steps.

*Table E-5 2016 Sysplex Extensions Parallel Sysplex implementation checklist*

| Task | Description                                                                          | Done? |
|------|--------------------------------------------------------------------------------------|-------|
| 1    | Install base ADCD z/OS system.                                                       |       |
| 2    | Download/Install base ADCD z/VM system.                                              |       |
| 3    | Download and gunzip five 2016 Sysplex Extensions volumes.                            |       |
| 4    | Allocate five new volumes by using the <b>a1cckd</b> command.                        |       |
| 5    | Create a backup of z/VM and z/OS volumes before you make changes.                    |       |
| 6    | Add 2016 Sysplex Extensions volumes to devmap.                                       |       |
| 7    | Add commands to devmap to start x3270 sessions.                                      |       |
| 8    | Restart zPDT and IPL z/VM.                                                           |       |
| 9    | Update VM directory to add the WRKALLEG keyword.                                     |       |
| 10   | Log on to BASEAD and IPL z/OS in monoplex mode.                                      |       |
| 11   | Initialize and vary online the five new z/OS volumes.                                |       |
| 12   | Run the IMPCONN job to IMPORT CONNECT user catalogs.                                 |       |
| 13   | Run the VSAMS0W2 job to create S0W2 system VSAM data sets.                           |       |
| 14   | Run the VSAM1A, VSAM1B, and VSAM1C jobs to reallocate S0W1 page data sets.           |       |
| 15   | Run the OMVSCLON job to create S0W2 copies of OMVS file systems.                     |       |
| 16   | Run the MKDIR job to create S0W2 mount point for S0W2 system-specific file systems.  |       |
| 17   | Run the HZSALLCP job to allocate the S0W2 Health Checker data set.                   |       |
| 18   | Run the RECATLG job to recatalog 2016 Sysplex Extensions CDSs in the master catalog. |       |
| 19   | Add the S0W2 system name and sysplex name to SMS config.                             |       |

| Task | Description                                                                                                                                            | Done? |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| 20   | Add the following SMS data classes for Logger and DB2: <ul style="list-style-type: none"> <li>▶ LOGR4K</li> <li>▶ LOGR24K</li> <li>▶ DPDGDC</li> </ul> |       |
| 21   | Create the SMS storage class for CICS and DB2 data sets: PSADDON.                                                                                      |       |
| 22   | Create the SMS storage groups for CICS and DB2 data sets: <ul style="list-style-type: none"> <li>▶ CICFILES</li> <li>▶ DB2FILES</li> </ul>             |       |
| 23   | Update the ACS routines to assign new data and storage classes and storage groups, translate routines, and activate the updated SCDS.                  |       |
| 24   | Run the ACDS5 job to change the SHAREOPTIONS of the SMS CDSs.                                                                                          |       |
| 25   | Update the MASDEF statements in the JES2PARM member.                                                                                                   |       |
| 26   | Run the COPYPA22 job to copy the 2016 Sysplex Extensions parmlib members to USER.PARMLIB and SYS1.IPLPARM.                                             |       |
| 27   | Run the COPYPROC job to copy 2016 Sysplex Extensions Proclib members to USER.PROCLIB.                                                                  |       |
| 28   | Run the COPYTCPP job to create new TCP PROFILE and TCPDATA members in USER.TCPPARMS.                                                                   |       |
| 29   | Run the COPYVTAM job to copy VTAMLST members to USER.VTAMLST.                                                                                          |       |
| 30   | Reload the BASEAD system in monoplex mode.                                                                                                             |       |
| 31   | Logoff BASEAD.                                                                                                                                         |       |
| 32   | AUTOLOG CF virtual machines from the CFCONSOL user.                                                                                                    |       |
| 33   | Log on to S0W1 and IPL by using the new Parallel Sysplex LOADPS member.                                                                                |       |
| 34   | Run the LOGREREP job to define the LOGREC log stream.                                                                                                  |       |
| 35   | Run the LOGROPR job to define the OPERLOG log stream.                                                                                                  |       |
| 36   | Run the LOGRRRS job to define the RRS log streams.                                                                                                     |       |
| 37   | Run the LOGRSMF job to define the SMF log stream.                                                                                                      |       |
| 38   | Run the LOGRHC job to define the Health Checker log stream.                                                                                            |       |

## Other steps for base sysplex under z/VM

If your target environment is a base sysplex under z/VM, other steps are required. Certain messages can be ignored when you start a base sysplex under z/VM. In this section, we list those messages and describe the reasons why you are receiving them.

### Changes to z/VM directory

Add definitions for virtual CTCs that are used for XCF communication between the members of the sysplex.

You can define these definitions dynamically in each virtual machine that connects through the CTCs (S0W1 and S0W2 in this case) by running the VM **DEFINE** command.

However, we believe that it is less work to add the definitions to the VM directory. To add these definitions, add the following statements to the directory entries for S0W1 and S0W2. (Use the same definitions for both virtual machines.)

```
SPECIAL E20 CTCA
SPECIAL E21 CTCA
SPECIAL E40 CTCA
SPECIAL E41 CTCA
SPECIAL E42 CTCA
SPECIAL E43 CTCA
```

After you make these changes, save the USER DIRECT C file and run the command **DIRECTXA USER DIRECT C** to update the directory.

You notice that these statements do not provide information about the CTC devices and the other devices to which they connect. For more information, see “Bringing up your base sysplex” on page 194.

## Changes in parmlib

From a parmlib perspective, the only thing that *must* be different in a base sysplex compared to a Parallel Sysplex is that GRS Star cannot be used in a base sysplex. The GRS parameter in IEASYSxx must specify something other than “STAR”.

This section describes the members that must be used when you run in base sysplex mode under z/VM.

### LOADxx member

Because you use the same z/VM virtual machines for base and Parallel Sysplex, you need a way to differentiate between base and Parallel Sysplex. This distinction is made by using a different LOADxx member when you run in base sysplex mode. The LOADBS member points at IEASYMBS.

The LOADBS member is copied into SYS1.IPLPARM when you run the BASPARM2 job.

### IEASYMBS member

The IEASYMBS member is identical to the IEASYMPS member, with the following exceptions:

- ▶ The IEASYMBS member sets the VTAMAPPL symbol to be VTAMBS, which is used by the **START VTAMPS** command in COMMNDPS.
- ▶ The IEASYMBS member adds one more member (IEASYSBS) to the sysparm concatenation.

The IEASYMBS member is copied into USER.PARMLIB when you run the BASPARM2 job.

### IEASYSBS member

The IEASYSBS member contains one parameter (the GRS=TRYJOIN statement). All of the other IEASYSxx parms are picked up from IEASYS00 in ADCD.Z22A.PARMLIB or IEASYSPS in USER.PARMLIB.

## SMFPRMBS

The SMFPRMBS member is identical to the SMFPRMPS member except that it specifies system-unique log streams rather than a single log stream that is shared by both systems.

The SMFPRMBS member is copied into USER.PARMLIB when you run the BASPARM2 job.

## VTAMBS member

The VTAMBS member starts each RRS with a different group name.

Now, run job BASPARM2 in SYSPLEX.PARALLEL.CNTL.Z0S22 to copy these members into USER.PARMLIB and SYS1.IPLPARM.

## Bringing up your base sysplex

Small changes to the IPL process are necessary if you are bringing up the systems in base sysplex mode.

Because the same virtual machines are used for Parallel Sysplex and base sysplex modes, the systems automatically use the CFs if they are running and connected to the S0W1 and S0W2 machines. Therefore, you must ensure that the coupling facility virtual machines are *not* logged on. Use the OPERATOR or MAINT ID and run a Q N command. If CFCC1 or CFCC2 appears in the list, run the following commands to stop it. (First, shut down any z/OS systems that might use it.)

```
FORCE CFCC1
FORCE CFCC2
```

When you run in a base sysplex under z/VM, the S0W1 and S0W2 systems communicate by using the CTCs that you defined in “Changes to z/VM directory” on page 192. Before they can be used, you must connect the CTCs to each other. Although this connection must be made only from one of the virtual machines, S0W1 and S0W2 must be logged on before you can issue these commands. After both machines are logged on, run the following commands in S0W1:

```
COUPLE E20 S0W2 E21
COUPLE E21 S0W2 E20
COUPLE E40 S0W2 E41
COUPLE E41 S0W2 E40
COUPLE E42 S0W2 E43
COUPLE E43 S0W2 E42
```

Now, you are ready to start the systems in base sysplex mode. To start the system in this mode, run the following command in one of the systems:

```
TERM CONMODE 3270
IPL A80 LOADP 0A82BSM1 (This command loads the system by using the LOADBS
member of SYS1.IPLPARM.)
```

Any time that you load the first system in a sysplex, you might receive a WTOR asking whether you want to initialize the sysplex. If you see that message, reply R 00,I. Expect to see messages when the system initializes that indicate that the system cannot access CF CFCC1 or CF CFCC2. You see these messages because the base sysplex uses the same CDSs (including the coupling facility resource management (CFRM) data set) as the Parallel Sysplex. Because you are starting in base sysplex mode, you can ignore those messages.

I When the first system finishes its initialization, log on to TSO and browse back through syslog to verify that everything started as expected. If everything is OK, submit the LOGRRRS job in SYSPLEX.PARALLEL.CNTL.ZOS22 to allocate the DASDONLY log streams that are used by RRS on each system.

The OPERLOG and LOGREC log streams include fixed names. The base sysplex supports only DASDONLY log streams, and DASDONLY log streams cannot be shared across systems. In a base sysplex, a maximum of one system can use OPERLOG or LOGREC, which negates much of the value of those two functions. As a result, we did not provide jobs to create DASDONLY versions of the OPERLOG or LOGREC log streams.

Address any unexpected messages. Then, load the other system by using the same IPL address and load parameters. When that system initializes, you see messages that indicate that XCF is connecting to the other system by using the CTC addresses that you specified in the **COUPLE** commands.

## Implementing a base sysplex without z/VM

The other option for running a base sysplex is more complex because it involves two PCs, cross-PC file sharing, Server Time Protocol (STP), and the use of a network to connect the PCs.

Base sysplex operation does not involve z/VM, but it involves other details that must be set up correctly.

The base sysplex configuration involves multiple Linux machines. A Linux file sharing function, such as Network File System (NFS), provides coordinated DASD functions at the Linux cache level. z/OS uses GRS functions, through the CTC links among z/OS systems, to coordinate data set sharing and other serialization. A synchronized time-of-day function is provided by the zPDT STP function<sup>5</sup>.

---

<sup>5</sup> The zPDT STP function requires zPDT General Availability (GA) 6 or later.

In the configuration that is shown Figure E-4, all of the emulated DASD for all z/OS partners are placed on one Linux machine (PC2), which becomes the Linux file server. In practice, the emulated DASD can be spread over several Linux machines, all of which can work as file sharing servers and clients. In Figure E-4, the solid lines between the PCs and the switch represent the physical connections between the PCs. The dotted lines represent logical connections. In practice, all of the traffic (STP, XCF, NFS, and so on) goes over the Ethernet connections.

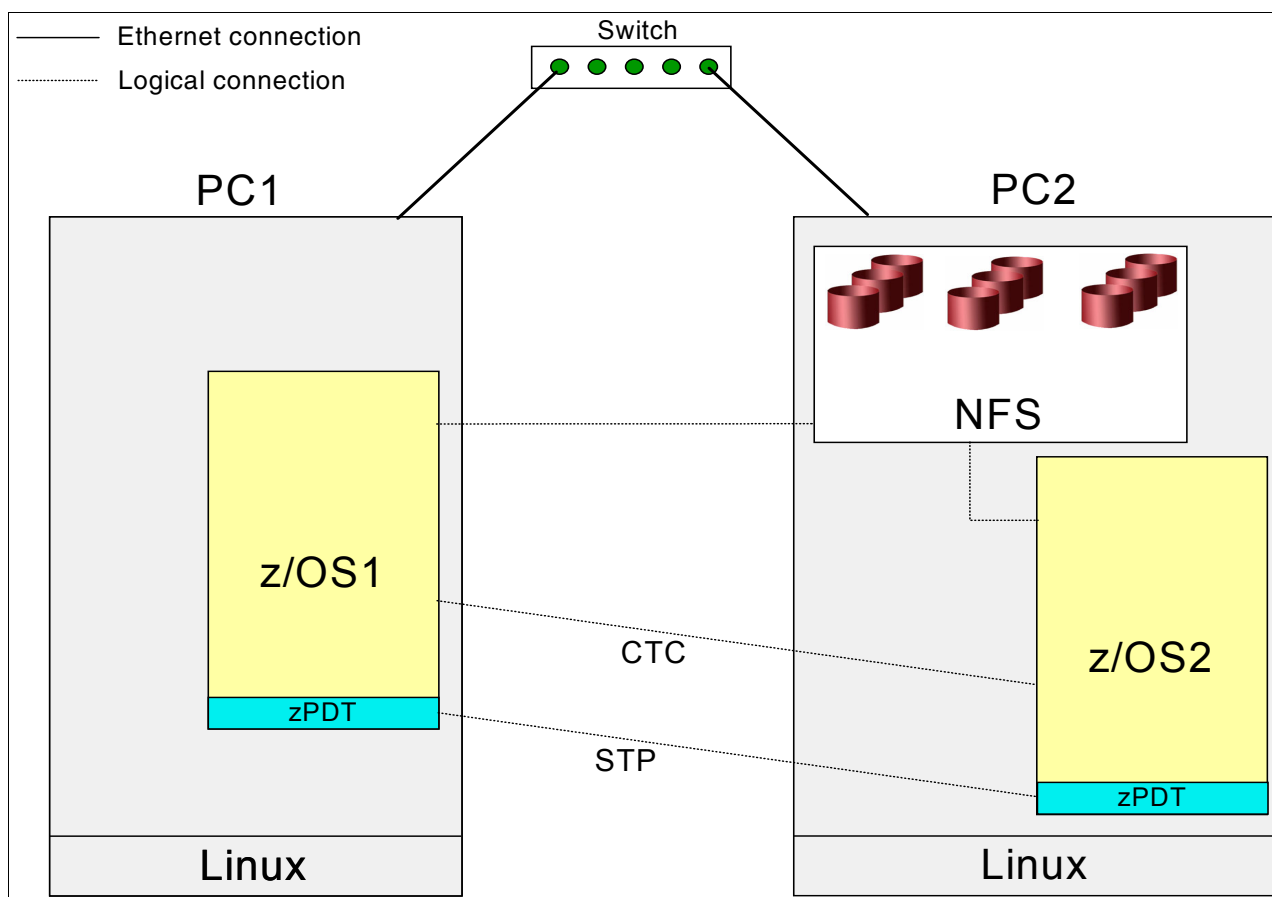


Figure E-4 Base sysplex without z/VM

## Setting up and starting STP

One of the fundamental requirements of any sysplex is that all of the systems in the sysplex must have a common time source. When you run a parallel or base sysplex under z/VM, z/VM provides the common time source. However, if the sysplex is spread over more than one PC, you need another mechanism, which is STP, to synchronize the z System clocks.

The zPDT implementation of STP timer facilities is described in the STP chapter of *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

To use STP, you must edit and configure the `/usr/z1090/bin/CCT_data` file on each PC. This file differs slightly on each member of the base sysplex because the IP address (or domain name) differs for each member.

In addition to the CCT\_data file update, you must update the devmap file to add the [stp] section. The STP chapter of *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, includes clear, step-by-step instructions to set up these files. An example of the definitions from our devmap file is shown in Example E-5.

*Example E-5 Sample stp definition statements from devmap file*

---

```
[stp]
ctn 00000000F1F0F9F0
node 1 W520 *      # For the W510 system, the asterisk is
node 2 W510        # moved to the second node statement
```

---

After you configure this file on each sysplex member, verify that all of the members of the base sysplex have TCP/IP connectivity to each other. This connection can be verified by using simple **ping** commands. Now, run the zPDT **stpserverstart** command on each member. In principle, running this command causes the STP functions to start automatically whenever you start Linux. The **stpserverquery** command can be used to verify that STP is functioning as you expect.

## Defining CTCs for XCF communication

Another requirement for a base sysplex or Parallel Sysplex is that the systems in the sysplex must communicate over XCF to the other sysplex members. If a system is trying to join the sysplex but it cannot communicate over XCF to the other sysplex members, it is not allowed to join the sysplex.

Because a base sysplex does not have CF structures for XCF communication, it uses CTCs to communicate. In a zPDT sysplex that does not use z/VM, zPDT emulates the CTC function by using TCP/IP to communicate between the PCs. For more information and sample configurations, see the “Channel-to-channel” chapter of *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248205.html?Open>

When z/VM is used to emulate the CTC function, you use the **COUPLE** command to instruct the two CTC devices to connect. Similarly, with zPDT, you need to provide each zPDT with the following information:

- ▶ Device number for each CTC device
- ▶ The remote CTC to which it needs to connect

Because TCP/IP is used for the CTC communication, you use the IP address of the remote CTC by using the IP address of the remote Linux system. You specify the IP address of the *Linux* system, *not* the address of the z/OS system.

Sample definitions for two PCs are shown in Example E-6 and Example E-7 on page 198.

*Example E-6 Sample devmap CTC definitions on PC1*

---

```
[manager]          #CTC definitions for PC1 in base sysplex
name awsc tc 0088
device E40 3088 3088 ctc://192.168.1.99:4000/E41
device E41 3088 3088 ctc://192.168.1.99:4001/E40
```

---

#### Example E-7 Sample devmap CTC definitions on PC2

---

```
[manager]          #CTC definitions for PC2 in base sysplex
name awsctc 0088
device E40 3088 3088 ctc://192.168.1.230:4001/E41
device E41 3088 3088 ctc://192.168.1.230:4000/E40
```

---

As described in “Bringing up your base sysplex” on page 194, the VM **COUPLE** commands connected CTC E40 in S0W1 to E41 in S0W2. XCF CTCs are unidirectional. That is, each CTC is a PATHIN (which means that XCF messages are received on that device) or a PATHOUT (which means that XCF messages are sent on that device). Therefore, you must always connect a PATHIN device to a PATHOUT device. The CTCs are defined by using the following statements in the COUPLEPS member:

```
/* DEFINE XCF TRANSPORT CLASS PATHS FOR BASE SYSPLEX */
PATHIN  DEVICE(E20)                                /* DEFAULT TC */
PATHIN  DEVICE(E40)                                /* DEF8K TC */
PATHIN  DEVICE(E42)                                /* DEF61K TC */
PATHOUT DEVICE(E21)                                CLASS(DEFAULT)
PATHOUT DEVICE(E41)                                CLASS(DEF8K)
PATHOUT DEVICE(E43)                                CLASS(DEF61K)
```

On system S0W1, you connect E20 (a PATHIN) to E21 (a PATHOUT) on S0W2, and E21 (a PATHOUT) to E20 (a PATHIN) on S0W2. Therefore, the devmap file needs to connect the CTCs devices to the opposite type of CTC on the target system. In the statements that are shown in Example E-6 on page 197, E40 on this system is connected to E41 on the remote system.

Also, the port number for this connection is 4000. If you review the definition of device E41 on the other PC, you see that it specifies device number E40 as the target CTC device. It also specifies the same port number as the E40 definition on PC1.

Similarly, E41 on PC1 is connected to E40 on PC2. But, this connection uses a different port number (4001) in this case. If you have multiple CTC connections, each one *must* use a different port number.

When you run the **awsstart** command on the second PC, you do not see any messages about the CTC connection unless a problem exists. If you want to check that the two PCs are connected, run the zPDT **awsstat** command.

## Sharing zPDT DASD across PCs

Our base sysplex configuration requires a shared DASD operation for all of the emulated 3390 volumes. Setting up Linux shared directories is beyond the scope of this book.



The process that is used to share zPDT DASD across PCs includes the following steps on an openSUSE 13.1 system:

1. On our openSUSE Linux system that holds all of the emulated 3390 volumes, we used YaST functions to create an NFS server with the following parameters:
  - NFS server: Start
  - Firewall: (Firewall is disabled; you might request opening a port in your firewall.)
  - Enable NFSv4: yes
  - Enable GSE security: no
  - Directories to export: /z (This directory contained all of our emulated volumes.)
  - Host wild card: 192.168.1.230 (the address of our second system Linux)
  - Options: fsid=0,crossmnt,rw,root\_squash,sync,no\_subtree...
2. On our second Linux system, we defined mount point /z2 (owned by user ibmsys1) with read and write access by everyone. We selected this mount point to be different from the /z mount point that we normally use for local volumes. (We did not want any confusion about which PC's disks we were referring to.) We then used YaST services to define an NFS client with the following parameters:
  - NFS shares:
 

| Server       | remote-directory | mount-point | NFS type | options     |
|--------------|------------------|-------------|----------|-------------|
| 192.168.1.99 | /z               | /z2         | nfs      | rw,defaults |
  - Enable NFS4: yes
  - NFSv4 Domain Name: localdomain (Use the default value unless you know better.)
  - Enable GSS Security: no
  - Firewall settings: (The firewall is disabled.)
3. In principle (perhaps after you reboot both systems), you will see the shared volumes from the second Linux system by running a command, such as `ls /z2`. In practice, we ran the following command on the second system (working as *root*) to see the shared volumes from the second system:
 

```
# mount -t nfs4 192.168.1.80:/ /z2
```

(We are not NFS experts and considerable experimentation at times was needed to make the sharing work.) Be certain that the *rw* (read/write) option is present on the NFS server and client.
4. Be certain that the `--shared` option is present on the `awsckd` device manager statements in the devmaps on both systems. (This statement enables RESERVE/RELEASE emulation.) Be certain that the devmap for the second system contains the correct mount points (/z2 in our example) for all of the emulated disk volumes.

When the cross-PC file sharing is working, update the devmap file for the second PC to change the file names to include the z2.

**Note:** Although functional and sufficient for function testing, this configuration might not deliver the levels of performance that you require. If you can make a larger financial investment to improve the performance of a multi-PC zPDT configuration, see Appendix C, “Alternative disk configuration” on page 141.

## Required parmlib changes

In addition to the infrastructure changes (STP, CTCs, and shared DASD) that are required to span the base sysplex across more than one PC, a few changes to parmlib are required. The systems must be instructed to use STP for their time synchronization, which means that the CLOCKxx member must be updated. (The ADCD default is to disable ETRMODE and STPMODE.) You also need to configure GRS for Ring mode rather than Star mode. Also, each RRS must be started with a different group name to stop them from trying to share log streams.

### LOADST member

You need a way to instruct the system that it must come up in base sysplex, not under z/VM mode. You also need a way to set system-specific system symbols. When you run under z/VM, you can use the VM user ID as a filter in the IEASYMxx member. However, this sysplex is not running under VM so that method is not an option.

However, you *can* use the LPAR name. When you run under zPDT, the instance name (which is set to the Linux user ID that started zPDT) is loaded into the LPAR name field. Therefore, you can filter by using the LPARNAME field in your IEASYMxx member.

**Important:** When you run a base sysplex that spans two PCs, you must start zPDT with a different Linux user ID on each system. Failing to use this configuration can result in problems because it appears that the same system is being loaded twice into the same sysplex.

Consider the following points:

- ▶ The LOADST member is identical to LOADPS, except that it points at IEASYMST.
- ▶ The LOADST member is copied into SYS1.IPLPARM when you run the STPPARM2 job.

### IEASYMST

IEASYMST is identical to IEASYMPS except that it filters based on the system name, and it specifies that the system uses IEASYS00, IEASYS05, and IEASYS06 when it initializes.

**Note:** It is likely that you must customize the IEASYMST member. It is set up on the assumption that the two z/OS systems include instance names of IBMSYS1 and IBMSYS2. If the Linux user IDs that you use to start zPDT have different names, you must adjust this member.

The IEASYMST member is copied into USER.PARMLIB when you run the STPPARM2 job.

### IEASYSST

The IEASYSST member contains only the following parameters:

- ▶ CLOCK: This parameter is set to ST to use the CLOCKST member.
- ▶ GRS: This parameter is set to TRYJOIN because it is a base sysplex.

The IEASYSST member is copied into USER.PARMLIB when you run the STPPARM2 job.

## CLOCKST member

The ADCD-provided CLOCK00 member specifies NO for STPMODE and ETRMODE. Because a sysplex that spans more than one PC requires STP, you must use a CLOCKxx member that specifies STPMODE YES. Consider the following points:

- ▶ The ADCDST member is a complete replacement for the CLOCK00 member.
- ▶ The CLOCKST member is copied into USER.PARMLIB when you run the STPPARM2 job.

## SMFPRMBS

The SMFPRMBS member is identical to the SMFPRMPS member except that it specifies system-unique log streams rather than a single log stream that is shared by both systems.

The SMFPRMBS member is copied into USER.PARMLIB when you run the STPPARM2 job.

## VTAMBS

The VTAMBS member starts each RRS with a different group name, which avoids both RRSs trying to share the log stream. (Cross-system log stream sharing is not possible in a base sysplex.)

Run the STPPARM2 job in SYSPLEX.PARALLEL.CNTL.ZOS22 to copy these members into USER.PARMLIB and SYS1.IPLPARM.

## Loading the systems

All of the pieces that are required to start a base sysplex without z/VM are in place now. We strongly suggest that you allow the first system to complete the entire startup process before you load the second system. We also advise that you load the system that is in the same PC as the NFS server first.

With Linux sharing enabled, certain operations are slower. For example, starting the second system (NFS client) with no connection to the NFS server causes the start function to be *much* slower. Loading the local system first might load the Linux cache with data that is used by the IPL, which provides a small benefit for later loads by remote systems. Even allowing for that benefit, a load of the second system is much slower than a load of the first system.

Both systems can be loaded by using the **ip1 a80 parm 0A82STM1** command.

Any time that you load the first system in a sysplex, you might receive a WTOR that asks whether you want to initialize the sysplex. If you receive that message, reply R 00,I. You can expect to see messages when the system initializes that indicate that the system cannot access CF CFCC1 or CF CFCC2. You see these messages because the base sysplex uses the same CDSs (including the CFRM data set) as the Parallel Sysplex. Because you are coming up in base sysplex mode, you can ignore those messages.

When the first system finishes its initialization, log on to TSO and browse back through the syslog to verify that everything started as expected. If everything is OK, submit job LOGRRRS in SYSPLEX.PARALLEL.CNTL.ZOS22 to allocate the DASDONLY log streams that are used by RRS on each system.

The OPERLOG and LOGREC log streams have fixed names. Base sysplex supports only DASDONLY log streams, and DASDONLY log streams cannot be shared across systems. In a base sysplex, a maximum of one system can use OPERLOG or LOGREC, which negates much of the value of those two functions. As a result, we did not provide jobs to create DASDONLY versions of the OPERLOG or LOGREC log streams.

Address any unexpected messages. Then, load the other system by using the same IPL address and load parameters. When that system initializes, you see messages that indicate that XCF is connecting to the other system by using the CTC addresses that you specified in the devmap file.

## Operating your sysplex

After your sysplex finishes its initialization, operating the systems is similar to running in a normal monoplex environment, except for all of the extra functions that are available.

For example, with a single system, you are concerned that the system might not come up after a change. With a sysplex, you can stay logged on to one system while you load the other system. If a problem occurs, you have a running, fully functional system that you can use to address whatever is stopping the system from loading.

## Managing your x3270 sessions

At least five 3270 sessions are needed for the sysplex. If you try to squeeze all sessions onto one window, the window can be cluttered, which makes it easy to enter commands in the wrong 3270 session.

We suggest that you use two Linux desktop workspaces with the arrangement that is shown in Figure E-5.<sup>6</sup> The 3270 windows that relate to the S0W2 system are in the second workspace, which helps to prevent confusion when multiple 3270 windows are used. The 3270 addresses that are shown in Figure E-5 correspond to the 3270 device addresses that are defined in the devmap file. An example of the interactions with the various x3270 sessions is provided in Appendix D, “Sample IPL flow for sysplex under z/VM” on page 145.

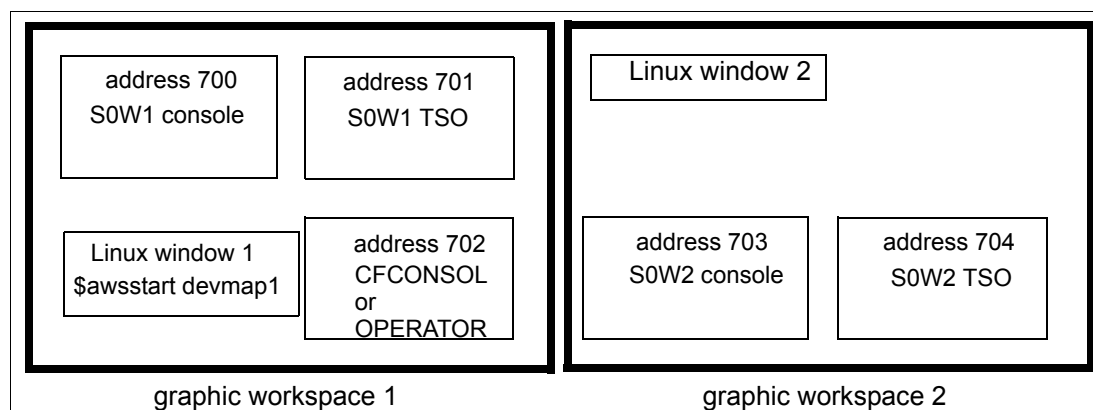


Figure E-5 Sample layout for x3270 sessions

<sup>6</sup> Another option is to use a separate PC for the 3270 sessions that relate to the S0W2 system.

## Post-loading messages

If you use the S0W1 system in monoplex mode, you might see unusual messages during startup because it appears to z/OS that systems in different logical partitions (LPARs) are trying to use the same page data sets, as shown in the following example:

```
ILR030A PAGE DATA SET MAY BE IN USE
ILR031A REPLY 'U' TO PREVENT ACCESS, 'CONTINUE' TO ALLOW USE OF SYS1.....
r 00,continue <---your reply
```

In this case, ensure that only one system is using those page data sets. If only one system is using those page data sets, reply `continue`, as shown in the example.

## SMF considerations

The default mode of operation for an ADCD system is for SMF collection to be disabled. In a pure development system (especially with one user or a few users), you might discover that you rarely use the SMF data. However, if you want to collect SMF records, the following options are available, depending on how you are running your systems:

- ▶ In a Parallel Sysplex, you can share a single log stream between both systems. This configuration is convenient because all of your SMF records are in a single location. You can leave the records in the log stream and request that system logger automatically delete them after a retention period that is specified on the log stream. (The SMF log stream that is delivered with the 2016 Sysplex Extensions has a retention period of 14 days, but you can change this setting at any time by updating the log stream definition.) The log stream for Parallel Sysplex mode is called `IFASMF.GENERAL`.
- ▶ In a base sysplex, a log stream cannot be shared across multiple systems. However, you can still write the SMF data from each system to its own log stream. This configuration provides most of the benefits of the single log stream, except that you no longer have a single repository of all SMF data. The log streams that we defined for base sysplex mode are called `IFASMF.S0W1` and `IFASMF.S0W2`.
- ▶ In any mode (monoplex, base sysplex, or Parallel Sysplex), you can use the traditional `SYS1.MANx` VSAM data sets.

The 2016 Sysplex Extensions provides two `SMFPRMxx` members: `SMFPRMBS` and `SMFPRMPS`. Both members contain definitions for `SYS1.MANx` data sets with data set names that contain the system name (for example, `SYS1.S0W1.MAN1`). The `SMFPRMBS` member is used when the system is loaded by using a loadparm that indicates that it is to come up in base sysplex mode. That member contains definitions for the system-specific log stream name. The `SMFPRMPS` member is used when the system is brought up in Parallel Sysplex mode, and it contains a definition for the shared log stream.

By default, SMF recording is disabled. If you enable recording, the default is to use the `SYS1.MAN` data sets. If you want to use the log streams instead, update the appropriate `SMFPRMxx` member to say `RECORDING(LOGSTREAM)` and run the `SET SMF=xx` command to get SMF to use the parms in the named member.

## Sysplex policies

The sysplex policies that are provided by the 2016 Sysplex Extensions are sufficient to get the sample Parallel Sysplex (complete with CICSplex and DB2 data sharing) up and running. Nevertheless, it is likely that you will want to change these policies over time. We suggest that you create a local copy of the provided jobs and make any changes that you want in those local copies. The sources for the provided policies are in the following members in `SYSplex.PARALLEL.CNTL.ZOS22`:

|                 |                                              |
|-----------------|----------------------------------------------|
| <b>POLARM1</b>  | Automatic Restart Manager policy             |
| <b>POLCFRM1</b> | Coupling Facility Resource Management policy |
| <b>POLSFM1</b>  | Sysplex Failure Management policy            |
| <b>LOGRCICS</b> | CICS model log streams definition            |
| <b>LOGREREP</b> | LOGREC log stream definition                 |
| <b>LOGRHC</b>   | z/OS Health Checker log stream definition    |
| <b>LOGROPR</b>  | OPERLOG log stream definition                |
| <b>LOGRRRS</b>  | RRS log streams definition                   |
| <b>LOGRSMF</b>  | SMF log streams definition                   |

If you want to change the ARM, CFRM, or SFM policies, we strongly advise that you give the new policies a different name than the active policy. The policy name is specified on the `NAME` parameter on the `DEFINE POLICY` statement in the policy. When you activate the new policy, the policy name is specified in the **SETXCF START,POLICY** command. By using a different name, you can fall back to the old policy if a problem occurs with the new policy.

The sources for the policies that are provided by the 2016 Sysplex Extensions are in Appendix B, “Sysplex couple data sets and policies” on page 127.

## Shutdown

Although you might be tempted to shut down your system by ending it, this process can result in data loss and unexpected messages the next time that the system is loaded. For these reasons, we strongly suggest that you take the time to complete an orderly shutdown of your system. This suggestion applies to any system, but *especially* to a system in a zPDT Parallel Sysplex because so many levels of virtualization exist. Buffering is used in both z/OS and in Linux to improve performance. If you simply kill the system, I/Os that appear to be complete might not be written to disk yet. By performing an orderly shutdown of z/OS, then z/VM, then zPDT, and then Linux, you decrease the risk of data loss.

ADCD provides the VTAMAPPL tool to automate the startup and shutdown of your system. The ADCD-supplied proc for this process is called `SHUTALL`. We created a slightly modified version called `SHUTSYS`. The `SHUTSYS JCL` is in `USER.PROCLIB`. It points at members `SHUTSOW1` or `SHUTSOW2` in `USER.PARMLIB`, depending on which system is stopped.

Depending on the started tasks that are running on your systems, you might want to customize these members to remove commands for started tasks that you do not use and to add commands for your started tasks that are not included in the provided members.

The example shows how you perform an orderly shutdown of *both* systems. (The numbers in parentheses at the start of certain lines refer to the device numbers that are used in Figure E-5 on page 202. Device 703 is the console for the S0W2 system, and device 700 is used as the console for the S0W1 system.)

```

log off all TSO sessions
(703)  S SHUTSYS                (start shutdown script for S0W2)
      (wait for the ALL AVAILABLE FUNCTIONS COMPLETE message)
(703)  $PJES2
      (wait for the JES2 ENDED message)
(700)  S SHUTSYS                (start shutdown script for S0W1)
      (wait for the ALL AVAILABLE FUNCTIONS COMPLETE message)
(700)  $PJES2
      (wait for the JES2 ENDED message)
(700)  V XCF,S0W2,OFFLINE
nn IXC371D CONFIRM REQUEST TO VARY SYSTEM S0W2 OFFLINE, REPLY
SYSNAME=S0W2 TO REMOVE S0W2 OR C TO CANCEL
(700)  nn,SYSNAME=S0W2
nn IXC102A XCF IS WAITING FOR SYSTEM S0W2 DEACTIVATION. REPLY DOWN
WHEN MVS ON S0W2 HAS BEEN SYSTEM RESET
      (If you simply wait for about 15 seconds, z/VM will end S0W2 so you do
      not need to reply DOWN)
      (wait for console activity to stop, usually after a CONSOLE
      PARTITION CLEANUP COMPLETE message)
(703)  LOGOFF
      (to log off the S0W2 virtual machine)

(700)  V XCF,S0W1,OFFLINE      (allows PDSE and so on to stop in a
      clean way)
(700)  nn,SYSNAME=S0W1
(700)  LOGOFF
      (to log off the S0W1 virtual machine)

(CFCONSOL) FORCE CFCC1
(CFCONSOL) FORCE CFCC2
(CFCONSOL SHUTDOWN                (shut down z/VM)
(Linux 1) $ awsstop

```

The **V XCF** command and the response to the WTOR can be issued on either system, but you must name the system that is being stopped. It is important to complete this step. If you do not complete this step, the other members of the sysplex might think that you are loading another system with a duplicate name (because you never removed it cleanly from the sysplex) the next time that you attempt to load the system.

## Useful commands

A few commands can be entered directly on a CFCC console. Because we use a common z/VM interface to the CFCC virtual machines, we must use the z/VM **SEND** command. The following examples of direct CFCC commands, which are entered on the CFCONSOL terminal window, can be used:

```

SEND CFCC1 DISPLAY MODE
SEND CFCC2 DISPLAY RESOURCES
SEND CFCC1 DISPLAY CHPIDS
SEND CFCC2 HELP

```

Several z/OS commands directly relate to CF usage. In principle, the following commands are issued through the MVS operator console. In practice, several of the commands produce too much output to be useful on the MVS console panel and they are best used through the System Display and Search Facility (SDSF) LOG interface:

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| <b>D CF</b>                      | Best use SDSF interface                     |
| <b>D XCF</b>                     | Brief XCF status                            |
| <b>D XCF,CF</b>                  | More detailed information                   |
| <b>D XCF,COUPLE</b>              | Lots of output; use SDSF interface          |
| <b>D XCF,POLICY</b>              |                                             |
| <b>D XCF,STR</b>                 | List defined structures                     |
| <b>D XCF,STR,STRNAME=ISGLOCK</b> | Details about specific structure; use SDSF  |
| <b>D LOGREC</b>                  | LOGREC status                               |
| <b>D LOGGER,L</b>                | LOGGER status                               |
| <b>V OPERLOG,HARDCPY</b>         | If not already using LOGGER for some reason |
| <b>SETLOGRC LOGSTREAM</b>        | Change LOGREC recording                     |
| <b>SET SMF=xx</b>                | Switch to new SMFPRMxx member               |

The following z/OS commands do not relate to CF usage directly, but they are helpful to determine the state of the system:

|                     |                            |
|---------------------|----------------------------|
| <b>D C,HC</b>       | Hardcopy log status        |
| <b>D ETR</b>        | External timer mode status |
| <b>D C</b>          | MVS console status         |
| <b>D IPLINFO</b>    |                            |
| <b>D M=CPU</b>      |                            |
| <b>D IOS,CONFIG</b> |                            |
| <b>D M=STOR</b>     |                            |
| <b>\$D MASDEF</b>   |                            |
| <b>\$D MEMBER</b>   |                            |

The following Linux commands might be useful for the base sysplex setup:

|                            |                              |
|----------------------------|------------------------------|
| <b>\$ cat /etc/exports</b> | Used on an NFS server system |
| <b># showmounts -e</b>     | Used on an NFS client system |

## Defining new CDSs

Situations might occur where you want to move to new CDSs. For example, the supplied CDSs are formatted for a sysplex that contains up to four systems. If you want more than four systems in your sysplex, you must create a set of CDSs that are formatted with an appropriate MAXSYSTEM value.

The following basic mechanisms are available for moving to a new set of CDSs:

- ▶ Define the new CDSs (by using the DEFCDSS job in SYSPLEX.PARALLEL.CNTL.Z0S22 as a model). Use the **SETXCF ACOUPLE** and **SETXCF PSWITCH** commands to copy the contents of the CDSs to the new CDSs.
- ▶ Shut down the sysplex, delete the data sets, define new (empty) data sets (by using the DEFCDSS job in SYSPLEX.PARALLEL.CNTL.Z0S22 as a model), and bring the sysplex back up again.

**Note:** Most CDSs contain information that is constantly updated by the systems in the sysplex. In addition, certain CDSs contain information (policies) that you create and place in the correct CDS, for example, ARM, SMF, or CFRM. If you delete and redefine the primary and alternative CDSs, all of that information is lost.



The first mechanism is almost always used in a production environment or *any* environment in which a sysplex-wide outage is unacceptable. It also has the advantage that all of the information (policies and status information) in the CDSs is carried forward to the new CDSs.

If you decide to use the first mechanism, the process includes the following steps:

1. Create a copy of the DEFCDSS job that allocates the data sets that you require, with the changes that you want to make (for example, a larger MAXSYSTEM value).
2. Run a **SETXCF ACOUPLE** command to replace the alternative data set with the data set that becomes the new primary.
3. Run a **SETXCF PSWITCH** command to replace the primary data set with your new primary data set.
4. Run another **SETXCF ACOUPLE** command to add the new alternative CDS.
5. Update the COUPLEPS member of USER.PARMLIB to specify the new CDS names. This step ensures that the sysplex uses the correct data sets the next time that you load a system or the entire sysplex.

At the end of this process, you are running on the new CDSs and all of the status and policy information from the corresponding old CDSs is carried forward.

However, if your objective is to discard all of the contents of the CDSs and start with a clean CDS, the process is more complex. It is especially important to understand that if you use the second mechanism to replace the Logger CDSs, all of the information in the existing log streams is inaccessible.

The process that is used to replace the CDSs by using the second mechanism includes the following steps:

1. Shut down the Parallel Sysplex and z/VM.<sup>7</sup>
2. Load the basic ADCD system (not under z/VM) with an IPL parameter, such as 0A82CS.
3. Use ISPF option **3.4** to delete all of the Parallel Sysplex CDSs on volume CF0001.<sup>8</sup> Those data sets have names that begin with PARALLEL.ADCDPL. (Do *not* delete the CDSs that are used by the basic ADCD system. These data sets have names that begin with SYS1.ADCDPL, and they are on other volumes.)

You might also want to delete the logger offload data sets, as described in “Logger data sets” on page 209. Failing to delete these sets can result in confusing messages in the future because Logger attempts to allocate new staging and offload data sets with names that are duplicates of the existing data sets.

4. Change whatever parameters you want in the coupling definitions in the DEFCDSS, and POLaaaa jobs in PARALLEL.SYSPLEX.CNTL.Z0S22.
5. Run job DEFCDSS to create the CDSs and verify that the job ends with return code of zero.
6. Update the COUPLEPS member of parmlib to specify the new CDS names, if necessary.
7. Run the POLaaaa jobs to install your policies in the new CDSs.

<sup>7</sup> Also, this step can be performed by using the BASEAD guest under z/VM. When this approach is used, you must stop the two coupling facility virtual machines after you stop and log off the S0W1 and S0W2 virtual machines.

<sup>8</sup> Our installation example in this chapter uses this volser. Other volumes can be used.

**Important:** The default action when you run the IXCMIAPU utility to create a policy is to install that policy in the corresponding in-use CDS. However, you are running with the ADCD-provided CDSs in this case, but you want to install your new policies into the CDSs that you just created. Therefore, it is *vital* to ensure that the POLaaaa jobs include the name of the target CDS on the DEFINE POLICY statement, as shown in Example E-8.

This method can be used only with the ARM, CFRM, and SFM policies. It is not possible to update the LOGR policy in any data other than the active LOGR CDS.

*Example E-8 Sample IXCMIAPU statements*

---

```
//POLARM1 JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID.
//ARMPOL EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
DATA TYPE(ARM) REPORT(YES)
DEFINE POLICY NAME(ARM1) DSN(SYSPLEX.ADCDPL.NEW.ARM.CDS01) REPLACE(YES)
```

---

8. If you created CFRM CDSs, update the COUPLEPS member to add the CFRMPOL(aaaaaa) statement. This step is necessary if you want to start in GRS Star mode and if the CFRM CDSs that are used for the IPL were not used previously.
9. Complete any other tasks that are easier in an unshared system.
10. Shut down the basic ADCD system.
11. Start z/VM and the two coupling facility virtual machines, and load the S0W1 system (with IPL parameter 0A82PS).

Several unexpected error messages might appear in the log. These messages occur because the various log streams are not yet defined.

12. Run the following jobs from the SYSPLEX.PARALLEL.CNTL.Z0S22 data set:

- LOGRCICS for the CICS model log streams
- LOGREREP for the LOGREC log stream
- LOGRHC for the z/OS Health Checker log stream
- LOGROPR for the OPERLOG log stream
- LOGRRRS for the RRS log streams
- LOGRSMF for the SMF log streams

Use the z/VM command **DIAL S0W1** to access a VTAM session.

13. Edit the COUPLEPS parmlib member again and remove the CFRMPOL statement.

The extra CF structures are now available, but your S0W1 system is not using the structures that were added by the LOGRaaaa jobs. You can use the structures immediately by running the following MVS commands:

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <b>V OPERLOG,HARDCPY</b>           | (starts OPERLOG operation)           |
| <b>SETLOGRC LOGSTREAM</b>          | (moves LOGREC to the log stream)     |
| <b>SETSMF RECORDING(LOGSTREAM)</b> | (changes SMF to use log stream mode) |
| <b>S RRS,SUB=MSTR</b>              | (restarts RRS to use CF structures)  |
| <b>D XCF,STR</b>                   | (displays the allocated structures)  |

The next time that you load (with IPL parameter PS), these log streams are used automatically.

## Logger data sets

Be aware that the CDSs might contain data that is needed across IPLs. The most obvious case involves **LOGGER** staging and offload data sets. In normal use, **LOGGER** automatically deletes its offloaded data sets when the retention period for all of the records in the data set expires. If you delete and re-create the couple data sets (CDSs), this retention period information for **LOGGER** data sets is lost and the data sets remain on your disks until they are explicitly deleted. These sets are **VSAM** data sets, and they are slightly more difficult to delete than simple sequential or partitioned data sets.

The **LOGGER** data sets normally have the high-level qualifier (HLQ) **IXGLOGR** (but they have a HLQ of **LOGGER** or **START1** for the CICS log streams in our Parallel Sysplex system) and full names, as shown in the following example:

```
IXGLOGR.IFASMF.GENERAL.A0000000          <==base cluster name
IXGLOGR.IFASMF.GENERAL.A0000000.DATA
IXGLOGR.SYSPLEX.LOGREC.ALLRECS.A0000000   <==base cluster name
IXGLOGR.SYSPLEX.LOGREC.ALLRECS.A0000000.D
      or
LOGGER.IFASMF.GENERAL.A0000000            <==base cluster name
LOGGER.IFASMF.GENERAL.A0000000.DATA
LOGGER.SYSPLEX.LOGREC.ALLRECS.A0000000    <==base cluster name
LOGGER.SYSPLEX.LOGREC.ALLRECS.A0000000.D
```

The A0000000 qualifiers in these examples are for the first **LOGGER** data set in each group. This number is incremented when the **LOGGER** data sets are created.

## Adding 3390 volumes in Parallel Sysplex

You might want to add emulated 3390 volumes to one or both z/OS systems. The z/VM directory for the z/OS guests that is described in this chapter has directory definitions for z/OS 3390 volumes at addresses A80 - ABF, which might be sufficient for your extra volumes.

Complete the following steps:

1. Create (or restore) the volumes. Creating a volume involves the use of the zPDT **a1cckd** command as described in “Download and create volumes” on page 156.
2. Add the new volume to the devmap. The devmap file that is used in our examples is called **devmapp**. Use an address in the A80 - ABF range, if possible. If this range cannot be used, complete the following steps to modify the z/VM directory:
  - a. Start zPDT (run the **awsstart devmapp** command) and IPL z/VM (run the **ip1 200 parm 0700** command).
  - b. Log on as user **maint** (password is **SSI1**<sup>9</sup>).
  - c. Edit the VM directory (run the **xedit user direct c** command).
  - d. Scroll to the section that begins with user ID **MVSDUMMY**.
  - e. For more information about the devices that are defined in the ADCD-provided IODF, see “Devices that are defined in the ADCD-provided IODF” on page 122. If possible, use an address that is defined to avoid needing to update the input/output definition file (IODF).
  - f. By using the patterns for user IDs **MVSDUMMY**, **BASEAD**, **S0W1**, and **S0W2**, add lines for your new volumes, if needed. Add these lines to all four user IDs.

<sup>9</sup> This password is the password for the z/VM 6.3 ADCD system. Other z/VM implementations feature their own passwords.

- g. Exit from Xedit (run the **file** command).
  - h. Rebuild the active z/VM directory (run the **directxa user direct c** command).
3. Start one of the z/OS systems. If you created 3390 volumes (instead of restoring the volumes from a DVD or a download), you must submit an ICKDSF job from TSO to initialize the volumes. (z/OS automatically varies offline any uninitialized volumes that it finds during IPL. After you initialize the volumes with ICKDSF, you can vary them online to z/OS.)

## Adding 3270 terminals in Parallel Sysplex

You can add more 3270 sessions for z/VM or for each z/OS. Consider the following points:

- ▶ z/VM must know about the extra local 3270 devices. This process is completed by adding 3270 definitions in the devmap. z/VM recognizes 3270 terminals, regardless of the addresses that are used. That is, it is not necessary to use addresses in the 7xx range (although our z/VM customization requires a local 3270 session at address 700 for the initial operator session). Remember that a maximum of 32 terminals can be defined for the aws3274 device manager. You might need to run a z/VM **enable all** command to obtain a logon logo on the new terminals.
- ▶ After you define the extra z/VM 3270 sessions, you can use these sessions to DIAL to a z/VM guest. For example, DIAL S0W1 to connect to our first z/OS system.
- ▶ The z/VM user ID definition for a guest (S0W1, S0W2, and BASEAD) must have sufficient SPECIAL statements for all of the 3270 sessions (except the session at address 700 that is used for the MVS console). You can add SPECIAL statements for these user IDs by editing the z/VM directory, as described in “Adding 3390 volumes in Parallel Sysplex” on page 209. These SPECIAL connections are hardware connections between a 3270 session and the guest z/OS virtual machine.
- ▶ z/OS must recognize the 3270 session. The current ADCD systems recognize local 3270 connections at addresses 701 - 73F. However, the A0600 member in VTAMLST allows only 10 TSO sessions through local 3270 connections. You must modify VTAMLST to increase this number. This task is a normal z/OS systems programming activity.
- ▶ You can connect 3270 sessions to z/OS TCP/IP. The current ADCD systems allow up to 30 of these connections (through definitions in the TCPIP PROFILE and in VTAMLST). These sessions<sup>10</sup> (directly to z/OS TCP/IP) do not require any modifications to z/VM or the devmap.

## Scaling up in Parallel Sysplex

Many of the limitations of the starter system that we described are important to understand. It is a relatively small configuration and various steps are required to expand it to a larger, more robust configuration. In particular, consider the following points:

- ▶ Our two z/OS systems are defined (in the z/VM directory) with a modest amount of memory for each system. Depending on the products that are used in the system and the volume of work that is processed, the system might require more memory.
- ▶ Our sample devmap defines an 8 or 10 GB zPDT system. This devmap must be increased to accommodate the larger z/OS systems.

<sup>10</sup> A limit of 30 exists because of TN3270 and VTAM definitions. These limits can be increased, if needed.

- ▶ Larger z/OS systems often require larger paging data sets. If large virtual memory applications (such as DB2 applications) are used and if IBM SAN Volume Controller (SVC) memory dumps might be expected, considerably larger paging areas are required for each z/OS. The MVS command **D ASM** is useful for determining the amount of paging space that is used. Under SDSF, the DA function can be used to informally sample dynamic paging rates.
- ▶ Larger z/VM and z/OS memory will have correspondingly larger PC server memory.
- ▶ Our z/VM system has one 3390-3 volume for paging. More z/VM paging volumes might be needed. Monitor for HCPPGT401I and HCPPGT400I messages on the VM OPERATOR console. These messages can precede a VM wait state. Therefore, if you encounter these messages, add more paging volumes to z/VM before you add more work to z/OS.
- ▶ Our CFs (in the z/VM directory) are defined with 1200 MB each. Depending on the number and size of your coupling structures, these CFs might need to be larger. Run a **D CF** command on z/OS to determine how much free space is in each CF. We strongly suggest that you aim for each CF to be less than 50% full to allow for moving all of the structures into one CF if you must restart the other CF.
- ▶ Our z/OS systems have limited STORAGE disk space (as defined in the VATLST00 members in PARMLIB). You might need to add disk volumes that are defined as STORAGE volumes.
- ▶ SVC memory dumps are directed to A2SYS1 and A2SYS2. You might want to direct them to larger volumes that you add.
- ▶ You need to monitor paging and swap space in your base Linux. The **xosview** program is convenient for this process if you installed it with your Linux.

Paging (by the base Linux, z/VM, or z/OS) often is a poor use of the limited disk I/O resources in a zPDT system. Larger PC memory, which is combined with a small amount of monitoring to balance memory usage best, can be the most effective way of improving the performance of the zPDT systems. Despite these efforts, you are likely to see base Linux swap activity. This activity appears to be mostly because of a strong Linux preference to use real memory for the disk cache. An effective disk cache is important for zPDT performance.





# **Transmission Control Protocol considerations**

This appendix contains supplementary information for readers that do not have extensive experience with Transmission Control Protocol (TCP) networking.

## TCP definitions

For more information about the search order for TCP definitions, see “Understanding search orders of configuration information” in *z/OS Communications Server IP Configuration Guide*, SC27-3650.

For more information about the files and data sets that contain TCP definition information, see “Configuring TCP.DATA” in *z/OS Communications Server IP Configuration Guide*, SC27-3650.





## Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

### Locating the Web material

The Web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server at this website:

<ftp://www.redbooks.ibm.com/redbooks/SG248315>

Alternatively, you can go to the following IBM Redbooks website:

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

Select **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG248315.

### Using the Web material

The additional Web material that accompanies this book includes the following files:

| <i>File name</i> | <i>Description</i>         |
|------------------|----------------------------|
| cf0001.gz        | Gzip copy of CF0001 volume |
| cics01.gz        | Gzip copy of CICS01 volume |
| cics02.gz        | Gzip copy of CICS02 volume |
| db2001.gz        | Gzip copy of DB2001 volume |
| db2002.gz        | Gzip copy of DB2002 volume |

For ease of operation, we recommend that you download these .gzip files directly into the zPDT directory that holds all of your other zPDT z/OS volumes. Then, run a gunzip against each one, as described in 4.3.3, “Download and create volumes” on page 42.

## **System requirements for downloading the Web material**

The Web material requires the a system configuration of at least 90 MB of Hard Disk Drive space.

# Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this book.

## IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Some publications that are referenced in this list might be available in softcopy only:

- ▶ *Merging Systems into a Sysplex*, SG24-6818
- ▶ *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205
- ▶ *S/390 Parallel Sysplex: Resource Sharing*, SG24-5666
- ▶ *Systems Programmer's Guide to: z/OS System Logger*, SG24-6898
- ▶ *Systems Programmer's Guide to Resource Recovery Services (RRS)*, SG24-6980
- ▶ *System z Parallel Sysplex Best Practices*, SG24-7817
- ▶ *SMF Logstream Mode: Optimizing the New Paradigm*, SG24-7919

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft, and additional materials, at the following website:

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

## Other publications

The following publications are also relevant as further information sources:

- ▶ *z/OS MVS Initialization and Tuning Reference*, SA23-1380
- ▶ *z/OS MVS System Commands*, SA38-0666
- ▶ *z/OS Security Server RACF System Programmers Guide*, SA23-2287
- ▶ *z/OS MVS Programming: Resource Recovery*, SA23-1395
- ▶ *z/OS Communications Server IP Configuration Guide*, SC27-3650
- ▶ *z/OS JES2 Initialization and Tuning Guide*, SA32-0991
- ▶ *IBM Health Checker for z/OS User's Guide*, SC23-6843

## Online resources

The ADCD Home Page also is relevant as a further information source. The page is available at this website:

<http://dtsc.dfw.ibm.com/adcd/adcd.html>

## Help from IBM

IBM Support and downloads:

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

IBM Global Services:

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

# Index

## Symbols

/etc/HOSTNAME 27

## Numerics

1090 token 20

1091tokens 20

3270 sessions, more 93, 210

3390 volumes, additional 92, 209

## A

ACDS5 job 60, 175

alcckd command 92, 209

awsstart devmap1 146

## B

Base AD-CD system under z/VM 92, 209

## C

CFCC1 146

CFCC1 DISPLAY MODE 89, 205

CFCC2 146

CFCC2 DISPLAY RESOURCES 89, 205

CFCONSOL 21, 146

CFCONSOL guest 21

COUPLE2 job 91, 207

cpuopt zVM\_CouplingFacility 21

## D

D XCF 89, 206

DASDOPT WRKALLEG 22

devmap statements 21

directxa 93, 210

## H

hardware required 22

Health Checker log stream 72, 74, 188, 190

HFS changes 53, 167

HFS4 job 53, 167

## I

Implementation checklist

for parallel sysplex 75, 191

IPL parameters 92, 209

ISMF panels 54, 169

## J

JES2 parms 60, 175

## L

LOGGER datasets 92, 209

LOGREC log stream 72, 74, 188, 190

LOGREC, start in LOGGER 92, 208

## M

memory size, PC 22

## N

Network Addressed Storage (NAS) 142

## O

OPERLOG log stream 72, 74, 189–190

OPERLOG operation, start 92, 208

OPTION CFUSER TODENABLE 22

OPTION CFVM TODENABLE 21

## P

paging data sets 94, 211

PC memory, partition 23

performance notes 23

## R

RACF

carrying changes forward to new database 11

RACFUNLD 11

Redbooks website 217

Contact us xii

RRS log streams 73–74, 189–190

## S

Shutting down the system 74, 190

SMF log stream 73–74, 189–190

SMF recording mode 74, 190

SMS ACS routines source 57, 171

SMS data classes 55, 169

SMS storage classes 56, 171

SMS storage groups 56, 170

STORAGE disk space 94, 211

SYS1.SCDS 54, 169

## T

TCPIP procedure 70, 185

## U

user direct c 93, 209

## V

V XCF,BDCD,OFFLINE 88, 205

VATLST00 member 94, 211  
VTAMAPPL started tasks 69, 185

## **W**

WRKALLEG option 48, 163

## **X**

x3270 146  
xosview program 94, 211

## **Z**

z/VM directory entry 22  
z/VM overhead 24  
z/VM paging volumes 94, 211











SG24-8315-01

ISBN 0738441767

Printed in U.S.A.

Get connected

